COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

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

List of Figures

List of Tables

1 Some alogrithm 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

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

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

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

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

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

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

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

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

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

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

where $\boldsymbol{\rho}_i = \boldsymbol{r}_i / V^{\frac{1}{3}}$, $i = 1, 2, \ldots, N$, 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 $\boldsymbol{\rho}$ is

$$\boldsymbol{\pi}_i = \frac{\partial \mathcal{L}_2}{\partial \dot{\boldsymbol{\rho}}_i} = m Q^{\frac{2}{3}} \boldsymbol{\rho}_i \,, \tag{6}$$

and which for Q is

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

The Hamiltonian is thus

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

So the equations of motions are

$$\dot{\boldsymbol{\rho}}_i = \frac{\partial \mathcal{H}}{\partial \boldsymbol{\pi}_i} = \frac{\boldsymbol{\pi}_i}{m Q^{\frac{2}{3}}} \,, \tag{9}$$

$$\dot{\boldsymbol{\pi}}_i = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{\rho}_i} = -Q^{\frac{1}{3}} \sum_{\substack{j=1 \\ j \neq i}} \frac{u' \boldsymbol{\rho}_{ij}}{|\boldsymbol{\rho}_{ij}|} \,, \tag{10}$$

$$\dot{Q} = \frac{\partial \mathcal{H}}{\partial \Pi} = \frac{\Pi}{M} \,, \tag{11}$$

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

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

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

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

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

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

where

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

The scaled system has a correspondence

$$V = Q, \tag{16}$$

$$\boldsymbol{r}_i = Q^{\frac{1}{3}} \boldsymbol{\rho}_i\,, \tag{17}$$

$$\boldsymbol{p}_i = \boldsymbol{\pi}_i / Q^{\frac{1}{3}}\,, \tag{18}$$

to the phase space of a system spanned by $\boldsymbol{r}_i, i = 1, \ldots, N$ and $\boldsymbol{p}_i, i = 1, \ldots, N$, where $\boldsymbol{r}_i$ is the particle coordinate, $\boldsymbol{p}_i$ is its momentum. Thus the trajectory $\boldsymbol{\rho}(t)$, $\boldsymbol{\pi}(t)$, $Q(t)$, and $\Pi(t)$ have its correspondence $V(t)$, $\boldsymbol{r}_i(t)$ and $\boldsymbol{p}_i(t)$ 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 $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$, which are time dependent. Stack them to form a matrix $h = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$, and the volume of a cell is then $\Omega = \boldsymbol{a}\cdot\boldsymbol{b}\times\boldsymbol{c}$, the metric tensor is $g = h^{\mathsf{T}}h$. Let $\boldsymbol{r}_i = \xi_i\boldsymbol{a} + \eta_i\boldsymbol{b} + \zeta_i\boldsymbol{c} = h\boldsymbol{s}_i$, where $\boldsymbol{s}_i$ store its coordinates ξ_i, η_i, and ζ_i, i.e., reduced coordinates mentioned below. Then they introduced a Lagrangian

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

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

Then the equations of motion are

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

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

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

$$\Omega\Pi = \sum_{i=1}^{N} m_i \boldsymbol{v}_i \boldsymbol{v}_i^{\mathsf{T}} + \sum_{i=1}^{N}\sum_{i<j} \frac{\phi'(r_{ij})}{r_{ij}} (\boldsymbol{r}_i - \boldsymbol{r}_j)(\boldsymbol{r}_i - \boldsymbol{r}_j)^{\mathsf{T}}, \tag{21}$$

with $\boldsymbol{v}_i = h\boldsymbol{s}_i$. Then Andersen's equations of motion is a special case of (20), where $h = \mathrm{diag}(\Omega^{\frac{1}{3}}, \dots, \Omega^{\frac{1}{3}})$ and $g^{-1}\dot{g} = \frac{2\dot{\Omega}}{3\Omega}$. Though his equation for $\ddot{V}$ 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 $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, 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

$$\mathcal{L} = \sum_{i=1}^{N} \dot{\boldsymbol{q}}_i^{\mathsf{T}} d \dot{\boldsymbol{q}}_i - \sum_{i=1}^{N}\sum_{i<j} \phi(r_{ij}) + \frac{W}{2}\,\mathrm{Tr}\left(\dot{\epsilon}\dot{\epsilon}^{\mathsf{T}}\right) - P\Omega, \tag{22}$$

