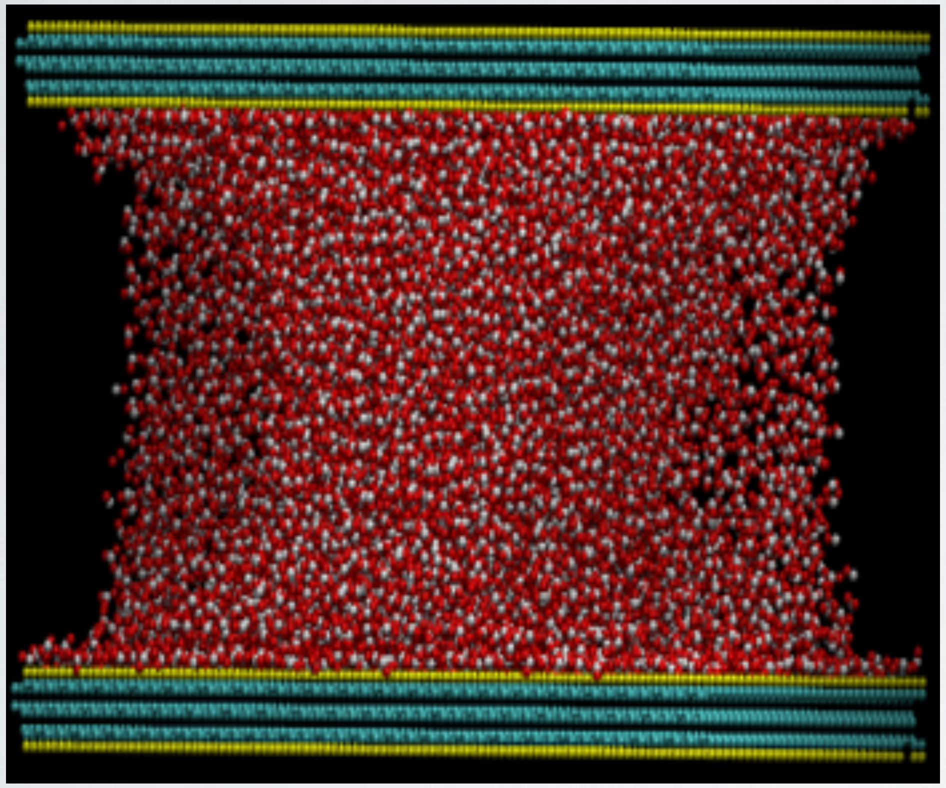
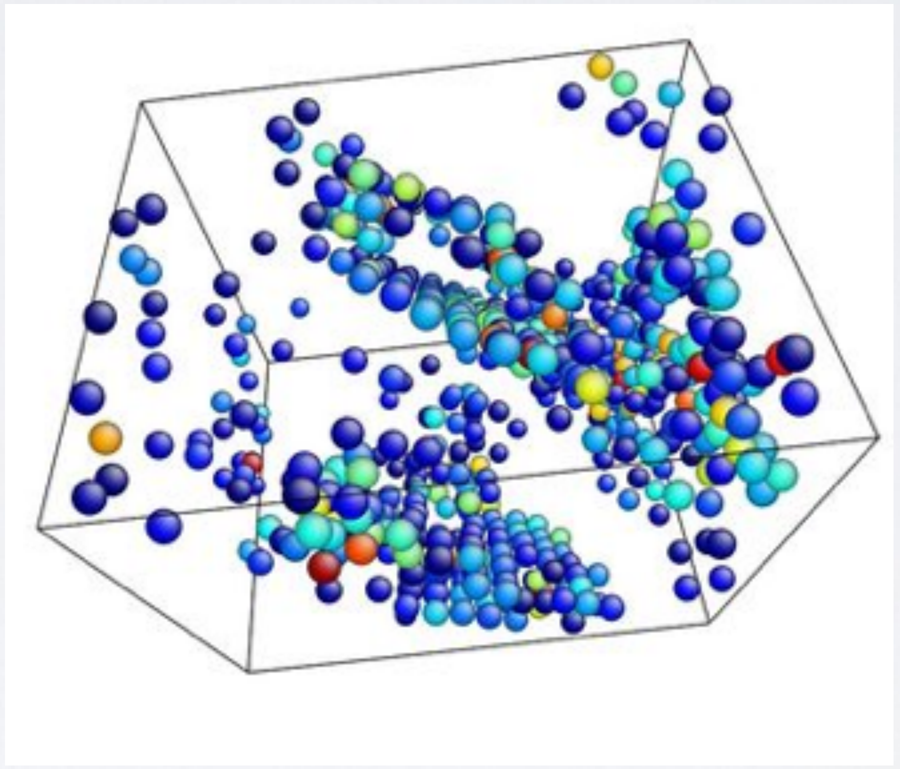
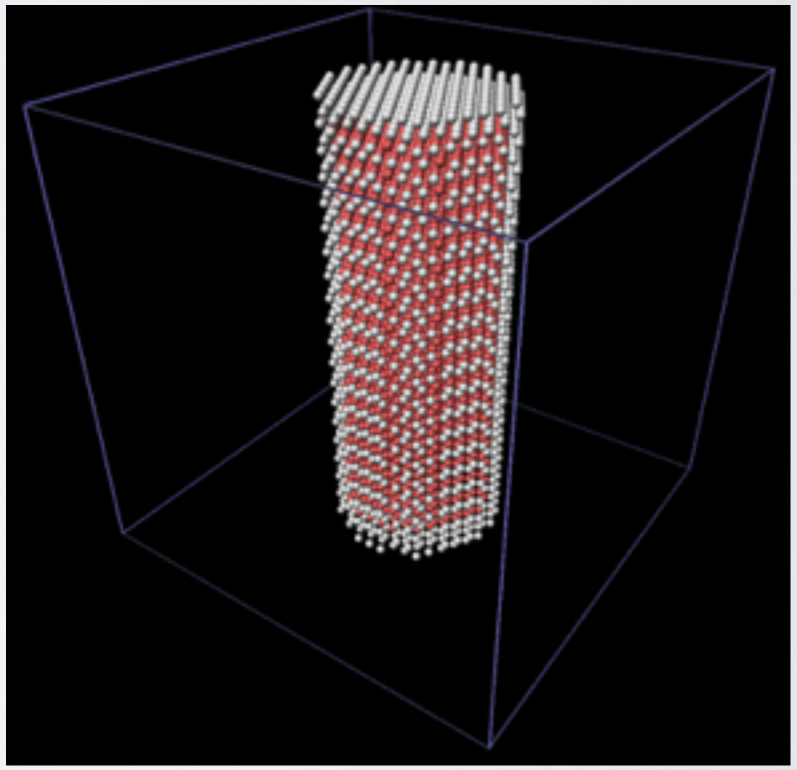
Elements of ICME Research Workshop

Molecular Dynamics with LAMMPS



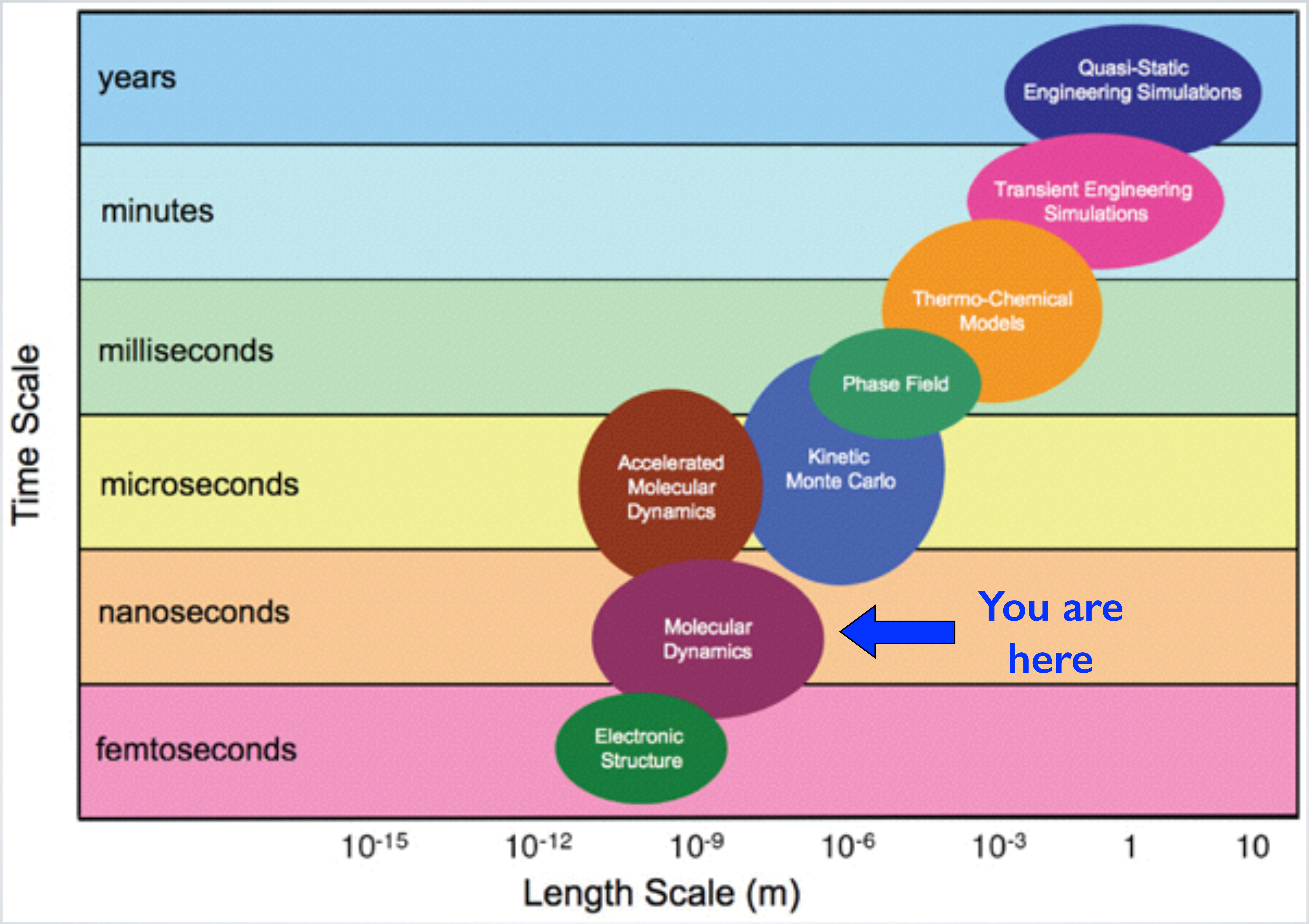




Elements of ICME Research Workshop
UIUC
July 23-25, 2014

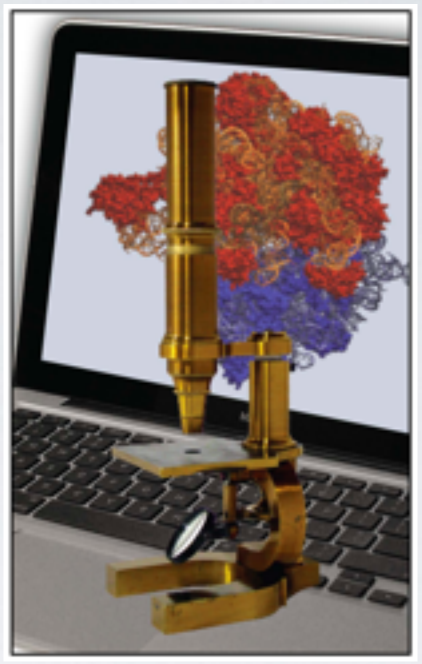
Andrew L. Ferguson
Materials Science and Engineering
University of Illinois at Urbana-Champaign

I. Introduction


Time Scale
years
minutes
milliseconds
microseconds
nanoseconds
femtoseconds
Quasi-Static Engineering Simulations
Transient Engineering Simulations
Thermo-Chemical Models
Phase Field
Kinetic Monte Carlo
Accelerated Molecular Dynamics
Molecular Dynamics
Electronic Structure
You are here
10-15
10-12
10-9
10-6
10-3
1
10
Length Scale (m)

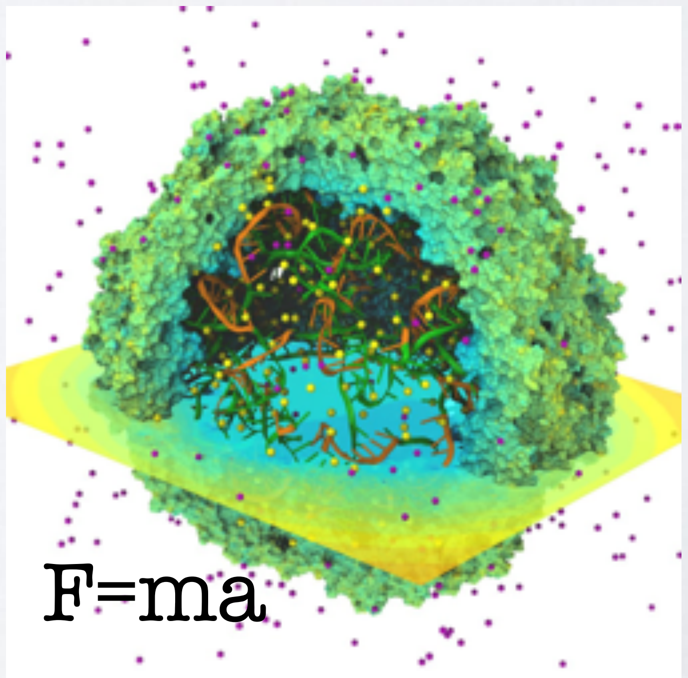
What is molecular dynamics?

- A computational microscope



- An experiment on a computer

- A simulation of the classical mechanics of atoms



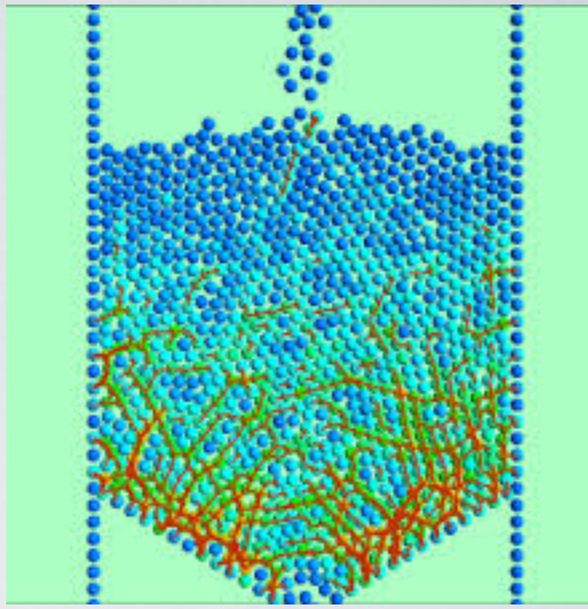
Why is it useful?

- By simulating atomic and molecular motions, we can gain **atomistic insight** into molecular structure and kinetics
- Powerful experimental techniques (X-ray diffraction, NMR) can resolve atomic structure, but not dynamics
- We can **predict and understand** molecular behavior and compare / interpret experimental observations
- Total control of molecular forces, structure, and conditions
- In principle, it can furnish **all** classical thermodynamics about **any** molecular system*

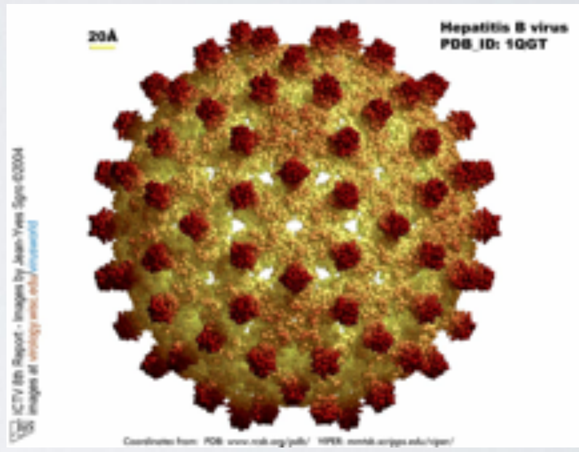
* subject to available force fields and sufficient computational power!

What is it used for?

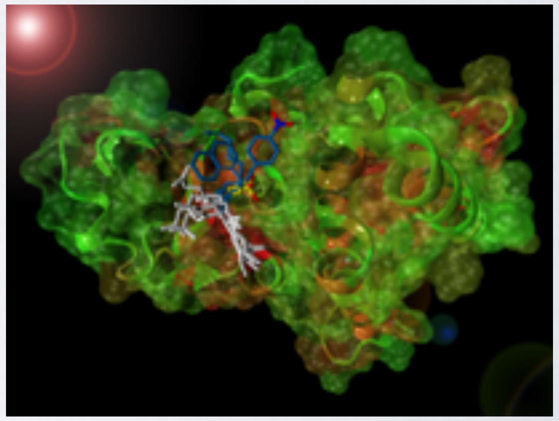
- Materials property prediction
  - bulk modulus, surface tension, shear viscosity, thermal conductivity, flow, gelation



- Biomolecular modeling
  - protein folding, viral capsids, cell membranes, ion transport


20Å
Hepatitis B virus
PDB_ID: 1QGT
ICTV 8th Report - Images by Jean-Yves Sgro ©2004
Images at virology.wisc.edu/virusworld
Coordinates from: PDB: www.rcsb.org/pdb/ VIPER: mmtsb.scripps.edu/viper/

- Ligand and drug design
  - docking, interaction, sterics

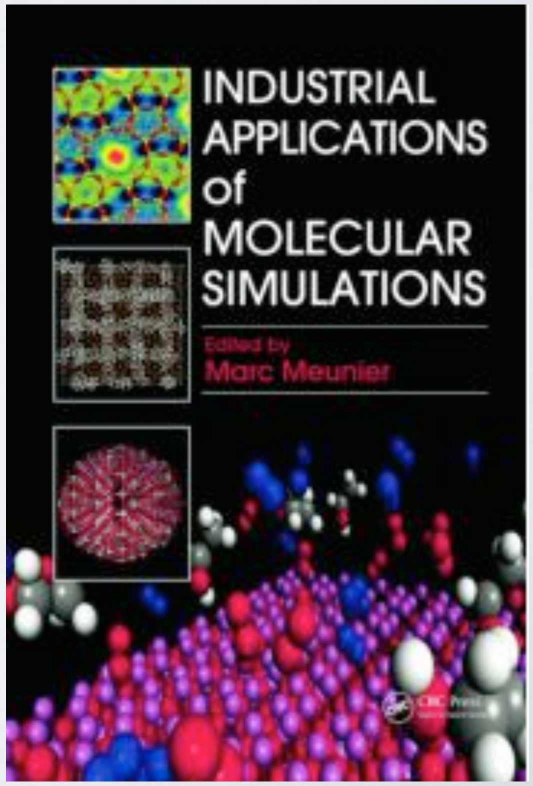


- High-throughput molecular screening
  - drugs, surfactants, self-assembling materials

http://cssb.biology.gatech.edu/sites/default/files/flhm1.png http://www.virology.wisc.edu/virusworld/viruslist.php?virus=hpb http://www.ph.biu.ac.il/~rapaport/visint/05_examp_grandyn.html

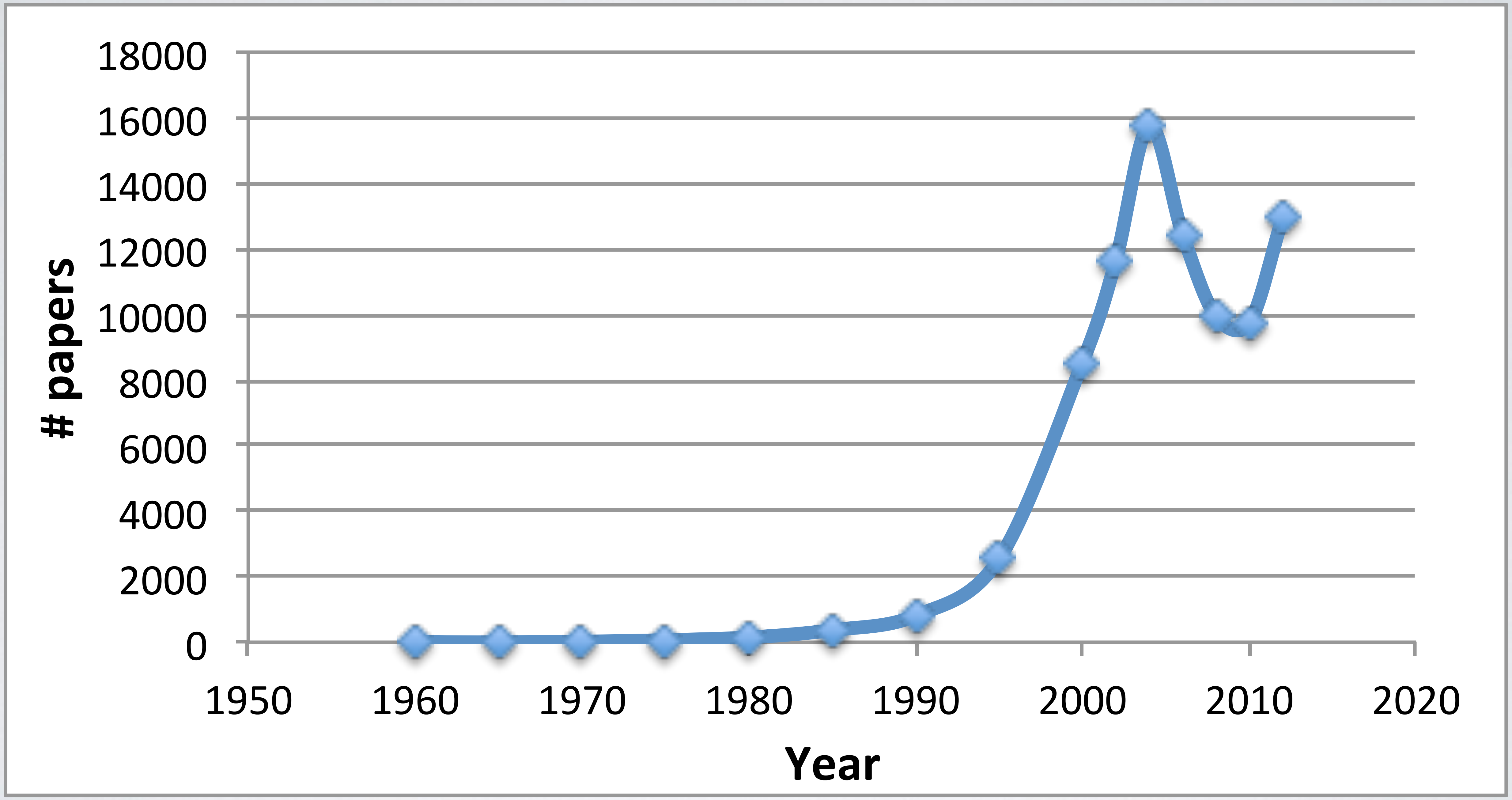
Is it used in industry?

- **YES!**
- Computer power (just) continues to follow Moore's Law, computation gets cheaper every year
- Reliable and validated computational exploration and testing is **much** cheaper and quicker than an R&D lab!
- MD is now a standard tool in pharma, nuclear, chemical, oil, aerospace, electronics, and plastics
- MD is maturing into an "off-the-shelf" tool similar to the emergence of CFD in the 90's



Academic publishing trends

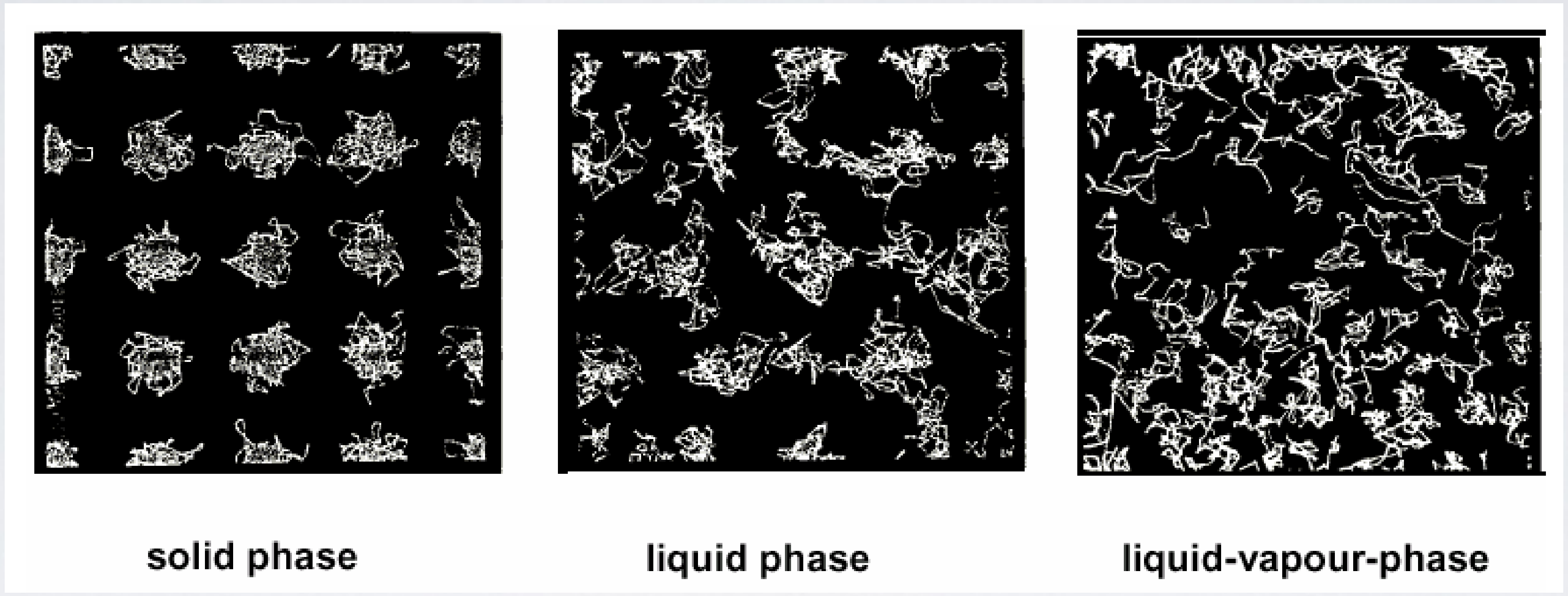
- Scopus abstract/title/keyword search "molecular dynamics"



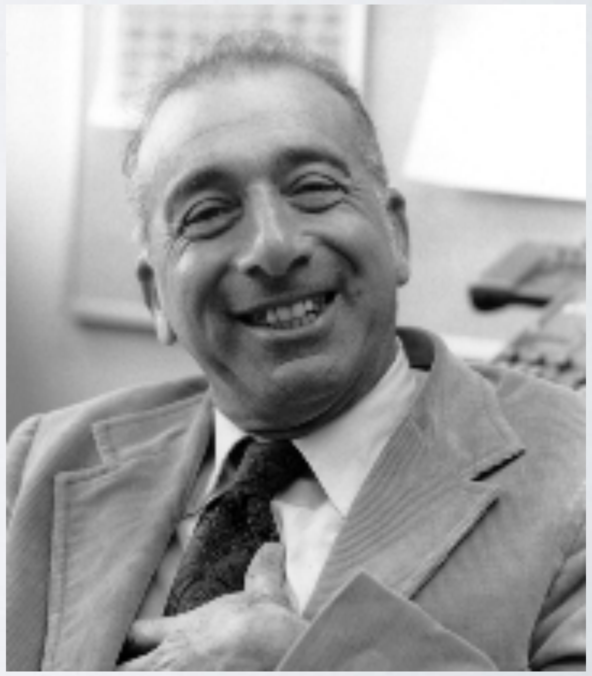
II. History

First MD simulation

- Alder & Wainwright (1957) invent molecular dynamics and perform first simulations of the hard sphere fluid

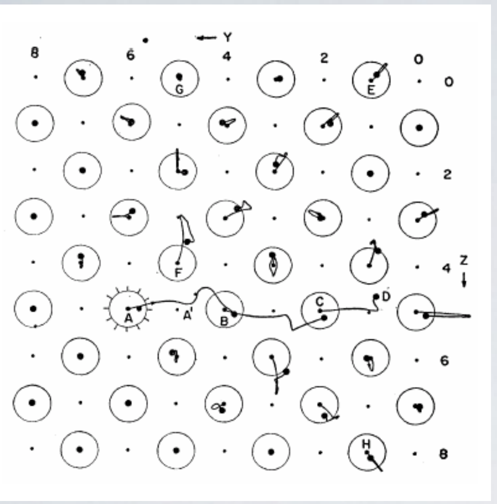


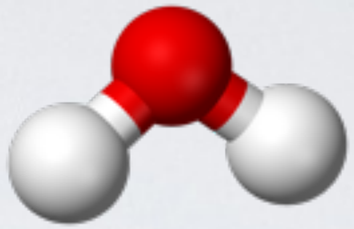
- Berni Alder receives Boltzmann Medal (2001) and National Medal of Science (2009) for this work



- Currently Professor Emeritus at UC Davis

Milestones in MD





1960
Gibson *et al.*
Simulation of Cu radiation damage
Gibson, J.B., Goland, A.N., Milgram, M., and Vineyard, G.H. Phys. Rev. 120 1229 (1960)

1974
Rahman & Stillinger
First simulation of liquid water
Stillinger, F.H. and Rahman, A. J. Chem. Phys. 60 1545 (1974)

1994
York *et al.*
BPTI hydrated xtal [1ns]
York, D.M., Wlodawer, A., Pedersen, L.G. and Darden, T.A. PNAS 91 18 8715 (1994)

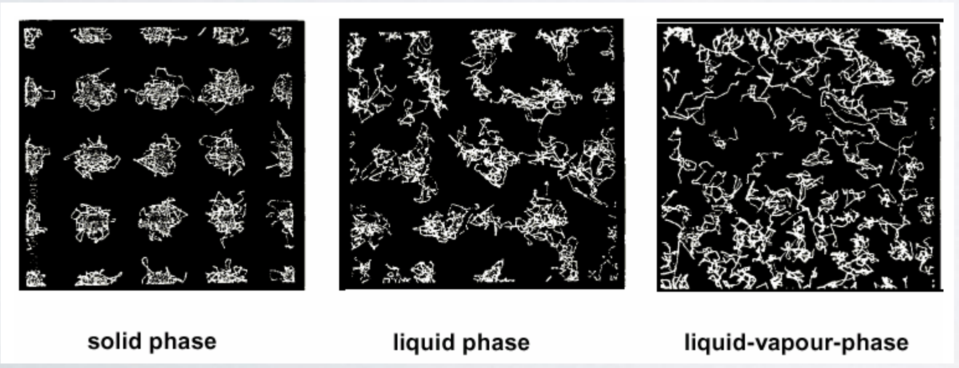
2010
Shaw *et al.*
BPTI in water [1ms]
Shaw, D.E. et al. Science 330 341 (2010)

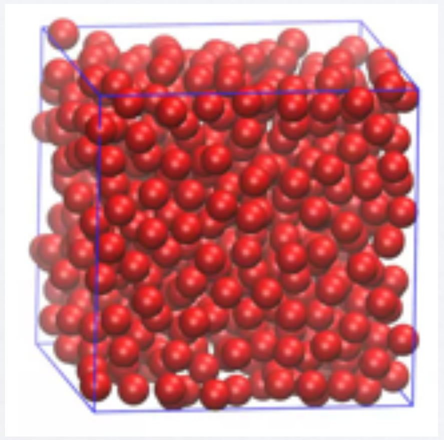
1957
Alder & Wainwright
First MD simulation of hard sphere fluid
Alder, B.J. and Wainwright, T.E. J. Chem. Phys. 27 1208 (1957)

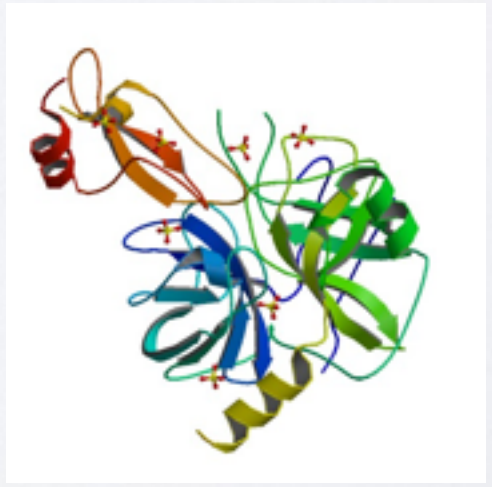
1964
Rahman
First simulation of liquid Ar using realistic potential
Rahman, A. Phys. Rev. A136 405 (1964)

1977
McCammon *et al.*
First protein simulation (BPTI) [8.8ps]
McCammon, J.A., Gelin, B.R., and Karplus, M. Nature 267 585 (1977)

1998
Duan & Kollman
Villin headpiece in water [1 μs]
Duan, Y., and Kollman, P.A. Science 282 5389 740 (1998)







III. Basic Principles

The fundamental idea

- MD simulates atomic motions using classical mechanics
- Running a simulation is like cooking - just follow the recipe!
- Three ingredients:

  1. An initial system configuration $[\vec{r}(t=0), \vec{v}(t=0)]$
  2. Interaction potentials for system $V(\vec{r})$
  3. A way to integrate F=ma

The fundamental idea

- Laplace's Demon / "The Clockwork Universe"

"Given for one instant an intelligence which could comprehend all the forces by which nature is animated and the respective positions of the beings which compose it, if moreover this intelligence were vast enough to submit these data to analysis, it would embrace in the same formula both the movements of the largest bodies in the universe and those of the lightest atom; to it **nothing would be uncertain, and the future as the past would be present to its eyes**."

- Pierre Simon de Laplace (1749-1827)

This is basically molecular dynamics!

But what about quantum effects?

- Classical MD treats atoms* as point particles that move deterministically via Newton's equations of motion

- Is this a valid description of atomic dynamics? **YES.**

(1) Born-Oppenheimer allows us to treat electrons implicitly. Their effect is "baked in" to nuclear interaction potential.

$\tau_{elec} \sim 10^{-18}$ s

$\tau_{nuc} \sim 10^{-15}$ s

Separation of time scales argues for pseudo-equilibrium of electrons with respect to nuclei

* or coarse-grained groups of atoms called "united atoms"

But what about quantum effects?

