



APPLIED PHYSICS AND APPLIED MATHEMATICS
WITH MATERIALS SCIENCE AND ENGINEERING

MSAE E6273

TBA

Submitted To:
Renata Wentzcovitch
Professor
APAM Department

Submitted By :
Qi Zhang
qz2280
Spring 2017

Contents

1	Some algorithm for molecular dynamics	2
1.1	Closed methods	2
1.1.1	Euler's method	2
1.1.2	Modified Euler's method	2
1.1.3	Gear algorithm	2
1.2	Open methods	2
1.2.1	Verlet algorithm	2
1.2.2	Leap-frog algorithm	2
1.2.3	Beeman algorithm	2
2	Invariant molecular dynamics with variable cell shape: the history	3
2.1	Andersen's approach	3
2.2	Rahman–Parrinello approach	4
2.3	Wentzcovitch's approach	5
3	First principles molecular dynamics	6
3.1	Car–Parrinello approach	6
3.2	Wentzcovitch's approach	7
4	Molecular dynamics code	8
4.1	Main program	8
4.2	Setup steps	8
4.2.1	CRSTL subroutine	8
4.2.2	INIT subroutine	9
4.2.3	RDPP subroutine	9
4.2.4	RANV subroutine	9
4.3	Force calculation	10
4.3.1	FORCLJ subroutine	10
4.3.2	FORC subroutine	11
4.3.3	UPDG subroutine	11
4.3.4	SIGS and SIGP subroutine	11
4.4	Input file	12

1 Some algorithm for molecular dynamics

1.1 Closed methods

1.1.1 Euler's method

1.1.2 Modified Euler's method

1.1.3 Gear algorithm

1.2 Open methods

1.2.1 Verlet algorithm

The Verlet algorithm starts from the basic assumption

$$\frac{x_{n+1} + x_{n-1} - 2x_n}{h^2} = F(x_n), \quad (1)$$

which is, the second-order central difference method.

1.2.2 Leap-frog algorithm

1.2.3 Beeman algorithm

Beeman's algorithm, which is used in the code, is also an open method. The basic equations are

$$x_{n+1} = x_n + hv_n + \frac{2}{3}h^2F_n - \frac{1}{6}h^2F_{n-1}, \quad (2a)$$

$$v_{n+1} = v_n + \frac{1}{3}hF_{n+1} + \frac{5}{6}hF_n - \frac{1}{6}hF_{n-1}, \quad (2b)$$

where $F_n = F(x_n)$ is the force on n th step.¹ This algorithm is equivalent to the Verlet algorithm. First consider

$$v_{n-1} = \frac{1}{h}(x_n - x_{n-1}) - \frac{1}{6}h(4F_{n-1} - F_{n-2}), \quad (3)$$

by transforming (2a), then use (2b) we derive

$$v_n = \frac{1}{h}(x_n - x_{n-1}) + \frac{1}{6}hF_{n-2} + \frac{5}{6}h(F_n - F_{n-1}) + \frac{1}{3}hF_{n+1}, \quad (4)$$

substitute (4) back into (2a) we recover (1).

2 Invariant molecular dynamics with variable cell shape: the history

Here lists some key papers to keep track the purposes and methods of this simulation.

2.1 Andersen's approach

Andersen first extends the molecular dynamics field to ensembles other than micro-canonical ensemble.² In his ground-breaking paper, he proposed ways to calculate properties average over isoenthalpic-isobaric (NPH) ensemble. In the article, he introduced a Lagrangian

$$\mathcal{L}(\rho, \dot{\rho}, Q, \dot{Q}) = \frac{1}{2} m Q^{\frac{2}{3}} \sum_{i=1}^N \dot{\rho}_i \cdot \dot{\rho}_i - \sum_{i<j=1}^N u(Q^{\frac{1}{3}} \rho_{ij}) + \frac{1}{2} M \dot{Q}^2 - \alpha Q, \quad (5)$$

where $\rho_i = \mathbf{r}_i / V^{\frac{1}{3}}$, $i = 1, 2, \dots, N$, is called scaled coordinates. Here α and M are constants, $\frac{1}{2} M \dot{Q}$ now is regarded as a kinetic energy with fictitious mass M , and αQ is regarded as a potential energy for the motion of Q . The generalized momentum conjugate to ρ is

$$\pi_i = \frac{\partial \mathcal{L}_2}{\partial \dot{\rho}_i} = m Q^{\frac{2}{3}} \dot{\rho}_i, \quad (6)$$

and which for Q is

$$\Pi = \frac{\partial \mathcal{L}_2}{\partial \dot{Q}} = M \dot{Q}. \quad (7)$$

The Hamiltonian is thus

$$\begin{aligned} \mathcal{H}(\rho, \pi, Q, \Pi) &= \sum_{i=1}^N \pi_i \cdot \dot{\rho}_i + \Pi \dot{Q} - \mathcal{L}_2(\rho, \dot{\rho}, Q, \dot{Q}) \\ &= \frac{1}{2m Q^{\frac{2}{3}}} \sum_{i=1}^N \pi_i \cdot \pi_i + \sum_{i<j=1}^N u(Q^{\frac{1}{3}} \rho_{ij}) + \frac{1}{2M} \Pi^2 + \alpha Q. \end{aligned} \quad (8)$$

