Thermodynamics and Materials Modeling

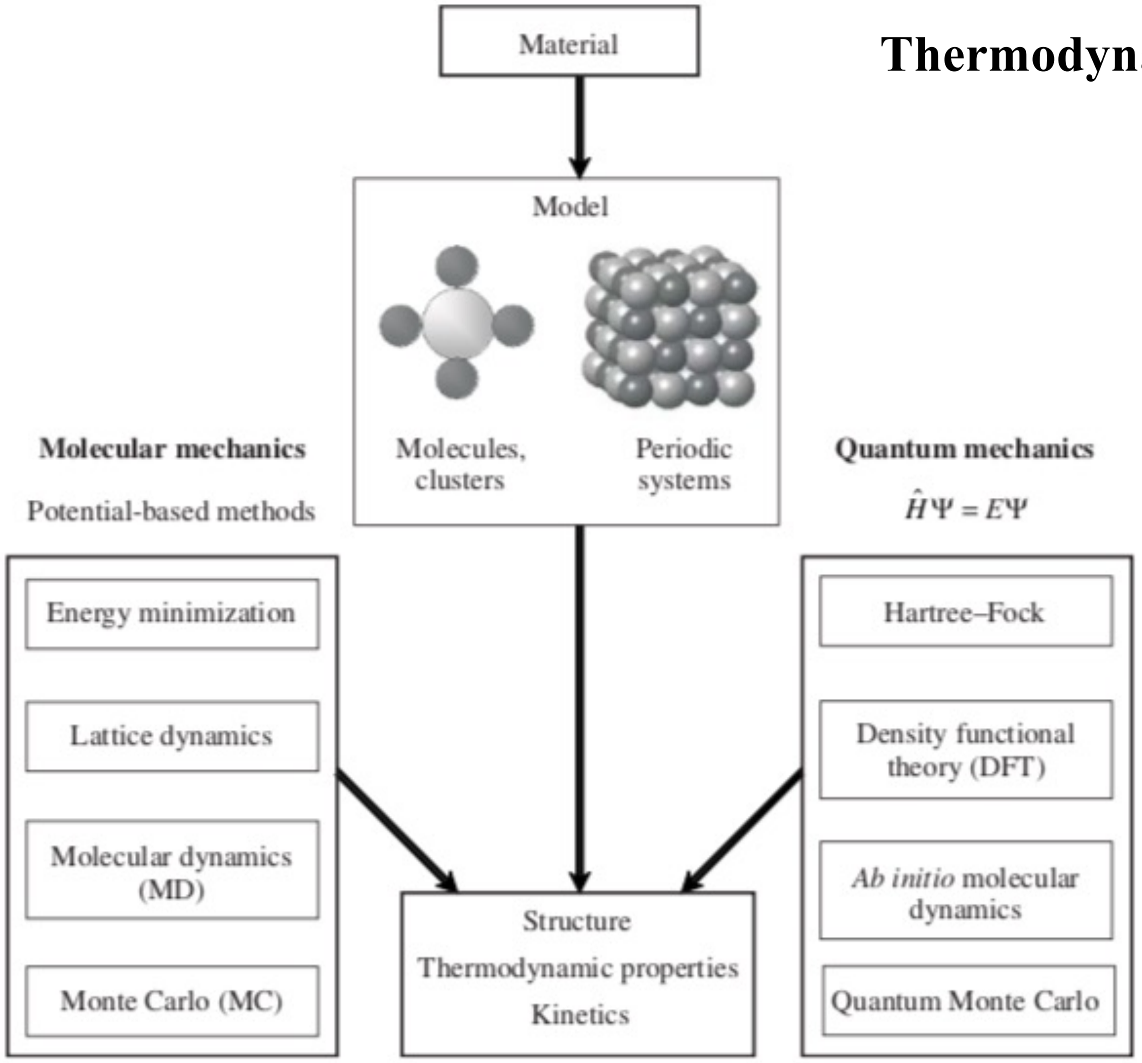


Figure 11.1

Coarse Grain Simulations

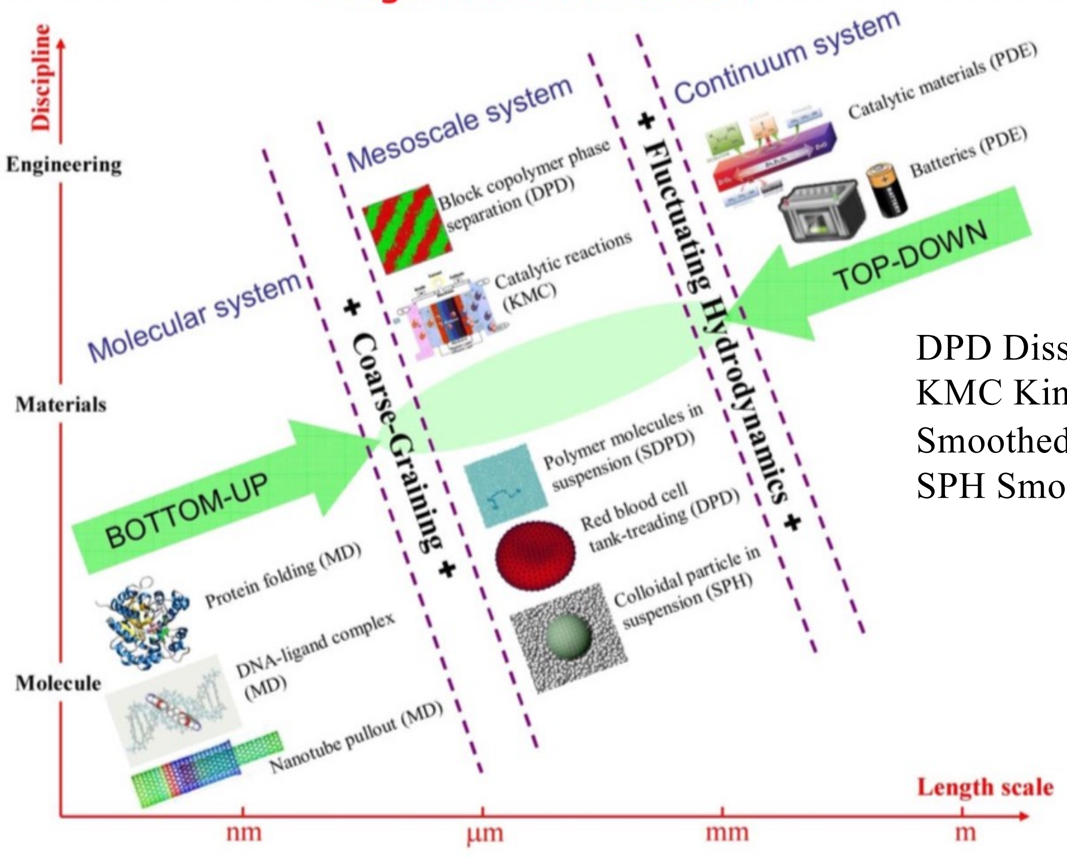
DPD Simulations
(Dissipative particle dynamics)

Machine Learning

Data Mining Techniques

Mesoscale Phenomena and Models

Due to wide range of characteristic lengths - times, several simulation methods that describe length and time scales have been developed:

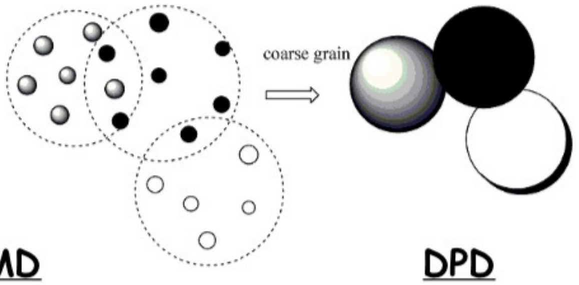


Discipline
Engineering
Materials
Molecule
Molecular system
Mesoscale system
Continuum system
+ Coarse-Graining +
+ Fluctuating Hydrodynamics +
BOTTOM-UP
TOP-DOWN
Protein folding (MD)
DNA-ligand complex (MD)
Nanotube pullout (MD)
Block copolymer phase separation (DPD)
Catalytic reactions (KMC)
Polymer molecules in suspension (SDPD)
Red blood cell tank-treading (DPD)
Colloidal particle in suspension (SPH)
Catalytic materials (PDE)
Batteries (PDE)
Length scale
nm
μm
mm
m

Partial differential equations

DPD Dissipative particle dynamics
KMC Kinetic Monte Carlo
Smoothed DPD for fluid surfaces
SPH Smoothed particle hydrodynamics

Dissipative Particle Dynamics (DPD)



MD

- MICROscopic level approach
- atomistic approach is often problematic because larger time/length scales are involved

DPD

- set of point particles that move off-lattice through prescribed forces
- each particle is a collection of molecules
- MESOscopic scales
- momentum-conserving Brownian dynamics

Navier-Stokes

- continuum fluid mechanics
- MACROscopic modeling

Ref on Theory: Lei, Caswell & Karniadakis, Phys. Rev. E, 2010

CRUNCH GROUP

Intermolecular/inter-particulate potentials

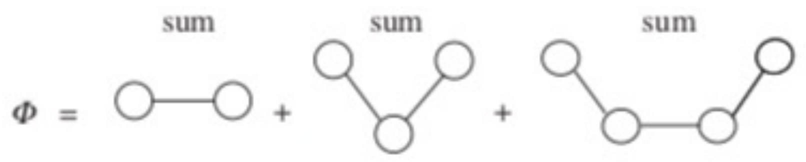


Figure 11.2

$$\Phi = \underset{\substack{\text{over} \\ \text{all pairs of} \\ \text{atoms}}}{\sum_{i<j} V_{ij}} + \underset{\substack{\text{over all} \\ \text{3 atom} \\ \text{combinations}}}{\sum_{i<j<k} V_{ijk}} + \underset{\substack{\text{over all} \\ \text{4 atom} \\ \text{combinations}}}{\sum_{i<j<k<l} V_{ijkl}} + \dots$$

2-body potential — 3-body potential — 4-body potential

$$\Phi = \underbrace{V_{12} + V_{13} + V_{14} + V_{23} + V_{24} + V_{34}}_{\text{2-body}} + \underbrace{V_{123} + V_{124} + V_{134} + V_{234}}_{\text{3-body}} + \underbrace{V_{1234}}_{\text{4-body}}$$

