The User Manual

PWmat

VERSION: BETA

Long Xun Kuang Teng Inc. Beijing, China.

May 10, 2024

Contents

Chapter 1

Introduction

PWmat is a plane wave pseudopotential package for density functional theory (DFT) calculations. It is designed to run efficiently on CPU/GPU processors. The best explanation of the algorithms used in PWmat can be found in Ref.[1, 2]. PWmat can perform the following calculations (indicated by the **JOB** tag in input file **etot.input**):

1. SCF (self-consistent-field calculation);
2. NONSCF (non-self-consistent-field calculations, e.g. for bandstructure calculation, which is usually done after a SCF calculation);
3. DOS (density of state calculation, which is usually done after a NONSCF or SCF calculation, it is used to do partial density of state, or k-point interpolation);
4. RELAX (atomic relaxation calculation and cell relaxation);
5. EGGFIT (a preprocess fitting procedure to remove the egghead problem in RELAX);
6. MD (ab initio molecular dynamics calculation);
7. TDDFT (real-time time dependent density functional theory calculations);
8. NAMD (non-adiabatic molecular dynamics, which is used to study carrier dynamics following a Born-Oppenheimer molecular dynamics);

9. NEB (nudged elastic band calculation for barrier heights);
10. DIMER (dimer method calculation for finding saddle points);
11. POTENTIAL (generate DFT potential from input charge density);
12. SCFEP (electron-phonon coupling constant calculation for a given pair of input electron states for all the phonon modes);
13. WKM (a special Wannier Koopmann's method calculation for DFT band gap correction);
14. ATOMIC_ORB(atom's atomic wavefunction calculation);
15. TRANS(quantum transport device calculation).

In order to run PWmat, one needs to provide the following necessary input files in the running directory: parameter file (must be named as **etot.input**); structure file (usually is **atom.config** in our examples and tutorials); pseudopotential files.

An example of workflow from the beginning to the end is:

1. Get crystal structure from online database, or build it from visualization packages (e.g.Q-Studio).
2. Download pseudopotential files from PWmat website, then generate **etot.input** and convert structure file to PWmat format by using PWkit.
3. Run command "mpirun -np 4 PWmat" to excute PWmat directly on a single computational node (Mstation). Or run "qsub job.pbs" if one is using TORQUE PBS on HPC clusters, "sbatch job.pbs" if one is using SLURM, to submit job on HPC clusters (Mcloud, Mcluster).
4. Collect the results by using utilities. Prepare for the next calculation if needed.

All the above programs are pre-installed on the server. In the current PWmat releasing package, we also include one directory: 'examples', which contains example cases for carrying out different jobs.

Besides to run PWmat alone, one may check our modules. They are recipes to carry out actual calculations for different scientific tasks. For example, calculating the defect level energies, the catalytic process in electrochemistry. These are designed as tutorials to help users to finish the actual tasks. It can also be packages for more sophisticated tasks, e.g., PyPWmat for phonon spectrum calculation, or using YAMBO for GW calculation. We will develop more modules in the future, we also welcome our users to provide their own modules.

(Note):*Due to rapid development, there could be some minor changes for the tutorial, but the basic steps and ideas are the same, and the changes should be obvious.*

Chapter 2

Input files

PWmat needs a few basic input files to start the calculation: parameter file (must be named as **etot.input**); structure file (usually is **atom.config** in our examples and tutorials); pseudopotential files.

In some cases, one might also need to provide some optional input files, such as charge density (IN.RHO), high-symmetry-kpoints (IN.KPT), detailed solvent parameters (IN.SOLVENT). Some of them are simple so they can be written by hand. Some of them are binary files, which will be generated from the previous calculations, then one can copy them to input file for the next calculation. For example, one should copy **OUT.VR** to **IN.VR** for non-self-consistent calculation.

In the following, we will explain the long version of these files respectively.

2.1 Parameter file (etot.input)

The parameter file must be named as **etot.input**. It is the most important input file, used to control how PWmat runs. Here is an example of the simplest:

```
4 1
IN.ATOM = atom.config
IN.PSP1 = Si.NCPP.UPF
JOB = SCF
```

The first line must be two positive integers, which correspond to the tags **NODE1**, **NODE2** respectively.

The following lines in etot.input specify the name of the structure file, the name of the pseudopotential file and type of calculation. You need to specify at least these parameters because they have no default values.

Except for the first line, the content has a format of "**TAG** = **VALUE**". The orders of different tags can be arbitrarily changed. The names of the tags are case insensitive. One can add annotations after # in line.

After running PWmat, one can also check the header of the output file **REPORT** and copy them as etot.input.

In the following, we will explain the meaning of each tag. The tags in red are the mandatory tags, the tags in green are job-dependent mandatory tags, and all the others are optional.

2.1.1 Control tags

2.1.1.1 NODE1

Format: the first integer in the first line

Default: none

Related tags: NODE2

The number of processors used to divide the G-space sphere and **N1 * N2 * N3** FFT grid. N1 * N2 * N3 must be divisible by **NODE1**.

The product of **NODE1** and **NODE2** is equal to the processors used by PWmat.

(Note): *If one run a same task with different* **NODE1**, *the automatically generated* **N123** *might be different, then the results might be slightly different.*

2.1.1.2 NODE2

Format: the second integer in the first line

Default: None

Related tags: NODE1

The number of processor groups to divide the k-points. One might check whether the number of k-points is divisible by **NODE2** for high efficiency. The larger **NODE2**, the larger the required memory.

The product of **NODE1** and **NODE2** is equal to the processors used by PWmat.

(WARNING): *Hybrid functional calculation, k-point interpolation and electron-phonon coupling calculation do not support k-points parallelization. One must set* **NODE2** = *1 in these cases.*

2.1.1.3 JOB

Format: JOB = [string]

Default: None

Controls what PWmat will do. **JOB** can be **SCF**, **NONSCF**, **DOS**, **MOMENT**, **RELAX**, **EGGFIT**, **MD**, **TDDFT**, **NAMD**, **NEB**, **DIMER**, **SCFEP**, **POTENTIAL**, **HPSI**, **WKM**, **ATOMIC_ORB** and **TRANS**.

JOB = SCF Do self-consistent field iterations.

Related tags: E_ERROR, RHO_ERROR, WG_ERROR, FERMIDE, SCF_ITER0_*, CHARGE_DECOMP, ...

SCF determines the charge density, the total energy. During SCF calculation, the atoms will not be moved.

Frequeuntly used etot.input settings for SCF calculation:

```
4 1
IN.ATOM = atom.config
JOB = SCF
IN.PSP1 = Si.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
MP_N123 = 9 9 9 0 0 0
```

JOB = NONSCF Do non-self-consistent calculations.

Related tags: IN.KPT, IN.VR, IN.NONSCF, ...

NONSCF calculation requires an input potential, usually from a previous SCF calculation. One must set **IN.VR = T** in **etot.input**.

NONSCF calculates the eigen wave functions non-self-consistently, but do not calculate total energy. A denser K-mesh can be set, but for the bandstructure calculation, one can generate an explicitly K-path file and convert it by using **split_kp.x**.

Frequeuntly used etot.input settings for NONSCF calculation:

```
4 1
IN.ATOM = atom.config
JOB = NONSCF
IN.PSP1 = Si.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
IN.KPT = T
IN.VR = T
```

In addition, some specific parameters can be set. Please refer to section 2.4.5 for details.

WARNING: *when using hybrid functional, one must copy* ***OUT.HSEWR(i)*** *files from previous SCF calculation to the current NONSCF directory.*

JOB = DOS Do density of state calculation.

Related tags: DOS_DETAIL, IN.WG, ...

DOS calculation requires input wave function and eigen energy, usually from a previous SCF or NONSCF calculations. One must set **IN.WG = T** in **etot.input**. one should also copy or link **OUT.EIGEN** file from previous calculation to the current DOS directory.

DOS uses input wave functions to calculate their projections on atomic orbitals, the nonlocal potential projector in this step is different from NONSCF and SCF. One can get partial and projected DOS by using **plot_DOS_interp.x**.

There are two ways to calculate DOS, one is conventional DOS calculation, the other is k-point interpolation scheme for DOS calculation, which can get a smooth DOS with very few k-points.

Frequently used etot.input settings for conventional DOS calculation:

```
4 1
IN.ATOM = atom.config
JOB = DOS
IN.PSP1 = Si.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
MP_N123 = 9 9 9 0 0 0 #keep it consistent with previous calculation
IN.WG = T
```

For k-point interpolation scheme method, it is controled by tag **DOS_DETAIL**.

Frequently used etot.input settings for k-point interpolation scheme DOS calculation:

```
4 1
IN.ATOM = atom.config
JOB = DOS
IN.PSP1 = Si.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
MP_N123 = 9 9 9 0 0 0 #keep it consistent with previous calculation
DOS_DETAIL = 1 9 9 9
IN.WG = T
```

WARNING: *one must copy* ***OUT.EIGEN*** *file from previous calculation to the current DOS directory.*

JOB = MOMENT Calculates the momentum matrix (oscillator strength) between Kohn-Sham orbitals, and the nonlocal potential is considered.

Related tags: IN.WG ...

MOMENT calculation requires input wave function, usually from a previous SCF or NONSCF calculations. One must set **IN.WG = T** in **etot.input**.

To calculate the optical absorption spectrum, or dielectric constant, the momentum matrix between Kohn-Sham orbitals $\{\psi_i\}$ needs to be calculated. Formally, this momentum matrix can be expressed as: $M_x(i,j) =< \psi_i|P_x|\psi_j >= - < \psi_i|\partial H[k]/\partial k_x|\psi_j >= -i < \psi_i|[H, r_x]|\psi_j >$. here subscript x actually stands for x, y, z directions. So, there are three matrix (in Cartesian coordinates). P_x is the momentum operator. In the case there is no nonlocal potential, $P_x = i\nabla_x$. If only the $i\nabla_x$ is needed in the calculation, one can use utility 'ug_moment.x' to calculate the matrix based on the output wave function OUT.WG.

However, if the nonlocal potential needs to be taken into account, there is an additional term $i(V_{NL}r_x - r_xV_{NL})$, which cannot be calculated easily. The JOB=MOMENT is to solve this problem, to include this additional term. The resulting momentum matrix is output in OUT.MOMENT_EXT_KPT. For example, it can be used for RPA calculation for absorption spectrum or dielectric constant calculations. Including this nonlocal term can increase the oscillator strength $|M_x|^2$ by about 10%. The calculated momentum matrix in JOB = MOMENT will be stored in output file 'OUT.momentK.(x).1'.

Frequently used etot.input settings for MOMENT calculation:

```
4 1
IN.ATOM = atom.config
JOB = MOMENT
IN.PSP1 = Si.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
MP_N123 = 9 9 9 0 0 0 #keep it consistent with previous calculation
IN.WG = T
```

JOB = RELAX Do atomic position relaxations and cell relaxation using DFT force and total energy.

Related tags: RELAX_DETAIL, RELAX_HSE ...

Inside the atom.config, in the POSITION section, the last three columns 1,1,1, determine whether this atom will move in the x,y,z directions (see section 2.2): 1,1,1, means move, 0,0,0 means fix. Similarly, if cell relaxation is specified in RELAX_DETAIL, then a STRESS_MASK section in atom.config can be used to specified whether one wants to relax all components of the unit cell vector, or only selective components of the cell. Also, if cell relaxation is specified, during the relaxation, the number of plane wave G-vectors are kept unchanged. As a result, after the relaxation, if one wants to redo a calculation with the same Ecut, then due to the change of the cell, the number of G-vector will be different, and the energy, stress etc might be different from the previous relaxation runs. So, one might want to do a relaxation again. Or, one can use a larger Ecut to do the cell relaxation. Another option is to use STRESS_CORR (see section 2.1.6.5) to make a correction for stress calculation, taking into account the effect of finite Ecut to the calculation of stress.

Each atomic relaxation step will do one SCF calculation. Optionally you can also have "RELAX_DETAIL" (for general RELAX) and "RELAX_HSE" (for RELAX in the case of HSE calculation) in the etot.input. See below2.1.5.4. A concise result will be reported in RELAXSTEPS. The atomic movements for each relaxation step will be reported in MOVEMENT, and final atomic configuration is reported in final.config.

Frequently used etot.input settings for atomic relaxation calculation:

```
4 1
IN.ATOM = atom.config
JOB = RELAX
RELAX_DETAIL = 1 500 0.01
IN.PSP1 = XXX.SG15.PBE.UPF #modify XXX according to your structure
XCFUNCTIONAL = PBE
Ecut = 50
Ecut2 = 200
MP_N123 = NK1 NK2 NK3 0 0 0 #modify "NK1 NK2 NK3" according to structure
lattice
```

Frequently used etot.input settings for cell relaxation calculation:

```
4 1
IN.ATOM = atom.config
JOB = RELAX
RELAX_DETAIL = 1 500 0.01 1 0.01
IN.PSP1 = XXX.SG15.PBE.UPF #modify XXX according to your structure
XCFUNCTIONAL = PBE
Ecut = 70
Ecut2 = 280
MP_N123 = NK1 NK2 NK3 0 0 0 #modify "NK1 NK2 NK3" according to structure
lattice
```

In addition, some specific RELAX parameters can be set. Please refer to section 2.4.6 for details.

JOB = EGGFIT This is a way to fix the "egghead" problem in atomic relaxation.

Related tags: EGG_DETAIL, EGG_CORR

The egghead problem is caused by the numerical discretization of the real space using grid n1*n2*n3. As a result, the atom can have an artificial force towards or away from the grid point. In many cases this problem can cause the system relaxing very slowly when the force is small, or the relaxation energy curve become not smooth. In most cases, this problem can be removed by using Ecut2=4Ecut, Ecut2L=Ecut2 (for norm conserving pseudopotential, NC-PSP). However, sometime even this cannot remove the "egghead" problem. In that case, the Ecut2=4Ecut, Ecut2L=4Ecut2 will almost always remove the egghead problem. But the "Ecut2=4Ecut,Ecut2L=4Ecut2" could be rather expensive. To keep the calculation in "Ecut2=4Ecut, Ecut2L=Ecut2" (for NC-PSP), we provide a JOB=EGGFIT procedure to remove the egghead problem. This can be useful for large system relaxation runs. In order to use this procedure, one needs to do the relaxation in two steps:

1. set the JOB = EGGFIT in the etot.input and give an additional setting: egg_detail = np1, np2, np3; ECUT2 = 4ECUT, ECUT2L = ECUT2; Here, np1,

np2, np3 indicates the point to probe inside a grid, usually they are 2,2,2 or 4,4,4. After running PWmat, it will give a new file "CC.egghead" which will be used in the following step.

Frequently used etot.input settings for eggfit calculation:

```
4 1
IN.ATOM = atom.config
JOB = EGGFIT
EGG_DETAIL = 2 2 2
IN.PSP1 = XXX.SG15.PBE.UPF #modify XXX according to your structure
XCFUNCTIONAL = PBE
Ecut = 50
Ecut2 = 200
MP_N123 = NK1 NK2 NK3 0 0 0 #modify "NK1 NK2 NK3" according to structure lattice
```

2. set the JOB = RELAX with an additional setting: EGG_CORR = T, ECUT2 = 4ECUT, ECUT2L = ECUT2. EGG_CORR = T means PWmat will read "CC.egghead" to do the egghead correction during relaxation.

   Frequently used etot.input settings for atomic relaxation with egg_corr:

```
4 1
IN.ATOM = atom.config
JOB = RELAX
RELAX_DETAIL = 1 500 0.01
IN.PSP1 = XXX.SG15.PBE.UPF #modify XXX according to your structure
XCFUNCTIONAL = PBE
Ecut = 50
Ecut2 = 200
EGG_CORR = T #read "CC.egghead" file from previous "JOB=EGGFIT"
MP_N123 = NK1 NK2 NK3 0 0 0 #modify "NK1 NK2 NK3" according to structure lattice
```

JOB = MD Do Born-Oppenheimer molecular dynamics (MD) simulations.

Related tags: MD_DETAILS, IN.MDOPT ..

Must have variable "MD_DETAIL" in etot.input (see section 2.1.5.7). The PWmat can perform: Verlet, Nose-Hoover, Langevin, Berendsen dynsmics. We have a concise output as reported in MDSTEPS. The atomic movements for every step are reported in MOVEMENT. One can also do special MD, e.g., with applied different force on each atom, or different specified temperature on each atom within the Langevin dynamics. Inside the atom.config, in the POSITION section, the last three column 1,1,1, determines whether this atom will move in the x,y,z directions (see section 2.2): 1,1,1, means move, 0,0,0 means fix. If dynamics can change the unit cell vector (e.g., in NPT calculation), one can also use STRESS_MASK section in atom.config to specify which cell vector component to change.

An example etot.input for MD calculation:

```
4 1
IN.ATOM = atom.config
JOB = MD
MD_DETAIL = 1 1000 1 300 300
IN.PSP1 = Si.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
MP_N123 = 1 1 1 0 0 0 2
```

In addition, some specific MD parameters can be set. Please refer to section 2.4.7 for details.

JOB = TDDFT Do real-time time-dependent DFT calculation (rt-TDDFT).

Related tags: MD_DETAIL, TDDFT_DETAIL, TDDFT_TIME, TDDFT_SPACE, IN.A_FIELD, TDDFT_BOLTZMANN, IN.TDDFTOPT ..

This is a major functionality in PWmat. It uses a new algorithm as reported in Ref.[3]. The detail of this JOB is described separately in appendix B. The rt-TDDFT can be used to simulate the dynamic process where both nuclei and electron movements are important, and the electron is no longer in the ground state during the nuclear movement (for example, in a high speed ion collision with a material). It can also be used to study optics (absorption spectrum, or nonlinear optics). It includes both the electron-electron interaction, and electron-phonon interaction. The TDDFT simulation is more expensive than the NAMD calculation. Mostly this is because one needs to use a smaller time step dt (e.g., 0.1 fs), and calculate more electron adiabatic states (to expand the time evolving wave functions).

JOB = NAMD Do non-adiabatic molecular dynamics.
Related tags: MD_DETAIL, NAMD_DETAIL, TDDFT_TIME, TDDFT_SPACE, TDDFT_STIME, IN.A_FIELD, IN.MDOPT ..

This is done under the approximation of Born-Oppenbeimer MD (BO-MD) for nuclear movement. So the time to do NAMD is almost the same as that for MD. The actual NAMD simulation is done as a post-processing after the DFT BO-MD. It will generate a file OUT.NAMD. Some post-processing program (e.g., the "namd_dm.x" in Boltzman-NAMD) can be used to study the single carrier dynamics during the BO-MD process. It only simulates the behavior of a single carrier. While it takes into account the effects from other electron and phonon to the dynamics of this single carrier (hence, include the electron-phonon coupling etc), it ignores the effects of this carrier to the dynamics of other electron and phonon (i.e, there is no feedback from carrier to phonon, or carrier to other electron, thus it cannot be used to study polaron effect). Advantageously, it also does not have the erroneous carrier self-interaction. It is suitable to study the carrier dynamics (e.g., charge transfer between molecule, or spin dynamics of one defect) of some large systems. Compare to TDDFT, one advantage is that it can do much bigger system with much longer time. The details are also described in

appendix B.

JOB = NEB Calculate the energy barriers using nudged elastic band method.

Related tags: NEB_DETAIL ..

It must have a variable "NEB_DETAIL" in etot.input file. Besides IN.ATOM, which gives the first valley site atomic position, there must be a second valley site position given in the NEB_DETAIL line. One must precalculate (e.g., using JOB=RELAX) the atomic configuration of these two valley sites before using JOB=NEB to calculate their barrier. See NEB_DETAIL (see section 2.1.5.14) for more details and how to set up the calculations. Output files: RELAXSTEPS, NEB.BARRIER, MOVEMENT. NEB.BARRIER gives the barrier height information, while MOVEMENT gives all the image atom.config files within each NEB step. **(WARNING)**: During NEB calculation, if you encounter the following error: "equivalent atom not found under symm op, stop", please turn off the symmetry, just set "MP_N123 = NK1 NK2 NK3 0 0 0 2" (see section 2.1.2.9)

An example etot.input for NEB calculation:

```
4 1
IN.ATOM = atom1.config
JOB = NEB
NEB_DETAIL = 5 100 0.03 5 1 2 -7946.015 -7946.015 1 atom2.config
ACCURACY = High
IN.PSP1 = C.SG15.PBE.UPF
IN.PSP2 = Li.SG15.PBE.UPF
XCFUNCTIONAL = PBE
Ecut = 50
MP_N123 = 1 1 1 0 0 0 2
```

Additional relaxation settings can be found in section 2.4.6.

JOB = DIMER This is a calculation of dimer method [33, 34].

Related tags: IN.RELAXOPT ..

Dimer method is used for finding saddle points without knowledge of the final state of the transition is described, and allows users to search for a nearby saddle point from a given initial configuration. The dimer method is designed to deal with problems with unkown reaction mechanisms.

Some specific DIMMER parameters can be set in file IN.RELAXOPT, please refer to section 2.4.6 for details.

The initial direction along the dimer can be set in structure file by tag DIMER_DIR_N (see section 2.2).

The final configuration is writen in file final.config, the configurations of each translation step are in file MOVEMENT.

Another file DIMERSTEPS can be used to check the convergency, see section 3.1.7.

An example etot.input for DIMMER calculation:

```
1 4
job = dimer
in.atom = atom.config
in.psp1 = H.SG15.PBE.UPF
in.psp2 = N.SG15.PBE.UPF
Ecut=50
Ecut2=200
e_error=0.0
mp_n123=2 2 2 0 0 0 2
     ! in somecases you should turn off the symmetry
fermidE=0.2
```

JOB = SCFEP This will carry out an electron-phonon coupling calculation.

Related tags: SCFEP_DETAIL

The procedure is the following, one first carries out a JOB = SCF calculation, and OUT.FORCE = T, so there will be atomic forces (or perhaps before that, there will be a JOB = RELAX, to relax the atoms). Copy OUT.FORCE file

into IN.FORCE (and set IN.FORCE = T). For JOB = SCFEP, we must have IN.WG = T, here IN.WG is copied from the OUT.WG from the previous JOB = SCF calculation. Now, in JOB = SCFEP calculation, the state $\psi(ist1, ikpt, ispin)$, and $\psi(ist2, kpt, ispin)$ (to be specied in the line of SCFEP_DETAIL) will be used to calculate: $< \psi(ist1, ikpt, ispin)|\delta H/\delta R|\psi(ist2, ikpt, ispin) >$, this result will be represented as an perturbed atomic forces (the perturbation is proportional to α as specified in SCFEP_DETAIL), and reported in OUT.FORCE. The actual coupling constants will be reported in OUT.EP_COEFF (here the FORCE_new has already be subtracted by IN.FORCE, and divided by α).

The α should be small, something like 0.1, 0.2.

The electron-phonon coupling constant reported in OUT.EP_COEFF, together with phonon calculations can be used to study non-adiabatic decay and charge trapping by defect states.