(2) The Schrödinger equation for nuclei replaced by F=ma

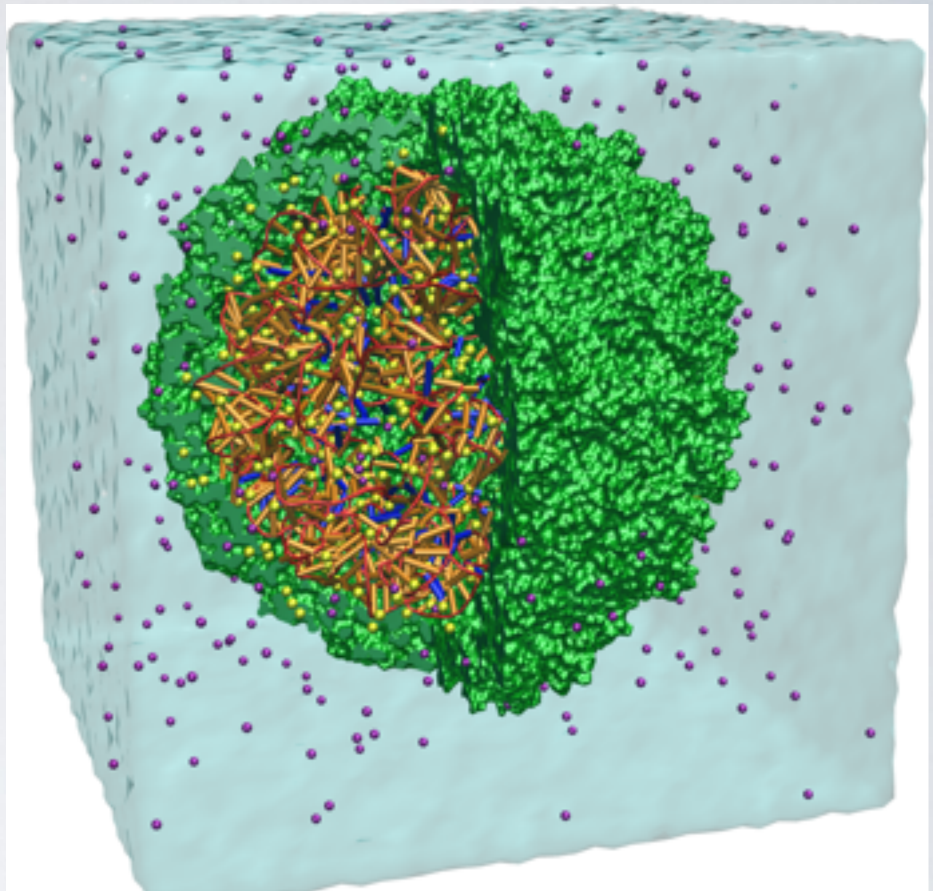
de Broglie wavelength:	$\Lambda_H \sim 1Å$, $\Lambda_C \sim 0.3Å$
characteristic atomic separation:	$d \sim 1Å$

For all but lightest atoms $d >> \Lambda$, allowing us to treat atoms as point particles and use classical mechanics*

**The quantum behavior of light elements (e.g., H, He, Ne) requires special treatment by fixing bond lengths or lumping light atoms into united atoms*

Ingredient 1: Initial configuration

- Specification of initial atomic coordinates and velocities
- Classical mechanics is deterministic: **initial state and interaction rules fully specify the system's future***
- Wind up Laplace's clockwork universe and — in principle — a "vast intelligence" could compute the future of the system
- Our intelligence is insufficiently vast — the equations are hard! — and thus **we resort to numerical simulation**


Theoretical and Computational Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

* neglecting numerical integration errors and finite precision (i.e., uncertainty)

Initializing coordinates

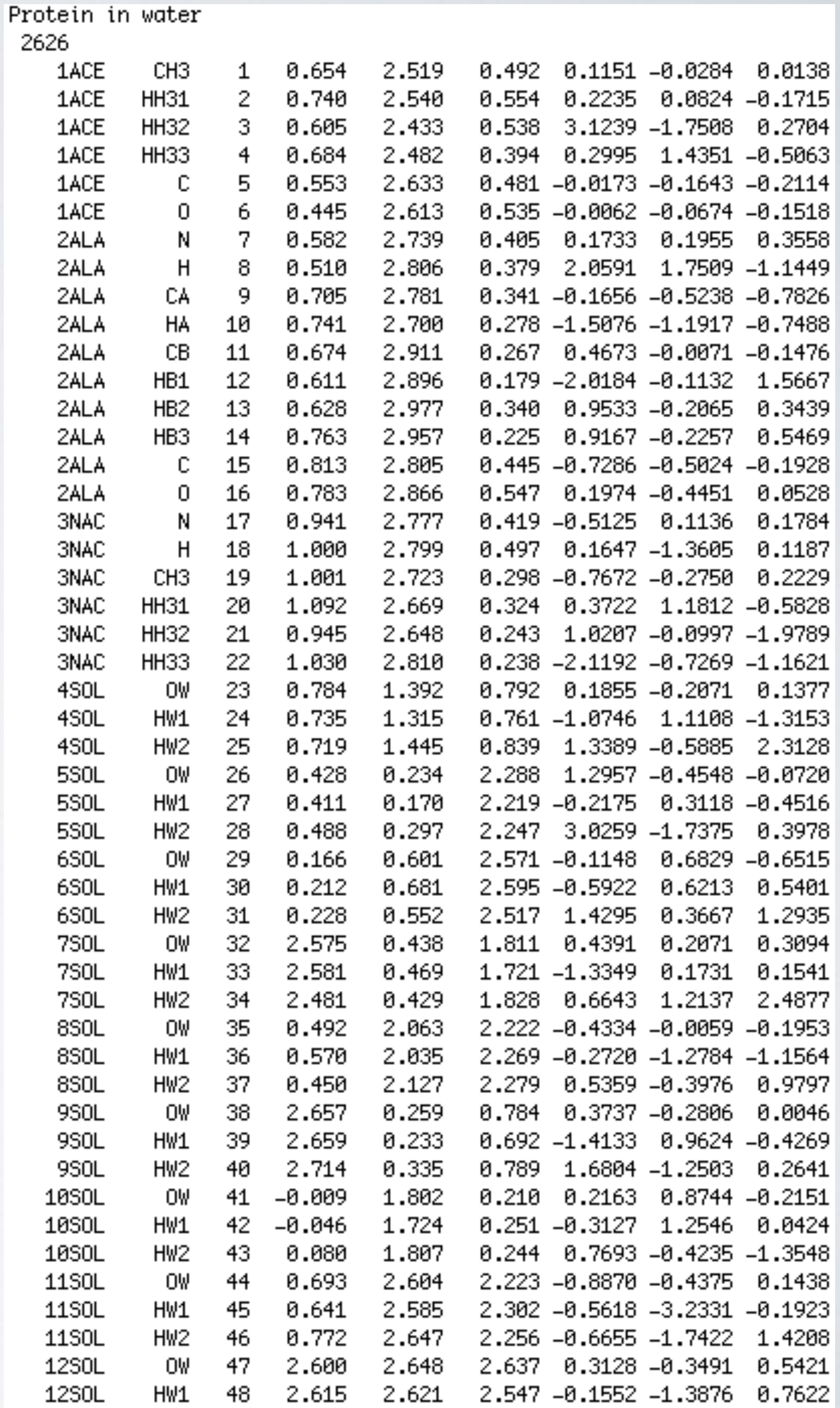
- Initial configurations can be generated "by hand" or short scripts for simple systems (e.g., liquid Ar, bulk Al)

- Software tools for complex systems (e.g., proteins, complex defect structures)

PRODRG (http://davapc1.bioch.dundee.ac.uk/prodrg/)
ATP (http://compbio.biosci.uq.edu.au/atb/)
PyMOl (http://www.pymol.org/)
Chimera (http://www.cgl.ucsf.edu/chimera/)

- Common protein structures are in Protein Data Bank

PDB (www.rcsb.org/pdb)



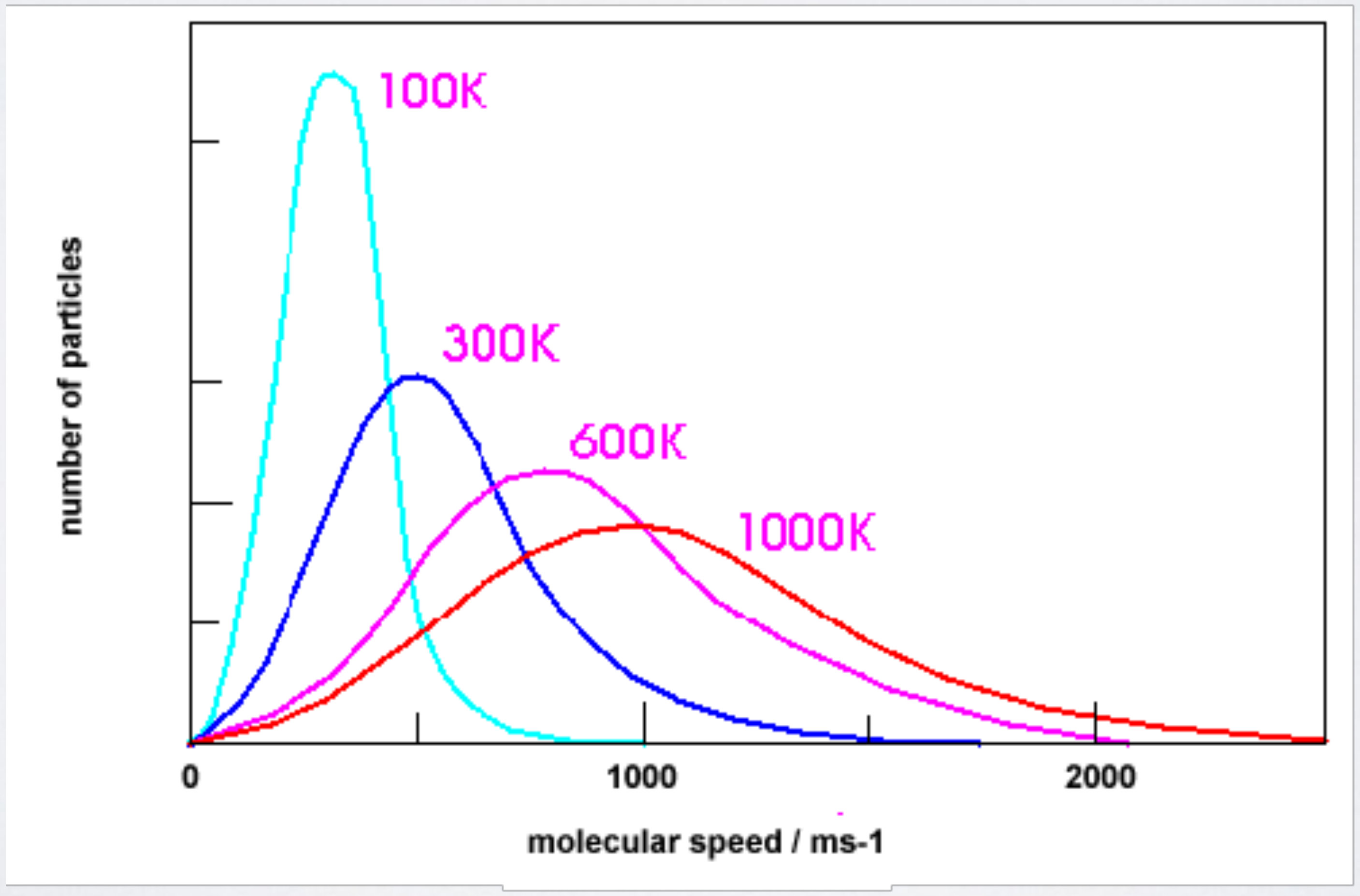
```
Protein in water
 2626
    1ACE    CH3    1   0.654   2.519   0.492  0.1151 -0.0284  0.0138
    1ACE   HH31    2   0.740   2.540   0.554  0.2235  0.0824 -0.1715
    1ACE   HH32    3   0.605   2.433   0.538  3.1239 -1.7508  0.2704
    1ACE   HH33    4   0.684   2.482   0.394  0.2995  1.4351 -0.5063
    1ACE      C    5   0.553   2.633   0.481 -0.0173 -0.1643 -0.2114
    1ACE      O    6   0.445   2.613   0.535 -0.0062 -0.0674 -0.1518
    2ALA      N    7   0.582   2.739   0.405  0.1733  0.1955  0.3558
    2ALA      H    8   0.510   2.806   0.379  2.0591  1.7509 -1.1449
    2ALA     CA    9   0.705   2.781   0.341 -0.1656 -0.5238 -0.7826
    2ALA     HA   10   0.741   2.700   0.278 -1.5076 -1.1917 -0.7488
    2ALA     CB   11   0.674   2.911   0.267  0.4673 -0.0071 -0.1476
    2ALA    HB1   12   0.611   2.896   0.179 -2.0184 -0.1132  1.5667
    2ALA    HB2   13   0.628   2.977   0.340  0.9533 -0.2065  0.3439
    2ALA    HB3   14   0.763   2.957   0.225  0.9167 -0.2257  0.5469
    2ALA      C   15   0.813   2.805   0.445 -0.7286 -0.5024 -0.1928
    2ALA      O   16   0.783   2.866   0.547  0.1974 -0.4451  0.0528
    3NAC      N   17   0.941   2.777   0.419 -0.5125  0.1136  0.1784
    3NAC      H   18   1.000   2.799   0.497  0.1647 -1.3605  0.1187
    3NAC    CH3   19   1.001   2.723   0.298 -0.7672 -0.2750  0.2229
    3NAC   HH31   20   1.092   2.669   0.324  0.3722  1.1812 -0.5828
    3NAC   HH32   21   0.945   2.648   0.243  1.0207 -0.0997 -1.9789
    3NAC   HH33   22   1.030   2.810   0.238 -2.1192 -0.7269 -1.1621
    4SOL     OW   23   0.784   1.392   0.792  0.1855 -0.2071  0.1377
    4SOL    HW1   24   0.735   1.315   0.761 -1.0746  1.1108 -1.3153
    4SOL    HW2   25   0.719   1.445   0.839  1.3389 -0.5885  2.3128
    5SOL     OW   26   0.428   0.234   2.288  1.2957 -0.4548 -0.0720
    5SOL    HW1   27   0.411   0.170   2.219 -0.2175  0.3118 -0.4516
    5SOL    HW2   28   0.488   0.297   2.247  3.0259 -1.7375  0.3978
    6SOL     OW   29   0.166   0.601   2.571 -0.1148  0.6829 -0.6515
    6SOL    HW1   30   0.212   0.681   2.595 -0.5922  0.6213  0.5401
    6SOL    HW2   31   0.228   0.552   2.517  1.4295  0.3667  1.2935
    7SOL     OW   32   2.575   0.438   1.811  0.4391  0.2071  0.3094
    7SOL    HW1   33   2.581   0.469   1.721 -1.3349  0.1731  0.1541
    7SOL    HW2   34   2.481   0.429   1.828  0.6643  1.2137  2.4877
    8SOL     OW   35   0.492   2.063   2.222 -0.4334 -0.0059 -0.1953
    8SOL    HW1   36   0.570   2.035   2.269 -0.2720 -1.2784 -1.1564
    8SOL    HW2   37   0.450   2.127   2.279  0.5359 -0.3976  0.9797
    9SOL     OW   38   2.657   0.259   0.784  0.3737 -0.2806  0.0046
    9SOL    HW1   39   2.659   0.233   0.692 -1.4133  0.9624 -0.4269
    9SOL    HW2   40   2.714   0.335   0.789  1.6804 -1.2503  0.2641
   10SOL     OW   41  -0.009   1.802   0.210  0.2163  0.8744 -0.2151
   10SOL    HW1   42  -0.046   1.724   0.251 -0.3127  1.2546  0.0424
   10SOL    HW2   43   0.080   1.807   0.244  0.7693 -0.4235 -1.3548
   11SOL     OW   44   0.693   2.604   2.223 -0.8870 -0.4375  0.1438
   11SOL    HW1   45   0.641   2.585   2.302 -0.5618 -3.2331 -0.1923
   11SOL    HW2   46   0.772   2.647   2.256 -0.6655 -1.7422  1.4208
   12SOL     OW   47   2.600   2.648   2.637  0.3128 -0.3491  0.5421
   12SOL    HW1   48   2.615   2.621   2.547 -0.1552 -1.3876  0.7622
```

Initializing velocities

- Bad idea to start atoms from rest (absolute zero = 0 K) due to thermal shock upon starting simulation

- Standard approach is to draw velocities randomly from a Maxwell-Boltzmann distribution at the temperature, T

$$f_{\mathbf{v}}(v_x, v_y, v_z) = \left(\frac{m}{2\pi kT}\right)^{3/2} \exp\left[-\frac{m(v_x^2 + v_y^2 + v_z^2)}{2kT}\right]$$



http://ibchem.com/IB/ibfiles/states/sta_img/MB2.gif

Ingredient 2: Interaction potentials

- The net force acting on each atom in the system is a result of its interactions with all other atoms
- These interaction amount to a set of rules known as a **force field** or **interaction potential**
- Accurate, robust, and transferable force fields are critical to perform physically realistic molecular simulations
- Force field development is an academic industry

metals:	EAM (Daw & Baskes), MEAM (Baskes)
biomolecules:	Amber (Kollman, UCSF), GROMOS (U. Groningen), CHARMM (Karplus, Harvard), OPLS (Jorgensen, Yale), MARTINI [coarse grained] (Marrink, U. Groningen)
n-alkanes:	TraPPE (Siepmann, U. Minnesota), MM2 (Allinger, UGA)
water:	SPC (Berendsen), SPC/E (Berendsen), TIPnP(Jorgensen), ST2 (Stillinger & Rahman)
general:	DREIDING (Mayo et al.), DISCOVER(Rappe et al.), UFF (Hagler et al.)

Energy, force, and acceleration

- The potential energy of the system is a complicated function of atomic coordinates (this is why we have to *simulate numerically* rather than *calculate analytically*)

- The net force on atom i is the negative gradient of the potential energy wrt the atomic coordinates

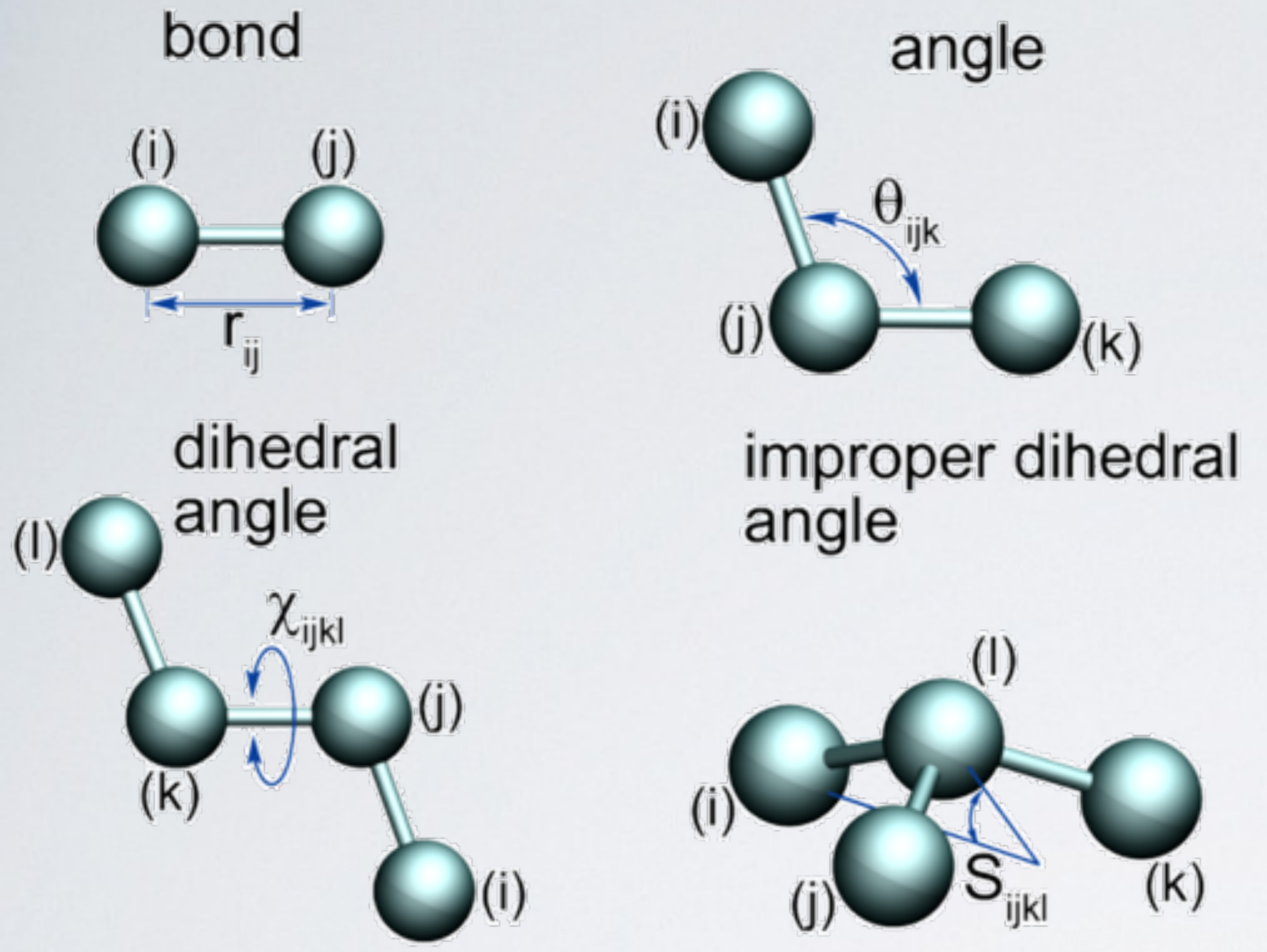
$$F_i = -\nabla_i [V(r_1, r_2, ..., r_N)]$$

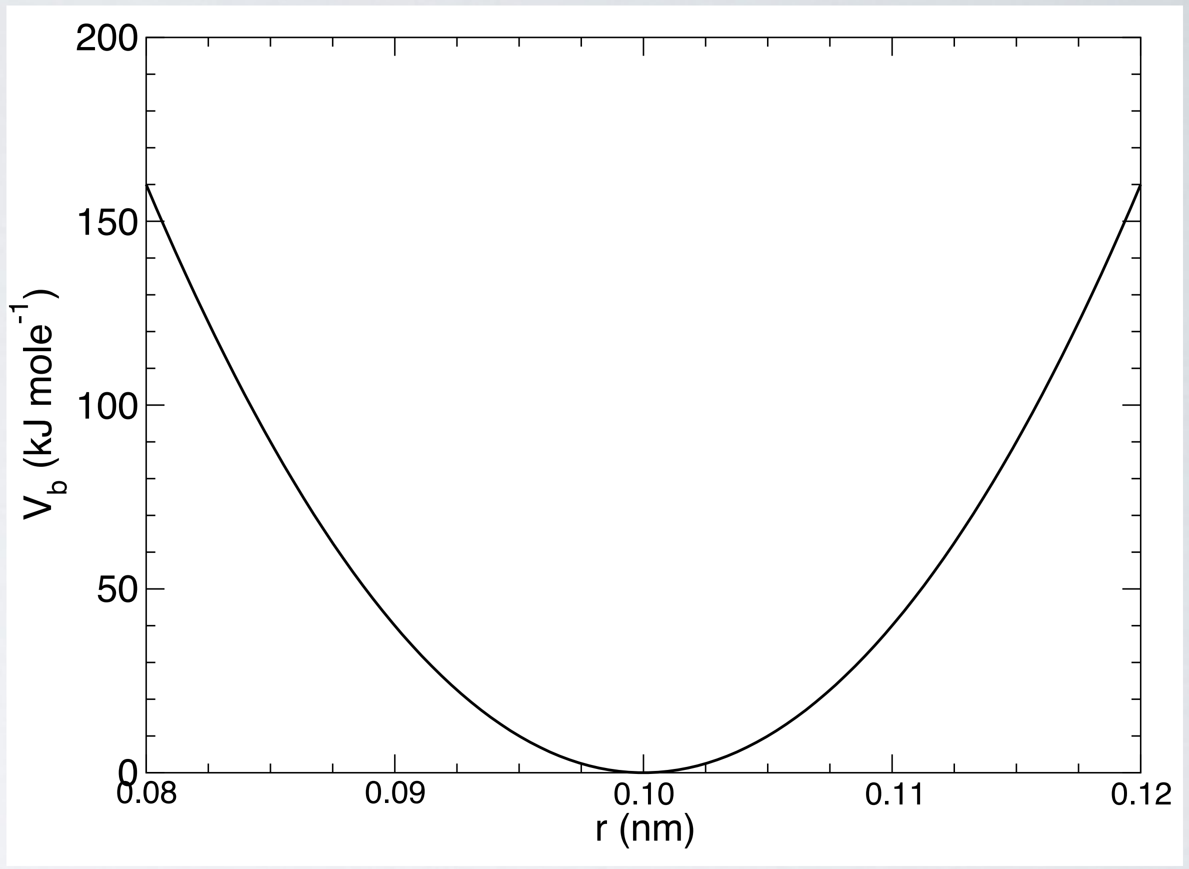
$$a_i = \frac{F_i}{m_i}$$

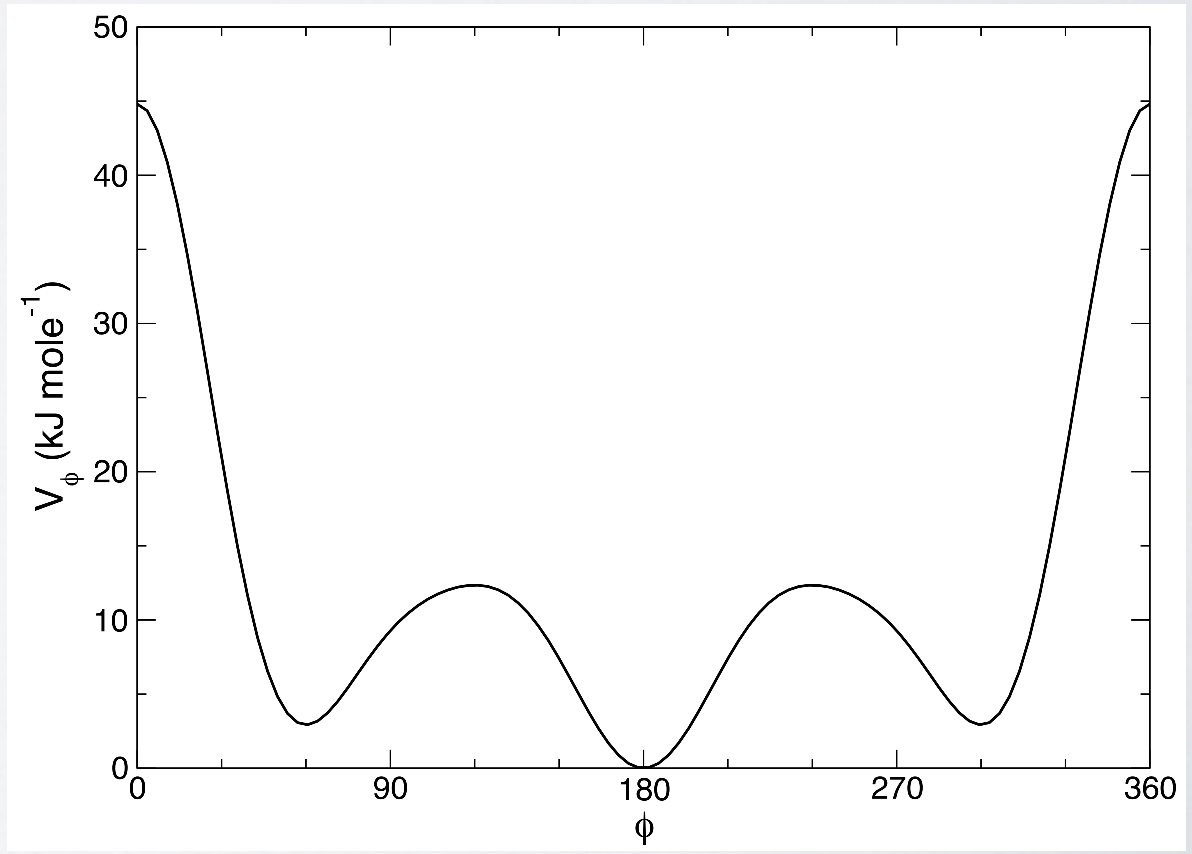
- The potential energy is typically broken into four parts:

$$V(\vec{r}) = V_{bonded} + V_{non-bonded} + V_{restraints} + V_{field}$$

Bonded







$$V_b\left(r_{ij}\right) = \frac{1}{2}k^b_{ij}(r_{ij} - b_{ij})^2$$

$$V_a(\theta_{ijk}) = \frac{1}{2}k^\theta_{ijk}(\theta_{ijk} - \theta^0_{ijk})^2$$

$$V_{rb}(\phi_{ijkl}) = \sum_{n=0}^{5} C_n(\cos(\psi))^n$$

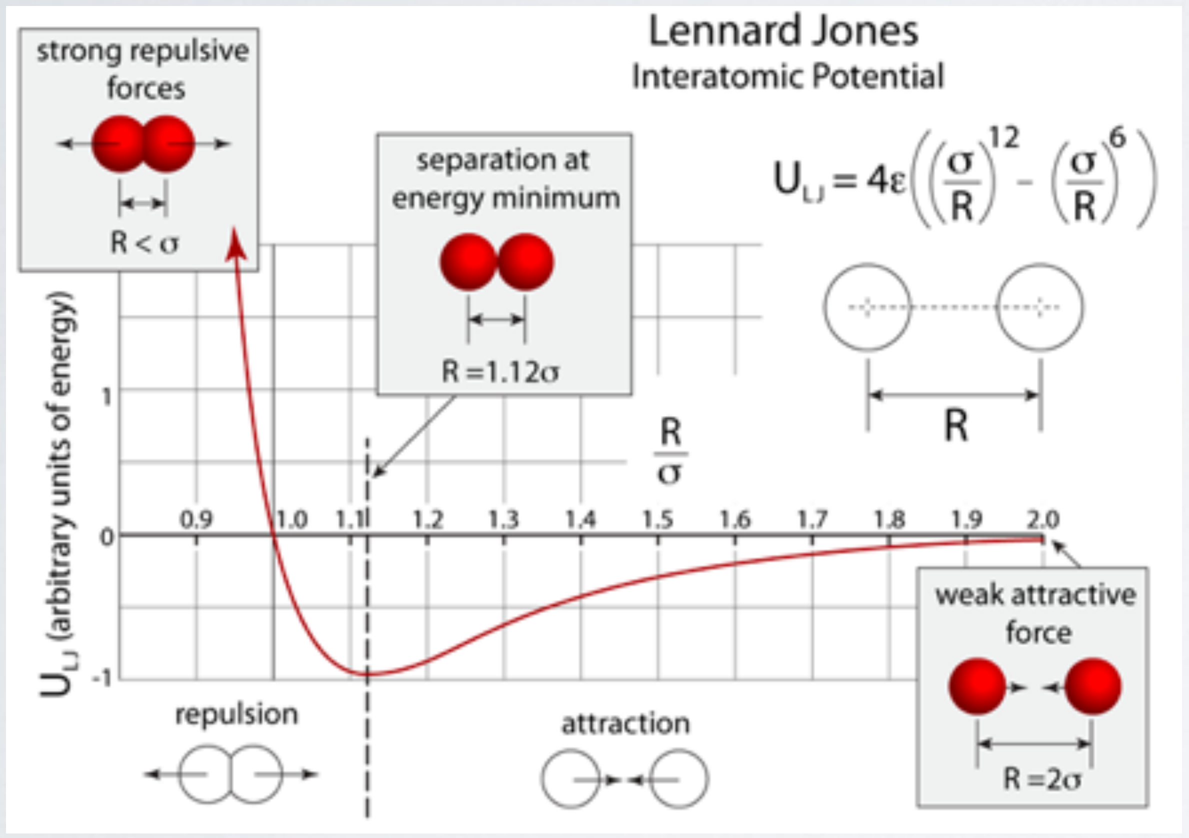
$$V_{id}(\xi_{ijkl}) = \frac{1}{2}k_\xi(\xi_{ijkl} - \xi_0)^2$$

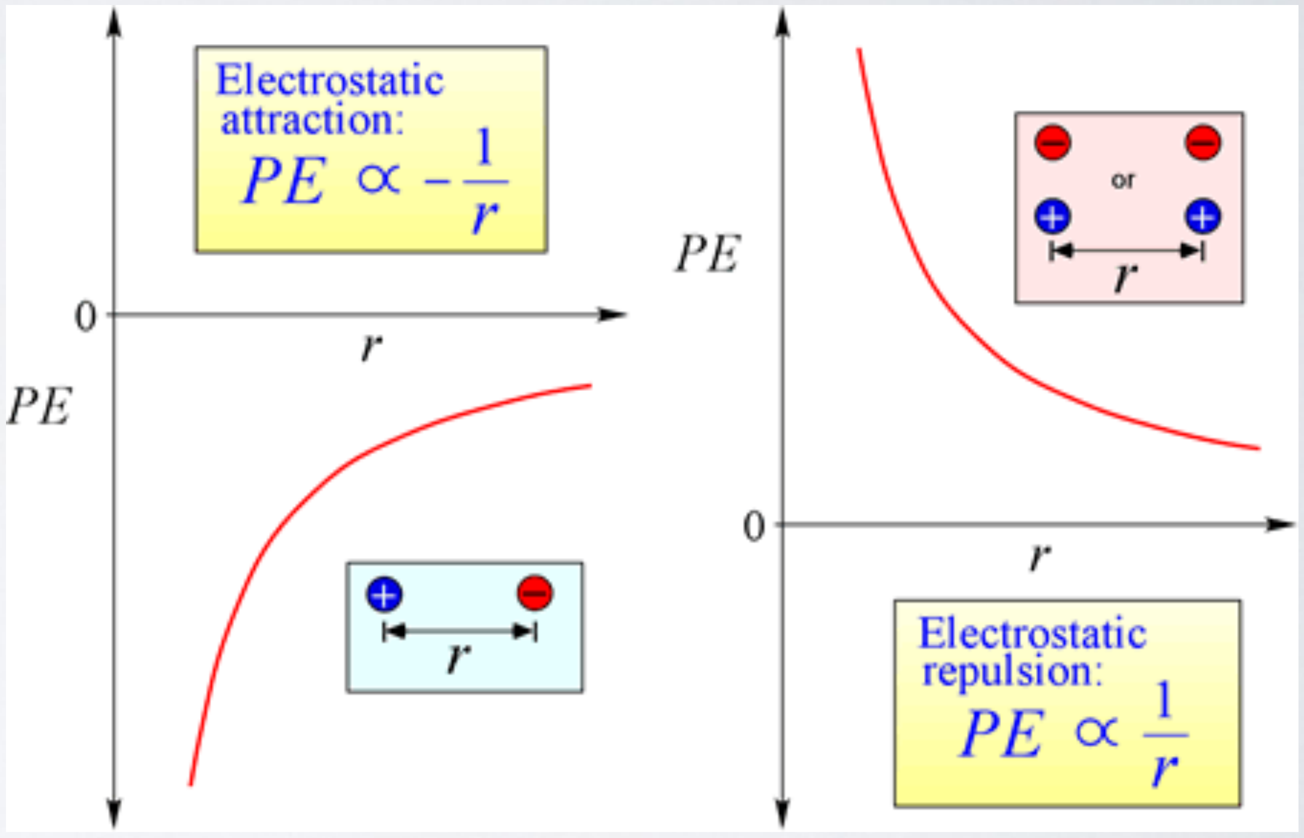
http://www.mbnexplorer.com/users_guide/users_guide743x.png Hess et al. GROMACS Manual v. 4.5.4 (2010) 22

Non-bonded

- Approximate full *n*-body interactions as pairwise additive for simplicity and computational efficiency (cf. (M)EAM)