Molecular Solid

Harmonic Potential 2-body

Harmonic Potential 3-body

$$\Phi = \sum_{\substack{\text{bond lengths}\\\text{in molecules}}} \frac{k_r}{2}(r-r_0)^2 + \sum_{\substack{\text{bond lengths}\\\text{in molecules}}} \frac{k_\theta}{2}(\theta-\theta_0)^2$$

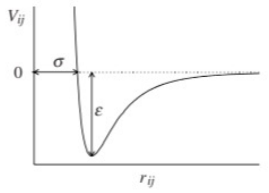
$$+ \sum_{\substack{\text{torsion angles}\\\text{in molecules}}} \frac{V_n}{2}(1+\cos(n\omega-\xi)) + \sum_{\text{atoms } i \text{ and } j} \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}$$

4-body torsion

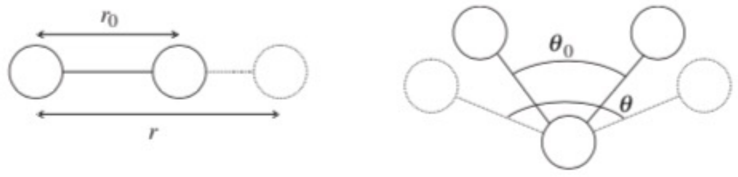
Coulomb Long-range

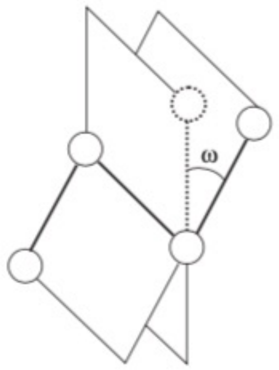
$$+ \sum_{\substack{\text{non-bonded}\\\text{atoms } i \text{ and } j}} 4\varepsilon_{ij}\left\{\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right\}$$

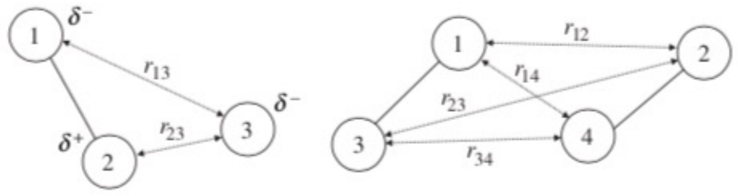
Van der Waals Repulsion and Pauli dispersion



Intramolecular interactions







Intermolecular interactions

Ionic Solids

$$V_{ij} = \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} + V_{\text{vdw}}(r_{ij})$$

Buckingham Potential (Rather than 6-12 potential)

$$V_{\text{vdw}}(r_{ij}) = A\exp(-r_{ij}/\rho) - \frac{C}{r_{ij}^6}$$

Energy Minimization/Lattice Statics

$\partial U / \partial Z_i = \partial \Phi / \partial Z_i = 0$

Static limit at 0K no vibrations
Yields the low temperature 0K phase

Z_i are length of bonds, separation distances, angles, positions

Elevated Temperatures and Lattice Dynamics

-SUV
H A
-pGT

Monte Carlo
Molecular Dynamics
Lattice Dynamics

$G = U - TS + pV$ At p = 0 $A = U - TS$

Quasi-harmonic approximation: at T we can write:

Good up to $2T_m/3$ where anharmonic effects become important

$$A = \Phi_{\text{stat}} + A_{\text{vib}}$$

$$A_{\text{vib}} = \sum_{q,j} \{ \tfrac{1}{2} h\nu_j(\mathbf{q}) + k_B T \ln[1 - \exp(-h\nu_j(\mathbf{q}) / k_B T)] \}$$

Simple Harmonic Oscillator

Zero point energy

Phonon frequencies at wave vector q

$$\omega(q) = \sqrt{\frac{4K}{m}} \left| \sin\left(\frac{qa}{2}\right) \right| \qquad -\frac{\partial U}{\partial u_n} = F = -K(2u_n - u_{n+1} - u_{n-1})$$

From Chapter 8.2

Frequency can be obtained from the potential function as a function of atomic positions

Lattice dynamics example

$$\alpha = \frac{1}{V}\left(\frac{\partial V}{\partial T}\right)_p$$

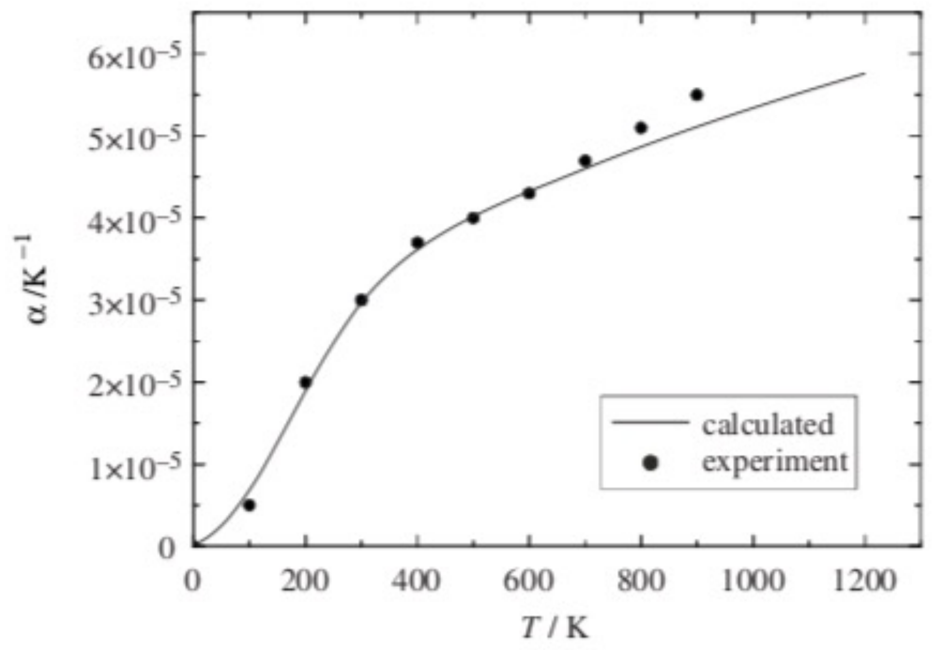


Figure 11.8 Calculated and experimental thermal expansion of MgF_2 [3].

T_m = 1536 K
2/3 T_m = 1024 K

N. L. Allan, G. D. Barrera, J. A. Purton, C. E. Sims and M. B. Taylor, *Phys. Chem., Chem. Phys.* 2000, **2**, 1099

$$A = \Phi_{\text{stat}} + A_{\text{vib}}$$

-SUV
H A
-pGT

$$A_{\text{vib}} = \sum_{\boldsymbol{q},j} \{\tfrac{1}{2} h\nu_j(\boldsymbol{q}) + k_B T \ln[1 - \exp(-h\nu_j(\boldsymbol{q}) / k_B T)]\}$$

$$S = -(\partial A/\partial T)_V \text{, and } C_V = (\partial U/\partial T)_V$$

$$S = \sum_{\boldsymbol{q},j} \frac{(h\nu_j(\boldsymbol{q}) / T)}{\exp(h\nu_j(\boldsymbol{q}) / k_B T) - 1} - k_B \ln[1 - \exp(-h\nu_j(\boldsymbol{q}) / k_B T)]$$

$$C_V = \sum_{\boldsymbol{q},j} k_B \left(\frac{h\nu_j}{k_B T} \right)^2 \frac{1}{[\exp(h\nu_j / k_B T) - 1][1 - \exp(-h\nu_j / k_B T)]}$$

Allows for experimental verification

Solid solutions and non-stoichiometric oxides

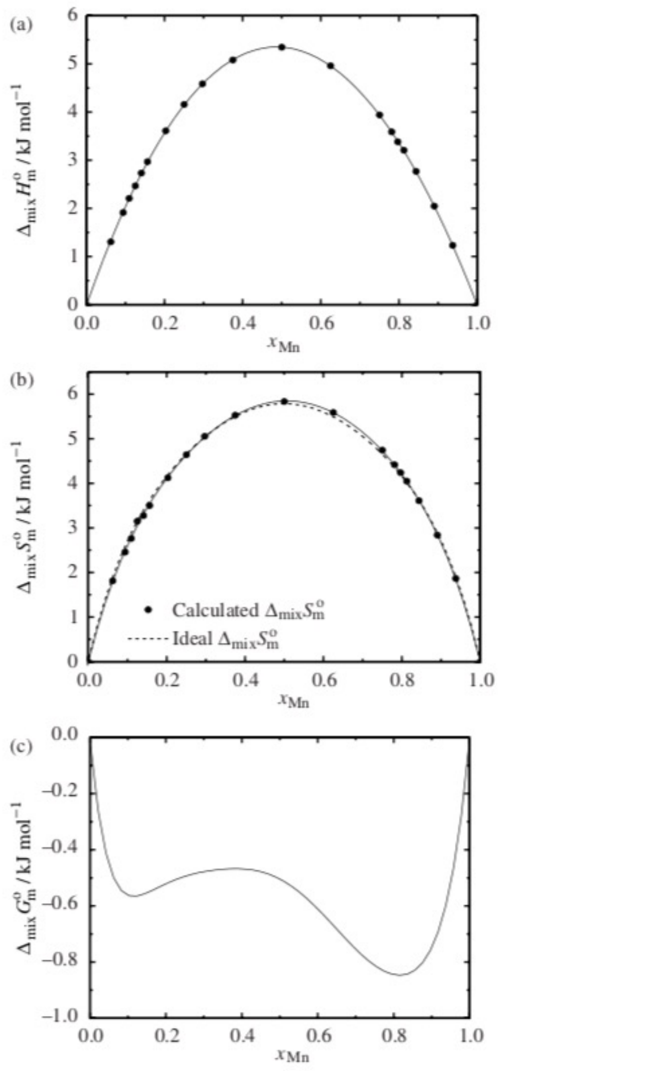


Figure 11.13 (a) Enthalpy, (b) entropy and (c) Gibbs energy of mixing of MnO–MgO at 1000 K, all calculated using the configurational averaging technique.

1) Fix unit cell size (super cell, 4 atoms, 8 atoms, 64 atom etc.) for a given composition find all random arrangements "k"
2) With lattice vectors and positions as variable at T minimize G_k
3) For all "K" arrangements find the average properties

$$H = \frac{\sum_{k=1}^{K} H_k \exp(-G_k/k_B T)}{\sum_{k=1}^{K} \exp(-G_k/k_B T)} \qquad G = -k_B T \ln \sum_{k=1}^{K} \exp(-G_k/k_B T)$$

4) Vary cell size and find convergence with larger cell sizes

N. L. Allan, G. D. Barrera, J. A. Purton, C. E. Sims and M. B. Taylor, *Phys. Chem., Chem. Phys.* 2000, **2**, 1099

E. Bakken, N. L. Allan, T. H. K. Barron, C. E. Mohn, I. T. Todorov and S. Stølen, *Phys. Chem., Chem. Phys.* 2003, **5**, 2237.