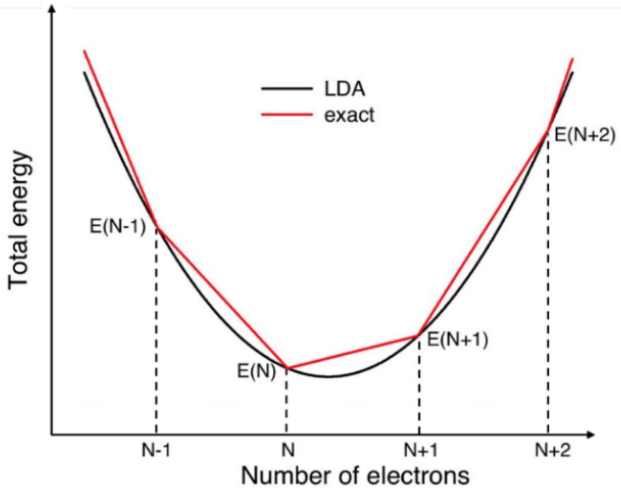
JOB = POTENTIAL This take the input charge, IN.RHO=T, output the potential: out.vr, out.vr_hion, then stop. It will be useful for charge patching and defect calculation. Basically it is a simple Poisson solver. In particular, for isolated systems, one can specify COULOMB=1 for Poisson solver without periodic image potential. This can be a very quick calculation.

JOB = HPSI This is used to calculate $hpsi_i = H\psi_i$ and output the wave function $hpsi_i$ in OUT.HPSI (and OUT.HPSI_2 for spin=2). It must has a $IN.WG = T$, and have $IN.VR = T$ or $IN.RHO = T$ to have the proper Hamiltonian. This is provided, so one can carry out some analysis, for example to calculate the electron-phonon coupling.

JOB = WKM A special calculation for Wannier Koopmann method (WKM)[28, 29].

When we do a DFT calculations, one difficulty is to calculate a right bandstructure with right band gaps which can agree well with experimental results. However, it is a common sense that LDA, PBE and even HSE (when α is set to be 0.25) ofter nderestimate band gap results. For band gap calculation, we often define the band gap

as the difference between the electron affinity (EA) energy and the ionization energy (IE). Here, EA can be expressed as E(N+1)-E(N) and IE as E(N)-E(N-1). N is the number of electrons in the neutral system, and N+1 and N-1 indicate the system has one more or one less electron respectively. E(N) is the self-consistent energy of the system with N electrons.



total energy profile describes an LDA total energy calculation and an "exact" energy result for an open system

One possible way to overcome this underestimation is to perform a Koopmans condition on normal DFT calculations. If we plot E(N) as a function of N in LDA calculations, a black parabola in the above figure can be plotted. However, this seems to be unphysical. If we want to calculate E(N+s), which $0 < s < 1$, the total energy of a system with a fractional number of electrons can be defined as a statistical mixture of the N electron and N+/-1 electron state.This leads to a linear segment total energy function of s, which is plotted in red "exact" straight lines in the above figure.This linear segment property is also called Koopmans condition.

However, if we just add s electrons in a unit cell, as a result, accordingto Janak's theory, the total energy difference is the same asKohn-Sham orbital eigen energy. To overcome this problem, we added an electron into a localized Wannier function instead of the extended Kohn-Sham orbitals. As a result, the WKM total energy can beexpressedas:

$$E_{WKM}(\{s_k\}) = E_{LDA}(\{s_k\}) + \sum_k E_k(s_k) \tag{2.1}$$

Here, w indicates the wannier functions. s_k ($0 < s_k < 1$) indicates the occupation number of this Wannier function. During the LDA calculation, we add or remove the electron from ϕ_k of one spin channel and all the other orbitals in this spin channel should be orthogonal to this Wannier function ϕ_k. All the other orbitals (except this oneWannier function) are variationally changed to minimize the total energy, which results in the ground state energy $E_{LDA}(s_k)$. Thus, a simple analytical expression of $E_{LDA}(s_k)$ can be writen as

$$E_k(s_k) = \lambda_k s_k(1 - s_k) \tag{2.2}$$

The λ_k can be determined from $E_{LDA}(s_k)$ (to make $E_{WKM}(s_k)$ a straight line vs. s_k). λ_k can be calculate by PWmat JOB = WKM mode.

It requires the input of Wannier wave functions (in real space), provided by files in the name of IN.WANNIER_00001.u, IN.WANNIER_00002.u IN.WANNIER_00001.d IN.WANNIER_00002.d etc. They are written in the same format as the charge density IN.RHO (thus the Wannier wave functions are real). Each of this file contain only one Wannier function ϕ_k. The information for these Wannier functions are provided in a file called: IN.S_WKM. They look like

```
S_WKM1
2            : the number of Wannier function in up spin
0.5   1.0    : s1_u ss1_u : the occupation of the first up wannier function
0.0   0.0    : s2_u ss2_u : the occupation of the second up wannier function
S_WKM2
1            : the number of Wannier function in down spin
0.9   1.9    : s1_d, ss1_d : the occupation of the first down wannier function
M_FIX_WKM    : This section is optional
nb_fix1,nb_fix2,iflag_wkm_Hxc,nb_exclude_Hxc
```

In the WKM calculation, the Wannier function ϕ_k will be occupied according to s1_u, or s2_u. The occupation ss1_u, ss2_u is used to make the system a full shell. This is only used when there is a special treatment for the exchange-correlatin functional by exclude some core level charge densities. Otherwise, they are not really used. In the WKM calculation, the other "normal" wave functions $\{\psi_i\}$ will be orthogonal to the Wannier functions ϕ_k included in the IN.S_WKM file. Their total charge (from $\{\psi_i\}$) is determined by $NUM\_ELECTRON$. So, the total charge is $NUM\_ELECTRON$ plus the s1_u etc. It is a good idea to always include $NUM\_ELECTRON$ in the WKM calculation.

The optional session M_FIX_WKM is used for a special exchange-correlation functional treatmenf for the WKM calculation with semicore states (the states very deep in energy). We found that, in WKM calculation for the lambda for a Wannier function ϕ_k, it might be necessary to fix some deep level bands (so they do not change during the SCF calculation. These bands are indicated by $[nb\_fix1.nb\_fix2]$. If $iflag\_wkm\_Hxc = 1$, then the bands: $[1, nb\_exclude\_Hxc]$ counted from the bottom will not be included in the exchange-correlation function evaluations, and in this case the $ss1\_u$ and $ss2\_u$ are used to occupy the Wannier function so to get a closed shell structure.

Note, one can also use JOB=WKM to do some other calculations. For example, to fix some wave function without change during SCF, but to relax all the other wave functions, while keeping all of them orthogonal at the same time. It is not straightforward to do due to the real space form in IN.WANNIER_00001.u etc, but it can be done.

Note, there is another related calculation, that is the SCF WKM calculation. For that calculation, it is done not by JOB=WKM, instead it is done by using JOB=SCF, and XCFUNCTIONAL = LDAWKM, XCFUNCTIONAL = LDAWKM2, or XCFUNCTIONAL = LDAWKM3. Those are used to carry out SCF WKM calculation when the WKM parameter λ_k has already been calculated, or they can be used to carry out linear response WKM calculations. Please check the XCFUNCTIONAL section for that. Detailed steps to perform a WKM calculation in PWmat code, please refer to

"Band structure calculation by WKM".

JOB = ATOMIC_ORB This will calculate the chosen atom's atomic wavefunction specified in its pseudopotential file.

Parameter ATOMIC_ORBITAL_IATOM_OUT in etot.input must be set to set the atom index range(the index is from the IN.ATOM configuration file).

The output filenames of atomic wavefunctions in format atomic_orb_iatom_chi_ichi_l_il_m_im, iatom is the index of chosen atom, ichi is the index of PP_CHI in pseudopotential file, il and im are the corresponding quantum numbers.

The output atomic wavefuncions are in the same format of OUT.RHO, one can use convert_rho_new.x to convert to xsf format. All datas of atomic wavefunctions are real numbers.

JOB = TRANS Calculate system state $\psi_l(\mathrm{r})$ of transport device based on auxiliary function $W_l(\mathrm{r})$. Please refer to pwmat_transport for more details.

An example file etot.input:

```
4 1
job = trans
in.atom = system.config
in.vr=T
SCF_ITER0_1 = 1 100000 3 0.0 0.2 1 # must have this line, will only do one iteration with many CG steps, so NITER0_1=1; NLINE0=100000
num_band=35 # number of W_l(r)
N123=480 96 32
Ecut=50
Ecut2=100
precision=double
wg_error=1.d-5
flag_cylinder=1 # will not use Ecut value for cutoff energy in direction x
in.kpt=T
IN.PSP1 = Cu.FHI.LDA.UPF
```

```
IN.PSP2 = S.FHI.LDA.UPF
IN.PSP3 = C.FHI.LDA.UPF
IN.PSP4 = H.FHI.LDA.UPF
```

2.1.1.4 ACCURACY

ACCURACY = NORM / HIGH / VERYHIGH.

Default:

ACCURACY = HIGH (for JOB = RELAX, NEB, DIMER)

ACCURACY = NORM (for other JOB)

We have introduced three control flags: Accuracy, precision and convergence for the user to easily control different aspects of the calculation. The values of these flags will change the settings of other more detailed parameters. However, one can also set those parameters directly. Those detailed parameters should have higher priority (if they are explicitly set) than these three control flags.

Control the calculation accuracy, helping to set up the default values for other parameters in etot.input. This parameter will influence the setting of default ECUT/ECUT2 and P123 (for HSE) (see the following).

ACCURACY	NORM	HIGH	VERYHIGH
ECUT	PSP/INPUT	PSP/INPUT	PSP/INPUT
ECUT2	2*ECUT	4*ECUT	4*ECUT
ECUT2L	ECUT2(NCPP)	ECUT2	4*ECUT2
	4*ECUT2(USPP)	4*ECUT2	4*ECUT2
ECUTP	ECUT	4*ECUT	4*ECUT
RCUT	PSP/INPUT	PSP/INPUT	1.1*PSP/INPUT

ACCURACY = NORM, the default ECUT will be used, and ECUT2 = 2 * ECUT, ECUT2L = ECUT2 for NCPP, and ECUT2L = 4 * ECUT2 for ultrasoft PSP. P123 = NP1, NP2, NP3, which equals 2/3 of N1, N2, N3 (or generated from ECUTP, using a FFT box just containing the ECUTP sphere).

ACCURACY = HIGH, if ECUT/ECUT2 are not specified, it will set ECUT = 1.0 * default value in the pseudopotential file and ECUT2 = 4 * ECUT, ECUT2L = ECUT2, and P123 = N123.

ACCURACY = VERYHIGH, if ECUT/ECUT2 are not specified, it will set ECUT = 1.0 * default value in the pseudopotential file and ECUT2 = 4 * ECUT, ECUT2L = 4 * ECUT2, and P123 = N123.

2.1.1.5 PRECISION

PRECISION = AUTO / SINGLE / DOUBLE / MIX.

Default:

PRECISION = AUTO

The precision controlling flag of GPU calculation.

PRECISION	AUTO(DEFAULT)	DOUBLE	SINGLE	MIX
SCF(HSE)	NCPP:DOUBLE	NCPP:DOUBLE	NCPP:DOUBLE	NCPP:DOUBLE
	USPP:SINGLE	USPP:SINGLE	USPP:SINGLE	USPP:SINGLE
RELAX_HSE(NUM_LDA>0)	LDA:SINGLE	LDA:DOUBLE	LDA:SINGLE	LDA:MIX
	HSE:DOUBLE	HSE:DOUBLE	HSE:DOUBLE	HSE:DOUBLE
SCF,RELAX(LDA/GGA)	SINGLE	DOUBLE	SINGLE	MIX

PRECISIION = AUTO, double or single precision in the calculation will be automatically adjusted.

PRECISION = SINGLE, use single precision of GPU calculation, default (except for HSE). For most cases, SINGLE is good enough. Typically, it can converge the total energy to 0.1 meV, and the error for rho to be about 1.E-5, and error for total energy to be about 1.E-4 (eV).

PRECISION = DOUBLE, use double precision of GPU calculation. Default for the PBE part of HSE calculation. This however can be slower on the Mstation. Use this only you really want to make sure the numerical precision is not a problem.

PRECISION = MIX, use both double and single precisions in the calculation, automatically adjust. Only some critical parts use DOUBLE, the other parts use

SINGLE. It is a compromise between SINGLE and DOUBLE precisions. Usually this should be good enough for almost any calculations.

Obviously, from **SINGLE**, **MIX** to **DOUBLE**, more accurate, but more costly. In most calculations, SINGLE is good enough, and MIX can be almost as good as the DOUBLE precision. If there are some issues in terms of convergence and the final result, one can use DOUBLE precision to check. Note, for HSE, the HSE Fock exchange term is calculated with SINGLE, but the PBE iterations are done using DOUBLE.

2.1.1.6 CONVERGENCE

CONVERGENCE = EASY / DIFFICULT

Default:

CONVERGENCE = EASY

Related input tags: RHO_RELATIVE_ERROR; WG_ERROR; RHO_ERROR; SCF_ITER0; SCF_ITER1; ACCURACY.

Control the convergence parameters of the SCF self-consistent iteration.

CONVERGENCE	EASY	DIFFICULT
WG_ERROR	1.0E-4	0.5E-4
E_ERROR	1.0E-7*TOTNEL*Har	0.01E-7*TOTNEL*Har
RHO_ERROR	0.5E-4	0.5*0.5E-4
RHO_RELATIVE_ERROR	0.0	0.0
(TOTNEL: total number of electrons)		
(Har: 27.21138602eV)		

CONVERGENCE = EASY, use less self-consistent iteration steps to do the calculation in default setting.For the normal calculation, we recommend to use this setting. In some cases, it is hard to make the self-consistent iteration converge, you can try the "DIFFICULT" value.

CONVERGENCE = DIFFICULT, decrease several parameters for a better SCF convergence.

2.1.1.7 NUM_MPI_PER_GPU

NUM_MPI_PER_GPU = N

Default:

NUM_MPI_PER_GPU = 1

This parameter is used to control how many threads are bound to a GPU at the same time.

2.1.1.8 NUM_BLOCKED_PSI

NUM_BLOCKED_PSI=T/F

Default:

NUM_BLOCKED_PSI = F

In NUM_BLOCKED_PSI = T, PWmat will divide the wavefunctions into N parts and then put the parts into GPU memory successively one after another during scf iteration. This is to save the use of GPU memory. If a previous run found the GPU out of memory, this can be tried. **(WARNING)**:It is not allowed to use this parameter when "ENERGY_DECOMP = T" in etot.input. If NUM_BLOCKED_PSI = T, the decomposed energy can suddenly be very wrong.

This parameter intends to save the GPU memory to calculate a larger or more complicated systems. So when PWmat tells "CUDA MEMORY INSUFFICENT", one can try this parameter by setting NUM_BLOCKED_PSI=T, and etc. Note that using this parameter will reduce the speed of PWmat (e.g., by a factor of 1.5).

2.1.1.9 WF_STORE2DISK

WF_STORE2DISK=1 / 0

Default:

WF_STORE2DISK = 0

If WF_STORE2DISK = 1, the wavefunctions will be written into disk, otherwise written into cpu memory. This parameter is used to save cpu memory to calculate a larger or more complicated systems, in particular for the case multiple k-points are

calculated (then one can use WF_STORE2DISK=1). Note that: it will reduce the performance of PWmat in some degree.

2.1.1.10 USE_GAUSSIAN

USE_GAUSSIAN = T/F

Default: **USE_GAUSSIAN = F**

If **USE_GAUSSIAN = T**, PWmat will use gaussian basis instead of plane wave basis.

If **USE_GAUSSIAN = T**, there related some other parameters need to be set in etot.input, IN.PSP, IN.BASIS, IN.GAUSSIANOPT, OUT.GAUSSIAN, OUT.GTH2UPF.

IN.PSP specifies the pseudopotentials of each type of atoms, the format of IN.PSP:

```
IN.PSP1 = H GTH-PBE-q1 GTH_POTENTIALS
IN.PSP2 = C GTH-PBE-q4 GTH_POTENTIALS
```

The first parameter is the name of element, the second is the name of pseudopoential, the third is the name of psedupotential file. The psedupotential file can be downloaded from PWmat's website (http://www.pwmat.com/potential-download) or other opensource GTH psedupotentials. When USE_GAUSSIAN = T, PWmat use both gaussian basis and analytical GTH psedupotentials to do the calculation.

File GTH_POTENTIALS looks like (GTH_POTENTIALS contains pseudopotentials of the needed elements):

```
......
C GTH-PBE-q4 GTH-PBE
2    2
0.33847124    2    -8.80367398     1.33921085
2
0.30257575    1     9.62248665
0.29150694    0
......
```

For example, We need to set IN.PSP using the element symbol 'C' , the follwed name 'GTH-PBE-q4', and the file name 'GTH_POTENTIALS'.

IN.BASIS specifies the basis of each type of atoms, the format of IN.BASIS:

```
IN.BASIS1 = H SZV-MOLOPT-SR-GTH BASIS_MOLOPT
IN.BASIS2 = C SZV-MOLOPT-SR-GTH BASIS_MOLOPT
```

The first is the name of element, the second is the name of basis, the third is the name of basis file. The basis file can be downloaded from PWmat's website (http://www.pwmat.com/potential-download) or other opensource gaussian basis. The order of element type should be the same with IN.PSP* in etot.input.

File BASIS_MOLOPT looks like (BASIS_MOLOPT contains basis of the needed elements):

```
......
H  SZV-MOLOPT-SR-GTH SZV-MOLOPT-SR-GTH-q1
1
2 0 0 5 1
10.068468228533 -0.033917444900
 2.680222868089 -0.122202212100
 0.791501539122 -0.443818861200
 0.239116150487 -0.453182186600
 0.082193184441 -0.131612861500
......
```

For example, We need to set IN.BASIS using the element symbol 'H' , the follwed name 'SZV-MOLOPT-SR-GTH', and the file name 'BASIS_MOLOPT'.

IN.GAUSSIANOPT = T/F specifies some parameters of gaussian basis. If IN.GAUSSIANOPT = T , when USE_GAUSSIAN = T PWmat will read the parameters in file IN.GAUSSIANOPT (you need to create the file yourself), otherwise PWmat will use the default parameters. The parameters in file IN.GAUSSIANOPT are: EPS_GAUSSIAN, IS_PERIODIC_XYZ, USE_PERTURBATION, USE_ELPA2_OR_1, USE_CPU_ELPA , the format of IN.GAUSSIANOPT (with default values):

```
EPS_GAUSSIAN = 1.E-8
IS_PERIODIC_XYZ = F F F
USE_PERTURBATION = F
USE_ELPA2_OR_1 = 2
USE_CPU_ELPA = F
```

EPS_GAUSSIAN = 1.E-8 is the accuracy of gaussian basis, usually should be < 1.E-8, if the eigen solver ELPA failed, you can try to change this value to smaller.

IS_PERIODIC_XYZ = T/F T/F T/F is whether the system is periodic along lattice directions, along the three lattice directions, not cartesian directions. T means yes, F means no.

USE_PERTURBATION = T/F is whether use perturbation method to solve eigen energy and eigen wavefunctions (only can be used for insulators).

USE_ELPA2_OR_1 = 2/1 is whether use elpa2 (two-stage eigen solver, set 2) or elpa1 (one-stage eigen sovler, set 1) to solve eigen energy and eigen wavefunctions.

USE_CPU_ELPA = T/F is whether use CPU version ELPA to solve eigen energy and eigen wavefunctions, or use GPU version ELPA.

OUT.GAUSSIAN = T/F is whether output some files about gaussian basis. If OUT.GAUSSIAN = T, PWmat will output some files about gaussian basis.

OUT.GTH2UPF = T/F, when OUT.GTH2UPF=T, the program will convert the GTH pseudopoentials to UPF format. The output files are with prefix "GTH".

In current version PWmat with gaussian basis can just do a limited JOB=SCF/NONSCF/MD/RELAX, without stresses, HSE, LDA+U, SOC and many other parameters. But you can check the total energy, band structure, charge density and forces, and use multiple k-points and spin=1 or 2.

An example of gaussian basis,

file atom.config:

```
5
Lattice vector
    15.0000000000      0.0000000000      0.0000000000
```

```
    0.0000000000     15.0000000000      0.0000000000
    0.0000000000      0.0000000000     15.0000000000
Position, move_x, move_y, move_z
6    0.500000000000    0.500000000000    0.500000000000 0 0 0
1    0.474669993000    0.430790007000    0.518549979000 0 0 0
1    0.474669993000    0.518549979000    0.430790007000 0 0 0
1    0.474669993000    0.550670028000    0.550670028000 0 0 0
1    0.575999975000    0.500000000000    0.500000000000 0 0 0
```

file etot.input:

```
4  1
JOB = SCF
IN.ATOM = atom.config
ECUT = 50
IN.PSP1   = H GTH-PBE-q1 GTH_POTENTIALS
IN.PSP2   = C GTH-PBE-q4 GTH_POTENTIALS
IN.BASIS1 = H SZV-MOLOPT-SR-GTH BASIS_MOLOPT
IN.BASIS2 = C SZV-MOLOPT-SR-GTH BASIS_MOLOPT
USE_GAUSSIAN   = T
IN.GAUSSIANOPT = F
OUT.GAUSSIAN   = F
```

More over, you need to download the pseudopotential file GTH_POTENTIALS and basis file BASIS_MOLOPT from PWmat's website (http://www.pwmat.com/potential-download), and put them in the same directory with etot.input. If you set IN.GAUSSIANOPT = T, you need to create a file IN.GAUSSIANOPT in the same directory with etot.input, and set the parameters in IN.GAUSSIANOPT. If you set OUT.GAUSSIAN = T, PWmat will output some files about gaussian basis, you can use them to do some other calculations.

When OUT.GAUSSIAN = T, PWmat will output following files about gaussian basis:

OUT.GAUSSIAN_H,

OUT.GAUSSIAN_S,

OUT.GAUSSIAN_H_T,

OUT.GAUSSIAN_H_S,

OUT.GAUSSIAN_BASIS_INDEX .

OUT.GAUSSIAN_H_T: The matrix elements of the Hamiltonian in the Gaussian basis, labeled by translation vector T, with the following format,

$H_{m,m'}(T) = \int dr \chi^*_m(r - \tau_m)\hat{H}\chi_{m'}(r - (\tau_{m'} + T))$

where m and m' are the index of the basis, T is the translation vector, χ_m is the m-th basis function, τ_m is the position of the m-th basis function(i.e. the position of the atom which the m-th basis function belongs to), $\hat{H}$ is the Hamiltonian operator.