So the equations of motions are

$$\dot{\rho}_i = \frac{\partial \mathcal{H}}{\partial \pi_i} = \frac{\pi_i}{m Q^{\frac{2}{3}}}, \quad (9)$$

$$\dot{\pi}_i = -\frac{\partial \mathcal{H}}{\partial \rho_i} = -Q^{\frac{1}{3}} \sum_{\substack{j=1 \\ j \neq i}}^N \frac{u' \rho_{ij}}{|\rho_{ij}|}, \quad (10)$$

$$\dot{Q} = \frac{\partial \mathcal{H}}{\partial \Pi} = \frac{\Pi}{M}, \quad (11)$$

$$\dot{\Pi} = -\frac{\partial \mathcal{H}}{\partial Q} = -\frac{1}{3Q} \left(-\frac{1}{m Q^{\frac{2}{3}}} \sum_{i=1}^N \pi_i \cdot \pi_i + Q^{\frac{1}{3}} \sum_{i<j}^N \rho_{ij} u'(Q^{\frac{1}{3}} \rho_{ij}) + 3\alpha Q \right). \quad (12)$$

With these equations, the trajectory of the scaled system are given by $\rho(t)$, $\pi(t)$, $Q(t)$, and $\Pi(t)$.

Use this trajectory, any function's time average are given by

$$\overline{G} = \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T dt G(\rho(t), \pi(t), Q(t), \Pi(t)), \quad (13)$$

and this can be given by the average of an NE ensemble. That is,

$$G_{NE}(N, E) = \frac{1}{N! \Omega(N, E)} \int d\rho \int d\pi \int dQ \int d\Pi \delta(\mathcal{H}(\rho, \pi, Q, \Pi) - E) G(\rho(t), \pi(t), Q(t), \Pi(t)), \quad (14)$$

where

$$\Omega(N, E) = \frac{1}{N!} \int d\rho \int d\pi \int dQ \int d\Pi \delta(\mathcal{H}(\rho, \pi, Q, \Pi) - E). \quad (15)$$

The scaled system has a correspondence

$$V = Q, \quad (16)$$

$$\mathbf{r}_i = Q^{\frac{1}{3}} \rho_i, \quad (17)$$

$$\mathbf{p}_i = \pi_i / Q^{\frac{1}{3}}, \quad (18)$$

to the phase space of a system spanned by $\mathbf{r}_i, i = 1, \dots, N$ and $\mathbf{p}_i, i = 1, \dots, N$, where \mathbf{r}_i is the particle coordinate, \mathbf{p}_i is its momentum. Thus the trajectory $\rho(t), \pi(t), Q(t)$, and $\Pi(t)$ have its correspondence $V(t), \mathbf{r}_i(t)$ and $\mathbf{p}_i(t)$ by (16). Since the time average \overline{F} of any function F derived by this trajectory is the same as an isoentahlpic-isobaric ensemble average of F_{NPH} , and $\overline{G} = \overline{F}$, thus we get the NPH ensemble average. The ensemble pressure P is α in (5) indeed.

2.2 Rahman–Parrinello approach

Rahman and Parrinello then proposed a method to perform MD simulations which allows volume and shape of the MD cell to change with time.³ If the MD cell edges are \mathbf{a}, \mathbf{b} and \mathbf{c} , which are time dependent. Stack them to form a matrix $\mathbf{h} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, and the volume of a cell is then $\Omega = \mathbf{a} \cdot \mathbf{b} \times \mathbf{c}$, the metric tensor is $\mathbf{g} = \mathbf{h}^T \mathbf{h}$. Let $\mathbf{r}_i = \xi_i \mathbf{a} + \eta_i \mathbf{b} + \zeta_i \mathbf{c} = \mathbf{h} \mathbf{s}_i$, where \mathbf{s}_i store its coordinates ξ_i, η_i , and ζ_i , each from 0 to 1. Then they introduced a Lagrangian

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^N m_i \dot{\mathbf{s}}_i^T \mathbf{g} \dot{\mathbf{s}}_i - \sum_{i=1}^N \sum_{i < j} \phi(r_{ij}) + \frac{1}{2} W \text{Tr}(\dot{\mathbf{h}}^T \dot{\mathbf{h}}) - P \Omega, \quad (19)$$

where $r_{ij}^2 = (\mathbf{s}_i - \mathbf{s}_j)^T \mathbf{g} (\mathbf{s}_i - \mathbf{s}_j)$, P is the external hydrostatic pressure, $\phi(r_{ij})$ is the pair potential, W is the fictitious mass.