- van der Waals
- Coulomb





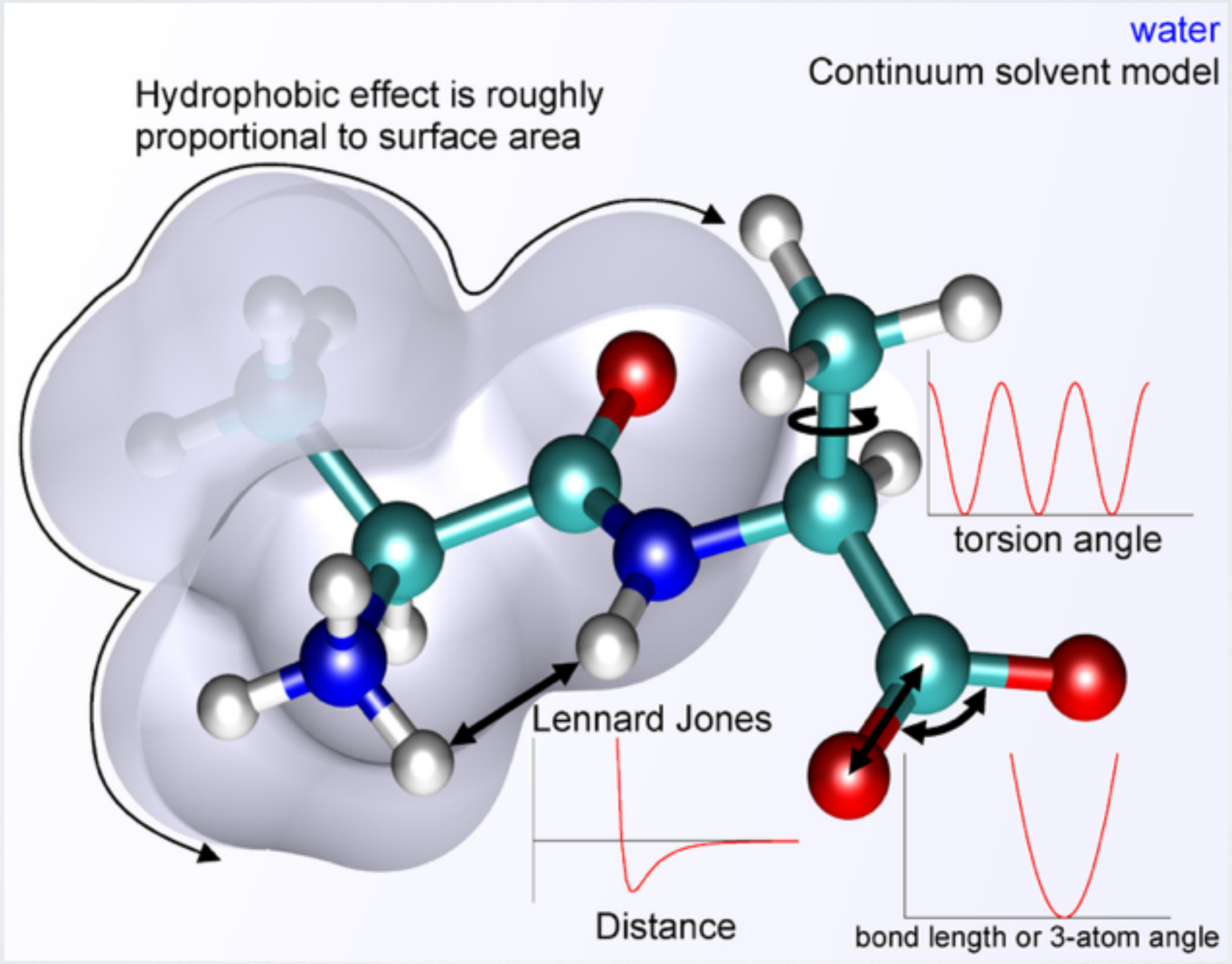
$$V_{LJ}(r_{ij}) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]$$

$$V_{Coul}(r_{ij}) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}}$$

http://atomsinmotion.com/book/chapter5/md http://guweb2.gonzaga.edu/faculty/cronk/chemistry/images/graph-electrostatic-PE-alt.gif

Fields

- Fields are commonly used to model:
  1. external potentials (e.g., electric, magnetic, flow)
  2. continuum solvation (no explicit solvent molecules)


water
Continuum solvent model
Hydrophobic effect is roughly proportional to surface area
torsion angle
Lennard Jones
Distance
bond length or 3-atom angle

EAM / MEAM

- Multi-body potential widely used for metallic solids

  EAM - **E**mbedded **A**tom **M**odel

  MEAM - **M**odified **E**mbedded **A**tom **M**odel

- Inherently many-body ⇒ slower than pairwise additive FF

  (2x - EAM, 3-5x - MEAM)

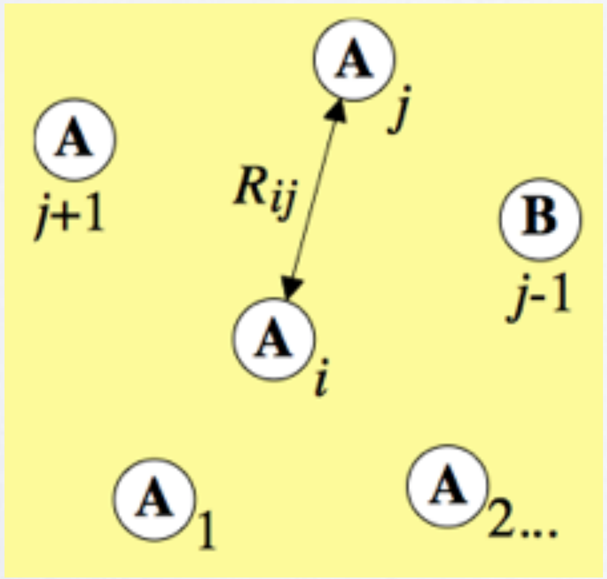
$$E_{\text{total}} = \frac{1}{2}\sum_{i,j}^{N} \Phi_{ij}\left(r_{ij}\right) + \sum_{i}^{N} F_i\left(n_i\right)$$

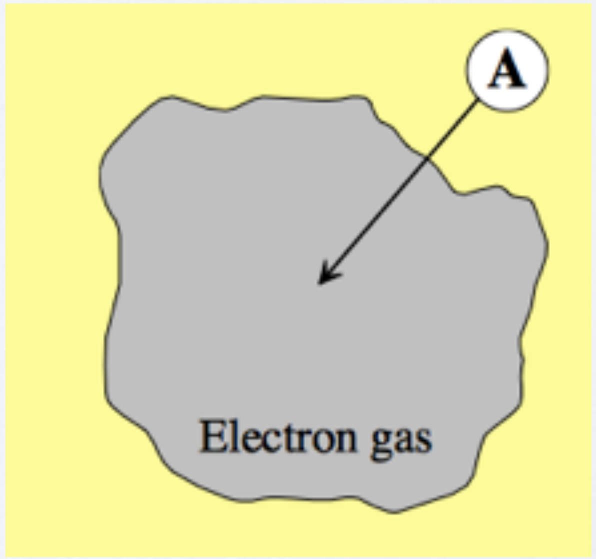
pairwise potential

interatomic separation

local e⁻ density

embedding function





EAM / MEAM

- Local e⁻ density functions

EAM

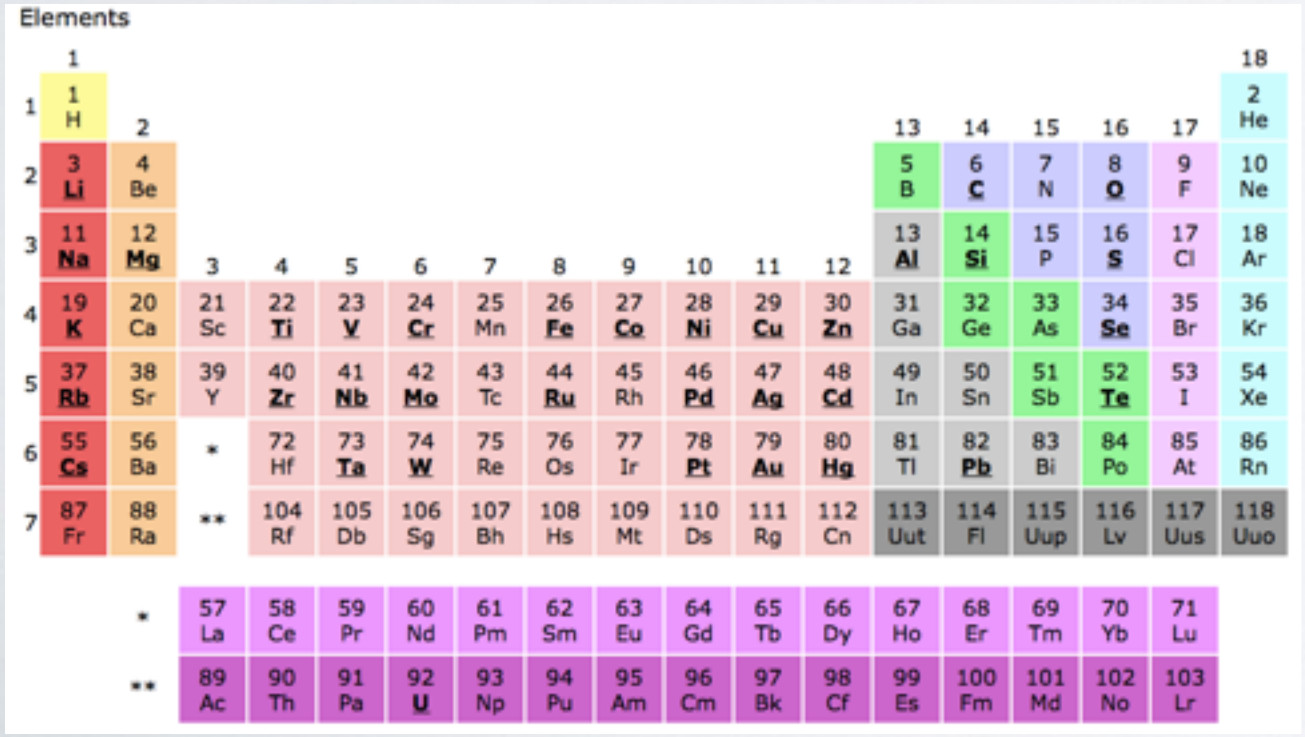
$$n_i = \sum_{j \neq i} \rho_j(r_{ij})$$

MEAM

$$n_i = \sum_{j \neq i}^{N} \rho_j\left(r_{ij}\right) + \frac{1}{2} \sum_{j,k \neq i}^{N} \underbrace{f_{ij}\left(r_{ij}\right) f_{ik}\left(r_{ik}\right)}_{\text{3-body radial}} \underbrace{g_i\left(\cos\theta_{jik}\right)}_{\text{3-body angular}}$$

➔ 3-body term in MEAM improves agreement for directional bonding (bcc, hcp, diamond)

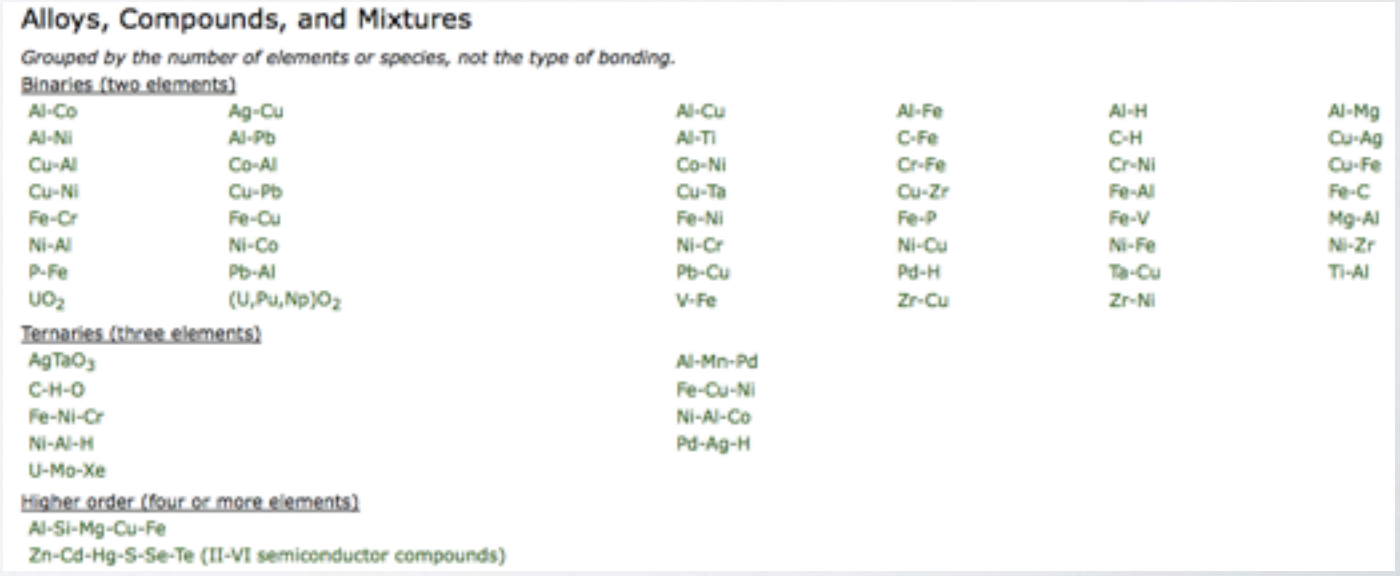
- NIST Interatomic Potentials Repository



Elements

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1 H																	2 He
2	3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
3	11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
4	19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
5	37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
6	55 Cs	56 Ba	*	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
7	87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt	110 Ds	111 Rg	112 Cn	113 Uut	114 Fl	115 Uup	116 Lv	117 Uus	118 Uuo
		*	57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu	
		**	89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr	

http://www.ctcms.nist.gov/potentials/



Alloys, Compounds, and Mixtures

Grouped by the number of elements or species, not the type of bonding.

Binaries (two elements)

Al-Co	Ag-Cu	Al-Cu	Al-Fe	Al-H	Al-Mg
Al-Ni	Al-Pb	Al-Ti	C-Fe	C-H	Cu-Ag
Cu-Al	Co-Al	Co-Ni	Cr-Fe	Cr-Ni	Cu-Fe
Cu-Ni	Cu-Pb	Cu-Ta	Cu-Zr	Fe-Al	Fe-C
Fe-Cr	Fe-Cu	Fe-Ni	Fe-P	Fe-V	Mg-Al
Ni-Al	Ni-Co	Ni-Cr	Ni-Cu	Ni-Fe	Ni-Zr
P-Fe	Pb-Al	Pb-Cu	Pd-H	Ta-Cu	Ti-Al
UO_2	$(U,Pu,Np)O_2$	V-Fe	Zr-Cu	Zr-Ni	

Ternaries (three elements)

$AgTaO_3$	Al-Mn-Pd
C-H-O	Fe-Cu-Ni
Fe-Ni-Cr	Ni-Al-Co
Ni-Al-H	Pd-Ag-H
U-Mo-Xe	

Higher order (four or more elements)

Al-Si-Mg-Cu-Fe
Zn-Cd-Hg-S-Se-Te (II-VI semiconductor compounds)

Ingredient 3: Integrators

- [initial atomic coordinates and velocities] + [force field]
  ⇒ entire future (and past!) modeled by **F=ma**
- Analytical solutions for the dynamical evolution cannot be computed for all but the simplest systems (>2 body)
- Solve Newton's equations by numerical integration
  ⇒ computers ideally suited to rapid, repetitive calculations
- Solving by hand would require thousands of years!

Verlet algorithm

- Many possible integration algorithms exist
  (e.g., explicit/implicit Euler, Gear predictor-corrector, nth order Runge-Kutta, Beeman, Newmark-beta)

- The method of choice is the **Verlet algorithm**

  ✓ **fast**
  ✓ **simple**
  ✓ **low-memory**
  ✓ **stable**
  ✓ **time-reversible**
  ✓ **symplectic (phase space volume & E conserving)**

  ✗ **poor accuracy for large time steps (Δt must be small)**

- First recorded use by Delambre in 1791
  Popularized in MD by Loup Verlet in 1967

Verlet algorithm

- Derived from Taylor series:

$$r(t+\delta t) = r(t) + \dot{r}(t)\delta t + \frac{1}{2}\ddot{r}(t)\delta t^2 + \ldots$$
$$= r(t) + v(t)\delta t + \frac{1}{2}a(t)\delta t^2 + \ldots$$

$$r(t-\delta t) = r(t) - \dot{r}(t)\delta t + \frac{1}{2}\ddot{r}(t)\delta t^2 + \ldots$$
$$= r(t) - v(t)\delta t + \frac{1}{2}a(t)\delta t^2 + \ldots$$

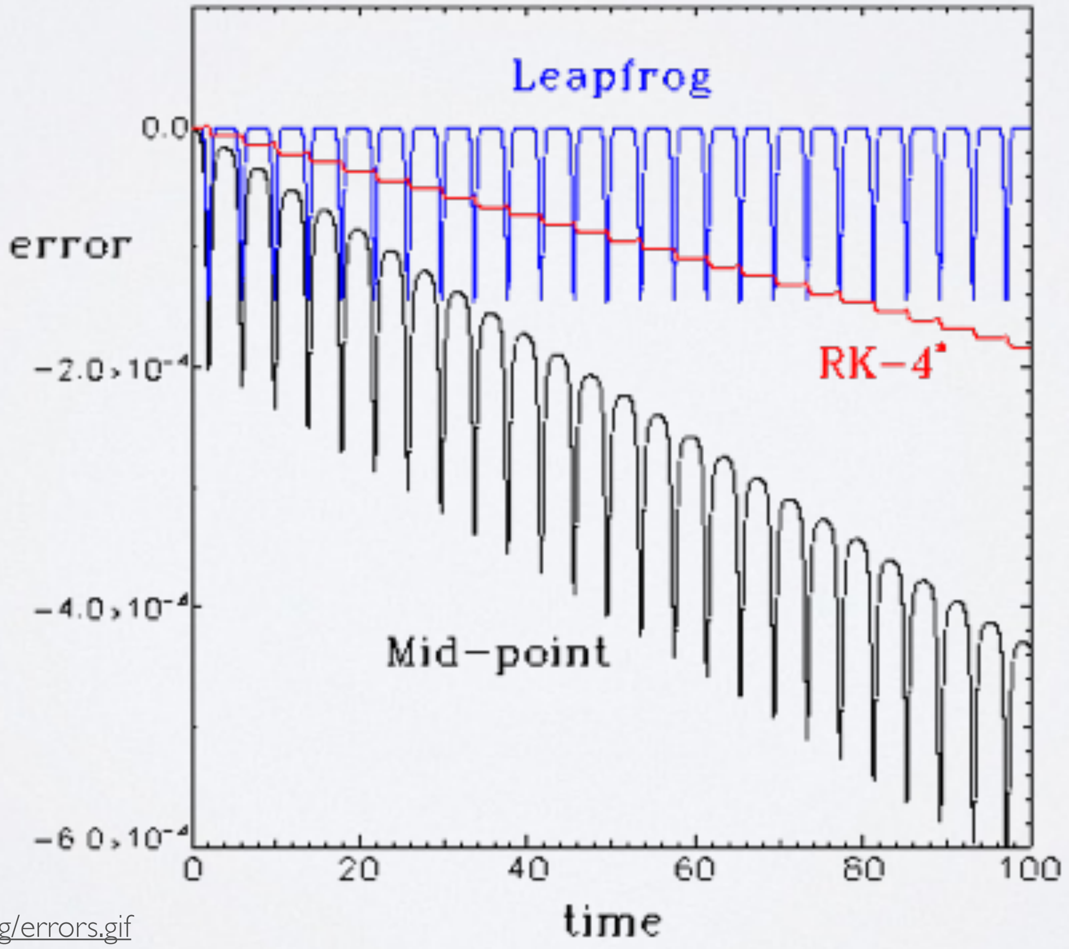
$$\boxed{\begin{aligned} r(t+\delta t) &= 2r(t) - r(t-\delta t) + a(t)\delta t^2 + \mathcal{O}\left(\delta t^4\right) \\ v(t) &= \frac{r(t+\delta t) - r(t-\delta t)}{2\delta t} + \mathcal{O}\left(\delta t^2\right) \end{aligned}}$$

$$a_i = \frac{F_i}{m_i}$$

Time-reversibility

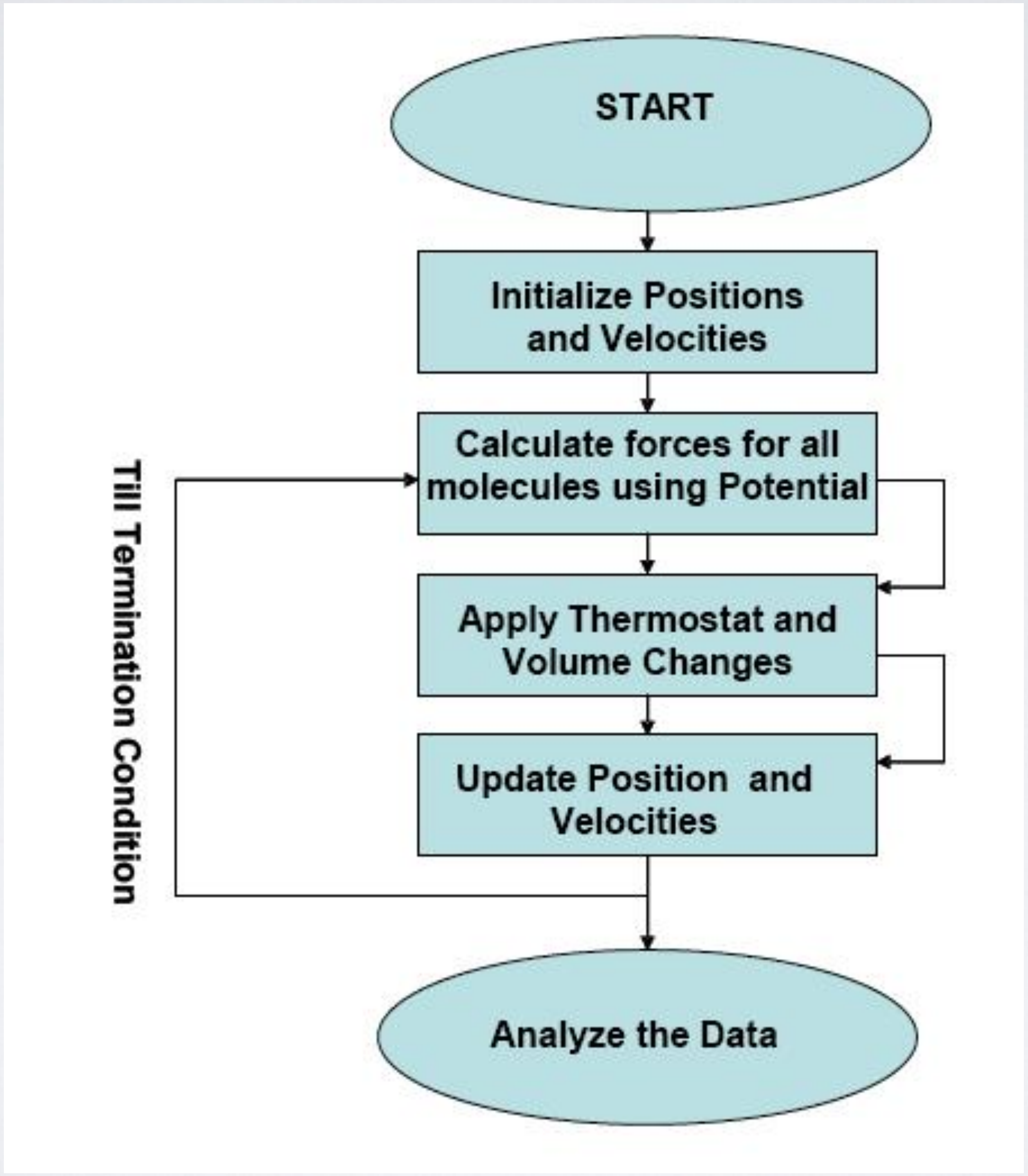
- Higher order integration algorithms have higher per step accuracy, enabling longer time steps and faster simulations (e.g., Runge-Kutta, Gear predictor-corrector)

- **But**, do not respect time reversibility of Newton's equations causing energy drift and error accumulation



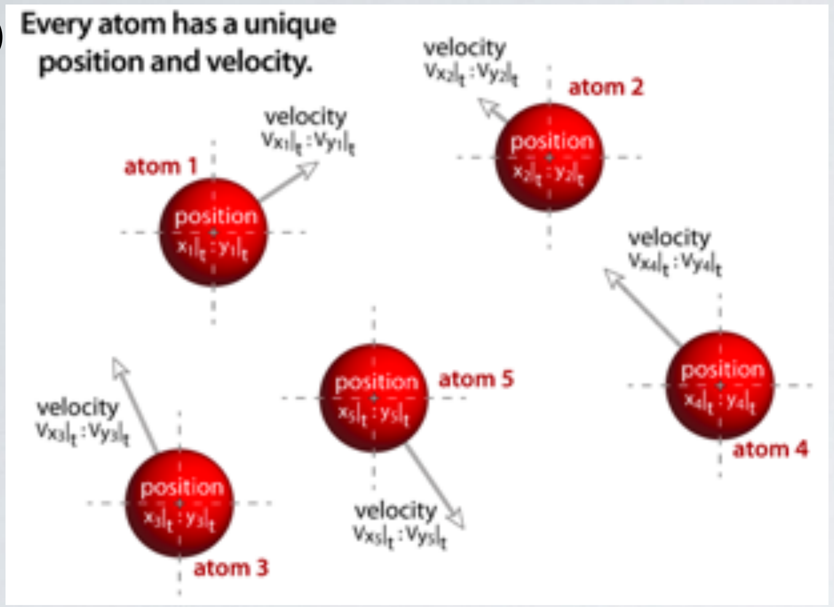
http://einstein.drexel.edu/courses/Comp_Phys/Integrators/leapfrog/errors.gif

Simulation Overview


START
Initialize Positions and Velocities
Calculate forces for all molecules using Potential
Apply Thermostat and Volume Changes
Update Position and Velocities
Till Termination Condition
Analyze the Data

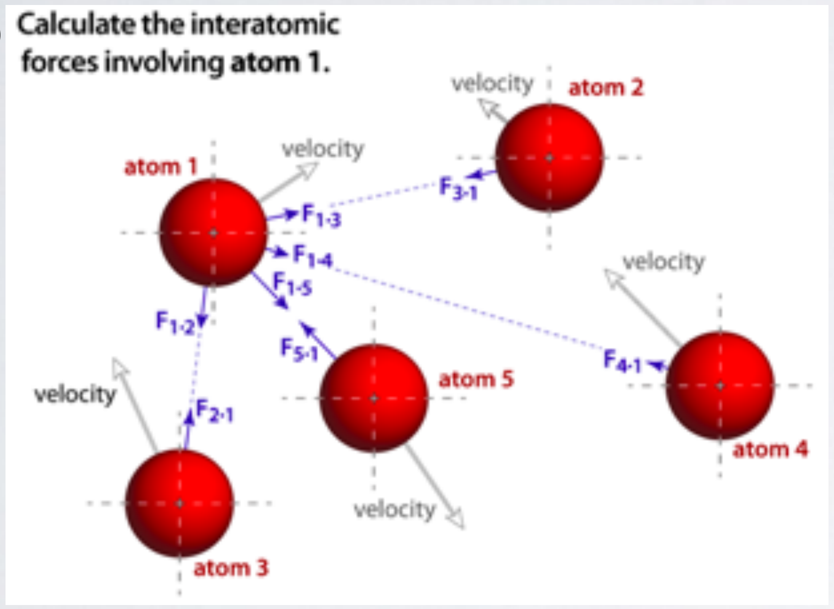
Simulation Overview

❶ Every atom has a unique position and velocity.

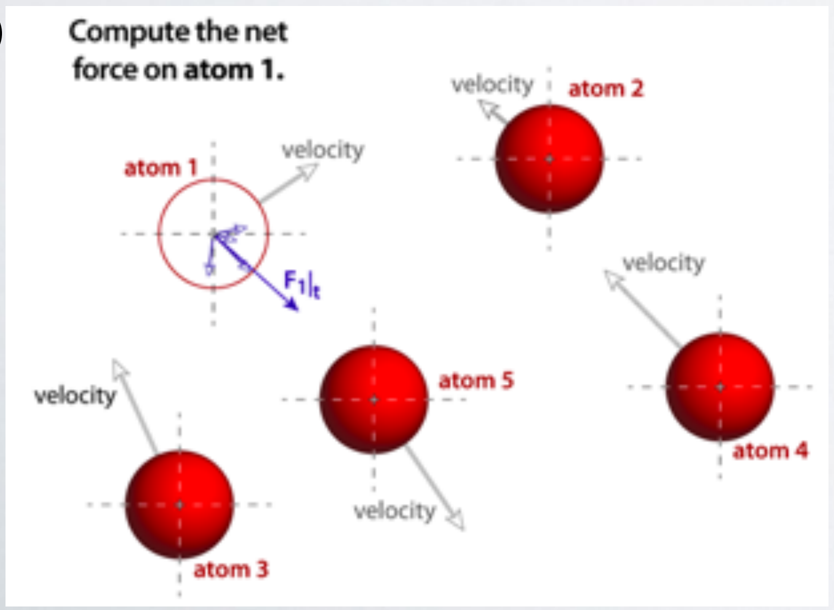


atom 1
position $x_1|_t : y_1|_t$
velocity $V_{x1}|_t : V_{y1}|_t$
atom 2
position $x_2|_t : y_2|_t$
velocity $V_{x2}|_t : V_{y2}|_t$
atom 3
position $x_3|_t : y_3|_t$
velocity $V_{x3}|_t : V_{y3}|_t$
atom 4
position $x_4|_t : y_4|_t$
velocity $V_{x4}|_t : V_{y4}|_t$
atom 5
position $x_5|_t : y_5|_t$
velocity $V_{x5}|_t : V_{y5}|_t$

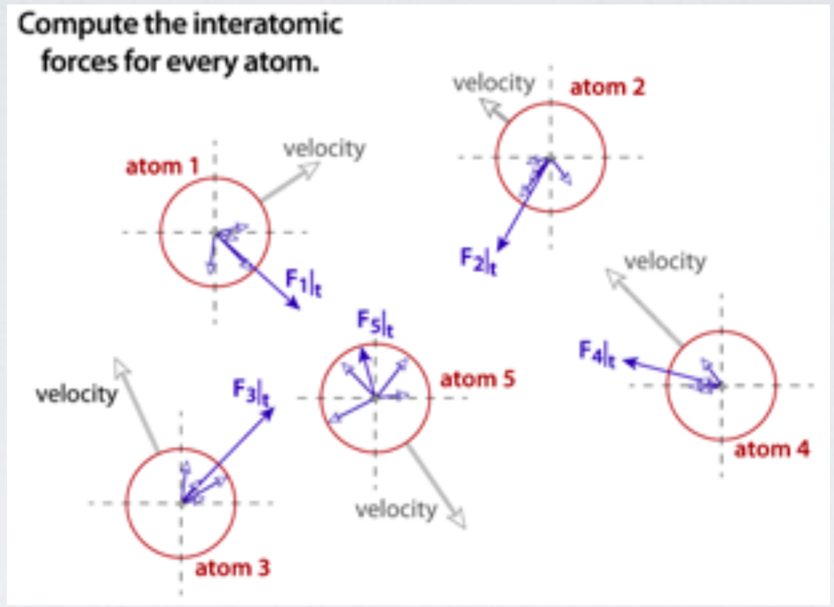
❷ Calculate the interatomic forces involving atom 1.



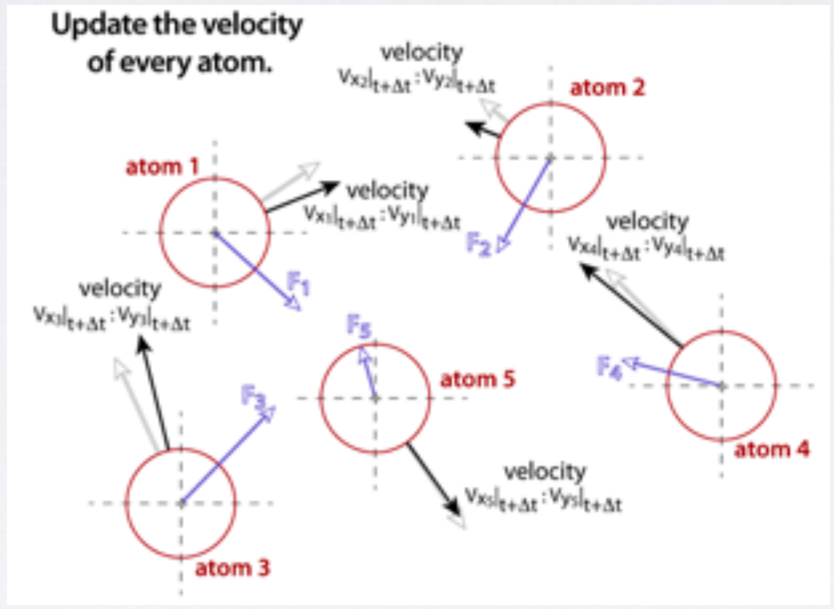
❸ Compute the net force on atom 1.



❹ Compute the interatomic forces for every atom.

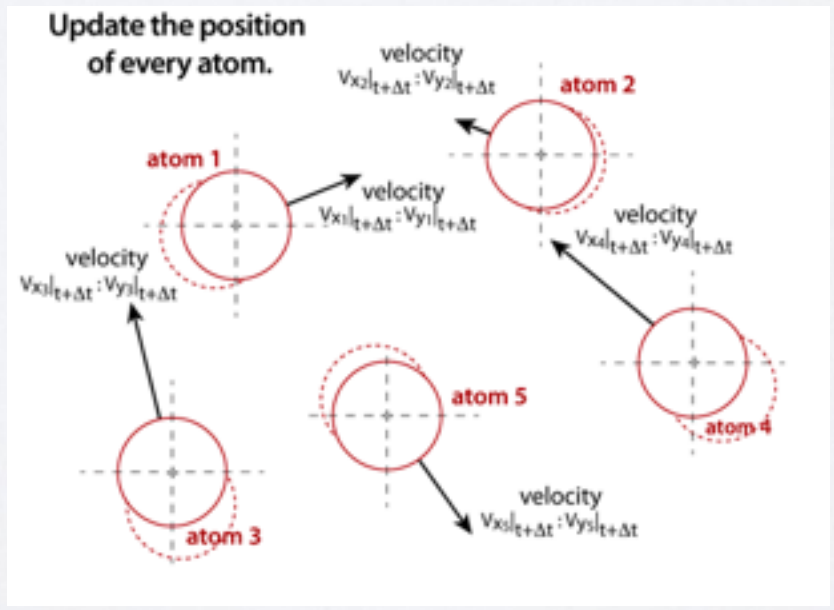


❺ Update the velocity of every atom.



