MODULE 2: MOLECULAR DYNAMICS

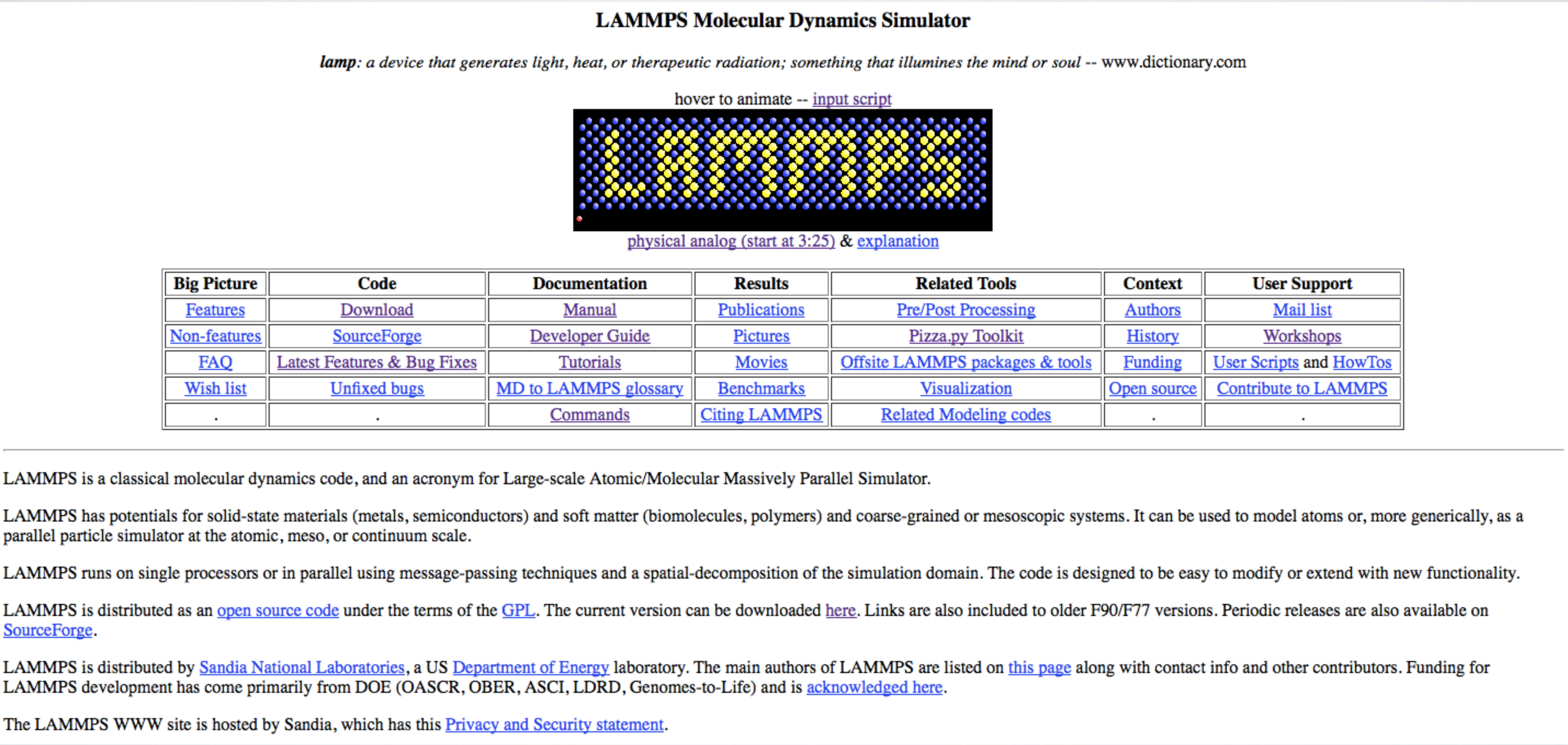
Practice: LAMMPS

I. What is 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.

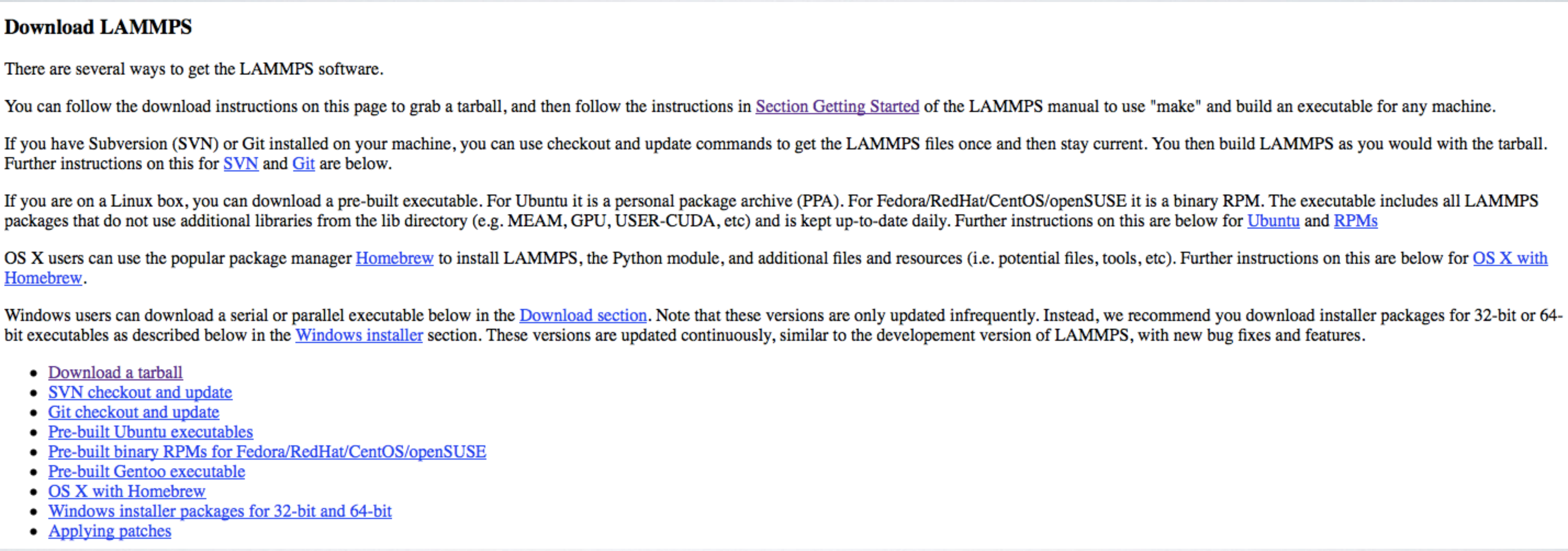
The LAMMPS WWW site is hosted by Sandia, which has this Privacy and Security statement.

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**

Installation

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



Download LAMMPS

There are several ways to get the LAMMPS software.

You can follow the download instructions on this page to grab a tarball, and then follow the instructions in Section Getting Started of the LAMMPS manual to use "make" and build an executable for any machine.

If you have Subversion (SVN) or Git installed on your machine, you can use checkout and update commands to get the LAMMPS files once and then stay current. You then build LAMMPS as you would with the tarball. Further instructions on this for SVN and Git are below.

If you are on a Linux box, you can download a pre-built executable. For Ubuntu it is a personal package archive (PPA). For Fedora/RedHat/CentOS/openSUSE it is a binary RPM. The executable includes all