Then the equations of motion are

$$\ddot{\mathbf{s}}_i = \frac{1}{m_i} \sum_{j \neq i} \frac{\phi'(r_{ij})}{r_{ij}} (\mathbf{s}_i - \mathbf{s}_j) - \mathbf{g}^{-1} \dot{\mathbf{g}} \dot{\mathbf{s}}_i, \quad i, j = 1, 2, \dots, N, \quad (20a)$$

$$\ddot{\mathbf{h}} = \frac{1}{W} (\Pi - P) \sigma, \quad (20b)$$

where $\sigma = \{\mathbf{a} \times \mathbf{b}, \mathbf{b} \times \mathbf{c}, \mathbf{c} \times \mathbf{a}\}$, matrix Π is given by

$$\Omega\Pi = \sum_{i=1}^N m_i \mathbf{v}_i \mathbf{v}_i^\top + \sum_{i=1}^N \sum_{i<j} \frac{\phi'(r_{ij})}{r_{ij}} (\mathbf{r}_i - \mathbf{r}_j)(\mathbf{r}_i - \mathbf{r}_j)^\top, \quad (21)$$

with $\mathbf{v}_i = h\mathbf{s}_i$. Then Andersen's equations of motion is a special case of (20), where $h = \text{diag}(\Omega^{\frac{1}{3}}, \dots, \Omega^{\frac{1}{3}})$ and $g^{-1}\dot{g} = \frac{2\Omega}{3\Omega}$. Though his equation for \dot{V} cannot be obtained from (20b). But this Lagrangian also results in a isoenthalpic, isobaric ensemble, though with a small correction from the third term.

2.3 Wentzcovitch's approach

Wentzcovitch stated that Rahman–Parrinello method is dependent on the choice of cell edges.⁴ For different \mathbf{a} , \mathbf{b} , and \mathbf{c} , the fictitious kinetic energy K_L term could be different. For MD simulation of $\sim 10^2$ particles the problem may not be too serious. But if the cell experience a modular transformation, the nominal value of K_L , and the resulted forces and trajectories will be dependent on the choice of h . Then she proposed a new Lagrangian

$$\mathcal{L} = \sum_{i=1}^N \dot{\mathbf{q}}_i^\top d \dot{\mathbf{q}}_i - \sum_{i=1}^N \sum_{i<j} \phi(r_{ij}) + \frac{W}{2} \text{Tr}(\dot{\epsilon} \dot{\epsilon}^\top) - P\Omega, \quad (22)$$

where ϵ is the strain, \mathbf{q}_i is the defined as $\mathbf{r}_i = (1 + \epsilon)\mathbf{q}_i$, thus $d = (1 + \epsilon)^\top(1 + \epsilon)$. The equations of motion are

$$\ddot{\mathbf{q}}_i = -\frac{1}{m_i} \sum_{\substack{i,j=1 \\ j \neq i}}^N \frac{\phi'(r_{ij})}{r_{ij}} (\mathbf{q}_i - \mathbf{q}_j) - d^{-1} \dot{d} \dot{\mathbf{q}}_i, \quad (23)$$

$$\ddot{\epsilon} = \frac{\Omega}{W} (\Pi - P) ((1 + \epsilon)^\top)^{-1}. \quad (24)$$

An equivalent form of (22) is

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^N m_i \dot{\mathbf{s}}_i^\top g \dot{\mathbf{s}}_i - \sum_{i=1}^N \sum_{i<j} \phi(r_{ij}) + \frac{1}{2} W \text{Tr}(\dot{h} f_0 \dot{h}^\top) - P\Omega, \quad (25)$$

with $f_0 = \sigma_0^\top \sigma_0$, where $\sigma_0 = \{\mathbf{a}_0 \times \mathbf{b}_0, \mathbf{b}_0 \times \mathbf{c}_0, \mathbf{c}_0 \times \mathbf{a}_0\}$. These vectors form the matrix h_0 , and the h in (19) is $h = (1 + \epsilon)h_0$. This will lead to (24) to be restated by using \ddot{h} as

$$\ddot{h} = \frac{1}{W} (\Pi - P) \sigma f_0^{-1}, \quad (26)$$

which makes it easier to implement in code.