Monte Carlo Method

Periodic Boundary Conditions

Fix T, V, N

$$\langle Q \rangle = \int Q(Z)P(Z)\,\mathrm{d}Z \qquad P(Z) = \frac{\exp(-U(Z)/k_BT)}{\int \exp(-U(Z)/k_BT)\,\mathrm{d}Z}$$

"Z" is a state of the system

1) Move atoms at random
2) Calculate "Q"
3) Take the average

This doesn't work because P(Z) depends on U(Z) and T
Low energy states have more weight

Metropolis Algorithm
Bias the probability with $\exp(-\Phi(Z)/k_BT)$

Monte Carlo Method

Periodic Boundary Conditions

Fix T, V, N

$$\langle Q \rangle = \int Q(Z)P(Z)\,\mathrm{d}Z \qquad P(Z) = \frac{\exp(-U(Z)/k_BT)}{\int \exp(-U(Z)/k_BT)\,\mathrm{d}Z}$$

"Z" is a state of the system

1) Calculate $\phi(Z)$ by molecular mechanics with potentials
2) Accept a configuration "Z" if it has a low energy relative to kT with some randomness
3) Calculate the average

$$\langle Q \rangle = \frac{1}{M}\sum_{i=1}^{M} Q(Z)$$

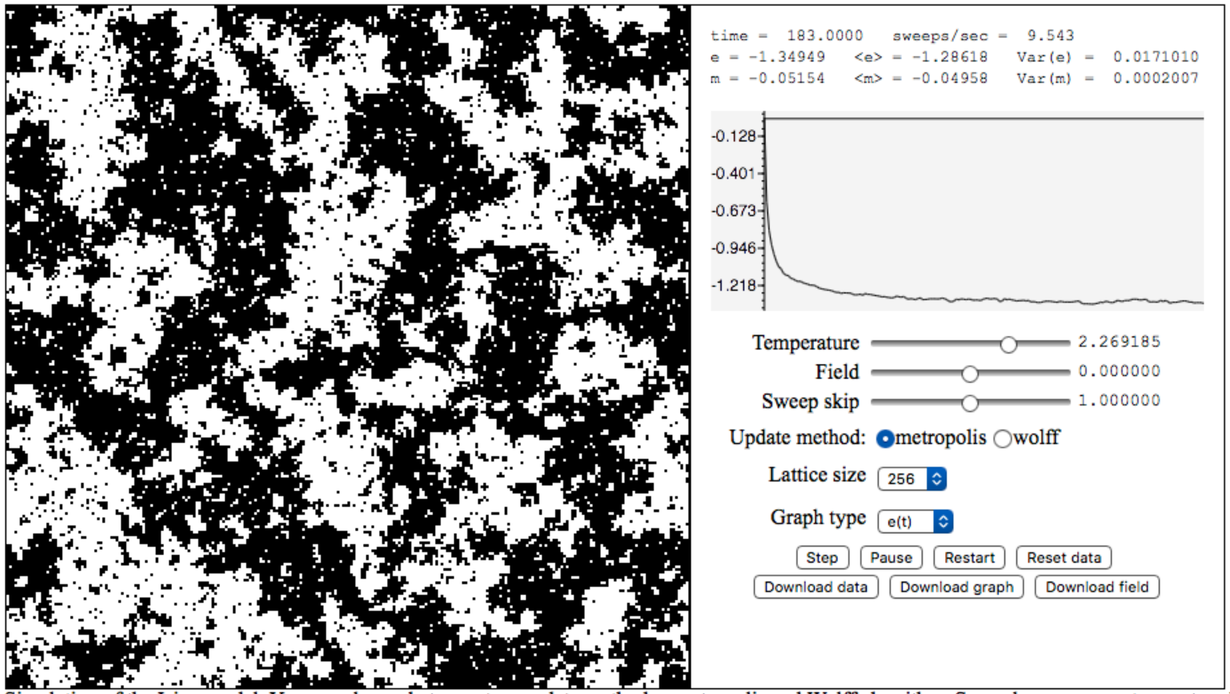
1) Start with a random configuration calculate $\phi(Z)$
2) Move one atom or molecule or group of molecules
3) Calculate $\phi(Z')$ if lower than $\phi(Z)$ accept
4) If higher than $\phi(Z)$ calculate $\exp(-\Delta\phi/kT)$ and a random number from 0 to 1
5) If lower than random number accept
6) Repeat

Monte Carlo Method

Ising Model Simulation

1) Start with a random configuration calculate $\phi(Z)$
2) Move one atom or molecule or group of molecules
3) Calculate $\phi(Z')$ if lower than $\phi(Z)$ accept
4) If higher than $\phi(Z)$ calculate $\exp(-\Delta\phi/kT)$ and a random number from 0 to 1
5) If lower than random number accept
6) Repeat

Ising simulation



```
time =  183.0000   sweeps/sec =  9.543
e = -1.34949   <e> = -1.28618   Var(e) =  0.0171010
m = -0.05154   <m> = -0.04958   Var(m) =  0.0002007
```

Temperature 2.269185
Field 0.000000
Sweep skip 1.000000
Update method: [✓ metropolis] wolff
Lattice size 256
Graph type e(t)

Simulation of the Ising model. You can choose between two update methods - metropolis and Wolff algorithm. Several measurements are stored while running including the current energy and magnetization, their averages, and their variances. The current value of each can be found at the top of the right panel. A graph of one value versus time can be directly below that. The particular variable can be changed via the 'Graph type' dropdown box. These measurements can be reset at any time by pressing 'Reset data'. The 'Reset' button restarts the simulation at infinite temperature (implies 'Reset data').

Molecular Dynamics

1) Generate initial condition with particles identified by position and velocity
2) Calculate the force on each particle using potentials
3) Forces (accelerations) remain constant for a time step, position and velocity change

$$\boldsymbol{v}_i(t+\Delta t/2) = \boldsymbol{v}_i(t-\Delta t/2) + \frac{\boldsymbol{f}_i}{m_i}\Delta t$$

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

4) Repeat 3) until temperature is constant

$$\frac{3}{2}Nk_B T = \frac{1}{2}\sum_i m_i v_i^2$$

5) After steady state record velocities and positions so that $<r^2> = 6Dt$ is found

Time calculation is on the order of nanoseconds.

Neither Monte Carlo nor Molecular Dynamics can calculate the free energy since they ignore large energy regions of phase space
They can calculate differences in free energy for phase diagram construction

Thermodynamic Perturbation Method

Which is more stable, B) MgO or A) a mixture of MnO and MgO?

$$A_A - A_B = -k_B T \ln Z_A + k_B T \ln Z_B = -k_B T \ln \frac{Z_A}{Z_B}$$

$$A_A - A_B = -k_B T \ln\left(\sum \exp(-(U_A - U_B)/k_B T)\right)$$

$$A_A - A_B = -k_B T \ln\left\langle \exp(-(U_A - U_B)/k_B T)\right\rangle_B$$

1) Simulate B using Monte Carlo Metropolis Method, calculate A_B
2) Temporarily substitute the potential functions for A and calculate A_A
3) Find the difference, $A_A - A_B$ to determine stability.

This can work if $(A_A - A_B) < kT$
If not, then use a coupling parameter between 0 and 1

$$U(\lambda) = \lambda U_A + (1-\lambda) U_B$$

Thermodynamic Integration

$$U(\lambda) = \lambda U_A + (1-\lambda) U_B$$

$$A_A - A_B = \int_0^1 \left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle d\lambda$$

$$A_A - A_B \approx \sum_l \left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle \Delta\lambda$$

Recent Developments in ab initio Thermodynamics

D. ALFÈ,[1] G. A. DE WIJS,[2] G. KRESSE,[3] M. J. GILLAN[4,5]

International Journal of Quantum Chemistry, Vol. 77, 871–879 (2000)

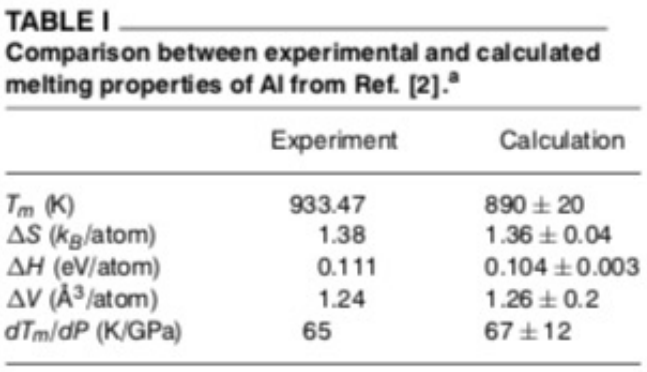


TABLE I
Comparison between experimental and calculated melting properties of Al from Ref. [2].[a]

	Experiment	Calculation
T_m (K)	933.47	890 ± 20
ΔS (k_B/atom)	1.38	1.36 ± 0.04
ΔH (eV/atom)	0.111	0.104 ± 0.003
ΔV (Å^3/atom)	1.24	1.26 ± 0.2
dT_m/dP (K/GPa)	65	67 ± 12

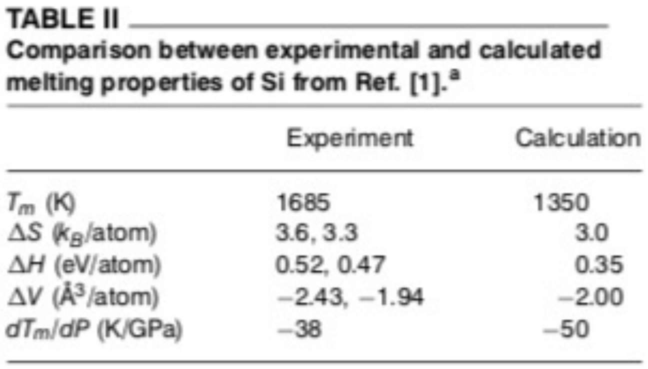


TABLE II
Comparison between experimental and calculated melting properties of Si from Ref. [1].[a]

	Experiment	Calculation
T_m (K)	1685	1350
ΔS (k_B/atom)	3.6, 3.3	3.0
ΔH (eV/atom)	0.52, 0.47	0.35
ΔV (Å^3/atom)	−2.43, −1.94	−2.00
dT_m/dP (K/GPa)	−38	−50

Quantum mechanical/ab initio methods

1) Electronic wavefunction is independent of the nuclei since electrons are much smaller and move much faster: Born-Oppenheimer Approximation
2) Solve the Schrodinger equation

$$\hat{H}\Psi = E\Psi$$

Hamiltonian in atomic units:

$$\hat{H} = -\frac{1}{2}\sum_i \nabla_i^2 - \sum_i \sum_\alpha \frac{Z_\alpha}{|\boldsymbol{r}_i - \boldsymbol{d}_\alpha|} + \sum_i \sum_{j>1} \frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} + \sum_\alpha \sum_{\beta>\alpha} \frac{Z_\alpha Z_\beta}{|\boldsymbol{d}_\beta - \boldsymbol{d}_\alpha|}$$

r_i electron positions; d_α nuclear positions, Z_α nuclear charge

Kinetic Energy – e^- nuc. attraction + e^- e^- repulsion + Nuc. Nuc. repulsion

3) Solve approximately since true wave function can't be found directly. Compare proposed function results with data. Variational Principle: lowest energy wins.

$$E = \frac{\int \Psi^* \hat{H}\Psi \, d\tau}{\int \Psi^* \Psi \, d\tau}$$

4) Obey Pauli exclusion principle.

