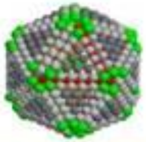
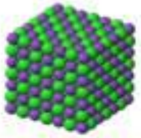




Introduction to classical

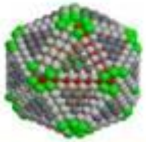
***M**olecular **D**ynamics Simulations*

Dr. Ali Kerrache

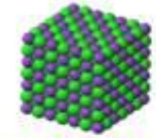
Faculty of Science, Univ. of Manitoba, Winnipeg
WestGrid / Compute Canada

***E*-mail:** ali.kerrache@umanitoba.ca
***Home Page*:** https://ali-kerrache.000webhostapp.com/

WESTGRID | UNIVERSITY OF MANITOBA | compute canada | calcul canada



Introduction to MD simulations



Who am I?

- **High Performance Computing Specialist**
  - WestGrid and Compute Canada.
  - Software and User Support.
  - National teams:
    - ✓ **BST:** Bio-molecular Simulation Team.
    - ✓ **RSNT:** Research Support National Team.

WESTGRID

compute canada | calcul canada

- **Computational Physicist**
  - Monte Carlo and Molecular Dynamics codes.
  - Study of the properties of materials using MD simulation.
  - Metals, Glasses: Silica, Amorphous silicon, Nuclear Glasses.
  - Mass transport, solid-liquid interfaces, kinetic coefficients, melting, crystallization, mechanical deformations, static and dynamical properties, He diffusion in glasses, …

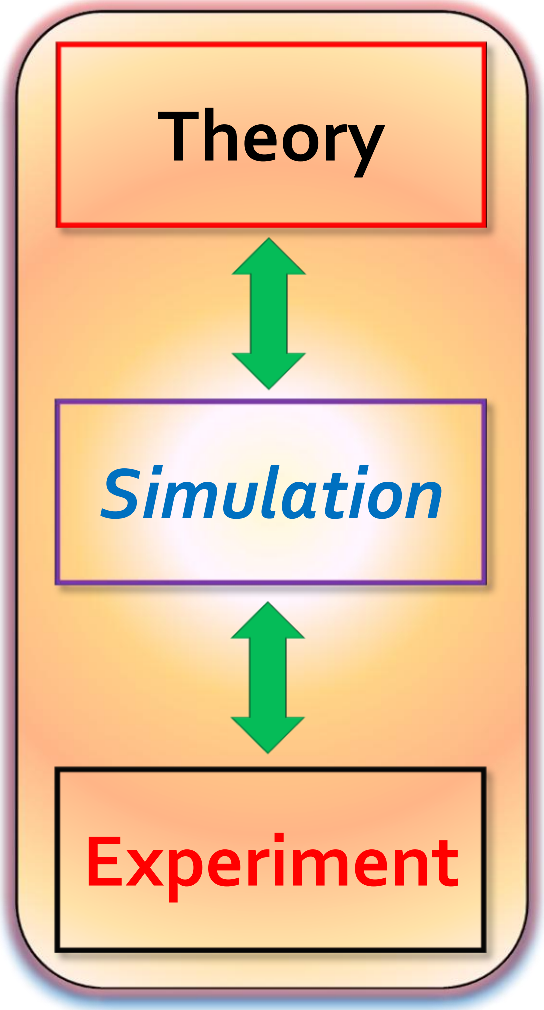
Introduction to MD simulations

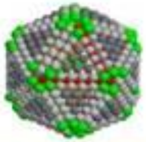
Outline:

- Introduction
  - Basic concepts of Molecular Dynamics Simulations.
  - Examples of Simulations using Molecular Dynamics.
- Setting and Running MD simulations (LAMMPS)
  - LAMMPS: Molecular Dynamics Simulator.
  - Building LAMMPS step by step.
  - Running LAMMPS (Input, Output, …).
- Readings and References

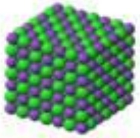
Why do we need simulations?

- **Except simple cases,** no analytical solutions for most of the problems.
- **In most cases, experiments are:**
  - Difficult or impossible to perform.
  - Too dangerous to …
  - Expensive and time consuming.
  - Blind and too many parameters to control.
- **Simulation is a powerful tool:**
  - can replace experiments.
  - provoke experiments.
  - explain and understand experiments.
  - complete the theory and experiments.





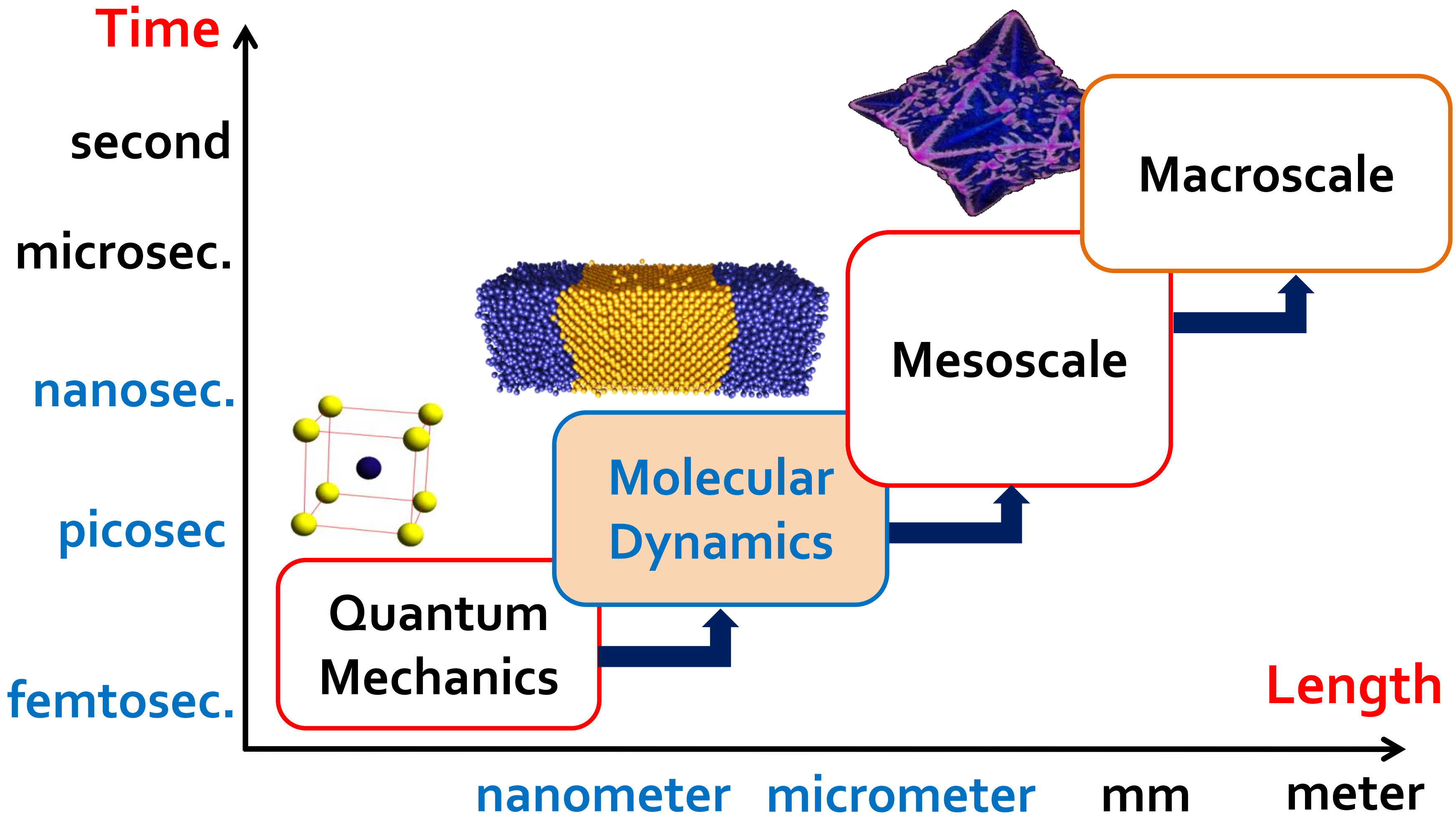
Atomistic / Molecular Simulations



- **What are the atomistic/molecular Simulation?**
  - a tool to get insights about the properties of materials at atomic or molecular level.
  - used to predict and / or verify experiments.
  - considered as a bridge between theory and experiment.
  - provide a numerical solution when analytical ones are impossible.
  - used to resolve the behavior of nature (the physical world surrounding us) on different time- and length-scales.

- **Applications,** simulations can be applied in, but not limited to:
  - ✓ Physics, Applied Physics, Chemistry, …
  - ✓ Materials and Engineering, …

Length and Time Scales



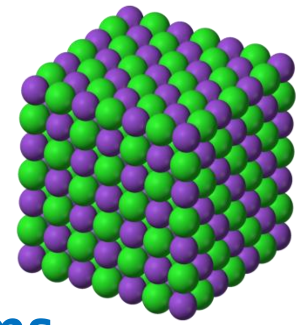
Classical MD Simulation

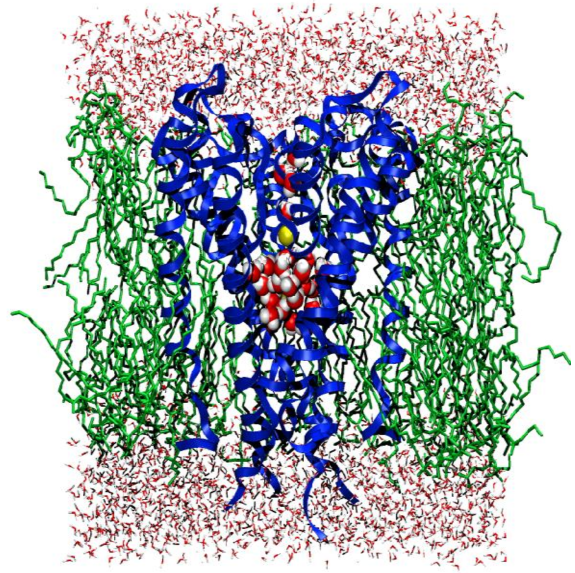
❑ Solution of Newton equations:

$$m_i \vec{a}_i = \vec{F}_i$$

$$\vec{F}_i = \sum_{j \neq i}^{N} \vec{f}_{ij}$$

- MD is the solution of the classical equations of motion for a system of N atoms or molecules in order to obtain the time evolution of the system.
- Uses algorithms to integrate the equations of motion.
- Applied to many-particle systems.
- Requires the definition of force field or potential to compute the forces.





$$\vec{f}_{ij} = -\vec{\nabla}_i V(r_{ij})$$

Structure of MD program

1. **Initialization** — $\mathbf{r}_i(t_0)$ $\mathbf{v}_i(t_0)$
2. **Compute the new forces** — $\mathbf{F}_i(\mathbf{r}_i)$
3. **Solve the equation of motion** — $\mathbf{r}_i(t) \rightarrow \mathbf{r}_i(t+\Delta t)$, $\mathbf{v}_i(t) \rightarrow \mathbf{v}_i(t+\Delta t)$
4. **Sample**
5. **Test and increase time** — $t = t + \Delta t$
6. **End of the simulation** — $t > t_{\max}$

Repeat as necessary (from "Test and increase time" back to "Compute the new forces")

Forces: Newton's Equation

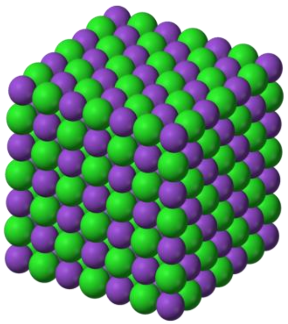
- **Potential function:**

$$U(\mathrm{r}) = U_{bond}(\ldots) + U_{non-bond}(\ldots) + U_{ext}(\ldots)$$

- **Evaluate the forces acting on each particle:**

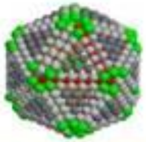
- **The force on an atom is determined by:** $\mathrm{F}_i = -\nabla U(\mathrm{r})$

  - $U(\mathrm{r})$: potential function
  - N : number of atoms in the system
  - r_{ij} : vector distance between atoms *i* and *j*

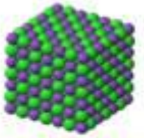


- **Newton equation:**

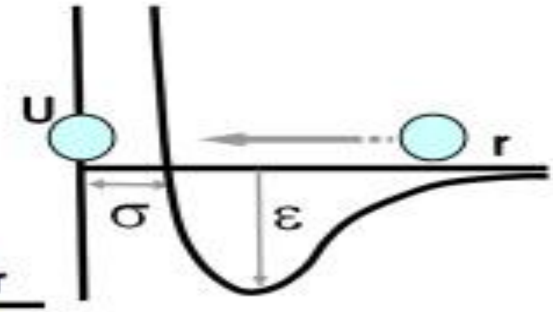
$$m_i \frac{d^2}{dt^2}\vec{x}_i = \vec{F}_i(\vec{x}_1, \ldots, \vec{x}_N) \qquad i = 1 \ldots N$$



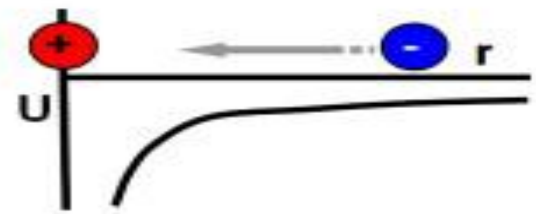
Force Fileds used in MD Simulations



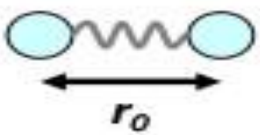
$$U = \sum_{i<j} \sum 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]$$



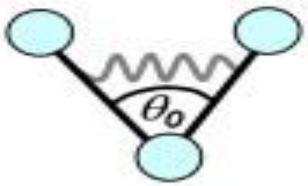
$$+ \sum_{i<j} \sum \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}$$



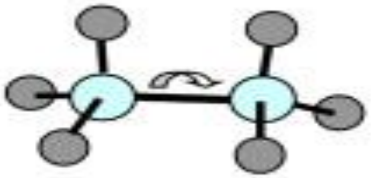
$$+ \sum_{bonds} \frac{1}{2} k_b (r - r_0)^2$$



$$+ \sum_{angles} \frac{1}{2} k_a (\theta - \theta_0)^2$$



$$+ \sum_{torsions} k_\phi [1 + \cos(n\phi - \delta)]$$



Interactions:

- Lenard-Jones
- Electrostatic
- Bonds
- Orientation
- Rotational

Derivation of Verlet Algorithm

Taylor's expansions: (position) (acceleration)

$$r(t+\Delta t) = r(t) + \dot{r}(t)\Delta t + \tfrac{1}{2}\ddot{r}(t)\Delta t^2 + \tfrac{1}{6}\dddot{r}(t)\Delta t^3 + O(\Delta t^4) \qquad (I)$$

$$r(t-\Delta t) = r(t) - \dot{r}(t)\Delta t + \tfrac{1}{2}\ddot{r}(t)\Delta t^2 - \tfrac{1}{6}\dddot{r}(t)\Delta t^3 + O(\Delta t^4) \qquad (II)$$

(velocity)

Add (I) and (II):

$$r(t+\Delta t) + r(t-\Delta t) = 2r(t) + \ddot{r}(t)\Delta t^2 + O(\Delta t^4)$$

or:

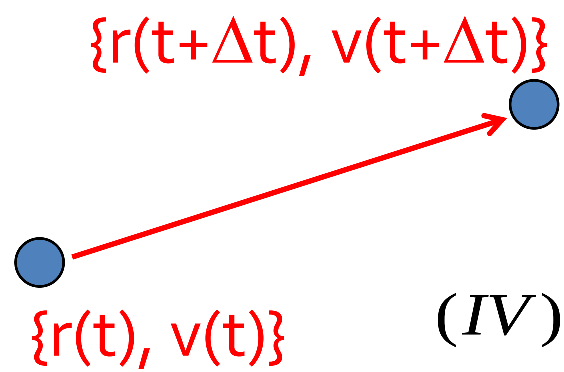
$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + f(t)\Delta t^2/m + O(\Delta t^4) \qquad (II)$$

Subtract (II) from (I):

$$r(t+\Delta t) - r(t-\Delta t) = 2\dot{r}(t)\Delta t + O(\Delta t^3)$$

or:

$$v(t) = \big(r(t+\Delta t) - r(t-\Delta t)\big)/2\Delta t + O(\Delta t^2) \qquad (IV)$$



Verlet and Leap-Frog Algorithms

- From the initial positions and velocities: $\mathbf{r}_i(t)$ $\mathbf{v}_i(t)$

$$\mathbf{a}(\mathbf{r}) = \frac{1}{m}\mathbf{F}(\mathbf{r}(t))$$

- Obtain the positions and velocities at: $t + \Delta t$
  - Velocity calculated explicitly
  - Possible to control the temperature
  - Stable in long simulation
  - Most used algorithm

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(\mathbf{r})\Delta t^2$$

$$\mathbf{a}(t+\Delta t) = \frac{1}{m}F(\mathbf{r}(t+\Delta t))$$

$$\mathbf{v}(t+\Delta t/2) = \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(\mathbf{r})\Delta t$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\Delta t/2) + \frac{1}{2}\mathbf{a}(t+\Delta t)\Delta t$$

- Leap-Frog algorithm

$$\boldsymbol{v}(t+\frac{\Delta t}{2}) = \boldsymbol{v}(t-\frac{\Delta t}{2}) + \frac{\boldsymbol{F}(t)}{m}\Delta t$$

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \boldsymbol{v}(t+\frac{\Delta t}{2})\Delta t$$