If further modify the fictitious kinetic energy term K_L to be $\frac{1}{2} W \text{Tr}(\dot{h} \sigma^\top \sigma \dot{h}^\top)$, then (26) will become

$$\ddot{h} = \frac{1}{W} (\Pi - P) \sigma f^{-1} + \frac{1}{2} \text{Tr} \left(e \frac{\partial f}{\partial h} \right) f^{-1} - \dot{h} \dot{f} f^{-1}, \quad (27)$$

with $f = \sigma^\top \sigma$, $e = \dot{h}^\top \dot{h}$. This Lagrangian coincide with (5) in isoshape limit.

She then performed MD simulations using (19) and (22), as well as the modified Lagrangian under zero temperature to compare. Her method eliminated the symmetry breaking introduced by Rahman–Parrinello non-invariant fictitious part of the dynamics.

3 First principles molecular dynamics

The first quantum molecular dynamics (QMD), a.k.a., “*ab initio*” or “first principles” simulation was carried out by Car and Parrinello in 1985.⁵ Their and subsequent work have led to a great development on treating real, complex molecules, solids and liquids with forces derived by density functional theory (DFT).⁶ Now we are going to have a look on this work.

3.1 Car–Parrinello approach

Car and Parrinello were dedicating to find the ground-state electronic solution for the electrons as the nuclei move. Their MD simulation was based on 2 assumptions: the validity of classical mechanics to describe ionic motions, and Born–Oppenheimer approximation to separate the nuclear and electronic coordinates. They invented a new strategy other than the Metropolis Monte Carlo method introduced by Kirkpatrick, Gelatt, and Vecchi,⁷ the so-called “dynamical simulated annealing” method. By this method, they could minimize the energy of electrons and solve for the motion of the nuclei simultaneously. This is achieved by introducing a Lagrangian

$$\mathcal{L} = \sum_i \frac{1}{2} \mu \int d\mathbf{r} |\psi_i(\mathbf{r})|^2 + \sum_I \frac{1}{2} M_I \dot{\mathbf{R}}_I^2 + \sum_v \frac{1}{2} \mu_v \dot{\alpha}_v^2 - E[\{\psi_i\}, \{\mathbf{R}_I\}, \{\alpha_v\}], \quad (28)$$

with the holonomic constraints

$$\int d\mathbf{r} \psi_i^*(\mathbf{r}, t) \psi_j(\mathbf{r}, t) = \delta_{ij}, \quad (29)$$

where M_I are the physical ionic masses, while μ and μ_v are just arbitrary parameters of appropriate units. In (28), the dynamics associated with the $\{\psi_i\}$ ’s and the $\{\alpha_v\}$ ’s is fictitious and should only be considered as a numerical tool. Combine (28) and (29) we derive

$$\begin{aligned} \mathcal{L}' = \sum_i \frac{1}{2} \mu \int d\mathbf{r} |\psi_i(\mathbf{r})|^2 + \sum_I \frac{1}{2} M_I \dot{\mathbf{R}}_I^2 + \sum_v \frac{1}{2} \mu_v \dot{\alpha}_v^2 - E[\{\psi_i\}, \{\mathbf{R}_I\}, \{\alpha_v\}] \\ + \sum_{ij} \Lambda_{ij} \left[\int d\mathbf{r} \psi_i^*(\mathbf{r}, t) \psi_j(\mathbf{r}, t) - \delta_{ij} \right], \end{aligned} \quad (30)$$

where $E[\{\psi_i\}, \{\mathbf{R}_I\}, \{\alpha_v\}]$ is a functional, Λ_{ij} is a Lagrangian multiplier. Thus the corresponding equations of motion are

$$\mu \ddot{\psi}_i(\mathbf{r}, t) = \frac{\partial \mathcal{L}'}{\partial \psi_i^*(\mathbf{r}, t)} = -\frac{\delta E}{\delta \psi_i^*(\mathbf{r}, t)} + \sum_k \Lambda_{ik} \psi_k(\mathbf{r}, t) = -\hat{H} \psi_i(\mathbf{r}, t) + \sum_k \Lambda_{ik} \psi_k(\mathbf{r}, t) \quad (31)$$

$$M_I \ddot{\mathbf{R}}_I = \frac{\partial \mathcal{L}'}{\partial \mathbf{R}_I} = \mathbf{F}_I = -\frac{\partial E}{\partial \mathbf{R}_I}, \quad (32)$$

$$\mu_v \ddot{\alpha}_v = \frac{\partial \mathcal{L}'}{\partial \alpha_v} = -\frac{\partial E}{\partial \alpha_v}, \quad (33)$$