LAMMPS packages that do not use additional libraries from the lib directory (e.g. MEAM, GPU, USER-CUDA, etc) and is kept up-to-date daily. Further instructions on this are below for Ubuntu and RPMs

OS X users can use the popular package manager Homebrew to install LAMMPS, the Python module, and additional files and resources (i.e. potential files, tools, etc). Further instructions on this are below for OS X with Homebrew.

Windows users can download a serial or parallel executable below in the Download section. Note that these versions are only updated infrequently. Instead, we recommend you download installer packages for 32-bit or 64-bit executables as described below in the Windows installer section. These versions are updated continuously, similar to the developement version of LAMMPS, with new bug fixes and features.

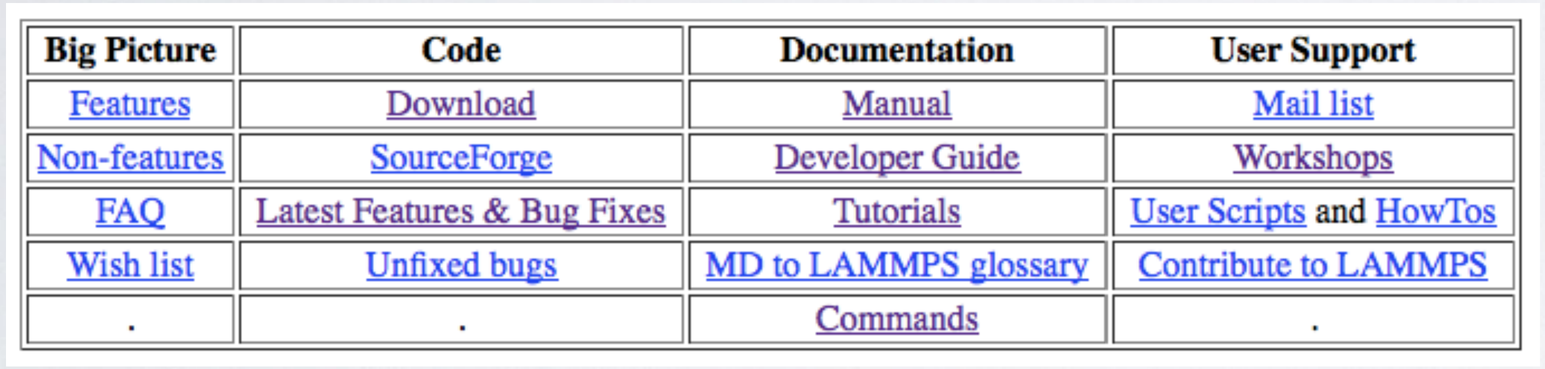
- Download a tarball
- SVN checkout and update
- Git checkout and update
- Pre-built Ubuntu executables
- Pre-built binary RPMs for Fedora/RedHat/CentOS/openSUSE
- Pre-built Gentoo executable
- OS X with Homebrew
- Windows installer packages for 32-bit and 64-bit
- Applying patches