You can use following fortran codes to read the file OUT.GAUSSIAN_H_T:

```
      subroutine read_gaussian_H_T()
      implicit none
      complex(kind=8), allocatable, dimension(:, :, :, :, :) :: H
      integer :: Nx_pbc_t, Ny_pbc_t, Nz_pbc_t, num_mcgtos_t
      integer :: i, j, k, l, m, n
      real*8 :: AL(3, 3)
      complex(kind=8), allocatable, dimension(:, :) :: T_all
      integer :: T(3)
      !
      open (10, file="OUT.GAUSSIAN_H_T", form="unformatted", status="old")
      read (10) Nx_pbc_t, Ny_pbc_t, Nz_pbc_t, num_mcgtos_t, AL
      allocate (H(num_mcgtos_t, num_mcgtos_t, &
      -Nx_pbc_t:Nx_pbc_t, -Ny_pbc_t:Ny_pbc_t, -Nz_pbc_t:Nz_pbc_t))
      allocate (T_all(num_mcgtos_t, num_mcgtos_t))
      do i = -Nx_pbc_t, Nx_pbc_t
          do j = -Ny_pbc_t, Ny_pbc_t
              do k = -Nz_pbc_t, Nz_pbc_t
                  read (10) T(1), T(2), T(3)
                  do l = 1, num_mcgtos_t
                      read (10) T_all(:, l)
                  end do
                  do l = 1, num_mcgtos_t
                      do m = 1, num_mcgtos_t
                          H(l, m, i, j, k) = T_all(l, m)
                      end do
                  end do
              end do
          end do
```

```
end do
close (10)
deallocate (T_all)
deallocate (H)
end subroutine read_gaussian_H_T
!
```

where AL(1:3,1:3) is the lattice vectors (in unit bohr), T(1:3) is the translation vector, num_mcgtos_t is the number of the basis functions. (Note, T is an integer array, the acctual translation vector is AL*T.)

OUT.GAUSSIAN_S_T: The matrix elements of the overlap matrix in the Gaussian basis, labeled by translation vector T, with the following format,

$S_{m,m'}(T) = \int dr \chi^*_m(r - \tau_m)\chi_{m'}(r - (\tau_{m'} + T))$

where m and m' are the index of the basis, T is the translation vector, χ_m is the m-th basis function, τ_m is the position of the m-th basis function(i.e. the position of the atom which the m-th basis function belongs to).

You can use following fortran codes to read the file OUT.GAUSSIAN_S_T:

```
subroutine read_gaussian_S_T()
implicit none
complex(kind=8), allocatable, dimension(:, :, :, :, :) :: S
integer :: Nx_pbc_t, Ny_pbc_t, Nz_pbc_t, num_mcgtos_t
integer :: i, j, k, l, m, n
real*8 :: AL(3, 3)
complex(kind=8), allocatable, dimension(:, :) :: T_all
integer :: T(3)
!
open (10, file="OUT.GAUSSIAN_S_T", form="unformatted", status="old")
read (10) Nx_pbc_t, Ny_pbc_t, Nz_pbc_t, num_mcgtos_t, AL
allocate (S(num_mcgtos_t, num_mcgtos_t, &
-Nx_pbc_t:Nx_pbc_t, -Ny_pbc_t:Ny_pbc_t, -Nz_pbc_t:Nz_pbc_t))
allocate (T_all(num_mcgtos_t, num_mcgtos_t))
do i = -Nx_pbc_t, Nx_pbc_t
    do j = -Ny_pbc_t, Ny_pbc_t
        do k = -Nz_pbc_t, Nz_pbc_t
            read (10) T(1), T(2), T(3)
```

```
                do l = 1, num_mcgtos_t
                    read (10) T_all(:, l)
                end do
                do l = 1, num_mcgtos_t
                    do m = 1, num_mcgtos_t
                        S(l, m, i, j, k) = T_all(l, m)
                    end do
                end do
            end do
        end do
    end do
    close (10)
    deallocate (T_all)
    deallocate (S)
    end subroutine read_gaussian_S_T
    !
```

where AL(1:3,1:3) is the lattice vectors (in unit bohr), T(1:3) is the translation vector, num_mcgtos_t is the number of the basis functions. (Note, T is an integer array, the acctual translation vector is AL*T.)

OUT.GAUSSIAN_H: The matrix elements of the Hamiltonian matrix in the Gaussian basis, with the following format,

$H^{\sigma}_{m,m'}(k) = \sum_T exp(ik \cdot T) H^{\sigma}_{m,m'}(T)$

where m and m' are the index of the basis, k is the K-point, σ is the spin index (Note in current version σ can just be 1).

You can use following fortran codes to read the file OUT.GAUSSIAN_H:

```
    subroutine read_gaussian_H()
    implicit none
    complex(kind=8), allocatable, dimension(:, :, :, :) :: H
    integer :: nkpt_t, islda_t, num_mcgtos_t
    integer :: i, j, k, l, m, n, iislda, ikpt,iislda_t, ikpt_t
    real*8 :: AL(3, 3)
    real*8 :: akx_2_t, aky_2_t, akz_2_t
    complex(kind=8), allocatable, dimension(:, :) :: T_all
    !
```

```
      open (10, file="OUT.GAUSSIAN_H", form="unformatted", status="old")
      read (10) nkpt_t, islda_t, num_mcgtos_t, AL

      allocate (H(num_mcgtos_t, num_mcgtos_t, nkpt_t, islda_t))

      allocate (T_all(num_mcgtos_t, num_mcgtos_t))
      do iislda = 1, islda_t
          do ikpt = 1, nkpt_t
              read (10) iislda_t, ikpt_t, akx_2_t, aky_2_t, akz_2_t
              do i = 1, num_mcgtos_t
                  read (10) T_all(:, i)
              end do
              do i = 1, num_mcgtos_t
                  do j = 1, num_mcgtos_t
                      H(i, j, ikpt, iislda) = T_all(i, j)
                  end do
              end do
          end do
      end do
      close (10)
      deallocate (T_all)
      deallocate (H)
      end subroutine read_gaussian_H
      !
```

where AL(1:3,1:3) is the lattice vectors (in unit bohr), akx_2_t, aky_2_t, akz_2_t are the K-point cartesian coordinates (in unit 1/bohr), num_mcgtos_t is the number of the basis functions. (Note, T is an integer array, the acctual translation vector is AL*T.)

OUT.GAUSSIAN_S: The matrix elements of the overlap matrix in the Gaussian basis, with the following format,

$S_{m,m'}(k) = \sum_T exp(ik \cdot T) S_{m,m'}(T)$

where m and m' are the index of the basis, k is the K-point.

You can use following fortran codes to read the file OUT.GAUSSIAN_S:

```
      subroutine read_gaussian_S()
      implicit none
```

```
complex(kind=8), allocatable, dimension(:, :, :, :) :: S
integer :: nkpt_t, islda_t, num_mcgtos_t
integer :: i, j, k, l, m, n, iislda, ikpt,iislda_t, ikpt_t
real*8 :: AL(3, 3)
real*8 :: akx_2_t, aky_2_t, akz_2_t
complex(kind=8), allocatable, dimension(:, :) :: T_all
!
open (10, file="OUT.GAUSSIAN_S", form="unformatted", status="old")
read (10) nkpt_t, islda_t, num_mcgtos_t, AL

allocate (S(num_mcgtos_t, num_mcgtos_t, nkpt_t, islda_t))

allocate (T_all(num_mcgtos_t, num_mcgtos_t))
do iislda = 1, islda_t
    do ikpt = 1, nkpt_t
        read (10) iislda_t, ikpt_t, akx_2_t, aky_2_t, akz_2_t
        do i = 1, num_mcgtos_t
            read (10) T_all(:, i)
        end do
        do i = 1, num_mcgtos_t
            do j = 1, num_mcgtos_t
                S(i, j, ikpt, iislda) = T_all(i, j)
            end do
        end do
    end do
end do
close (10)
deallocate (T_all)
deallocate (S)
end subroutine read_gaussian_S
!
```

where AL(1:3,1:3) is the lattice vectors (in unit bohr), akx_2_t, aky_2_t, akz_2_t (in unit 1/bohr) are the K-point coordinates, num_mcgtos_t is the number of the basis functions. (Note, T is an integer array, the acctual translation vector is AL*T.)

The unit of AL is Bohr, the unit of akx_2_t, aky_2_t, akz_2_t is 2π/bohr.

OUT.GAUSSIAN_BASIS_INDEX: The index of the basis functions of all atoms,

with the following format,

```
subroutine read_gaussian_basis_index()
implicit none
integer :: natom_t, num_mcgtos_t
integer, allocatable, dimension(:, :) :: basis_range_iatom
integer :: i, j, k, l, m, n
!
open (10, file="OUT.GAUSSIAN_BASIS_INDEX", form="unformatted", &
      status="old")
read (10) natom_t, num_mcgtos_t
allocate (basis_range_iatom(2, natom_t))
read (10) basis_range_iatom(:, 1:natom_t)
close (10)
deallocate(basis_range_iatom)
end subroutine read_gaussian_basis_index
!
```

where basis_range_iatom(1, i) is the index of the first basis function of the i-th atom, basis_range_iatom(2, i) is the index of the last basis function of the i-th atom. Note, the index of atoms may not be the original index of atoms in atom.config(specified by IN.ATOM), PWmat will reorder the atoms, so check output file ORIGIN.INDEX to check index mapping.

Note: Only the upper triangle in output matrix (OUT.GAUSSIAN_H_T, OUT.GAUSSIAN_S_T, OUT.GAUSSIAN_H, OUT.GAUSSIAN_S) has correct values, the lower triangle is not correct. The lower triangle is just for the convenience of reading the matrix elements. You can get the lower triangle by using the Hermite conjugate of the upper triangle.

Note: If you want to use fractional hydrogen potentials, you need to add them in potential files, and use the correct symbols in IN.PSP. For example,

```
in.psp1= N GTH-PBE-q5      GTH_POTENTIALS
in.psp2= Ga GTH-PBE-q3     GTH_POTENTIALS
in.psp3= H7 GTH-PBE-q0.75  GTH_POTENTIALS
in.psp4= H8 GTH-PBE-q1.25  GTH_POTENTIALS
```

In PWmat, the symbol of hydrogen is “H”, the symbol of fractional hydrogen is “H1,H2,...”, as follows,

symbol	n_electron	element number
H	1	1
H1	0.25	109
H2	0.33	110
H3	0.42	111
H4	0.50	112
H5	0.58	113
H6	0.66	114
H7	0.75	115
H8	1.25	116
H9	1.33	117
D0	1.50	118
D1	1.66	119
D2	1.75	120
D3	0.17	121

2.1.1.11 USE_DFTB

Default: **USE_DFTB= F**

If **USE_DFTB= T**, PWmat will use density function based tight binding method

This is a module that links to the open source software DFTBPLUS (following the LGPL agreement), Provide PWMAT’s user with convenient and fast density functional based tight-binding DFTB computing capabilities.

Currently the main features of PWMAT-DFTB are:

- supports PWMAT pre- and post-processing format and simplified parameter interface
- Supports structural optimization and MD calculation of periodic structures
- Supports density matrix construction and Real Hamitonian diagonalization on GPU

If **USE_DFTB= T**, there related some other parameters need to be set in etot.input.

- **etot.input** (control file)
- **atom.config** (structure file)
- **X-X.skf/X-Y.skf** (Slater-Koster files(SKF))

When running, the intermediate temporary file HSD file "dftb_in.hsd_pwmat" will be generated in the current directory, and dftb+ will read the generated file for calculation. Informations such as geometry trajectory, forces, stress and energy will be saved in the MOVEMENT file.

This is a example "etot.input" for DFTB calculation.

```
1    1            # The first line are ignored when USE\_DFTB=T
USE_DFTB=T # use dftb method
JOB =MD   # molecular dynamic
MD_DETAIL = 1 10 0.1 100.0 100.0 # NVE, timestep 0.1fs run 10 steps at temperature 100K
IN.ATOM=atom.config  # structure file
IN.SKF='./' # skf files's dir
DFTB_DETAIL = 1 1 0 100 1e-5 1 0 #
MP_N123 = 1 1 1 0 0 0 1 # K points setting
NUM_GPU = 1 # optional, using number of GPU cards
```

When turning on DFTB calculation, you need to set "USE_DFTB = T", "IN.SKF" is used to specify the path of the SKF parameter files, which defaults to the current path. For more information about Slater-Koster parameters, it is recommended to refer to dftb website. "JOB" currently supports RELAX and MD keywords; The keyword "DFTB_DETAIL" is a key setting parameter, which will be explained in detail below.

2.1.1.12 DFTB_DETAIL

Format: DFTB_DETAIL = isemi, iscc, diag_method, scc_maxiter, scc_tolerance, mixer_method, disp_method

Default: DFTB_DETAIL = 1, 1, 0, 200, 1e-6, 1, 0

isemi:The DFTB method used. Can be set to 0/1/2/3/4. Respectively

1. isemi=0, Non-Scc DFTB [35];
2. isemi=1, SCC-DFTB [36];
3. isemi=2, DFTB3;[39];
4. isemi=3, GFN1-XTB[49];
5. isemi=4, GFN2-XTB[48];

In most cases, for semiconductor or metal oxide systems, isemi is recommended to be set to 1; for organic molecular biological systems, isemi is recommended to be 2 or 3.

When using the GFN-xTB method, the program reads the built-in parameters of xtb by default. In addition, we can also read the specified parameter file through the "**IN.XTB**" keyword.

Format: IN.XTB=your-gfn-xtb-param-file For how to create and fit the parameters of GFN-XTB, It is recommended to refer to tblite tutorial

iscc: Is the switch to turn on charge self-consistency. Set to 0/1. Enabled by default;

diag_method: Hamiltonian matrix diagonalization method, can be set to 0/1/2/3/4, default is 0.

1. diag_method = 0, Divide and Conquer (cpu version)
2. diag_method=1, Divide and Conquer (gpu version)
3. diag_method=2, RelativelyRobust
4. diag_method=3, QR
5. diag_method=4, ELPA (gpu version)

When diag_method=1, the program will use GPU to solve the matrix. You can perform single-node multi-card calculations by setting the keyword "NUM_GPU=number of cards". It is not recommended to use GPU for calculations in systems with a small number of atoms (perhaps <1000), because a small system means a small matrix dimension, and DFTB calculations on a GPU are not necessarily faster than on a CPU. Large systems generally only require single gamma point calculations, and the matrices are all real numbers, so the GPU version only supports real hamitional diagonalization calculations.

scc_maxiter: Maximal number of SCC cycles to reach convergence. If convergence is not reached after the specified number of steps, the program stops. The default is 200 steps.

scc_tolerance: Stopping criteria for the SCC. Specifies the tolerance for the maximum difference in any charge between two SCC cycles. The default is 1e-6.

mixer_method: Mixer type for mixing the charges in an SCC calculation. Set to 1/2/3/4.

1. mixer_method=1, Broyden method (mixingParameter=0.2)
2. mixer_method=2, Anderson method
3. mixer_method=3, DIIS
4. mixer_method=4, Simple method (mixingParameter=0.05)

disp_method: Specifies which kind of dispersion correction to apply. Set to 0/1/2/3/4, default is 0.

The *disp_method* controls whether DFTB interactions should be empirically corrected for van der Waals interactions, since DFTB (and SCC-DFTB) does not include these effects. Currently, four different dispersion correction schemes are used in PWMAT-DFTB (for the detailed description of the methods see the following subsections):

1. disp_method=0, None

2. disp_method=1, LennardJones style[40]

3. disp_method=2, Grimme's D3[42]

4. disp_method=3, Grimme's D4[44]

5. disp_method=4, Tkatchenko-Scheffler[45]

when disp_method=1, Dispersion is included via a Lennard-Jones potential between each pair of atoms. The program can automatically take the parameters from the Universal Force Field (UFF)[51].

The Lennard-Jones dispersion model in dftbplus follows the method of Ref. [40], using the following potential:

$$\begin{aligned} U_{ij}(r) &= d_{ij}\left[-2\left(\frac{r_{ij}}{r}\right)^6+\left(\frac{r_{ij}}{r}\right)^{12}\right] \qquad r >= r_0 \\ U_{ij}(r) &= U_0 + U_1 r^5 + U_2 r^{10} \qquad r < r_0 \end{aligned}$$

where r_0 is the distance at which the potential turns from repulsive to attractive. The parameters d_{ij} and r_{ij} are built from atomic parameters d_i, d_j and r_i, r_j via the geometrical mean ($d_{ij} = \sqrt{d_i d_j}$, $r_{ij} = \sqrt{r_i r_j}$). The parameters U_0, U_1, U_2 ensure a smooth functional form at r_0.

2.1.1.13 D3_DETAIL

When disp_method=2, Dispersion is calculated as in the s-dftd3 library [42, 43]

Format: D3_DETAIL = d3_method, a1, a2, s6, s8

Default: D3_DETAIL = 0, 0.5719, 3.6017, 1.0, 0.5883

d3_method use to set DFTD3's implement methods, current only to 0, which dispersion is calculated as in the s-dftd3 library. The parameters a1, a2, s6, s8 can be specified manualy adjusted.

As the DFTB energy functional is largely determined by the underlying parameterisation (the Slater-Koster-files) and the chosen DFTB model (e.g. non-scc,

scc, 3rd order, etc.), there are no universal parameter choices which can be used with all settings, but some relevant choices for various parameterisation are given in the following tables2.1.1.13.

Becke-Johnson damping	old default	3OB	OB2 (base)	OB2 (shift)	OB2 (split)
a_1	0.5719	0.746	0.717	0.816	0.497
a_2	3.6017	4.191	2.565	2.057	3.622
s_6	1.0	1.0	1.0	1.0	1.0
s_8	0.5883	3.209	0.011	0.010	0.010

2.1.1.14 D4_DETAIL

disp_method=4 When using the D4 dispersion correction method, you can set detailed parameters or use default values.

Format: D4_DETAIL = a1, a2, s8, s9

Default value: D4_DETAIL = 0.5467502, 4.4955068, 0.4727337, 0.0

is similar to D3. It is recommended to use different parameters for different parameter sets. The following table gives recommended values for reference.

Table 2.1: Becke–Johnson damping parameters for various Slater–Koster parametrizations of the DFTB hamiltonian. Parametrizations are done both with non-additive contributions and without.

parameters	s_8	s_9	a_1	a_2 [a_0]
3ob	0.4727337	0	0.5467502	4.4955068
	0.6635015	1	0.5523240	4.3537076
matsci	2.7711819	0	0.4681712	5.2918629
	3.3157614	1	0.4826330	5.3811976
mio	1.1948145	0	0.6074567	4.9336133
	1.2916225	1	0.5965326	4.8778602
ob2(base)	2.7611320	0	0.6037249	5.3900004
	2.9692689	1	0.6068916	5.4476789
pbc	1.7303734	0	0.5546548	4.7973454
	2.1667394	1	0.5646391	4.9576353

2.1.1.15 DFTB's MD_DETAIL

The parameter format of molecular dynamics simulations is similar to PWMAT.

Format: MD_DETAIL = iMD, MDstep, dtMD, Temperature1, Temperature2

1. iMD=1, NVE
2. iMD=2, NVT nose-hoover
3. iMD=21, NVT Thermostat=Andersen
4. iMD=22, NVT Thermostat=Berendsen
5. iMD=3, Xlbomd
6. iMD=4, XlbomdFast

For several systems Born-Oppenheimer molecular dynamics simulations can be significantly sped up by using the extended Lagrangian formalism described in Ref. [46]. The XLBOMD integrator can be used in two different modes:

- Conventional XLBOMD scheme (*Xlbomd*): The extended Lagrangian is used to predict the input charge distribution for the next time step, instead of taking charges that were converged for the geometry in the previous time step. The predicted starting charges should then require fewer SCC iterations to converge.
- Fast XLBOMD scheme, *XlbomdFast* (one diagonalisation per time step): The extended Lagrangian is used to predict the population for each time step. This predicted population is then used to build the Hamiltonian, but in contrast to the conventional Xlbomd scheme, there is no self consistent cycle the forces are calculated immediately after the diagonalisation of the first Hamiltonian. The fast Xlbomd method usually only works for systems without SCC instabilities (e.g. wider gap insulators or molecules without degenerate states). See Ref. [46] for details.

The "XLBOMD_DETAIL" parameter can be set as:

Format: XLBOMD_DETAIL = pre_steps, initg_steps, minScc

1. **pre_steps=3** Number of molecular dynamics time steps before the XLBOMD integration becomes activated.

2. **init_steps =5** Number of time steps used for determining the population for the next time step. Currently, only integration schemes for 5, 6 or 7 steps are implemented.

3. **minScc=1** Minimum number of SCC iterations to perform at each time step.

When iMD=4, The "XLBOMD_FAST_DETAIL" parameter can be set as

1. **pre_steps=3** Number of molecular dynamics time steps before the XLBOMD integration becomes activated.

2. **init_steps =5** Number of time steps used for determining the population for the next time step. Currently, only integration schemes for 5, 6 or 7 steps are implemented.

3. **transient_steps=10** Enables a smoother transition between Born-Oppenheimer and extended Lagrangian dynamics by carrying out intermediate additional steps with full SCC convergence, during which the converged population and the one predicted by the extended Lagrangian integrator are averaged.

4. **scale=0.5** Scaling factor for the predicted charge densities $\in (0, 1]$. The optimal value is system dependent. One should take the highest possible value that still produces stable dynamics (good conservation of energy).

2.1.1.16 DFTB's RELAX_DETAIL

Format: RELAX_DETAIL = IMIN, NSTEP, FORCE_TOL, LatticeOpt
IMIN types of the geometry optimiser to use

1. IMIN = 0, Rational function based optimiser

2. IMIN = 1, ConjugateGradient

3. IMIN = 2, LBFGS
4. IMIN = 3, SteepestDescent
5. IMIN = 4, FIRE (The fast inertial relaxation engine)[50]

When IMIN=0 is used, the final.config and RELAXSTEPSOPT files will be output. The option of IMIN>0 is planned to be removed in dftbplus in the future, and the code structure has changed greatly, so the output of RELAXSTEPSOPT is not yet supported.
NSTEP Maximum number of steps after which the optimisation should stop (unless already stopped by achieving convergence). Default is 1000 steps
FORCE_TOL Force Optimisation is stopped, if the force component with the maximal absolute value goes below this threshold.
LattOpt Options for whether to perform lattice optimization

1. LattOpt = 0 does not perform lattice optimization
2. LattOpt = 1 Consider lattice optimization
3. LattOpt = 2 Lattice optimization considering pressure, requires new keyword "PRESSURE"

PRESSURE Set external pressure options.
Format: PRESSURE =
Default value: PRESSURE = 0.0
FIXANGLES When **LattOpt>0**, this keyword can be turned on to fix the angle of the crystal vector during lattice optimization.
FIXLENGTHS This keyword is turned on when **LattOpt>0** and is used to limit the length of the three axes of the lattice. Default value: **FIXLENGTHS=F F F**

2.1.1.17 DFTB's LDA+U

Currently the *FLL* (fully localised limit) and *pSIC* [38] (pseudo self interaction correction) forms of the LDA+U corrections [**?**] are implemented. These potentials

effect the energy of states on designated shells of particular atoms, usually increasing the localisation of states at these sites. The *FLL* potential lowers the energy of occupied states localised on the specified atomic shells while raising the energy of unoccupied states. The *pSIC* potential corrects the local part of the self-interaction error and so lowers the energy of occupied states (see Ref. [38] for a discussion of the relation between these two potentials, and possible choices for the UJ constant). These particular corrections are most useful for lanthanide/actinide f states and some localised d states of transition metals (Ni3d for example).