Predictor Corrector Algorithm

- **Predictor step:**
  - from the initial $\mathbf{r}_i(t)$, $\mathbf{v}_i(t)$ ➔ $\mathbf{a}(\mathbf{r}) = \frac{1}{m}\mathbf{F}(\mathbf{r}(t))$
  - predict $\mathbf{r}_i(t+\Delta t)$, $\mathbf{v}_i(t+\Delta t)$ using Taylor's series

$$\mathbf{r}^P(t+\Delta t) \cong \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{\mathbf{a}(t)}{2}\Delta t^2$$

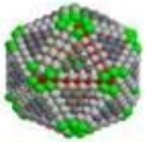
$$\mathbf{v}^P(t+\Delta t) \cong \mathbf{v}(t) + \mathbf{a}(t)\Delta t$$

$$\mathbf{a}^P(t+\Delta t) \cong \mathbf{a}(t) + \mathbf{r}^{iii}(t)\Delta t$$

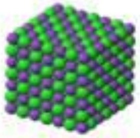
$\mathbf{r}^{iii}$: 3rd order derivatives

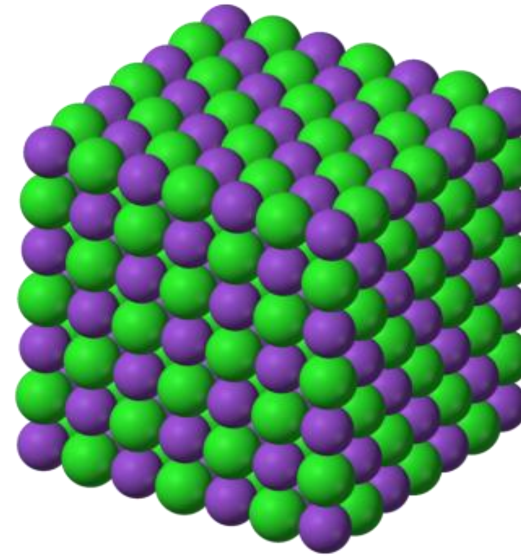
- **Corrector step:**
  - get corrected acceleration: $\mathbf{a}^C(\mathbf{r}) = \frac{\mathbf{F}(\mathbf{r}^P(t+\Delta t))}{m}$
  - using error in acceleration: $\Delta\mathbf{a}(t+\Delta t) \cong \mathbf{a}^C(t+\Delta t) - \mathbf{a}^P(t+\Delta t)$
  - correct the positions: $\mathbf{r}(t+\Delta t) \cong \mathbf{r}^P(t+\Delta t) + C_0\frac{\Delta t^2}{2}\Delta\mathbf{a}(t+\Delta t)$
  - correct the velocities: $\mathbf{v}(t+\Delta t) \cong \mathbf{v}^P(t+\Delta t) + C_1\Delta t\Delta\mathbf{a}(t+\Delta t)$

C_n: constants depending accuracy



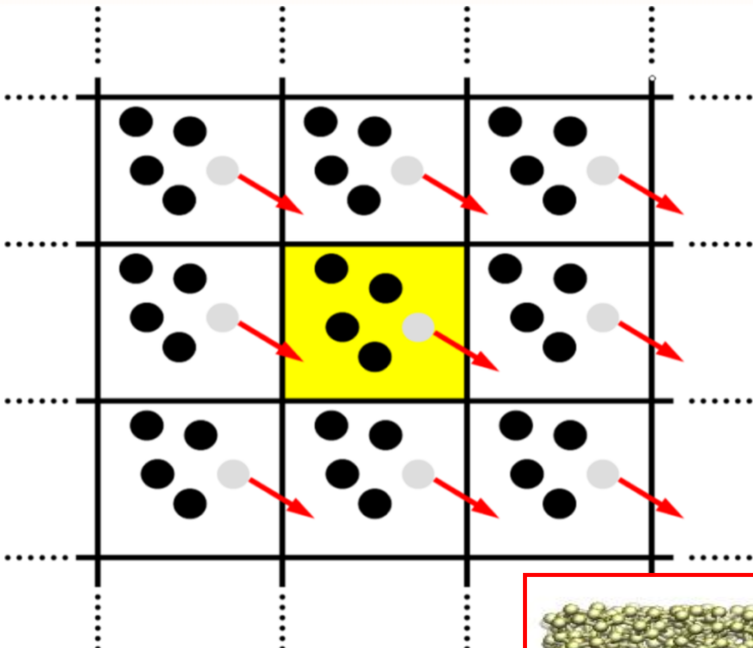
MD Simulation settings





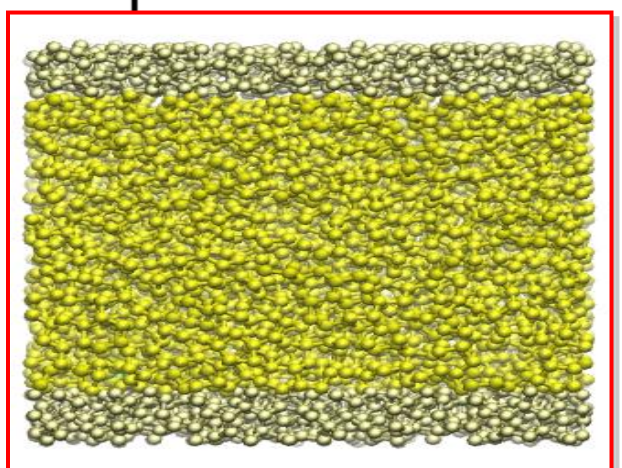
- ❑ Starting configuration:
  - ➢ Atomic positions (x,y,z)
  - ➢ density …
  - ➢ mass, charge, ….
- ❑ Initial velocities: depend on temperature
- ❑ periodic boundary conditions (PBC):
  - ➢ required to simulate bulk properties.
- ❑ set the appropriate potential:
  - ➢ Depend on the system to simulate (literature search).
- ❑ set the appropriate time step: should be short (order of 1fs).
- ❑ set the temperature control:
  - ➢ define the thermodynamic ensemble (NVT, NPT, NVE, …).

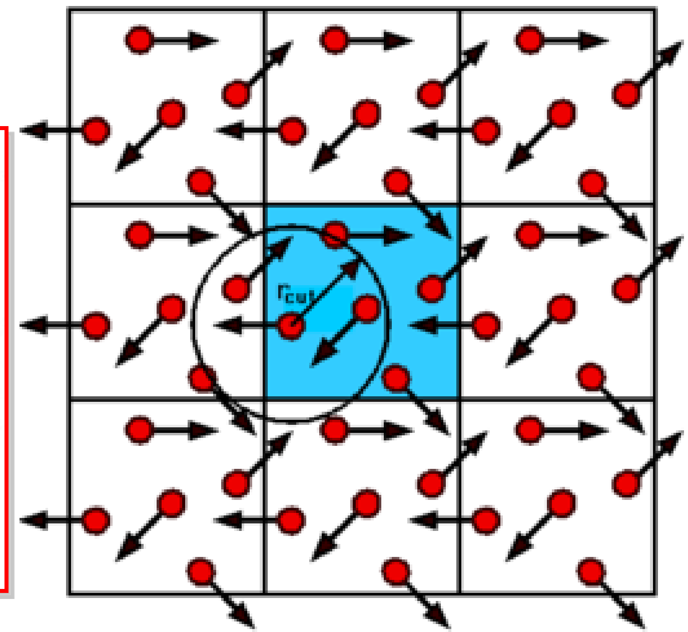
Periodic Boundary Conditions



- Create images of the simulation box: duplication in all directions (x, y and z)
- An atom moving out of boundary comes from the other side.

- **PBC:**
  in x, y directions
- **Walls:**
  fixed boundaries in z direction.





Neighbour Lists

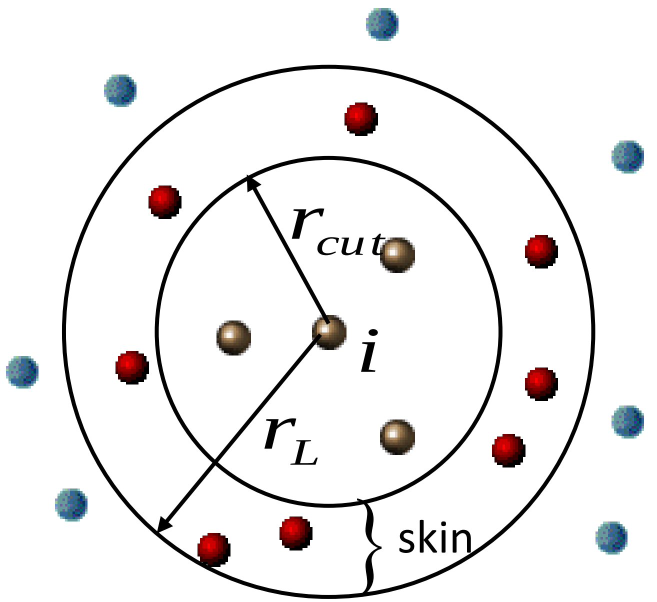
- Evaluate forces is time consuming:
- Pair potential calculation: $\propto O(\mathrm{N}^2)$
- Atom moves $< 0.2\,\mathring{\mathrm{A}}$ per time step
- Not necessary to include all the possible pairs.

Solution: Verlet neighbor list

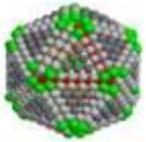
- Containing all neighbors of each atom within: r_L
- Update every N_L steps

For each particle: N-1

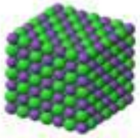
For N particles: N(N-1)



$$r_L - r_{cut} > \frac{N_L \bar{\mathbf{v}} \Delta t}{2}$$



Thermodynamic Ensembles



- **Ensembles:**
  - **NVE – micro-canonical ensemble**
  - **NVT – canonical ensemble**
  - **NPT – grand-canonical ensemble**

Each ensemble is used for a specific simulation:
- *Equilibration …*
- *Production run …*
- *Diffusion (NVE), …*

- **Temperature control:**
  - **Berendsen thermostat (velocity rescaling)**
  - **Andersen thermostat**
  - **Nose-Hoover chain**

- **Pressure control:**
  - **Berendsen volume rescaling**
  - **Andersen piston**

✓ *Choose the ensemble that best fits your system and the properties you want to simulate*
✓ *start the simulation.*
✓ *Check the thermodynamic properties as a function of time.*

Statistical Mechanichs

Goal of MD simulations:

- The prime purpose of MD is to sample the *phase space* of the statistical mechanics ensemble.
- Most physical properties can be related the atomic trajectories and obtained as average as a function of time.

Structural properties:

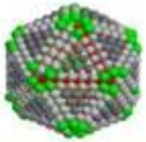
- obtained from spatial correlation functions e.g. radial distribution function (RDF, S(Q), Van-Hove, ...).

Dynamical Properties:

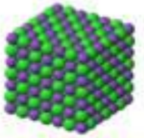
- Time dependent properties (MSD, diffusion coefficients) obtained via temporal correlation functions e.g. velocity autocorrelation function.

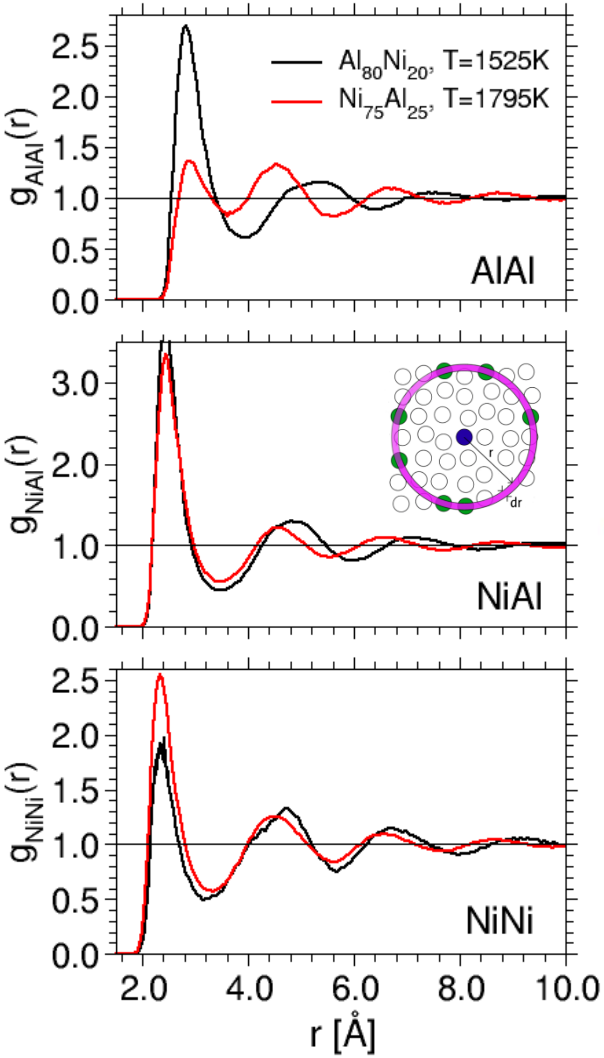
Thermodynamic Properties

- **Kinetic Energy** → $\langle K.E. \rangle = \left\langle \frac{1}{2} \sum_{i}^{N} m_i v_i^2 \right\rangle$
- **Temperature** → $T = \frac{2}{3Nk_B} \langle K.E \rangle$
- **Configuration Energy** → $U_c = \left\langle \sum_{i} \sum_{j>i}^{N} V(r_{ij}) \right\rangle$
- **Pressure** → $PV = Nk_B T - \frac{1}{3} \left\langle \sum_{i=1}^{N-1} \sum_{j>i}^{N} \vec{r}_{ij} \cdot \vec{f}_{ij} \right\rangle$
- **Specific Heat** → $\left\langle \delta (U_c)^2 \right\rangle_{NVE} = \frac{3}{2} N k_B^2 T^2 (1 - \frac{3Nk_B}{2C_v})$



Structural properties: AlNi

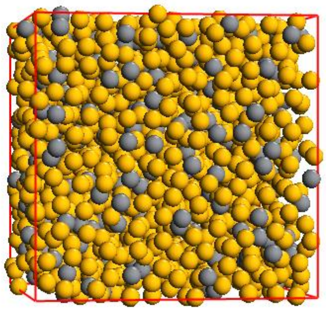


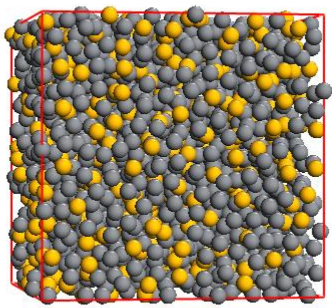


❖ **Radial Distribution Function (simulation)**

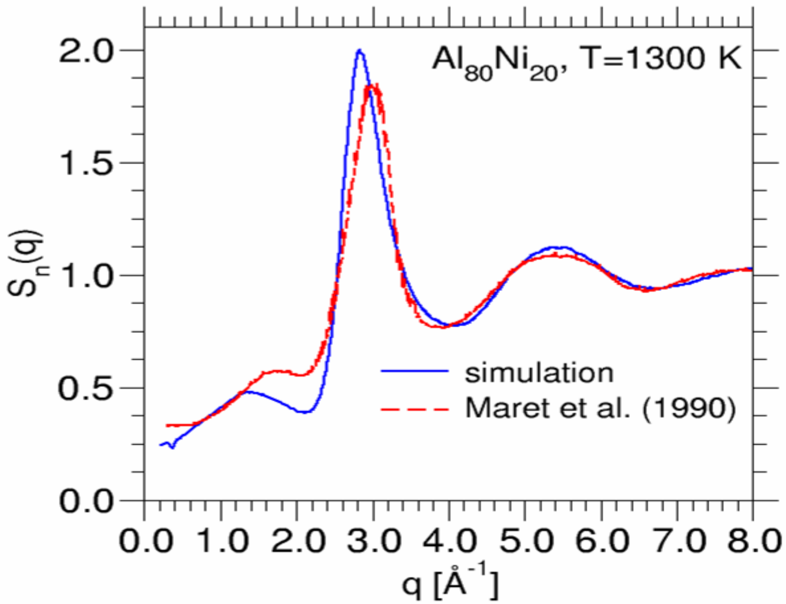
$Al_{25}Ni_{75}$

$$g(r) = \frac{\langle n(r) \rangle}{4\pi\rho r^2 \Delta r} = \frac{V}{N^2}\left\langle \sum_{i}\sum_{j\neq i}^{N} \delta(r - r_{ij}) \right\rangle$$