Density functional theory

1) Ground state can be obtained through minimization of E(ρ) of ρ(r)
2) Parallel non-interacting system (NIS)

$$\rho(\boldsymbol{r}) = \sum_{i=1}^{N} |\psi_i(\boldsymbol{r})|^2$$

3) Write the energy functional as

$$\begin{aligned} E[\rho] &= T_S[\rho] + V_{\text{nuc}}[\rho] + J[\rho] + E_{\text{xc}}[\rho] \\ &= -\frac{1}{2}\sum_{i=1}^{N}\int \psi_i^*(\boldsymbol{r})\nabla^2\psi_i(\boldsymbol{r})\,\mathrm{d}\boldsymbol{r} - \sum_{\alpha}\int \rho(\boldsymbol{r})\frac{Z_\alpha}{|\boldsymbol{r}-\boldsymbol{d}_\alpha|}\mathrm{d}\boldsymbol{r} \\ &\quad + \frac{1}{2}\iint \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{r}' + E_{\text{xc}}[\rho] \end{aligned}$$

KE of NIS + e^- nuc. int. + Coulomb + exchange correlation energy

4) Minimize E[ρ] to obtain wave functions then iterate to obtain the ground state density and energy

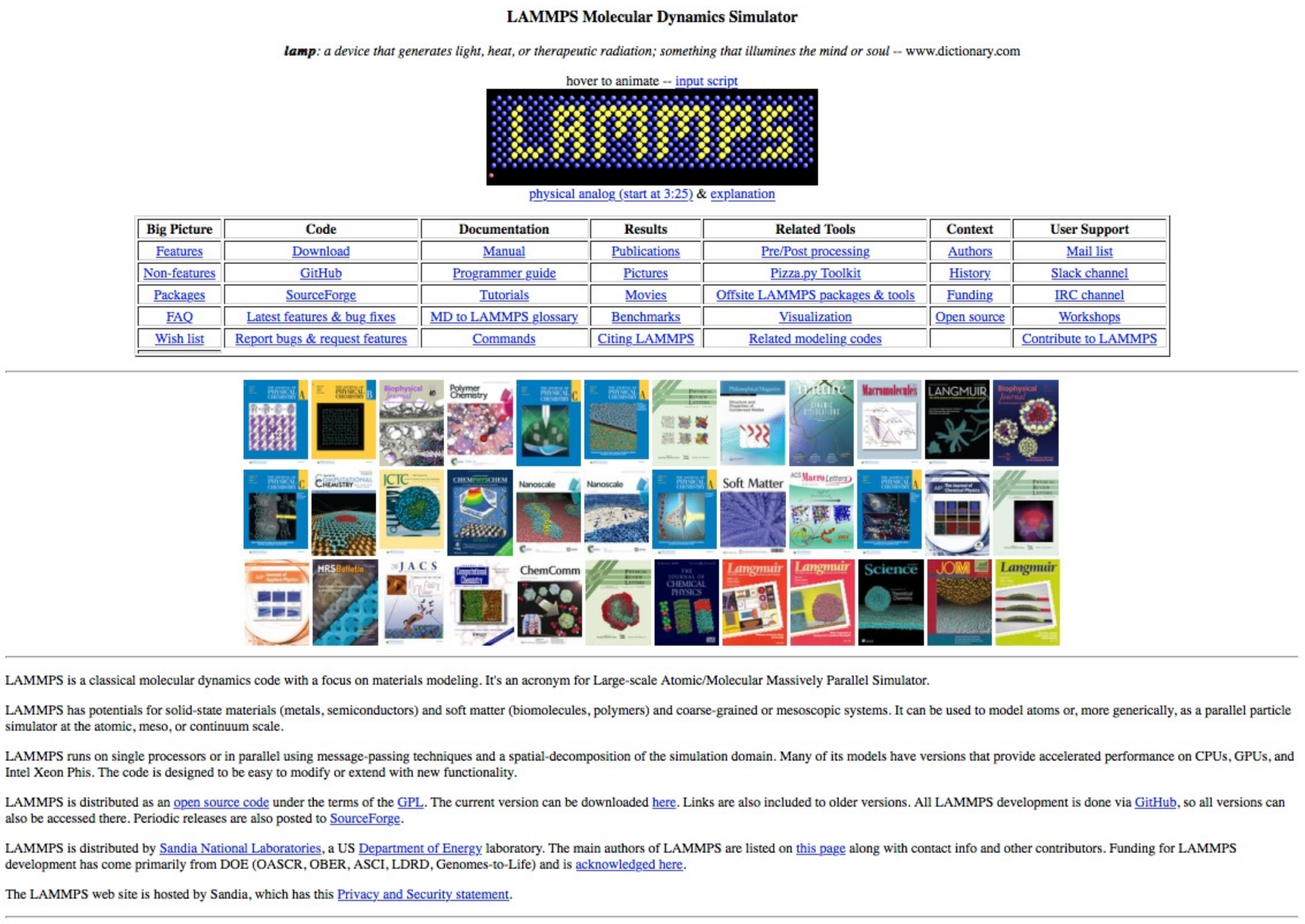
Exchange energy is energy of swapping identical electrons
Correlation energy is energy of interaction with all other electrons, how is the movement of one electron impacted by all other electrons
Neither of these are known so you use a Low-Density Approximation (LDA)

Molecular Dynamics

LAMMPS

In order to do molecular dynamics, you do not write a program.

Software is available to perform the calculations. LAMMPS



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	GitHub	Programmer guide	Pictures	Pizza.py Toolkit	History	Slack channel
Packages	SourceForge	Tutorials	Movies	Offsite LAMMPS packages & tools	Funding	IRC channel
FAQ	Latest features & bug fixes	MD to LAMMPS glossary	Benchmarks	Visualization	Open source	Workshops
Wish list	Report bugs & request features	Commands	Citing LAMMPS	Related modeling codes		Contribute to LAMMPS

LAMMPS is a classical molecular dynamics code with a focus on materials modeling. It's 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. Many of its models have versions that provide accelerated performance on CPUs, GPUs, and Intel Xeon Phis. The code is designed to be easy to modify or extend with new functionality.

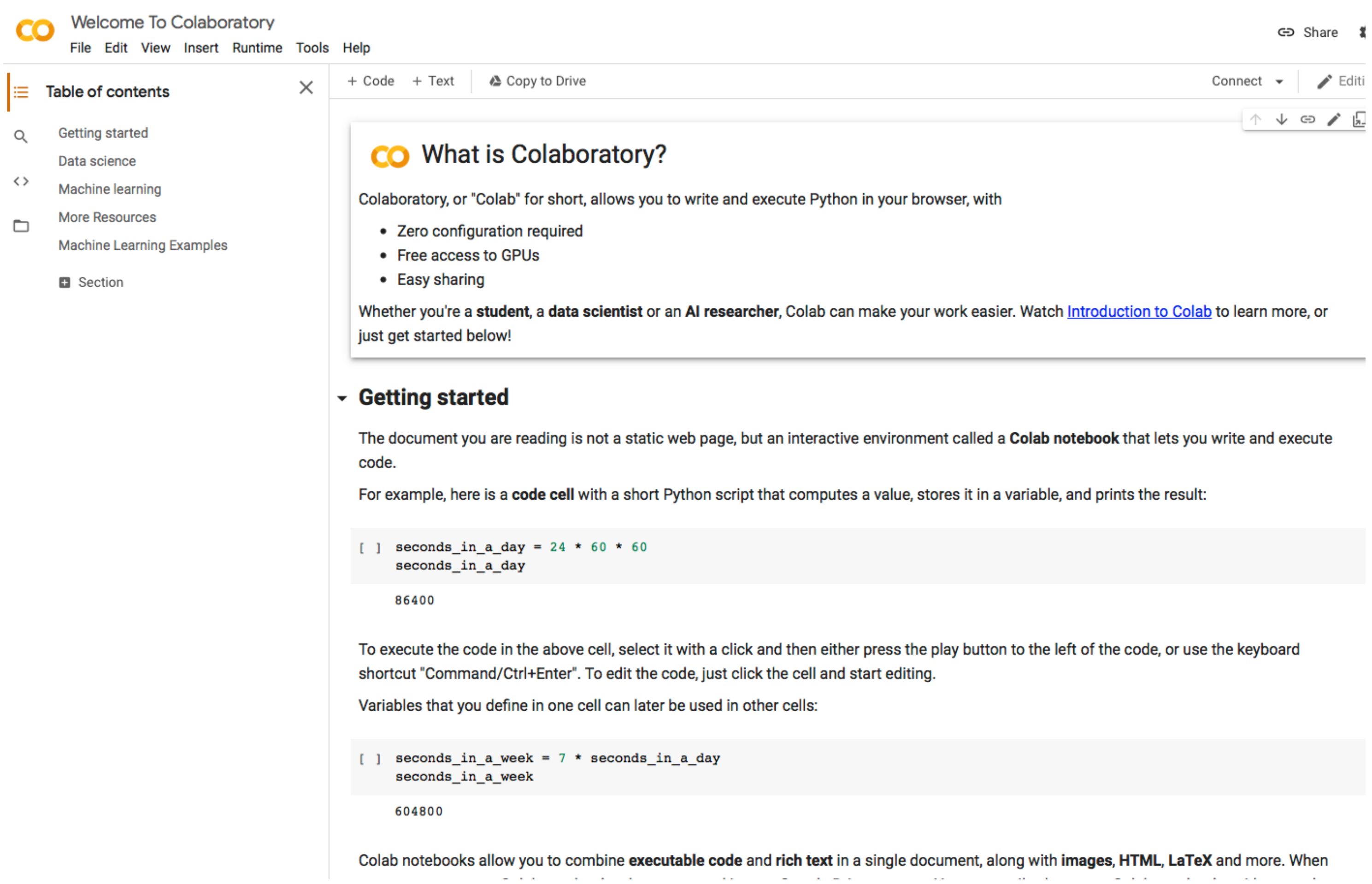
LAMMPS is distributed as an open source code under the terms of the GPL. The current version can be downloaded here. Links are also included to older versions. All LAMMPS development is done via GitHub, so all versions can also be accessed there. Periodic releases are also posted to SourceForge.