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

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

$$\ddot{\epsilon} = \frac{\Omega}{W}(\Pi - P)\left((1+\epsilon)^{\mathsf{T}}\right)^{-1}. \tag{24}$$

An equivalent form of (22) is

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{N} m_i \dot{\boldsymbol{s}}_i^{\mathsf{T}} g \dot{\boldsymbol{s}}_i - \sum_{i=1}^{N}\sum_{i<j} \phi(r_{ij}) + \frac{1}{2} W\,\mathrm{Tr}\left(\dot{h} f_0 \dot{h}^{\mathsf{T}}\right) - P\Omega, \tag{25}$$

with $f_0 = \sigma_0^{\mathsf{T}}\sigma_0$, where $\sigma_0 = \{\boldsymbol{a}_0\times\boldsymbol{b}_0, \boldsymbol{b}_0\times\boldsymbol{c}_0, \boldsymbol{c}_0\times\boldsymbol{a}_0\}$. 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

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

which makes it easier to implement in code.

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

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

with $f = \sigma^{\mathsf{T}}\sigma$, $e = \dot{h}^{\mathsf{T}}\dot{h}$. 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

$$\mathcal{L} = \sum_i \frac{1}{2}\mu \int d\boldsymbol{r}|\dot{\psi}_i(\boldsymbol{r})|^2 + \sum_I \frac{1}{2}M_I\dot{\boldsymbol{R}}_I^2 + \sum_\nu \frac{1}{2}\mu_\nu\dot{\alpha}_\nu^2 - E[\{\psi_i\},\{\boldsymbol{R}_I\},\{\alpha_\nu\}], \tag{28}$$

with the holonomic constraints

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

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

$$\mathcal{L}' = \sum_i \frac{1}{2}\mu \int d\boldsymbol{r}|\dot{\psi}_i(\boldsymbol{r})|^2 + \sum_I \frac{1}{2}M_I\dot{\boldsymbol{R}}_I^2 + \sum_\nu \frac{1}{2}\mu_\nu\dot{\alpha}_\nu^2 - E[\{\psi_i\},\{\boldsymbol{R}_I\},\{\alpha_\nu\}] + \sum_{ij}\Lambda_{ij}\left[\int d\boldsymbol{r}\psi_i^*(\boldsymbol{r},t)\psi_j(\boldsymbol{r},t) - \delta_{ij}\right], \tag{30}$$

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

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

$$M_I\ddot{\boldsymbol{R}}_I = \frac{\partial\mathcal{L}'}{\partial\boldsymbol{R}_I} = \boldsymbol{F}_I = -\frac{\partial E}{\partial\boldsymbol{R}_I}, \tag{32}$$

$$\mu_\nu\ddot{\alpha}_\nu = \frac{\partial\mathcal{L}'}{\partial\alpha_\nu} = -\frac{\partial E}{\partial\alpha_\nu}, \tag{33}$$

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

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

$$\boldsymbol{R}_I^{n+1} = 2\boldsymbol{R}_I^n - \boldsymbol{R}_I^{n-1} + \frac{h^2}{M_I}\boldsymbol{F}_I\,, \tag{34b}$$

$$\alpha_\nu^{n+1} = 2\alpha_\nu^n - \alpha_\nu^{n-1} - \frac{h^2}{\mu_\nu}\frac{\partial E}{\partial \alpha_\nu}\,, \tag{34c}$$

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

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

which shows that Λ is the transpose of usual Kohn–Sham hamiltonian matrix $\boldsymbol{H}$, i.e., $\Lambda_{ij} = \boldsymbol{H}_{ji}$. 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.