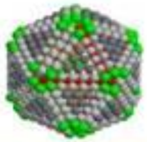


$Al_{80}Ni_{20}$

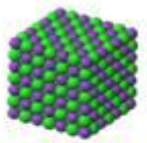


$$S(k) = 1 + 4\pi\rho\int_0^\infty \frac{\sin(kr)}{kr}(g(r) - 1)\, r^2 dr$$

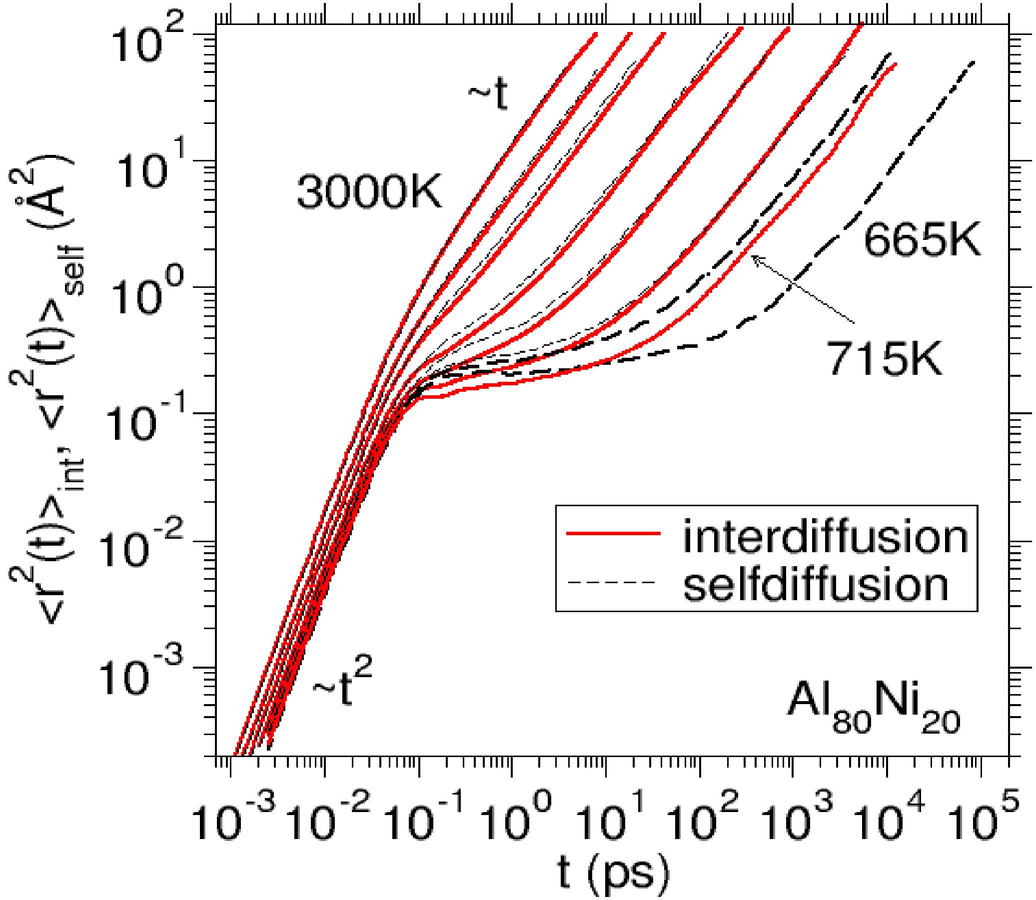
➢ **Structure Factor (experiments)**



Dynamical Properties: AlNi



Mean Square Displacement (Einstein relation)



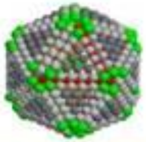
$$2Dt = \frac{1}{3}\left\langle |\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2 \right\rangle$$

$$\mathrm{MSD} = c_{\mathrm{Al}}\left\langle (\vec{r}_{\mathrm{s,Ni}}(t) - \vec{r}_{\mathrm{s,Ni}}(0))^2 \right\rangle + c_{\mathrm{Ni}}\left\langle (\vec{r}_{\mathrm{s,Al}}(t) - \vec{r}_{\mathrm{s,Al}}(0))^2 \right\rangle$$

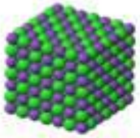
Diffusion constants

$$D = \lim_{t\to\infty} \frac{1}{N}\sum_{i=1}^{N} \frac{\left\langle (r(t) - r(0))^2 \right\rangle}{6t}$$

WESTGRID   UNIVERSITY OF MANITOBA   compute canada | calcul canada



Avilable MD Programs

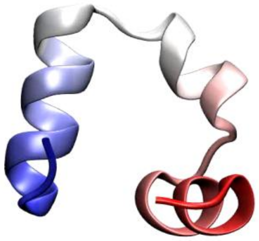


- **Open source: free access**
  - ✓ **LAMMPS:** http://lammps.sandia.gov/index.html
  - ✓ **DL_POLY:** http://www.scd.stfc.ac.uk/SCD/44516.aspx
  - ✓ **CP2K:** https://www.cp2k.org/about
  - ✓ **NAMD:** http://www.ks.uiuc.edu/Research/namd/
  - ✓ **GROMACS:** http://www.gromacs.org/
  - ✓ ….

- **Commercial software:**
  - ✓ **Amber:** http://ambermd.org/

- **Home made codes:**
  - ✓ C, C++
  - ✓ Fortran, … etc

- **Visualization:**
  - ➢ VMD
  - ➢ OVITO, …
- **Analysis?**



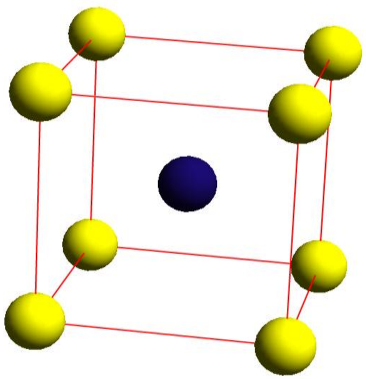
Molecular Dynamics: some Results

- **Binary Metallic alloys:**
  - Melting and crystallization.
  - Solid-Liquid interfaces.
  - Crystal growth from melt.
  - Crystal growth is diffusion limited process.

- **Glasses:**
  - How to prepare a glass using MD simulation?
  - Glass Indentation using MD.

Solid-Liquid interface velocities

Why B2-$Al_{50}Ni_{50}$?



- ✓ B2-$Al_{50}Ni_{50}$: prototype of binary ordered metals
- ✓ simulations of interfacial growth in binary systems rare
- ✓ growth kinetics of binary metals: diffusion limited?
- ✓ crystal growth slower than in one-component metals
- ✓ understand crystal growth of alloys on microscopic level

Questions:

- crystal growth & accurate estimation of T_m?
- solid-liquid interface velocity from interface motion?
- kinetic coefficients and their anisotropy?
- solid-liquid interface motion controlled by mass diffusion?
- solid-liquid coexistence, interface structure?
- how to distinguish between solid-like & liquid-like particles?

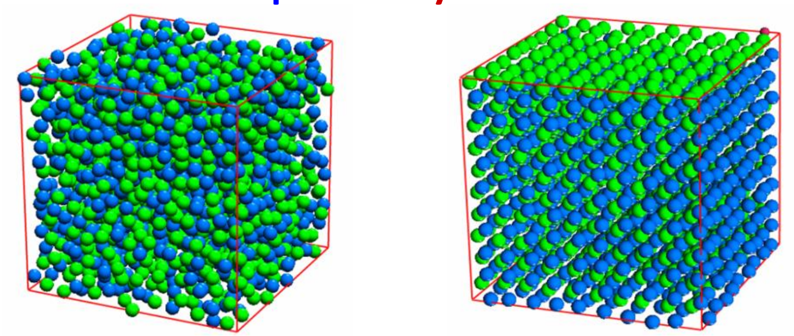
- Wilson H.A., Philos. Mag. , **50** (1900) 238.
- Frenkel J., Phys. Z. Sowjetunion, **1** (1932) 498.

Solid-Liquid Interfaces: AlNi

- solve Newton's equation of motion for system of *N* particles:
  - **velocity Verlet algorithm (time step = 1 fs)**
  - **NPT ensemble:**
    constant pressure (Anderson algorithm): p = 0
    constant temperature: stochastic heat bath
  - **periodic boundary conditions in all directions**

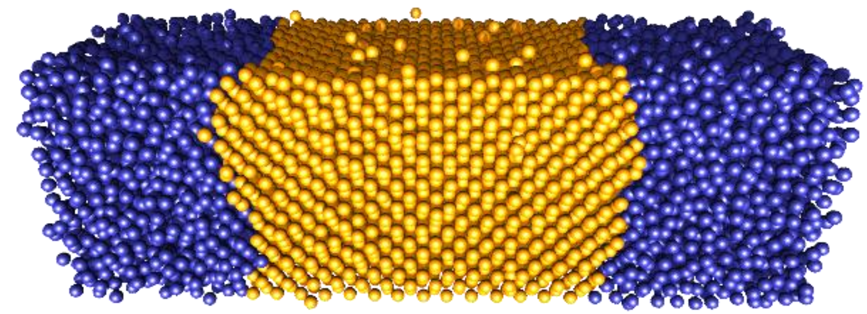
Allen M.P. and Tildeslay D.J., Computer simulation of liquids, 1987
Anderson H.C., JCP **72** (1980) 2384

MD of pure systems



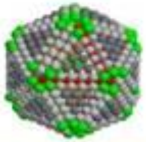
- lattice properties
- T dependence of density
- Structural quantities
- Self-diffusion constant

MD of inhomogeneous systems

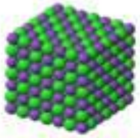


- melting temperature T_m
- kinetic coefficients & their anisotropy
- solid-melt interface structure
- crystal growth

WESTGRID University of Manitoba compute canada | calcul canada



Simulation Parameters



- **Binary metallic mixtures** - **simple**: **Lennard-Jones potential**
  - **better**: **EAM**

- **EAM** potential:

$$U_{\text{pot}} = \frac{1}{2}\sum_{k,l} u(r_{kl}) + \sum_{k} F(\bar{\rho}_k)$$

$$\bar{\rho}_k = \sum_{l \neq k} \rho_l(r_{kl})$$

- two body interactions.
- many body interactions (e-density).
- fitting to both experimental and *ab-initio* data.
- reproduces the lattice properties & point defects.
- structure and dynamics of AlNi melts.

Y. Mishin *et al.*, PRB **65**, (2002) 224114.
J. Horbach *et al.*, PRB **75**, (2007) 174304.

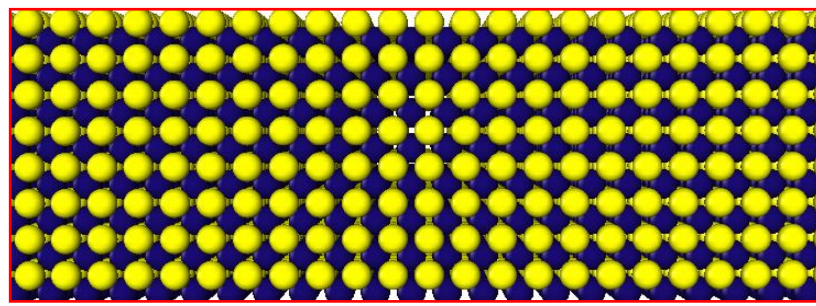
- **Solid** and **liquid** properties:

2000 particles ($L_x = L_y = L_z = 24.6$ Å)

- **Solid-liquid** interfaces (N particles):

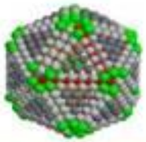
$N_{Al} = N_{ni} \Rightarrow D = L_z \simeq 3 \times L_x \simeq 3 \times L_y$

10386 and 12672 particles

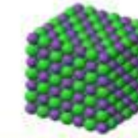


$D \simeq 3 \times L_x$

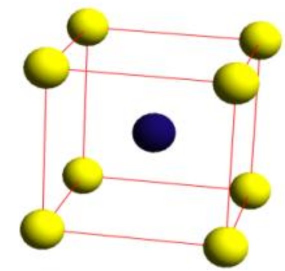
WestGrid | University of Manitoba | compute canada | calcul canada

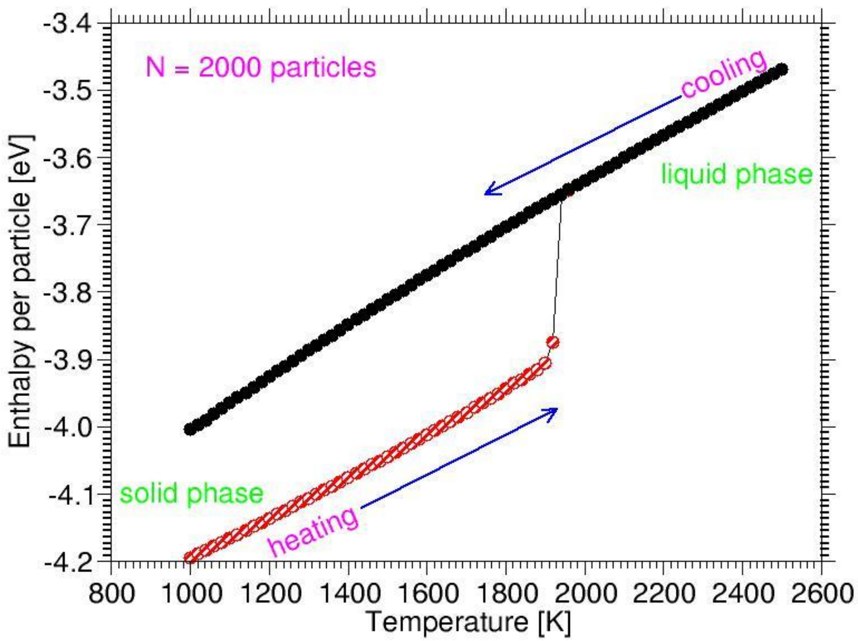


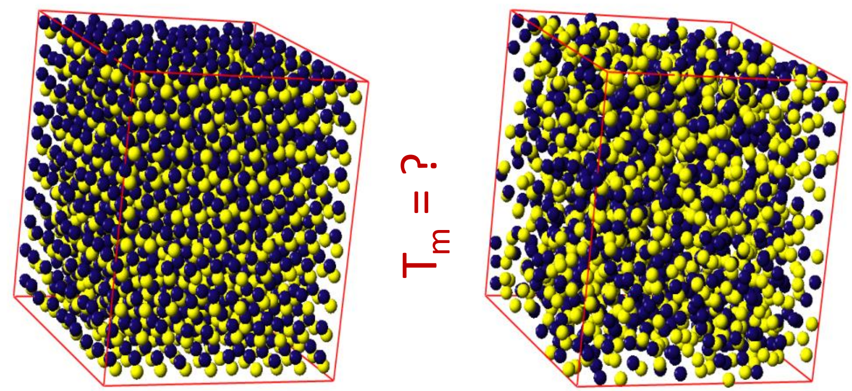
Pure Phases: crystal, liquid



- How to go from crystal to melt & from melt to crystal?
  - ✓ start from B2 phase: equilibration at 1000 K
  - ✓ try to melt the crystal: heating process
  - ✓ cool down the melt: cooling process



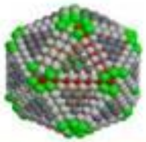




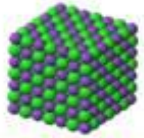
- ➢ binary alloys: glass formers.
- ➢ crystallization: process too slow
- ➢ brute force method:

not appropriate to estimate T_M

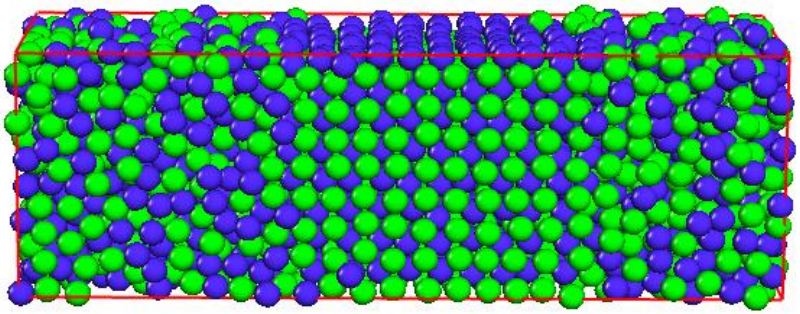
- How to study crystallization?



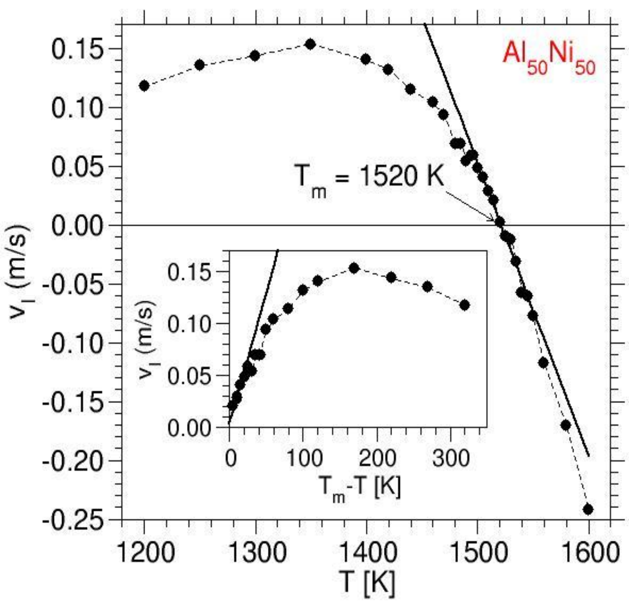
Estimation of Melting Temperature



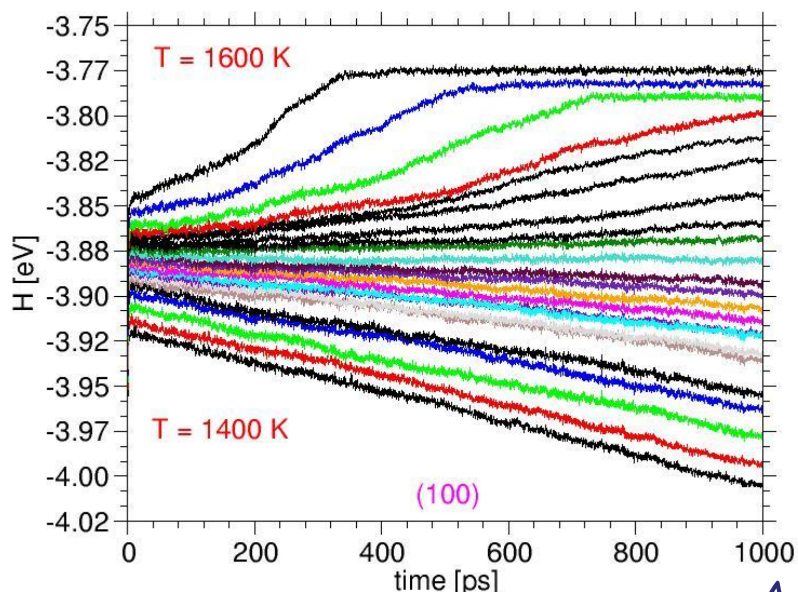
- Equilibrate a crystal (NPT, p=0)
- Fix the particles in the middle of the box
- Heat away the two other regions
- Quench at the target temperature