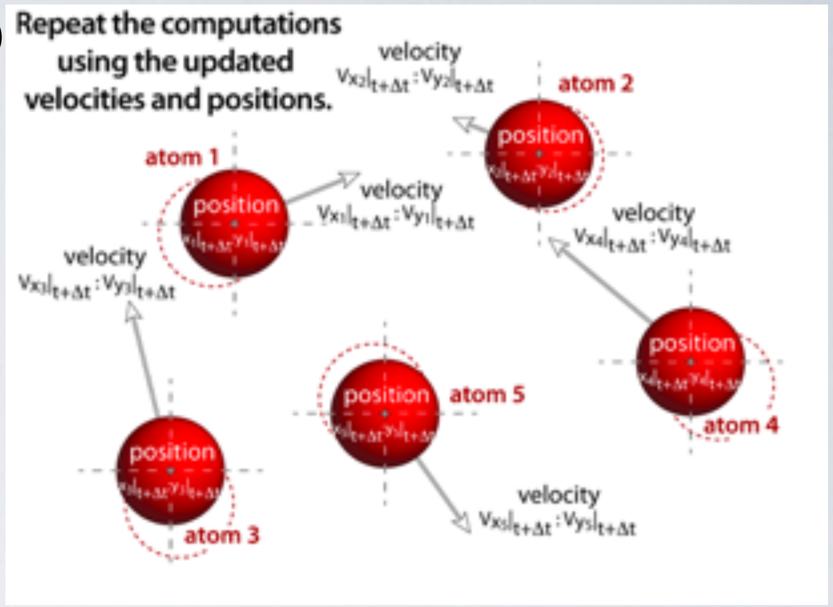
atom 1
atom 2
atom 3
atom 4
atom 5
velocity $V_{x1}|_{t+\Delta t} : V_{y1}|_{t+\Delta t}$
velocity $V_{x2}|_{t+\Delta t} : V_{y2}|_{t+\Delta t}$
velocity $V_{x3}|_{t+\Delta t} : V_{y3}|_{t+\Delta t}$
velocity $V_{x4}|_{t+\Delta t} : V_{y4}|_{t+\Delta t}$
velocity $V_{x5}|_{t+\Delta t} : V_{y5}|_{t+\Delta t}$
F_1
F_2
F_3
F_4
F_5

❻ Update the position of every atom.



atom 1
atom 2
atom 3
atom 4
atom 5
velocity $V_{x1}|_{t+\Delta t} : V_{y1}|_{t+\Delta t}$
velocity $V_{x2}|_{t+\Delta t} : V_{y2}|_{t+\Delta t}$
velocity $V_{x3}|_{t+\Delta t} : V_{y3}|_{t+\Delta t}$
velocity $V_{x4}|_{t+\Delta t} : V_{y4}|_{t+\Delta t}$
velocity $V_{x5}|_{t+\Delta t} : V_{y5}|_{t+\Delta t}$

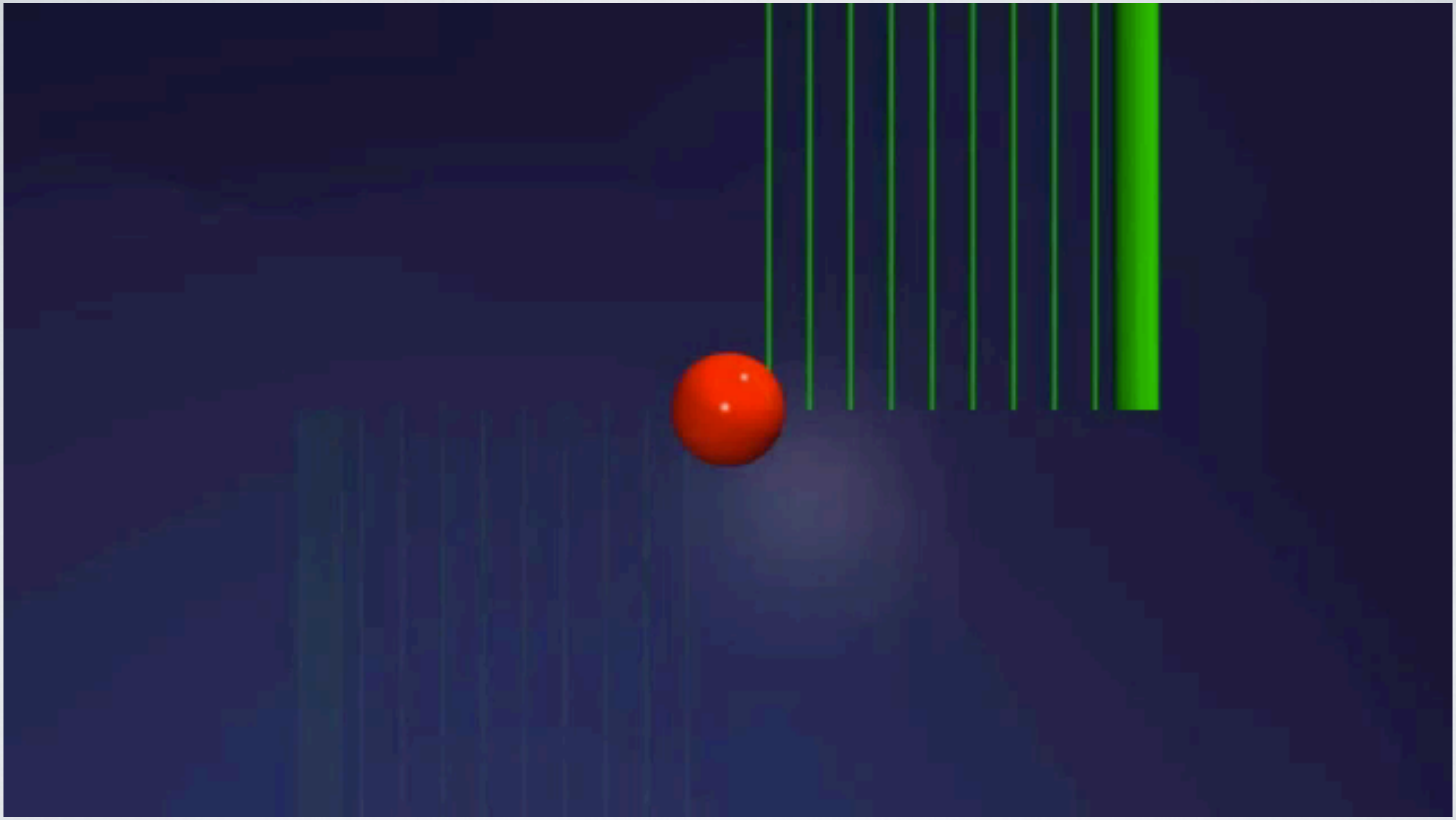
❼ Repeat the computations using the updated velocities and positions.



atom 1
atom 2
atom 3
atom 4
atom 5
position
velocity $V_{x1}|_{t+\Delta t} : V_{y1}|_{t+\Delta t}$
velocity $V_{x2}|_{t+\Delta t} : V_{y2}|_{t+\Delta t}$
velocity $V_{x3}|_{t+\Delta t} : V_{y3}|_{t+\Delta t}$
velocity $V_{x4}|_{t+\Delta t} : V_{y4}|_{t+\Delta t}$
velocity $V_{x5}|_{t+\Delta t} : V_{y5}|_{t+\Delta t}$

http://atomsinmotion.com/book/chapter5/md

Simulation Overview



http://www.youtube.com/watch?v=ILFEqKI3sm4

IV. Advanced Topics

Ensembles

- Naturally MD ensemble is microcanonical (NVE):
  N - fixed # atoms
  V - fixed volume
  E - fixed energy

- What if we want to simulate in other thermodynamic ensembles that are closer to experimental systems?

Canonical (isothermal-isochoric)	-	**NVT**
Isothermal-isobaric	-	**NPT**
Isenthalpic-isobaric	-	**NPH**

- MD is typically restricted to fixed **N**

Thermostats

- The temperature of a classical system is defined by the average molecular velocity

$$E_{kin} = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 \qquad \frac{1}{2} N_{df} kT = E_{kin}$$

- All thermostats are based on rescaling molecular velocities:

V-rescaling	- simple uniform rescaling of $\{v_i\}$
	- does **not** yield canonical ensemble
Berendsen	- weak first-order coupling of v_i to target T
	- does **not** yield canonical ensemble
Andersen	- periodic v_i replacement with M-B dist^n
	- correct coord canonical ensemble, **but** unsuitable for for studying dynamics due to v_i discontinuities
Nosé-Hoover	- weak coupling of v_i to target T via fictitious oscillators
	- correct coord & velocity canonical dist^n and fluctuations*

* for N-H **chains**, single N-H thermostat non-ergodic in certain systems

Barostats

- Pressure is computed from the virial equation

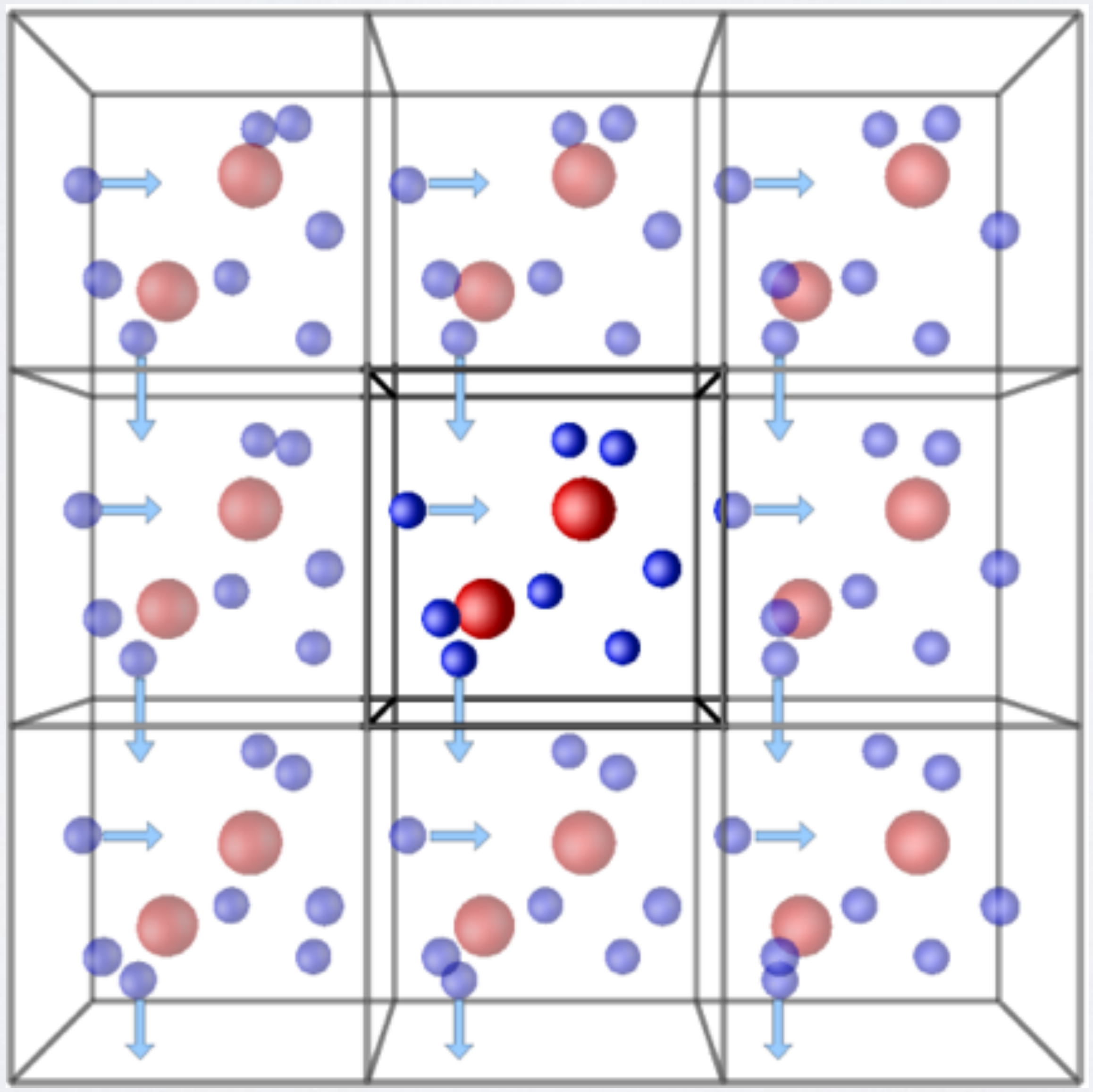
$$\mathbf{P} = \frac{2}{V}(\mathbf{E}_{kin} - \mathbf{\Xi}) \qquad \mathbf{\Xi} = -\frac{1}{2}\sum_{i<j} \boldsymbol{r}_{ij} \otimes \boldsymbol{F}_{ij}$$

- Barostats control pressure by scaling the box volume:

Berendsen	- weak first-order coupling of V to target P
	- does **not** yield isobaric ensemble
Parrinello-Rahman	- weak coupling of V to target P via fictitious oscillators
	- similar to Nosé-Hoover T coupling scheme
	- correct coord & velocity isobaric dist^n and fluctuations

Periodic boundary conditions

- Can only simulate small (nanoscopic) patch of space
- "Trick" the system into thinking it is infinite by tiling space with periodic replicas of fundamental simulation cell
- Molecules exiting one wall re-enter through the opposite!

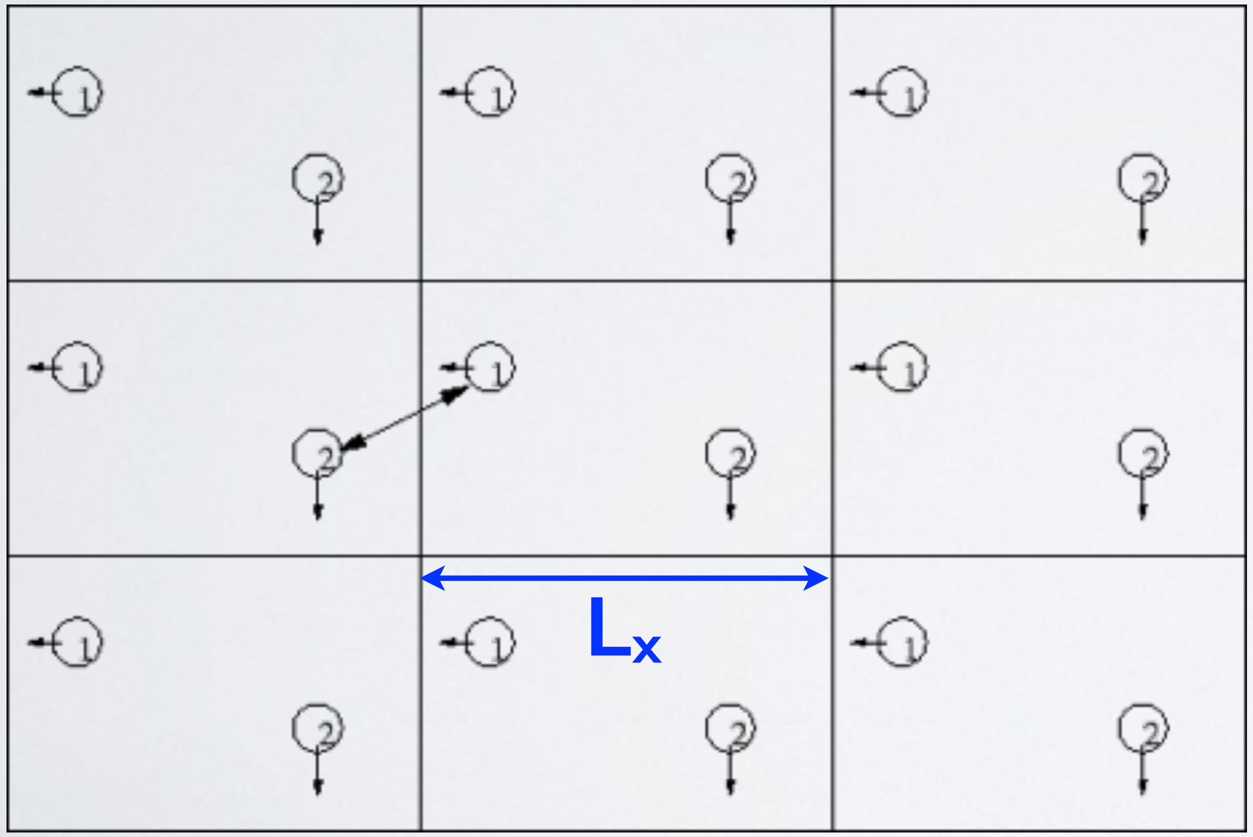


http://isaacs.sourceforge.net/phys/images/these-seb/pbc-seb.png

Minimum image convention

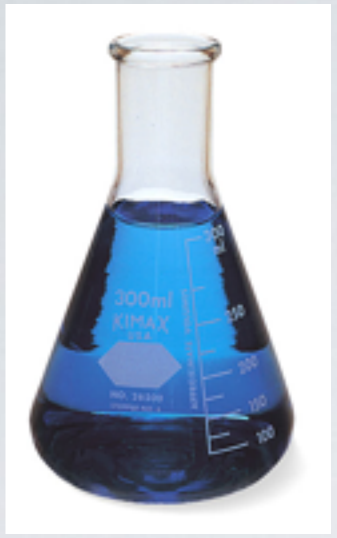
- Under PBC, inter-particle distances are measured using the **minimum image convention**

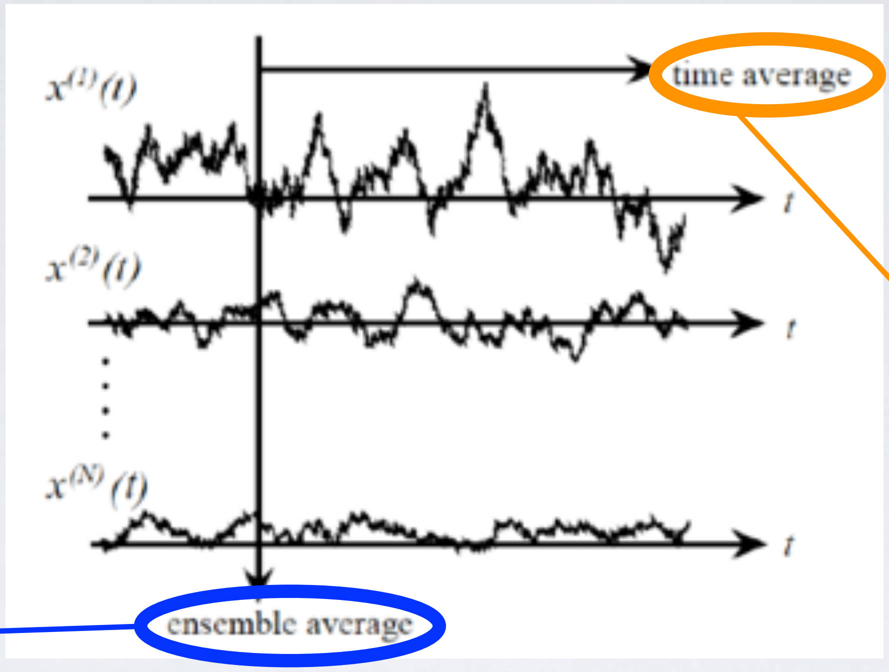
- We must ensure $r_{cutoff} < L/2$ so particles do not interact with multiple images of neighbors

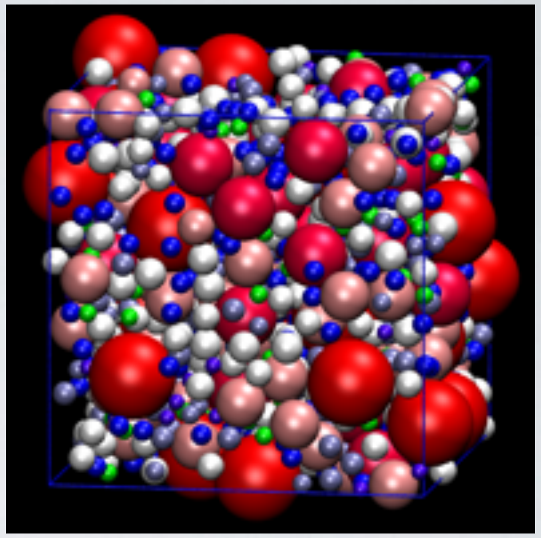


$$\Delta x_{MI} = \Delta x - L_x \, \mathrm{int}\left(\frac{\Delta x}{L_x}\right)$$

Ensemble and time averages







Experiment

Simulation

Ensemble average

- Average over all possible system configurations
- Naturally attained in experiments containing N_{Av} number of particles
- Very hard integral to perform numerically!

$$\langle A \rangle = \int\int dr^N dp^N A\left(r^N, p^N\right) \rho\left(r^N, p^N\right)$$

$$\rho\left(r^N, p^N\right) = \frac{1}{Q} exp\left[-\beta H\left(r^N, p^N\right)\right]$$

$$Q = \int\int dr^N dp^N exp\left[-\beta H\left(r^N, p^N\right)\right]$$

Time average

- Average over a single simulation trajectory
- Approximate time integral by summation

$$\bar{A} = \lim_{\tau \to \infty} \int_{t=0}^{\tau} dt A\left(r^N(t), p^N(t)\right)$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} A\left(r^N(m), p^N(m)\right)$$

http://cssb.biology.gatech.edu/cell_simulation http://www.stanford.edu/~rsasaki/EEAP248/slide1

Ergodic hypothesis

- The **ergodic hypothesis** states that for $\tau \to +\infty$

$$\langle A \rangle = \bar{A}$$

- So we can compute thermodynamic averages from **sufficiently long** MD trajectories
  - Intuition is that long simulations explore all of the important (low energy) terms in the ensemble average
  - How long is long enough is often unknown *a priori* and we rely on internal checks that observables reach steady state

- For **slow processes**, we may need accelerated sampling

Accelerated sampling

- Hardware limits the attainable MD time scales to O(μs), making it hard to study processes with >μs relaxations
- Energetically, the system can be trapped behind large barriers, with the transition an exceedingly rare event
- Accelerated sampling techniques use artificial biases to speed up sampling of conformational space:

umbrella sampling	- restrain system to hi E configurations using biasing potentials
replica exchange	- use T swaps to accelerate system dynamics at hi T
Hamiltonian exchange	- use H swaps to make exploration easier
hyperdynamics	- modify H with boost potential to enhance sampling
metadynamics	- lay down history dependent potential to flatten H
parallel replica	- simulate multiple system copies to accelerate escape
T accelerated	- hi T/hi mass coupling of part of system

Specialized MD variants

- **Car-Parrinello MD**
  - ab initio MD (no FF rqd!)
  - nuclear forces from solution of the electronic problem
  - prohibitively expensive and slow for big systems

- **ReaxFF**
  - reactive MD force field
  - enables classical modeling of chemical reactions

- **GPU enabled MD**
  - massive speedups on commodity graphics cards

- **Implicit field models**
  - trades accuracy for time scale

Car, R. and Parrinello, M. PRL 55 22 2471 (1985)   Nielson, K.D. et al. The Journal of Physical Chemistry A 109 3 493 (2005)

Limitations and Caveats

- No electrons and so no chemical reactions (but ReaxFF)
- No quantum effects (but QM/MM)
- Availability, transferability, and quality of force fields
- Time and length scale limitations
- Statistical significance of single trajectories
- Equilibrated?

Common mistakes

- **Simulation too short (#1 problem!)**
  - answers are not meaningful
  - out of thermodynamic equilibrium
- **Inadequate forcefield**
  - GIGO
- **Δt too large**
  - E not conserved, unstable trajectory
- **System too small**
  - finite size effects
  - hard to model low conc. in small box
- **Missing important physics or chemistry**
  - e.g., salt, surface, impurity
- **Cut-offs too short**
  - improper treatment of long-range interactions

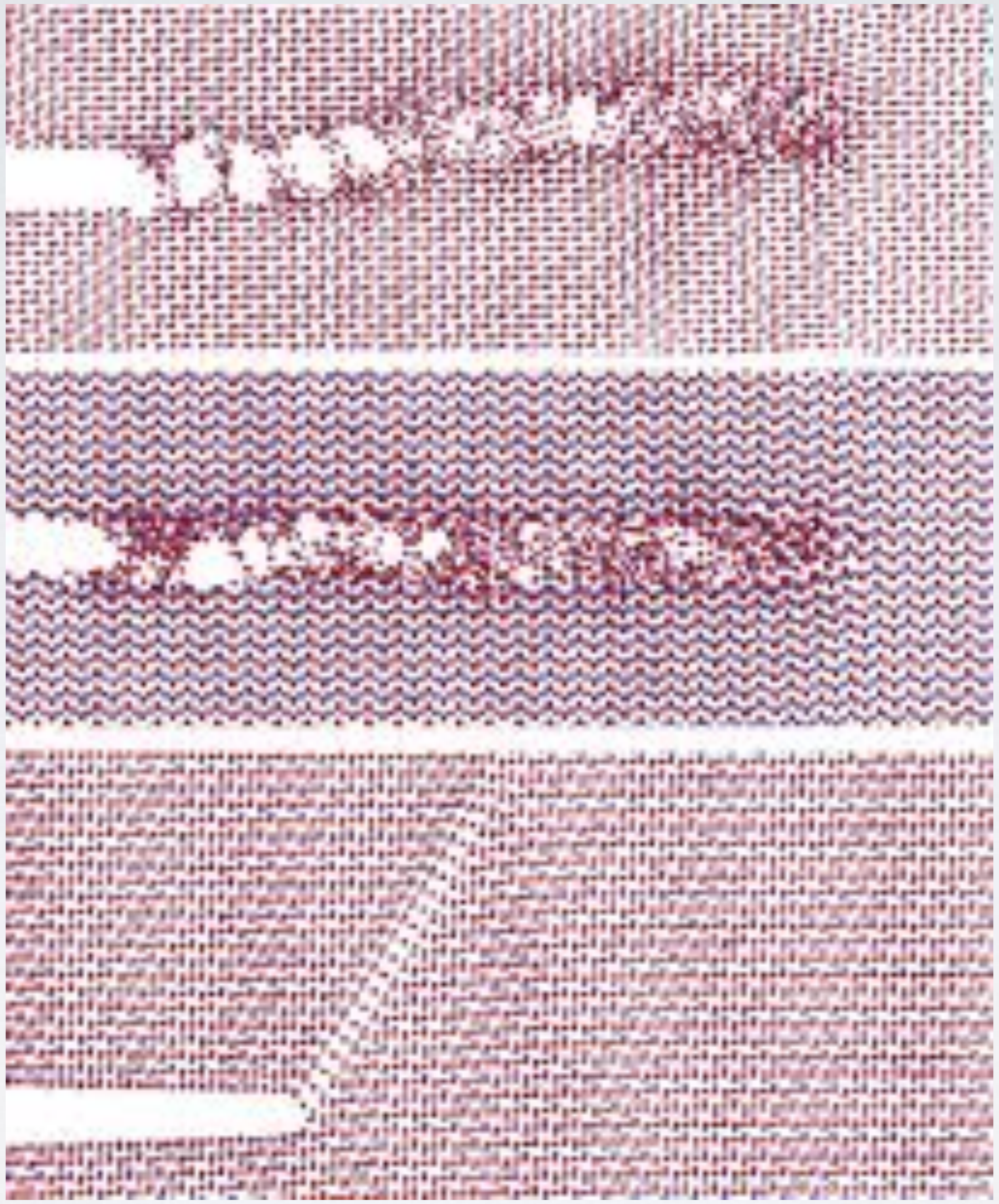
V. Molecular Dynamics Packages

MD software

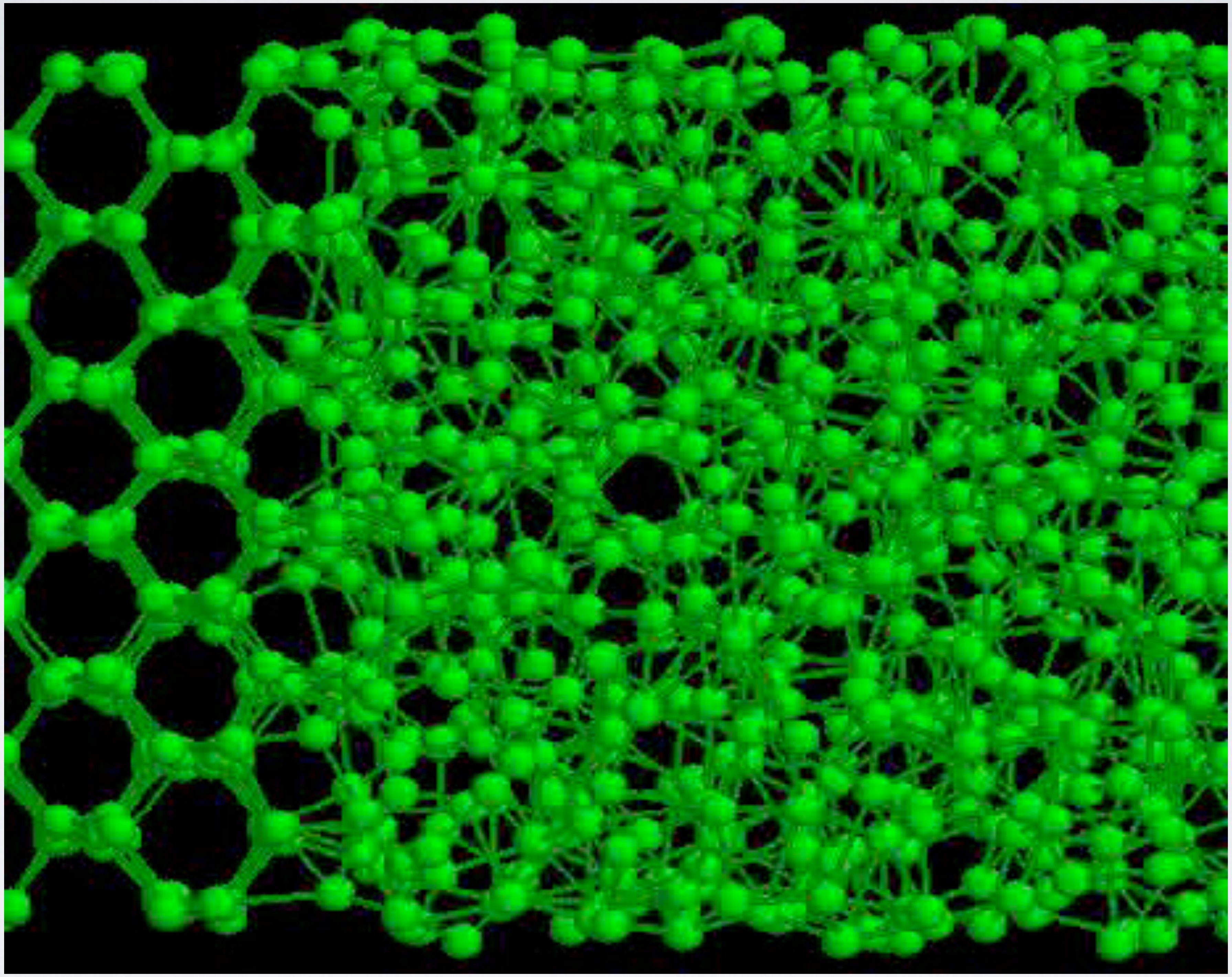
GROMACS FAST. FLEXIBLE. FREE.	U. Groningen www.gromacs.org	FREE
CHARMM	Harvard www.charmm.org	$600
AMBER	Rutgers *et al.* www.ambermd.org	$400
NAMD Scalable Molecular Dynamics	UIUC www.ks.uiuc.edu	FREE
Desmond D E Shaw Research	D.E. Shaw Research www.deshawresearch.com	FREE
LAMMPS	Sandia National Lab http://lammps.sandia.gov	FREE
HOOMD blue	U. Michigan http://codeblue.umich.edu/hoomd-blue/	FREE
Folding@home distributed computing	Folding@home http://folding.stanford.edu	FREE

VI. Applications

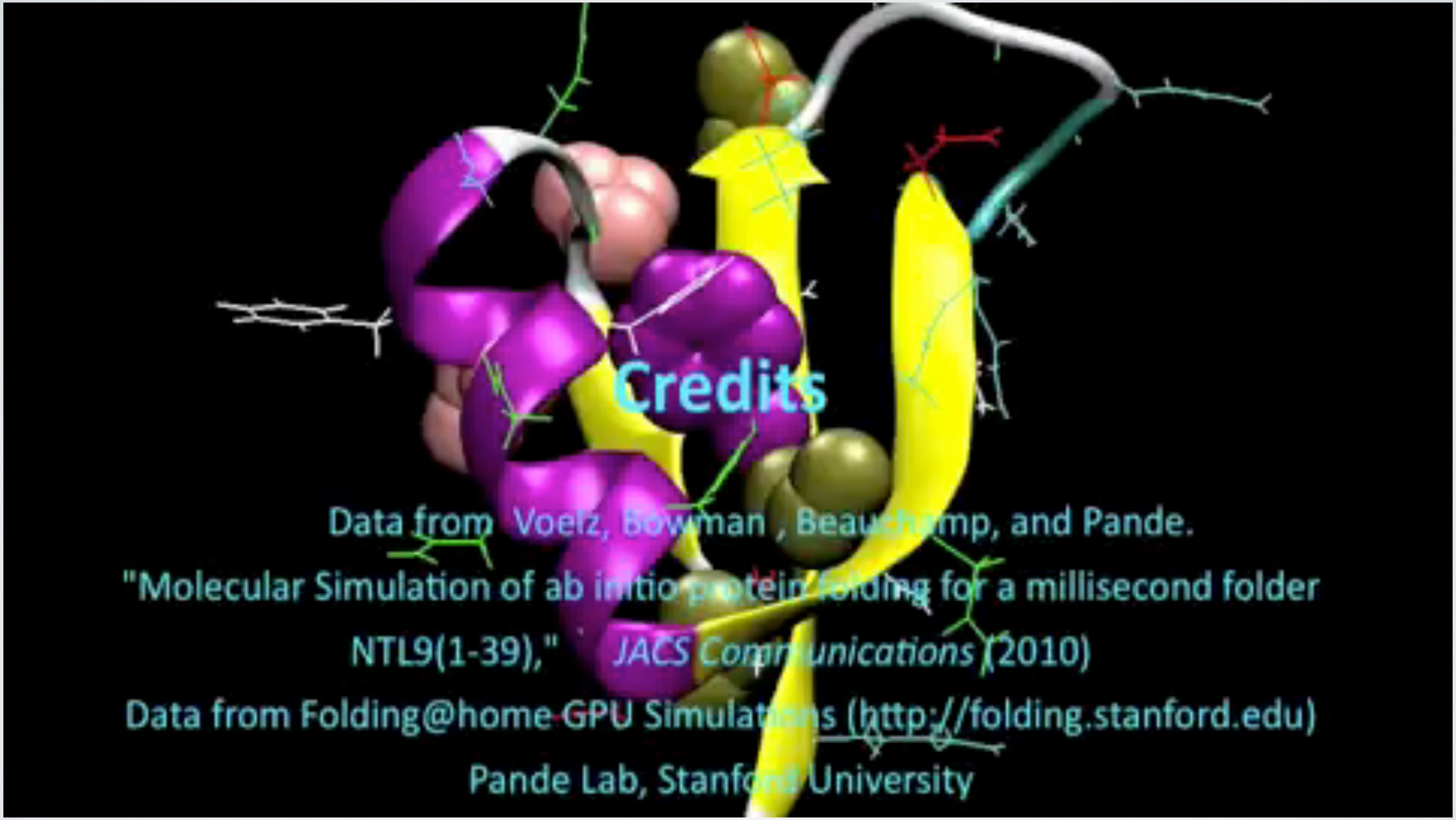
Fracture mechanics


Crack propagation in crystal planes of alumina

Phase transitions


Silicon crystallization

Protein folding



Credits

Data from Voelz, Bowman, Beauchamp, and Pande.