`mxdtyp` 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 `mxdtyp` $= 1$.

4.1 Main program

Read `nstep` from input file, determine how many steps are to be performed in the following MD loop.

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 $p\Omega$ 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 $\overline{E} = \frac{3}{2}k_B T$, where $\overline{E}$ is the average kinetic energy `avk`. Then do rescaling on Cartesian velocity `v` and reduced velocity `ratd`.

Table 1: List of some output variables of main program.

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

At the end of MD loop, write outputs to files. Some of the output variables are listed in Tab. 1. The atomic and lattice contributions could be also understood as ionic and cell contributions,

respectively. Another important output file is `avec`, which stores lengths and angles between primitive cell vectors for each MD step, i.e., $\{a, b, c\}$ and $\{\alpha, \beta, \gamma\}$, denoted by `bmodm` and `thetam`, respectively. `bmodm` at each MD step is set to lattice vectors moduli `avmod`, computed by `MOVE` subroutine. `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`.

First read the 3×3 matrix `avec` from `inp` file. If `ic` flag is `'s'`, since it is in reduced coordinates, we need to rewrite it in Cartesian coordinates. First store the sign of each entry, the do some transformation according to the corresponding `iop` matrix entry(What is this?). The rules are

$$h(i,j) = \operatorname{sgn}\big(h(i,j)\big) \times a_0 \times n(j) \begin{cases} \sqrt{h(i,j)}, & \texttt{iop(i,j)} \text{ is } \texttt{'s'}; \\ h(i,j)^{1/3}, & \texttt{iop(i,j)} \text{ is } \texttt{'c'}; \\ \frac{1}{3}h(i,j), & \texttt{iop(i,j)} \text{ is } \texttt{'c'}; \\ \frac{1}{\sqrt{3}}h(i,j), & \texttt{iop(i,j)} \text{ is } \texttt{'t'}, \end{cases} \tag{36}$$

where $n(j)$ denotes the number of primitive cells along jth direction. If `ic` is `'c'`, i.e., in an intermediate run, then read `avec`, `avecd`, `avec2di` from last run as input for subsequent runs, and write them to standard output.

Table 2: List of some variables in `CRSTL` subroutine.

variable name	variable symbol	variable
lattice vectors matrix	$h = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$	`avec`
atomic position in reduced coordinates	$\boldsymbol{s}_i = \{\xi_i, \eta_i, \zeta_i\}$	`rat`
atomic velocity in reduced coordinates	$\dot{\boldsymbol{s}}_i$	`ratd`
number of primitive cells	n	`nsc`
lattice parameter	a_0	`alatt`
fictitious mass	W	`cmass`
external pressure	P	`press`
reciprocal lattice vectors matrix	σ	`sigma`
metric tensor	$g = h^{\mathsf{T}} h$	`g`
metric tensor velocity	$\dot{g} = \dot{h}^{\mathsf{T}} h + h\dot{h}^{\mathsf{T}}$	`gd`
metric tensor inverse	g^{-1}	`gm1`

Then calculate cell volume Ω, metric tensor g and g^{-1}, which are treated as part of the outputs of this subroutine.

Then read positions `rat` for each atom of each type. If `ic` flag is `'s'`, then do similar transformation of these positions as (36) does; if it is `'c'`, i.e., it is an intermediate step, then read `avec`, `avecd`, `avec2di` from last run, and write to standard output.

4.2.2 `INIT` subroutine

First call `RANV` subroutine, generate some initial random positions for each atom of each type. `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`, Rahman–Parrinello approach is conducted, see 2.2 for detail. First, `gmgd`, i.e., $g^{-1}\dot{g}$ is calculated. Then for each atom of each type, `rat2d`, i.e., $\ddot{\boldsymbol{s}}_i = g^{-1}\dot{g}\dot{\boldsymbol{s}}_i$ is calculated. Also, `pim`, i.e., Π is calculated according to (21). Then $\ddot{h}$ is calculated according to (20b).