The Melting temperature T_M from solid-liquid interface motion:



Interface velocity



Enthalpy as a function of time

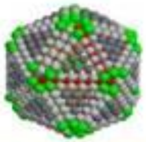
$T > T_M$: *melting*

$T = T_M$: *coexistence*

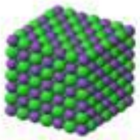
$T < T_M$: *crystallization*

A. Kerrache et al., EPL 2008.

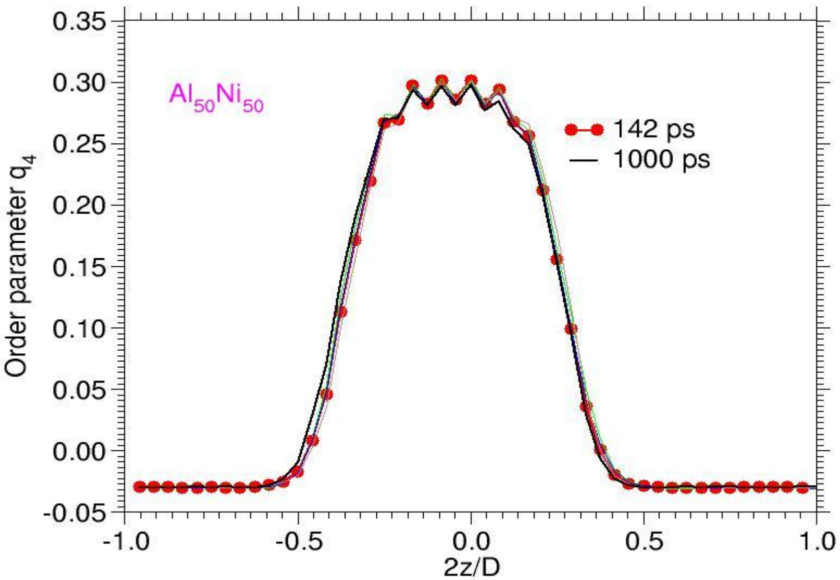
WESTGRID · UNIVERSITY OF MANITOBA · compute canada | calcul canada



Characterization of S-L Interfaces



Bond order parameter profile For different times

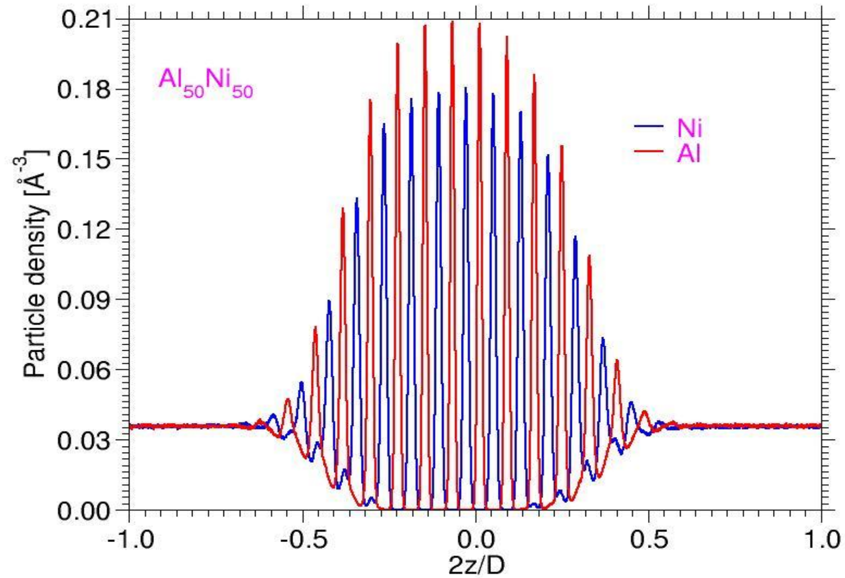


$$q_n = \left\langle \frac{1}{N} \sum_{i,j,k} \cos\left(n\theta_{xy}(i,j,k)\right) \right\rangle$$

$$n = 1,2,\ldots,6$$

I,j and k: indices for nearest neighbors, θ(I,j,k): bond angle formed by I, j and k atoms.

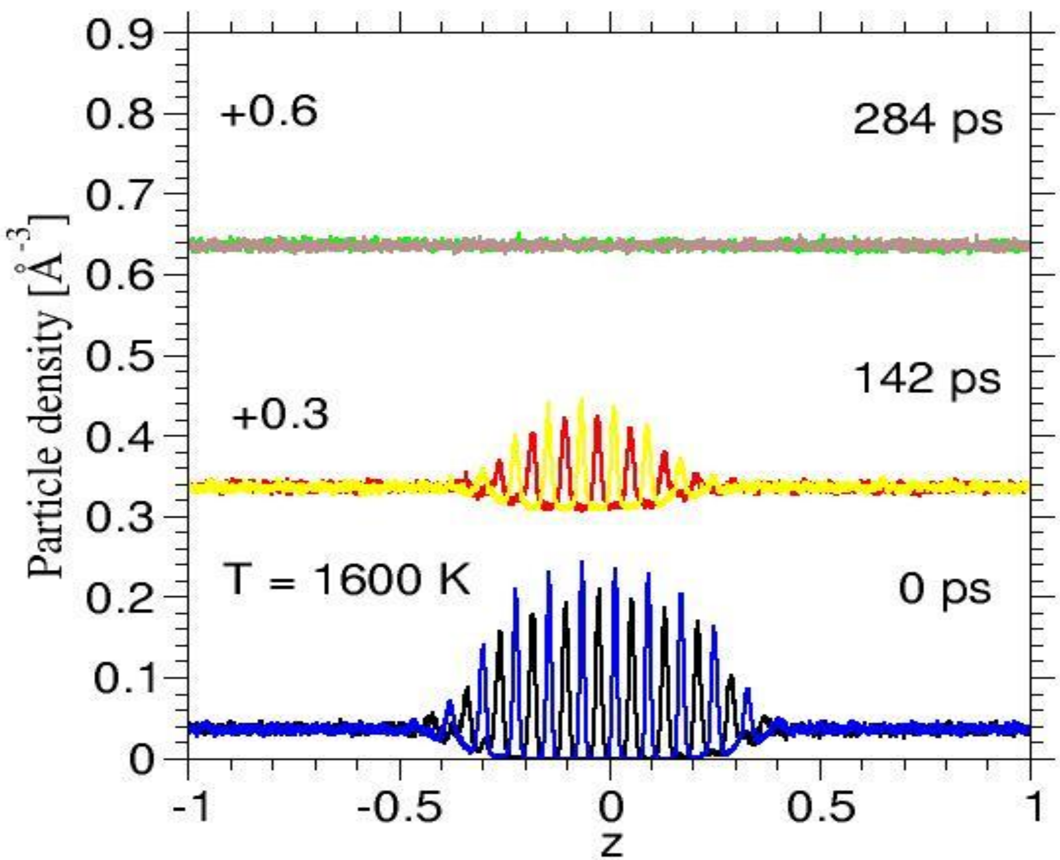
Partial particle density profile



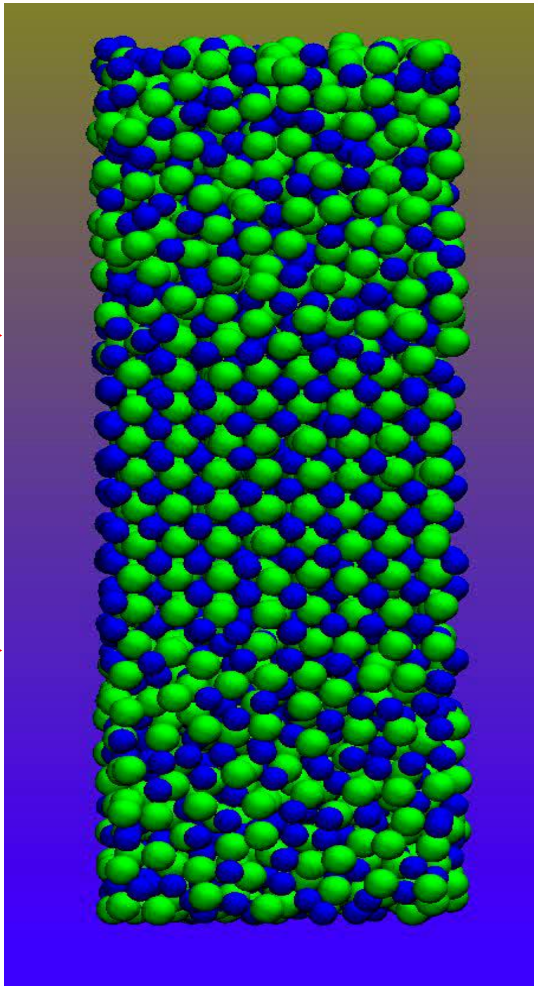
Constant density in the liquid region.
Solid-liquid interface over several layers.
Pronounced chemical ordering in the solid region: Mass transport required for crystal growth.

WESTGRID
UNIVERSITY OF MANITOBA
compute canada | calcul canada

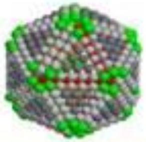
Melting of AlNi: 1600 K



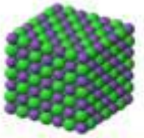
Particle density along the solid-liquid interface



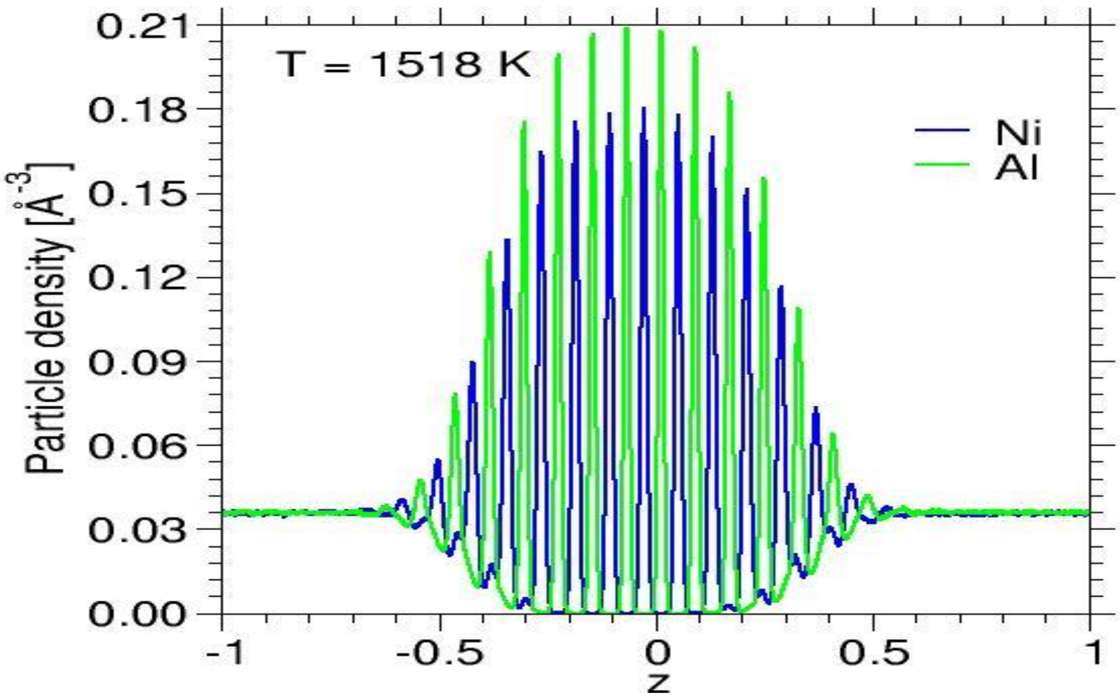
- Melting

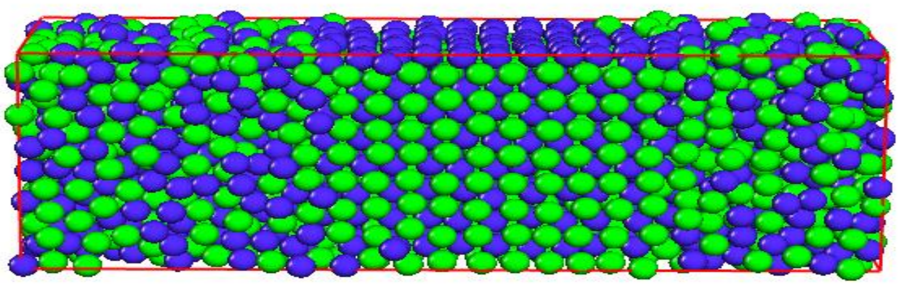


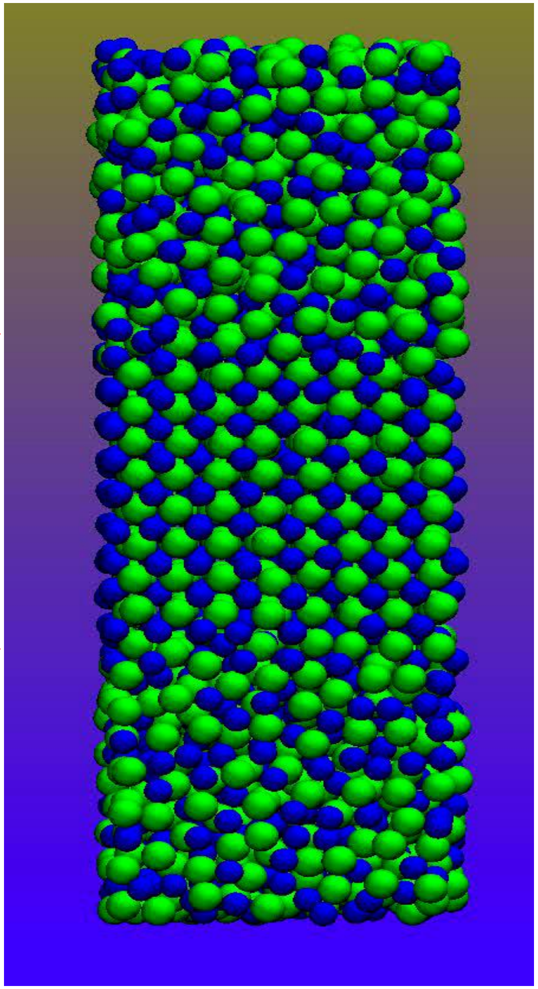
Solid-Liquid coexistence: 1518 K



Particle density along the solid-liquid interface

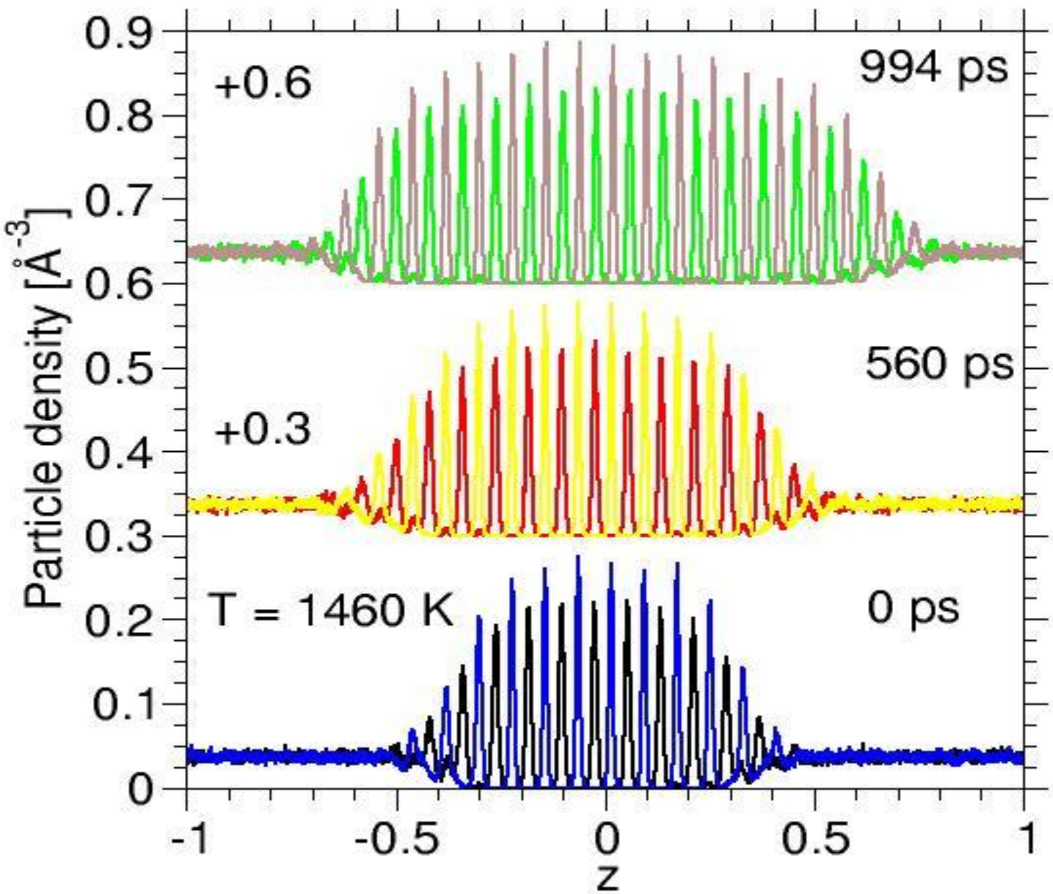




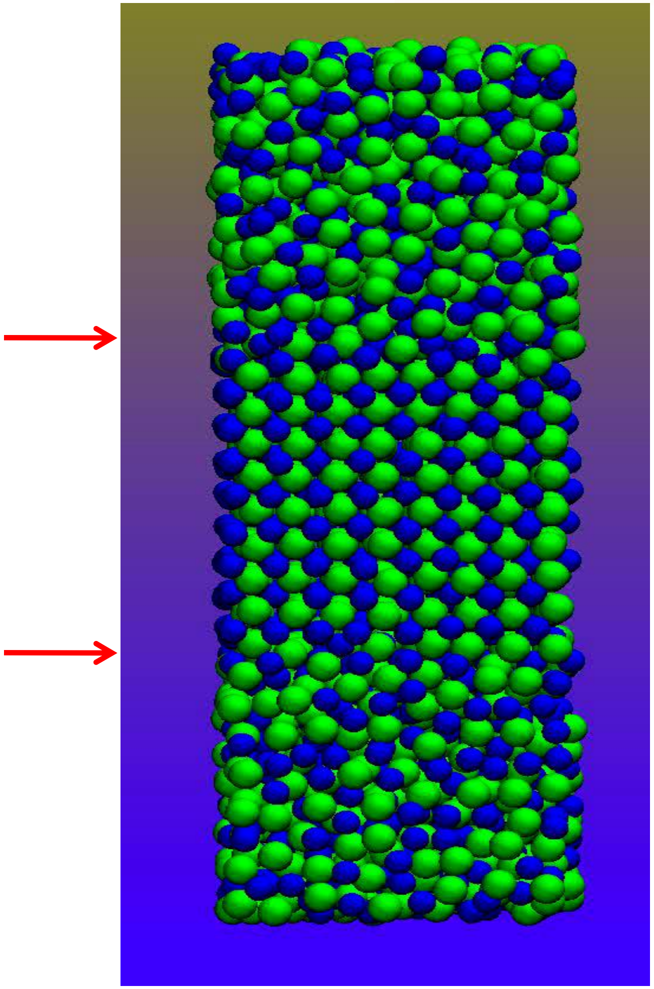


- Coexistence

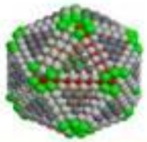
Crystallization of AlNi: 1460 K



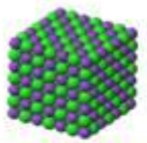
Particle density along the solid-liquid interface