where \hat{H} is the hamiltonian. These equations can be simulated by the Verlet algorithm, that is,

$$\psi_i^{n+1}(\mathbf{r}) = 2\psi_i^n(\mathbf{r}) - \psi_i^{n-1}(\mathbf{r}) - \frac{h^2}{\mu} \left[\hat{H}\psi_i^n(\mathbf{r}) - \sum_k \Lambda_{ik}\psi_k(\mathbf{r}, t) \right], \quad (34a)$$

$$\mathbf{R}_I^{n+1} = 2\mathbf{R}_I^n - \mathbf{R}_I^{n-1} + \frac{h^2}{M_I} \mathbf{F}_I, \quad (34b)$$

$$\alpha_v^{n+1} = 2\alpha_v^n - \alpha_v^{n-1} - \frac{h^2}{\mu_v} \frac{\partial E}{\partial \alpha_v}, \quad (34c)$$

where h is the time step interval. The holonomic constraints are handled by a method called SHAKE.⁸

At stationary state, all time derivatives are 0 so that (34a) leads to

$$\hat{H}\psi_i^n(\mathbf{r}) = \sum_k \Lambda_{ik}\psi_k(\mathbf{r}, t), \quad (35)$$

which shows that Λ is the transpose of usual Kohn–Sham hamiltonian matrix H , i.e., $\Lambda_{ij} = H_{ji}$. Thus the eigenvalues of Λ coincides with Kohn–Sham eigenvalues. And the solution is stationary if and only if the Kohn–Sham energy is at a variational minimum.⁶ This method leads to the possibility to consider real dynamics of the nuclei in *ab initio* electronic structure algorithms.

3.2 Wentzcovitch's approach

In Car–Parrinello method, as (34) has shown, at each timestep $\hat{H}\psi_i$ is calculated. However, this is the most time-consuming operation in the algorithm.

4 Molecular dynamics code

The following part is an introduction on how to use Wentzcovitch's code to perform MD simulation. The code is based on one of her papers.⁴

There are 4 types of simulations in code, denoted as `md`, `cd`, `nd`, `sd`, respectively. The first one means "molecular dynamics", based on Andersen's paper,² the second one means "cell dynamics", based on RahmanParrinello approach,³ the main equations are (20). The third case is called "new cell dynamics" and is based on (20a) and (27), the last one is called "strain dynamics" and is based on (20a) and (26).

The simulation is done under zero temperature, as stated above.

`mxdtype` denotes the array dimension for type of atoms, in the input file there will be a line labeled by `(ntype)`, it contains a scalar, i.e., number of atom types, so `mxdtype` = 1.

4.1 Main program

Then we start a MD loop. First update the current step `nzero`, and then update those accumulators `acu`, `ack` and `acp`, then calculate their averages by dividing `nzero`. Then call `move` subroutine, `p` is updated during this step, however `vcell` is not. Then calculate pV and do a corresponding accumulator `acpv` and average `avpv`. And then, update arrays like `utm` with length `nstep` (read from input). Then calculate the new temperature, `tnew`, by assuming equipartition theorem $\bar{E} = \frac{3}{2}k_B T$, where \bar{E} is the average kinetic energy `avk`. Then do rescaling on Cartesian velocity `v` and reduced velocity `ratd`.

At the end of MD loop, write energy outputs to file `e`: total potential energy U_t as `utm` array, total kinetic energy E_t as `ekintm` array, total energy $\mathcal{E}_t = E_t + U_t$ as `etotm` array, and total $p\Omega$ as `pvm`. And, write decomposed energy outputs to file `eal`: atomic contribution to total potential energy as `utam` array, to total kinetic energy as `ekam` array, to total energy as `etam` array; and lattice contribution to total potential energy as `utlm` array, to total kinetic energy as `eklm` array, to total energy as `etlm` array. It could be also understood as ionic and cell contributions, respectively. Then write average accumulated outputs to file `ave`, by `avum`, `avkm`, etc. Another important output file is `avec`, which stores lengths and angles between primitive cell vectors for each MD step, denoted by `bmodm` and `thetam` matrices. `bmodm` at each MD step is set to lattice vectors moduli `avmod`, computed by `MOVE`. `tv` file stores the "instantaneous" temperature and volume for each MD step.