"Molecular Simulation of ab initio protein folding for a millisecond folder NTL9(1-39)," *JACS Communications* (2010)

Data from Folding@home GPU Simulations (http://folding.stanford.edu)

Pande Lab, Stanford University

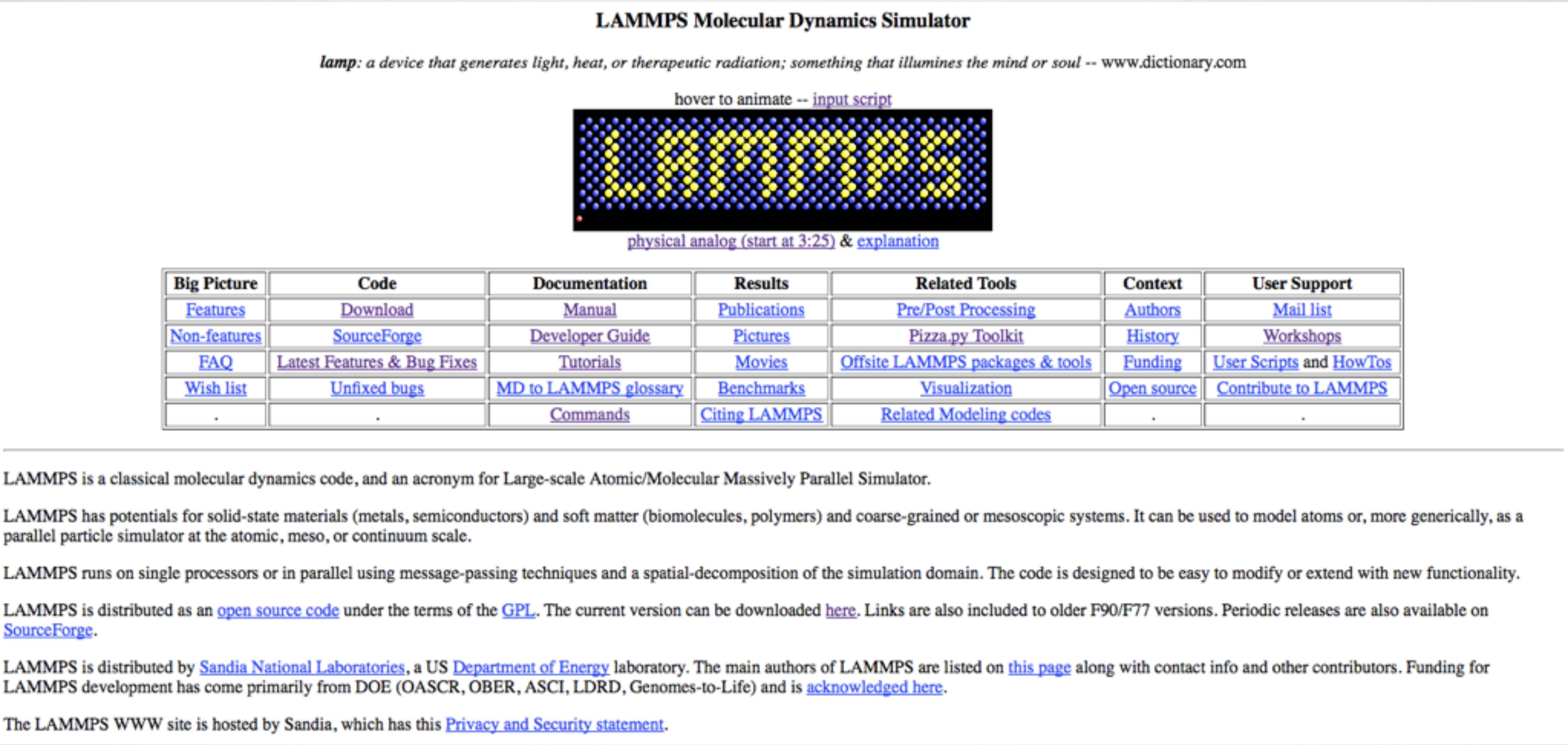
https://www.youtube.com/watch?v=gFcp2Xpd29I

VII. LAMMPS

LAMMPS

http://lammps.sandia.gov

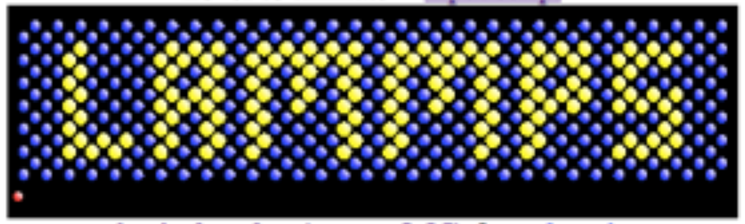
Large-scale **A**tomic/**M**olecular **M**assively **P**arallel **S**imulator



LAMMPS Molecular Dynamics Simulator

***lamp**: a device that generates light, heat, or therapeutic radiation; something that illumines the mind or soul* -- www.dictionary.com

hover to animate -- input script



physical analog (start at 3:25) & explanation

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
.	.	Commands	Citing LAMMPS	Related Modeling codes	.	.

LAMMPS is a classical molecular dynamics code, and an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator.

LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale.

LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. The code is designed to be easy to modify or extend with new functionality.

LAMMPS is distributed as an open source code under the terms of the GPL. The current version can be downloaded here. Links are also included to older F90/F77 versions. Periodic releases are also available on SourceForge.

LAMMPS is distributed by Sandia National Laboratories, a US Department of Energy laboratory. The main authors of LAMMPS are listed on this page along with contact info and other contributors. Funding for LAMMPS development has come primarily from DOE (OASCR, OBER, ASCI, LDRD, Genomes-to-Life) and is acknowledged here.

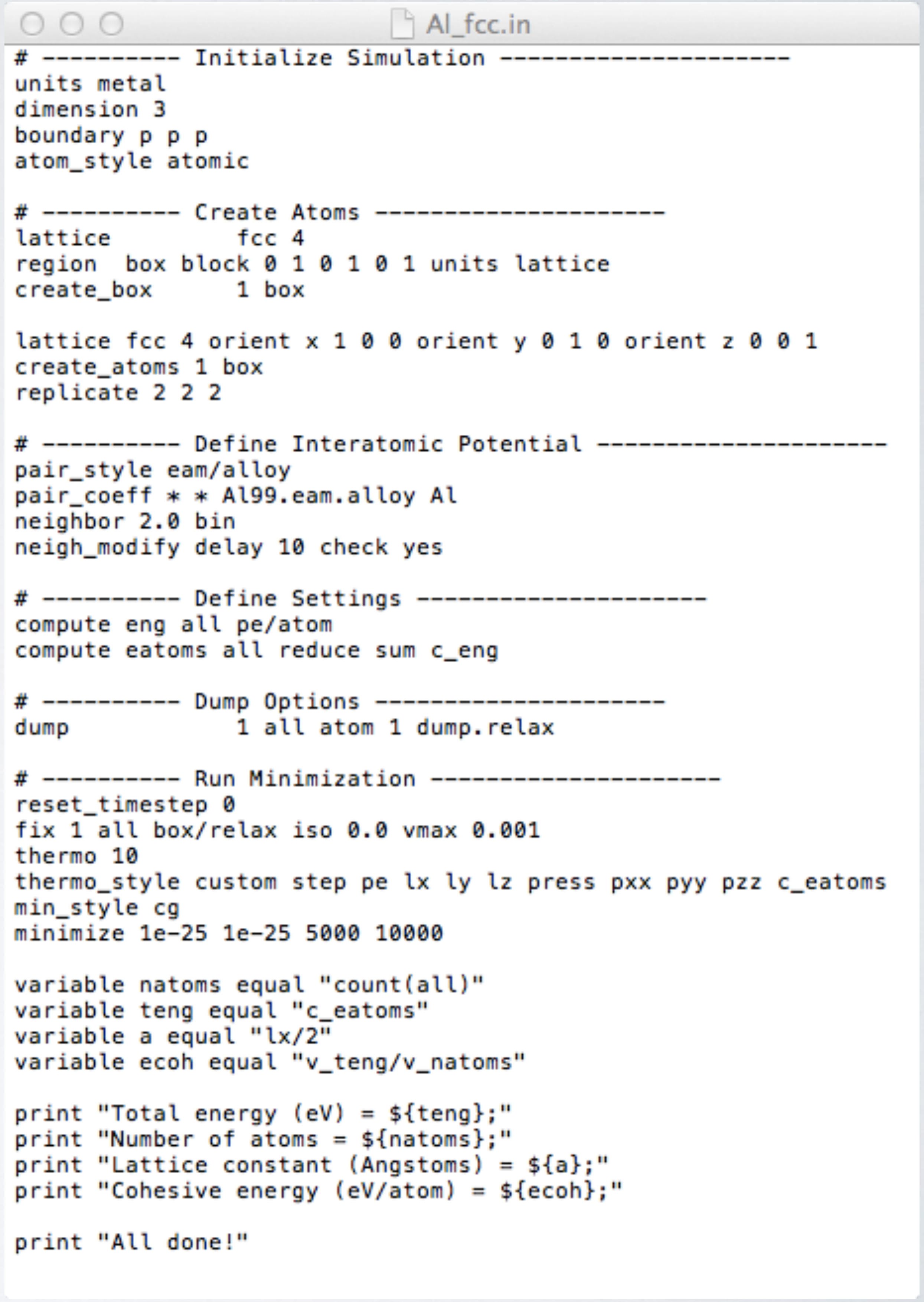
The LAMMPS WWW site is hosted by Sandia, which has this Privacy and Security statement.

History

- Born mid-90's in cooperation between Sandia, LLNL, Cray, Bristol Meyers Squibb, and Dupont — now developed at Sandia under DOE funding
- Current release in C++ w/ MPI
- **Open source and free under GPL**
- *Platforms:* Linux, Mac, Windows
- *Format:* exe, RPM, PPA, SVN, Git, Homebrew, tarball

Usability

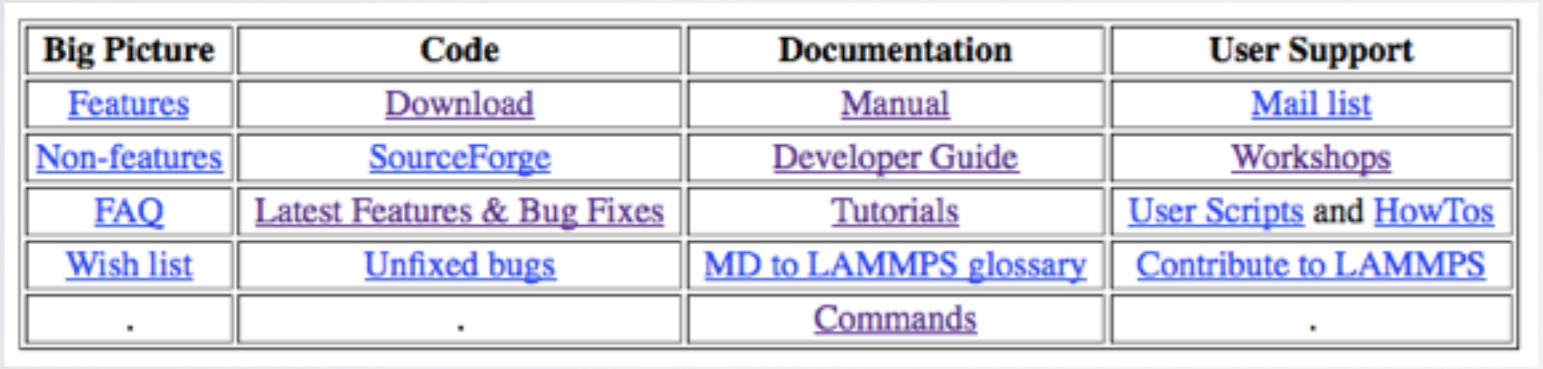
- Run initialization and control via **input script**
- Call from command line as `./lmp_linux < in.comp`
- **No GUI**, but some python tools available (http://lammps.sandia.gov/doc/Section_python.html)



```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

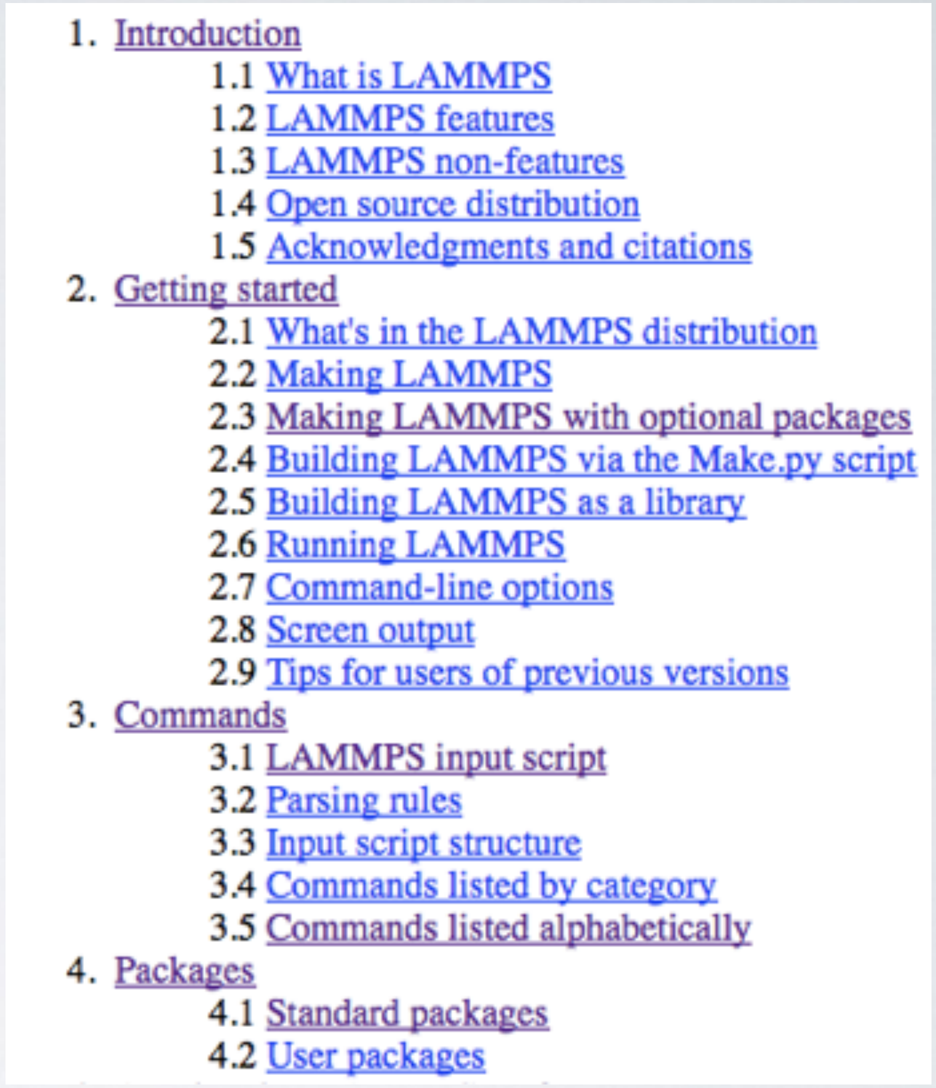
Documentation

- Excellent manual (http://lammps.sandia.gov/doc/Manual.html)
- Introductory Tutorials and HowTos (http://lammps.sandia.gov/howto.html)



Big Picture	Code	Documentation	User Support
Features	Download	Manual	Mail list
Non-features	SourceForge	Developer Guide	Workshops
FAQ	Latest Features & Bug Fixes	Tutorials	User Scripts and HowTos
Wish list	Unfixed bugs	MD to LAMMPS glossary	Contribute to LAMMPS
.	.	Commands	.

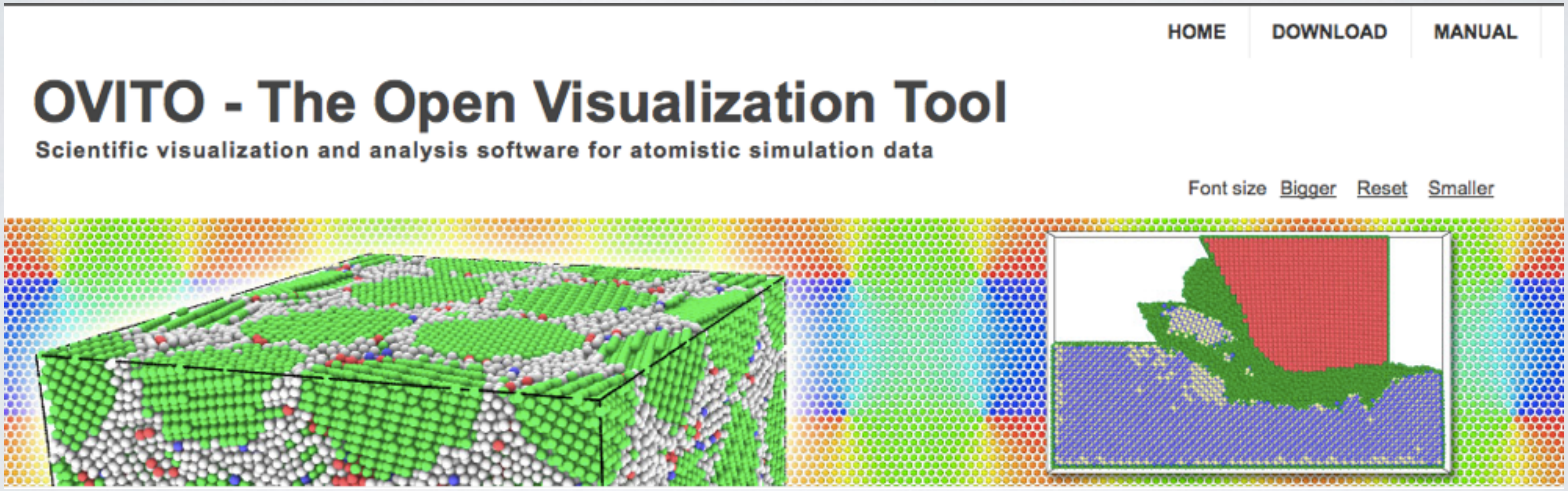
- Friendly user base and mailing list (http://lammps.sandia.gov/mail.html)
- Excellent third-party tutorials hosted by CAVS @ MSU (https://icme.hpc.msstate.edu/mediawiki/index.php/LAMMPS_tutorials)



Visualization

- LAMMPS has no built-in visualization capability

- **OVITO** is a free, user-friendly and powerful visualization engine available for Linux, Mac and Windows

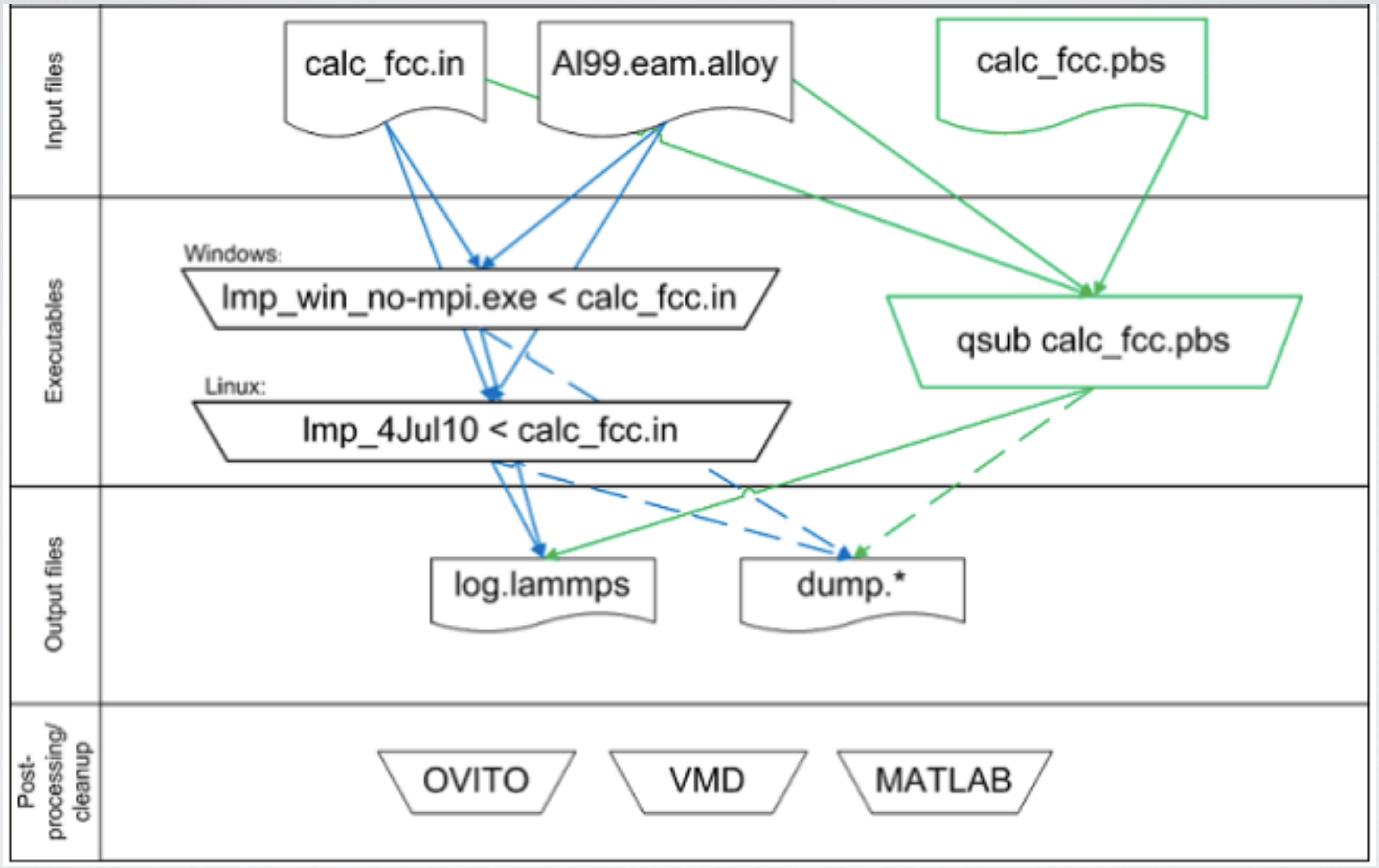


OVITO - The Open Visualization Tool

Scientific visualization and analysis software for atomistic simulation data

http://www.ovito.org

Running a simulation



VIII. Hands-on with LAMMPS

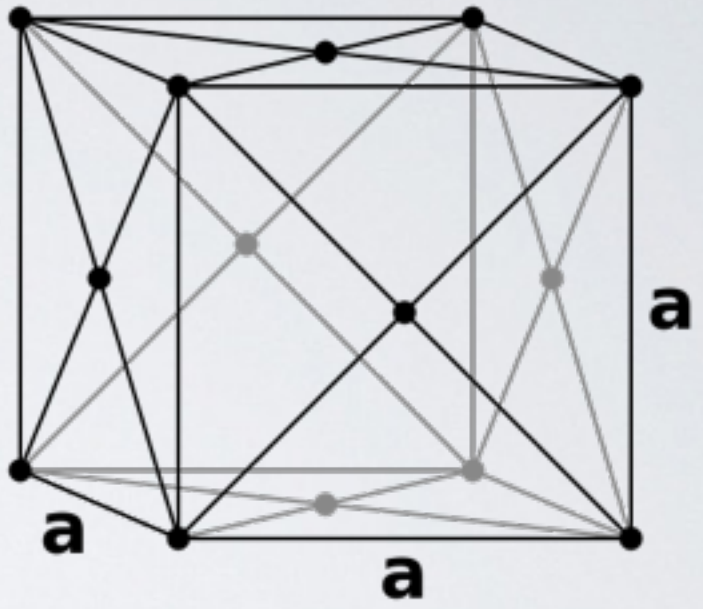
Adapted from materials developed by Mark A. Tschopp (US ARL) and hosted at https://icme.hpc.msstate.edu

Tutorial 1: Al cohesive energy

Tutorial 1: Al cohesive energy

- We will use LAMMPS to estimate the Al fcc cohesive energy, **E_{cohe}**, and lattice parameter, **a**

$$E_{cohe} = E_{solid} - \cancelto{0}{\sum_{atoms} E_{isolated}}$$



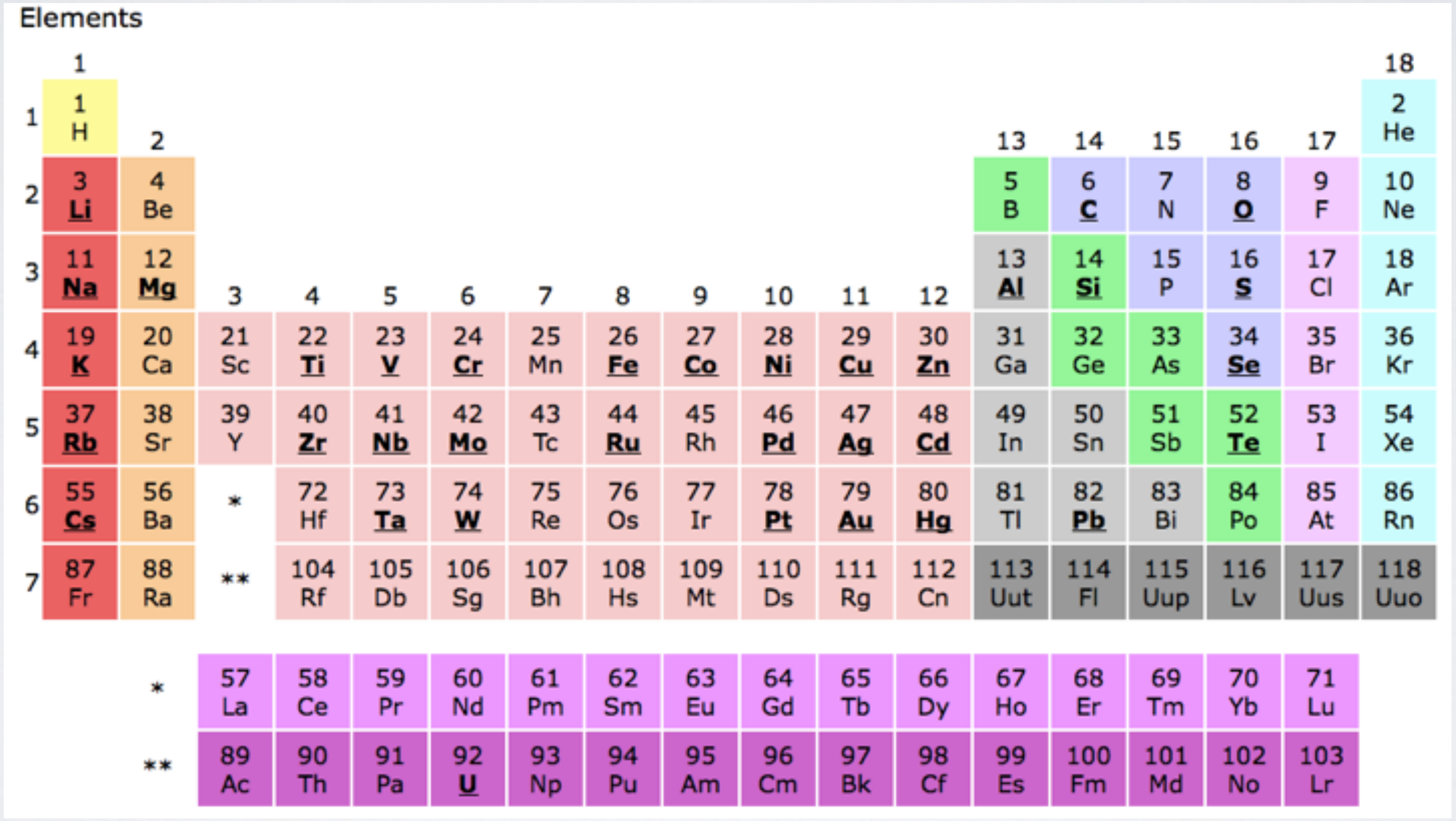
- Experimentally, **E_{cohe}** = -3.39 eV/atom* and **a** = 4.0495 Å*

- **Strategy:** We shall use a modern EAM potential for Al and optimize **E_{cohe}** as a function of **a**

*Charles Kittel. Introduction to Solid State Physics, 8th edition. Hoboken, NJ: John Wiley & Sons, Inc, 2005. *http://periodictable.com/Elements/013/data.html

Tutorial 1: Al cohesive energy

1. Download **Al99.eam.alloy** EAM potential from NIST Interatomic Potentials Repository Project (http://www.ctcms.nist.gov/potentials)



Elements

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1 H																	2 He
2	3 **Li**	4 Be											5 B	6 **C**	7 N	8 **O**	9 F	10 Ne
3	11 **Na**	12 **Mg**											13 **Al**	14 **Si**	15 P	16 **S**	17 Cl	18 Ar
4	19 **K**	20 Ca	21 Sc	22 **Ti**	23 **V**	24 **Cr**	25 Mn	26 **Fe**	27 **Co**	28 **Ni**	29 **Cu**	30 **Zn**	31 Ga	32 Ge	33 As	34 **Se**	35 Br	36 Kr
5	37 **Rb**	38 Sr	39 Y	40 **Zr**	41 **Nb**	42 **Mo**	43 Tc	44 **Ru**	45 Rh	46 **Pd**	47 **Ag**	48 **Cd**	49 In	50 Sn	51 Sb	52 **Te**	53 I	54 Xe
6	55 **Cs**	56 Ba	*	72 Hf	73 **Ta**	74 **W**	75 Re	76 Os	77 Ir	78 **Pt**	79 **Au**	80 **Hg**	81 Tl	82 **Pb**	83 Bi	84 Po	85 At	86 Rn
7	87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt	110 Ds	111 Rg	112 Cn	113 Uut	114 Fl	115 Uup	116 Lv	117 Uus	118 Uuo

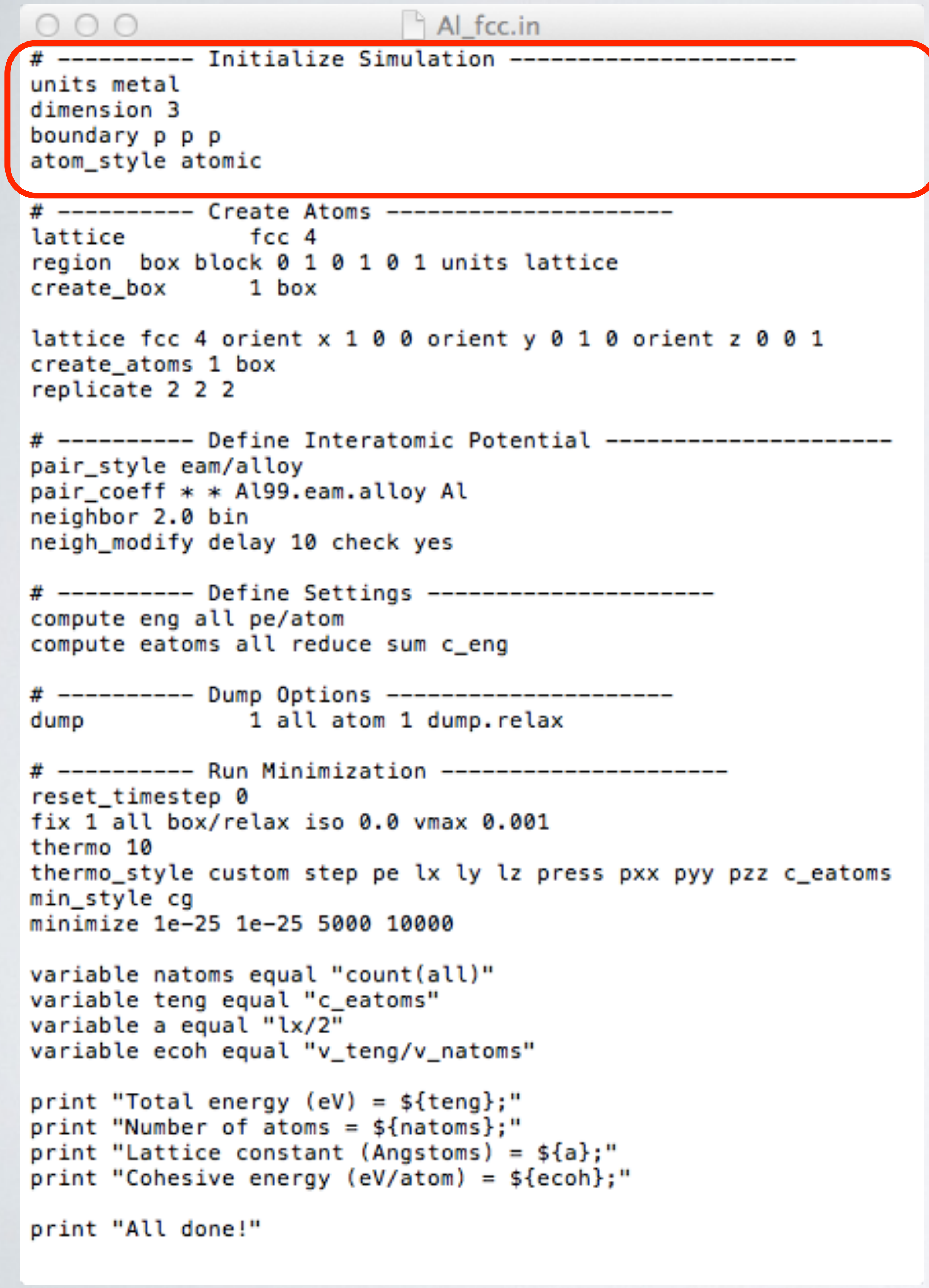
*	57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
**	89 Ac	90 Th	91 Pa	92 **U**	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

Tutorial 1: Al cohesive energy

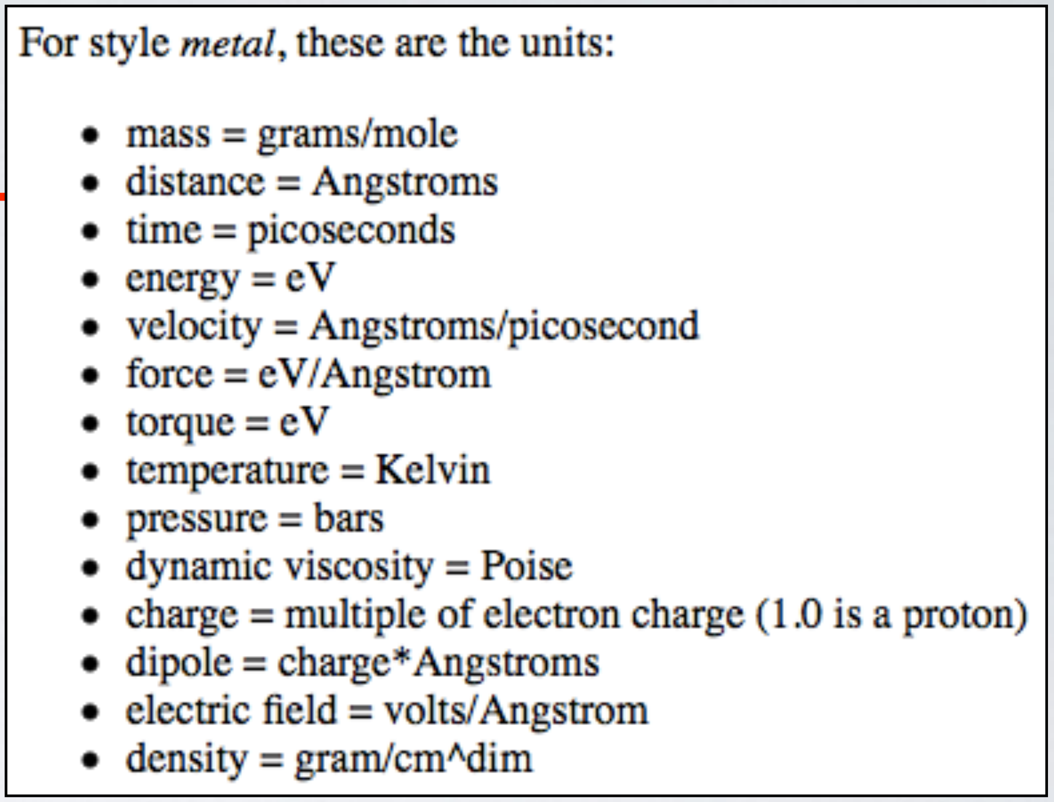
2. Obtain LAMMPS input file **Al_fcc.in** from http://ferguson.matse.illinois.edu/download/Al.zip

Al_fcc.in   Al99.eam.alloy   lmp_mac

Tutorial 1: Al cohesive energy