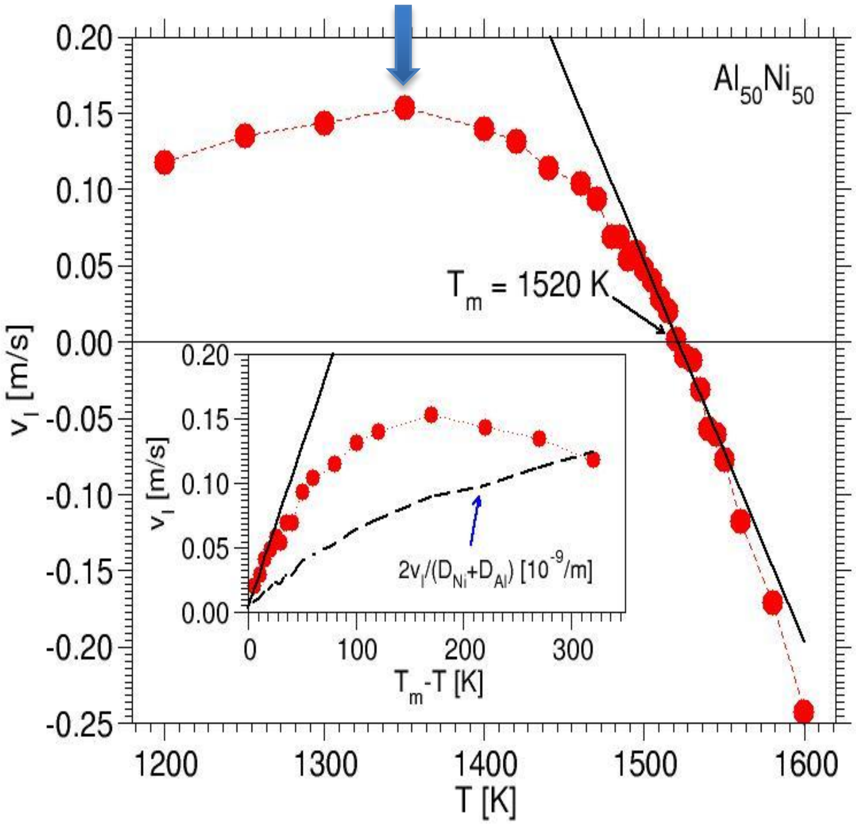


- Crystallization



Crystal Growth: Diffusion Limitted





Solid-liquid interface velocity as a function of temperature

Inset: as a function of under-cooling

❑ Why the solid-liquid interface velocity presents a maximum?

✓ Maximum of 0.15 m/s at 180 K

Interface velocity divided by the average self diffusion constant.

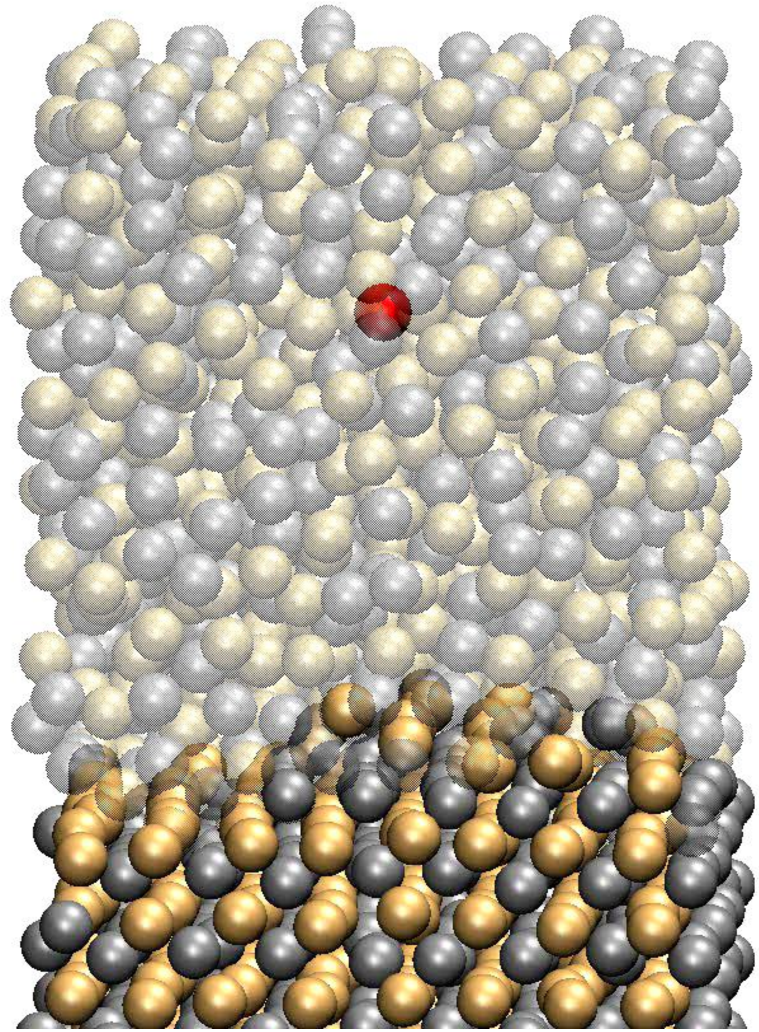
✓ Maximum due to decreasing of diffusion constant.

✓ Linear regime only up to 30 K of under-cooling.

What about the mass transport across the solid-liquid interface?

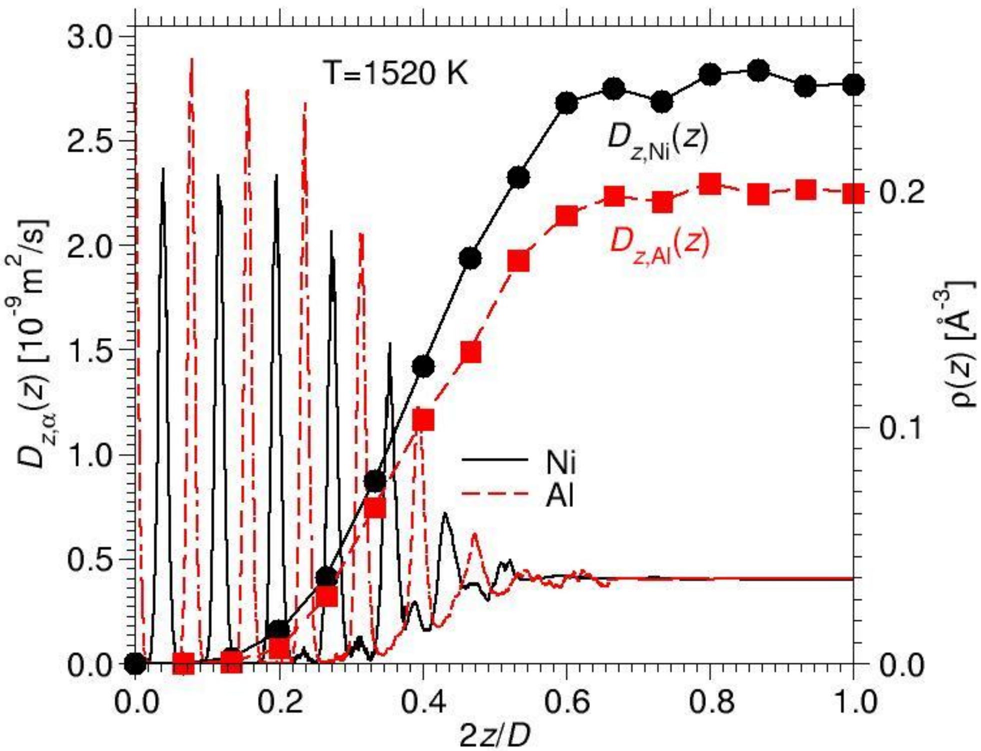
Mass Transport across the interface

- Order parameter to distinguish solid and liquid particles locally
- compute the particle density and mass density profiles
- Order parameter profile
- Number of solid-like particles
- Solid-liquid interface velocities from the number of solid-like particles
- Diffusion along the interface



WESTGRID UNIVERSITY OF MANITOBA compute canada | calcul canada

Mass Transport across the interface



Mass transport and particle density across the solid-liquid interface

$$D_{z_s,\alpha}(z_s) = \lim_{t\to\infty} \frac{1}{N_s} \sum_{i_s=1}^{N_s} \frac{\left\langle \left(z_{i_s}(t) - z_{i_s}(0)\right)^2 \right\rangle}{2t}$$

The diffusion constants decrease when we cross the solid-liquid interface.

Wilson-Frenkel theory:
activated process controlled by mass diffusion in the liquid phase

Wilson H.A. Philos. Mag. , **50** (1900) 238.
Frenkel J., Phys. Z. Sowjetunion, **1** (1932) 498.
A. Kerrache et al. EPL, 2008.

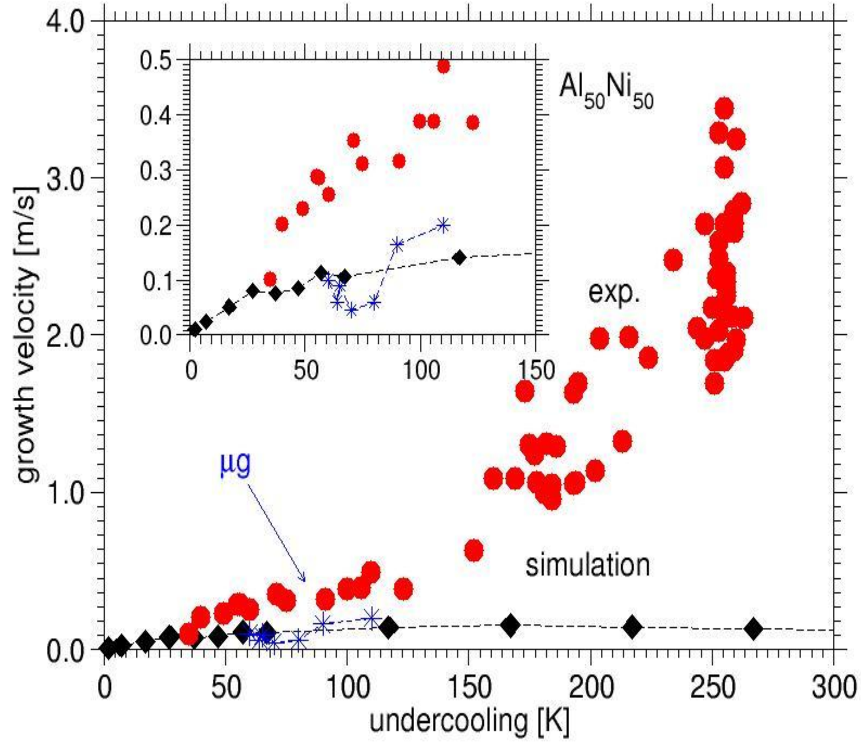
Crystal growth: controlled by mass transport in the **liquid** phase and **solid-liquid** interface

Experimental data?

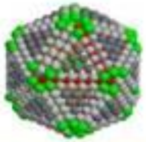
Comparison to Experimental Data

- ✓ terrestrial data (Assadi *et al.*)
- ✓ μg data (parabolic flight) , H. Hartmann (PhD thesis)

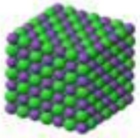
H. Assadi, *et al.*, Acta Mat. 54, 2793 (2006).



A. Kerrache *et al.*, EPL **81** (2008) 58001. good agreement with experimental data



Glasses

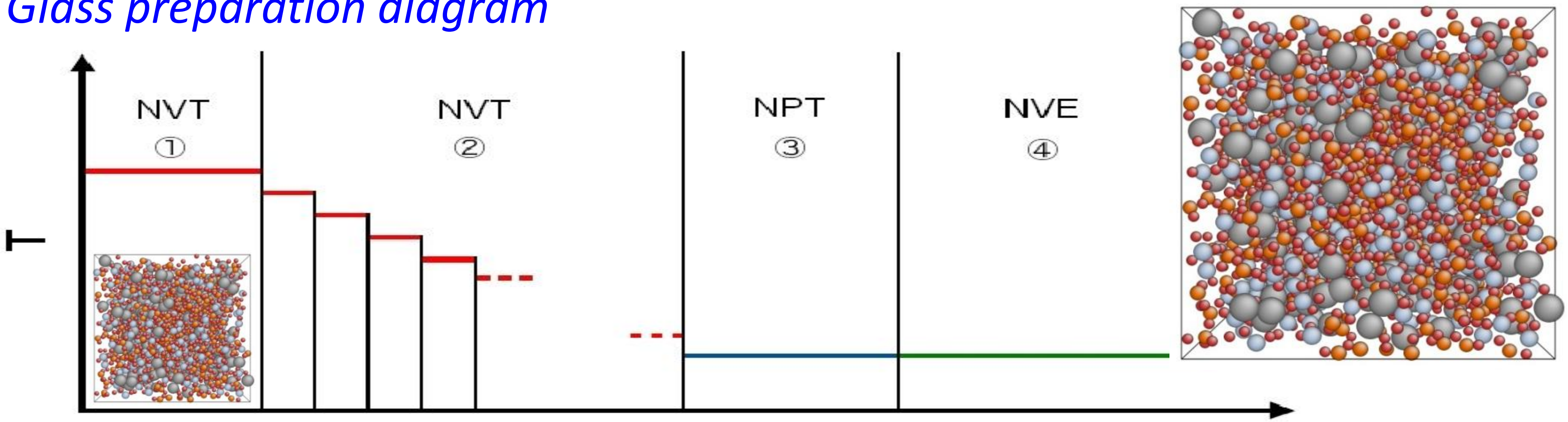


- **Binary Metallic alloys:**
  - **Melting and crystallization.**
  - **Solid-Liquid interfaces.**
  - **Crystal growth from melt.**
  - **Crystal growth is diffusion limited process.**

- **Glasses:**
  - **How to prepare a glass using MD simulation?**
  - **Glass Indentation using MD.**

How to prepare a glass?

Glass preparation diagram



Cooling rates: 10^{12} to 10^{13} K/s

Glass preparation procedure:

- ✓ Random configuration (N atoms).
- ✓ Liquid equilibration du at 5000 K (NVT).
- ✓ Cooling per steps of 100 K– (NVT).
- ✓ Glass equilibration at 300 K (NPT).
- ✓ Trajectory simulation at 300 K (NVE).