- Platforms: Linux, Mac, Windows
- Format: exe, RPM, PPA, SVN, Git, Homebrew, tarball

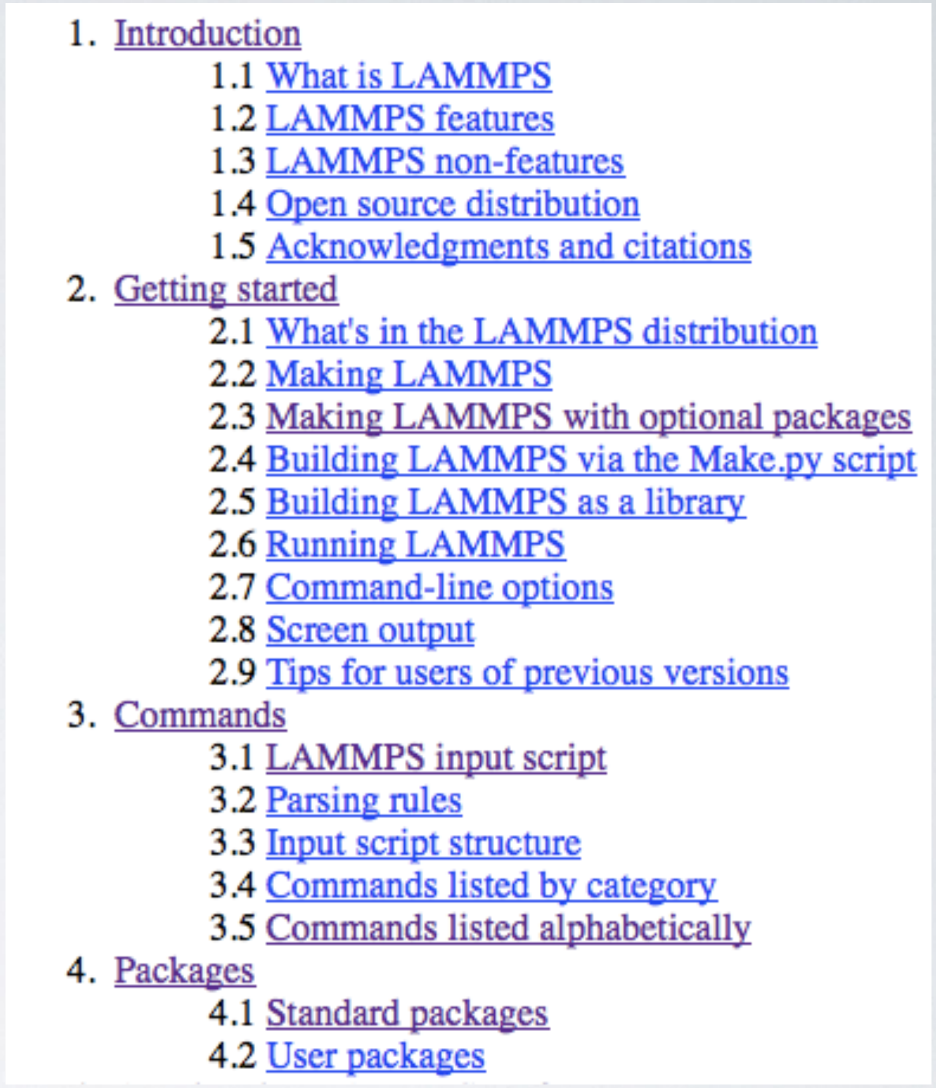
Documentation

- 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)