```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

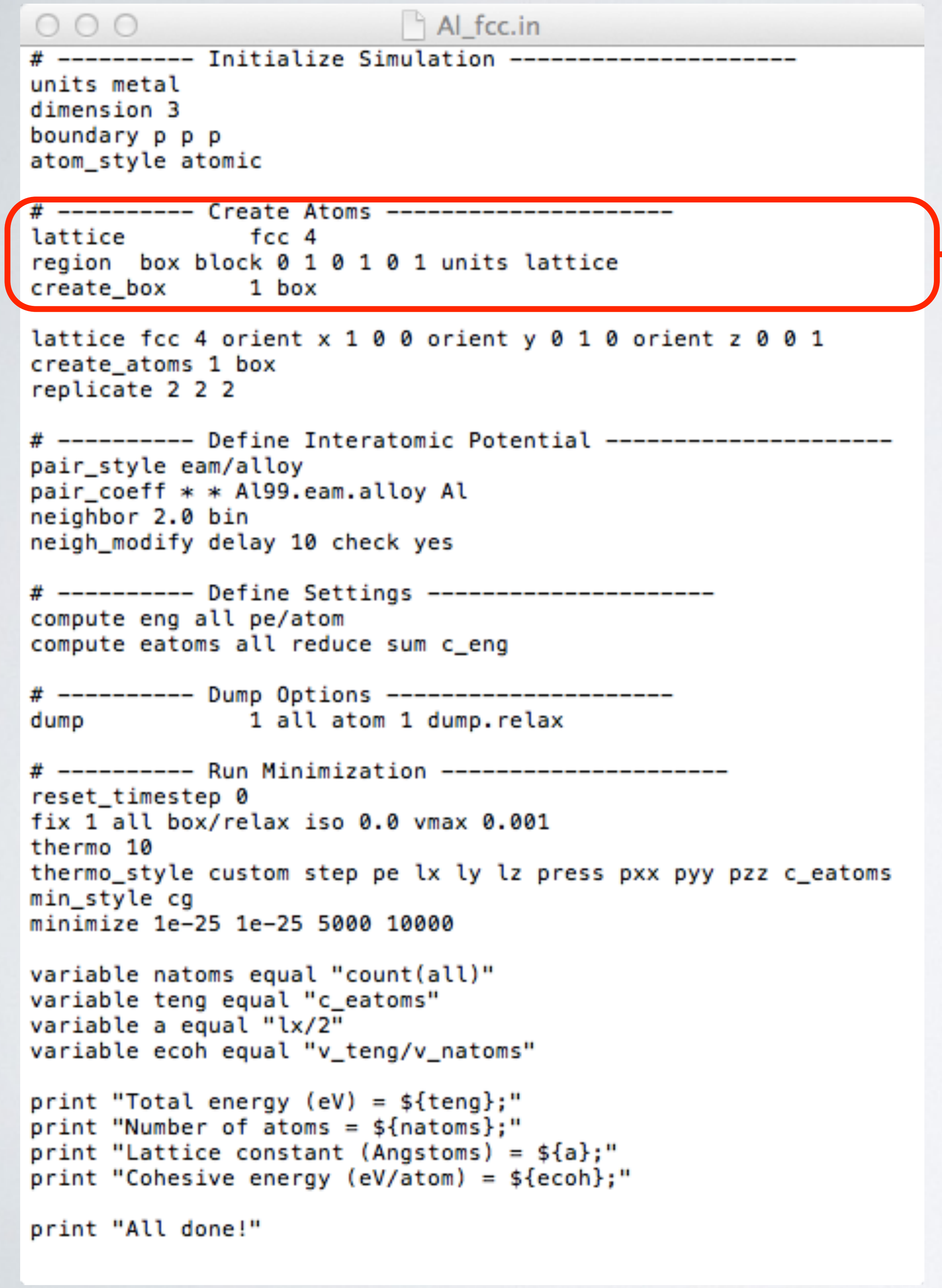


For style *metal*, these are the units:

- mass = grams/mole
- distance = Angstroms
- time = picoseconds
- energy = eV
- velocity = Angstroms/picosecond
- force = eV/Angstrom
- torque = eV
- temperature = Kelvin
- pressure = bars
- dynamic viscosity = Poise
- charge = multiple of electron charge (1.0 is a proton)
- dipole = charge*Angstroms
- electric field = volts/Angstrom
- density = gram/cm^dim

- **#** specifies a comment
- x,y,z **periodic boundaries**

Tutorial 1: Al cohesive energy

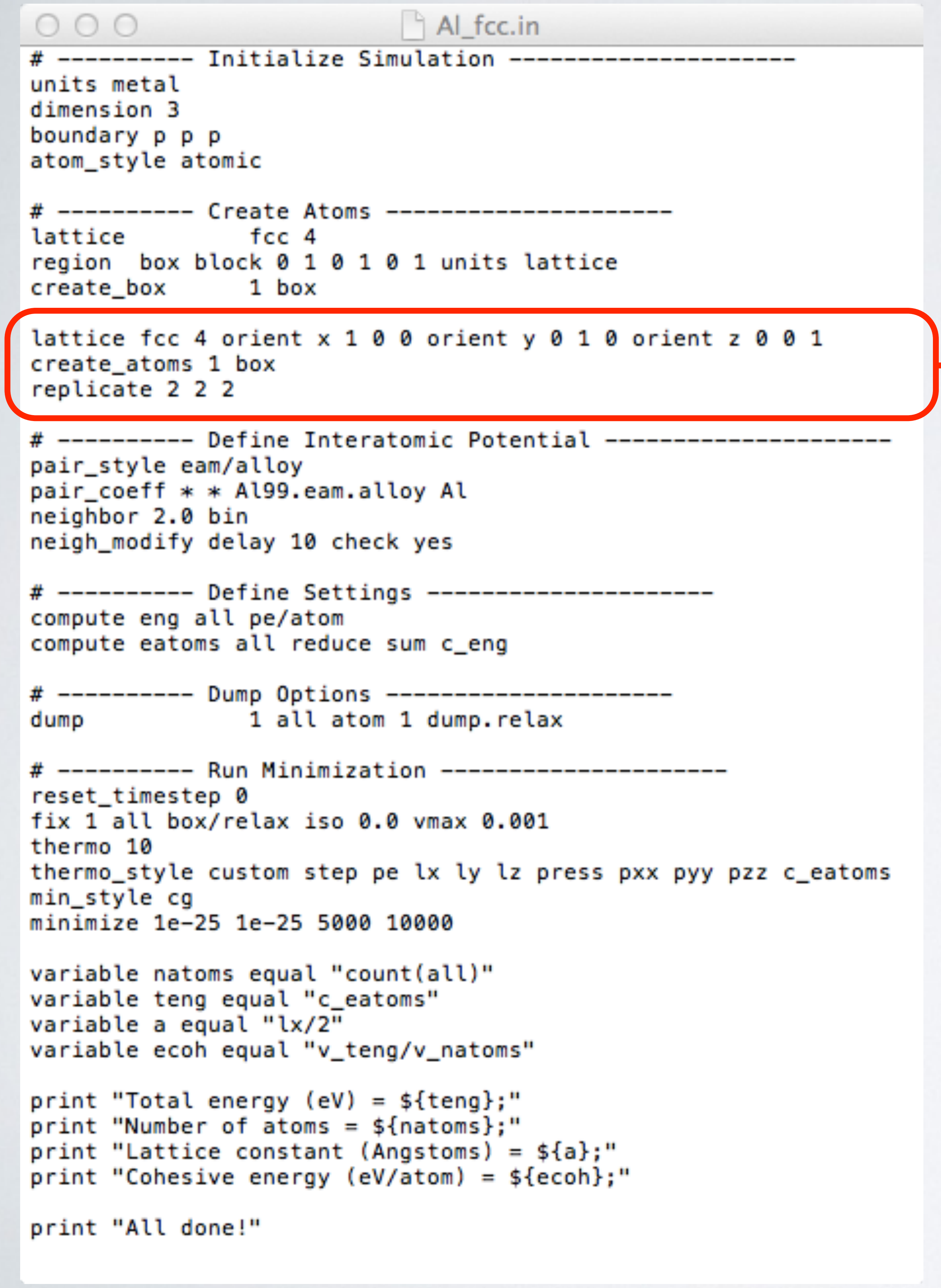


Al_fcc.in

```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Specify **fcc lattice** with **a**=4 Å
- Define **cuboidal block** labeled **box** holding **one lattice cell**
- Create **box** with **1** atom type

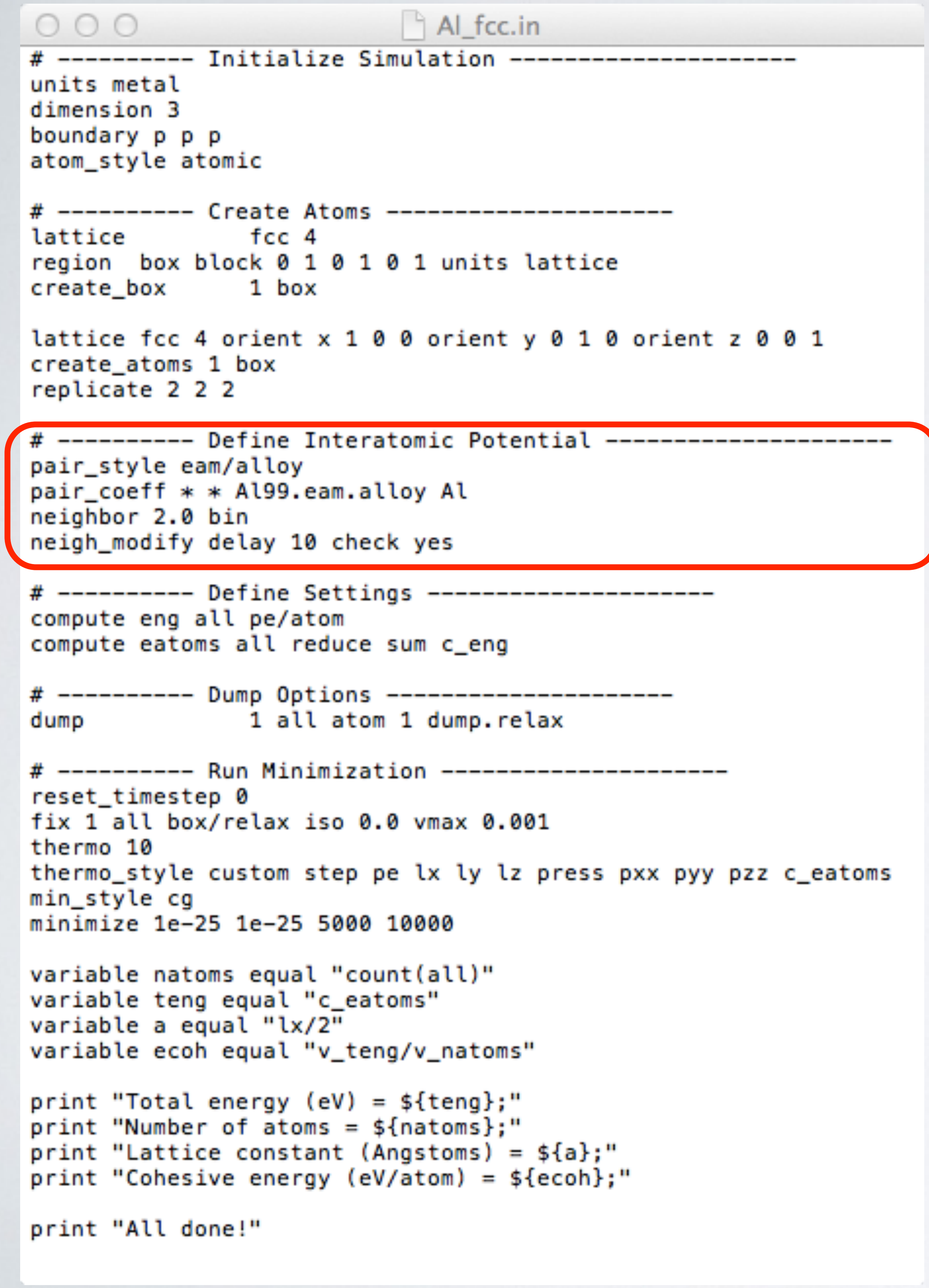
Tutorial 1: Al cohesive energy



```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Specify fcc lattice **orientation**
- Create atoms of type **1** on lattice sites within **box**
- **Replicate domain** by **2x2x2** in x,y,z [`replicate 1 1 1` would be more parsimonious for this trivially periodic system]

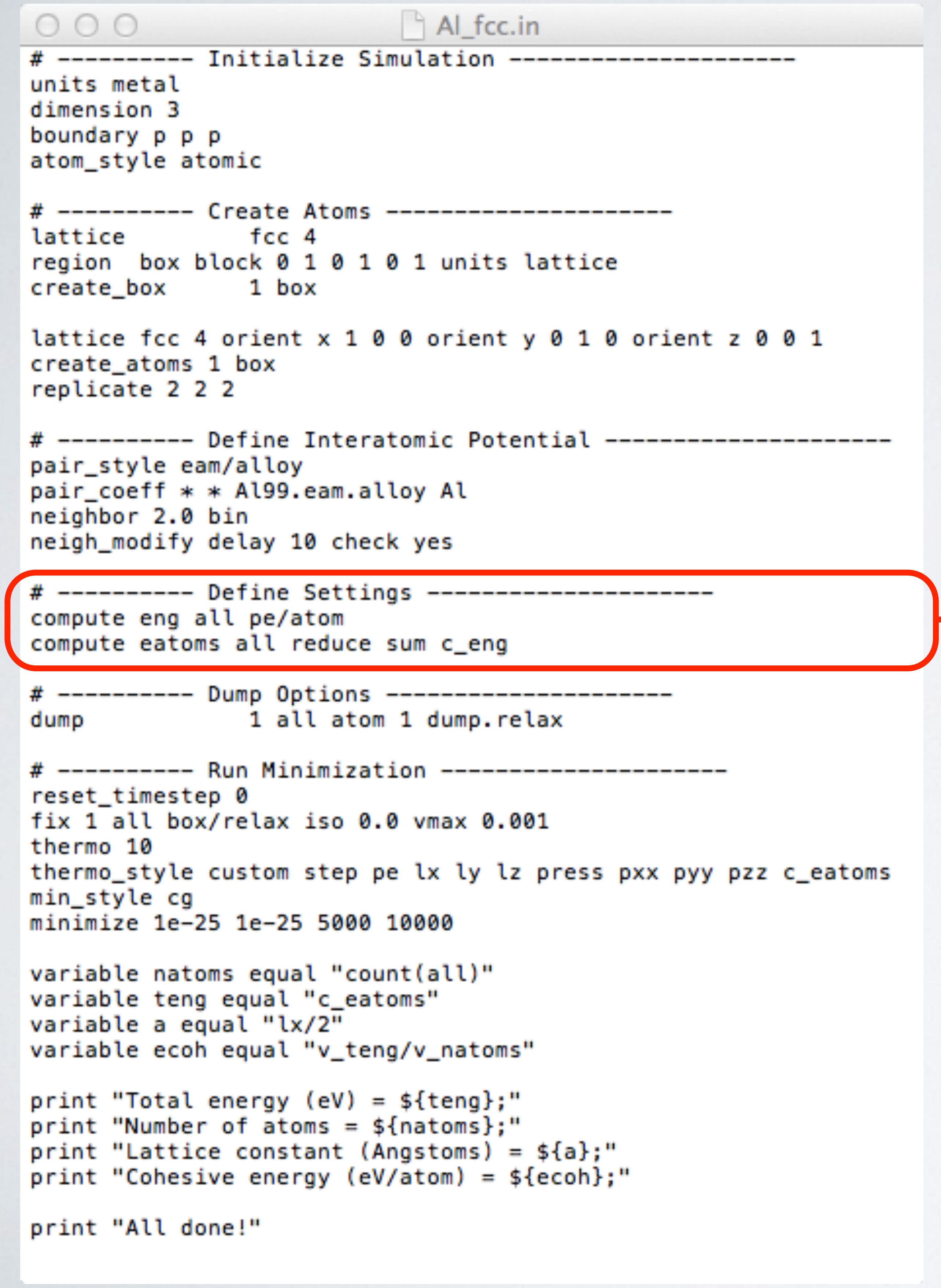
Tutorial 1: Al cohesive energy


Al_fcc.in

```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Define form of pairwise interaction potential as **eam/alloy** [misnomer, EAM is n-body]
- Use **Al** block of **Al99.eam.alloy** - specifies cutoff, F, **ρ**, and **Φ** - for all pairs [for one atom type, **1 1** fine]
- **2 Å skin thickness** for **neighbor list binning**
- Build neighbor list every **10 steps**, but **check** atom moved more than half skin thickness

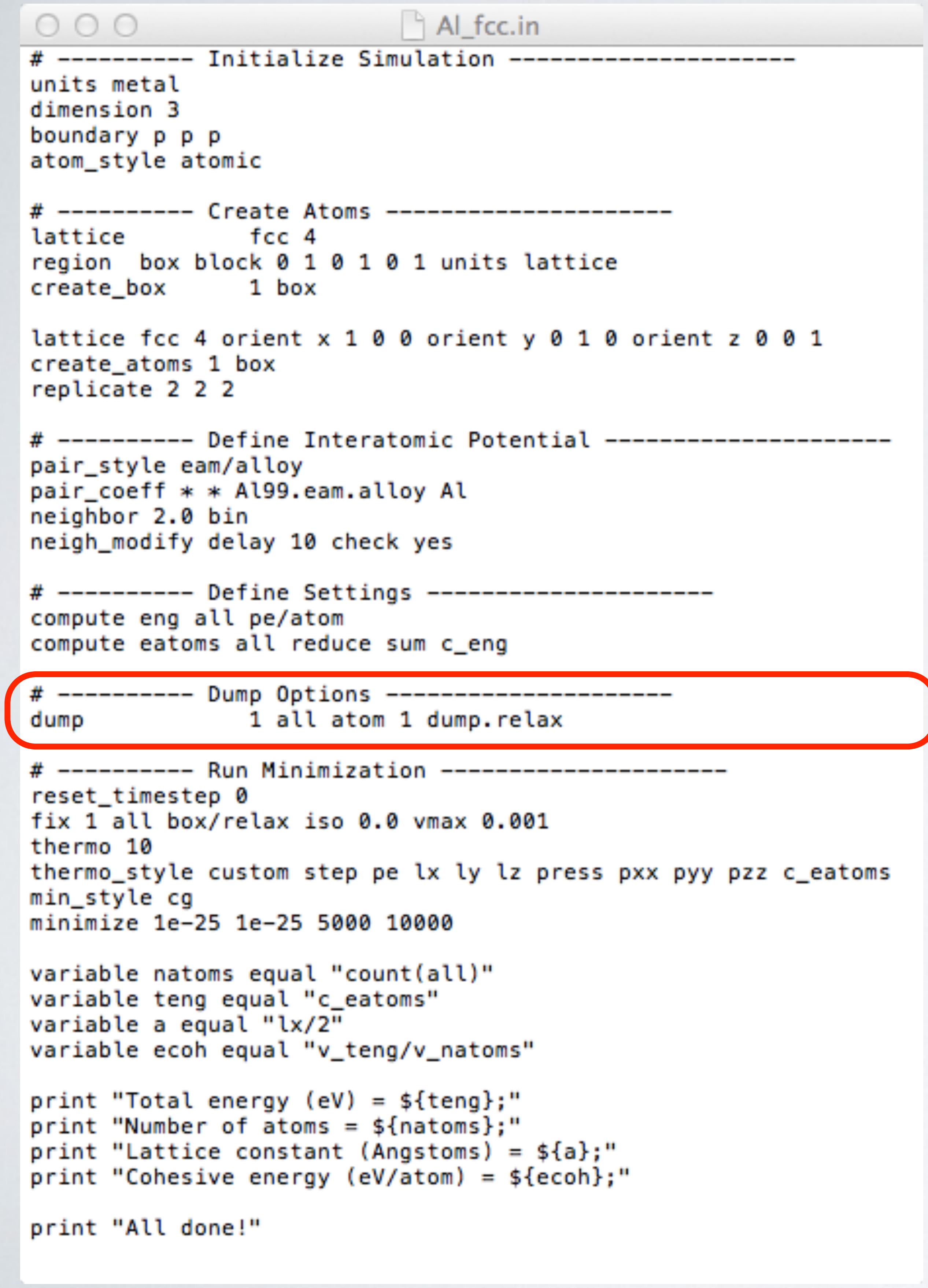
Tutorial 1: Al cohesive energy


Al_fcc.in

```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Define **computes** - quantities recalculated every time step [cf. **variables**, which evaluate a formula when called]
- Reference computes as **c_<name>**
- **c_eng** defined over **all** atoms to compute **potential energy per atom**
- **c_eatoms** performs **sum reduce** of c_eng vector over **all** atoms [alternatively: **`compute eatoms all pe`**]

Tutorial 1: Al cohesive energy

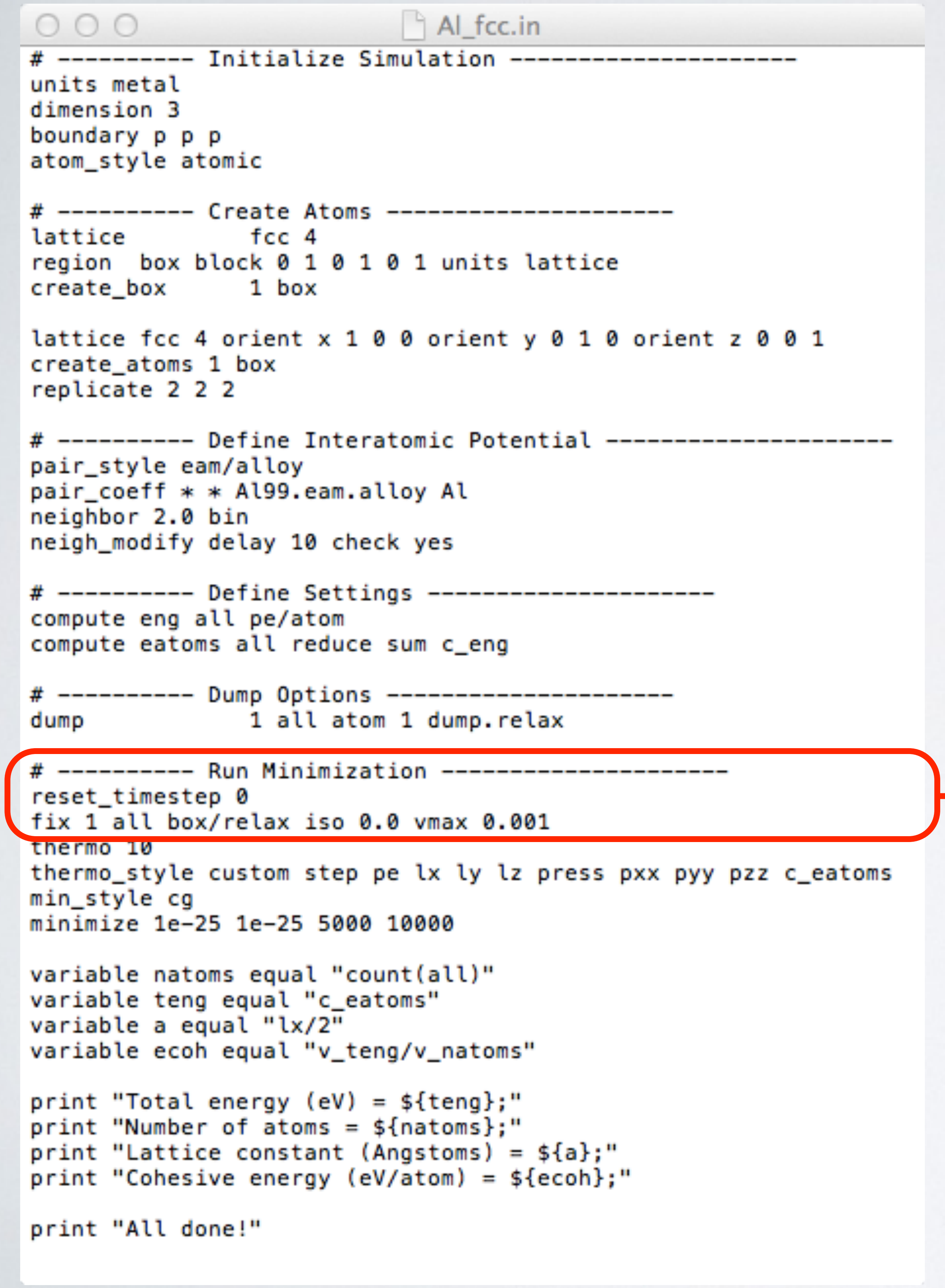


```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- A **dump** specifies how to write output data
- Tag dump with id **1** to write to **dump.relax** every **1** steps the coords of **all** of the **atoms**
- Dump format:

```
ITEM: TIMESTEP
0
ITEM: NUMBER OF ATOMS
32
ITEM: BOX BOUNDS pp pp pp
0 8
0 8
0 8
ITEM: ATOMS id type xs ys zs
1 1 0 0 0
2 1 0.25 0.25 0
3 1 0.25 0 0.25
4 1 0 0.25 0.25
...
```

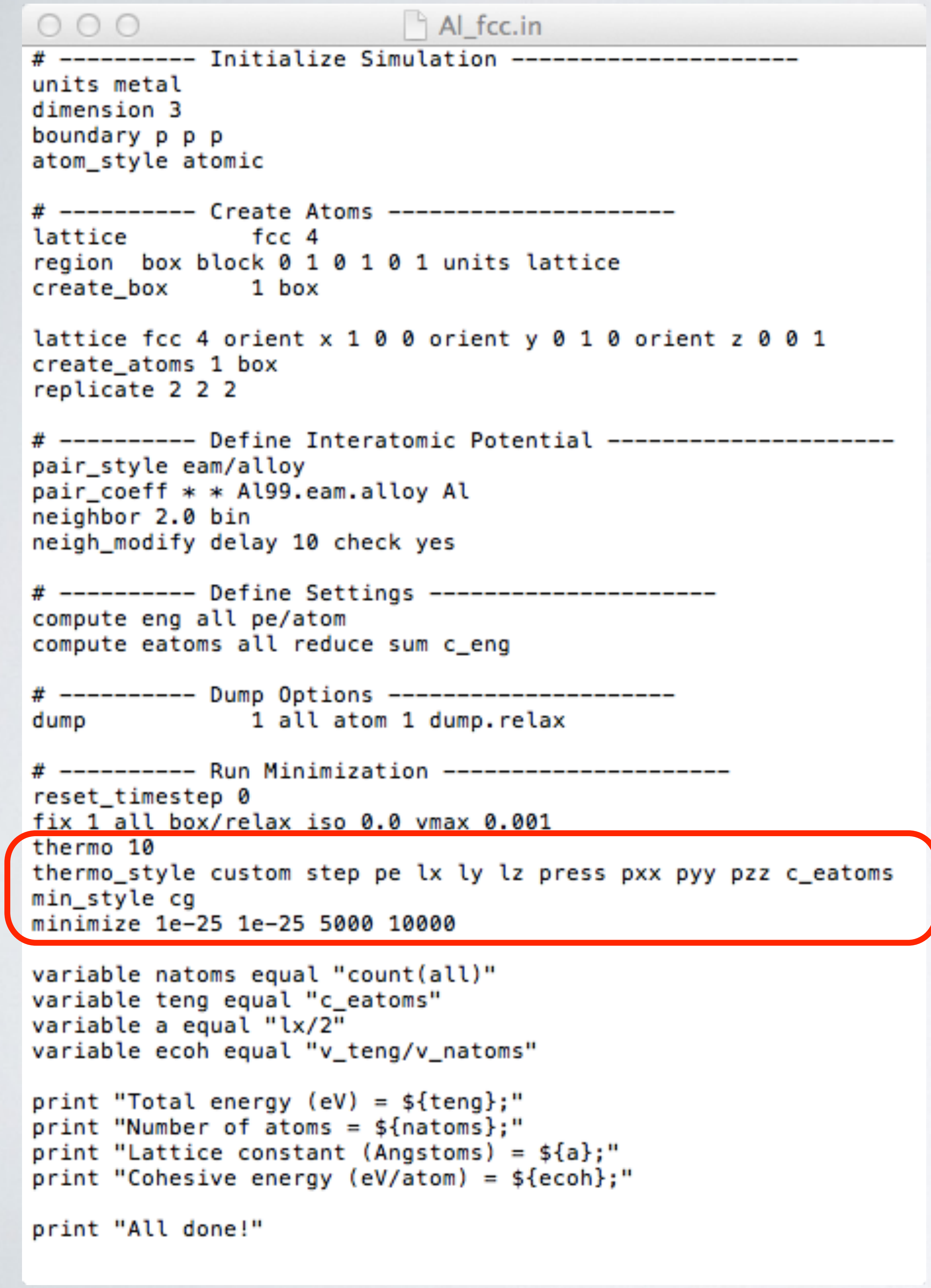
Tutorial 1: Al cohesive energy


Al_fcc.in

```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Reset time steps to **0**
- A **fix** is an operation applied at every time step
- Define fix **1** operating on **all** atoms **relaxes box** to an external **isotropic pressure** of **0.0 bar** with a **0.1% maximum fractional volume change per step**

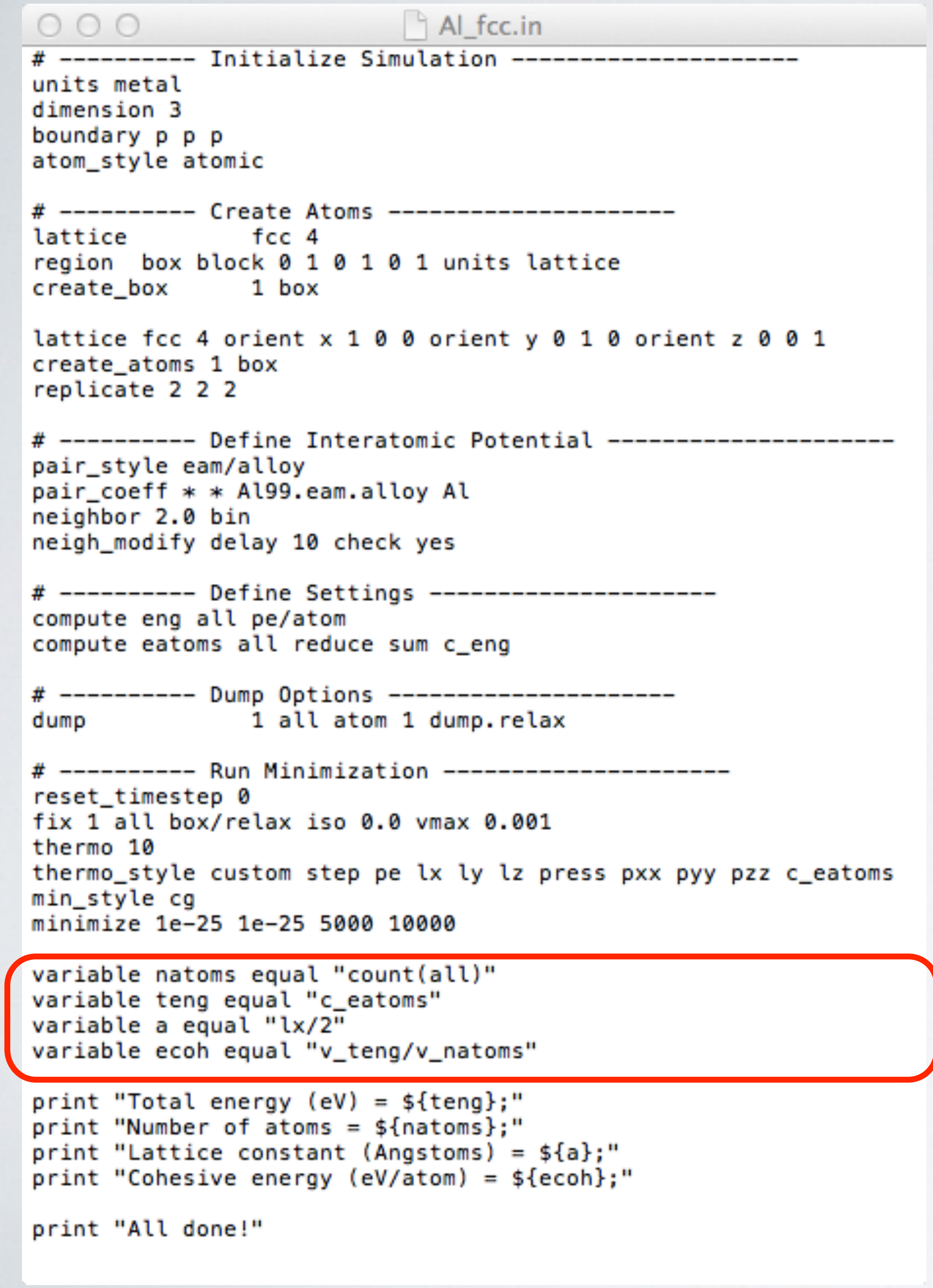
Tutorial 1: Al cohesive energy



```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Output **thermodynamic info** to screen every **10** steps [use **fix** / **dump** for file write]
- Customize thermo output
- Perform energy minimization by **conjugate gradient**
- **Minimize** $E = E_{FF} + E_{fix}$ with $\Delta E=10^{-25}$ (i.e., 1 part in 10^{25}) and $\Delta f=10^{-25}$, and a maximum of 5000 iterations and 10000 energy evaluations

Tutorial 1: Al cohesive energy


Al_fcc.in

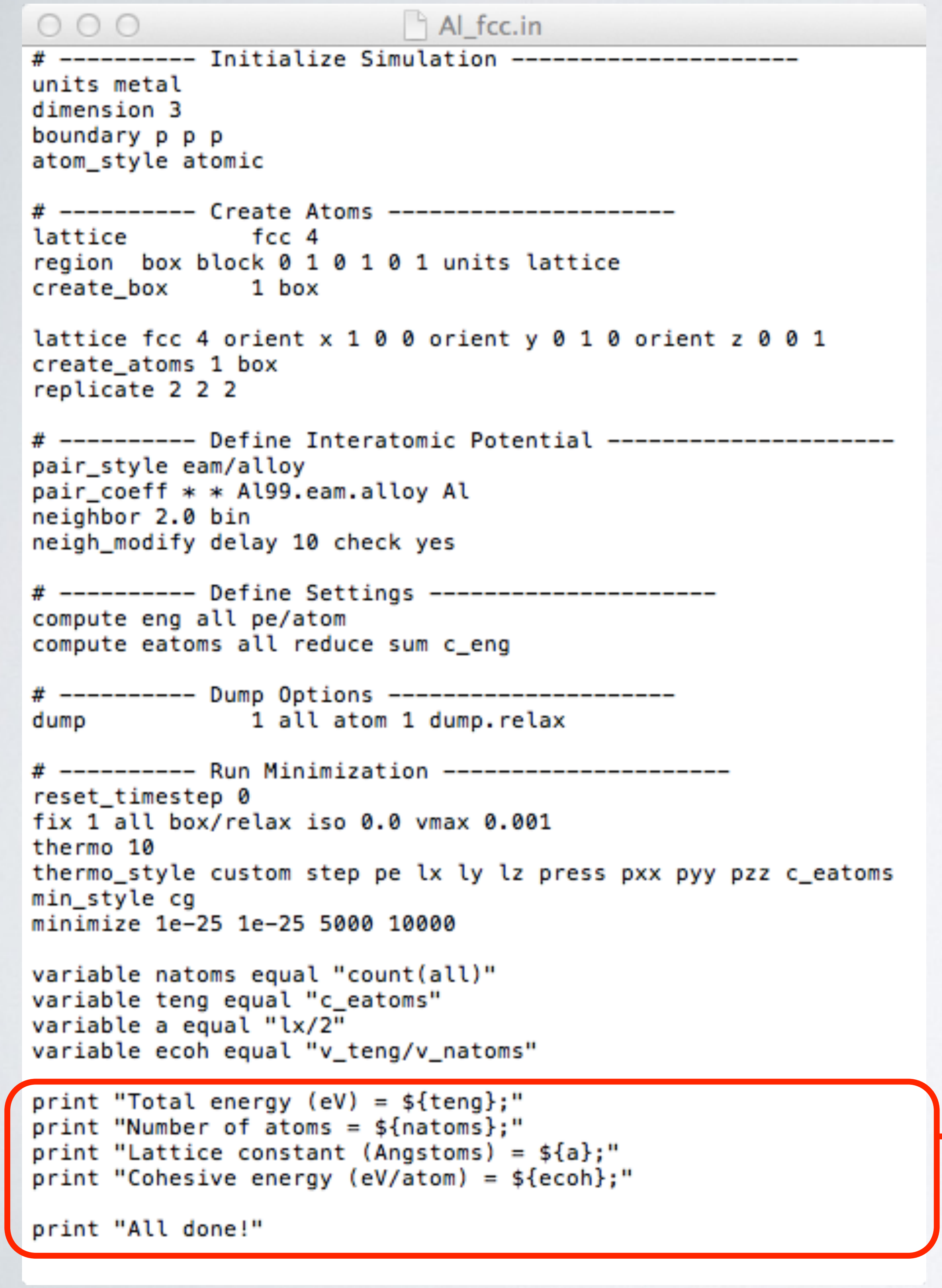
```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Define **variables** as formulas evaluated when called [cf. **computes**, simulation values recomputed each step]