4.2 Setup steps

4.2.1 CRSTL subroutine

This subroutine does some pre-setting work before `INIT`. The `avec` parameter is the 3 lattice primitive vectors in Cartesian coordinates, i.e., $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, denoted as \mathbf{h} in Wentzcovitch's paper.⁴ `rat` is the atomic positions in terms of lattice primitive vectors, `ratd` is its first-order time derivative. `g` is just $\mathbf{g} = \mathbf{h}^T \mathbf{h}$ and `gm1` is \mathbf{g}^{-1} . `cmass` is the fictitious mass W and `press` is the external pressure P in Wentzcovitch's paper.⁴

Table 1: List of variables.

	variable name	variable symbol	variable	write to file
total	potential energy	U_t	utm	
	total kinetic energy	E_t	ekintm	
	energy	\mathcal{E}_t	etotm	e
	$p\Omega$	$p\Omega$	pvm	
atomic contribution to total	potential energy		utam	
	kinetic energy		ekam	
	energy		etam	
lattice contribution to total	potential energy		utlm	eal
	kinetic energy		eklm	
	energy		etlm	
average of accumulated	potential energy	\bar{U}_t	avum	
	kinetic energy	\bar{E}_t	avkm	ave

4.2.2 INIT subroutine

First, call **RANV** subroutine, and initialize **avecd**, **avec2d** and **gd**, etc. **ilj** is the only variable set by hand in code, if it is equal to 1, the code will call **FORCLJ** subroutine, else it will call **FORC** subroutine, see 4.3.2 and 4.3.1 for differences.

Then if **calc** flag is not set to **md** and **mm** then **gmgd** is calculated.

If **calc** flag is set to **nd** or **nm** then **SIGP** is called, if it is set to **sd** or **sm** then **SIGS** is called, see 4.3.4 for more detail. With $\sigma_0 = \{\mathbf{a}_0 \times \mathbf{b}_0, \mathbf{b}_0 \times \mathbf{c}_0, \mathbf{c}_0 \times \mathbf{a}_0\} = \frac{V_0}{2\pi} \{\mathbf{c}_0^*, \mathbf{a}_0^*, \mathbf{b}_0^*\}$, we know $\ddot{\mathbf{d}} = \frac{1}{V_0} \ddot{\mathbf{h}} \sigma_0$. Then we do strain symmetrization, $\ddot{d}_{ij} = \frac{1}{2}(\ddot{d}_{ij} + \ddot{d}_{ji})$. Then $\ddot{\mathbf{h}} = \ddot{\mathbf{d}} \mathbf{h}_0$.

If in **nd** or **nm**, we do $\text{Tr}(\dot{\mathbf{h}}^T \sigma \sigma \dot{\mathbf{h}})$.

If in **sd** or **sm**, we do $\text{Tr}(\dot{\mathbf{h}}^T \sigma_0 \sigma_0^T \dot{\mathbf{h}})$.

Then in both cases, we do **ekl** = **ekl** + $\text{Tr}(\dot{\mathbf{h}}^T \dot{\mathbf{h}})$.

Total kinetic energy is $E_t = \mathbf{eka} + \frac{1}{2} W \mathbf{ekl}$, where W is the fictitious mass, total potential energy $U_t = ??? E_t + P\Omega$, where P is the external pressure. Total energy is $\mathcal{E}_t = E_t + U_t$.

4.2.3 RDPP subroutine

This subroutine reads the pair-potential file. Here **ntype** is the number of different types of atoms. It reads the potential of i th atom and j th atom, where $j \geq i$. So in the code we need to do some assignments like

$$U_{ij} = U_{ji}, \quad (36)$$

$$\mathbf{F}_{ij} = \mathbf{F}_{ji}, \quad (37)$$

where U_{ji} and \mathbf{F}_{ji} are what we read from file, thus we can save time on IO operations.

4.2.4 RANV subroutine

This subroutine is dedicated for setting up Maxwell distributed random velocities at temperature T .

4.3 Force calculation

4.3.1 FORCLJ subroutine

The subroutine will first calculate the Cartesian distance between 2 atoms. It will call `ELJ` subroutine for each pair of atoms i and j , in which the force between these 2 atoms in both Cartesian and reduced coordinates are calculated, and denoted as `fo` and `fos`, respectively. More detailedly, `ELJ` uses the Lennard–Jones scheme.

The Lennard–Jones potential has general form

$$U(r) = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right), \quad (38)$$

where r is the distance between 2 atoms, σ and ϵ are constants. The force is calculated by

$$\mathbf{F}_{ji} = \frac{24\epsilon}{\sigma} (\mathbf{r}_j - \mathbf{r}_i) \left(2 \left(\frac{\sigma}{r} \right)^{14} - \left(\frac{\sigma}{r} \right)^8 \right), \quad (39)$$

where $r = |\mathbf{r}_j - \mathbf{r}_i|$, or equivalently, its scalar form

$$F(r) = \frac{24\epsilon}{\sigma} \left(2 \left(\frac{\sigma}{r} \right)^{13} - \left(\frac{\sigma}{r} \right)^7 \right), \quad (40)$$

which is plotted in Fig. 1.