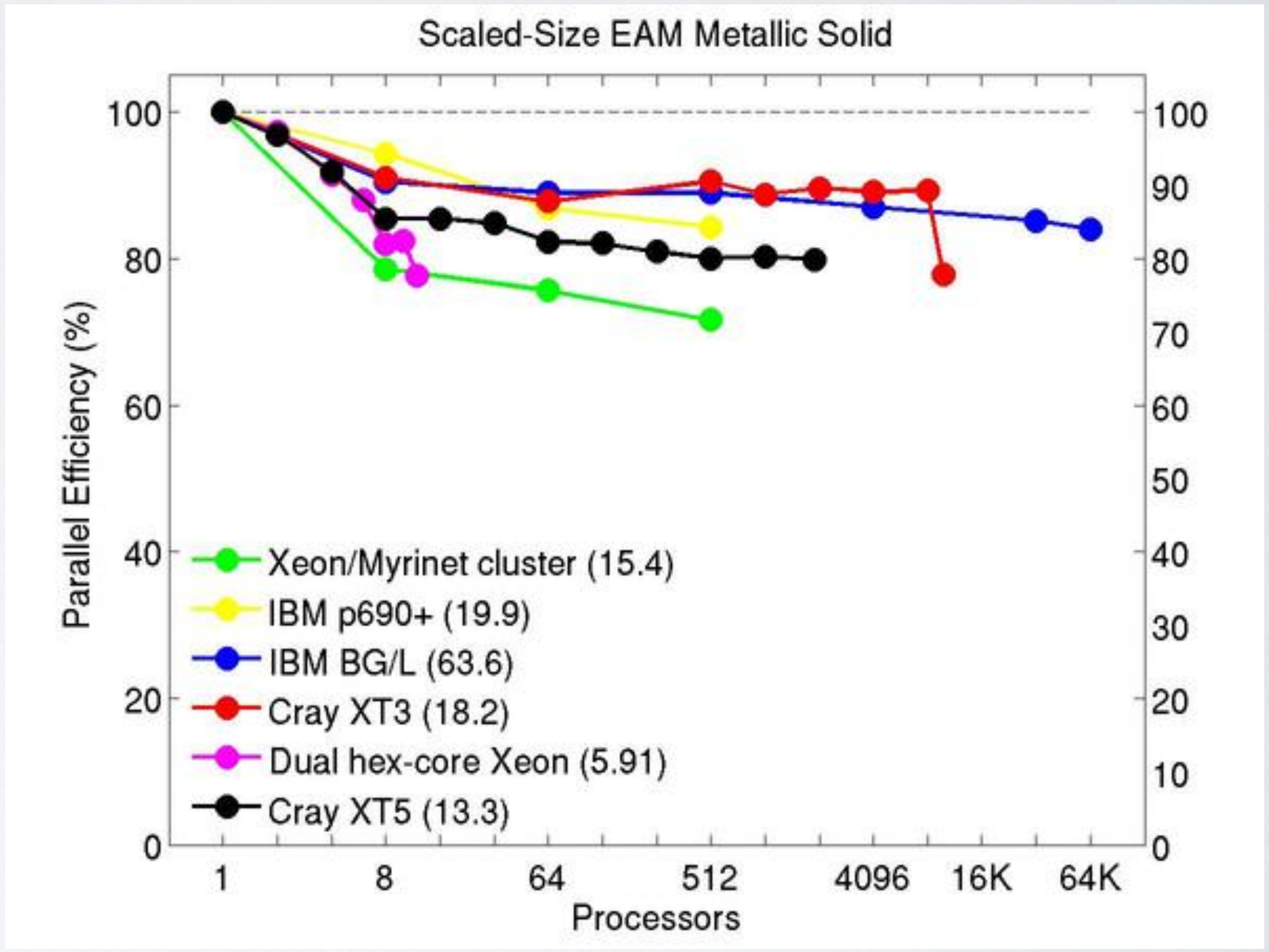
- Excellent third-party tutorials hosted by CAVS @ MSU
  (https://icme.hpc.msstate.edu/mediawiki/index.php/LAMMPS_tutorials)

Performance

- Extremely fast, efficient, and scalable
  - MPI parallelism
  - GPU CUDA support for most routines

Weak scaling parallel efficiency:

- Cu metallic solid w/ EAM potential
- 32k atoms per processor
  (e.g., 64k proc run = 2 billion atoms)
- $\mathrm{P.E.}(N) = \dfrac{t_1}{t_N}$
- Single processor timings in seconds in parentheses