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 web site is hosted by Sandia, which has this Privacy and Security statement.

Free access to graphical processing units (GPU)

Colaboratory



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day

    86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

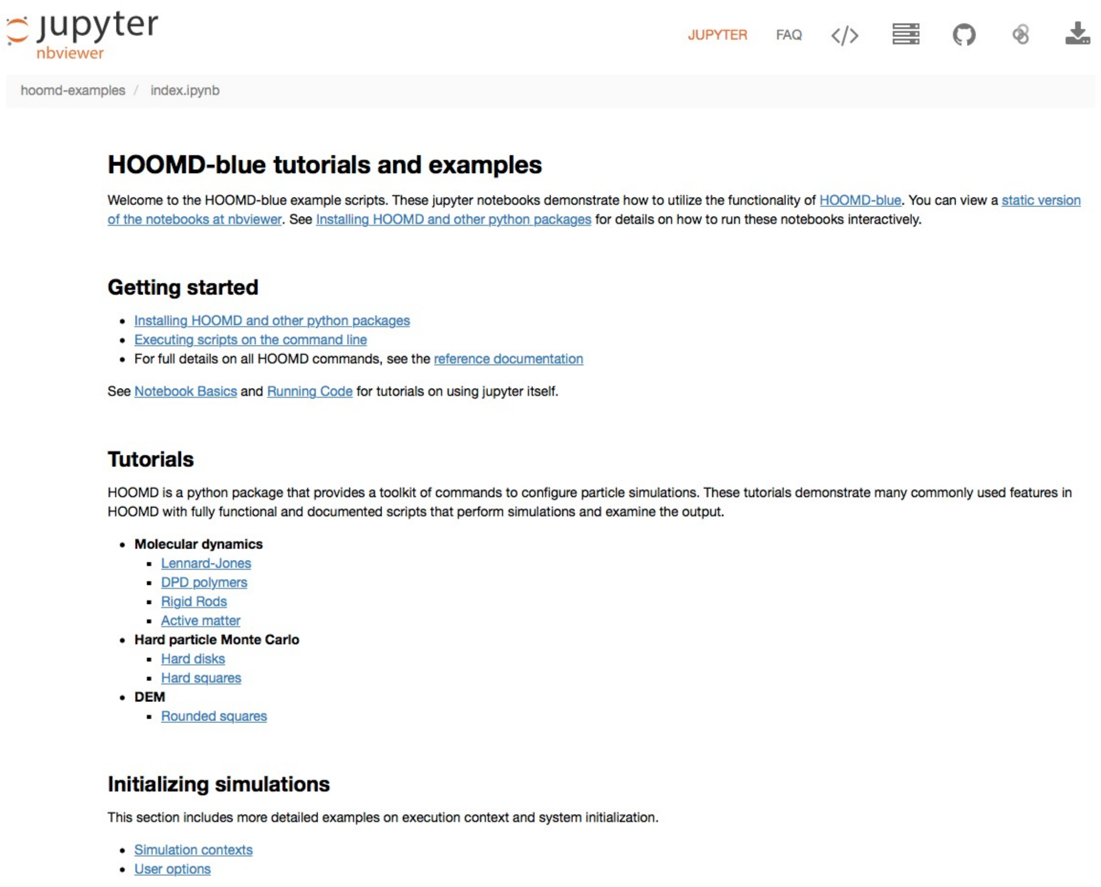
Variables that you define in one cell can later be used in other cells:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week

    604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When

HOOMD-blue MD simulations



HOOMD-blue tutorials and examples

Welcome to the HOOMD-blue example scripts. These jupyter notebooks demonstrate how to utilize the functionality of HOOMD-blue. You can view a static version of the notebooks at nbviewer. See Installing HOOMD and other python packages for details on how to run these notebooks interactively.

Getting started

- Installing HOOMD and other python packages
- Executing scripts on the command line
- For full details on all HOOMD commands, see the reference documentation

See Notebook Basics and Running Code for tutorials on using jupyter itself.

Tutorials

HOOMD is a python package that provides a toolkit of commands to configure particle simulations. These tutorials demonstrate many commonly used features in HOOMD with fully functional and documented scripts that perform simulations and examine the output.

- **Molecular dynamics**
  - Lennard-Jones
  - DPD polymers
  - Rigid Rods
  - Active matter
- **Hard particle Monte Carlo**
  - Hard disks
  - Hard squares
- **DEM**
  - Rounded squares

Initializing simulations

This section includes more detailed examples on execution context and system initialization.

- Simulation contexts
- User options

Dissipative particle dynamics

Off-lattice particles moving in space and time
A particle is a molecule or fluid region (not a single atom)
Particles experience dissipative and random forces (details are integrated)
Access to long times and distances
Up to 0.1 micron and 50 microseconds

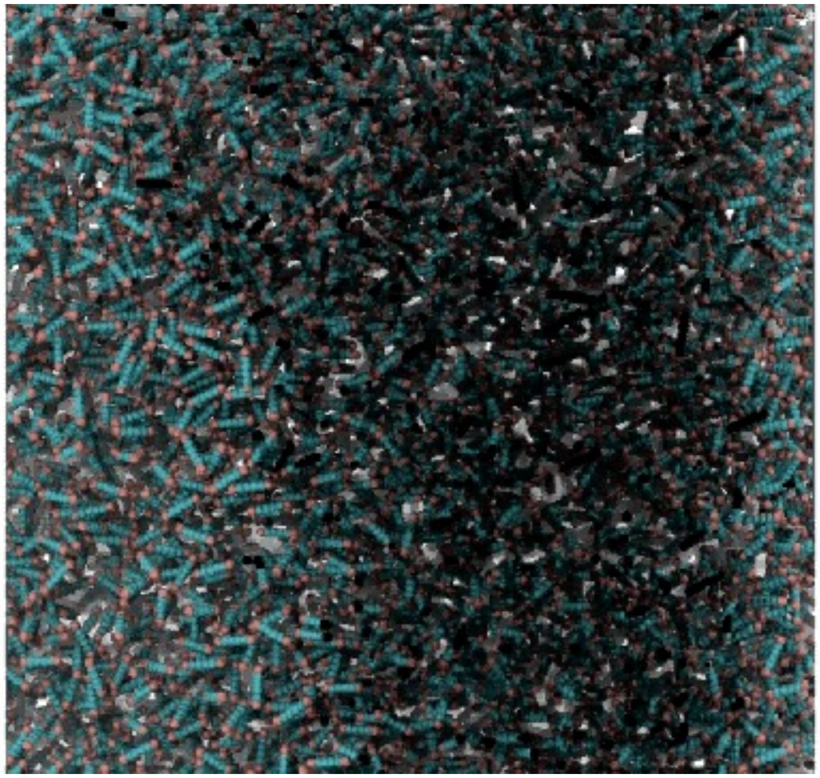
$$f_i = \sum_{j \neq i} (F_{ij}^C + F_{ij}^D + F_{ij}^R)$$

Non-bonded forces within a set cutoff distance are considered for particle “i” interaction with particles “j”
Conservative Force, Dissipative Force, Random Force
Conservative Force- gives particle an identity
Random and dissipative forces act as a thermostat
Momentum is conserved locally so you can work with small numbers of particles
Random force between two interaction particles must be of opposite sign
One random force calculation for each pair of interacting particles
(*This differs from Brownian motion where each particle experiences an independent random force*)
Particles connected by Hookean springs if desired
Usually N, V and T are kept constant

Parallelization in Dissipative particle dynamics

DPD interactions are short range so different processors can run in parallel for very large systems (micron and milliseconds (1000 microseconds))

LCD self-assembly
Smectic Crystal
Rod-Shaped particles
1 microsecond time



http://gpiutmd.iut.ac.ir/en/gallery/video-gallery

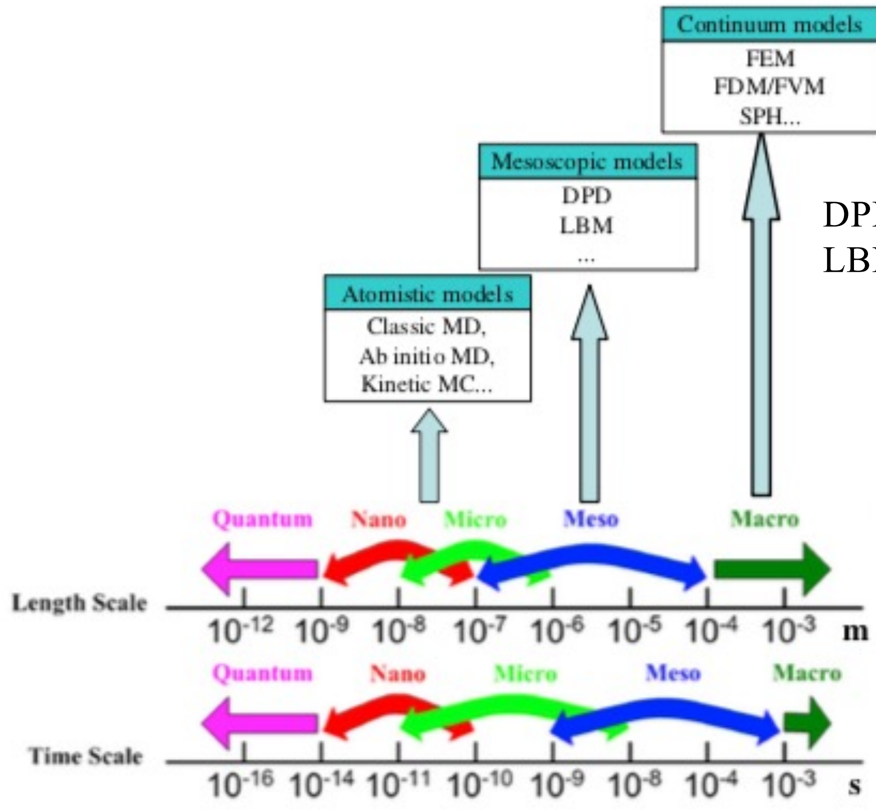


Fig. 1 Different length and time scales and corresponding computational methods

FEM Finite Element (Difference or Volume) Method
SPH Smoothed Particle Hydrodynamics