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 $\sigma_0 = \{\boldsymbol{a}_0 \times \boldsymbol{b}_0, \boldsymbol{b}_0 \times \boldsymbol{c}_0, \boldsymbol{c}_0 \times \boldsymbol{a}_0\} = \frac{V_0}{2\pi}\{\boldsymbol{c}_0^*, \boldsymbol{a}_0^*, \boldsymbol{b}_0^*\}$, we know $\ddot{d} = \frac{1}{V_0}\ddot{h}\sigma_0$. Then we do strain symmetrization, $\ddot{d}_{ij} = \frac{1}{2}(\ddot{d}_{ij} + \ddot{d}_{ji})$. Then $\ddot{h} = \ddot{d}h_0$.

If in `nd` or `nm`, we do $\mathrm{Tr}(\dot{h}^\top\sigma\sigma\dot{h})$.

If in `sd` or `sm`, we do $\mathrm{Tr}(\dot{h}^\top\sigma_0\sigma_0^\top\dot{h})$.

Then in both cases, we do `ekl` = `ekl` + $\mathrm{Tr}(\dot{h}^\top\dot{h})$.

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

Table 3: List of some variables in `INIT` subroutine.

variable name	variable symbol	variable
Π matrix	Π	`pim`
forces on basis vectors	$\ddot{h}$	`avec2d`
pressure	P	`press`

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 ith atom and jth atom, where $j \geq i$. So in the code we need to do some assignments like

$$\boldsymbol{U}_{ij} = \boldsymbol{U}_{ji}, \tag{37}$$

$$\boldsymbol{F}_{ij} = \boldsymbol{F}_{ji}, \tag{38}$$

where $\boldsymbol{U}_{ji}$ and $\boldsymbol{F}_{ji}$ 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

$$U(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right), \tag{39}$$

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

$$\boldsymbol{F}_{ji} = \frac{24\varepsilon}{\sigma}(\boldsymbol{r}_j - \boldsymbol{r}_i)\left(2\left(\frac{\sigma}{r}\right)^{14} - \left(\frac{\sigma}{r}\right)^{8}\right), \tag{40}$$

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

$$F(r) = \frac{24\varepsilon}{\sigma}\left(2\left(\frac{\sigma}{r}\right)^{13} - \left(\frac{\sigma}{r}\right)^{7}\right), \tag{41}$$

which is plotted in Fig. 1.

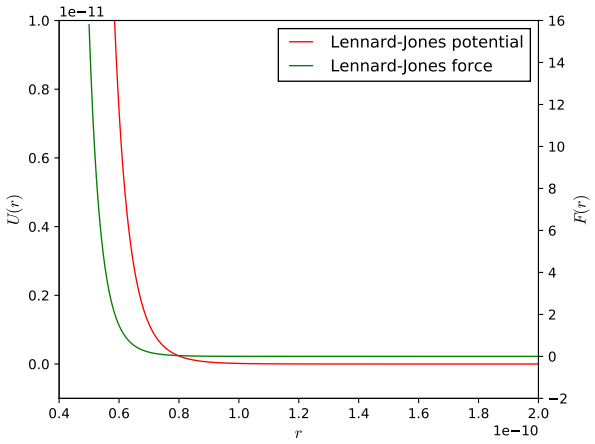


Figure 1: Lennard–Jones potential and force for Ar, as a function of distance between 2 atoms, with $\sigma = 3.405 \times 10^{-10}$m, $\varepsilon = 1.654 \times 10^{-21}$J.[?]

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.