1. LDAU_METHOD = NONE, default value
2. LDAU_METHOD = FLL, full localized limit
3. LDAU_METHOD = PSIC, pseudo self interaction correction

When using LDA+U, the atomic orbital to which U is to be added must be specified for each element type (i), as well as the U value.

Format:

TB_LDAU_PSP1= LDAU_L1, s_hubbard_U1 p_hubbard_U1 d_hubbard_U1 f_hubbard_U1

TB_LDAU_PSP2= LDAU_L2, s_hubbard_U2 p_hubbard_U2 d_hubbard_U2 f_hubbard_U2

...

Indent:

TB_LDAU_PSP1 = -1

TB_LDAU_PSP2 = -1

...

Note: *TB_LDAU_PSP(i) should correspond to the type order of atom.config. spdf_hubbard_U adds the U value of each shell in turn, When U is not added, it is expressed as 0.0, and the program will not read the U value of this shell.*

TB_LDAU_L(i) = -1/1, -1 means not using LDA+U, and an integer greater than -1 is used to enable LDA+U,

2.1.1.18 DFTB's SPIN

Format: SPIN = 1 / 2 / 222

Default value: SPIN = 1

SPIN = 1, non-spin polarization calculation (default value). Each orbital will be occupied by 2 electrons.

SPIN = 2, spin polarization calculation along Z axis. After turning it on, you can set the parameter "SPIN_COLINEAR" to control the number of unpaired electrons and whether to relax the total spin.

Format: SPIN_COLINEAR = UNPARIED_ELCE, RELAX_TOTALSPINS

Default value: SPIN_COLINEAR = 0.0, 0

UNPAIRED_ELEC Sets the number of unpaired electrons. If the keyword "RELAX_TOTALSPINS"(>0) is used, the number of unpaired electrons will change.

RELAX_TOTALSPINS This item has two values, 0 or 1; when 1, it means that spin relaxation is turned on, and the total spin of the system will change. The "UNPAIRED_ELEC" value is set as the initial value. When it is 0, the spin polarization of the system remains unchanged.

INITIAL_SPINS_PSP Set the initial spin according to the order of element types in atom.config. The default value is 0.0

Format:

INITIAL_SPINS_PSP1 = value1

INITIAL_SPINS_PSP2 = value2

...

When considering nonlinear spin polarization, "INITIAL_SPINS_PSP" is used to set the spin vector (vector(3)) of each atom

Format:

INITIAL_SPINS_PSP1 = value1, value2, value3

INITIAL_SPINS_PSP2 = value21, value22, value23

...

Provides atomic constants required for spin polarization calculations or evaluation of properties that depend on spin interactions, such as triplet excitations. In these cases, for each atomic species in the calculation, the spin coupling constant for that atom must be specified. When shell-resolved spin constants are specified, they must be ordered according to the shell pairs to which they are coupled. For the case when based on spd orbitals, the following ordering is given:

$$w_{ss}, w_{sp}, w_{sd}, \ldots, w_{ps}, w_{pp}, w_{pd}, \ldots, w_{ds}, w_{dp}, w_{dd}, \ldots \tag{2.3}$$

The keyword "SPIN_CONSTANTS_PSP" gives the specific value When using the SCC-DFTB/DFTB3 method, there is only one corresponding value of "SPIN_CONSTANTS_PSP"; When using the GFN-xTB method, different numbers need to be set according to the spdf shell, s corresponds to 1, p corresponds to 4, d corresponds to 9, and f corresponds to 16.

The settings in this part are prone to errors, and we often encounter problems such as "Superfluous data found." Please use it with caution. we simple judge that there may be a problem with the parsing parameters in dftbplus.

Format:

SPIN_CONSTANTS_PSP1 = value1, value2, ...

SPIN_CONSTANTS_PSP2 = value21, value22, ...

...

USE_SOC is used to set up Spin-Orbit calculations then specifies that the LS coupling should be included in the calculation. Spin non-polarization and non-collinear spin polarization are currently supported, but collinear spin polarization is not supported. For each atomic species present in the calculation, the spin-orbit coupling constant for that atom must be specified for all shells present. And the constants must be ordered according to the shell list of a given atom. The value of the s orbital is set to 0.0

SOC_PSP is used to specify soc for each atom type

Format:

SOC_PSP1 = 0.0

SOC_PSP2 = 0.0, 0.2

SOC_PSP3 = 0.0, 0.2, 0.2

SOC_PSP4 = 0.0, 0.2, 0.2, 0.2

SPIN = 222, spin-orbit coupling calculation including non-collinear magnetization (nonlinear magnetic moment). Need to set the "SOC_PSP" keyword parameter

2.1.1.19 DFTB's Other parameters

1. **ELEC_TEMP=300.0** Electron temperature(K), used in Filling method(Fermi distribution)
2. **CHARGE=0.0**, Total charge of the system in units of the electron charge, Negative values mean an excess of electrons.
3. **RANDOM_SEED=123456**, Sets the seed for the random number generator.
4. **SCC_MAXITER=100**, Maximal number of SCC cycles to reach convergence.
5. **SCC_TOLERANCE=1e-6**, Stopping criteria for the SCC.
6. **IN.HSD**='Your HSD filename', Use to direct read dftbplus's hsd input file, when using it, Other etot.input's parameters will not work

HBOND_CORR=1/2 turns on hydrogen bond correction when used in DFTB3. The default value is currently used.

1. HBOND_CORR=1 H5 method
2. HBOND_CORR=2 Damping method

"HBOND_CORR=1" is to enable the DFTB3-D3H5 method, which adds correction for non-covalent bonding based on DFTB3. This method is paired with the 3ob parameter set and replaces gamma-function damping with H5 correction and additional D3 dispersion correction. In addition, a repulsive term is introduced to prevent unphysical behavior caused by H atoms being too close.

"HBOND_CORR=2" will add an attenuation factor to the short-range interaction between two atoms during SCC calculation to modify the effect of the H bond (so at least one of the two atoms is an H atom)

$$e^{-\left(\frac{U_{Al}+U_{Bl}}{2}\right)^{\zeta} r_{AB}^2} \tag{2.4}$$

The ζ parameter here defaults to 4.05

"JOB" adds a new option "PDOS", which is used to analyze the partial wave density of states, and will output the partial wave density of states of all atoms.

2.1.2 System tags

2.1.2.1 ECUT

Default:

ECUT = "WFC_CUTOFF" in pseudopotential file

The plane wave cutoff energy for wavefunction (in Ryd, note: $1Ryd = 13.6057eV$). The default value of ECUT is taken from the pseudopotential files atom.upf from its WFC_CUTOFF value. Note, in an plane wave calculation, the plane wave functions with their G-vector ($exp(-iG * x)$) within the energy sphere of ECUT is used as the basis function. Thus, ECUT control the size of the plane wave basis set, is one of the most important calculating parameter.

2.1.2.2 ECUT2

Default:

ECUT2 = 2*ECUT (ACCURACY = NORM)

ECUT2 = 4*ECUT (ACCURACY = HIGH or ACCURACY = VERYHIGH)

The cutoff energy for the soft charge density and the potential (in Ryd). In a plane wave calculation, not only the orbital are expanded by the plane waves, the charge density is also expanded by the plane waves. However, the plane wave basis set (within

energy ECUT2) used to expand the charge density is larger than the plane wave basis set (within energy ECUT) used to expand the orbital.

Ideally (for high accurate calculations), ECUT2 should equal 4*ECUT. But in reality, smaller ECUT2 can sometime be used, e.g., 3*ECUT, or 2*ECUT. By default, ECUT2 = 2*ECUT for normal accuracy calculation (ACCURACY=NORM), and ECUT2=4*ECUT for high accuracy calculation (ACCURACY=HIGH or VERYHIGH). For JOB=RELAX, to avoid the egghead jittering effect, we recommend to use ECUT2=4*ECUT.

The N1, N2, N3 are determined by ECUT2. Note, the RHO_CUTOFF value in the pseudopotential files atom.upf is not used. Also note that, if N1, N2, N3 are not set (by $N123 =$), they will be generated by ECUT2, together with NODE1. So, if different number of node NODE1 are used, the N1, N2, N3 values could be different even for the same ECUT2. This is because N1*N2 must be evenly divided by NODE1.

2.1.2.3 ECUT2L

Default:

ECUT2L = ECUT2 (NCPP, ACCURACY = NORM or ACCURACY = HIGH)

ECUT2L = 4*ECUT2 (NCPP, ACCURACY = VERYHIGH)

ECUT2L = 4*ECUT2 (USPP)

The cutoff energy for the hard charge density (in *Ryd*).

Sometime it is necessary to further increase the accuracy of the description for the charge density rho(r) before it is used to calculate the potential via the exchange-correlation functional. Thus, we have a so-called hard charge density, which is described by a plane wave basis set within ECUT2L.

Usually, ECUT2L = ECUT2 for norm conserving pseudopotentials, and ECUT2L = 4 * ECUT2 for ultrasoft pseudopotentials. However, sometime to completely remove the egghead problem, we can also use ECUT2L = 4 * ECUT2 even for the norm conserving psp. Nevertheless, that egghead problem can usually be solved by using EGG_FIT, so we can still use ECUT2L = ECUT2 for norm conserving pseudopotential (but usually

require ECUT2 = 4 * ECUT).

2.1.2.4 ECUTP

Default:

ECUTP = ECUT (ACCURACY = NORM)

ECUTP = 4*ECUT (ACCURACY = HIGH or ACCURACY = VERYHIGH)

The cutoff energy to generate P123 for Fock exchange integral evaluation for HSE calculations (in *Ryd*). If P123 is exlicitly input, the P123 will have higher priority.

Note, in order to have accurate results, one needs to have ECUTP = 4*ECUT. This is necessary for accurate force calculations, e.g., during atomic relaxation or phonon mode calculations. Otherwise, the total force might not be zero.

However, if only electronic structure is needed, or for molecular dynamics, or evern TDDFT simulations, one might be able to set a smaller ECUTP, for example, ECUTP=ECUT

2.1.2.5 N123

The format is like this:

N123 = N1, N2, N3

N1, N2, N3 are the real space grid to describe the wave function or soft charge density in real space. It is also the FFT grid. The default values are determined by ECUT2 (i.e., make sure the ECUT2 sphere can be held inside the $N1, N2, N3$ reciprocal box). Roughly speaking, $N_i = \sqrt{2 * ECUT2}/(\pi * |ALI(:, i)|)$, here $ALI(:, i)$ is the reciprocal lattice of the input cell lattice $AL(:, i)$ in atom.config file, in *Bohr* unit, and $|ALI(:, i)|$ is the length of the vector. Also, in this formula, $ECUT2$ is in the unit of Hartree.

Note, in our current implementation, $N1*N2*N3$ need to be divided evenly by node1. So, sometime it might be necessarily to readjust $N1, N2, N3$ manually (or to change node1). If $N123$ is not explicitly set, their values will be determined automatically by ECUT2 and NODE1. Note, for the same ECUT2, for different NODE1, it can lead to different $N123$.

2.1.2.6 N123_METH

N123_METH = 0/1

Default: **N123_METH = 0**

If N123_METH = 0, check the section N123. If N123_METH = 1, the way to generate grid of real space will be different with the detault settings, N1 will not be required to be divisible by NODE1. Instead, N1*N2 must be divisible by NODE1 and N1*(N3/2+1) be divisible by NODE1.

2.1.2.7 N123L

The format is:

N123L = N1L, N2L, N3L

N1L, N2L, N3L are the real space grid for hard charge density. The default values are determined by ECUT2L. For norm conserving pseudopotential, the soft charge equals hard charge, ECUT2L=ECUT2, so N1L, N2L, N3L equal N1, N2, N3. For ultrasoft, ECUT2L = 4 * ECUT2, N1L, N2L, N3L = 2 * N1, 2 * N2, 2 * N3.

2.1.2.8 NS123

The format is :

NS123 = N1S, N2S, N3S

N1S, N2S, N3S are the real space FFT grid point to calculate the real space nonlocal pseudopotential projector function. So, these are only used for NONLOCAL = 2. For small systems, N1S,N2S,N3S can be larger than N1,N2,N3. For large systems, smaller values can be used to save time for projector generation. Usually these parameters are set automatically.

2.1.2.9 MP_N123

MP_N123=NK1, NK2, NK3, SK1, SK2, SK3, FLAG_SYMM

Default:

MP_N123 = 1 1 1 0 0 0

This variable is the Monkhorst-Pack grids to generate the reduced k-points. When this line is provided, the PWmat will generate the OUT.SYMM and OUT.KPT using the above Monkhorst-Pack parameters, and the PWmat will continue to run the JOB using these k-points and symmetries.

(WARNING): if one wants to generate only gamma point, but does not want to use symmetry, one needs to set: "MP_N123 = 1 1 1 0 0 0 2".

Note: if file "IN.KPT" exists and IN.KPT=T, but at the same time, MP_N123 is also specified, PWmat will ignore "MP_N123" and use kpoints readin from file "IN.KPT"(i.e, IN.KPT=T has higher priority than MP_N123). If you want to use the input kpoints and symmetry, you should set IN.KPT=T, IN.SYMM=T, then PWmat will read the files "IN.KPT", "IN.SYMM" for the calculation.

The SK1, SK2 and SK3 must be either 0 (no offset) or 1 (grid displaced by half a grid point in the corresponding direction). This is the standard options to generate the Monkhorst-Pack k-point grid.

The FLAG_SYMM controls the symmetry operation(the operations are stored in OUT.SYMM) of k-points. One can refer to the OUT.SYMM for the specific symmetry operations.

FLAG_SYMM=0, generate kpoints with spatial symmetry and time reversal symmetry. This flag will have the full symmetry operations.

FLAG_SYMM=1, generate kpoints with spatial symmetry but no time reversal symmetry. This may generate lower symmetry than flag=0. This for example, can be used for magnetic system calculation and for systems with an external magnetic field.

FLAG_SYMM=2, generate kpoints without any symmetry, i.e. the symmetry is identity operation. This for example can be used for rt-TDDFT simulation with external potential.

FLAG_SYMM=3, generate kpoints with time reversal symmetry but no spatial symmetry. This may generate lower symmetry than flag=0. This for example, can be used for rt-TDDFT without magnetic moment.

Special attention needs to be paid for the symmetry operation when magnetic system is calculated, e.g., when calculating antiferromagnetic system, since the symmetry operation might ignore the magnetic moment difference between different atoms.

In above, the symmetry includes both point group symmetry operations and space group symmetry operations. Also, for best point group symmetry, one should always place the high symmetry point at the origin (0,0,0) position, since that is the symmetry operation point.

There are several issues one needs to know. If IN.VEXT is used, it is the user's responsibility to figure out what symmetry one should use, since when the above symmetry operation are generated, it does not consider IN.VEXT. For system with MAGNETIC moment section in atom.config and SPIN=2, the MAGNETIC moment of each atom is also used to figure out the symmetry (e.g., one atom with magnetic moment 2 will be different from another atom with same atomic number, but magnetic moment equals -2).

Notes about IN.KPT, IN.SYMM and MP_N123:

1. Default: IN.KPT=F, IN.SYMM=F, MP_N123= 1 1 1 0 0 0; IN.KPT=T has higher priority then MP_N123;

2. IN.KPT=T: read kpoints from file IN.KPT; write OUT.KPT;

3. IN.KPT=F: use MP_N123 to generate kpoints; write OUT.KPT;

4. IN.SYMM=T and IN.KPT=T: read symmetry operations from file IN.SYMM; write OUT.SYMM;

5. IN.SYMM=T and IN.KPT=F: read symmetry operations from file IN.SYMM, then MP_N123 will use these symmetry operations to generate kpoints; write OUT.SYMM;

6. IN.SYMM=F and IN.KPT=T: no symmetry used; write 'identity' operation to file OUT.SYMM;

7. IN.SYMM=F and IN.KPT=F: use MP_N123 to generate kpoints and symmetry operations; write OUT.KPT and OUT.SYMM;

8. for JOB=MD,NEB,TDDFT,NAMD or SPIN=222: default FLAG_SYMM = 2.

2.1.2.10 SYMM_PREC

SYMM_PREC = distance_tolerance

Default:

SYMM_PREC = 1.d-5

SYMM_PREC is the distance tolerance in Cartesian coordinates to find crystal symmetry.

2.1.2.11 P123

P123 = NP1, NP2, NP3

Related lines: XCFUNCTIONAL = HSE ; ECUTP = XXX.

Default:

Using ECUTP=ECUT to determine NP1, NP2, NP3.

When using HSE method, a small box FFT (with grid NP1,NP2,NP3) can be used to calculate the explicit FOCK exchange integral if only electronic structure is needed. This can significantly speedup the calculation without much loss of accuracy (for electronic structure, molecular dynamics, or TDDFT). Sometime NP1, NP2, NP3 can be as small as half of N1, N2, N3. NP1,2,3 are generated using ECUTP. However, in order to have higher accuracy force and energy, one should used ECUTP=4*ECUT, or usually at least ECUTP=ECUT2 (e.g., for atomic relaxation or phonon mode calculations).

2.1.2.12 SPIN

SPIN = 1 / 2 / 22 / 222

Default:

SPIN=1

SPIN = 1, non-spin-polarized calculation (default). Each orbital will be occupied by 2 electron.

SPIN = 2, spin-polarized calculation, LSDA (magnetization along z axis). For systems except ferromagnetic system, please specify the initial magnetic moment in "atom.config" with the tag section: "MAGNETIC". Note, for SPIN=2, IN/OUT charge density will have: IN.RHO, IN.RHO_2, and OUT.RHO, OUT.RHO_2 (spin up and down components). Similarly, IN/OUT potential will also have spin up and down components: IN.VR, IN.VR_2, OUT.VR, OUT.VR_2.

SPIN = 22, spin-orbit coupling (SOC) calculation, but without magnetic moment. This is suitable for semiconductors like CdSe. In this case, each orbital will have spin-up and spin-down components (spinor). But there are also spin-up orbital and spin-down orbital (each of them has spin-up and spin-down components), and both are occupied. As a result there is no magnetic moment. Since there is no magnetic moment, the charge density and potential will only have one component, or say the spin up and down components are the same. So, there will be IN.RHO,OUT.RHO,IN.VR,OUT.VR, but there will be no IN/OUT.RHO_2, IN/OUT.VR_2 counterparts.

SPIN = 222, spin-orbit coupling calculation, with noncollinear magnetization in generic directions. For SPIN=222, please specify the initial magnetic moment in "atom.config" with the tags "MAGNETIC_XYZ". In this case, the IN/OUT.RHO will also have IN/OUT.RHO_SOM (a complex 2x2 spin matrix density). IN/OUT.VR will also have IN/OUT.VR_SOM (a complex 2x2 spin matrix potential) and IN/OUT.VR_DELTA (a real up-down diagonal potential, note, there is no IN/OUT.RHO_DELTA).

For SPIN=22 and SPIN=222, the SOC pseudopotentials need to be used. Check the pseudopotential sets, choose the proper SOC pseudopotentials. Note, in our calculation, all the SOC comes from the core levels, so only the heavy atoms have the SOC pseudopotentials. We do not calculate the valence band SOC. For light elements like C, N, O, there is no SOC pseudopotential. However, the SOC pseudopotential and non-SOC pseudopotential can be used in mix within a single calculation. **WARNING**: for

SPIN=22 or SPIN=222 ,you must set parameter "ECUT" in etot.input file, and do not use the default value in pseudopotential file.

SOC can also used together with HSE calculations. This is important for topological system calculations.

2.1.2.13 NUM_ELECTRON

NUM_ELECTRON=value

The total number of occupied valence electron in the system. One can use this to make the system charged, or not charged. Note, for charged system calculations, a uniformed back ground charge is used to solve the Possion equation for COULOMB=0. Default value is the value for neutral system.

2.1.2.14 NUM_ELECTRON_SPIN

NUM_ELECTRON_SPIN = NUM_UP NUM_DN

Default:

This parameter has no default settings.

If NUM_ELECTRON_SPIN is explicitly set in etot.input, it will seperately fix the spin-up and spin-down number of electrons to NUM_UP and NUM_DN.

2.1.2.15 NUM_BAND

NUM_BAND=value

Default:

NUM_BAND = min[1.05*NUM_ELECTRON/2+10] (SPIN = 1)

NUM_BAND = min[1.2*min[1.05*NUM_ELECTRON/2+10]] (SPIN = 2)

NUM_BAND = min[1.05*NUM_ELECTRON+10] (SPIN = 22/222)

The number of orbitals to be calculated. When SPIN=2, there are NUM_BAND spin-up orbitals and NUM_BAND spin-down orbitals.

2.1.2.16 RCUT

RCUT = value

Default:

RCUT = max[IN.PSP_RCUT1, IN.PSP_RCUT2, ..., IN.PSP_RCUTi]

The RCUT (in *Bohr* unit, note: $1Bohr = 0.529177 \times 10^{-10}m$) is for the cut off radius for nonlocal pseudopotential implementations. It defines the core radius of the nonlocal part. If RCUT is not specified, PWmat will use the maximum of IN.PSP_RCUTi as the value of Rcut.

2.1.2.17 IN.PSP_RCUT1,2

IN.PSP_RCUT1 = rcut1

IN.PSP_RCUT2 = rcut2

...

IN.PSP_RCUTi = rcuti

The Rcut of each element type. The IN.PSP_RCUTi are in the unit of Bohr, not Amgstron. This provides a way to selectively choose the Rcut for different atoms for the nonlocal pseudopotentials.

If the IN.PSP_RCUTi are not specified, the value of IN.PSP_RCUTi is taken from the pseudopotential file from its "Rcut" value. If these IN.PSP_RCUTi are not provided in the pseudopotential file, the default value will be set to 3.5. Note, to get a smooth force and energy (e.g., for good RELAX convergence), sufficiently large rcuti should be used (e.g., 4.0 Bohr). However, if Ecut2 is very large, then a slightly smaller rcuti can be used.

2.1.2.18 SOM_SPHERE_RCUT

SOM_SPHERE_RCUT = value

Default:

SOM_SPHERE_RCUT = RCUT

SOM_SPHERE_RUCT is used to determine the spin component for each atom. Roughly, it should be half the bond length (in *Angstrom*). **WARNING:The default value is large, and you usually need to set a appropriate value according to the bond length.**

2.1.2.19 NQ123 (obsoleted)

NQ123 = NQ1, NQ2, NQ3

Related lines: XCFUNTIONAL = HSE; JOB=NONSCF.