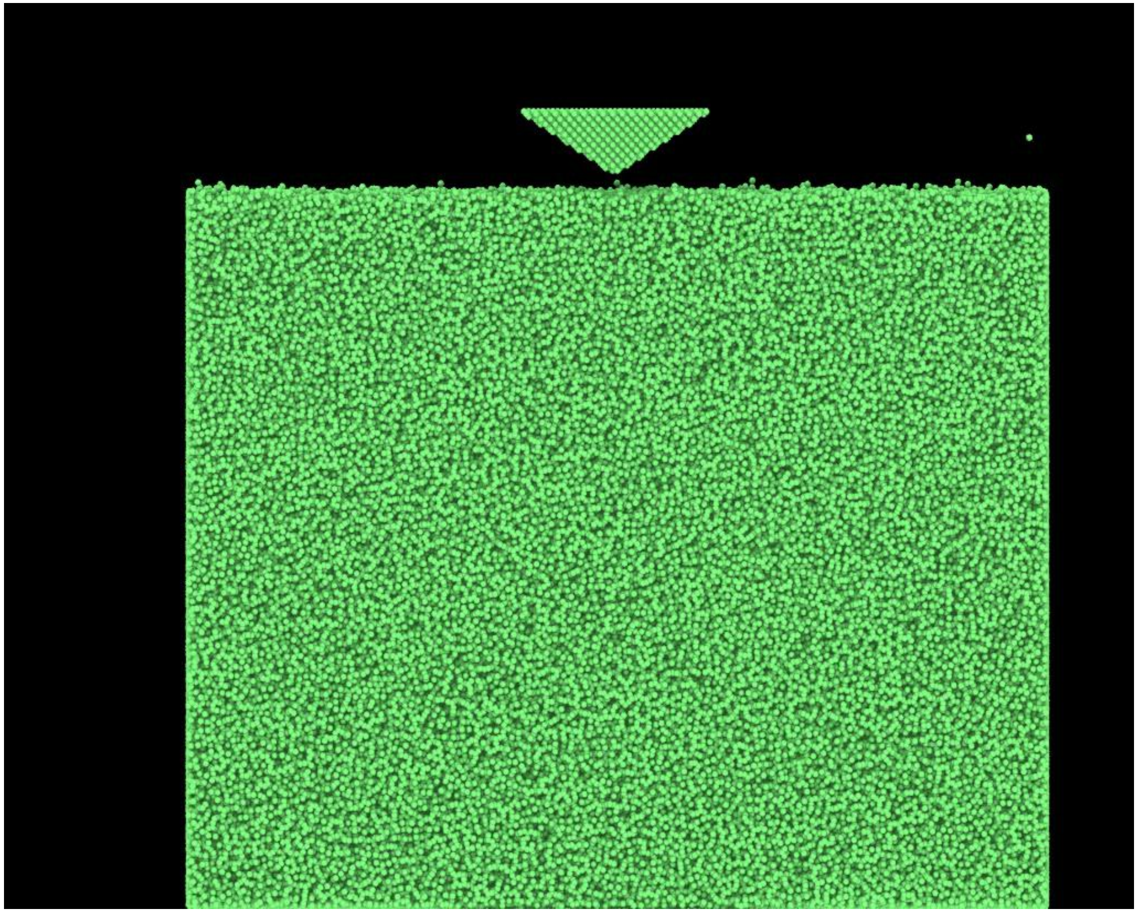
Model:

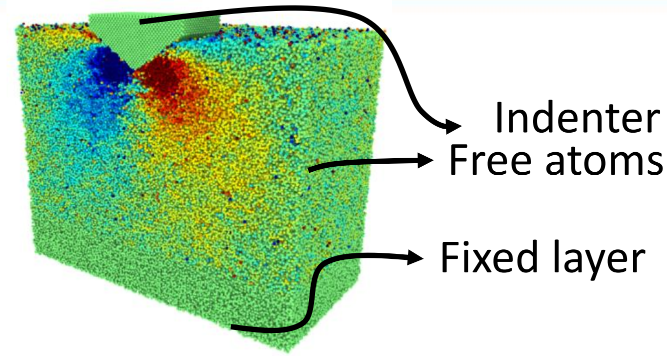
- ➢ MD Simulations (**DL-POLY**).
- ➢ Systems of N particules.
- ➢ Time step: 1 fs

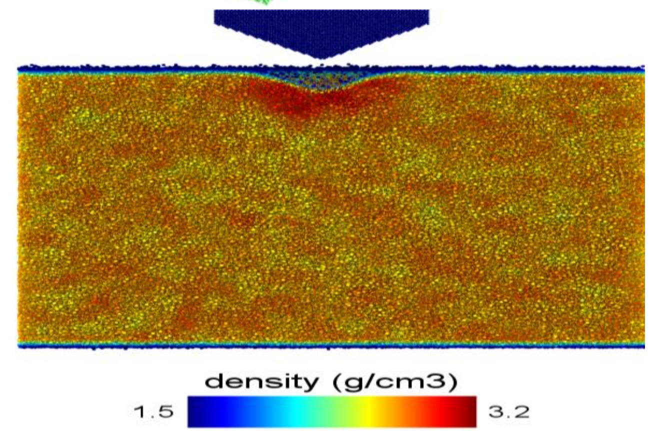
SBN glasses:

- ➢ SiO_2-B_2O_3-Na_2O
  - ✓ R = [Na_2O] / [B_2O_3]
  - ✓ K = [SiO_2] / [B_2O_3]

Glass Indentation







- N = 2.1 x 10^6 atoms
- Temperature : 300 K
- Speed : 10 m/s
- Depth: ~3.0 nm

Movie provided by: Dimitrios Kilymis
UMR 5221 CNRS-Univ. Montpellier, France.

Acknowledgments

Prof. Dr. Jürgen Horbach, Dusseldorf, Germany.
Prof. Dr. Kurt Binder, Mainz, Germany.
Prof. A. Meyer and Prof. D. Herlach (DLR), Koln.

Prof. Normand Mousseau, Qc, Canada.
Prof. Laurent J. Lewis, Qc, Canada.

Dr. Dimitrios Kilymis, Montpellier, France.
Prof. Jean-Marc Delaye, CEA, France.

Dr. Victor Teboul, Angers, France.
Prof. Hamid Bouzar, UMMTO, Tizi-Ouzou, Algeria.

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

DLR

Calcul Québec

Université de Montréal

LABORATOIRE CHARLES COULOMB MONTPELLIER

cea

UNIVERSITE D'ANGERS

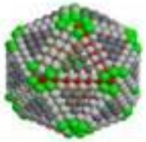
WESTGRID

UNIVERSITY OF MANITOBA

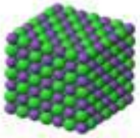
compute canada | calcul canada

Setting and Running MD simulations (LAMMPS)

- LAMMPS: Molecular Dynamics Simulator (introduction).
- Building LAMMPS step by step.
- Running LAMMPS (Input, Output, …).
- Benchmark and performance tests.

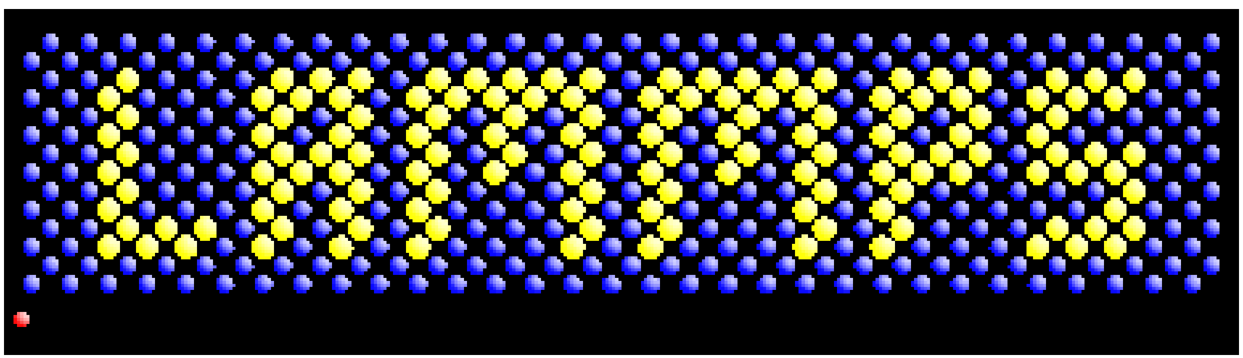


Intorducion to LAMMPS

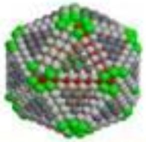


LAMMPS

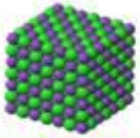
Large-scale **A**tomic / **M**olecular **M**assively **P**arallel **S**imulator



Source: some material and images were adapted from LAMMPS home page



Start with LAMMPS



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

S. Plimpton, A. Thompson, R. Shan, S. Moore, A. Kohlmeyer …

Sandia National Labs: *http://www.sandia.gov/index.html*

- **Home Page: http://lammps.sandia.gov/**

Results:

- **Papers: http://lammps.sandia.gov/papers.html**
- **Pictures: http://lammps.sandia.gov/pictures.html**
- **Movies: http://lammps.sandia.gov/movies.html**

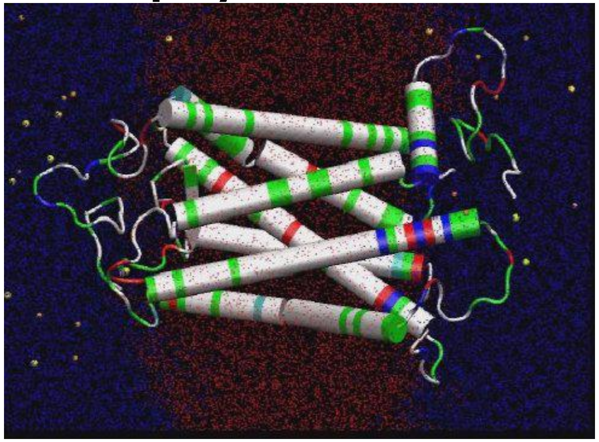
Resources:

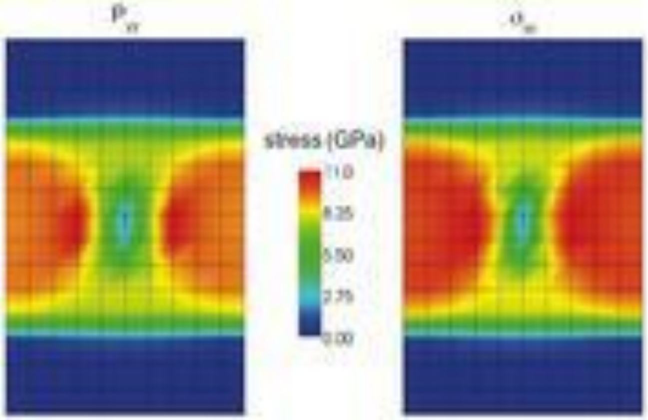
- **Online Manual: http://lammps.sandia.gov/doc/Manual.html**
- **Search the mailing list: http://lammps.sandia.gov/mail.html**
- **Subscribe to the Mailing List:**

https://sourceforge.net/p/lammps/mailman/lammps-users/

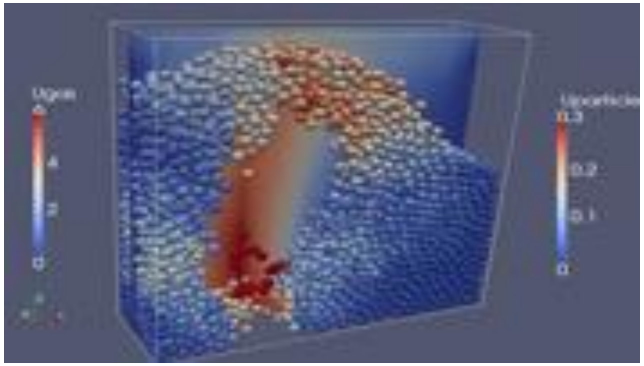
LAMMPS use cases

- Biophysics

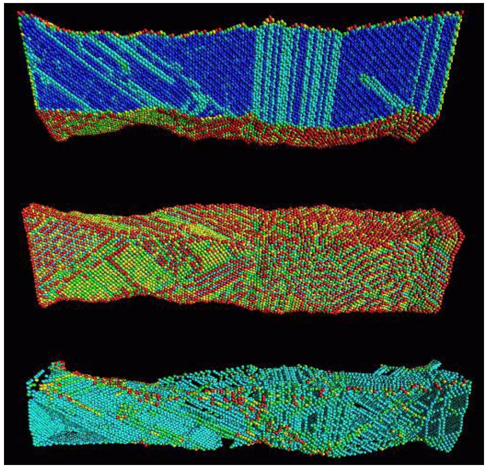




- Solid Mechanics

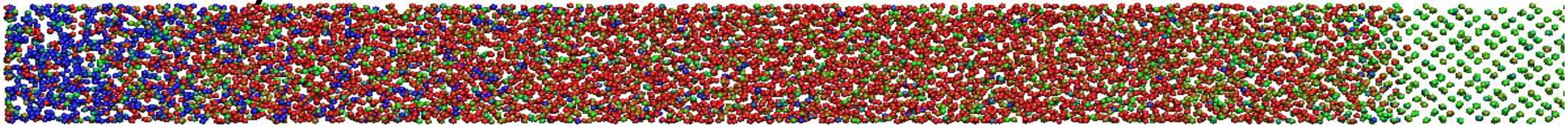


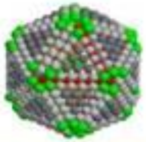
- Granular Flow



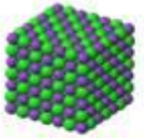
- Material Science

- Chemistry





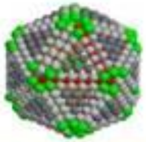
LAMMPS Home Page



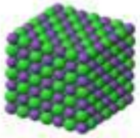
Big Picture	Code	Documentation	Results	Related Tools	Context	User Support
Features	Download	Manual	Publications	Pre/Post processing	Authors	Mail list
Non-features	SourceForge	Developer guide	Pictures	Pizza.py Toolkit	History	Workshops
FAQ	Latest features & bug fixes	Tutorials	Movies	Offsite LAMMPS packages & tools	Funding	User scripts and HowTos
Wish list	Unfixed bugs	MD to LAMMPS glossary	Benchmarks	Visualization	Open source	Contribute to LAMMPS

Recent LAMMPS News

- NEW (9/17) Wrapper on the LATTE DFTB (density-functional tight-binding) quantum code via the fix latte command. See details here.
- NEW (9/17) USER-MESO package from the Karniadakis group at Brown University, with various dissipative particle dynamics (DPD) models, including eDPD, mDPD, tDPD. See details here.
- NEW (8/17) New stable release, 11Aug17 version.
- NEW Biennial LAMMPS Workshop and Symposium in ABQ, NM. PDFs of talks and posters and the tutorial sessions are available at the workshop link.
- NEW (3/17) New stable release, 31Mar17 version.
- NEW (1/17) Added a fix mscg command to enable building of multi-scale coarse-graining (MSCG) models via the Voth group's (U Chicago) MS-CG library.
- NEW (12/16) Significant features added to LAMMPS in the fourth quarter of 2016 include these new commands: compute global/atom global_atom.html, temper/grem and fix grem, pair tersoff/mod/c, pair agni, pair born/coul/dsf and pair_style born/coul/dsf/cs, dump nc and dump nc/mpiio, fix halt, fix dpd/energy, dump_modify thresh LAST option, and fix wall/gran/region. See authors here and details here.



Design of LAMMPS code

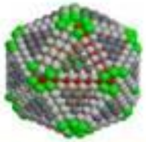


❖ License

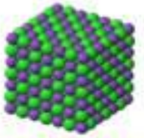
- LAMMPS is provided through **GNU Public License**
  https://www.gnu.org/licenses/licenses.en.html#GPL
- Free to Use, **Modify**, and Distribute.
- Contribute to LAMMPS:
  http://lammps.sandia.gov/contribute.html

❖ Code Layout

- C++ and Object-Oriented approach
- Parallelization via **MPI** and **OpenMP**; runs on **GPU**.
- is invoked by commands through input scripts.
- **p**ossibility to customized output.
- could be interfaced with other codes (python, …).



How to obtain LAMMPS?