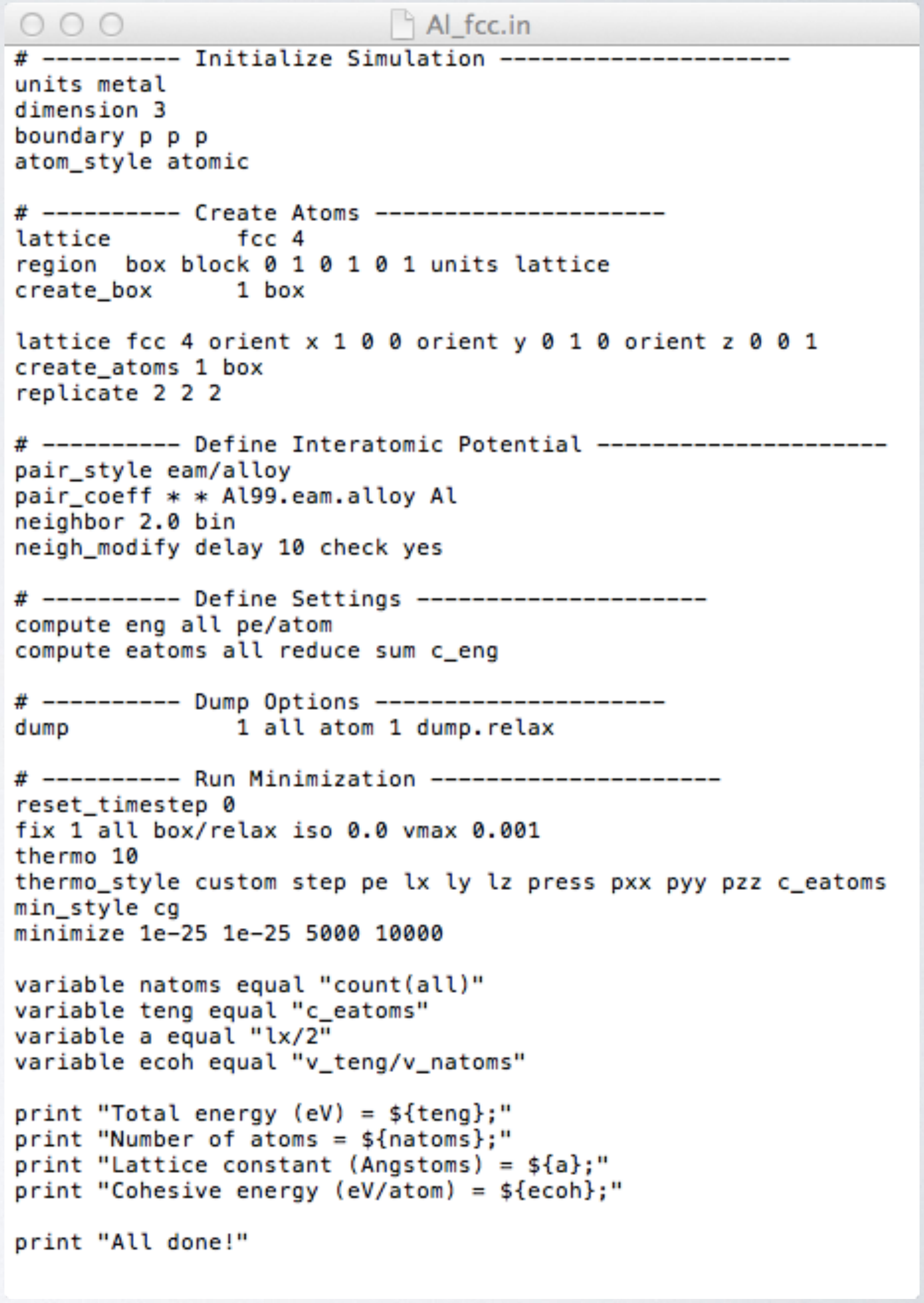
II. Running LAMMPS

Usability

- Run initialization and control via **input script**

- Call from command line as `./lmp < in.comp`