The NQ1, NQ2, NQ3 are for a NONSCF HSE calculation, and they should be the NK1, NK2, NK3 values from the previous step SCF HSE run in the previous step "MP_N123=NK1, NK2, NK3, SK1, SK2, SK3" statement. This is required for NONSCF HSE calculation. In such a calculation, one previous SCF HSE calculation has been carried out, and the OUT.HSEWR1, OUT.HSEWR2 have been generated. In the previous step SCF HSE calculation, the "MP_N123=NK1, NK2, NK3, SK1, SK2, SK3" has been used. In a continued NONSCF calculation (e.g., to generate the DOS, or bandstructure), the "MP_N123" can be changed (so we need to introduce the NQ1,NQ2,NQ3), or a bandstructure k-point set has been input through IN.KPT. But the original MP_N123 must be input to the program, so it knows what are the k-points in the kernel functions of OUT.HSEWR1, OUT.HSEWR2, ..., OUT.HSEWR(i) in the FOCK exchange integral. This NQ1,NQ2,NQ3 is also used to generate the G=0 compensation term for the Fock exchange integral, following the F. Gygi paper. This NQ123 should not be used for SCF HSE calculation.

2.1.3 Electron tags

2.1.3.1 E_ERROR

E_ERROR = value

Default:

E_ERROR = 2.7211E-6*(total number of electrons)

The error tolerance (convergence criterion) for the total energy (Hartree) in the SCF iterations. The default value is: 2.7211E-6*(total number of electrons) (eV). This is related to the SCF_ITER0, SCF_ITER1 lines. It can terminate the SCF iteration before the maximum steps (NITER0, NITER1) have been reached. Note, since E_ERROR is only determined by one number (the total energy), so it can accidently reach the convergence. It might be dangerous to relied on this error to stop the SCF iteration. To avoid that, one can reduce this value, e.g., to 1.E-8 (eV).

2.1.3.2 RHO_ERROR

RHO_ERROR = value

Default:

RHO_ERROR = 0.5E-4

The error tolerance (convergence criterion) using the SCF iteration difference between the input and output charge density. If the relative error of input and output charge density in one SCF step is less than RHO_ERROR, the SCF iteration will be stopped.

2.1.3.3 RHO_RELATIVE_ERROR

RHO_RELATIVE_ERROR = value

Default:

RHO_RELATIVE_ERROR = 0.0

A variable to control the stopping of the internal CG iterations. This is to estimate the charge density error due to the wave function of CG iteration error. The estimated charge density error should be less than (output-input) SCF charge density error multiplied by RHO_RELATIVE_ERROR, in order to stop the CG steps. Note, this, like WG_ERROR, is to stop the CG iteration (within one SCF step), not the SCF iteration. In contrast, E_ERROR, RHO_ERROR are used to stop the SCF iterations. The old default value is 7.0E-2, smaller this value, more stringent requirement, as a result more likely this is not used. Now the default value is set to 0.0, for more accurate and stable convergence of wavefunction, but will slow down the speed of SCF calculcation,

you can make your own decision to change this parameter.

2.1.3.4 WG_ERROR

WG_ERROR = value

Default:

WG_ERROR = 1.0E-4

The error tolerance (convergence criterion) for the wave function conjugate gradient iterations (Hartree). This is related to SCF_ITER0, SEC_ITER1 lines. It can terminate the CG steps before the NLINE0, NLINE1 have been reached. This is to stop the CG iterations.

2.1.3.5 FORCE_RELATIVE_ERROR

FORCE_RELATIVE_ERROR = value

Default:

FORCE_REVATIVE_ERROR = 0.0 (JOB=RELAX)

FORCE_RELATIVE_ERROR = 0.0 (JOB=MD)

A variable to control the stopping of the SCF iterations during MD and RELAX. This is to estimate the atomic force error due to the charge density error of SCF iterations. The estimated force error should be smaller than the previous MD or RELAX step force multiplied by FORCE_RELATIVE_ERROR in order to stop the SCF iterations. Smaller this value, more accurate SCF is used. The old default value for JOB = RELAX is 0.003; the old default value for JOB=MD is 0.02. For more accurate and stable convergence of force, we change the default value to 0.

2.1.3.6 FERMIDE

Default:

FERMIDE = 0.025

FERMIDE is the same with dE in parameter SCF_ITER, the kT equivalent energy for Fermi-Dirac formula to calculate the electron occupations. Default = 0.025 eV.

2.1.3.7 MIN_SCF_ITER

Default:

MIN_SCF_ITER = 1

This parameter specifies the minimum number of SCF iterations steps. One can set MIN_SCF_ITER to a value between 2 and 8 when JOB = RELAX/MD for a more reliable result.

2.1.3.8 MAX_SCF_ITER

Default:

MAX_SCF_ITER = 100

This parameter specifies the maximum number of SCF iterations steps, i.e. the NITER* in SCF_ITER* lines. One can set MAX_SCF_ITER with value 1 to 1000. The SCF_ITER* lines has higher priority than MAX_SCF_ITER. If the SCF_ITER* lines are not specified, the program will use MAX_SCF_ITER to control the maximum number of SCF iterations steps.

2.1.3.9 SCF_ITER0_1/2/3...

SCF_ITER0_1 = NITER0_1, NLINE0, imth, icmix, dE, Fermi-Dirac
SCF_ITER0_2 = NITER0_2, NLINE0, imth, icmix, dE, Fermi-Dirac
SCF_ITER0_3 = NITER0_3, NLINE0, imth, icmix, dE, Fermi-Dirac

...

Default:

SCF_ITER0_1 = 6 4 3 0.0 0.025 1

SCF_ITER0_2 = 94 4 3 1.0 0.025 1

These variables control the charge density self-consistent iterations for the first SCF run for JOB = SCF, RELAX, MD. For RELAX, MD, the first step SCF run (with the initial atomic positions) uses "SCF_ITER0" lines , and subsequent steps (for moved atomic positions) uses the values of "SCF_ITER1" lines. They are set differently because normally the first run requires much more steps. It is also used for NONSCF run, in

which the ICMIX = 0 for all the NITER0 (NITER0_1+NITER0_2+...) lines in the following. This variable is not used for JOB = DOS.

NITER0: the number of self-consistent (SCF) iterations steps.

$$NITER0 = NITER0\_1 + NITER0\_2 + NITER0\_3 + ... \tag{2.5}$$

The Default value for NITER0 is 100. Note the SCF iteration can be stopped before the NITER0 has been reached if the E_ERROR has been satisfied, or the condition specified by FORCE_RELATIVE_ERROR has been reached. So, the stopping of SCF iteration is controlled by four parameters: **NITER0, E_ERROR, RHO_ERROR, FORCE_RELATIVE_ERROR**, whichever is satisfied first.

NLINE0: the number of CG line minimization steps to solve the wave functions according to $H\psi_i = \varepsilon_i\psi_i$ for a given potential (hence H) at each charge self-consistent step. The default value of NLINE0 is 4. Note, the CG line minimization can be stopped if the error is smaller than WG_ERROR, or the condition specified by RHO_RELATIVE_ERROR is reached. So, the stopping of CG iterations is controlled by three parameters: **NLINE0, WG_ERROR, RHO_RELATIVE_ERROR, whichever is satisfied first**.

IMTH=1, the old band-by-band CG algorithm. It should not be used unless for some special situation.

IMTH=3, the all band conjugate gradient method. This is the default method. We strongly recommend the use of this method.

IMTH=2, the DIIS method. This could be faster than IMTH = 3, but could also have stability problems. It should only be used in SCF iteration steps where the wave function is in some degree converged (e.g., not for random wave functions).

ICMIX=0, no charge mixing and update at this SCF step. In other word, at this step, it is a NONSCF step. For JOB = SCF, RELAX, MD, by default, for the first four SCF steps, ICMIX = 0, and ICMIX = 1 for subsequent steps. For JOB = NONSCF, for all steps, ICMIX = 0.

ICMIX=1, with charge mixing and update for this SCF step. Note, this is a floating point number.

Note: cne can specify something like ICMIX=1.05, as a parameter for Kerker mixing, sometime this can significantly increase the convergence speed. For most cases, ICMIX=1.00 is good enough.

DE: the kT equivalent energy (in eV) for Fermi-Dirac formula to calculate the electron occupations of the eigen wave functions according to their eigen energies ε_i. The default value is $0.025eV$. For semiconductor, especially for defect calculation, $0.025eV$ should be used. However, for metallic system where there are many states near the Fermi energy, one might choose a larger value, e.g., $0.1eV$ or even $0.2eV$.

FERMI-DIRAC: (with possible values: 0, 1, 2, 3, 4, 5). Different formulas for the Fermi-Dirac-equivalent function to calculate the wave function occupation using ε_i and dE. These formulas are: 0, need input external files 'IN.OCC' for SPIN = 1 and 'IN.OCC', 'IN.OCC_2' for SPIN = 2; 1, Fermi-Dirac; 2, Gaussian; 3,4,5 Gaussian with other prefactor polynomials. The default value is 1. However, for metallic systems, one might like to choose 2,3,4, with larger DE values.

Files IN.OCC, IN.OCC_2 format,

```
1.0 1.0 1.0 0.6 0.0 0.0 0.0 ... #occupations for k-point1
1.0 1.0 1.0 0.6 0.0 0.0 0.0 ... #occupations for k-point2
```

2.1.3.10 SCF_ITER1_1/2/3...

SCF_ITER1_1 = NITER1_1, NLINE1, imth, icmix, dE, Fermi-Dirac
SCF_ITER1_2 = NITER1_2, NLINE1, imth, icmix, dE, Fermi-Dirac
SCF_ITER1_3 = NITER1_3, NLINE1, imth, icmix, dE, Fermi-Dirac

...

Default:

SCF_ITER1_1 = 40 4 3 1.0 0.025 1

This is for subsequent SCF calculations for JOB = RELAX, MD, except the first SCF step. It has the same meaning as in SCF_ITER0_1/2/3.... Usually however, the ICMIX is always 1. As default, NITER1=40, NLINE1 = 4, IMTH = 3, ICMIX = 1, DE = 0.025, FERMI-DIRAC=1.

2.1.3.11 SET_OUT_FERMI_POS

SET_OUT_FERMI_POS = T / F 0/1

Default:

SET_OUT_FERMI_POS = F 0

This parameter is used to set fermi energy for calculations of insulator. If SET_OUT_FERMI_POS = T 0, output VBM in file OUT.FERMI as the fermi energy. If SET_OUT_FERMI_POS = T 1, output CBM in file OUT.FERMI as the fermi energy.

2.1.3.12 SCF_MIX

SCF_MIX = CHARGE (default) / POTENTIAL

Default:

SCF_MIX = CHARGE

Related input line: JOB=SCF, or any other calculations using SCF calculations.

The pulay mixing method for the SCF iterations: charge-mixing or potential-mixing.

(WARNING): potential-mixing not support SPIN=222.

2.1.3.13 PULAY_MIX_OPT

PULAY_MIX_OPT = MAX_PULAY_LENGTH, IFLAG_PULAY_WRAP, IFLAG_OUTPUT_PULAY, WEIGHT_Q0_PULAY, PENALTY_AA_PULAY

Default:

PULAY_MIX_OPT = 30 1 0 1.0 0.0

Pulay mixing method will use charge density of previous steps to precondition the input charge density of next SCF iteration. This parameter provides some options to adjust the pulay mixing.

MAX_PULAY_LENGTH is the maximal steps used for pulay mixing.

IFLAG_PULAY_WRAP indicates whether to restart pulay mixing when total pulay mxing step is great than MAX_PULAY_WRAP. When total pulay mixing steps is great than MAX_PULAY_WRAP, if IFLAG_PULAY_WRAP = 0, discards all datas of previous steps and set total mixing step to zero; if IFLAG_PULAY_WRAP = 1, discards the datas of the one most early step, continues to do pulay mixing with the datas of previous MAX_PULAY_LENGTH steps. For heterostructure and low dimensional system with vacuum, MAX_PULAY_LENGTH = 10 and IFLAG_PULAY_WRAP = 0 are recommended for better SCF convergence.

If IFLAG_OUTPUT_PULAY = 1, output move pulay mixing infomation in REPORT. Default is 0.

WEIGHT_Q0_PULAY is the pulay mixing weight for charge density (in G-space) at G=0, Default is 1.0. This weight is useful for FIX_FERMI = T. A possible setting for FIX_FERMI = T:

PULAY_MIX_OPT = 100, 1, 1, 0.0001, 1.0

PENALTY_AA_PULAY is used to add a penalty weight to datas of previous steps, you can change the strength of the penalty by adjust this parameter. Default is 0.0, i.e. no penalty.

2.1.3.14 PULAY_KERK_PARAMETERS

This includes several parameter which can be tuned to improve the SCF converged in Pulay mixing and Kerk mixing:

KERK_AMIN = a_min (default 0.3)

KERK_AMIX = a_mix (default 0.4)

KERK_AMIX_MAG = a_mix_mag (default 0.4)

KERK_BMIX = b (default 0.5)

KERK_BMIX_MAG = b (default 1.e-5)

LDAU_MIX = ldau_mix (default 0.7)

PULAY_WEIGHT_SPIN = pulay_weight_spin (default 1.0)

PULAY_WEIGHT_NS = pulay_weight_ns (default 1.0)

In Kerk mixing, for a given V_{in} and V_{out} pair (comes out from the Pulay mixing), we get a new V_{in} for the next SCF interation as (in G-space):

$$V_{in}(G) = \frac{0.5aG^2 + c}{0.5G^2 + b} V_{out}(G) + (1 - \frac{0.5aG^2 + c}{0.5G^2 + b}) V_{in}(G)$$

here c is from the line: "SCF_ITER0_2 = 100,4,4,1.0x,0.025, 2", here the c=x. By default, c=0. So, one can use b to control the transition from the big G to small G region, and a is used to control the mixing rate at large G, and c is used to control the mixing rate at small G. We found that, some time using none zero c (e.g., 0.02 or 0.05) can accelerate the converges for non metallic system, and sometime even for metallic system. The default values for a and b are usually good enough.

Note, for spin=2, for the Kerk mixing, the charge densities are group into total charge and magnetic charge. The above mixing formula only applies to the total charge. For the magnetic charge, a simple mixing scheme is used, the the mixing parameter is 1 (which means the output magnetic charge density is used as the input for the next iterations). Right now, there is no input parameter to control this magnetic moment mixing.

ldau_mix is used to control the simple mixing parameter for the LDA+U local orbital n occupation number. Simply the $n(new) = ldau\_mix * n_{out} + (1 - ldau\_mix) * n_{in}$, here n is the local orbital occupation number matrix in LDA+U. Note, the default value for this parameter is 0.7, but sometime one can use a large ldau_mix to accelerate the converges, e.g., even ldau_mix=5.

pulay_weight_spin control the weight we used in Pulay mixing for SPIN=2 calculations. In SPIN=2, the charge densities of spin up and down are recombined into total charge density and magnetic density (the difference between up and down). The pulay_weight_spin control what weight we give to the magnetic density when we

carry out pulay mixing. Pulay mixing is done by mixing the previous density in and out pair, trying to reduce the resulting in and out difference. When we judge how large is the in-out difference, we need to use a weight for the total charge part and spin part.

pulay_weight_ns is the weight factor in the pulay mixing for the case of LDA+U to place in the local orbital occupation matrix n, against the charge density.

2.1.3.15 NONLOCAL

NONLOCAL = 1 / 2 / 3

Default:

NONLOCAL = 2

The nonlocal is the nonlocal pseudopotential implementation flag.

NONLOCAL = 1, no nonlocal potential. One has to know what he/she is doing. This usually should never be used, unless for testing purpose.

NONLOCAL = 2, the default, real space nonlocal pseudo potential implementation. It used the mask function method.

NONLOCAL = 3, the g space nonlocal pseudo potential implementation.

2.1.3.16 USE_PWSCF_INTE_METHOD

USE_PWSCF_INTE_METHOD = T / F

Default:

USE_PWSCF_INTE_METHOD = F

Whether or not to use the PWSCF method to do the 1D vloc potential integrations.

USE_PWSCF_INTE_METHOD=T, use Simpson method and 10 Bohr radius cutoff. When using this, PWmat will have the exact same energy results with PWSCF.

USE_PWSCF_INTE_METHOD=F (default), use smooth radius cutoff (smaller cutoff). This is the origin PWmat integration method. We believe it is more accurate (although the result is slightly different from that of PWscf.

2.1.3.17 SYS_TYPE (obsoleted)

SYS_TYPE = 1 / 2

Default:

SYS_TYPE = 1

This parameter helps to identify whether the system is semiconductor, insulator or metallic. If the sys_type = 1, the system will be semiconductor or insulator(default). The smearing width (the "dE" in SCF_ITER0/1 line) is 0.025; If the sys_type = 2, the system will be metallic. The "dE" will be 0.2.

2.1.3.18 NMAP_MAX (obsoleted)

NMAP_MAX = num

Default:

NMAP_MAX = 50000

If you encounter the error "nmap_tot too large, reduce rcut, grid density,or increase NMAP_MAX stop", you can try to increase the NMAP_MAX.

2.1.4 Exchange-correlation tags

2.1.4.1 XCFUNCTIONAL

XCFUNCTIONAL=LDA/PBE/HSE/PBESOL/PW91/TPSS/SCAN.

If you want to use other functional from LIBXC, you can set xcfunctional as following:

XCFUNCTIONAL=XC_LDA_X+XC_LDA_C_PZ

OR

XCFUNCTIONAL=XC_GGA_C_PBE+XC_GGA_X_PBE

OR

XCFUNCTIONAL=XC_HYB_GGA_XC_B3LYP

OR

XCFUNCTIONAL=XC_MGGA_C_TPSS+XC_MGGA_X_TPSS

OR

XCFUNCTIONAL= SCAN + rVV10

OR

XCFUNCTIONAL= LDA/PBE/HSE + rVV10

OR

XCFUNCTIONAL= LDAWKM/LDAWKM2/LDAWKM3

The parameter is case insensitive. You can refer to [21] and [22] for more information about the parameters of LIBXC.

Default:

XCFUNCTIONAL = PBE

XCFUNCTIONAL is used to Control the exchange-correlation functional. PWmat supports the LIBXC library for its LDA/GGA/METAGGA functional. The most commonly used functional include: LDA, PBE, HSE, PBESOL, PW91, TPSS etc. If you want to use other functional from LIBXC, you can set xcfunctional as indicated about.

If XCFUNCTIONAL = HSE, PWmat will do HSE calculation. One could use the optional parameter: HSE_OMEGA, HSE_ALPHA. Their default values are 0.2, 0.25 (the HSE06). Note PBE0 (the long range exchange integral) is just a special form of HSE with HES_OMEGA parameter close to zero (see the section HSE_OMEGA, HSE_ALPHA).

For hybrid-functionals in libxc except HSE and B3LYP, you should set HSE_ALPHA and HSE_BETA that consistent with your chosen xc functional. PWmat will set default HSE_ALPHA and HSE_BETA for B3LYP and HSE, but not for others.

If there are rVV10, one can optionally use RVV10_DETAIL to specific RVV10 parameters.

The LDAWKM, LDAWKM2, LDAWKM3 (the same for PBEWKM,PBEWKM2,PBEWKM3) options are really WKM calculations (together with JOB=SCF). In particular, what it is doing is the following: For LDAWKM:

$$H = H_{LDA} + \sum_k \lambda_k s_k (1 - s_k)$$

and for LDAWKM2:

$$H = H_{LDA} + \sum_k \lambda_k s_k$$

Here s_k are the occupation number of Wannier functions ϕ_k defined as:

$$S(k1, k2) = \sum_j < \psi_j | \phi_{k1} >< \phi_{k2} | \psi_j > occ(j)$$

Here $occ(j)$ is the occupation of the Kohn=Sham wave function ψ_j. Thus: $s_k = S(k, k)$.

The parameters λ_k are input from the file IN.WANNIER_PARAM. The Wannier functions in spin-up channel and spin-down channel are input from files: IN.WANNIER_k.u, IN.WANNIER_k.d respectively. You can check 2.4.12 for details.

For the LDAWKM, LDAWKM2 calculations, besides the usually results shown in REPORT, there are also results in OUT.WANNIER_S (the diagonal result of s_k) and OUT.WANNIER_SS (the full S(k1,k2) matrix) for all the Wannier functions.

Note, LDAWKM3 is like the LDAWKM2 (same for PBEWKM3 and PBEWKM2), but instead of input the real space localized wave functions, in LDAWKM3, a reciprocal space Bloch wave function (the IN.WG style) at different kpoints are provided. In that case, the ϕ_k is provided by a IN.WG0 (and IN.WG0_2) (which could be the wave function from previous calculations), with the same kpoints as the kpoints of the current calculations. The λ_k is provided in the input file: IN.WANNIER_PARAM3 (the IN.WANNIER_PARAM is no longer used). The λ_k indicates the shifts projected on different ϕ_k (using LDAWKM2 formula above). Please check You can check 2.4.12 for the formate of IN.WANNIER_PARAM3. Note, use LDAWKM3, one can artificially modify the band structure (shift the band), e.g., for a MD or TDDFT calculations.

2.1.4.2 HSE_DETAIL

HSE_DETAIL = HSE_MIX, MAX_SXP, TOLHSE_MIX, HSE_DN, HSE_PBE_SCF, CHECK_DSXH

Default:

HSE_DETAIL = 1, 1, 0.0, 6, 1, 1

This is an optional choice when functional=HSE. It specifies all the details for a HSE SCF calculation.

Besides these parameters, there are other parameter which can also affect the convergence of the HSE atomic relaxation or phonon calculation, in particular the Ecutp parameter. Ecutp=4Ecut might be needed to provide sufficient accurate force. However, since increase Ecutp can significantly increase the computational time, one only use this option is Ecutp=Ecut (the default) has some problems. See the section for Ecutp.

HSE_MIX: A real number, describe the Fock exchange kernel mixing parameter during SCF calculation (no, not confuse this with the HSE alpha parameter in the functional itself). It could be larger than one (e.g., 1.2). This is a bit like the charge mixing factor. Default is 1. Recommend 1 for most cases. If it is too large, it can blow up the convergence.

MAX_SXP: Maximum number of Fock exchange kernel mixing terms. Numbers larger than one (e,g., 2, 3) can speed-up the HSE convergence. But each increase will cost one extra memory usage at the size of a wave function. This is like the length of Pulay mixing for charge mixing algorithm. But it is for the Fock exchange term. The default value is 1 (no Pulay mixing).

TOLHSE_MIX: The tolerance for the Fock exchange term mixing for the HSE SCF iteration to stop. The default value is 0.d0. One can also use value 1.E-03

Note, in the output (screen printing), REPORT , there is one line:

```
update_sxp(err)(eV) = 0.2222E-02, 0.1111E-02
```

This item (UPDATE_SXP) correspond to the TOLHSE_MIX value. The first value (0.2222E-02) represents the actual Fock exchange term changes (error) after the Fock exchange term 'sxp' has been updated. The second value (0.1111-02) represent the predicted value (error) after doing Fock exchange pulay mixing when MAX_SXP > 1.