- Reference variables as **v_<name>**

- natoms = # atoms
  teng = total PE (c_eatoms)
  a = lattice parameter
  (box side in x divided by
  # x replicas = 2)
  ecoh = cohesive energy /atom

Tutorial 1: Al cohesive energy

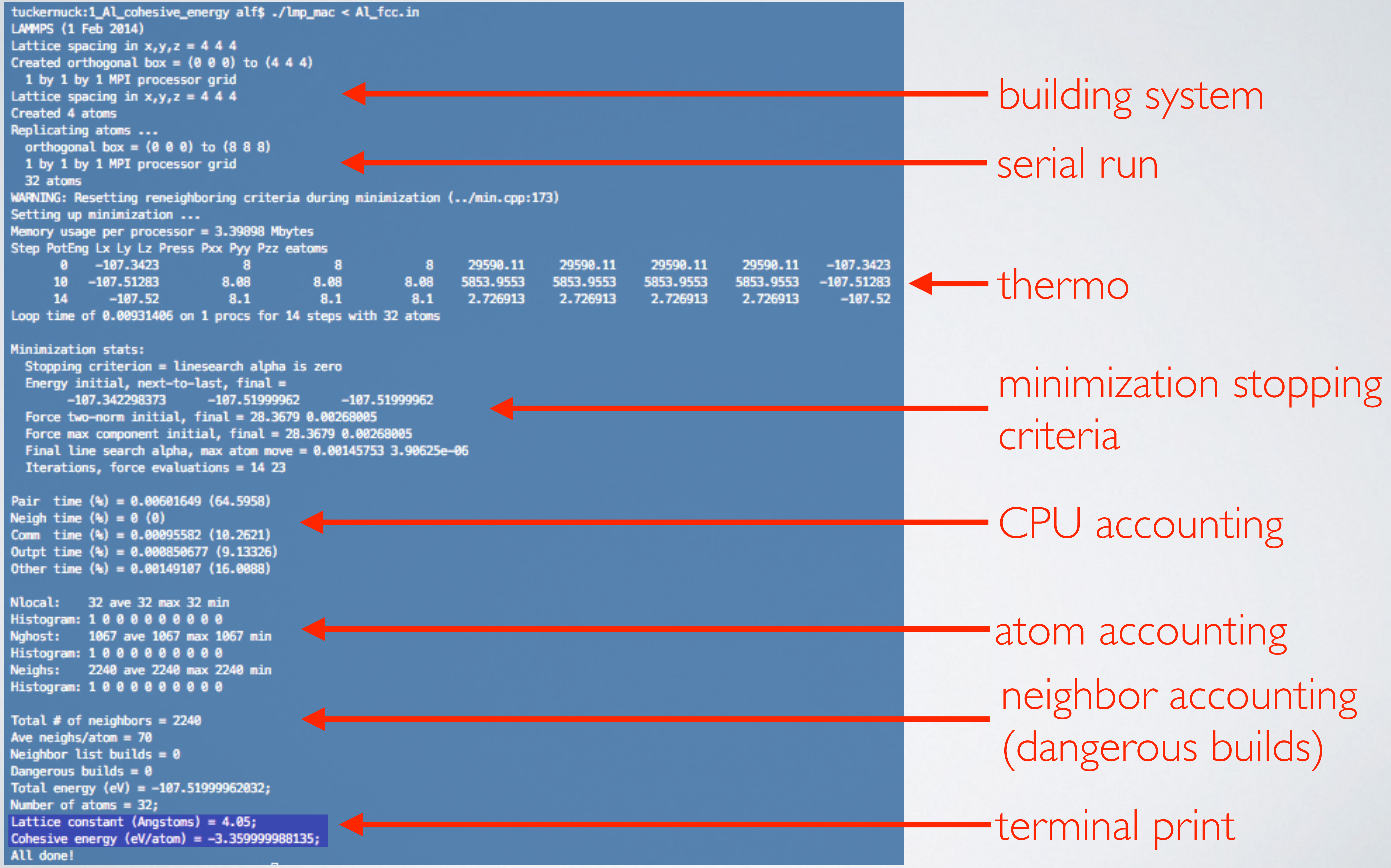


```
# ---------- Initialize Simulation ---------------------
units metal
dimension 3
boundary p p p
atom_style atomic

# ---------- Create Atoms ---------------------
lattice         fcc 4
region  box block 0 1 0 1 0 1 units lattice
create_box      1 box

lattice fcc 4 orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms 1 box
replicate 2 2 2

# ---------- Define Interatomic Potential ---------------------
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
neighbor 2.0 bin
neigh_modify delay 10 check yes

# ---------- Define Settings ---------------------
compute eng all pe/atom
compute eatoms all reduce sum c_eng

# ---------- Dump Options ---------------------
dump            1 all atom 1 dump.relax

# ---------- Run Minimization ---------------------
reset_timestep 0
fix 1 all box/relax iso 0.0 vmax 0.001
thermo 10
thermo_style custom step pe lx ly lz press pxx pyy pzz c_eatoms
min_style cg
minimize 1e-25 1e-25 5000 10000

variable natoms equal "count(all)"
variable teng equal "c_eatoms"
variable a equal "lx/2"
variable ecoh equal "v_teng/v_natoms"

print "Total energy (eV) = ${teng};"
print "Number of atoms = ${natoms};"
print "Lattice constant (Angstoms) = ${a};"
print "Cohesive energy (eV/atom) = ${ecoh};"

print "All done!"
```

- Print terminal output to screen

Tutorial 1: Al cohesive energy

3. Let’s run! `./lmp_mac < Al_fcc.in`



```
tuckernuck:1_Al_cohesive_energy alf$ ./lmp_mac < Al_fcc.in
LAMMPS (1 Feb 2014)
Lattice spacing in x,y,z = 4 4 4
Created orthogonal box = (0 0 0) to (4 4 4)
  1 by 1 by 1 MPI processor grid
Lattice spacing in x,y,z = 4 4 4
Created 4 atoms
Replicating atoms ...
  orthogonal box = (0 0 0) to (8 8 8)
  1 by 1 by 1 MPI processor grid
  32 atoms
WARNING: Resetting reneighboring criteria during minimization (../min.cpp:173)
Setting up minimization ...
Memory usage per processor = 3.39898 Mbytes
Step PotEng Lx Ly Lz Press Pxx Pyy Pzz eatoms
       0    -107.3423          8          8          8   29590.11   29590.11   29590.11   29590.11  -107.3423
      10   -107.51283       8.08       8.08       8.08  5853.9553  5853.9553  5853.9553  5853.9553 -107.51283
      14      -107.52        8.1        8.1        8.1   2.726913   2.726913   2.726913   2.726913    -107.52
Loop time of 0.00931406 on 1 procs for 14 steps with 32 atoms

Minimization stats:
  Stopping criterion = linesearch alpha is zero
  Energy initial, next-to-last, final =
        -107.342298373     -107.51999962     -107.51999962
  Force two-norm initial, final = 28.3679 0.00268005
  Force max component initial, final = 28.3679 0.00268005
  Final line search alpha, max atom move = 0.00145753 3.90625e-06
  Iterations, force evaluations = 14 23

Pair  time (%) = 0.00601649 (64.5958)
Neigh time (%) = 0 (0)
Comm  time (%) = 0.00095582 (10.2621)
Outpt time (%) = 0.000850677 (9.13326)
Other time (%) = 0.00149107 (16.0088)

Nlocal:    32 ave 32 max 32 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    1067 ave 1067 max 1067 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    2240 ave 2240 max 2240 min
Histogram: 1 0 0 0 0 0 0 0 0 0

Total # of neighbors = 2240
Ave neighs/atom = 70
Neighbor list builds = 0
Dangerous builds = 0
Total energy (eV) = -107.51999962032;
Number of atoms = 32;
Lattice constant (Angstoms) = 4.05;
Cohesive energy (eV/atom) = -3.359999988135;
All done!
```

Tutorial 1: Al cohesive energy

4. Analysis

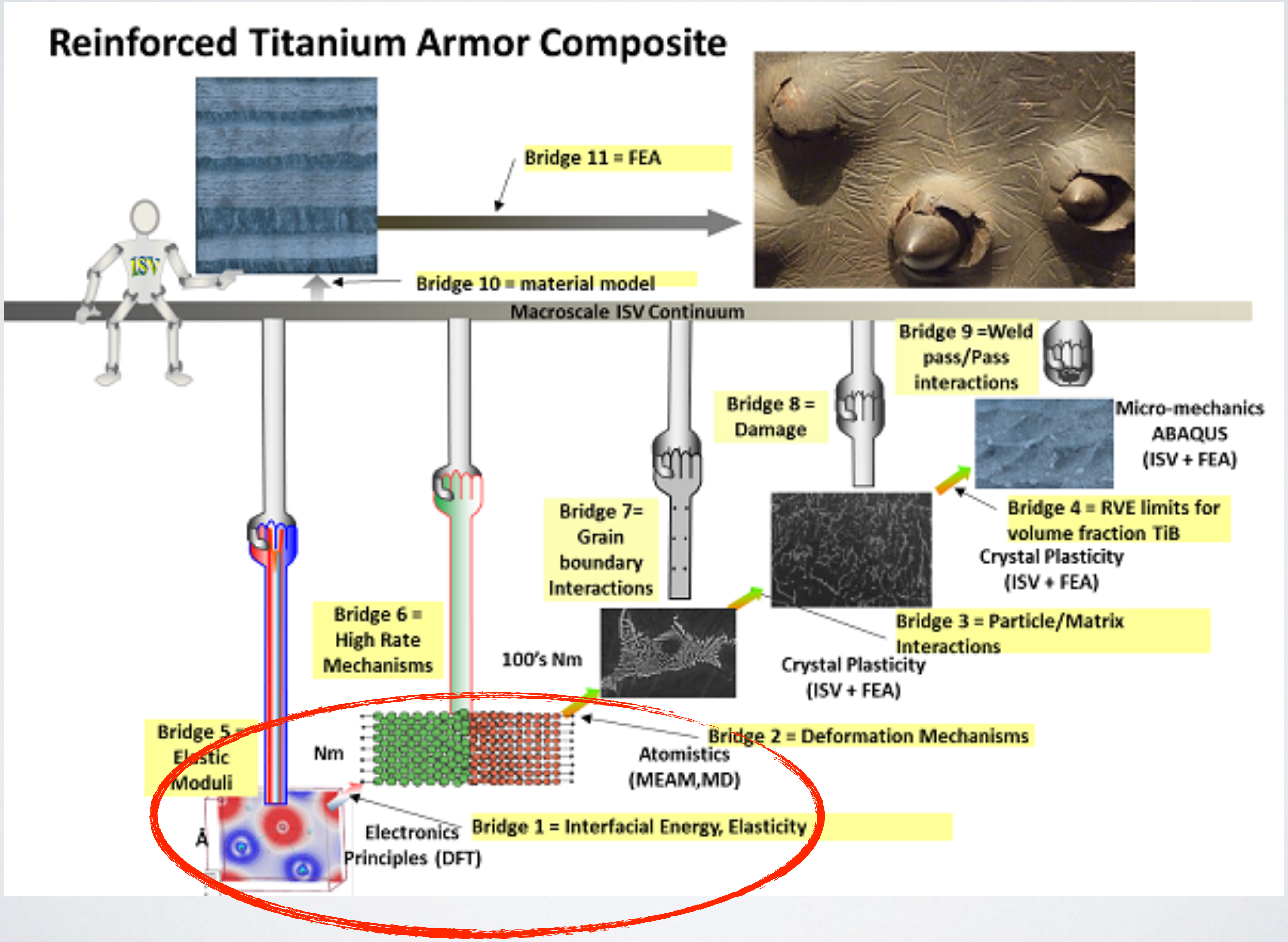
	LAMMPS	Expt.
Lattice constant / Å	4.05	4.0495 *
Cohesive energy / eV/atom	-3.36	-3.39 *

- We should be shocked if these quantities did **not** agree — EAM FF parametrized wrt experimental data
- **Q.** What about if we were studying a new material with experimentally unknown $\mathbf{E_{cohe}}$ and **a**?

*Charles Kittel. Introduction to Solid State Physics, 8th edition. Hoboken, NJ: John Wiley & Sons, Inc, 2005. *http://periodictable.com/Elements/013/data.html

Tutorial 1: Al cohesive energy

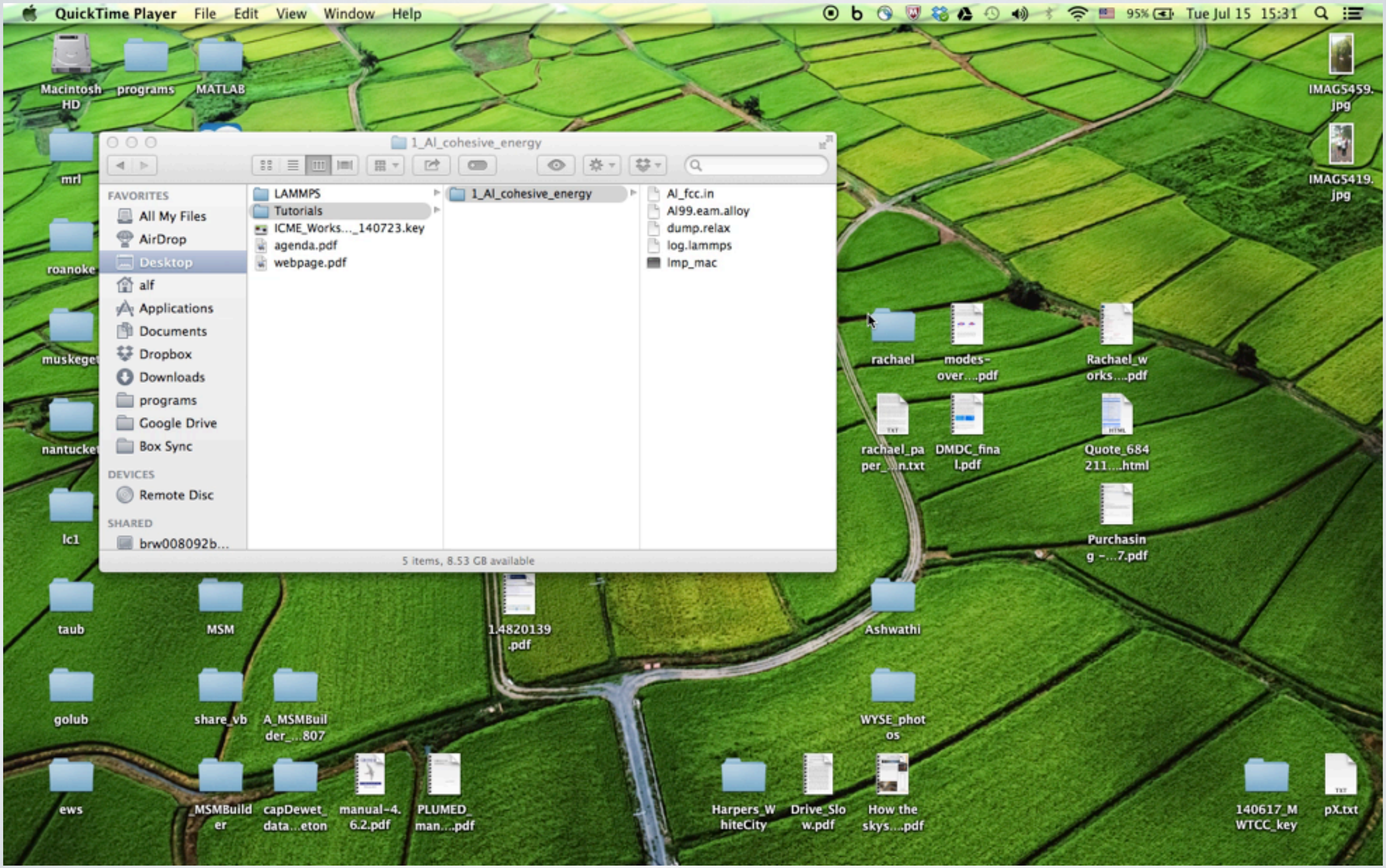
- **A.** ICME!



Reinforced Titanium Armor Composite
Bridge 11 = FEA
ISV
Bridge 10 = material model
Macroscale ISV Continuum
Bridge 9 =Weld pass/Pass interactions
Bridge 8 = Damage
Micro-mechanics ABAQUS (ISV + FEA)
Bridge 7= Grain boundary Interactions
Bridge 4 = RVE limits for volume fraction TiB
Crystal Plasticity (ISV + FEA)
Bridge 6 = High Rate Mechanisms
100's Nm
Bridge 3 = Particle/Matrix Interactions
Crystal Plasticity (ISV + FEA)
Bridge 5 = Elastic Moduli
Nm
Bridge 2 = Deformation Mechanisms
Atomistics (MEAM,MD)
Å
Electronics Principles (DFT)
Bridge 1 = Interfacial Energy, Elasticity

Tutorial 1: Al cohesive energy

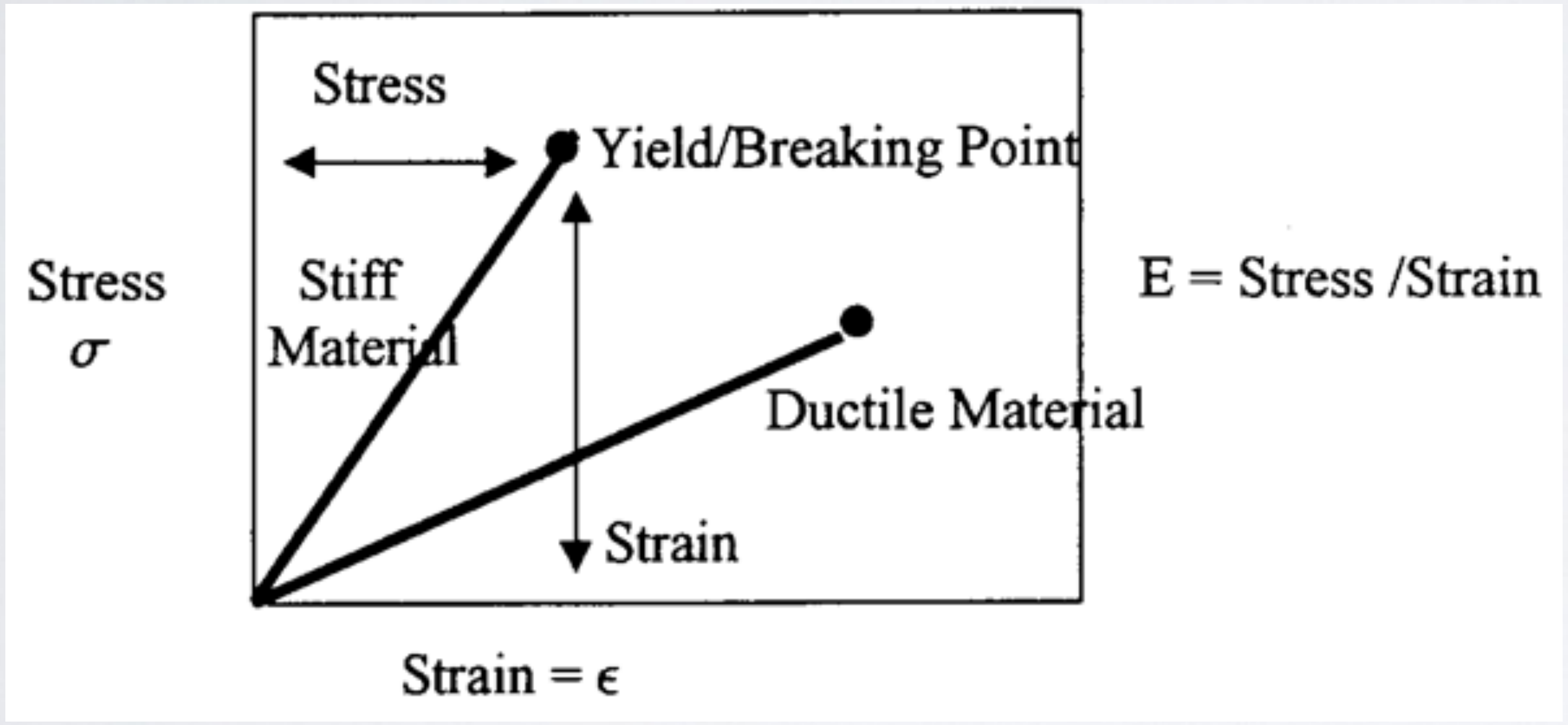
5. Visualization in OVITO



Tutorial 11: Young’s modulus of Al

Tutorial 11: Young's modulus of Al

- OK, but weren't we meant to do MD?
- Right! Now that we can generate an equilibrated Al fcc lattice, let's use LAMMPS to estimate Young's modulus, **E**



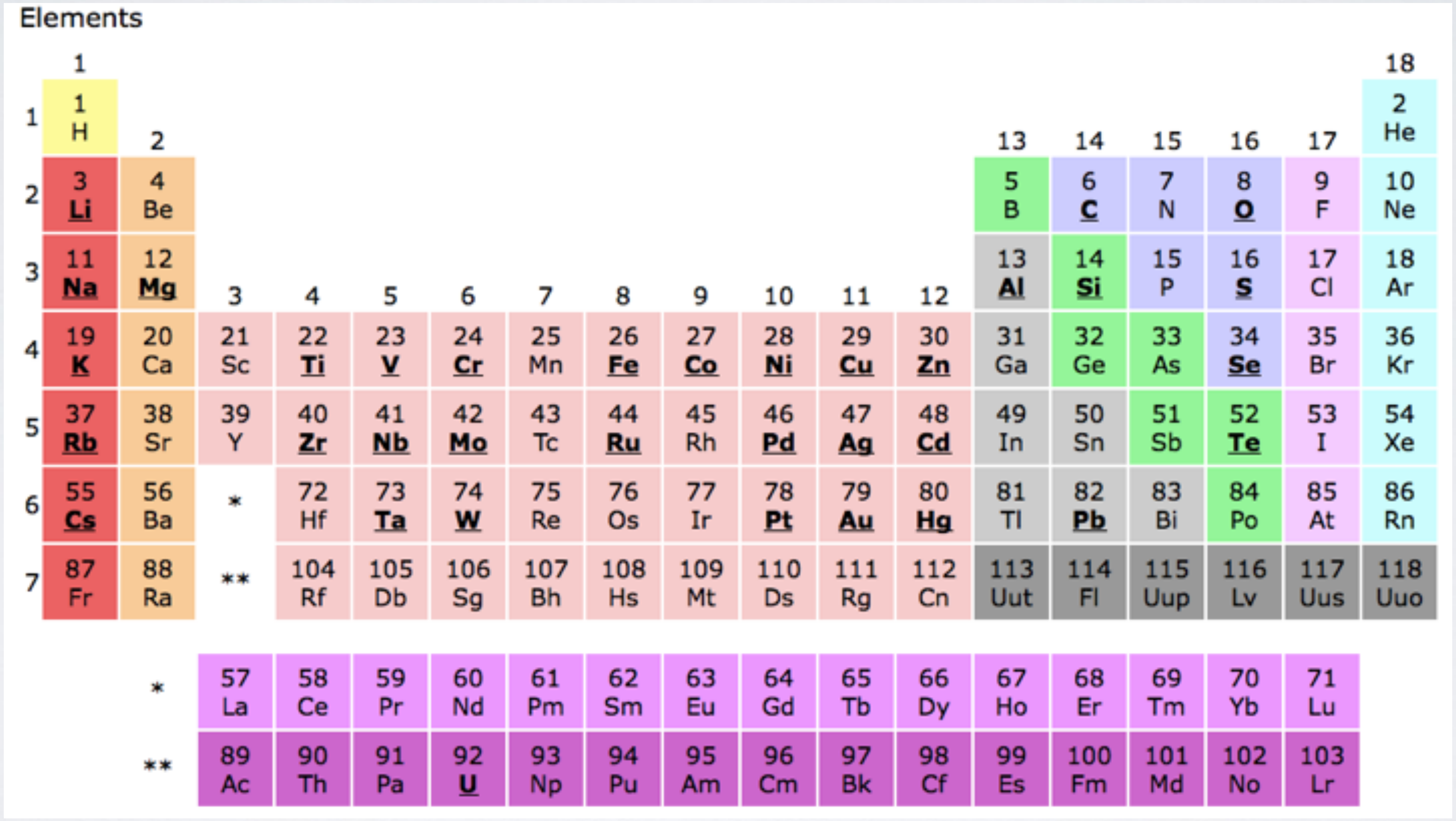
$$E_{Al}^{exptl} = 69 \text{ GPa}^{*}$$

- **Strategy:** Apply an artificial extensional force to a fcc Al xtal and measure stress/strain relationship

*http://www.engineeringtoolbox.com/young-modulus-d_417.html http://www.mbari.org/staff/conn/botany/methods/methods/mmaterial.htm

Tutorial 11: Young's modulus of Al

1. Download **Al99.eam.alloy** EAM potential from NIST Interatomic Potentials Repository Project (http://www.ctcms.nist.gov/potentials)



Elements

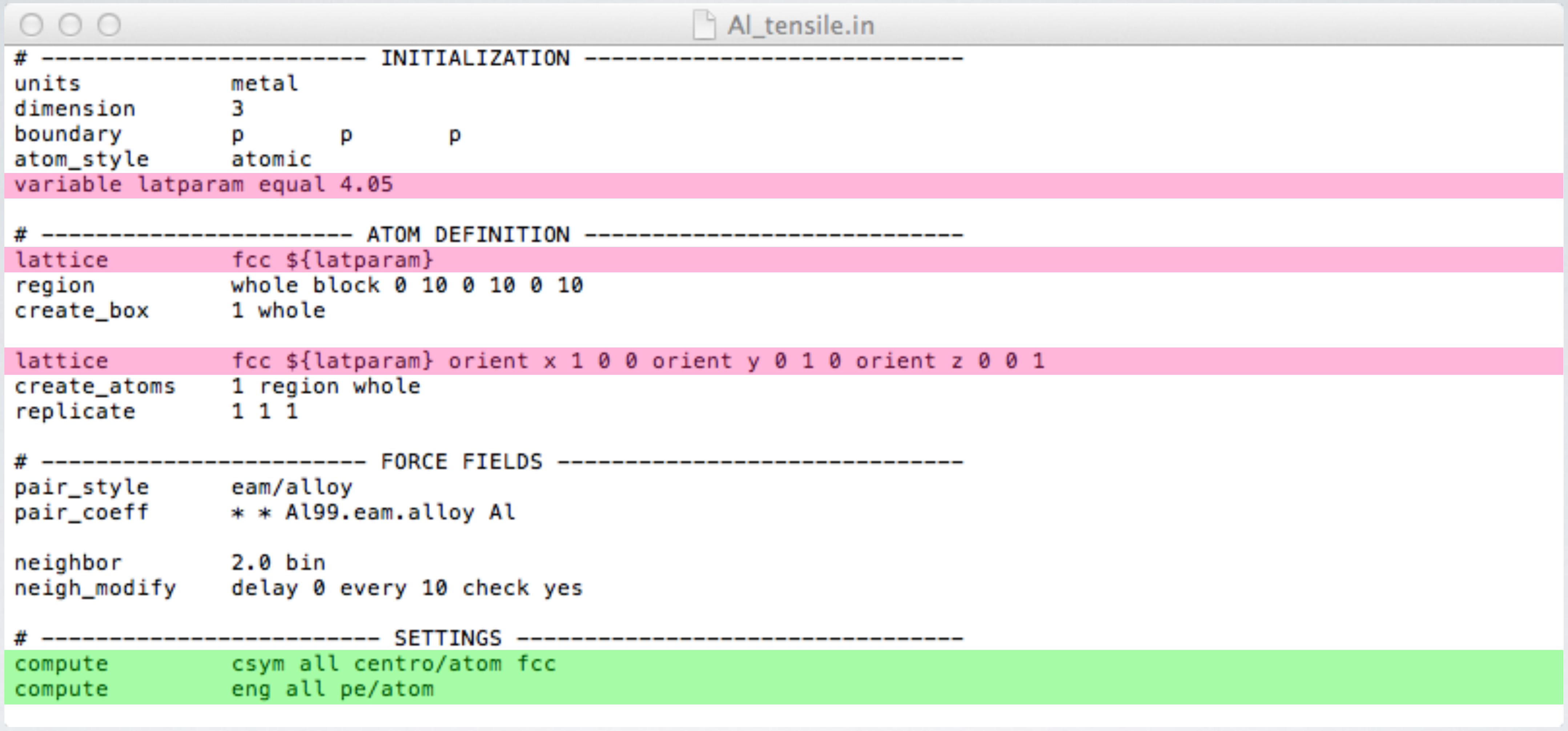
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1 H																	2 He
2	3 **Li**	4 Be											5 B	6 **C**	7 N	8 **O**	9 F	10 Ne
3	11 **Na**	12 **Mg**											13 **Al**	14 **Si**	15 P	16 **S**	17 Cl	18 Ar
4	19 **K**	20 Ca	21 Sc	22 **Ti**	23 **V**	24 **Cr**	25 Mn	26 **Fe**	27 **Co**	28 **Ni**	29 **Cu**	30 **Zn**	31 Ga	32 Ge	33 As	34 **Se**	35 Br	36 Kr
5	37 **Rb**	38 Sr	39 Y	40 **Zr**	41 **Nb**	42 **Mo**	43 Tc	44 **Ru**	45 Rh	46 **Pd**	47 **Ag**	48 **Cd**	49 In	50 Sn	51 Sb	52 **Te**	53 I	54 Xe
6	55 **Cs**	56 Ba	*	72 Hf	73 **Ta**	74 **W**	75 Re	76 Os	77 Ir	78 **Pt**	79 **Au**	80 **Hg**	81 Tl	82 **Pb**	83 Bi	84 Po	85 At	86 Rn
7	87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt	110 Ds	111 Rg	112 Cn	113 Uut	114 Fl	115 Uup	116 Lv	117 Uus	118 Uuo
		*	57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu	
		**	89 Ac	90 Th	91 Pa	92 **U**	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr	

Tutorial 11: Young's modulus of Al

2. Obtain LAMMPS input files **Al_tensile.in**, **Al_eq.m**, and **Al_deform.m** from http://ferguson.matse.illinois.edu/download/Al.zip

Al_tensile.in Al_eq.m Al_deform.m Al99.eam.alloy lmp_mac

Tutorial 11: Young's modulus of Al


Al_tensile.in

```
# ------------------------ INITIALIZATION ----------------------------
units             metal
dimension         3
boundary          p       p       p
atom_style        atomic
variable latparam equal 4.05

# ----------------------- ATOM DEFINITION ----------------------------
lattice           fcc ${latparam}
region            whole block 0 10 0 10 0 10
create_box        1 whole

lattice           fcc ${latparam} orient x 1 0 0 orient y 0 1 0 orient z 0 0 1
create_atoms      1 region whole
replicate         1 1 1

# ------------------------ FORCE FIELDS ------------------------------
pair_style        eam/alloy
pair_coeff        * * Al99.eam.alloy Al

neighbor          2.0 bin
neigh_modify      delay 0 every 10 check yes

# ------------------------- SETTINGS ---------------------------------
compute           csym all centro/atom fcc
compute           eng all pe/atom
```