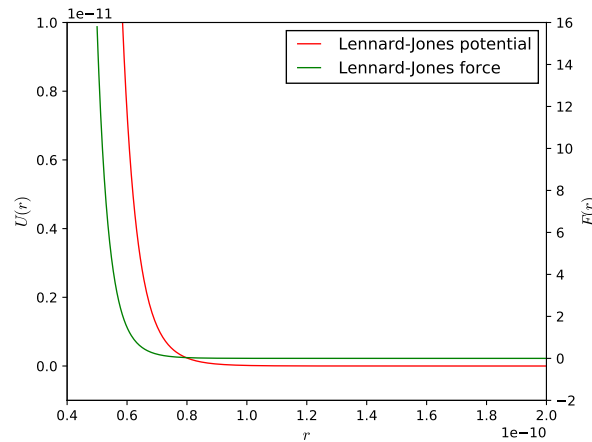


Figure 1: Lennard–Jones potential and force for Ar, as a function of distance between 2 atoms, with $\sigma = 3.405 \times 10^{-10}\text{m}$, $\epsilon = 1.654 \times 10^{-21}\text{J}$.⁹

Back to `FORCLJ` subroutine, it considers 2 cases: particle j and i in the same cell and in different cells, the first case is trivial, in the second case, we need to add multiple times of primitive vectors and then calculate distance r_{ji} , and the rest steps are the same as first case. In both cases, we do not consider interaction out of the radius r_{cut} . It appends `fo` and `fos` modified by `ELJ` subroutine to the `f` and `fs`, respectively.

4.3.2 FORC subroutine

If **FORCLJ** is used for analytical computation, then this subroutine is used as numerical methods. We have talked about **RDPP** subroutine before, stating that it reads $n(n+1)$ columns of potentials and forces, denoted as **vpp** and **fpp**, respectively. The potentials interpolated by quadratic functions, the forces are also calculated similar process.

4.3.3 UPDG subroutine

This part is used to update several quantities of cell during calculations.

Here **avec** is h , **avecd** is \dot{h} , thus $g = h^T h$ and $\dot{g} = \dot{h}^T h + h \dot{h}^T$, **gm1** is g^{-1} and **gmgd** is $g^{-1} \dot{g}$. **sigma** is the reciprocal lattice vectors σ .

It first reads a flag **itg** to see if σ and V need to be calculated, if it is 'yes', then do the following things: Firstly calculates σ by the components of h , and then calculate the MD cell volume by

$$V = \sigma \cdot h, \quad (41)$$

and then calculate g , \dot{g} , g^{-1} , $g^{-1} \dot{g}$, respectively.

4.3.4 SIGS and SIGP subroutine

SIGP subroutine is used to calculate lattice vectors accelerations based on 'new cell dynamics', i.e., according to (20a) and (27); while **SIGS** is based on 'strain dynamics', i.e., (20a) and (26). At last, the result returned is \ddot{h} .

In **SIGS** subroutine, firstly calculates f_0^{-1} , by definition it is

$$f_0^{-1} = \frac{h_0^T h_0}{V_0^2}, \quad (42)$$

then set an argument **avint** to temporally store \ddot{h} , and perform calculation $\ddot{h} = \ddot{h} f_0^{-1}$. This subroutine also returns \ddot{h} in the end.

In **SIGP** subroutine, firstly calculates f^{-1} , by definition we know it is

$$f^{-1} = \frac{h^T h}{V^2}, \quad (43)$$

and $e = \dot{h}^T \dot{h}$, as stated above, as well as a $3 \times 3 \times 3 \times 3$ tensor

$$f' = \frac{\partial f}{\partial h_{ij}} = (\sigma'_{ij})^T \sigma + \sigma^T \sigma'_{ij}, \quad (44)$$