HSE_DN: The number of SCF steps for each Fock exchange kernel update. Default value is 6. Recommend 3 to 10. In our algorithm, each Fock exchange kernel update

is followed by HSE_DN steps of the SCF step (without updating the Fock exchange kernel). This HSE_DN steps have the cost of PBE to run for each step, thus it is relatively cheap. Since each Fock exchange kernel update is expensive, this parameter is important. One wants the HSE_DN SCF step to converge the charge density etc. following each Fock exchange kernel update. But too large HSE_DN might not be so helpful and can still be costly to run. So, one wants to choose this parameter carefully. If one finds the HSE calculation does not converge, one might want to increase HSE_DN. Note, the total number of SCF steps specified in SCF_ITER0_X, SCF_ITER1_X counts both the SCF update step and the Fock excahnge update steps.

HSE_PBE_SCF: The parameter define, for HSE SCF calculation, whether one wants to do one PBE calculation first. The default is 1 (do PBE SCF calculation first). If it is set to 0, that means doing the HSE calculation first, without doing a pre-PBE calculation.

CHECK_DSXH: The parameter define, for HSE SCF calculation, whether check the convergency of ACE projector. The check may fail that SCF can not get the demanded accuracy(check REPORT file:ending_scf_reason = d_sxp.gt.d_sxp_laststep), then one can try CHECK_DSXH=0. The default is 1.

Overall, we recommend to use HSE_MIX=1, MAX_SXP=1, TOLHSE_MIX=0.0, HSE_DN=3-6, HSE_PBE_SCF=1. If there are problems to converge the HSE SCF, one might want to consider MAX_SXP=2,3, and increase HSE_DN.

2.1.4.3 HSE_OMEGA

HSE_OMEGA = ω

Default:

HSE_OMEGA = 0.2

Related input line: XCFUNCTIONAL = HSE

The screening parameter for HSE like hybrid functionals. Refer to J. Chem. Phys. 118, 8207 (2003) and and J. Chem. Phys. 124, 219906 (2006) for more information. PWmat support range separated hybrid functional calculations. It separate the exchange

integration into long range and short range. This parameter is used to provide the range, used in the way of ωr. The default value is 0.2 1/Å. So, bigger this value, shorter the cut-off to distinguish the short range and long range.

2.1.4.4 HSE_ALPHA

HSE_ALPHA = α

Default:

HSE_ALPHA = 0.25

Related input line: XCFUNCTIONAL=HSE

The mixing parameter of the explicit short range Fock exchange part. The default is 0.25. Combined HSE_OMEGA, the exchange correlation energy is: $E_{xc} = \alpha * E_x(FOCK, \omega) - \alpha * E_x(PBE, \omega) + E_x(PBE) + E_c$. Here $E_x(FOCK, \omega)$ is the explicit short range Fock exchange integral with the Coulomb integration truncated by ω, E_x(PBE) is the PBE exchange density functional energy, $E_x(PBE, \omega)$ is the short range PBE exchange density functional energy with the range defined by ω.

2.1.4.5 HSE_BETA

HSE_BETA = β

Default:

HSE_BETA = 0.0

Related input line: XCFUNCTIONAL=HSE

The mixing parameter of the explicit long range Fock exchange part. The default is 0.0. PWmat provides support for a separated range hybrid calculations (all to be called HSE). The default β is zero, thus no long range part. But one can also input a nonzero β. Sometime it is argued that, to get the correct optical properties, one should use 1/dielectric-constant as β for bulk systems.

When β is nonzero, the final XC functional is: $E_{xc} = \alpha * E_x(FOCK, \omega) + \beta * (E_x(FOCK, full) - E_x(FOCK, \omega)) + E_x(PBE) - \alpha * E_x(PBE, \omega) - \beta * (E_x(PBE) - E_X(PBE, \omega)) + E_c$. Here $E_x(FOCK, \omega)$ is the explicit short range Fock exchange

integral with the Coulomb integration truncated by ω, $E_x(FOCK, full)$ is the full explicit exchange integral, E_x(PBE) is the PBE exchange density functional energy, $E_x(PBE, \omega)$ is the short range PBE exchange density functional energy with the range defined by ω.

It has been argued that such a range separated HSE function can be used to replacing more advanced method like GW to provide approximated electronic structures.

2.1.4.6 HSEMASK_PSP

The format of the parameter is:

HSEMASK_PSP1 = ampl1 size1

HSEMASK_PSP2 = ampl2 size2

This is a special option for HSE, for cases where one wants to use different HSE mixing parameters for different regions (e.g., atoms) in heterostructural calculations.Since the HSE mixing parameter is very much an empirical parameter, in order to get the correct band gap at different regions (e.g., in a Si/SiO2 heterostructure), one might want to use different mixing parameters for these regions [30]. The parameter provided here can accomplish that. note: if you use the same pseudo for different parameter, Si for example, you can make a copy of the Si.SG15.PBE.UPF and rename it as Ge.SG15.PBE.UPF, change the element line Si to Ge in the UPF file, if you want to do MD or TDDFT calculations, plesae add an extra mass line under the element line in the pseudo UPF file, etc. mass="28.085" , if you don't do this, the software will use the mass of element Ge, that will make error result. These are parameters used to provide an element specified HSE mixing parameter HSE_ALPHA.

Related input line: XCFUNTIONAL=HSE.

The size1, size2... are in Bohr unit. The parameter can adjust the gap of HSE calculation with different setting for different atomic types.

The explicit Fock exchange integral (including the mixing parameter) is:

$$E_x(Fock) = \alpha \sum_i \sum_j o(i)o(j) \int\int \psi_i^*(r)\psi_j(r)m(r)\frac{erfc(|r-r'|\omega)}{|r-r'|}m(r')\psi_i(r')\psi_j^*(r')d^3rd^3r'$$

Here the o(i) is the orbital i occupation function, and mask function m(r) is: $m(r) = 1 + \sum_R ampl_R exp(-(r-R)^2/size_R^2)$, here ampl_R and size_R are the parameters input from HSEMASK_PSP1,etc.

Note, the effective HSE_ALPHA is the default HSE_ALPHA multiplied by m(r)^2, which is then determined by ampl1, ampl2, etc. The ampl1 itself is not HSE_ALPHA.

2.1.4.7 HSE_KPT_TREATMENT

The format of the parameter is:

HSE_KPT_TREATMENT = flag_double_grid flag_vq_interp vq_interp_range

This is a special option for HSE, especially for JOB=NONSCF calculations for HSE bandstructures. In HSE calculation, when bandstructure is calculated, new kpoints are provided from IN.KPT, and the wave functions from JOB=SCF calculation on a MP kpoint grid k_prime=(NQ1,NQ2,NQ3) are used for the Fock exchange integral kernel. However, since the new kpoints k from the IN.KPT are not in the k_prime grid, some special treatments are needed to deal with this. We provided two options for this. The default option is flag_double_grid=1, flag_vq_interp=0. The default value should be good for most cases. flag_double_grid=1 means a special treatment for q=k-k_prime+G. A double grid is defined on top of (within) the (NQ1,NQ2,NQ3) grid (i.e, the grid consists of lines 0, 2, 4,.. etc), the Fock exchange contribution of the k_prime for its q on the double grid will be deleted (its corresponding Coulomb kernetl vq(q) is set to zero), while multipling the contribution from the other k_prime points by a factor of 8/7. This can be useful, e.g., to have the correct symmetry, and make sure X point result is the same as the -X point result. Note, for JOB=SCF, one can choose flag_double_grid=1 or 0, although we do recommend 1 for SCF. flag_vq_interp=0

means the vq interpolation is not used (vq_interp_range is not used in this case). For JOB=SCF, we can only use flag_vq_iinterp=0. In this case, a shifting of k_prime is used. Basically, the 8 k_prime points on the cube enclosing k are shifted to k, one at a time, so to make k points on the k_prime grid. The 8 shifting are then linearly added using linear interpolation weights. Note, only the kernel vq are averaged, so there is no extra FFTs. We strongly recommend this option for JOB=NONSCF for most cases. However, due to the linear interpolation, on the very small energy scale, one can have a small linear kink on the bandstructure, especially at the high symmetry point. Here, we provide another option: flag_double_grid=0 and flag_vq_interp=1. In this case, the k_prime points are not shifted, and vq are calculated using the formula, but for q close to zero, within a cutoff qcut=dq*vq_interp_range (dq is the NQ1,NQ2,NQ3 grid interval size), the calculated vq (which is kind of diverging for q close to zero) is switched (using a cos function) into the Gygi's formula exxvq term. The vq_interp_range (usually around 1) can be used to control the range of this switching (morphing). Larger the range, smoother will be the bandstructure. Note, since we are using flag_double_grid=0, the energy of k point even when k=k_prime might slightly different from the SCF value when this option is used. But when flag_vq_interp=0 is used, on the k_prime point, the JOB=NONSCF bandstructure value is exactly the same as the JOB=SCF result.

2.1.5 JOB-related tags

2.1.5.1 DOS_DETAIL

Format: DOS_DETAIL = IDOS_interp, NQ1, NQ2, NQ3

Default: DOS_DETAIL = 0 IDOS_interp=0 : it will not use the k-point interpolation scheme for the JOB=DOS calculation.

IDOS_interp=1 : it will use the standard interpolation scheme and generate OUT.overlap_uk.

IDOS_interp=2 : it will use the second order interpolation scheme and generate OUT.overlap_uk.2.

When IDOS_interp=1 or 2, the interpolation borrows the method used in HSE calculation, the wave functions are first FFT into real space, then the overlap between different k-points are done. For very large systems and many k-points, it might run out of memory, one can try a smaller ECUTP / P123 to reduce the memory requirement.

NQ1, NQ2, NQ3 : must equal to the MP_N123 in the last SCF or NONSCF run which generated the wave functions OUT.WG (used as IN.WG in the current DOS run).

(WARNING): *When IDOS_interp = 1 / 2, kpoint parallelization is not allowed.*

(WARNING): *When IDOS_interp = 2, NQ1,NQ2, NQ3 needs to be greater than or equal to 4.*

2.1.5.2 NUM_DOS_GRID

NUM_DOS_GRID = num

Default:

NUM_DOS_GRID = 4000

This is the number of energy grid points when calculating DOS, the default is 1500. This grid is within the window [E_{min}, E_{max}], and the E_{min}, E_{max} are determined by the minimum and maximum eigen energies.

2.1.5.3 DOS_GAUSSIAN_BROADENING

DOS_GAUSSIAN_BROADENING = value

Default:

DOS_GAUSSIAN_BROADENING = 0.05

This is the gaussian broadening of DOS plot for JOB=DOS, unit is eV.

2.1.5.4 RELAX_DETAIL

Format: RELAX_DETAIL = IMTH, NSTEP, FORCE_TOL, ISTRESS, TOL_STRESS, TOL_LINECORRECTION

Default:

RELAX_DETAIL = 1, 200, 0.02, 0, 0, -0.001 (ACCURACY = NORM)

RELAX_DETAIL = 1, 200, 0.01, 0, 0, -0.001 (ACCURACY = HIGH / VERYHIGH)

RELAX_DETAIL = 1, 200, 0.03, 0, 0, -0.001 (XCFUNCTIONAL = HSE)

This is an optional line for "JOB = RELAX". It controls the atomic relaxation steps. Note PWmat1.5+ can relax the lattice vectors.

IMTH indicate the method of relaxation.

IMTH = 1 / 2 / 3 / 4 / 5 / 6

1. IMTH=1(default), conjugated gradient;
2. IMTH=2, BFGS method;
3. IMTH=3, steepest decent (this is mostly for JOB=NEB);
4. IMTH=4, Preconditioned Conjugate Gradient (PCG), experimental feature, See 2.1.5.6 before you use this;
5. IMTH=5, Limited-memory BFGS method;
6. IMTH=6, FIRE: Fast Inertial Relaxation Engine.

Please note that: all imths (1,2,3,4,5,6) can be used for atomic relaxation, but only imth=1,5,6 can be used for cell relaxation. In addition, some options can be set for the optimizers in file IN.RELAXOPT(need to set IN.RELAXOPT=T in etot.input), see section 2.4.6 for more details.

NSTEP is the maximum number of relaxation steps (each total energy calculation is one step, i.e., it counts the steps inside the line minimization in the total steps. In another word, NSTEP is more (at least twice) than the CG steps).

FORCE_TOL (in $eV/$Å) is the force tolerance for the maximal residual force. If the maximum force is less than FORCE_TOL, the relaxation will stop.

ISTRESS controls whether to relax the lattice vectors. If ISTRESS=0 (or the last two number do not exist), the lattice will not be relaxed. If ISTRESS=1, PWmat will relax lattice vectors. One can add external stress tensor by setting STRESS_EXTERNAL

or PTENSOR_EXTERNAL in file atom.config, and external pressure by setting PSTRESS_EXTERNAL in file IN.RELAXOPT, the latter need to set IN.RELAXOPT=T in etot.input.

If you have set STRESS_EXTERNAL or PTENSOR_EXTERNAL, make sure the settings are consistent with sysmetry operations you have used (IN.SYMM=T) or generated by MP_N123, if not you should turn off the symmetry operations. Check MP_N123 for details about symmetry.

If ISTRESS=1 and XCFUNCTIONAL=HSE, one should set RELAX_HSE as follows:

RELAX_HSE = 0 0.0 2

Basic settings for cell relax with xcfunctional=pbe

```
JOB = RELAX
XCFUNCTIONAL = PBE
RELAX_DETAIL = 1 1000 0.02 1 0.05
ECUT = 70 # maybe 1.4 * Ecut_default
ECUT2 = 280
```

Basic settings for cell relax with xcfunctional=hse

```
JOB = RELAX
XCFUNCTIONAL = HSE
RELAX_DETAIL = 1 1000 0.03 1 0.05
ECUT = 70 # maybe 1.4 * Ecut_default
ECUT2 = 280
RELAX_HSE    =    0   0.50000E-01     2
```

TOL_STRESS (in $eV/Natom$) is the stress tolerance for the maximal residual stress. Natom is the total number of atoms. (here it is defined as $\partial Etot/\partial STRAIN/Natom$, Etot is the energy of the whole system (not the energy of unit volume)).

TOL_LINECORRECTION (in eV) is the energy tolerance for the line minimization alone one search direction. When we see Etot(step) becomes a linear line, perhaps we should use a smaller value to have more correction steps. However, for more accurate relaxations, we can set a smaller value for this parameter, so the relaxation can continue.

We can turn off the energy tolerance checking for the line minimization by setting TOL_LINECORRECTION < 0. Some time energy is not that much accurate, the energy tolerance checking is not reliable. And the default value is -0.001.

The JOB = RELAX will output a RELAXSTEPS and MOVEMENT files. While RELAXSTEPS gives a summary of the steps, MOVEMENT records the atomic positions and lattice vectors for all the steps.

SPECIAL FORCE ON EACH ATOM:

In the RELAX calculation, one can add atom specified external force on each atom. This is done by using IN.EXT_FORCE=T in etot.input. In that case, A file called IN.EXT_FORCE needs to be provided, it will give the external force on each atom during the atomic relaxation. Please see the section IN.EXT_FORCE for more details.

SOME DISCUSSION ABOUT RELAXATION:

Atomic relaxation is one of the most used feature in DFT calculations. One has to balance the speed with the stability. Here, we have implemented 6 different methods. For most common problem, we suggest to use imth=1 (conjugate gradient,CG). For very large system, to accelerate the convergence, one can test the use of imth=4, which is the VFF accelerated relaxation. In the best case (e.g., very large systems), imth=4 can speed up imth=1 by a factor of 10. However, one might want to test its stability. One can also used imth=4 (the BFGS method), sometime it is faster than the CG method. Another new method is imth=6, it uses a molecular dynamics but with a friction term, so the system will eventually relax to a local minimum. This can be stable, but one needs to test the parameter FIRE_DT in IN.RELAXOPT. Lastely, if all these methods are unstable, one can always use imth=3, and use a very small maximum step by setting RELAX_MAXMOVE in IN.RELAXOPT. This might take a long time, but it should be stable if sufficient small RELAX_MAXMOVE is used.

When doing atomic relaxation, one must be mindful of the egghead problem, which is the artificial forces caused by the real space lattice. One can remove this problem by setting Ecut2=4Ecut, and Ecut2L=4Ecut2. Most likely, the second condition Ecut2L=4Ecut2 might not be necessary, or one can use JOB=EGGFIT

and EGG_DETAIL to remove the egghead effect without using Ecut2L=4Ecut2. Unfortunately, one might always need to use Ecut2=4Ecut.

To check the relaxation convergence, one should always check the energy as a function of iteration steps in RELAXSTEPS. Note, that, some steps might be trial steps, so they are not so important (you might see some spike). The important one is the "NEW" step energy.

If funcitonal=HSE is used for atomic relaxation, some special algorithms are used for its acceleration. Please check RELAX_HSE for details.

Finally, if cell lattices are relaxed, great care must be taken. During the lattice relaxation, the number of plane wave basis (the G-vectors within the Ecut sphere) is not changed. So, at the end of the relaxation, the plane wave set (it might no longer be a sphere) might not correspond to the Ecut sphere. So, if one uses the Ecut to run it again, a new plane wave set based on the Ecut sphere will be chosen, and the energy and the stress might be changed. So, either one needs to do this multiple times, or one needs to use a rather large Ecut, so the change of basis will have minimum effect. One can use the STRESS_CORR to mitigate (compensate) this problem in some degree, but cannot really remove it completely. This will be particularly problematic if the volume change is very large. In some cases, it might be more reliable to do the cell relaxation by hand, specially if only one degree of freedom is used. Note, one can use stress_mask in the atom.config to choose what lattice components can be changed. One can also used imov in atom.config to determine which atom, and which x,y,z direction can be moved for internal atomic relaxation.

2.1.5.5 RELAX_HSE

RELAX_HSE=NUM_LDA, FACT_HSELDA, LDA_PBE

Default:

RELAX_HSE = 20, 0.05, 2

This is an optional line for "JOB = RELAX" when XCFUNTIONAL=HSE. It uses special techniques to accelerate the atomic relaxation under HSE. Currently, this option

only works for conjugated gradient atomic relaxation as defined in RELAX_DETAILS. In this option, the additional LDA or PBE atomic relaxations are used as preconditioner for the HSE relaxation.

NUM_LDA is the maximum number of relaxation steps for the LDA/PBE preconditioner run. It uses the LDA/PBE atomic relaxation as a preconditioner for the HSE atomic relaxation. If NUM_LDA=0, then no precondition is used, it is the plain CG atomic relaxation based on HSE. The default is 20.

FACT_HSELDA is the prefactor to stop the LDA/PBE relaxation: if LDA/PBE force is less than FACT_HSELDA multiplied the HSE force, then stop. The default is 0.05.

LDA_PBE is the indicator for LDA or PBE functional used for the atomic relaxation to find the preconditioner of HSE relaxation. If LDA_PBE=1, use LDA; LDA_PBE=2, use PBE. One should use the xcfunctional which is closest to HSE. The default value is 2.

Comments: before one use this option (NUM_LDA >0), one better make sure the PBE, or LDA atomic relaxation of the system is smooth. So, one might want to use Ecut2=4*Ecut.

2.1.5.6 VFF_DETAIL

Format: VFF_DETAIL = FF_IMTH, FF_NSTEP, FF_FORCE_TOL, K_BOND, K_ANGLE, K_DIHEDRAL, K_SHIFT

Default: VFF_DETAIL = 1, 500, 0.01, 30.0, 5.0, 0.0, 1.5

Note, if you want to use PCG(IMTH=4) method to accelerate the relaxation, you should know some basic concepts about force field theory. The PCG method supports any systems such as molecular, metallic, semiconductor and it has a well optimized energy function to the metallic systems like Al, Ni, Au, Cu, Ag, Pt, Ir, Pd, Rh, La, Ce, Mg, Ca, Sr. This method has a high stability for molecular systems, semiconductors and gives high speedup factors for molecule absorbed on surface cases.

FF_IMTH: the optimization algorithm used in force field relaxation.

FF_NSTEP: the maximum optimized steps in force field relaxation.

FF_FORCE_TOL: the tolerance of force in force field relaxation.

K_BOND, K_ANGLE, K_DIHEDRAL, K_SHIFT: the force constant for bond, angle, dihedral, shift terms.

$$E_{tot} = k_b(b - b_0)^2 + k_a(\theta - \theta_0)^2 + k_d(\phi - \phi_0)^2 + k_s(r - r_0)^2 \tag{2.6}$$

The defaults is **VFF_DETAIL = 1, 500, 0.01, 30.0, 5.0, 0.0, 1.5**. Note, it is not recommended to modify these parameters unless you are an expert on force field. If there is a metallic area in your system, that is, some metallic atoms have only metallic atoms neighbors and only metallic bond, you have to set additional parameters in the 'atom.config' file like this:

```
29    0.22    0.11    0.06   1  1  1  1  1
 1    0.44    0.49    0.39   1  1  1  1  0
```

Here the Copper atom(Z=29) has five integers after coordinates, first three integers determine whether the atom will move along particular direction, the forth integer number has some meaning for DOS calculation and the last integer number (the ninth column) determine whether the atom is in the metallic area. As we know, the hydrogen atom is not a metallic atom, we set the last integer as 0, while Copper is metallic, so we have set it to 1. Note, you can set Copper to 0, so it will be dealt as a covalent bond element. Usually we only set it to 1 when there is a large piece of metal.

2.1.5.7 MD_DETAIL

Format: MD_DETAIL = MD, MSTEP, DT, TEMP1, TEMP2

Default: None

This line is required when JOB=MD, or JOB=TDDFT, or JOB=NAMD. There is no default values, hence must be input by hand.

MD: the method of MD algorithm

MD = 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 11 / 22 / 33 / 44 / 55 / 66 / 77 / 88 / 100 / 101 (if MD > 10, that means the continue run of MD following the previous runs.)

1. MD = 1, Verlet (NVE) [4];
2. MD = 2, Nose-Hoover (NVT) [6, 7];
3. MD = 3, Langevin (NVT) [5];
4. MD = 4, Constant pressure Langevin dynamics (NPT) [10, 11];
5. MD = 5, Constant pressure Nose-Hoover dynamics (NPT) [12];
6. MD = 6, Berendsen dynamics (NVT) [8];
7. MD = 7, Constant pressure Berendsen dynamics (NPT) [9];
8. MD = 8, Multi-Scale Shock Technique (MSST) [14].
9. MD = 100, 101, calculating multiple configurations stored in IN.MOVEMENT

Verlet is for NVE (fixed number of atom N, fixed volume V, and fixed total energy E), Langevin and Nose-Hoover are for NVT (fixed number of atom N, fixed volume, and fixed temperature T), and Constant pressure Langevin or Nose-Hoover dynamics are for NPT (fixed number of atom N, fixed pressure P, and fixed temperature T). Currently, we do not have NPE. We also provide MSST (Multi-Scale Shock Technique) to simulate a compressive shock wave passing over the system.