DPD Dissipative Particle Dynamics
LBM Lattice Boltzmann Method (fluids)

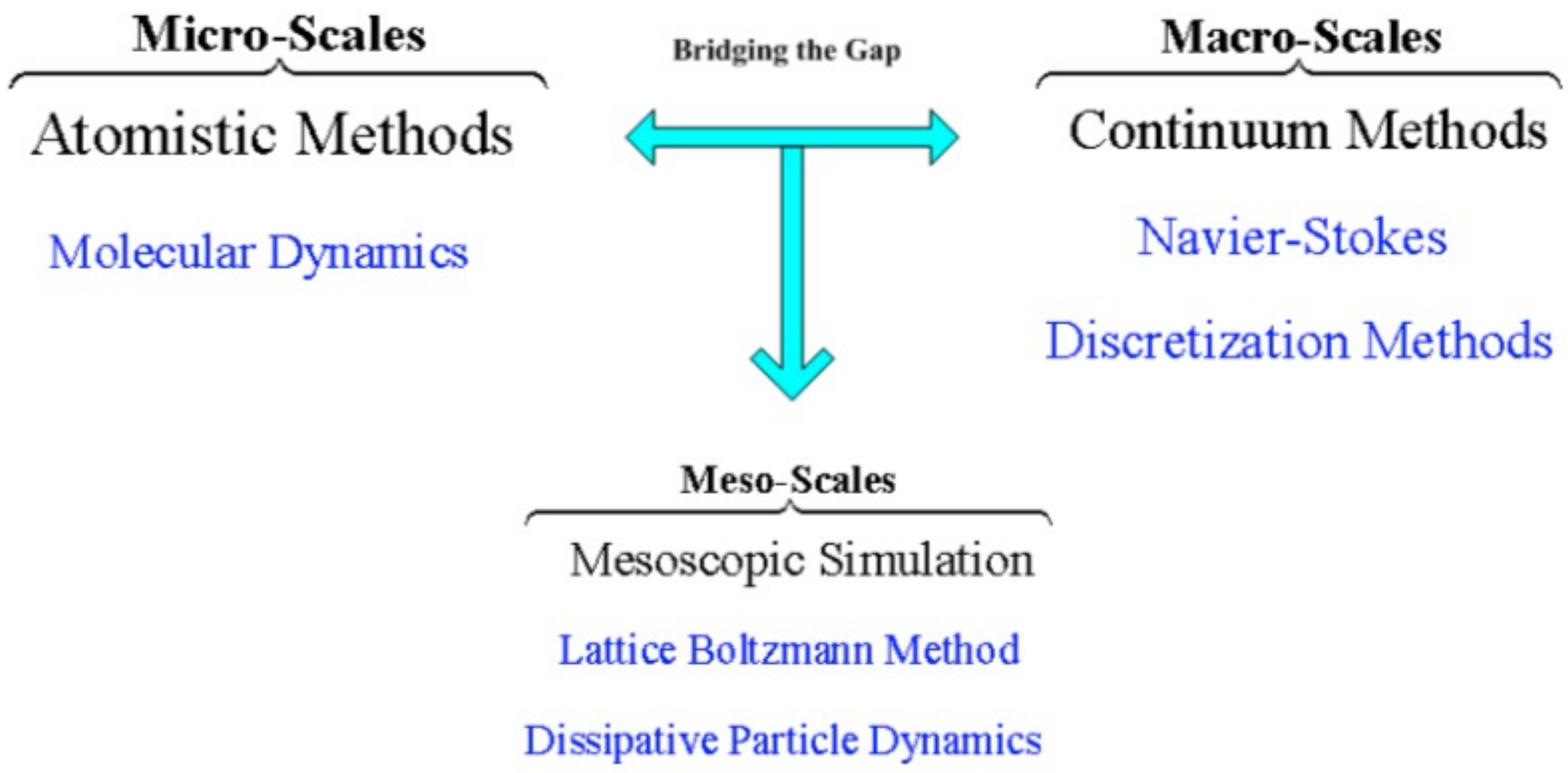


Fig. 1. Dissipative Particle Dynamics: a mesoscale technique for bridging the gap between the micro- and macro-scales.

Dissipative particle dynamics

Coarse Graining Parameter N_m = number of molecules per particle

Speedup time is 1000 $N_m^{8/3}$ for N_m = 3 you get 20,000X; for 7, 200,000X

Particles are "soft" not LJ particles

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \quad \frac{d\mathbf{p}_i}{dt} = \sum_{j \neq i} \mathbf{F}_{ij},$$

$$\mathbf{F}_{ij} = \mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R$$

$$\mathbf{F}_{ij}^C = w^C(r_{ij})\mathbf{e}_{ij},$$ Repulsive

$$\mathbf{F}_{ij}^D = -\gamma w^D(r_{ij})[\mathbf{v}_{ij} \cdot \mathbf{e}_{ij}]\mathbf{e}_{ij}$$ Drag

$$\mathbf{F}_{ij}^R = \sigma w^R(r_{ij})\theta_{ij}\mathbf{e}_{ij},$$ Random

$$\begin{aligned} \mathbf{e}_{ij} &= \mathbf{r}_{ij}/r_{ij} \\ \mathbf{r}_{ij} &= \mathbf{r}_i - \mathbf{r}_j \\ r_{ij} &= |\mathbf{r}_i - \mathbf{r}_j| \\ \mathbf{v}_{ij} &= (\mathbf{v}_i - \mathbf{v}_j) \end{aligned}$$

All forces act within a sphere of interaction, r_c

w are "weight functions"

γ and σ are coefficients

$\theta_{ij} = \theta_{ji}$ White noise function

Thermostat requirements:

$$w^D(r) = [w^R(r)]^2, \sigma^2 = 2\gamma k_B T/m$$

$$w^C(r_{ij}) = \begin{cases} a_{ij}\left(1 - \frac{r_{ij}}{r_c}\right) & r_{ij} \le r_c \\ 0 & r_{ij} > r_c \end{cases}$$

$$\langle \theta_{ij}(t) \rangle = 0, \langle \theta_{ij}(t)\theta_{kl}(t') \rangle = (\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\,\delta(t - t')$$

a_{ij} is the repulsion between particles "i" and "j"

Dissipative Particle Dynamics (DPD)

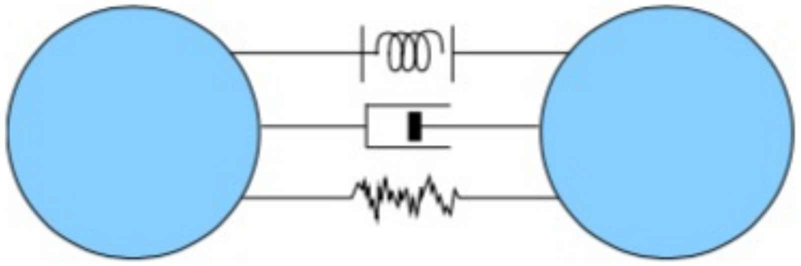


FIG. 1. Dissipative particles interact pair-wise with a conservative linear repulsive force, and a Brownian dashpot made of a friction force that reduces the relative velocity between the particles and a stochastic force that gives kicks of equal size and opposite directions to the particles. These forces vanish beyond a cutoff radius r_c.

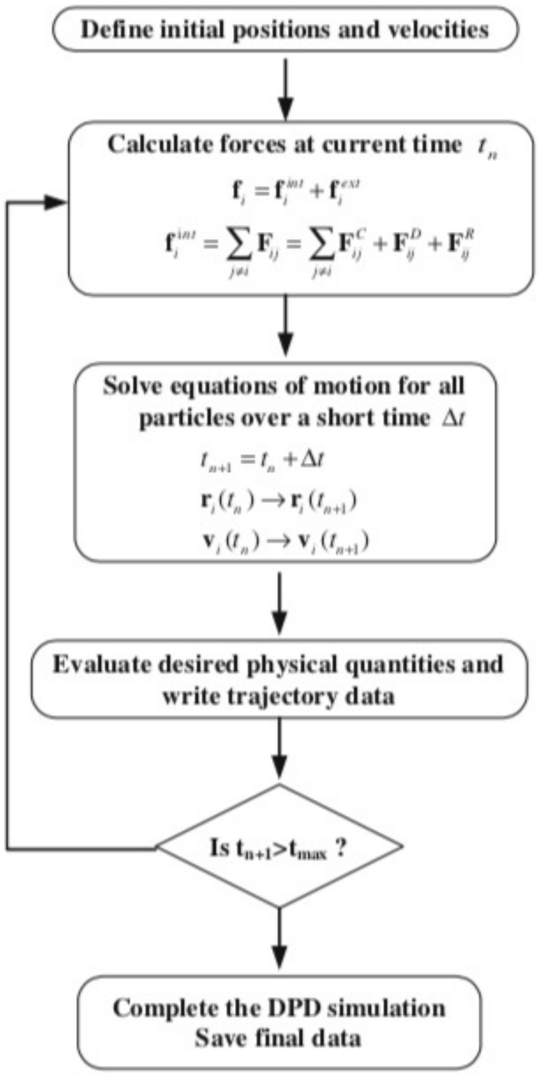

Define initial positions and velocities
Calculate forces at current time t_n
$\mathbf{f}_i = \mathbf{f}_i^{int} + \mathbf{f}_i^{ext}$
$\mathbf{f}_i^{int} = \sum_{j \neq i} \mathbf{F}_{ij} = \sum_{j \neq i} \mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R$
Solve equations of motion for all particles over a short time Δt
$t_{n+1} = t_n + \Delta t$
$\mathbf{r}_i(t_n) \rightarrow \mathbf{r}_i(t_{n+1})$
$\mathbf{v}_i(t_n) \rightarrow \mathbf{v}_i(t_{n+1})$
Evaluate desired physical quantities and write trajectory data
Is $t_{n+1} > t_{max}$?
Complete the DPD simulation Save final data