Table 4: List of some variables in UPDG subroutine.

variable name	variable symbol	variable
lattice vectors matrix	h	avec
lattice velocity vector	$\dot{h}$	avecd
metric tensor	$g = h^{\mathsf{T}}h$	g
metric tensor velocity	$\dot{g} = \dot{h}^{\mathsf{T}}h + h\dot{h}^{\mathsf{T}}$	gd
metric tensor inverse	g^{-1}	gm1
	$g^{-1}\dot{g}$	gmgd

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

$$V = \sigma \cdot h, \tag{42}$$

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

$$f_0^{-1} = \frac{h_0^{\mathsf{T}}h_0}{V_0^2}, \tag{43}$$

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

$$f^{-1} = \frac{h^{\mathsf{T}}h}{V^2}, \tag{44}$$

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

$$f' = \frac{\partial f}{\partial h_{ij}} = (\sigma'_{ij})^{\mathsf{T}}\sigma + \sigma^{\mathsf{T}}\sigma'_{ij}, \tag{45}$$

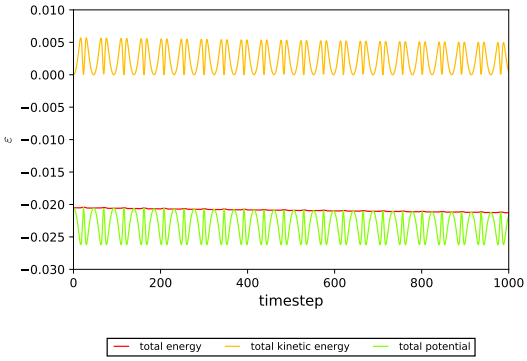
where σ' is denoted as `sigmap` in code, another 3×3×3×3 tensor. $\dot{f}_0 = \dot{\sigma}^{\mathsf{T}}\sigma + \sigma^{\mathsf{T}}\dot{\sigma}$ is also calculated. $f^{-1} = \sum_{k,l} e_{lk} f'_{ijkl}$, and $\sigma^{-1} = h\dot{f}$. Then final returns $\ddot{h}$. As stated above, $h = (1+\epsilon)h_0$,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

4.4 Input file

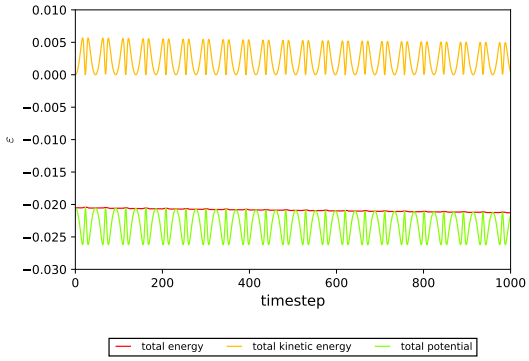
Now let's have a look at the input file.

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.

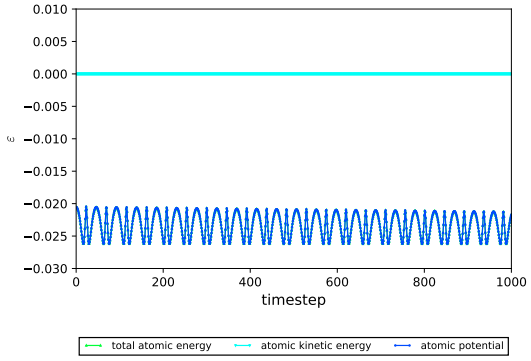
Figure 2: Input 1



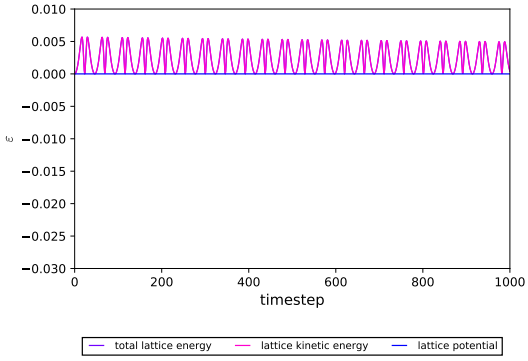
(a)



(b)



(c)



(d)

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.