One can also set: MD=11,22,33,44,55,66,77,88 which means the continue run of MD following the previous runs.

MSTEP: the number of MD steps.

DT: the time length for each MD step (in the unit of fs, $1fs = 1 \times 10^{-15}s$). Note, usually, with H atoms, dt should be $1fs$, and with heavier atoms, dt could be $2fs$. However, for rt-TDDFT run, dt should be much smaller, like 0.1 fs to 0.2 fs.

TEMP1: the beginning temperature (in $Kelvin$). When there is no velocity session in the atom.config file, the TEMP1 will be used to randomly generated an initial velocity

(the initial kinetic energy is generated as twice the 0.5*K*T, with the expectation that half of its energy will be converted into potential energy). So, in the simulation, the istep=1 temperature will be 2*TEMP1.

TEMP2: the final temperature (in $Kelvin$). TEMP2 will not be used for MD=1, or 11 (NVE) (but still, it should be there as a place holder). During the MD, the program will adjust the temperature linearly, let it goes from TEMP1 to TEMP2. For MD=3,5 (Langevin), one can use a LANGEVIN_ATOMFACT_TG section in the atom.config file to specify a local atomic specified temperature (and Langevin parameter gamma). In that case, the desired atomic temperature at a given time equals the global desired temperature calculated from TEMP1 to TEMP2, then multiplied by the atomic scaling factor specified in the LANGEVIN_ATOMFACT_TG.

For method 1-8, one can use file IN.MDOPT to set detailed parameters by setting IN.MDOPT=T, all the parameters will be written in file OUT.MDOPT. Please refer to section 2.4.7 for details.

In the Berendsen method (MD=6,7,66,77), the kinetic energy is scaled at every MD step as (1+(Tdesired/Tcurrent-1)*dt/tau). For MD=7,77 (NPT), the cell box is scaled at every MD step as: cell(i1,i2) = cell(i1,i2) * (1+(press(i1,i2) - press_ext(i1,i2)) * stress_mask(i1,i2) * dt / tauP). Here press, press_ext are pressures in the unit of $eV/Angstrom^3$. Note, press(3,3) equals the stress(3,3)/volume, so it is a 3x3 tensor. The external pressure press_ext(i,j)=delta_i,j MD_NPT_PEXT_XYZ(i). Note, this is input from IN.MDOPT, not the stress from the atom.config. If MD_NPT_PEXT_XYZ is not specified, then MD_NPT_PEXT_XYZ(:) = MD_NPT_PEXT. If both MD_NPT_PEXT_XYZ and MD_NPT_PEXT are not speficied, then the external pressure is zero. The stress_mask is from atom.config. The detault value is stress_mask(i1,i2)=1 (for every element of the matrix).

One can always set MD_SEED and MD_AVET_TIMEINTERVAL. For MD=4 or 5, one can check the internal pressure in file MDSTEPS and MOVEMENT.

In the MSST method(MD=8,88), you always need to set MD_TAU_CELL, MD_MSST_VS, MD_MSST_DIR. MD_TAU_CELL will set the masslike parameter

for the simulation cell size, i.e. Q in paper [14], but MD_TAU_CELL itself is the time to arrive equilibrium. MD_MSST_VS is the shock speed v_s in paper [14]. MD_MSST_DIR is the shock direction, its value can be 0, 1 or 2 for in x, y or z direction. IN file MOVEMENTS you can check MD_MSST_INFO for additional outputs. IN file MDSTEPS there adds a new column "V/V0", which shows the change of vulome. In the process of MSST, if MD_MSST_VS is kind of large in some way and the size of box changes alot, PWmat will hard to converge or evan crash. You need to reduce the time step, i.e. use smaller DT in MD_DETAIL.

When MD=11/22/33/44/55/66/77/88, it is a continue run for Verlet/Langevin/Nose-Hoover constant pressure, Langevin/constant pressure Nose-HooverNPT,Berendsen NVT, Berendsen NPT respectively. In these cases, the atom.config file should include the velocity section. Note, for JOB=MD, if there is velocity section in atom.config, the velocity will be used, and there is no initial scaling of the velocity using temperature TEMP1.

SPECIAL LV: There is a special feature for LV dynamics (either 3 or 4). In this special feature, we can specify the desired temperature for each atom. We can also specify the desired GAMMA value (the MD_LV_GAMMA) for each atom. In another words, you can make one atom very hot, and let the temperature decaying from this atom. The temperatue decaying length will be controlled by the MD_LV_GAMMA. Smaller this value, decaying length will be longer, i.e., more graduate. The atom specific temperature and Gamma are controlled by scaling factors, they are specified in the atom.config file, with a special session called: "LANGEVIN_ATOMFACT_TG". They have the following formats (see section 2.2):

```
LANGEVIN_ATOMFACT_TG
30  0.5   1.0
30  0.2   1.2
..............
atom scaleT  scaleG
```

Here the desired temperature for one atom equals the original desired temperature specified by temp1, temp2 and the steps, then multiplied by scaleT(iatom). The Gamma

for one atom equals the MD_LV_GAMMA specified in the above table, or its default value, multipled by scaleG(iatom).

SPECIAL FORCE: In all the MD calculation, one can add atom specified external force on each atom. This is done by using IN.EXT_FORCE=T in etot.input. In that case, A file called IN.EXT_FORCE needs to be provided, it will give the external force on each atom during the molecular dynamics, or during atomic relaxation. Please see the section IN.EXT_FORCE for more details.

MD=100,101: for these choices, instead of doing an actual MD simulation following the atomic forces, the multiple configurations stored in a file IN.MOVEMENT will be calculated one after another, so the forces will not really be used, and the trajectory follows the one in IN.MOVEMENT. Note, in the running directory, a IN.MOVEMENT file need to be provided. This is mostly used for some special purposes, for example for machine learning force field development, while the IN.MOVEMENT is generated by force field. For MD=101, the charge and wave function interpolation is turned off. This might be useful if the configures in IN.MOVEMENT change dramatically from one frame to another frame. While the format in IN.MOVEMENT is the same as in the output MOVEMENT, it must provide a header. It is like this:

```
The formate of IN.MOVEMENT
------------------------------
nstep,nskip1,nskip2,nkip3,njump
64, atoms,Iteration .....
...
Lattice vector (Angstrom)
0.1130000000E+02    0.0000000000E+00    0.0000000000E+00
0.0000000000E+00    0.1130000000E+02    0.0000000000E+00
0.0000000000E+00    0.0000000000E+00    0.1130000000E+02
Position (normalized), move_x, move_y, move_z
31         0.99973    0.99973    0.99973     1  1  1
31         0.99985    0.24985    0.24985     1  1  1
.......
.......
----------------------------------------
The code used to read the IN.MOVEMENT file is as following:
```

```
-------
do istep=1,nstep
do ii=1,njump
do i=1,nskip1
read(IN.MOVEMENT,*)
enddo
read(IN.MOVEMENT,*) AL(1,1),AL(2,1),AL(3,1)
read(IN.MOVEMENT,*) AL(1,1),AL(2,1),AL(3,1)
read(IN.MOVEMENT,*) AL(1,1),AL(2,1),AL(3,1)
do i=1,nskip2
read(IN.MOVEMENT,*)
enddo
do i=1,natom
read(IN.MOVEMENT,*) iat(i),x1(i),x2(i),x3(i)
enddo
do i=1,nskip3
read(IN.MOVEMENT,*)
enddo
enddo ! ii=1,njump
enddo ! istep=1,nstep
----------------------------------------
```

The nskipt1,nskipt2,nskip3 are the skips of lines in different segment of the IN.MOVEMENT file. The njump indicates whether you like to calculate every configuration (njump=1), or you like to jump over some configurations, e.g., njump>1. nstep is the total number of steps to calculate. You might need to check the IN.MOVEMENT file to determine the nskip1,nskip2,nskip3. nskip2 is usually 1.

2.1.5.8 MD_VV_SCALE

MD_VV_SCALE = NSTEP

Default:

MD_VV_SCALE = 100

To scale the kinetic energy in Verlet MD (for JOB=MD, iMD=1/11) for every NSTEP steps, so the total energy is conserved. The default NSTEP is 100. This is used for enforce the total energy conservation for Verlet. Note, in TDDFT, NAMD,

or some MD when there is external potential or electric field, the total energy is not supposed to be conserved. In those case, please set MD_VV_SCALE to a very large number, so it will never be used. The default value is MD_VV_SCALE=100 for MD, and not used for TDDFT and NAMD.

2.1.5.9 TDDFT_DETAIL

Format: TDDFT_DETAIL = m_1, m_2, mstate

Default: TDDFT_DETAIL = 1, NUM_BAND, NUM_BAND

This will be read in when JOB = TDDFT. Note if mstate=-1, this is for TDDFT_NOB calculation, see below.

Note, when JOB=TDDFT, besides TDDFT_DETAIL, it also reads in parameters from MD_DETAIL, and optionally from TDDFT_SPACE, TDDFT_TIME, TDDFT_STIME.

In the TDDFT calculation, we expand the time dependent electron orbital $\psi_j(t)$ in terms of the adiabatic eigenstates $\phi_i(t)$

$$\psi_j(t) = \sum_i C_{ji}\phi_i(t) \tag{2.7}$$

There will be *mstate* electron wavefunctions [j=1,mstate] which will be occupied by their occupation number o(j) and described by $\psi_j(t)$.

However, for the first $m_1 - 1$ orbital ($j = 1, m_1 - 1$), $\psi_j(t)$ is just $\phi_i(t)$, i.e., these m_1 states are fully occupied like in Born-Oppenheimer MD, no electron excitation:

$$\psi_j(t) = \phi_j(t), j = 1, m_1 - 1 \tag{2.8}$$

For the next $j = m_1, mstate$ state (so, mstate include the [1,$m_1 - 1$] state!), we will expand the $\psi_j(t)$ in the $\phi_i(t)$ window of $i = m_1, m_2$:

$$\psi_j(t) = \sum_i C_{ji}(t)\phi_i(t), j = m1, mstate; i = m1, m2 \tag{2.9}$$

Thus, in total, the expansion window is $[m_1, m_2]$, and the total number of time dependent orbital is: mstate (they will be occupied by o(j), so mstate can be larger than the NUM_ELECTRON/IPSIN. However, within the mstate, the first $m_1 - 1$ states are fully occupied, and just equal to the adiabatic eigen states, the next $mstate - m_1$ state will be expanded using adiabatic state within the window of $[m_1, m_2]$, and their occupation might follow the Fermi-Dirac rule, or to be input by IN.OCC/IN.OCC_2 (see Appendix B). The initial C_{ji} can also be input from IN.CC/IN.CC_2 (see Appendix B).

$[m1, m2]$	Adiabatic window $\phi_{i,i=m1,m2}$. The $[1, m1-1]$ will always be occupied by the first $\psi_{j,j=1,m1-1}$ states. $m2 \in [m1, NUM\_BAND]$, usually $m2$ is smaller than NUM_BAND by a few states, because the last few states maybe not converge well.
$[1, mstate]$	Wavefunction index. $\psi_{j,j=1,mstate}$. $mstate \in [m1, m2]$

The choice of m2 is important for the physical correctness of the TDDFT simulations. The choice might depend on the physical problems at hand. Larger the m2, more accurate will be the simulation, but it can also cost more time to calculate. Typically, from mstate to m2, one should include all the possible electron excitations. For example, for a light excitation, if the hot electron can be 1-2 eV above the bottom of conduction band, then m2 should be choosen to include all these bands. Sometime m2 can be twice as mstate. But tests are needed to determine this. All depend on how high the electron can be excited to the conduction band.

If mstate=-1, this is a special case, for TDDFT_NOB calculation. NOB stands for natural orbital branching. In this case, we need another line, which is:

TDDFT_NOB = iseed, S_c, tau, temp2, kin_scale, select_opt

iseed (negative integer) is a random number seed for the stochastic NOB calculation. S_c is the cut-off entropy for the branching. tau is the dephasing time (in fs). If tau is negative, then IN.BOLTZMANN_TAU will be used to input tau(i) for each state,

and the tau_{ij} will be determined from $\sqrt{tau(i)tau(j)}$. temp2 is the temperature for the case of kin_scale=2. kin_scale=1,2,3 are three different ways for kinetic energy scaling after branching, similar to the flag_scale=1,2,3 in the TDDFT_BOLTZMANN flag. kin_scale=1 will scale all the atom's velocity uniformly to conserve the total energy. kin_scale=2 will scale all the atom's velocity uniformly to keep the temperature at temp2. As a result this method will not conserve the total energy, and the total energy will usually graduately decrease. kin_scale=3 will be the standard way to scale the kinetic energy in the transition degree of freedom, and conserve the total energy.

Select_opt=1,2,3 is the option for branching algorithm. 1: for using the transition eigen energy difference and the current temperature (derived from the current kinetic energy) with an Boltzmann factor to determine the natural orbital branching probability to restore the detailed balance. 2: for using the SCF total energy difference after a trial branching and the current temperature to determine the natural orbital branching probability. 3: this will also use the actual SCF total energy, instead of eigen energies, to determine whether a branching is allowed. However, instead of using a temperature and an Boltzamn factor, in this scheme, the transition degree of freedom is first determines, and whether this degree of freedom has enough kinetic energy to compensate the SCF total energy increase, is used to determine whether one particular branching is allowed. Note, for select_opt=3, one must also choose kin_scale=3.

Note, kin_scale=3, select_opt=3 will be the standard way of doing statistical branching.

2.1.5.10 TDDFT_SPACE

TDDFT_SPACE = itype_space, N, a(1), ..., a(N)

itype_space = 0 / 1 / 2 / 3 / -1

Default:

TDDFT_SPACE = 0

This controls the real space Vext_tddft(r). Vext_tddft(r) refers to the external potential in real space for tddft calculation.

itype_space	
0	No external input term.
1	Read vext_tddft from file IN.VEXT_TDDFT(all capital, same format as in IN.VEXT).
2	$Vext\_tddft(r) = (x - a(1))a(4) + (x - a(1))^2a(5) + (y - a(2))a(6) + (y - a(2))^2a(7) + (z - a(3))a(8) + (z - a(3))^2a(9)$, a(1),a(2),a(3) in fractional coordinates, a(4)-a(8) in unit of Hartree/Bohr. output file OUT.VEXT_TDDFT.
3	$Vext\_tddft(r) = a(4)e^{-[(x-a(1))^2+(y-a(2))^2+(z-a(3))^2]/a(5)^2}$, a(1),a(2),a(3) in fractional coordinates, a(4) in unit of Hartree, a(5) in unit of Bohr. output file OUT.VEXT_TDDFT.
-1	Not use real space format, but use G-space,it wil use IN.A_FIELD

The 'IN.VEXT_TDDFT' file can be copied from other TDDFT calculation output file 'OUT.VEXT_TDDFT', or generated by utility programs add_field.x.

2.1.5.11 TDDFT_TIME

TDDFT_TIME = itype_time, N, b(1), ..., b(N)

itype_time = 0 / 1 / 2

Default:

TDDFT_IIME = 0

This is used to control the time dimension of the external function fTDDFT(i).

itype_time	
0	$ftddft(t) = 1.0$
1	read in $ftddft(i)$ from IN.TDDFT_TIME
2	$ftddft(t) = b(1)e^{-(t-b(2))^2/b(3)^2)}\sin(b(4)t + b(5))$. $b(2)$,$b(3)$ in unit of fs; $b(4)$ in unit of rad/fs unit, $b(5)$ in unit of rad; $b(1)$ no unit. output file OUT.TDDFT_TIME
22	$ftddft(t) = \int_0^t [b(1)e^{-(t-b(2))^2/b(3)^2)}\sin(b(4)t+b(5))]dt$. $b(2)$,$b(3)$ in unit of fs; $b(4)$ in unit of rad/fs, $b(5)$ in unit of rad; $b(1)$ no unit. output file OUT.TDDFT_TIME

File IN.TDDFT_TIME format,

```
0 ftddft(0)
1 ftddft(1)
...
N ftddft(N)
```

For TDDFT Hamiltonian, we have,

itype_space	
$\neq -1$	$H(t) = H_0 + Vext\_tddft(r)ftddft(t)$
-1	$H(t) = -1/2(\nabla_x + iA_x * ftddft(t))^2 - 1/2(\nabla_y + iA_y * ftddft(t))^2 - 1/2(\nabla_z + iA_z * ftddft(t))^2$

2.1.5.12 TDDFT_BOLTZMANN

TDDFT_BOLTZMANN = flag_b, flag_scale, temp, tau, istep_start(opt), nstep_CG(opt)

This line controls whether to introduce Boltzmann factor in order to keep the correct detailed balance between two adiabatic states $\phi_{i1}(t)$ and $\phi_{i2}(t)$. This goes beyond the usual Ehrenfest dynamics. In the Ehrenfest dynamics, the electronic system will be

over heated due to the lack of detailed balance (which means the transition from the lower energy adiabatic state $\phi_{i1}(t)$ (with eigen energy $E_{i1}(t)$) to higher energy adiabatic state $\phi_{i2}(t)$ (with eigen energy $E_{i2}(t)$) is a factor of $exp(-(E_{i2} - E_{i1})/kT)$ smaller than the transition from $\phi_{i2}(t)$ to $\phi_{i1}(t)$. This suppression of the up-lifting transition can be realized by adding this Boltzmann factor. Adding this Boltzmann factor is critical in order to have the proper hot electron cooling. However, an dephasing time tau has to be used with an special algorithm when adding this Boltzmann factor, otherwise the cooling will be too fast (e.g., an energy conservation between the transition states and an phonon mode in the weak coupling regime will not be satisfied). A special algorithm is implemented in PWmat to properly take into account the tau. In a way, this is like the Tully's surface hopping, but without the stochastic feature in the dynamics. Compared with Tully's algorithm, it has more correct dephasing behavior. Compared with wave function collapsing behavior, it can have more proper treatment for tau. However, this is a meanfield treatment for the nuclear trajectory. Unlike the stochastic potential energy surface hopping, or wave function collapsing treatment, the current meanfield treatment does not provide a branching for the trajectory (thus might not be good if you like to calculate the probability for different chemical reaction, etc). The Boltzmann factor is not applied to each individual electron state $\psi_j(t)$ (and its collapsing), instead, it is applied to the occupation of adiabatic state $\phi_i(t)$ (collectively for all $\{\psi_j(t)\}$), as a result, it has a property of unitary rotational invariance among the group of $\psi_j(t)$.

flag_b: indicates whether to use the Boltzmann factor: 0 means no Boltzmann factor (Ehrenfest dynamics); 1 means with Boltzmann factor. Note, one has to detemine according to the physics, whether to use this Boltzmann factor. For example, if the dynamics does not involve phonon, and it is very short (e.g., using rt-TDDFT to simulate the light absorption), then one might not want to include the Boltzmann factor. One thing needs to be mindful: for long time rt-TDDFT simulations, the use of Boltzmann factor can make the simulation more stable, since all the states are moving towards equilibrium.

flag_scale: indicates what method to be used to scale the kinetic energy. When using

the Boltzmann factor, the total energy will not be conserved without the rescaling of the velocity of the nuclei (usually, the total energy will decrease due to the electron cooling). There are different ways to deal with this.

flag_scale=0: this means without scaling the kinetic energy, as a result, the total energy will decrease with time.

flag_scale=1: this means the velocities of all the nuclei will be rescaled with a uniform scaling factor, so the total energy will be an constant (thus the lost electronic energy will be given to the phonon movement). For example, this will be useful for isolated molecule. In this case, if there is an initial hot electron, the temperature of the system will likely increase with time. Note, the Boltzmann factor $exp(-(E_{i2}-E_{i1})/kT)$ dynamically depends on this temperature T(t).

flag_scale=2: this means the velocities of all the nuclear will be scaled to keep the temperature (the total kinetic energy) to a constant, specified by *temp*. In this case, the temperature in the MD_DETAIL will not be used, but the *temp* specified here will be used. This might be a good approximation if the studied system is embedded in a thermal bath, which is always kept in a constant temperature.

flag_scale=3: this is like $flag\_scale = 1$, where the kinetic energy is modified to keep the total energy conserved. However, instead of uniformly scale the velocity of all the atoms by a constant factor, here the electron-phonon coupling constant is used to specify a force of a given atom, which is then used to be added to the current velocity to conserve the total energy. This is more rigorous treatment for what phonon degree of freedom to give the extra kinetic energy for. More specifically, let's use TCD(i1,i2) to indicate the Boltzmann factor introduced modification of the density matrix, which represents the charge transfer between adiabatic states i1 and i2 (this modification maintains the detailed balance between adiabatic states i1 and i2, but also cause the violation of energy conservation). We also define $D(i1,i2) = \sum_j C*(i1,j)C(i2,j)o(j)$, here C(i,j) is the expansion coefficient of electron wave function $\psi_j(t)$ on adiabatic state $\phi_i(t)$, i.e, $\psi_j(t) = \sum_i C(i,j)\phi_i(t)$, and o(j) is the occupation (not change with time) of electron state $\psi_j(t)$. Then the extra

atomic force due to the TCD(i1,i2) between adiabatic states $\phi_{i1}(t)$ and $\phi_{i2}(t)$ will be: $F(R) = \sum_{i1,i2} TCD(i1,i2)D(i1,i2)/abs(D(i1,i2) < \phi_{i1}|\partial H/\partial R|\phi_{i2} >$, here R is the nuclear coordination. Then to conserve the total energy we have used: $V(R) = V(R) + xF(R)/m(R)$, here m(R) is the nuclear mass, and x is chosen for the smallest value which satisfies the total energy conservation. This x is reported in the screen output of PWmat in the line: "TDDFT,boltzmann,kin:istep,x,dE,a*x**2+b*x+c=0", together with dE(eV), which is the energy drop due to the Boltzmann factor. The a*x**2+b*x+c=0 is the equation used to solve for x.

temp: the fixed temperature when $flag\_scale = 2$. Note, the temp1,temp2 in the MD_DETAIL line will not be used for rt-TDDFT simulation (except the temp1 is used to set an initial velocity of the system). We provide this temp here, to distinguish the temp1 for the initial velocity and the temperature you want to use in the Boltzmann factor.

tau: the dephasing time (in fs unit) for all the adiabatic state transitions. If tau=-1 (negative), then one needs to provide an file $IN.BOLTZMANN\_TAU$, which specifies the tau for different states as described below.

$IN.BOLTZMANN\_TAU$: the tau input file when tau < 0 in above. This is recommended for the stability of the TDDFT algorithm. It has the following format:

tau(m_1), tau(m_1+1),tau(m_1+2),....., tau(m_2).