Fig. 2 Computational procedure of a DPD simulation

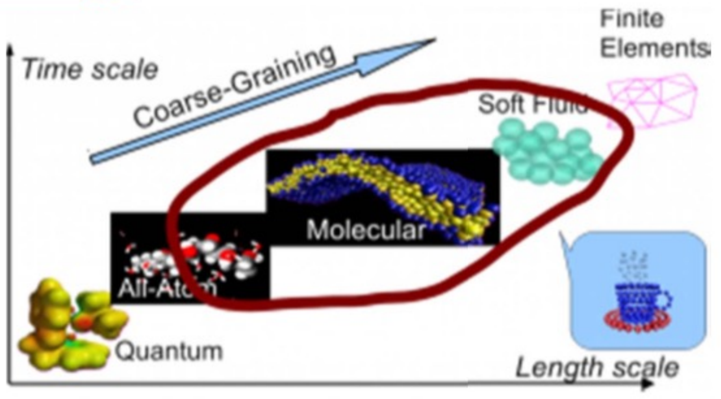
DPD Sofware/Package

LAMMPS: http://lammps.sandia.gov/

- Highly parallelized
- Highly portable C++
- Distributed-memory MPI
- Open-source distribution
- GPU and OpenMP support for many code features

ESPResSo: http://espressomd.org/

- Extensible
- Open-source
- Parallelized
- Portable



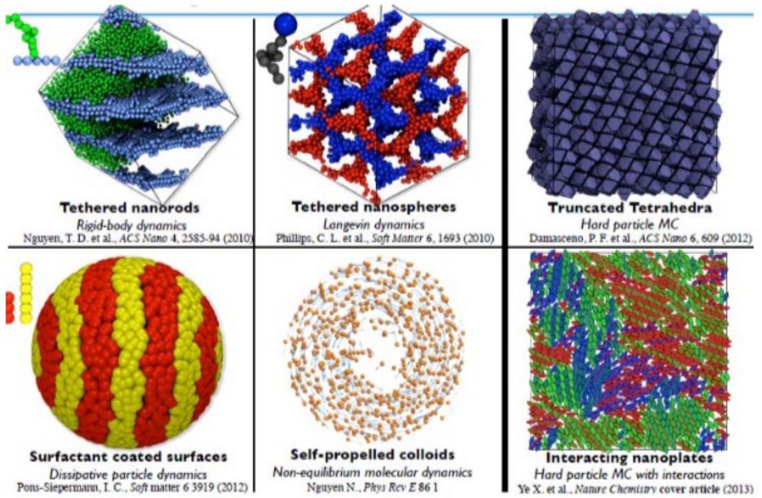
Length and time scales where ESPResSo works best

DPD Sofware/Package

HOOMD-blue: http://codeblue.umich.edu/hoomd-blue/

HOOMD-blue

- Fast GPU performance
- Scalable
- Flexible



Tethered nanorods
Rigid-body dynamics
Nguyen, T. D. et al., *ACS Nano* 4, 2585-94 (2010)

Tethered nanospheres
Langevin dynamics
Phillips, C. L. et al., *Soft Matter* 6, 1693 (2010)

Truncated Tetrahedra
Hard particle MC
Damasceno, P. F. et al., *ACS Nano* 6, 609 (2012)

Surfactant coated surfaces
Dissipative particle dynamics
Pons-Siepermann, I. C., *Soft* matter 6 3919 (2012)

Self-propelled colloids
Non-equilibrium molecular dynamics
Nguyen N., *Phys Rev E* 86 1

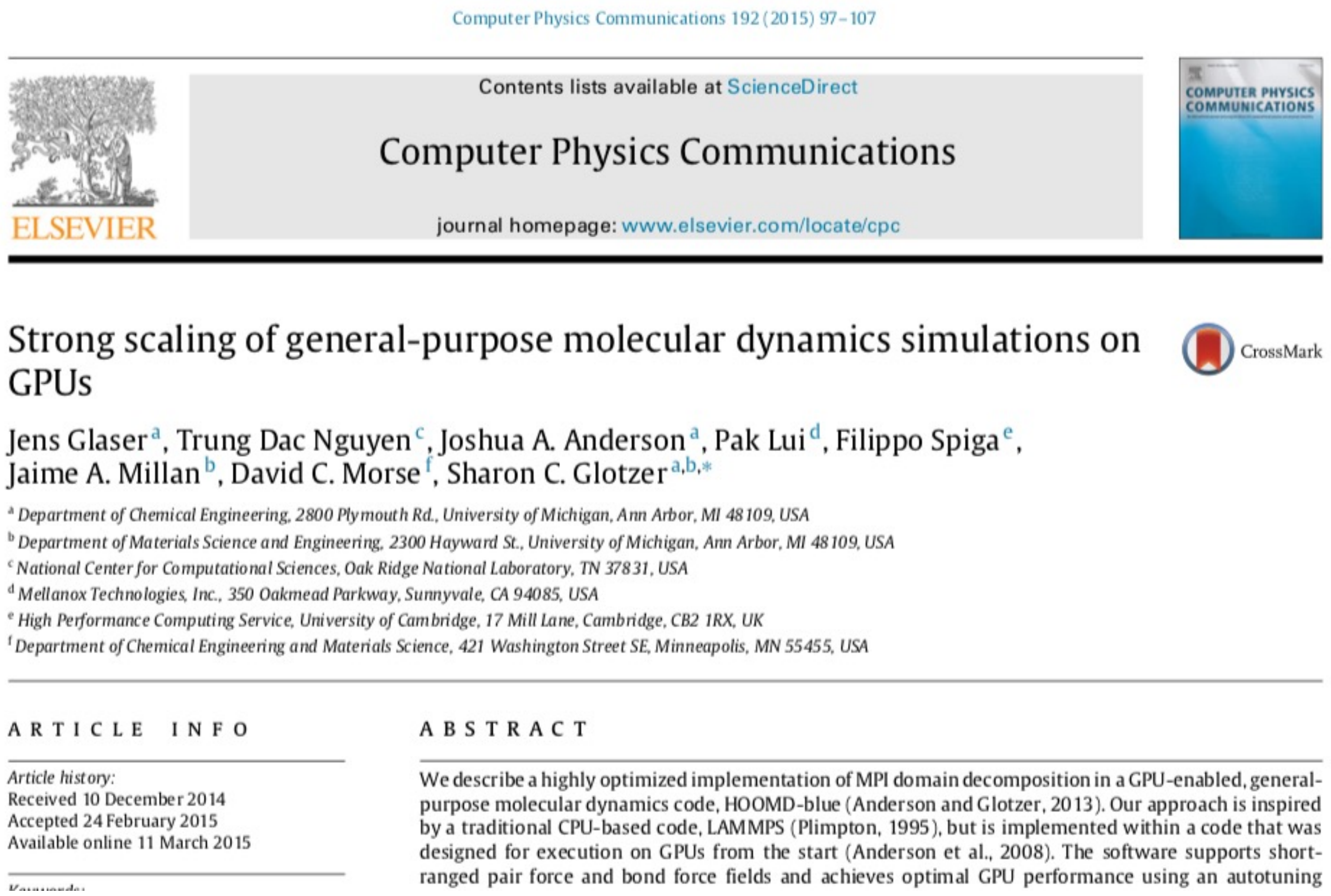
Interacting nanoplates
Hard particle MC with interactions
Ye X. et al., *Nature Chemistry* cover article (2013)

DPDmacs: http://www.apmaths.uwo.ca/~mkarttu/dpdmacs.shtml

- Compatible with Gromacs

MyDPD: http://multiscalelab.org/mydpd

- Simple, serial but functional



Computer Physics Communications 192 (2015) 97–107

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Strong scaling of general-purpose molecular dynamics simulations on GPUs

Jens Glaser[a], Trung Dac Nguyen[c], Joshua A. Anderson[a], Pak Lui[d], Filippo Spiga[e], Jaime A. Millan[b], David C. Morse[f], Sharon C. Glotzer[a,b,*]

[a] Department of Chemical Engineering, 2800 Plymouth Rd., University of Michigan, Ann Arbor, MI 48109, USA

[b] Department of Materials Science and Engineering, 2300 Hayward St., University of Michigan, Ann Arbor, MI 48109, USA

[c] National Center for Computational Sciences, Oak Ridge National Laboratory, TN 37831, USA

[d] Mellanox Technologies, Inc., 350 Oakmead Parkway, Sunnyvale, CA 94085, USA

[e] High Performance Computing Service, University of Cambridge, 17 Mill Lane, Cambridge, CB2 1RX, UK

[f] Department of Chemical Engineering and Materials Science, 421 Washington Street SE, Minneapolis, MN 55455, USA

ARTICLE INFO

Article history:
Received 10 December 2014
Accepted 24 February 2015
Available online 11 March 2015

ABSTRACT

We describe a highly optimized implementation of MPI domain decomposition in a GPU-enabled, general-purpose molecular dynamics code, HOOMD-blue (Anderson and Glotzer, 2013). Our approach is inspired by a traditional CPU-based code, LAMMPS (Plimpton, 1995), but is implemented within a code that was designed for execution on GPUs from the start (Anderson et al., 2008). The software supports short-ranged pair force and bond force fields and achieves optimal GPU performance using an autotuning

Some available simulation packages that can (also) perform DPD simulations are:

•CULGI: The Chemistry Unified Language Interface, Culgi B.V., The Netherlands

•DL_MESO: Open-source mesoscale simulation software.

•DPDmacs

•ESPResSo: Extensible Simulation Package for the Research on Soft Matter Systems - Open-source

•Fluidix: The Fluidix simulation suite available from OneZero Software.

•GPIUTMD: Graphical processors for Many-Particle Dynamics

•Gromacs-DPD: A modified version of Gromacs including DPD.