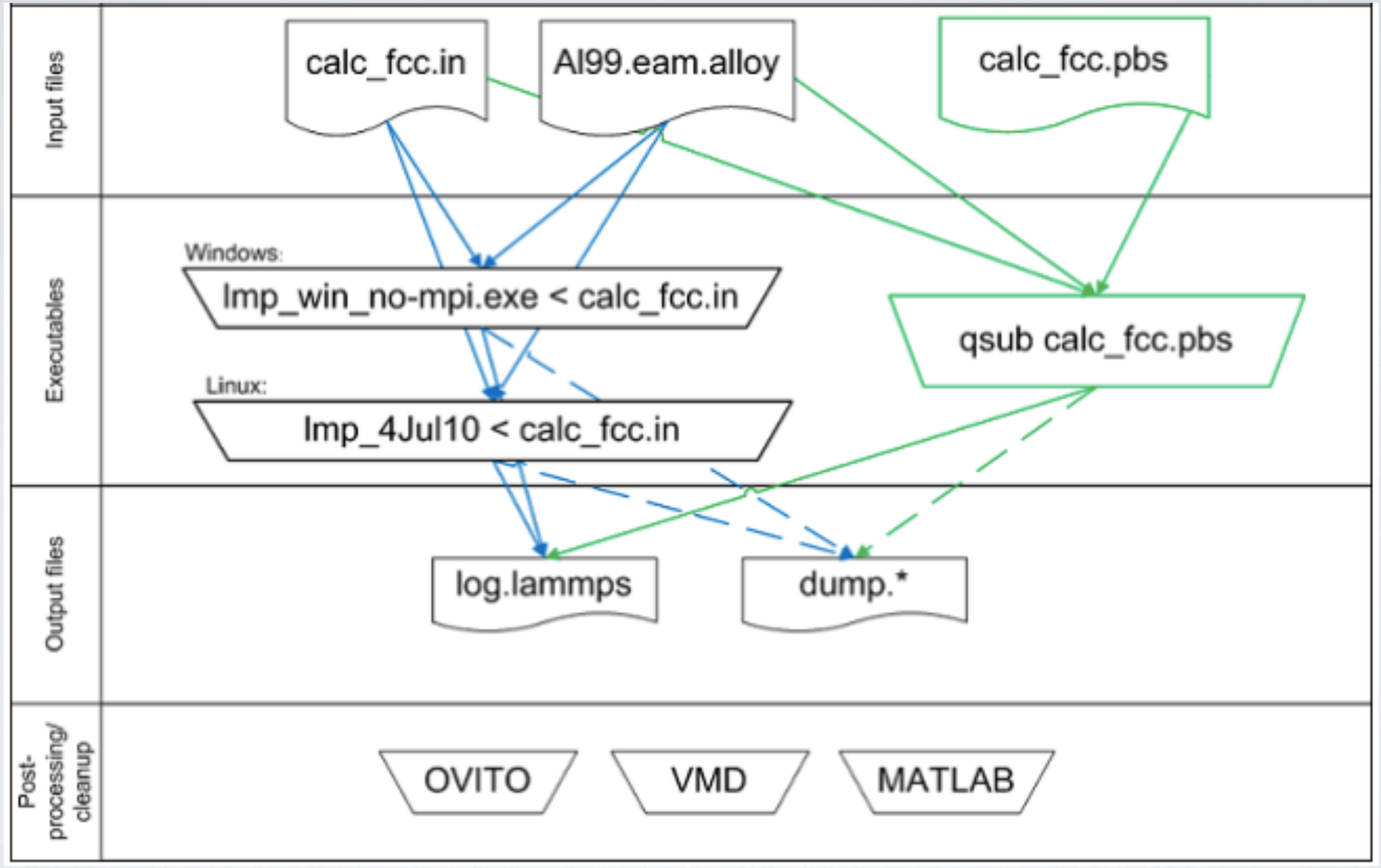
- **No GUI**, but some python tools available (http://lammps.sandia.gov/doc/Section_python.html)