❖ Download Page:

http://lammps.sandia.gov/download.html

➢ Distributions:

- ✓ Download a tarball ← **Source Code**
- ✓ Git checkout and update
- ✓ SVN checkout and update
- ✓ Pre-built Ubuntu executables ← **Executable Ubuntu**
- ✓ Pre-built binary RPMs for Fedora/RedHat/CentOS/openSUSE ← **RPMs - Linux**
- ✓ Pre-built Gentoo executable
- ✓ OS X with Homebrew ← **Mac**
- ✓ Windows installer package ← **Installation under Windows**
- ✓ Applying patches

Building LAMMPS

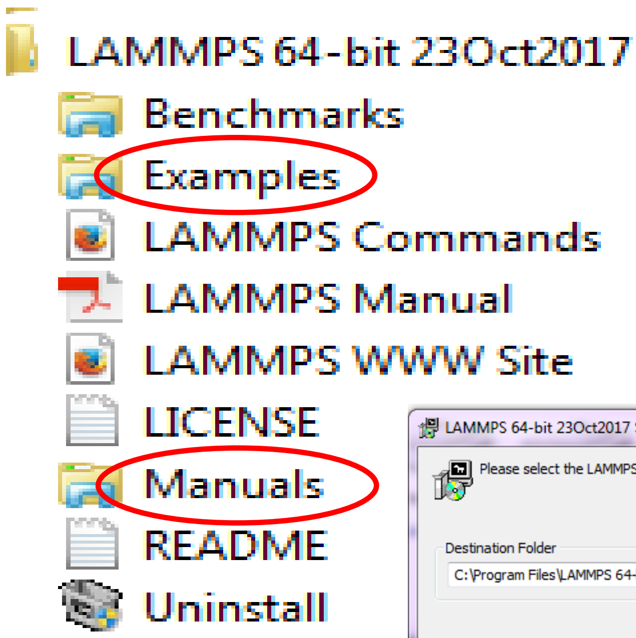
- Build from RPMs
  - ✓ Pre-built Ubuntu executables
  - ✓ Pre-built binary RPMs for Fedora/RedHat/CentOS/openSUSE
  - ✓ Pre-built Gentoo executable
  - ✓ OS X with Homebrew
- Install under windows
  - ✓ Windows installer package
- Build from source code
  - ✓ Download a tarball
  - ✓ Git checkout and update
  - ✓ SVN checkout and update
  - ✓ Applying patches

does not include all packages

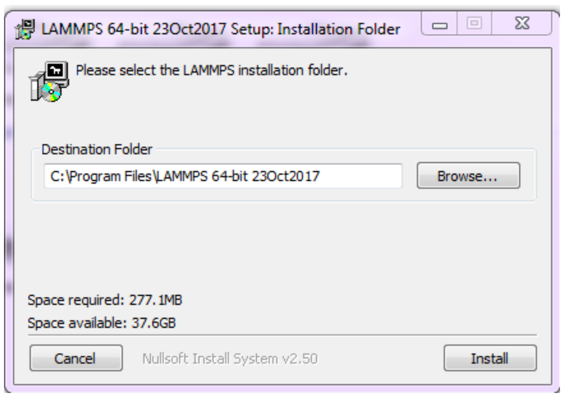
for a customized installation, build from source files: modules

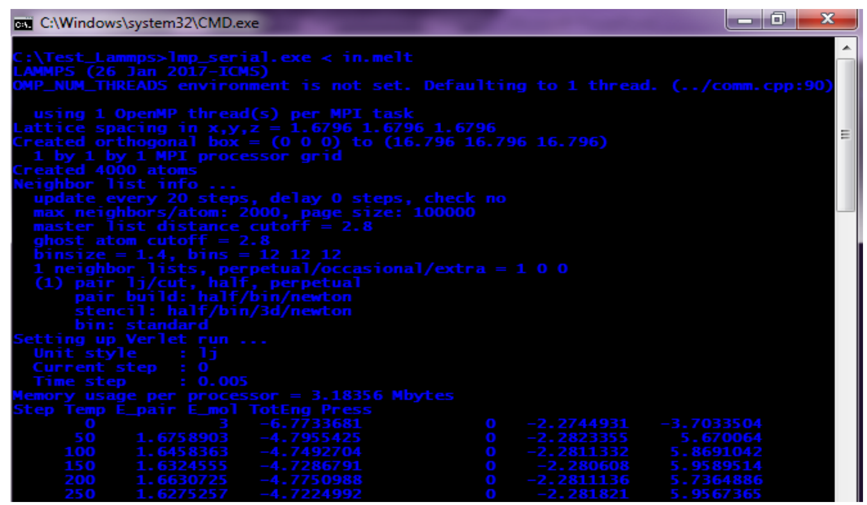
LAMMPS under Windows

- Download Page: http://rpm.lammps.org/windows.html
- Installer: lammps-64bit-latest.exe



Directory:
Program Files\LAMMPS 64-bit 20171023
Executable under bin:
abf_integrate.exe ffmpeg.exe **lmp_mpi.exe** **restart2data.exe** binary2txt.exe **lmp_serial.exe** chain.exe msi2lmp.exe createatoms.exe





- Execute:

lmp_serial.exe < in.lammps

Building LAMMPS from source

http://lammps.sandia.gov/download.html#tar

Download a tarball

Select the code you want, click the "**Download Now**" button, and your browser should download a gzipped tar file. Unpack it with the following commands, and look for a README to get you started.

```
tar -xzvf file.tar.gz
```

There have been ~256,700 downloads of LAMMPS from Sept 2004 thru Dec 2016.

LAMMPS molecular dynamics package:

- ○ LAMMPS --- Stable version (11 Aug 2017) - Recent C++ version source tarball, GPL license, ~121 Mb. Includes all bug fixes and new features described on this page, up to the date of the most recent stable release.
- ○ LAMMPS --- Development version - Most current C++ version source tarball, GPL license, ~121 Mb. Includes all bug fixes and new features described on this page.
- ○ LAMMPS 2001 --- older f90 version source tarball, GPL license, 1.1 Mb, last updated 17 Jan 2005
- ○ LAMMPS 99 --- older f77 version source tarball, GPL license, 840 Kb

◉ No package

[Download Now]

Archive: lammps-stable.tar.gz

LAMMPS source overview

- Download the source code: lammps-stable.tar.gz
- LAMMPS directory: lammps-11Aug17
  - ✓ bench: Benchmark tests (potential, input and output files).
  - ✓ doc: documentation (PDF and HTML)
  - ✓ examples: input and output files for some simulations
  - ✓ lib: libraries to build before building LAMMPS
  - ✓ LICENSE and README files.
  - ✓ potentials: some of the force fields supported by LAMMPS
  - ✓ python: to invoke LAMMPS library from Python
  - ✓ src: source files (*.cpp, PACKAGES, USER-PACKAGES, ...)
  - ✓ tools: some tools like xmovie (similar to VMD but only 2D).

Building LAMMPS

- First: Build libraries if required.
- Choose a Makefile compatible with your system
- Choose and install the packages you need.
  - make package # list available packages
  - make package-status (ps) # status of all packages
  - make yes-package # install a single package in src
  - make no-package # remove a single package from src
  - make yes-all # install all packages in src
  - make no-all # remove all packages from src
  - make yes-standard (yes-std) # install all standard packages
  - make no-standard (no-std) # remove all standard packages
  - make yes-user # install all user packages
  - make no-user # remove all user packages
- Build LAMMPS:
  - make machine # build LAMMPS for machine

Use GNU Make to build LAMMPS

- machine is one of these from src/MAKE:
  - # mpi = MPI with its default compiler
  - # serial = GNU g++ compiler, no MPI
- … or one of these from src/MAKE/OPTIONS:
  - # icc_openmpi = OpenMPI with compiler set to Intel icc
  - # icc_openmpi_link = Intel icc compiler, link to OpenMPI
  - # icc_serial = Intel icc compiler, no MPI
- … or one of these from src/MAKE/MACHINES:
  - # cygwin = Windows Cygwin, mpicxx, MPICH, FFTW
  - # mac = Apple PowerBook G4 laptop, c++, no MPI, FFTW 2.1.5
  - # mac_mpi = Apple laptop, MacPorts Open MPI 1.4.3, …
  - # ubuntu = Ubuntu Linux box, g++, openmpi, FFTW3
- … or one of these from src/MAKE/MINE: (write your own Makefile)

Building LAMMPS: demonstration

- Download the latest stable version from LAMMPS home page.
- Untar the archive: tar -xvf lammps-stable.tar.gz
- Change the directory and list the files: cd lammps-11Aug17

  bench bin doc examples lib LICENSE potentials

  python README src tools
- Choose a Makefile (for example: machine=icc_openmpi)

  src/MAKE/OPTIONS/Makefile.icc_openmpi
- Load the required modules (Intel, OpenMPI, …)
- Check the packages:

  package, package-status, yes-package, no-package, …
- to build LAMMPS, run: **make icc_openmpi**
- Add or remove a package (if necessary), then recompile
- If necessary, edit Makefile and fix the path to libraries.

Running LAMMPS

- Executable: lmp_machine

- Files:
  - Input File: in.lmp_file
  - Potential: see examples and last slides for more details
  - Initial configuration: can be generated by LAMMPS, or another program or home made program.

- Interactive Execution:

  $ ./lmp_machine < in.lmp_file

  $ ./lmp_machine –in in.lmp_file

- Redirect output to a file:

  $ ./lmp_machine < in.lmp_file > output_file

  $ ./lmp_machine –in in.lmp_file –l output_file

Command line options

❑ Command-line options:

At run time, LAMMPS recognizes several optional command-line switches which may be used in any order.

-e or -echo, -h or –help, -i or –in, -k or –kokkos, -l or –log,

-nc or –nocite, -pk or –package, -p or –partition, -pl or –plog,

-ps or –pscreen, -r or –restart, -ro or –reorder, -sc or –screen,

-sf or –suffix, -v or –var

❑ For example:

mpirun -np 8 lmp_machine -l my.log -sc none -in in.alloy

mpirun -np 8 lmp_machine < in.alloy > my.log

Overview of a simulation run

INPUT	RUNNING	OUTPUT
• Initial positions • Initial velocities • Time step • Mass • PBC • Units • Potential • Ensemble • …. etc.	• Molecular Dynamics Simulation (NPT, NVT, NVE) • Minimization • Monte Carlo • Atomic to Continuum	• Trajectories • Velocities • Forces • Energy • Temperature • Pressure • Density • Snapshots • Movies • … etc.

Overview of a Simulation Run

- **Command Line:**
  - Every simulation is executed by supplying an input text script to the LAMMPS executable: lmp < lammps.in > log_lammps.txt
- **Parts of an input script:**
  - **Initialize:** units, dimensions, PBC, etc.
  - Atomic positions (built in or read from a file) and velocities.
  - **Settings:**
    - ✓ Inter-atomic potential (pair_style, pair_coeff)
    - ✓ Run time simulation parameters (e.g. time step)
    - ✓ Fixes: operations during dynamics (e.g. thermostat)
    - ✓ Computes: calculation of properties during dynamics
- **Run the simulation for N steps.**

LAMMPS input example: LJ melt

```
# 3d Lennard-Jones melt

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

# Potential

pair_style     lj/cut  2.5
pair_coeff     1 1  1.0  1.0  2.5
```

- Comment
- Define units
- Create the simulation box
  Or read data from a file
- Initialize the velocities
- Define the potential

LAMMPS input example: LJ melt

```
# Neighbour list:
neighbor          0.3 bin
neigh_modify      every 20 delay 0 check no

#  set the thermodynamic ensemble:
fix               1 all nve

dump              id all atom 50 dump.melt
#dump_modify ....

log               log.melt
thermo_style  custum  step temp etotal ....
thermo            50

run               250
# End of the simulation.
```

- Monitor the neighbour list (neighbor / neigh_modify)
- Thermodynamic Ensemble (fix)
- Store the trajectory (dump)
- Log file: customize output (log / thermo_style / thermo)
- Run the simulation for N steps (run)

LAMMPS: input commands

Initialization

- **Parameters:** set parameters that need to be defined before atoms are created: ***units***, *dimension*, *newton*, *processors*, ***boundary***, ***atom_style***, *atom_modify*.

- If force-field parameters appear in the files that will be read:
  pair_style, *bond_style*, *angle_style*, *dihedral_style*, *improper_style*.

- **Atom definition:** there are 3 ways to define atoms in LAMMPS.
  - ✓ Read them in from a data or restart file via the ***read_data*** or ***read_restart*** commands.
  - ✓ Or create atoms on a lattice (with no molecular topology), using these commands: *lattice*, *region*, *create_box*, *create_atoms*.
  - ✓ Duplicate the box to make a larger one the ***replicate*** command.

LAMMPS: settings

- Once atoms are defined, a variety of settings need to be specified: **force field coefficients**, **simulation parameters**, **output options** ...

  - **Force field coefficients:**
    pair_coeff, *bond_coeff*, *angle_coeff*, *dihedral_coeff*, *improper_coeff*, *kspace_style*, *dielectric*, *special_bonds*.

  - **Various simulation parameters:**
    neighbor, *neigh_modify*, *group*, *timestep*, *reset_timestep*, *run_style*, *min_style*, *min_modify*.

  - **Fixes:** *nvt*, *npt*, *nve*, ...

  - **Computations during a simulation:**
    compute, *compute_modify*, and *variable* commands.

  - **Output options:** *thermo*, *dump*, and *restart* commands.

Cutumize the output

thermo **freq_steps**
thermo_style **style args**

- **style =** *one* or *multi* or *custom*
- **args =** list of arguments for a particular style
  - *one* args = none
  - *multi* args = none *custom*
  - args = list of keywords possible

- **keywords = step, elapsed, elaplong, dt, time, cpu, tpcpu, spcpu, cpuremain, part, timeremain, atoms, temp, press, pe, ke, etotal, enthalpy, evdwl, ecoul, epair, ebond, eangle, edihed, eimp, emol, elong, etail, vol, density, lx, ly, lz, xlo, xhi, ylo, yhi, zlo, zhi, xy, xz, yz, xlat, ylat, zlat, bonds, angles, dihedrals, impropers, pxx, pyy, pzz, pxy, pxz, pyz ….. etc**

Running LAMMPS: demonstration

- After compiling LAMMPS, run some examples:

- Where to start to learn LAMMPS?
  - Make a copy of the directory examples in your working directory.
  - Choose and example to run.
  - Indicate the right path to the executable.
  - Edit the input file and check all the parameters.
  - Check the documentation for the commands and their arguments.
  - Run the test case: lmp_icc_openmpi < in.melt .
  - Check the output files (log files), plot the thermodynamic properties, ...

LAMMPS: output example

LAMMPS (30 Jul 2016)
using 1 2 OpenMP thread(s) per MPI task
 # 3d Lennard-Jones melt

units ljatom_style atomic
lattice fcc 0.8442Lattice spacing in x,y,z = 1.6796 1.6796 1.6796
region box block 0 10 0 10 0 10
create_box 1 box

Created orthogonal box = (0 0 0) to (16.796 16.796 16.796) 2 by 2 by 3 MPI processor grid

create_atoms 1 box
Created 4000 atoms

mass 1 1.0

LAMMPS: output example

```
thermo          100
run             25000
Neighbor list info ...
    1 neighbor list requests
    update every 20 steps, delay 0 steps, check no
    max neighbors/atom: 2000, page size: 100000
    master list distance cutoff = 2.8
    ghost atom cutoff = 2.8
    binsize = 1.4 -> bins = 12 12 12
    Memory usage per processor = 2.05293 Mbytes
Step Temp E_pair E_mol TotEng Press
0    3          -6.7733681  0  -2.2744931  -3.7033504
100  1.6510577  -4.7567887  0  -2.2808214   5.8208747
200  1.6393075  -4.7404901  0  -2.2821436   5.9139187
300  1.6626896  -4.7751761  0  -2.2817652   5.756386
```

LAMMPS: output example

25000 1.552843 -4.7611011 0 -2.432419 5.7187477

Loop time of 15.4965 on 12 procs for 25000 steps with 4000 atoms
Performance: 696931.853 tau/day, 1613.268 timesteps/s
90.2% CPU use with 12 MPI tasks x 1 OpenMP threads

MPI task timing breakdown:

Section	min time	avg time	max time	%varavg	%total
Pair	6.6964	7.1974	7.9599	14.8	46.45
Neigh	0.94857	1.0047	1.0788	4.3	6.48
Comm	6.0595	6.8957	7.4611	17.1	44.50
Output	0.01517	0.01589	0.019863	1.0	0.10
Modify	0.14023	0.14968	0.16127	1.7	0.97
Other		0.2332			1.50

Total wall time: 0:00:15

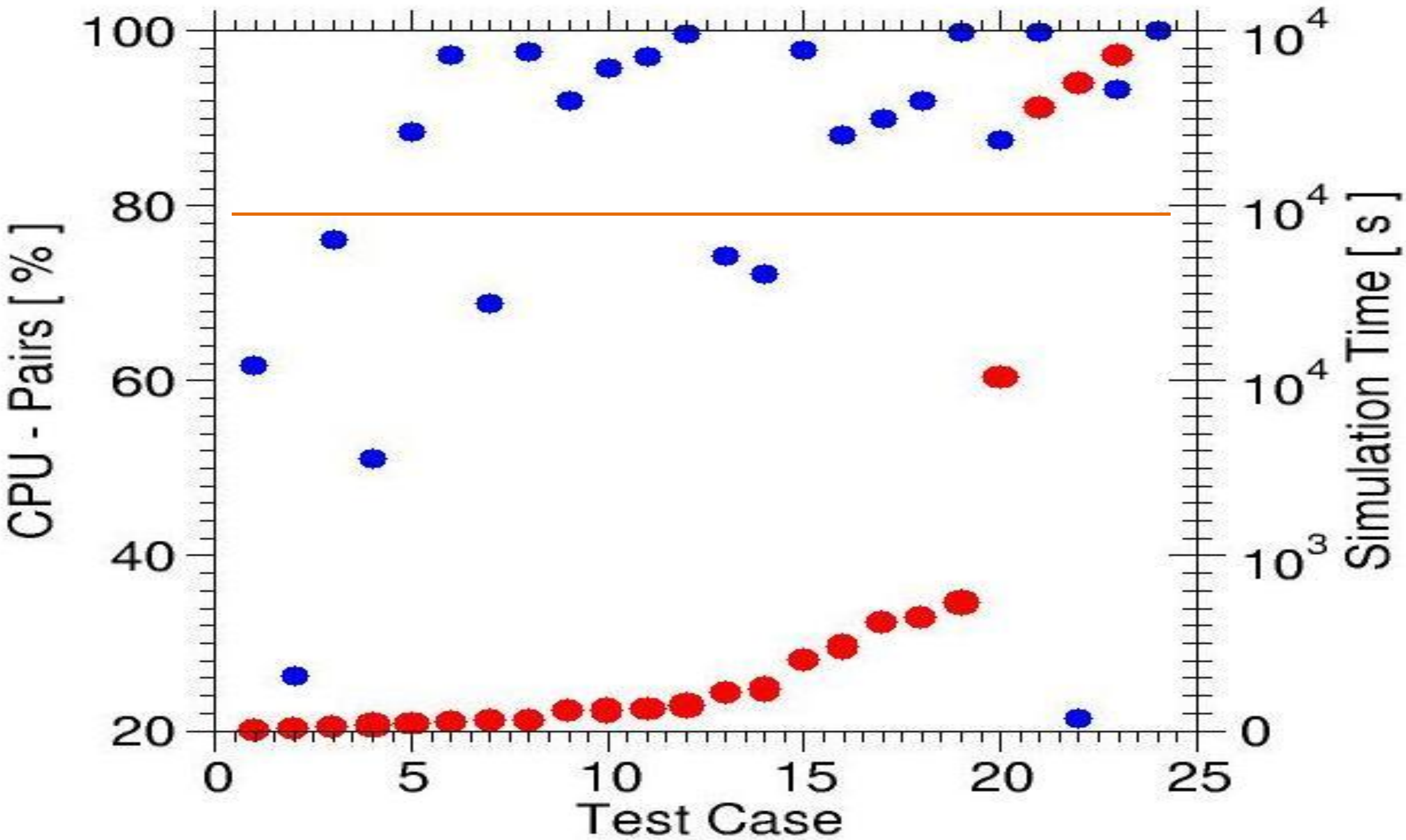
Potential Benchmark

1. granular
2. fene
3. lj
4. dpd
5. eam
6. sw
7. rebo
8. tersoff
9. eim
10. adp
11. meam
12. peri
13. spce
14. protein
15. gb
16. reax_AB
17. airebo
18. reaxc_rdx
19. smtbq_Al
20. vashishta_table_sio2
21. eff
22. comb
23. vashishta_sio2
24. smtbq_Al2O3