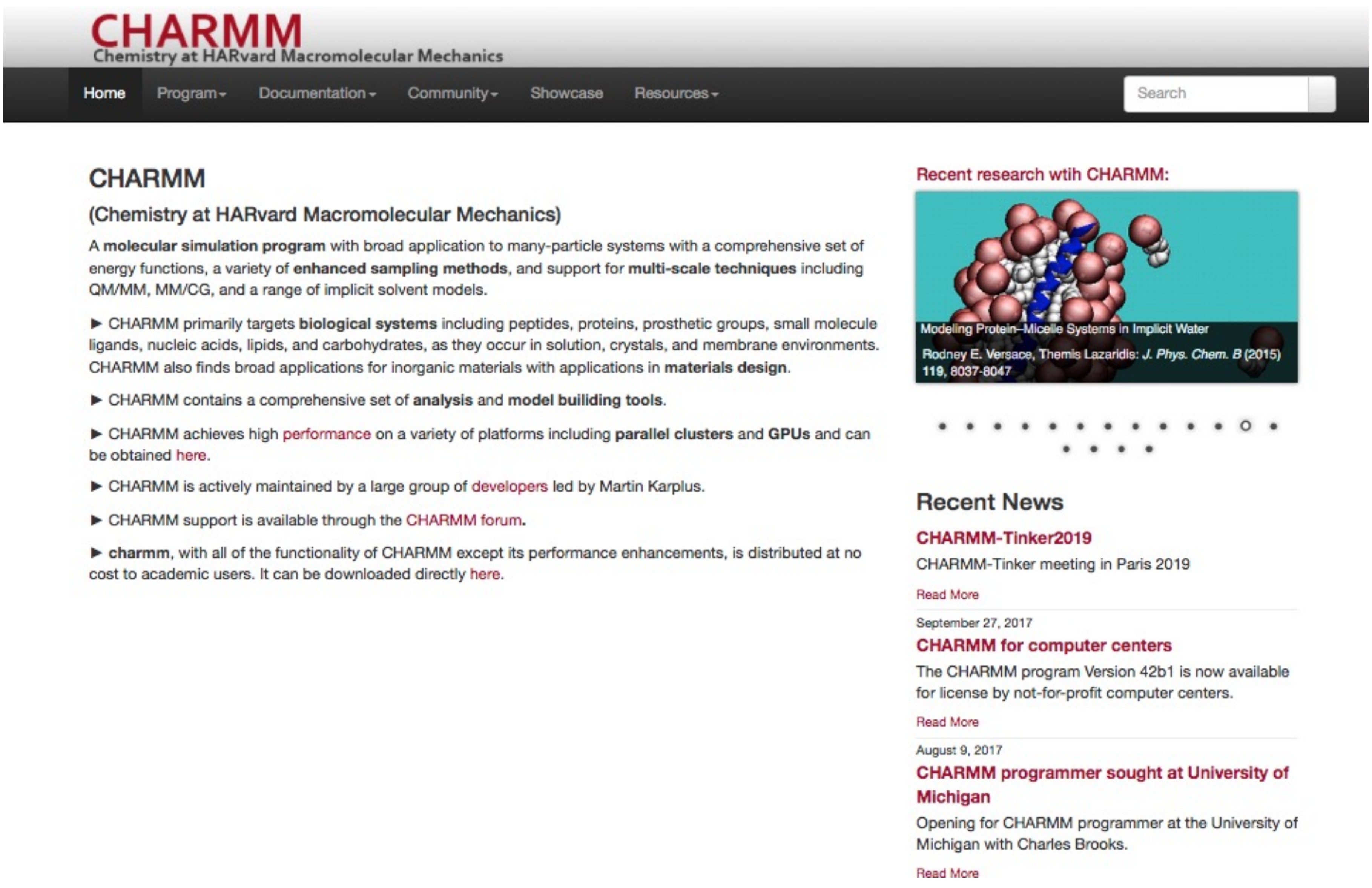
•HOOMD-blue: Highly Optimized Object-oriented Many-particle Dynamics—Blue Edition

•LAMMPS

•Materials Studio: Materials Studio - Modeling and simulation for studying chemicals and materials, Accelrys Software Inc.

•SYMPLER: A freeware SYMbolic ParticLE simulatoR from the University of Freiburg.

•SunlightDPD: Open-source (GPL) DPD software.



CHARMM

(Chemistry at HARvard Macromolecular Mechanics)

A **molecular simulation program** with broad application to many-particle systems with a comprehensive set of energy functions, a variety of **enhanced sampling methods**, and support for **multi-scale techniques** including QM/MM, MM/CG, and a range of implicit solvent models.

▶ CHARMM primarily targets **biological systems** including peptides, proteins, prosthetic groups, small molecule ligands, nucleic acids, lipids, and carbohydrates, as they occur in solution, crystals, and membrane environments. CHARMM also finds broad applications for inorganic materials with applications in **materials design**.

▶ CHARMM contains a comprehensive set of **analysis** and **model builiding tools**.

▶ CHARMM achieves high performance on a variety of platforms including **parallel clusters** and **GPUs** and can be obtained here.

▶ CHARMM is actively maintained by a large group of developers led by Martin Karplus.

▶ CHARMM support is available through the CHARMM forum**.**

▶ **charmm**, with all of the functionality of CHARMM except its performance enhancements, is distributed at no cost to academic users. It can be downloaded directly here.

Recent research wtih CHARMM:

Modeling Protein–Micelle Systems in Implicit Water

Rodney E. Versace, Themis Lazaridis: *J. Phys. Chem. B* (2015) 119, 8037-8047

Recent News

CHARMM-Tinker2019

CHARMM-Tinker meeting in Paris 2019

Read More

September 27, 2017

CHARMM for computer centers

The CHARMM program Version 42b1 is now available for license by not-for-profit computer centers.

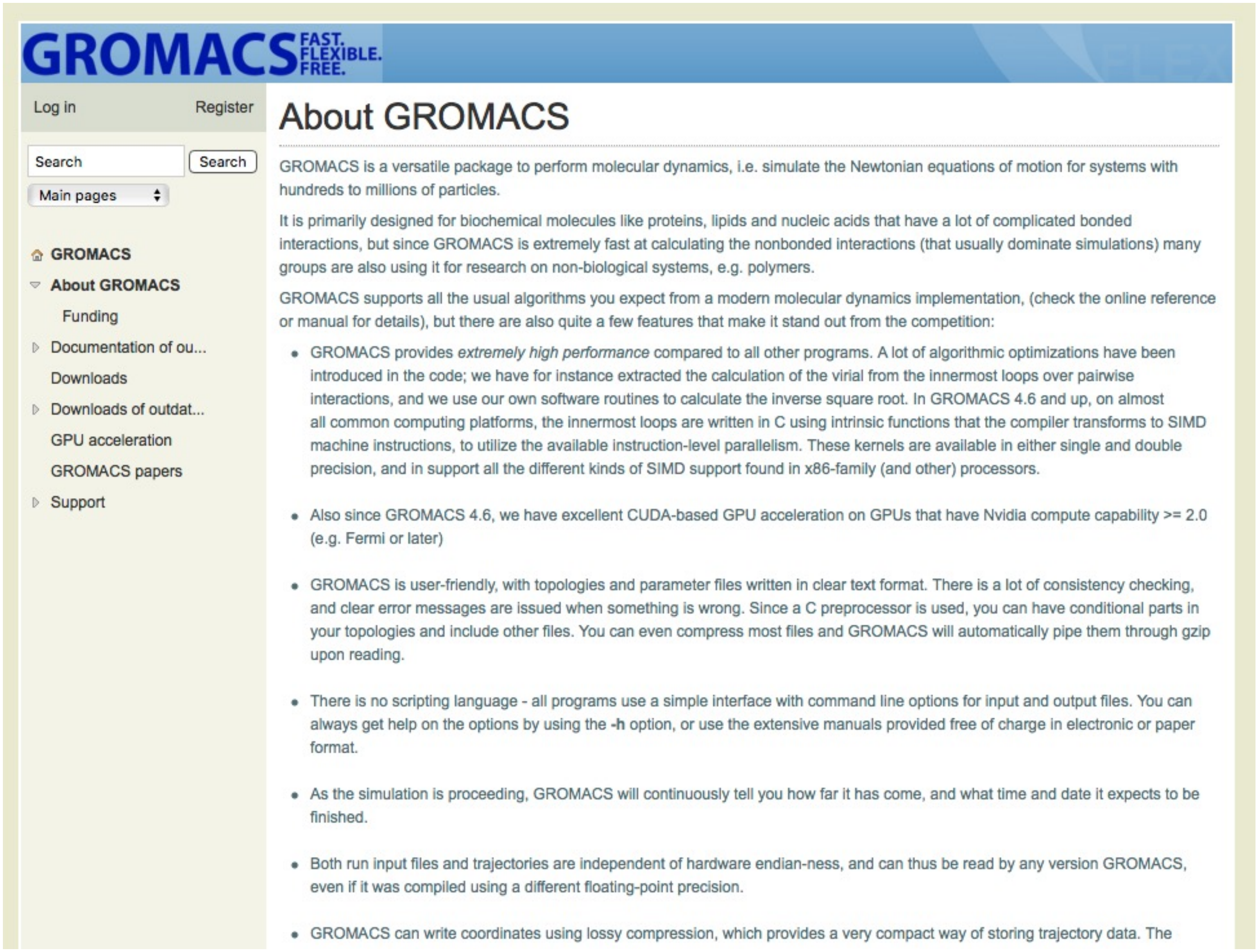
Read More

August 9, 2017

CHARMM programmer sought at University of Michigan

Opening for CHARMM programmer at the University of Michigan with Charles Brooks.

Read More



About GROMACS

GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles.

It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers.

GROMACS supports all the usual algorithms you expect from a modern molecular dynamics implementation, (check the online reference or manual for details), but there are also quite a few features that make it stand out from the competition:

- GROMACS provides *extremely high performance* compared to all other programs. A lot of algorithmic optimizations have been introduced in the code; we have for instance extracted the calculation of the virial from the innermost loops over pairwise interactions, and we use our own software routines to calculate the inverse square root. In GROMACS 4.6 and up, on almost all common computing platforms, the innermost loops are written in C using intrinsic functions that the compiler transforms to SIMD machine instructions, to utilize the available instruction-level parallelism. These kernels are available in either single and double precision, and in support all the different kinds of SIMD support found in x86-family (and other) processors.
- Also since GROMACS 4.6, we have excellent CUDA-based GPU acceleration on GPUs that have Nvidia compute capability >= 2.0 (e.g. Fermi or later)
- GROMACS is user-friendly, with topologies and parameter files written in clear text format. There is a lot of consistency checking, and clear error messages are issued when something is wrong. Since a C preprocessor is used, you can have conditional parts in your topologies and include other files. You can even compress most files and GROMACS will automatically pipe them through gzip upon reading.
- There is no scripting language - all programs use a simple interface with command line options for input and output files. You can always get help on the options by using the **-h** option, or use the extensive manuals provided free of charge in electronic or paper format.
- As the simulation is proceeding, GROMACS will continuously tell you how far it has come, and what time and date it expects to be finished.
- Both run input files and trajectories are independent of hardware endian-ness, and can thus be read by any version GROMACS, even if it was compiled using a different floating-point precision.
- GROMACS can write coordinates using lossy compression, which provides a very compact way of storing trajectory data. The