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!"
```

Anatomy of a LAMMPS simulation

local run batch job



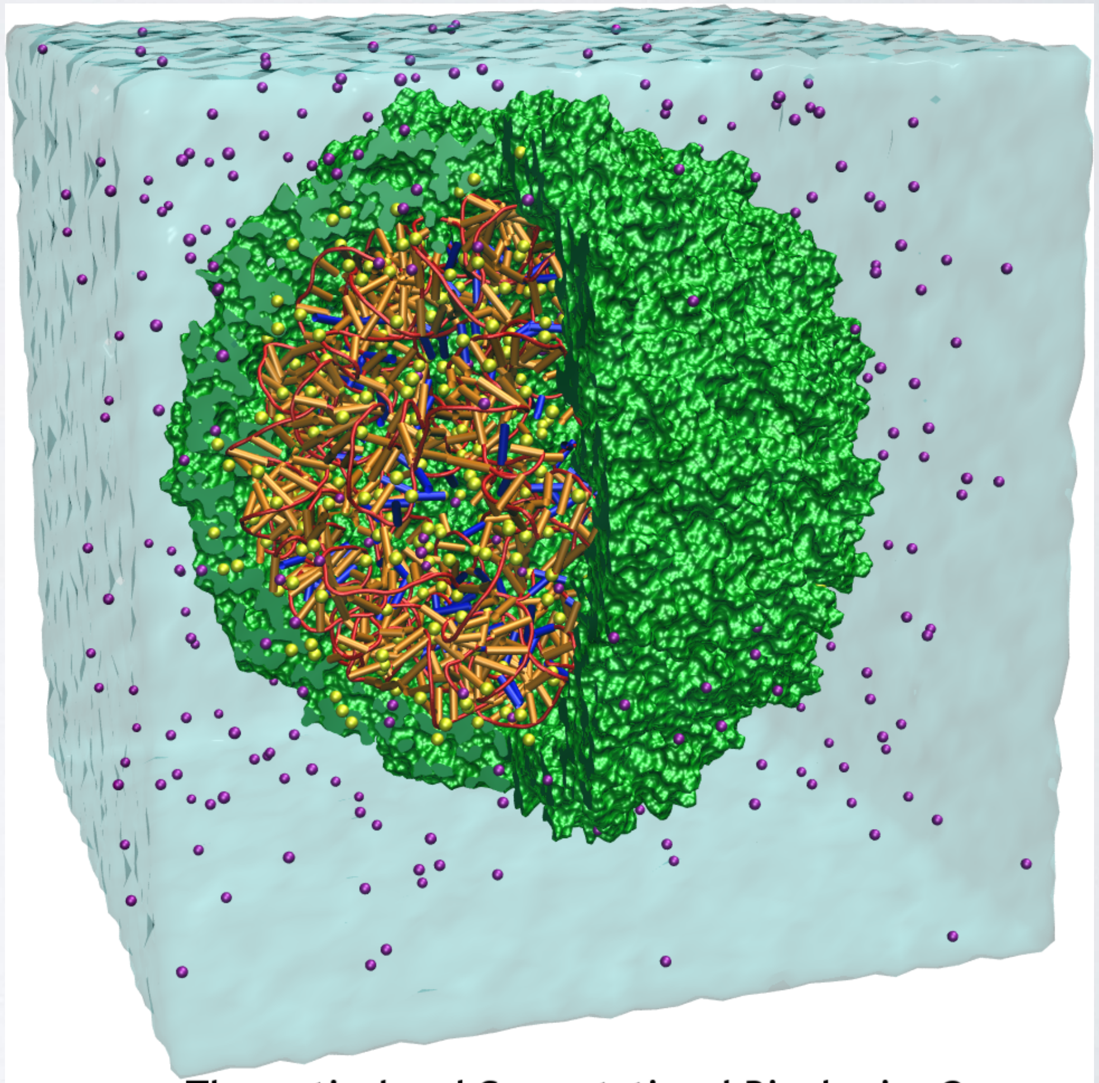
III. Visualizing trajectories

Visualization

- LAMMPS contains no built-in visualization program

- Trajectory visualization is an extremely important part of molecular simulation:

  - "sanity check"
  - error diagnosis
  - global overview of structure and dynamics
  - guide and inform where to study / analyze
  - required for presentations of MD research!

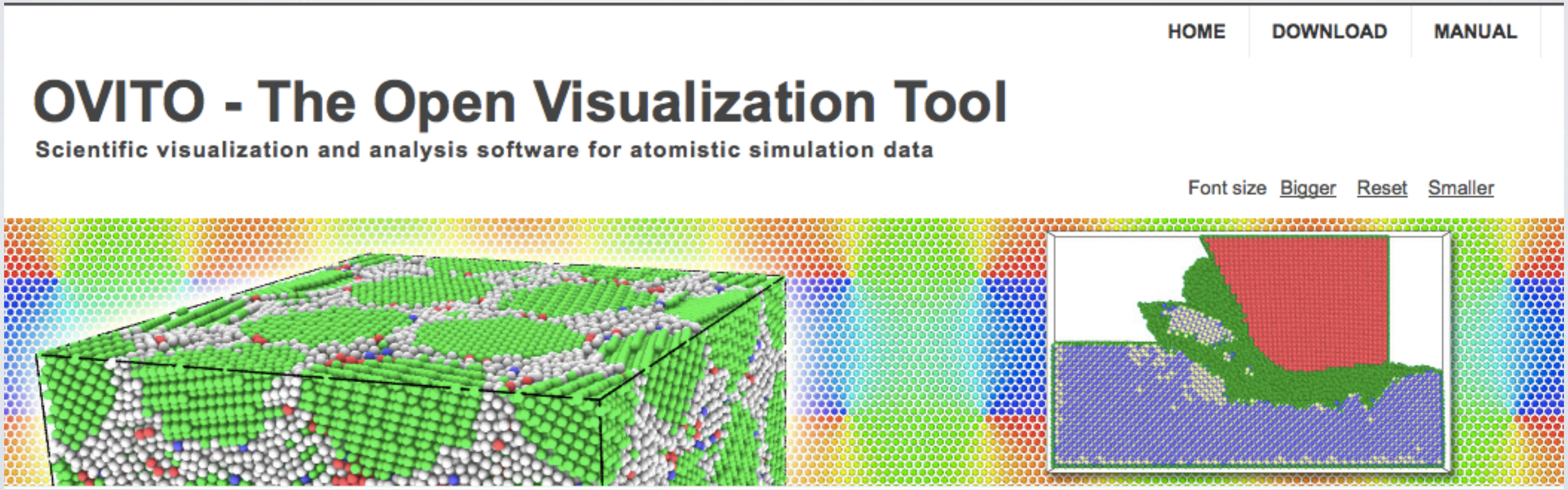


Theoretical and Computational Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

OVITO

- Many visualization programs exist (e.g., VMD, Chime, Rasmol, Chimera, PyMol)