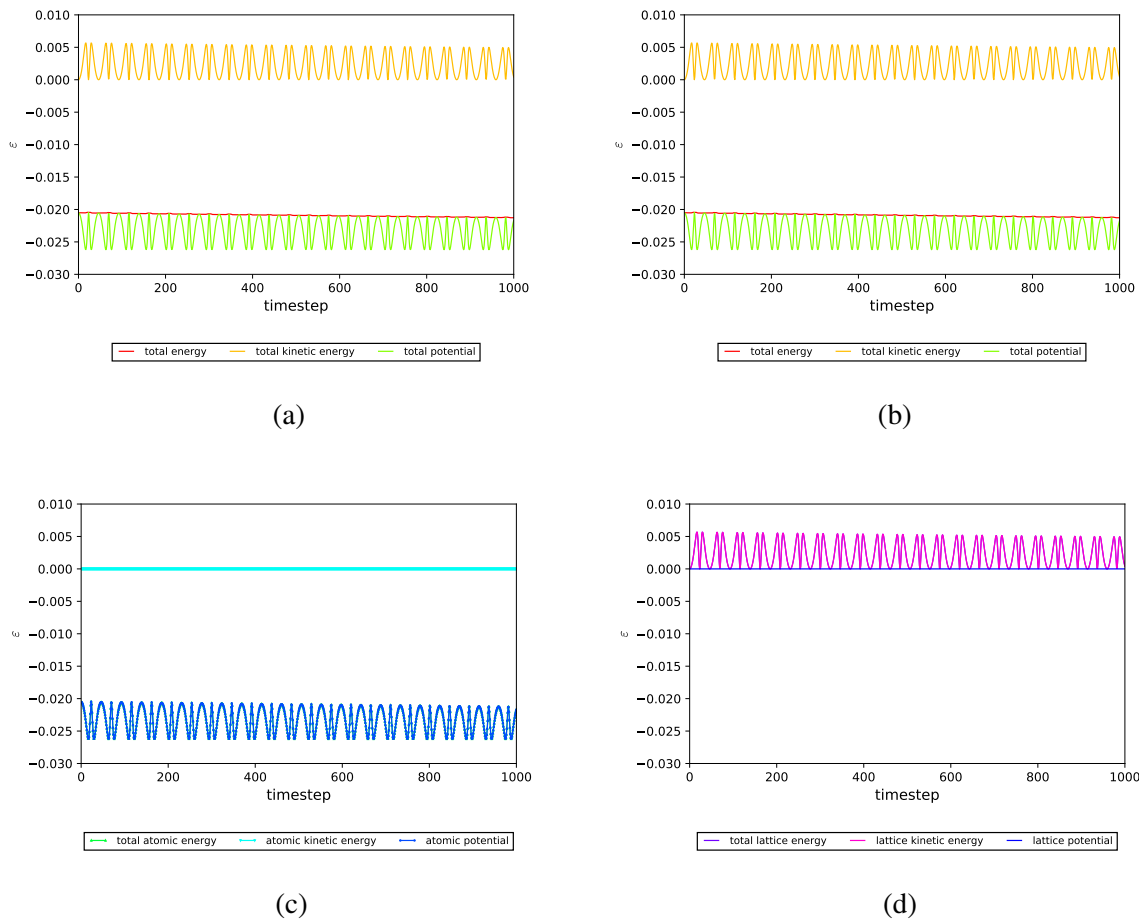
where σ' is denoted as **sigmap** in code, another $3 \times 3 \times 3 \times 3$ tensor. $\dot{f}_0 = \dot{\sigma}^T \sigma + \sigma^T \dot{\sigma}$ is also calculated. $f^{-1} = \sum_{k,l} e_{lk} f'_{ijkl}$, and $\sigma^{-1} = h \dot{f}$. Then final returns \ddot{h} . As stated above, $h = (1 + \epsilon)h_0$, where $h_0 = \{a_0, b_0, c_0\}$.

4.4 Input file

Now let's have a look at the input file.

The second line is its calculation type, denoted by `calc`. `cmass` is the fictitious mass W talked in R and P's and Wentzcovitch's scheme, and `press` is the external pressure P .

Figure 2: Input 1



References

- [1] JW Dufty, JJ Brey, A Santos, G Ciccotti, and WG Hoover. Molecular dynamics simulation of statistical mechanical systems. In *Proc. Int. School of Physics Enrico Fermi, North-Holland, Amsterdam*, 1986.
- [2] H C Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of chemical physics*, 1980.
- [3] M Parrinello and A Rahman. Crystal Structure and Pair Potentials: A Molecular-Dynamics Study. *Physical Review Letters*, 45(14):1196–1199, October 1980.
- [4] R M Wentzcovitch. Invariant molecular-dynamics approach to structural phase transitions. *Physical Review B*, 1991.
- [5] R Car and M Parrinello. Unified approach for molecular dynamics and density-functional theory. *Physical Review Letters*, 1985.
- [6] Richard M Martin. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2004.
- [7] S Kirkpatrick, C D Gelatt, and M P Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.
- [8] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J C Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(3):327–341, March 1977.
- [9] Richard J. Gonsalves. Molecular Dynamics of Lennard-Jones System. <http://www.physics.buffalo.edu/phy516/jan28.pdf>. Accessed: 2017-04-10.