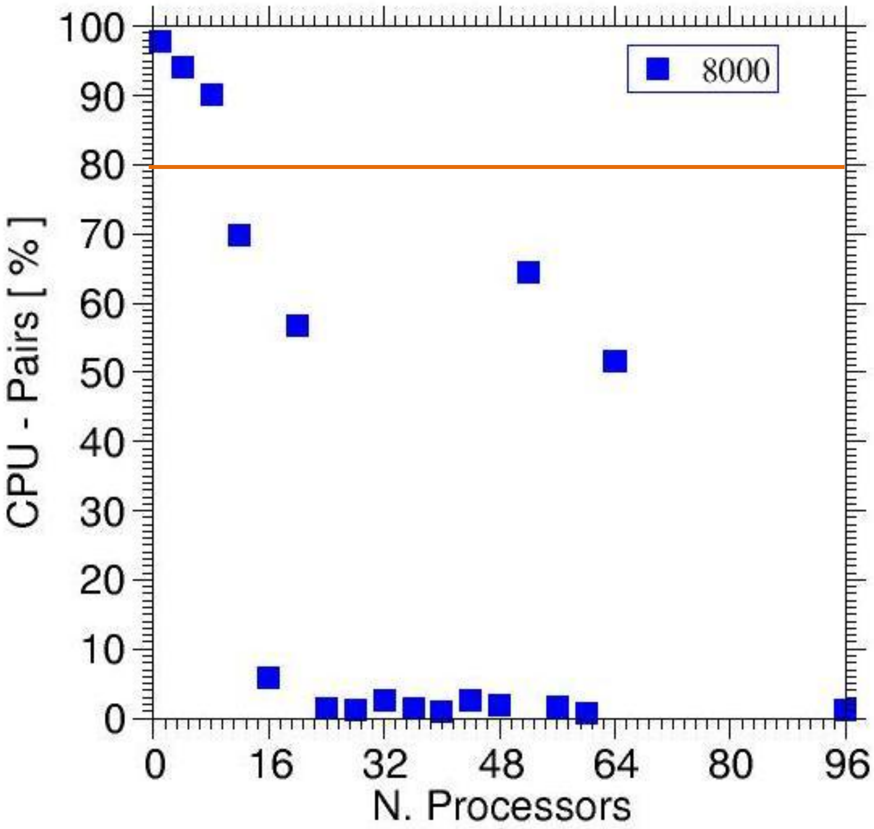
Parameters:

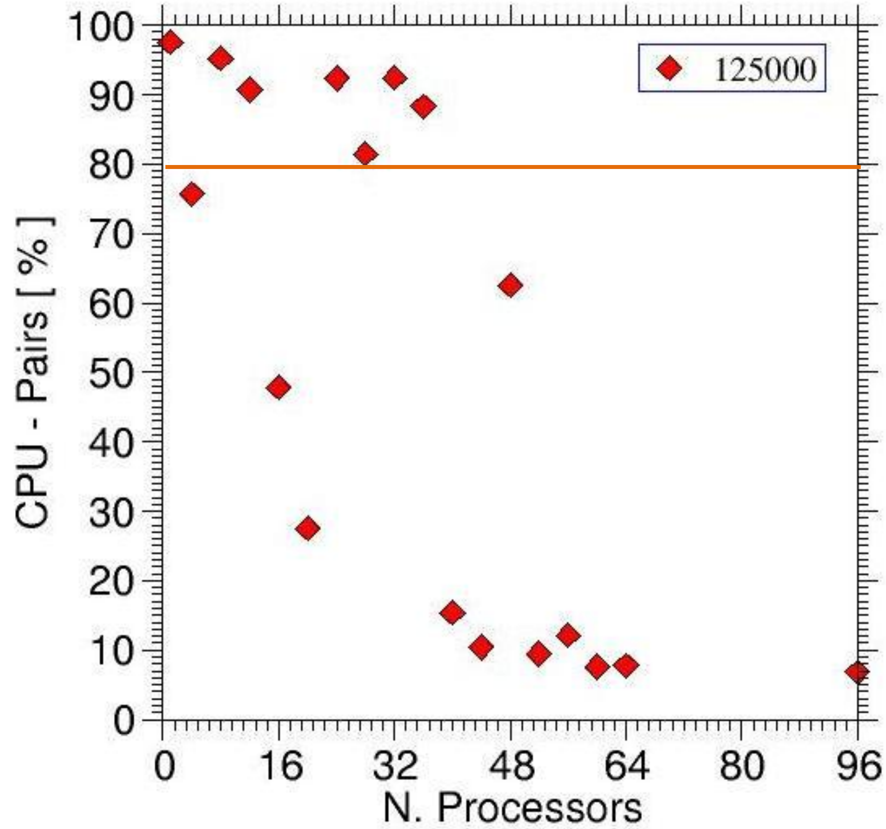
- 24 different cases.
- Number of particles: about 32000
- CPUs = 1
- MD steps = 1000
- Record the simulation time and the time used in computing the interactions between particles.

Potential Benchmark



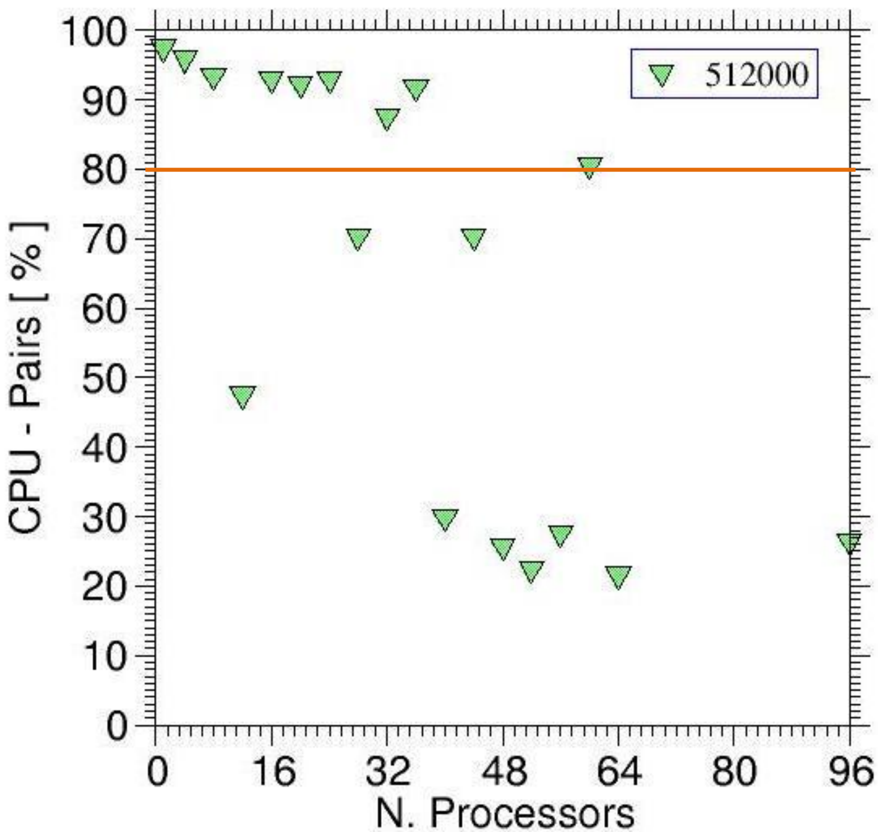
Performance Test: Tersoff potential

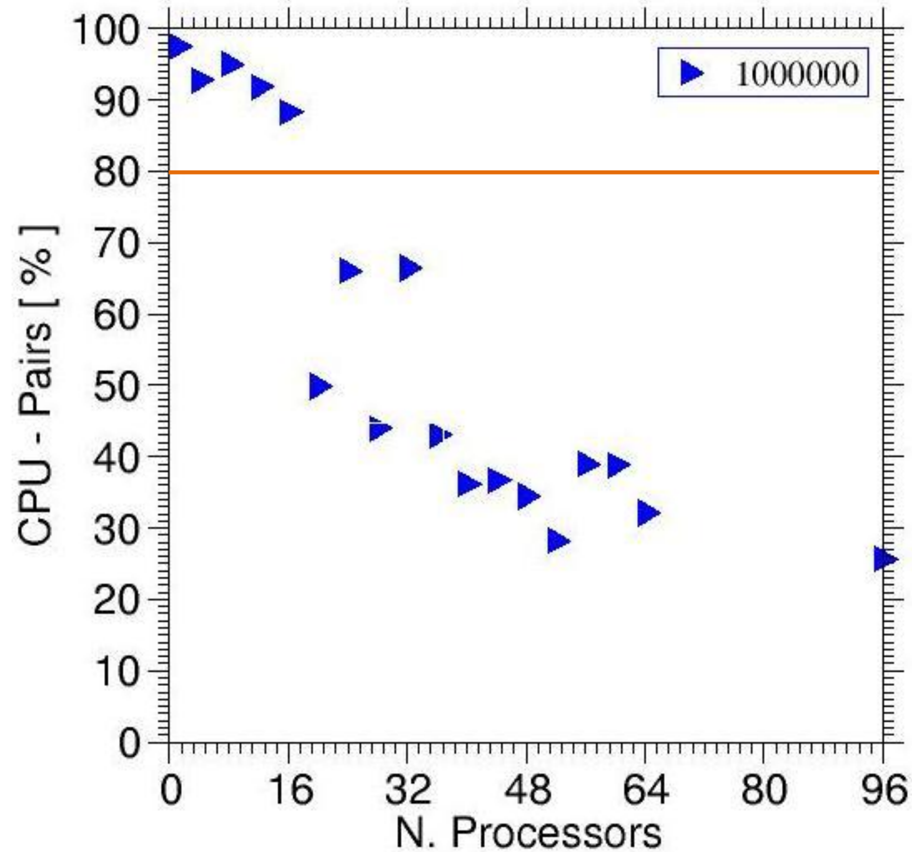




CPU time used for computing the interactions between particles as a function the number of processors for different system size.

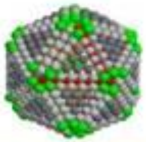
Performance Test: Tersoff potential



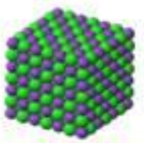


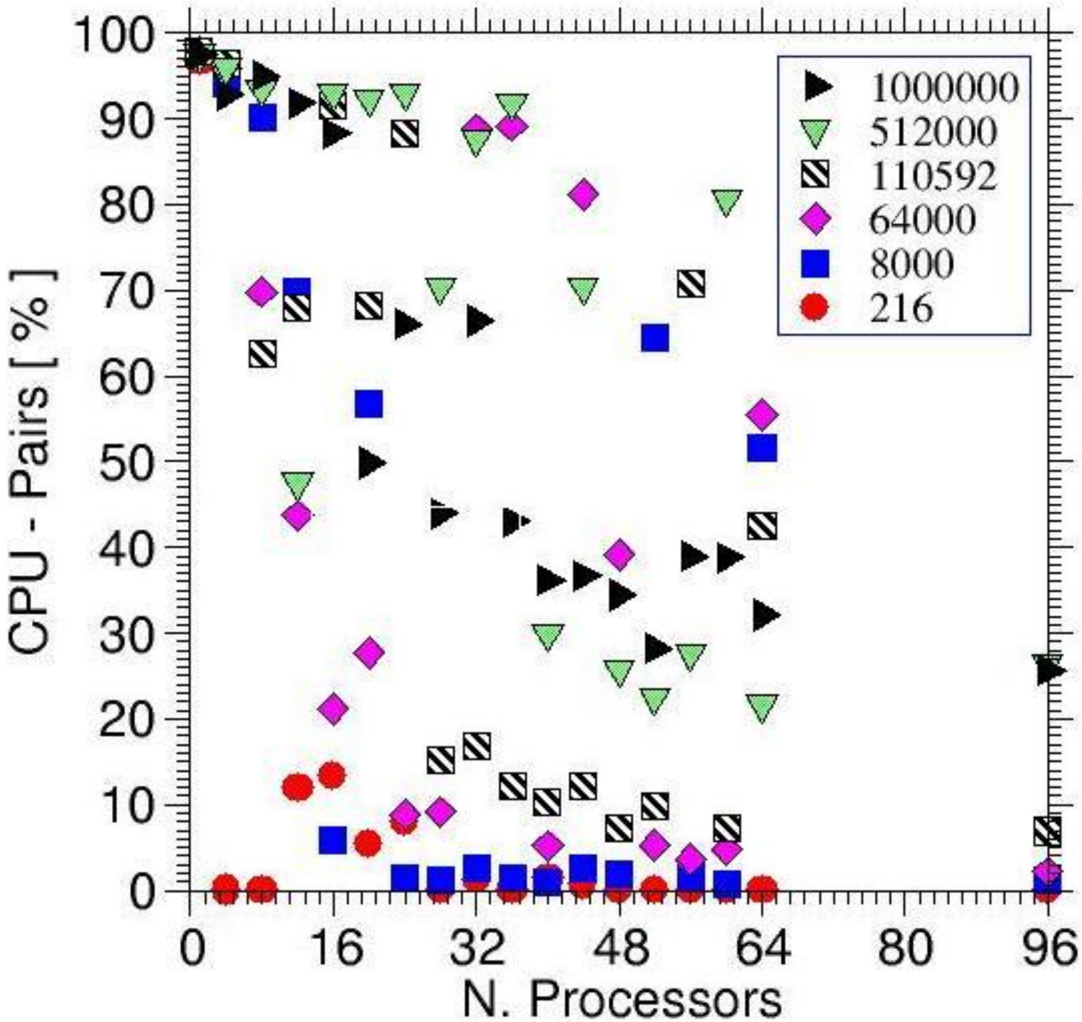
CPU time used for computing the interactions between particles as a function the number of processors for different system size.

WESTGRID UNIVERSITY OF MANITOBA compute canada | calcul canada

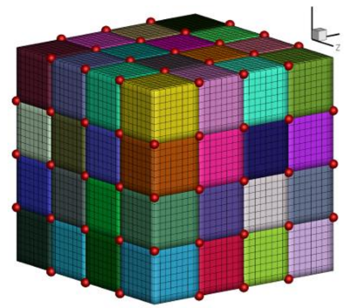


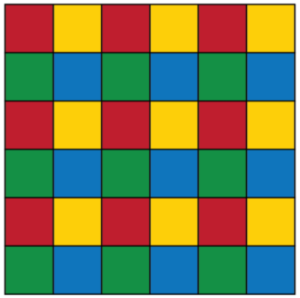
Performance Test: Tersoff potential





- **Domain decomposition**





- **Size, shape of the system.**
- **Number of processors.**
- **size of the small units.**
- **correlation between the communications and the number of small units.**
- **Reduce the number of cells to reduce communications.**

Learn more about LAMMPS

- **Home Page:** http://lammps.sandia.gov/

- **Examples:** deposit, friction, micelle, obstacle, qeq, streitz, MC, body, dipole, hugoniostat, min, peptide, reax, tad, DIFFUSE, colloid, indent, msst, peri, rigid , vashishta, ELASTIC, USER, comb, eim, nb3b, pour, shear, voronoi, ELASTIC_T, VISCOSITY, coreshell, ellipse, meam, neb, prd, snap, HEAT, accelerate, crack, flow, melt, nemd

- **Results:**
  - Papers: http://lammps.sandia.gov/papers.html
  - Pictures: http://lammps.sandia.gov/pictures.html
  - Movies: http://lammps.sandia.gov/movies.html

- **Resources:**
  - Online Manual: http://lammps.sandia.gov/doc/Manual.html
  - Search the mailing list: http://lammps.sandia.gov/mail.html
  - Mailing List:
    https://sourceforge.net/p/lammps/mailman/lammps-users/

Thanks to LAMMPS developers

Thanks to LAMMPS contributors

Thank you for your attention

Potentials: classified by materials

- **Bio-molecules:** CHARMM, AMBER, OPLS, COMPASS (class 2), long-range Coulombic via PPPM, point dipoles, ...
- **Polymers:** all-atom, united-atom, coarse-grain (bead-spring FENE), bond-breaking, …
- **Materials:** EAM and MEAM for metals, Buckingham, Morse, Yukawa, Stillinger-Weber, Tersoff, EDIP, COMB, SNAP, ...
- **Chemistry:** AI-REBO, REBO, ReaxFF, eFF
- **Meso-scale:** granular, DPD, Gay-Berne, colloidal, peri-dynamics, DSMC...
- **Hybrid:** combine potentials for hybrid systems: water on metal, polymers/semiconductor interface, colloids in solution, …

Potentials: classified by functional form

- **Pair-wise potentials:** Lennard-Jones, Buckingham, ...
- **Charged Pair-wise Potentials:** Coulombic, point-dipole
- **Many-body Potentials:** EAM, Finnis/Sinclair, modified EAM (MEAM), embedded ion (EIM), Stillinger-Weber, Tersoff, AI-REBO, ReaxFF, COMB
- **Coarse-Grained Potentials:** DPD, GayBerne, ...
- **Meso-scopic Potentials:** granular, peri-dynamics
- **Long-Range Electrostatics:** Ewald, PPPM, MSM
- **Implicit Solvent Potentials:** hydrodynamic lubrication, Debye
- **Force-Field Compatibility with common:** CHARMM, AMBER, OPLS, GROMACS options