- **OVITO** is a free, user-friendly and powerful visualization engine available for Linux, Mac and Windows that plays well with LAMMPS

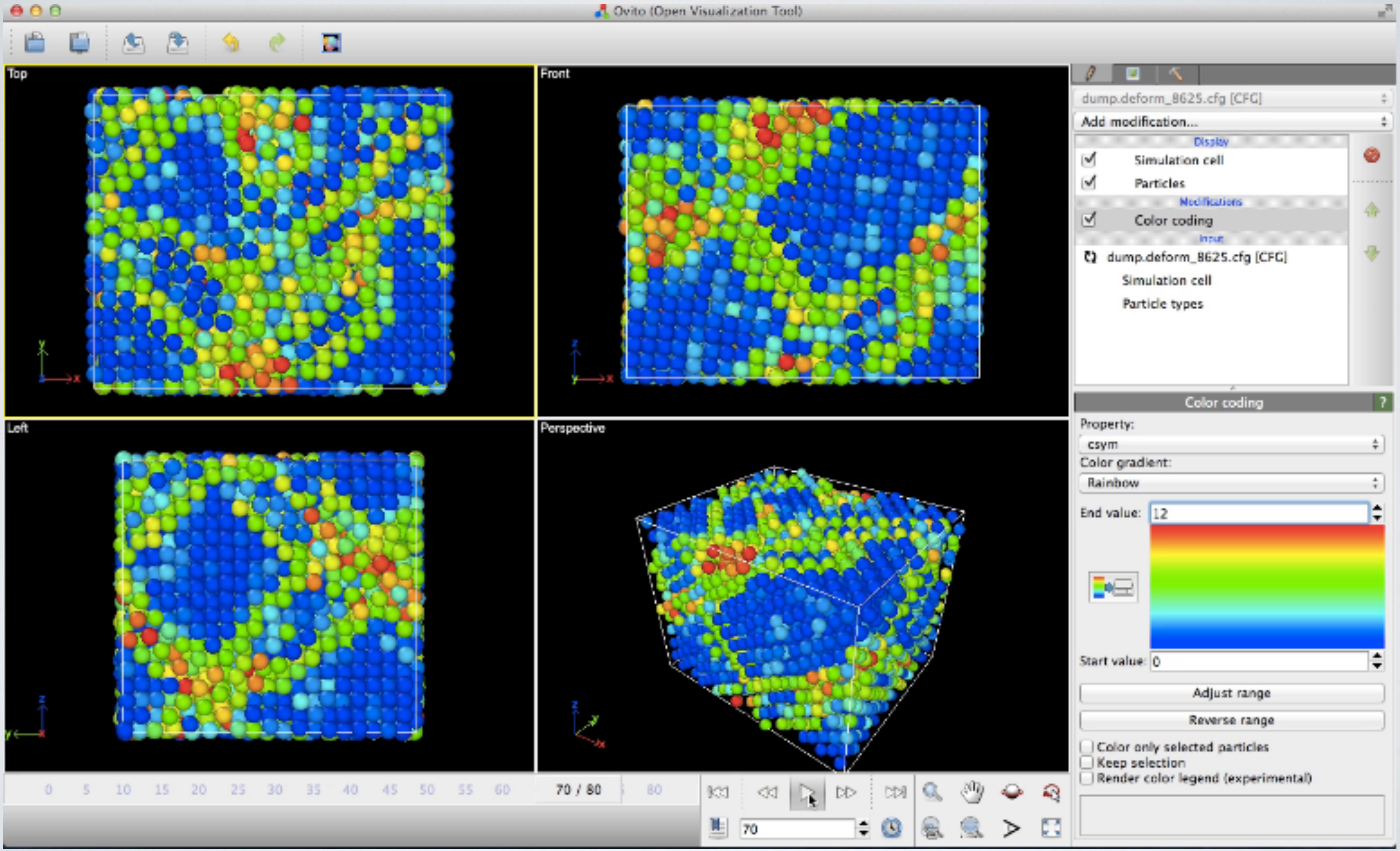


OVITO - The Open Visualization Tool

Scientific visualization and analysis software for atomistic simulation data

http://www.ovito.org

OVITO



dump.deform_8625.cfg [CFG]

Add modification...

Display
- [✓] Simulation cell
- [✓] Particles

Modifications
- [✓] Color coding

Input
- dump.deform_8625.cfg [CFG]
  - Simulation cell
  - Particle types

Color coding

Property: csym

Color gradient: Rainbow

End value: 12

Start value: 0

- [ ] Color only selected particles
- [ ] Keep selection
- [ ] Render color legend (experimental)

70 / 80

70