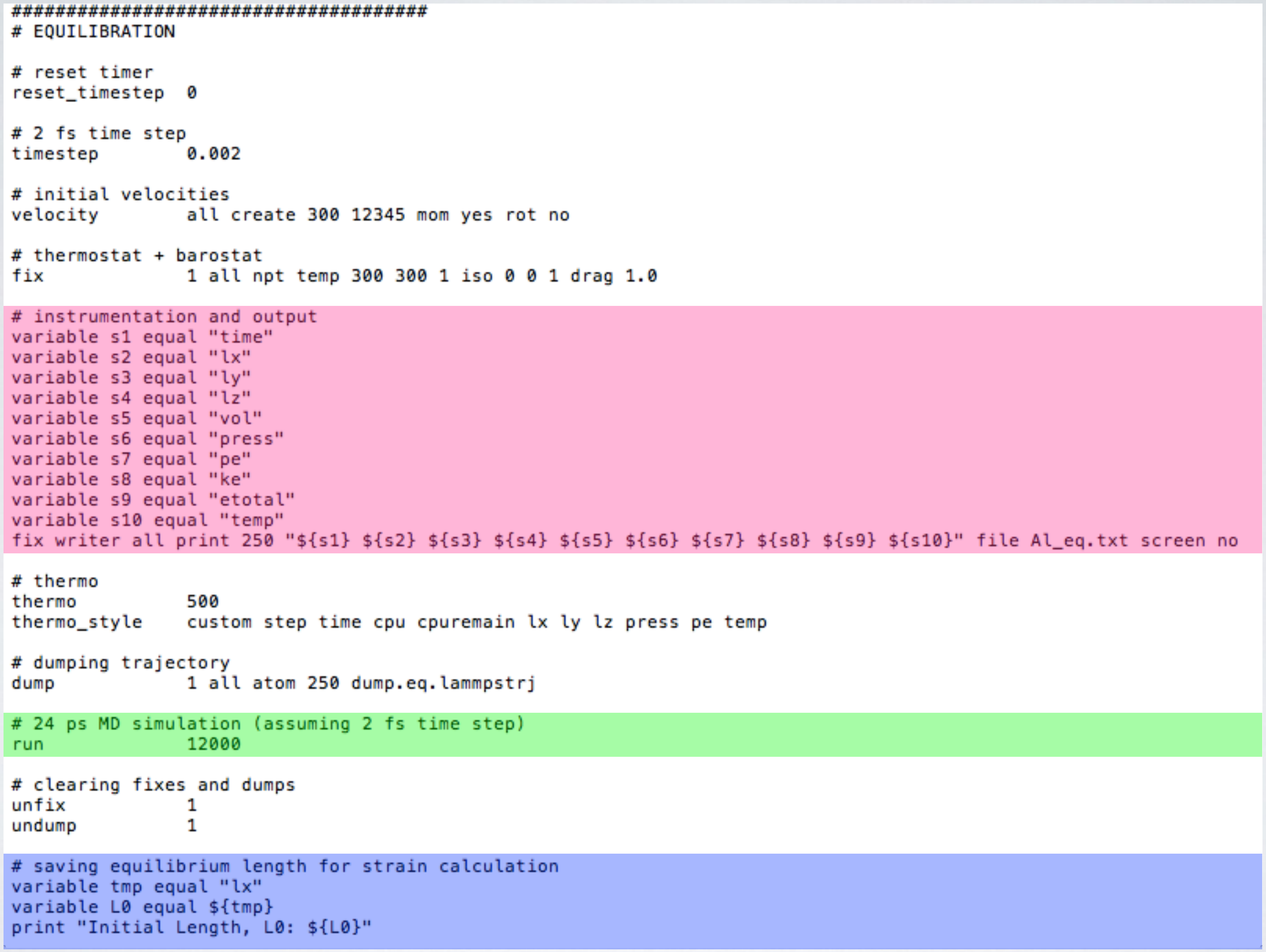
- Set lattice parameter **variable** to **a** = **a$_{eq}$** = 4.05 Å
- Specify two **computes** to calculate **pe/atom** and **centrosymmetry parameter**

$$CS = \sum_{i=1}^{N/2} |\vec{R}_i + \vec{R}_{i+N/2}|^2$$

```
Bulk lattice = 0
Dislocation core ~ 1.0
Stacking faults ~ 5.0
Free surface ~ 23.0
```

http://lammps.sandia.gov/doc/compute_centro_atom.html#Kelchner

Tutorial 11: Young's modulus of Al



```
#########################################
# EQUILIBRATION

# reset timer
reset_timestep  0

# 2 fs time step
timestep        0.002

# initial velocities
velocity        all create 300 12345 mom yes rot no

# thermostat + barostat
fix             1 all npt temp 300 300 1 iso 0 0 1 drag 1.0

# instrumentation and output
variable s1 equal "time"
variable s2 equal "lx"
variable s3 equal "ly"
variable s4 equal "lz"
variable s5 equal "vol"
variable s6 equal "press"
variable s7 equal "pe"
variable s8 equal "ke"
variable s9 equal "etotal"
variable s10 equal "temp"
fix writer all print 250 "${s1} ${s2} ${s3} ${s4} ${s5} ${s6} ${s7} ${s8} ${s9} ${s10}" file Al_eq.txt screen no

# thermo
thermo          500
thermo_style    custom step time cpu cpuremain lx ly lz press pe temp

# dumping trajectory
dump            1 all atom 250 dump.eq.lammpstrj

# 24 ps MD simulation (assuming 2 fs time step)
run             12000

# clearing fixes and dumps
unfix           1
undump          1

# saving equilibrium length for strain calculation
variable tmp equal "lx"
variable L0 equal ${tmp}
print "Initial Length, L0: ${L0}"
```

- Instrumentation, perform MD integration with Verlet (default) algorithm, and record terminal relaxed box size

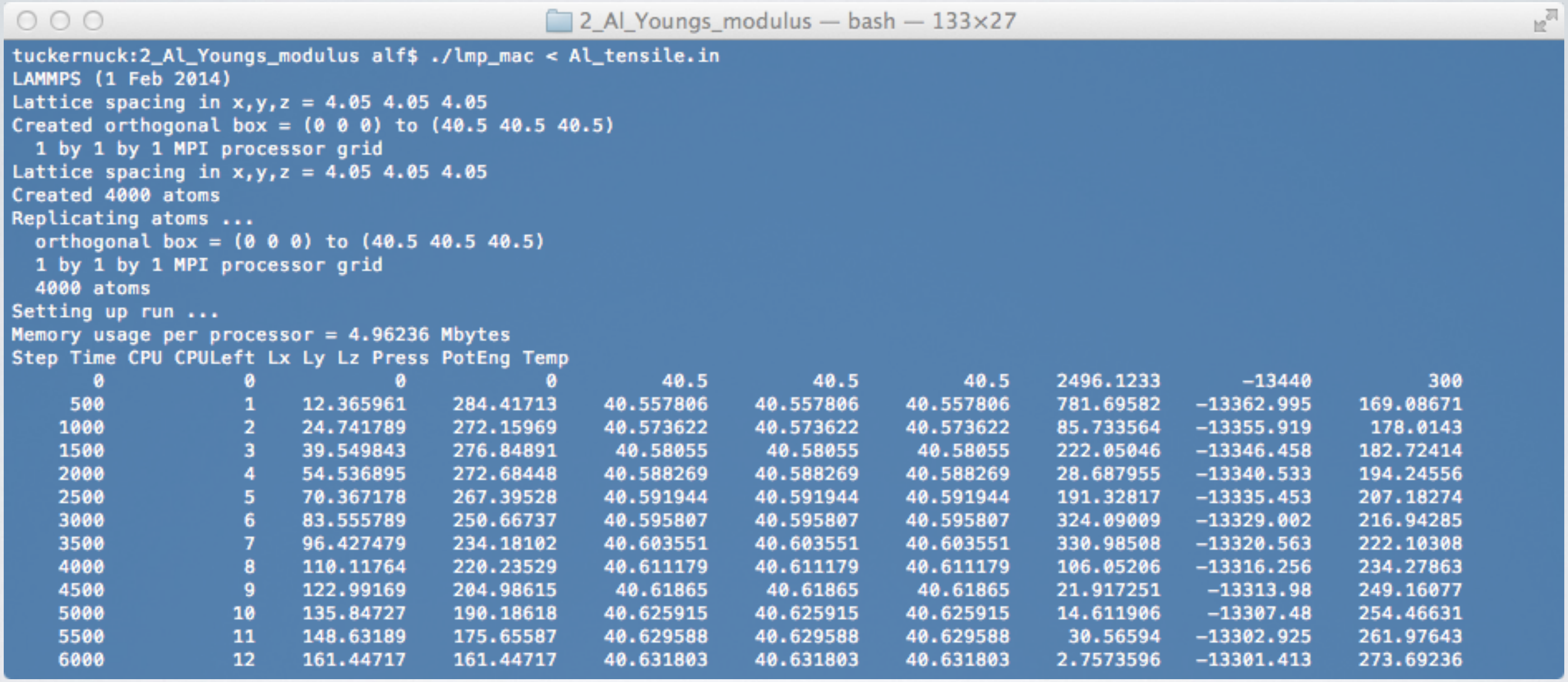
Tutorial 11: Young's modulus of Al

```
##########################################
# DEFORMATION

# reset timer
reset_timestep  0

# 2 fs time step
timestep 0.002

# thermostat + barostat
fix             1 all npt temp 300 300 1 y 0 0 1 z 0 0 1 drag 1.0

# nonequilibrium straining in x-direction at strain rate = 1x10^10 / s = 1x10^-2 / ps in units metal
variable srate equal 1.0e10
variable srate1 equal "v_srate / 1.0e12"
fix             2 all deform 1 x erate ${srate1} units box remap x

# instrumentation and output
# for units metal, pressure is in [bars] = 100 [kPa] = 1/10000 [GPa] => p2, p3, p4 are in GPa
variable strain equal "(lx - v_L0)/v_L0"
variable p1 equal "v_strain"
variable p2 equal "-pxx/10000"
variable p3 equal "-pyy/10000"
variable p4 equal "-pzz/10000"
fix writer all print 125 "${p1} ${p2} ${p3} ${p4}" file Al_deform.txt screen no

# thermo
thermo          500
thermo_style    custom step cpuremain v_strain v_p2 v_p3 v_p4 press pe temp

# dumping standard atom trajectories
dump            1 all atom 125 dump.deform.lammpstrj

# dumping custom cfg files containing coords + ancillary variables
dump            2 all cfg 125 dump.deform_*.cfg mass type xs ys zs c_csym c_eng fx fy fz
dump_modify     2 element Al


# 20 ps MD simulation (assuming 2 fs time step)
run             10000

# clearing fixes and dumps
unfix           1
unfix           2
unfix           writer
undump          1
undump          2


##########################################
# SIMULATION DONE
print "All done!"
```

- Nonequilibrium straining, instrumentation, and cfg trajectory dump

Tutorial II: Young's modulus of Al

3. Let's run! `./lmp_mac < Al_tensile.in`



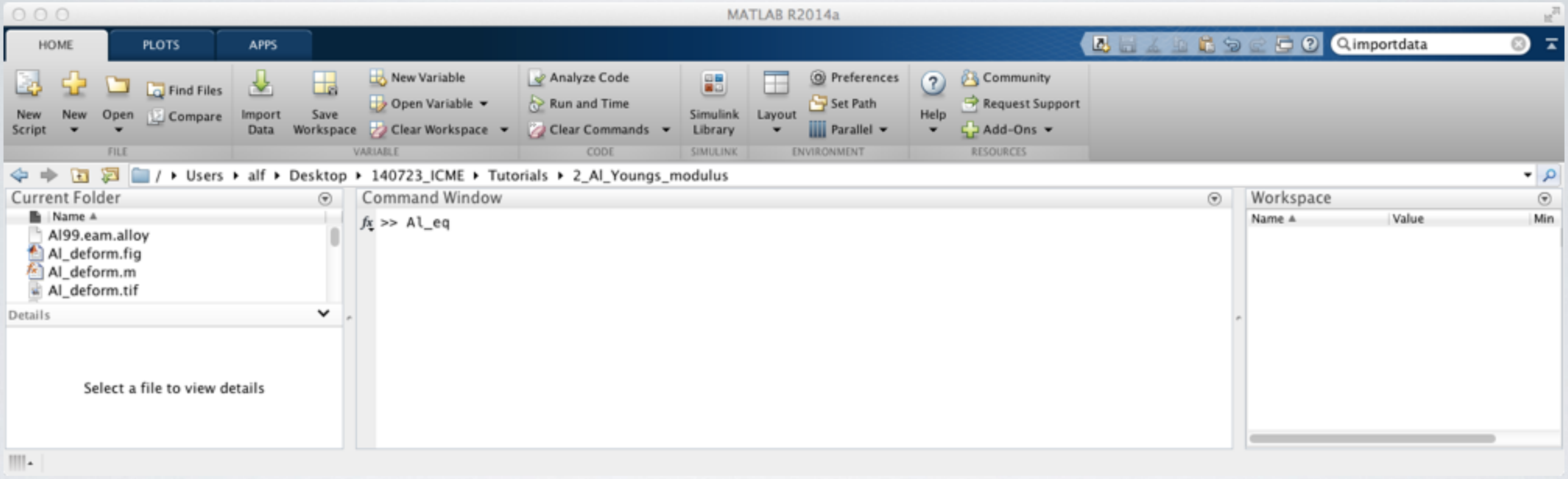
```
tuckernuck:2_Al_Youngs_modulus alf$ ./lmp_mac < Al_tensile.in
LAMMPS (1 Feb 2014)
Lattice spacing in x,y,z = 4.05 4.05 4.05
Created orthogonal box = (0 0 0) to (40.5 40.5 40.5)
  1 by 1 by 1 MPI processor grid
Lattice spacing in x,y,z = 4.05 4.05 4.05
Created 4000 atoms
Replicating atoms ...
  orthogonal box = (0 0 0) to (40.5 40.5 40.5)
  1 by 1 by 1 MPI processor grid
  4000 atoms
Setting up run ...
Memory usage per processor = 4.96236 Mbytes
Step Time CPU CPULeft Lx Ly Lz Press PotEng Temp
       0            0            0            0         40.5         40.5         40.5    2496.1233       -13440          300
     500            1    12.365961    284.41713    40.557806    40.557806    40.557806    781.69582   -13362.995    169.08671
    1000            2    24.741789    272.15969    40.573622    40.573622    40.573622    85.733564   -13355.919     178.0143
    1500            3    39.549843    276.84891     40.58055     40.58055     40.58055    222.05046   -13346.458    182.72414
    2000            4    54.536895    272.68448    40.588269    40.588269    40.588269    28.687955   -13340.533    194.24556
    2500            5    70.367178    267.39528    40.591944    40.591944    40.591944    191.32817   -13335.453    207.18274
    3000            6    83.555789    250.66737    40.595807    40.595807    40.595807    324.09009   -13329.002    216.94285
    3500            7    96.427479    234.18102    40.603551    40.603551    40.603551    330.98508   -13320.563    222.10308
    4000            8    110.11764    220.23529    40.611179    40.611179    40.611179    106.05206   -13316.256    234.27863
    4500            9    122.99169    204.98615     40.61865     40.61865     40.61865    21.917251    -13313.98    249.16077
    5000           10    135.84727    190.18618    40.625915    40.625915    40.625915    14.611906    -13307.48    254.46631
    5500           11    148.63189    175.65587    40.629588    40.629588    40.629588     30.56594   -13302.925    261.97643
    6000           12    161.44717    161.44717    40.631803    40.631803    40.631803    2.7573596   -13301.413    273.69236
```

N.B. This could take 8-10 minutes if your machine is old and slow (like mine)
Speed things up by reducing system size by factor of 2^3 in **Al_tensile.in**:

```
region      whole block 0 5 0 5 0 5
```

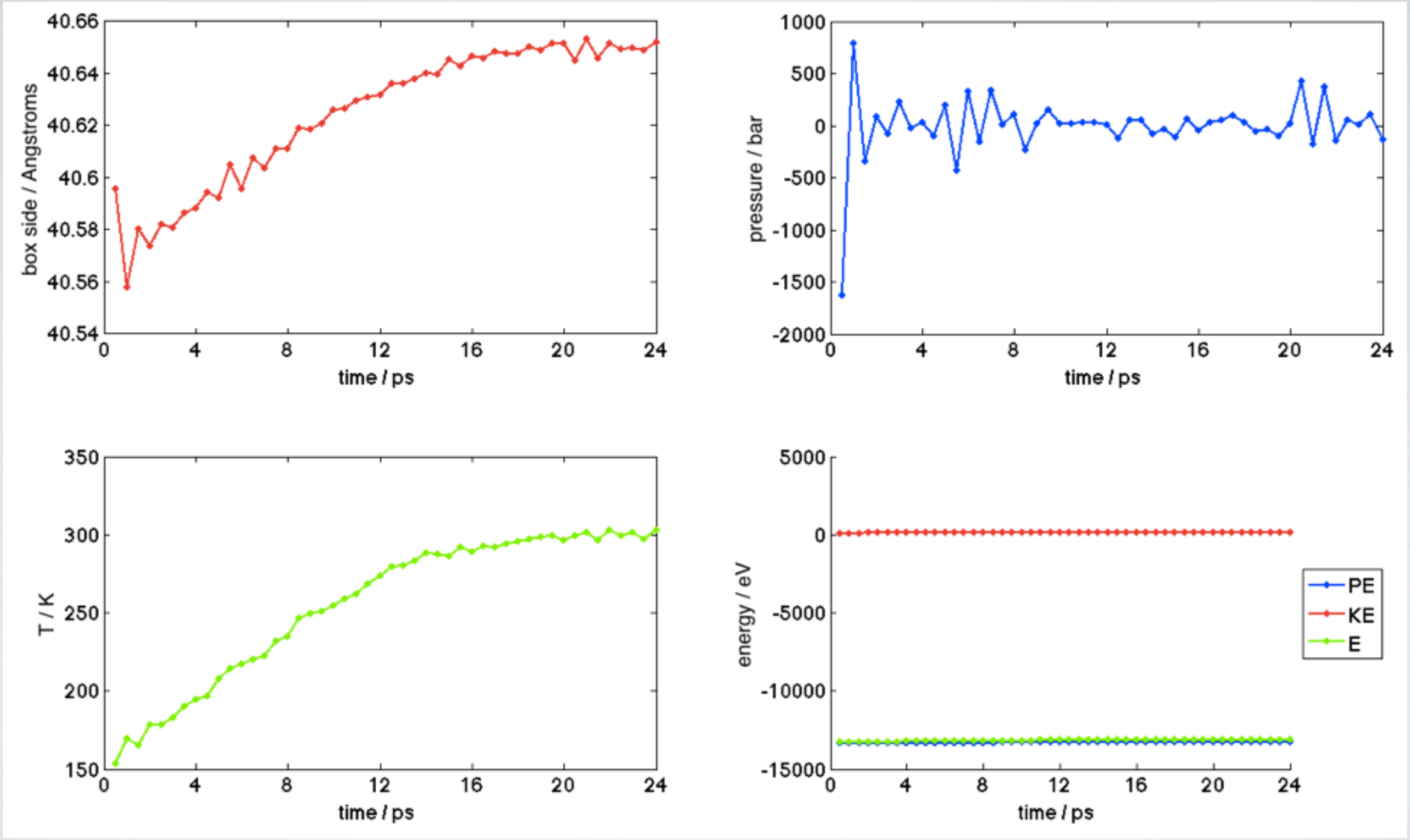
Tutorial II: Young's modulus of Al

4. Analyze approach to equilibration using **Al_eq.m**



```
>> Al_eq
```

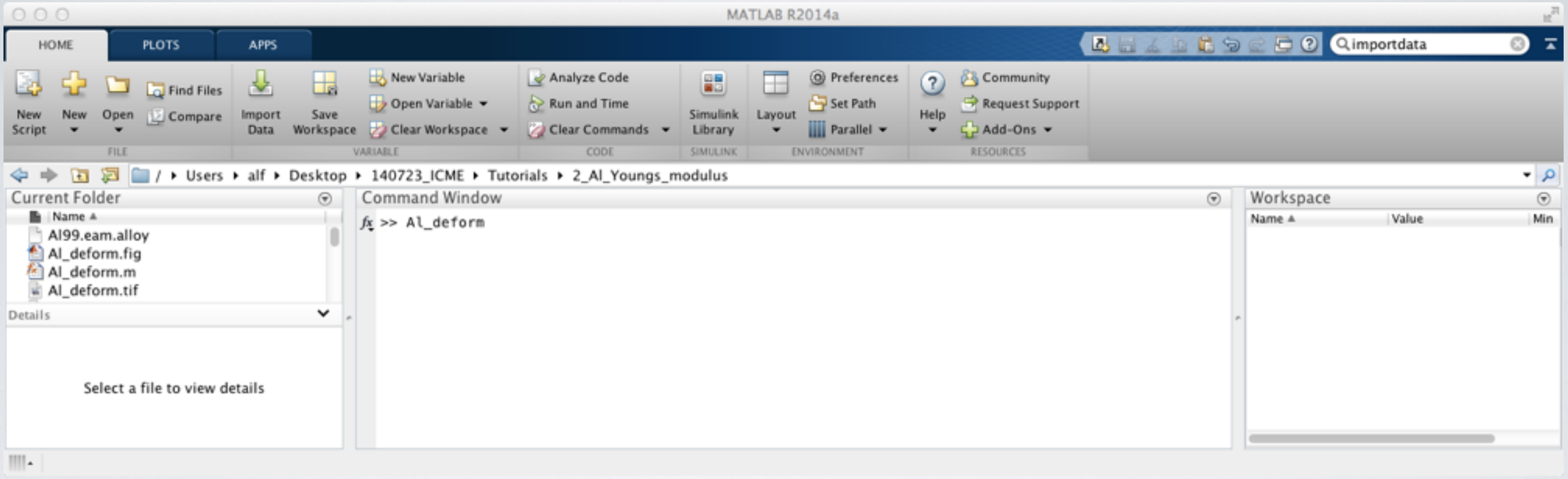
Tutorial 11: Young's modulus of Al



• Equilibrium attained in ~24 ps

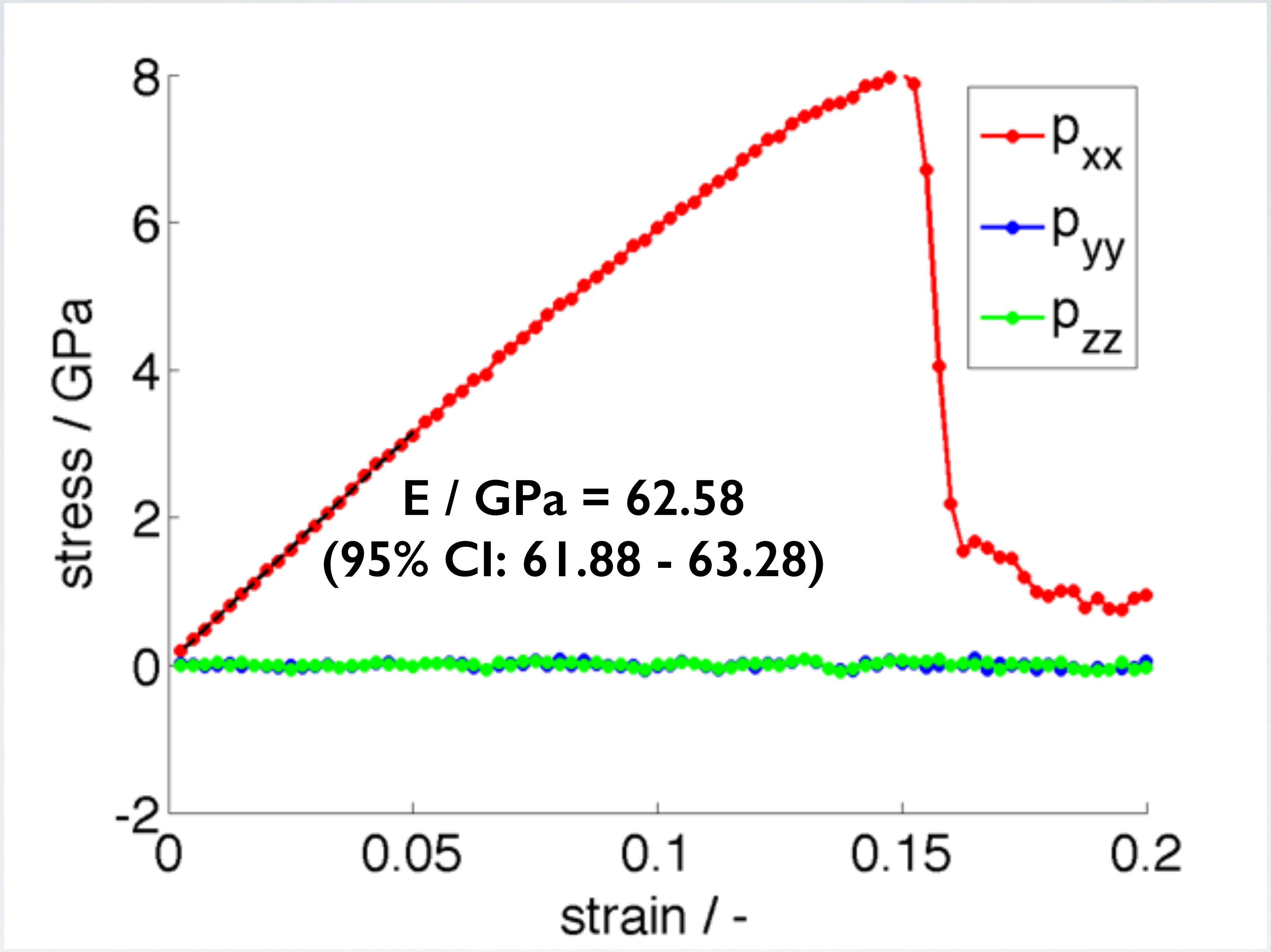
Tutorial II: Young's modulus of Al

5. Analyze deformation and estimate **E** using **Al_deform.m**



```
>> Al_deform
```

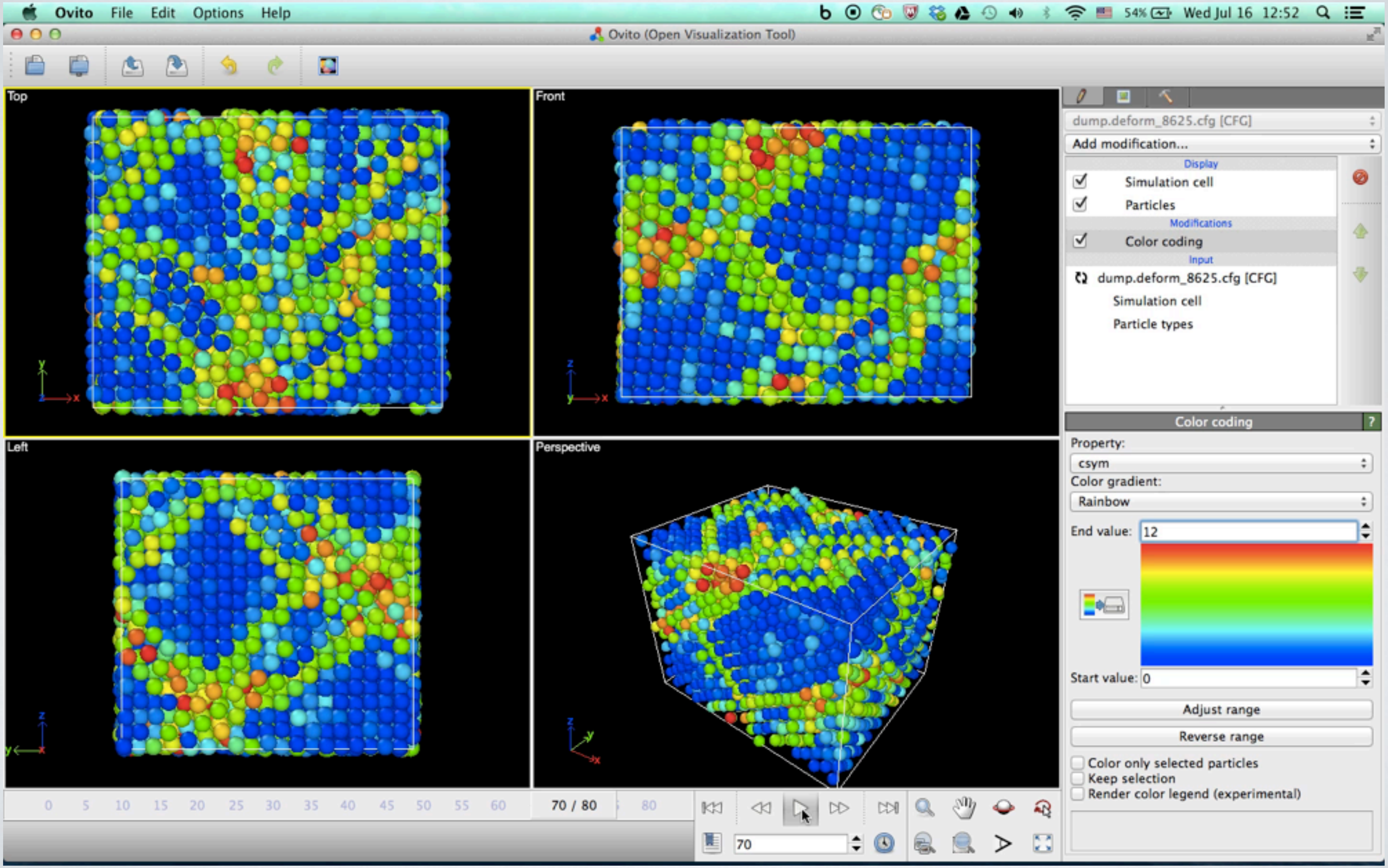
Tutorial 11: Young's modulus of Al



- Onset of homogeneous dislocation nucleation and end of elastic deformation at ~8 GPa
- **E** estimated by slope of linear fit over strain range [0-0.05]

Tutorial 11: Young's modulus of Al

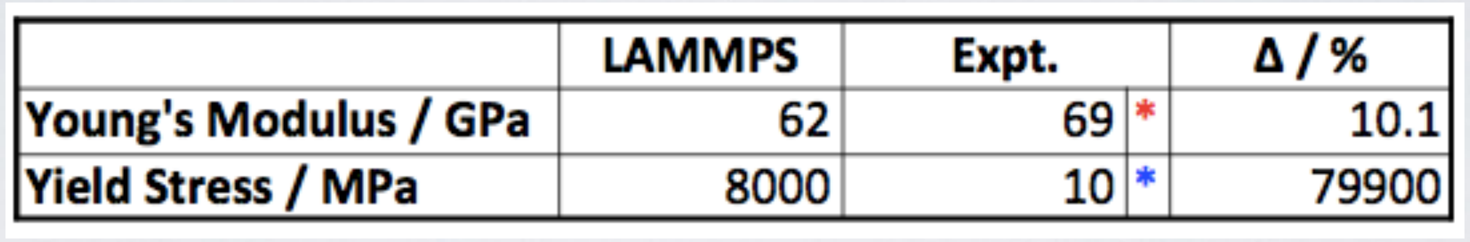
6. Visualization of deformation in OVITO



Property: csym
Color gradient: Rainbow
End value: 12
Start value: 0

Tutorial 11: Young’s modulus of Al

7. Comparison to experiment



	LAMMPS	Expt.		Δ / %
Young's Modulus / GPa	62	69	*	10.1
Yield Stress / MPa	8000	10	*	79900

Young’s Modulus

- We did pretty well, **E** within ±10%
- A rigorous study would check **E** convergence as a function of **system size**

Yield Stress

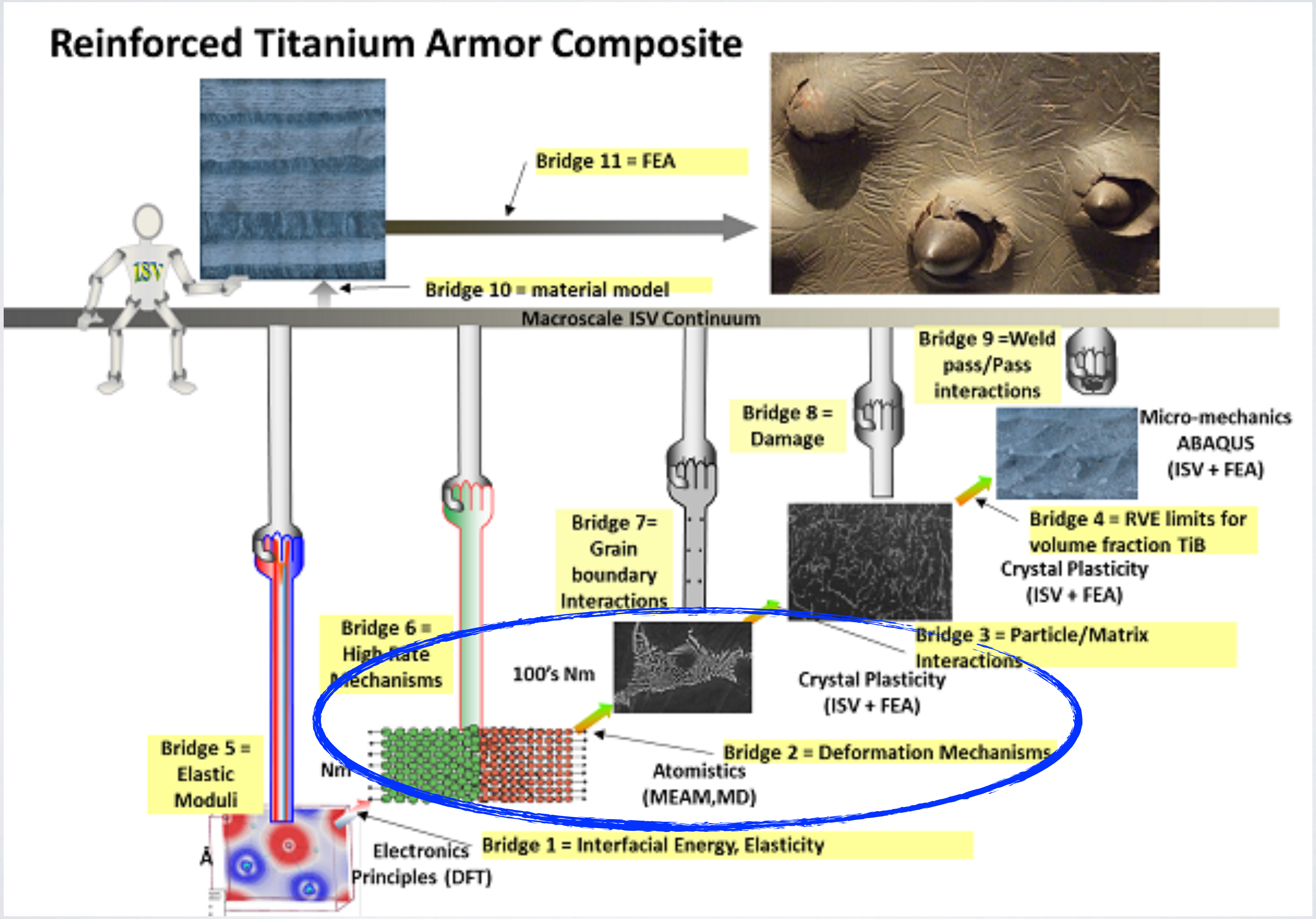
- Our estimate for yield stress is horrible! Off by ~3 orders of magnitude!
- Why did we do so badly? — We have a **perfect crystal**, *homogeneous* vs. *heterogeneous* nucleation.

*http://www.engineeringtoolbox.com/young-modulus-d_417.html

*Polmear, I. J. (1995). Light Alloys: Metallurgy of the Light Metals (3rd ed.). Butterworth-Heinemann

Tutorial 11: Young's modulus of Al

- **ICME** — for an experimentally uncharacterized material can "bridge up" MD **E** estimate to FEA model


Reinforced Titanium Armor Composite
Bridge 11 = FEA
ISV
Bridge 10 = material model
Macroscale ISV Continuum
Bridge 9 =Weld pass/Pass interactions
Bridge 8 = Damage
Micro-mechanics ABAQUS (ISV + FEA)
Bridge 7= Grain boundary Interactions
Bridge 4 = RVE limits for volume fraction TiB
Crystal Plasticity (ISV + FEA)
Bridge 6 = High Rate Mechanisms
100's Nm
Bridge 3 = Particle/Matrix Interactions
Crystal Plasticity (ISV + FEA)
Bridge 5 = Elastic Moduli
Nm
Bridge 2 = Deformation Mechanisms
Atomistics (MEAM,MD)
Å
Electronics Principles (DFT)
Bridge 1 = Interfacial Energy, Elasticity

Questions?