There should be $m_2 - m_1 + 1$ numbers in a single line, here the m_1 and m_2 are the first and last adiabatic state index specified in the $TDDFT\_DETAIL$ line. Note, all the numbers are in fs unit. Also note that, for an adiabatic pair i1 and i2 transition, the actual tau for this tau(i1,i2) is sqrt(tau(i1)*tau(i2)). It is recommended that for the few highest adiabatic states in the adiabatic state window [m_1,m_2], use very small tau. E.g., for the last few tau close to m_2, use 0.0001fs. This will make sure these adiabatic states will not have large amplitude $C(i,j)$, which will help the convergence of the TDDFT algorithm.

There are actually more rigorous formulas to calculate tau(i1,i2) from their eigen energies time dependence:

$$\tau_{i,j} = 2kT[< |\partial\epsilon_i(t)/\partial t - \partial\epsilon_j(t)/\partial t|^2 >_{ave}]^{-1/2}$$

here $\epsilon_i(t)$ is the eigen energy of adiabatic state $\phi_i(t)$ and the $<>_{ave}$ means for a time average. However, in our simulation, we did not calculate this $\tau_{i,j}$, although one can use the above formulat to estimate the decoherent time between a given pair of adiabatic states.

istep_start (optional): This is an optional value, to indicate from which TDDFT MD step it begins to use the Boltzmann method. If not input, the default value is 1. One can use this parameter to delay the deployment of Boltzmann method, in order to make the algorithm more stable, or for other purposes. One reason is that, for the first 1 or 2 steps, it is possible the TDDFT step itself is not converged. If the SCF TDDFT is not converged, it might appear to be there are large rotations between different eigen states, then that can introduce large Boltzmann correction, and it is wrong, and make can the algorithm unstable. In that case, one can set istep_start to be 3 or 5, to make the algorithm more stable. Of course, one can also use this to investigate different physics.

nstep_CG (optional): This is an optional parameter to control the number of steps in an internal conjugate gradient linear solver. Note, to input this parameter, one has to have the istep_start paremeter before nstep_CG. The default value of nstep_CG is 1000. However, for large systems, the solution of a linear equation can significantly slow down the calculation. One can test the use of nstep_CG=500, or 250 to speed up this step.

2.1.5.13 NAMD_DETAIL

Format:

NAMD_DETAIL = m_1, m_2, **nstep_out**

or

NAMD_DETAIL = m_1, m_2, **nstep_out, icycle, nstep_cycle, icrossk, hc**

Default: None

This line is needed (either the first format, or the second format) when JOB=NAMD.

Note, when JOB=NAMD, besides NAMD_DETAIL, it also reads in parameters from MD_DETAIL, and other possible options from TDDFT_SPACE, TDDFT_TIME, TDDFT_STIME, NAMD_SPECIAL.

In the NAMD calculation, it performs an conventional Born-Oppenheimer MD, but outputs the wave function overlap between consecutive time steps for the adiabatic eigenstates within the window $[m_1, m_2]$. It uses the MD parameters from MD_DETAIL. The output is written in OUT.NAMD, and will be used in post-process programs like "namd_dm.x" in Boltzman-NAMD, to carry out non-adiabatic MD simulation for carrier dynamics. The carrier wave function $\psi(t)$ will be described by the adiabatic eigen states set within the window $[m_1, m_2]$:

$$\psi(t) = \sum_i C_i(t)\phi_i(t), i = m1, m2 \tag{2.10}$$

Note, inside OUT.NAMD (which is an unformatted file), it has the following write-out format:

```
write(10) istep0,islda,nkpt,time,mst_win
do iislda=1,islda
do kpt=1,nkpt
write(10) kpt,iislda,mst_win
write(10) eigen(1:mx,kpt,iislda)
enddo
enddo
do istep=1,nstep
write(10) istep,islda,nkpt,time,mst_win
do iislda=1,islda
do kpt=1,nkpt
write(10) kpt,iislda,mst_win
write(10) hh(1:mst_win,1:mst_win,kpt,iislda)
write(10) eigen(1:mx,kpt,iislda)
enddo
enddo
enddo
```

The first step is different from the rest of the steps. mx is the number of bands calculated, while $mst\_win = m_2 - m_1 + 1$ is the window of the NAMD output. Eigen is the eigen energy in atomic unit (Hartree). $hh(m1, m2, kpt, iislda) =< \psi^*_{m1}(istep - 1)|\psi_{m2}(istep) >$ for this kpoint and spin iislda.

There are also more advanced options (the second line form). There icrossk=0 or 1: 0, no action for this, 1, output the kpoint cross product in another file: OUT.NAMD_CROSSK. This is only useful if multiple kpoints are used. This file has the following format:

```
write(10) istep0,islda,nkpt,time,mst_win
do iislda=1,islda
do kpt=1,nkpt
write(10) kpt,iislda,mst_win
write(10) eigen(1:mx,kpt,iislda)
enddo
enddo
do istep=1,nstep
write(10) istep,islda,nkpt,time,mst_win
do iislda=1,islda
do kpt=1,nkpt
write(10) kpt,iislda,mst_win
write(10) hh2(1:mst_win,1:mst_win,1:nkpt,kpt,iislda)
write(10) eigen(1:mx,kpt,iislda)
enddo
enddo
enddo
```

The only difference is that, in OUT.NAMD, each time step we have: hh(m1,m2), here we have $hh2(m1, m2, kpt2, kpt, iislda) =< u^*(m1, kpt, istep-1)|u'(m2, kpt2, istep >$, so you have cross kpoint dot-product. Note, u is the Bloch part of the wave function, so the cross-k point dot product is not zero. For kpt2.eq.kpt, u' is just the u. But if kpt2.ne.kpt, then u'(istep) has been project out the component of the u(istep-1) in the following way:

$$u'(m_2, kpt_2, istep) = u(m_2, kpt_2, istep) - \sum_{m_3} h(m_3, m_2, kpt_2) *$$

$$[1 - exp(-(|h(m_3, m_2, kpt_2)|^4/hc^2)]u(m_3, kpt_2, istep - 1)$$

here $h(m_3, m_2, kpt_2) =< u^*(m_3, kpt_2, istep - 1)|u(m_2, kpt_2, istep >$. Everything is done within the same iislda. The above output can be used for a special algorithm for postprocess NAMD calculations allowing the hot carrier to jump k-points. One can take hc to be about 0.1, for time step 1fs MD. For better result, one can reduce the time step, while further decrease hc.

Another special option is the icycle option. If icycle=0, no action will be taken. If icycle=1, then the program will try to make the MD periodic in time. Note, for this scheme, one cannot restart the MD. So, iMD must equal to 1, not 11, etc. The nstep_cycle is the buffer region. One possibility, for example, is MDstep=1000 (or 2000), nstep_cycle=200. The idea is that, the code will make the last step (istep=MDstep) the same (in both atomic position and velocity) as the nstep_cycle step (note, not the first step). Hence the MD will do a time cycle with a periodicity of MDstep-nstep_cycle. In the NAMD post-process, this can be used to carry out the NAMD forever, while the nuclear movement has a periodicity of MDstep-nstep_cycle steps. Note, ath the MDstep-th step, the adiabatic wave function has used the same one as that for the nstep_cycle-th step. This allows the NAMD calculations to continue forever. Note, it must be critical to check the potential and kinetic energy in MDSTEPS file, make sure the transition period is smooth. It is critical, for this to work, the whole system from beginning to the end should not drift away. Instead, most atom should only have vibrations (e.g., as in a crystal system).

The nstep_out is the number of steps interval to output the wave functions (within the window $[m_1, m_2]$) into the ugio.allxxxxxx file. Note, this could be large files, so you probably don't want to output the wave function at every step (e.g., nstep_out=1). However, sometime the output wave functions can be used to do some postprocessing, which are not done during the NAMD simulation. For example, the wave functions can

be used to introduce some additional term δH, so you have $< \phi_i(t)|\delta H|\phi_j(t) >$ during the postprocess steps. In order to do this, you still don't need to output $\phi_i(t)$ at every time step, because the code does output the overlap $S_{ij} =< \phi_i(t)|\phi_j(t+dt) >$ between consecutive steps, so you can use the S_{ij} to link the adiabatic states at different time. Nevertheless, you might still want to use a relatively small nstep_out for this regard. As a balance, we found that nstep_out=50 might be a good choice for many problems. But be prepared for a lot of files!

For more information, please see APPENDIX B.

2.1.5.14 NEB_DETAIL

Format: NEB_DETAIL = IMTH, NSTEP, FORCE_TOL, NIMAGE, AK, TYPE_SPRING, E_0, E_N, ITYPE_AT2, ATOM2.CONFIG

Default: None

The **NEB_DETAIL** line is needed when JOB=NEB.

For the NEB algorithm, please refer to Ref.[16]. In the NEB run, NIMAGE+2 atomic configurations are used, the NIMAGE intermediate configurations connect the initial configuration (in ATOM.CONFIG) with the final configuration (in ATOM2.CONFIG). This is also called the string. During the NEB run, the NIMAGE intermediate configurations will be relaxed together, almost like a NIMAGE*natom atom large system. However, during the atomic relaxation, the atomic force component along the string will be removed (hence not be minimized), so this is call the nudged elastic band method. The goal is to have the force perpendicular to the string to be zero (the string will be moved during the relaxation), while leave alone the atomic force component along the string, meanwhile hopefully keep the distance roughly equal among the NIMAGE+2 image points. Larger the NIMAGE, more difficult is the calculation. For simple problems, typically NIMAGE can be about 5. The output of NEB is written in RELAXSTEPS, and MOVEMENT.

IMTH: the algorithm used for atomic relaxation.

IMTH = 1 / 2 / 3 / 4 / 5 / 6

1. IMTH=1, conjugate gradient;
2. IMTH=2, BFGS;
3. IMTH=3, steepest decent.
4. IMTH=4, VFF preconditioned conjugate gradient
5. IMTH=5, Limited-memory BFGS
6. IMTH=6, FIRE: Fast Inertial Relaxation Engine [20]

For NEB calculation, IMTH=5 and 6 are recommended for good convergency. Some options can be set for the optimizers in the optional file IN.RELAXOPT(need to set IN.RELAXOPT=T in etot.input). The IN.RELAXOPT is as follows:

```
RELAX_MAXMOVE = 1.0
    ! max move distance. unit bohr – for method=1,5,6.
LBFGS_MEMORY = 30
    ! LBFGS storage size – for method=5.
FIRE_DT = 1.0
    ! initial time step. unit fs – for method=6.
    ! the max time step for FIRE method is 10*FIRE_DT.
RHOWG_INTER_TYPE = 1
    ! interpolation type for NEB,0–both rho and wave function; 1–rho.
    ! default = 1, save time by not writing wavefunction to disk
```

NSTEP: the maximum number of line-minimization steps in the relaxation process. This is the NEB steps.

FORCE_TOL: the atomic force tolerance ($eV/$Å) to stop the relaxation. This is the maximum atomic force (after the component along the string direction has been projected out) of all the atoms and all the images.

NIMAGE: the number of images in the NEB method (these are the images except the initial and final two valleys). So, there are in total NIMAGE+2 configurations in the string of images connection the initial and final configurations. In a NEB

calculation, NIMAGE+2 atomic configurations (called images) are used, which connect the configuration from the initial state to the final state. Initially, the NIMAGE intermediate images are generated by linear interpolations of the two end images (the two end images, one initial, one final, are input by the user). The two end images will not be changed, while the NIMAGE intermediate images will be relaxed.

AK: the spring constant for the image string ($eV/\text{Å}^2$). In the NEB, a string connecting the images are used to ensure the coverage between the initial and final configurations. AK=0.1 to 1 $eV/\text{Å}^2$ are reasonable values. Larger AK (especially for TYPE_SPRING=2), better the convergence, but it can introduce bigger errors (for TYPE_SPRING=2).

TYPE_SPRING: the type of string used in NEB algorithm.

1. TYPE_SPRING=1, the original NEB algorithm (where the string force perpendicular to the string tangent is removed);
2. TYPE_SPRING=2, a conventional string, the perpendicular string force is not removed;
3. TYPE_SPRING=3, the regular NEB algorithm(Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points) [17]

TYPE_SPRING=2 converges better, but it can introduce an error (larger AK, larger the error). But one can first use larger AK, then after the initial NEB relaxed, re-runs NEB using smaller AK (or TYPE_SPRING=1,3). This will help the convergence.

Usually the TYPE_SPRING=1,2,3 converge good with IMTH=5,6. One can directly try the TYPE_SPRING=1 or 3 with IMTH=5 or 6, if bad, then follow the above advice(use TYPE_SPRING=2 first).

Additionally, one can use TYPE_SPRING=11,22,33 for CI-NEB(11-original CI-NEB, 22-conventional string CI-NEB, 33-regular CI-NEB). The climbing image NEB(CI-NEB) method constitutes a small modification to the NEB method[18]. Information about the shape of the MEP is retained, but a rigorous convergence to a

saddle point is also obtained. CI-NEB will choose the image with highest energy as the climbing image, then reformat the force on the climbing image. The force on the climbing image is the full force due to the potential with the component along the elastic string inverted, so the climbing image is not affected by the spring forces, and will climb up along the elastic string to converge rigorously on the highest saddle point.

Some suggestions

1. How to choose NEB method – TYPE_SPRING

As mentioned above, TYPE_SPRING=2 is easy to converge, but can introduce large error. So if TYPE_SPRING=1 or 3 converges bad, you can first run the TYPE_SPRING=2 to get a better guess of the MEP, then rerun NEB with TYPE_SRPING=1 or 3.

For TYPE_SPRING=1 (original NEB) and TYPE_SPRING=3 (regular NEB), both has intrinsic instability. For the original NEB, "In systems where the force along the minimum energy path is large compared to the restoring force perpendicular to the path and when many images of the system are included in the elastic band (string), kinks can develop and prevent the band from converging to the minimum energy path". For the regular NEB, the spring force's formulation omits the the spring force that is perpendicular to the local tangent, then you may encounter that pathway deviate from the MEP and the overall path lengths may grow out of proportion. [19]

Maybe your first choice can be TYPE_SPRING=1, if you find the 'kinks' is the problem, then try TYPE_SPRING=3.

2. How to choose relaxation method – IMTH

All IMTH=1,2,3,4,5,6 are methods based on forces, and IMTH=1,2,3,4 each has an exact line-minimization, IMTH=5,6 do not have exact line-minimization. If the forces are not smooth in the calculations, methods with exact line-minimization may not converge well (you will see jumps of the energy as a function of iteration steps). In NEB calculations we recommend the IMTH=5,6. IMTH=5 (Limited-memory BFGS) is fast, and you can set RELAX_MAXMOVE in file IN.RELAXOPT to make it more

stable or more aggressive. IMTH=6 (Fast Inertial Relaxation Engine) is stable, and you can set FIRE_DT in file IN.RELAXOPT to get more stable or more faster convergence.

We recommend the first choice is IMTH=5.

3. How to choose the pseudopotentials

PWmat provide several types of pseudopotentials, NCPP-SG15, NCPP-PD03, NCPP-PD04, ONCV-PWM-PBE, etc. We recommend to try to use the ONCV-PWM-PBE first, it is more smooth, and more easy to converge for relaxation. However, you might want to test this pseudopotential with the more accurate ones, like SG15 and PD03. There are cases the PWM-PBE gives the wrong results. But if it works, you can use it for large system calculations. If you want to use SG15,PD03,or PD04, we recommend to use bigger Ecut (e.g, 50, 60 Ryd), and set Ecut2=4*Ecut (to avoid egghead problem).

4. How to do CI-NEB

Do not use CI-NEB from the beginning, that will converges bad. PWmat CI-NEB will choose the climbing image automatically, so the climbing image could change during the convergence process if do CI-NEB from the beginning. We recommend first converge your NEB calculation, then rerun use the CI-NEB. (CI-NEB is specified use TYPE_SPRING: 11-original CI-NEB, 22-conventional string CI-NEB, 33-regular CI-NEB).

5. The last but not least

Make sure the SCFs are converged.

E_0, E_N: the precalculated (e.g., using JOB=RELAX) initial (E_0) and final (E_N) local minima energies (in eV) for configurations in ATOM.CONFIG and ATOM2.CONFIG. Actually, these numbers are not used in the algorithm, but will make plotting more straight forward.

ITYPE_AT2, ATOM2.CONFIG: the type of ATOM2.CONFIG file and the atomic position file name: ATOM2.CONFIG.

1. ITYPE_AT2=1, ATOM2.CONFIG is the second minimum configuration (the first local minimum configuration is given in IN.ATOM = ATOM.CONFIG). Then,

from ATOM.CONFIG to ATOM2.CONFIG, NIMAGE equal distance images will be created by linear interpolations.

2. ITYPE_AT2=2, ATOM2.CONFIG contains all the NIMAGE+2 image configurations, including the initial and final images. Thus ITYPE_AT2=2 is a continued NEB run following the previous NEB runs. You can use the final.config from a previous unconverged NEB run as ATOM2.CONFIG, or copy a group of images from 'MOVEMENT' file. In this case, the ATOM.CONFIG in IN.ATOM = ATOM.CONFIG is not used (but that line still need to be provided). There are cases where the linear interpolation between the first and the last configuration will generate some unphysical NIMAGE IMAGIES (e.g., with atoms too close together, or bond orders wrong). In that case, the user can use ITYPE_AT2=2 to provide NIMAGE intermediate images manually, to avoid the unphysical interpolation.

2.1.5.15 SCFEP_DETAIL

Format: SCFEP_DETAIL = Level1, Level2, α, Numkpt, Numspin

Default: None

This is a required line for JOB=SCFEP electron-phonon coupling constant calculation. The calculated electron-phonon coupling constant will be reported in OUT.EP_COEFF. See JOB=SCFEP (see 2.1.1.3) for details. In the JOB=SCFEP calculation, one must use IN.WG=T, an input wave function file will be provided. The electron-phonon coupling constant written in OUT.EP_COEFF will report: $< \psi(i_1,k,s)|\partial H/\partial R|\psi(i_2,k,s) >$, here $\psi(i,k,s)$ is the input wave function from IN.WG for state index i, kpoint k, and spin s.

Level1, Level2: The wave function index i_1 and i_2

α: a small number (e.g., 0.1) used to add $\alpha\psi(i1) * \psi(i2)$ onto the charge density to do SCFEP calculation. Here, we assume $\psi(i1)$ and $\psi(i2)$ are real. Suggested α is 0.1. Smaller this number, the numerical derivative will be more accurate, but it also require higher level convergence for the SCF calculations.

Numkpt: The kpoint index k for $\psi(i,k,s)$.

Numspin: The spin index s (1 or 2) for $\psi(i, k, s)$.

2.1.5.16 SCF_SPECIAL

Format: SCF_SPECIAL = iflag, Ef0, i1, i2, j1, j2, k1, k2

Default: SCF_SPECIAL = 0

This special option is for nonequilibrium boundary condition calculation for JOB=SCF. Currently, it does not support JOB=NONSCF and JOB=MD (it can run, but the results might not be good). In many problem (for example, device simulation), there could be fixed electrode potentials, there we like the potential to satisfy some specific boundary condition. Note, this might be different from the fixed Fermi Grand canonical calculation, where one electrode potential is fixed. The fixed Fermi is often used together with solvent model with Poisson-Boltzmann method. In the current case, it is often for pure solid calculation, and there is no implicit solvent model. Besides, it is often the case, there are several electrodes. We like the potential on some boundaries (enclosing boundary) to be the given values (e.g., representing the on and off of a CMOS gate, or the source and drain bias potential). Furthermore, this is truely an nonequilibrium simulation, as these boundary values (of the electrode voltage) is often related to the local Fermi energy. So, there is not a single Fermi energy in the simulation. We thus have to use position dependent Fermi energy. Thus, iflag=1 will represent such JOB=SCF calculations. It will give the SCF potential file under such boundary condition. This potential file can be used for quantum transport calculation.

To do such nonequilibrium calculation, we will do two steps.

In the step one, a normal JOB=SCF calculation (iflag=0, or without the SCF_SPECIAL line) will be carried out. We will take its global Fermi energy as Ef0 (eV) input in the above line. Besides, copy OUT.VR into IN.VR0, which will be used for iflag=1 calculation. Also please copy OUT.RHO into IN.RHO, OUT.WG into IN.WG for subsequent iflag=1 calculation for fast convergence.

In the second step, set iflag=1. Place Ef0 as mentioned above. Now, we need to use i1,i2,j1,j2,k1,k2 to specify the boundary condition. These are the grid points in the

n123 grid of the iflag=0 calculation. More specifically, $i1, i2 \in [1, n_1]$, $j1, j2 \in [1, n_2]$, $k1, k2 \in [1, n_3]$. These are the planes where the fixted potential will be speficied. Note, the boundary condition is not determined by the edge of the periodic box, instead they are specified by these plane. These plane will define a smaller box inside the periodic box, the corner of this smaller box is at [i1,j1,k1], while the size of this box is: (i2-i1,j2-j1,k2-k1). Note, for this calculation, it is essential to shift the coordinates, so this small box is in the middle of the periodic box. Besides, it is also for some dimention to be periodic. in that case, the corresponding ijk1,ijk2 should both be zero. For example, to have the periodic condition in the second dimension, we should have j1=0,j2=0.

Now, for the dimensions which are not periodic boundary condition (their ijk1,2 are not zero), we need to prepare the corresponding IN.2D_VR.1, IN.2D_VR.2, IN.2D_VR.3 file. For example, if i1,i2 are not zero, then we need a IN.2D_VR.1 file. This file specifies the dEf(r) on the two planes of i1,i2. More specifically, it is written in the following format (ascii file, so it can be viewed and plotted):

```
do k=1,n3
do j=1,n2
write(IN.2D_VR.1,*) dEf1(j,k), dEf2(j,k)
enddo
write(IN.2D_VR.1,*)
enddo
```

Note, the two columns are for the i1,i2 two planes. There is an empty line, and the for i,j,k (n1,n2,n3), the earlier index are numerated first. This special format is used, so it can be plotted in gnuplot using "splot".

The units for dEf1,dEf2 are in eV. These IN.2D_VR.1,2,3 files are prepared by utility programs (e.g., gen_D2V.f), or written by the user. Note, the dEf1,2 are shift of the Fermi energy (also the potential V(r)) in reference to the Ef0 (V0(r)) at those boundary. Thus, the potential V(r) at those boundaries equals to V0(r)+dEf(r). After this, one can run the PWmat again. It will generate the system under the fixed boundary condition, the OUT.VR can be used to calculate the quantum transport for device simulation.

Here, we explain the underlying algorithm used to carry out the calculation. There are two aspects. One is the Poisson solution to satisfy the fixed boundary condition, another one is the occupation of the wave functions. For the Poisson solution, for a given charge $\rho(r)$, we first use the conventional FFT method to solve a periodic Poisson solution, to get $V_P(r)$. Note, in order to satisfy V(r) equals V0(r)+dEf(r) at the boundary, we can define: $dV_B(r) = V0(r) + dEf(r) - V_P(r)$ on the boundary (of the inner box, defined by i1,i2,j1,j2,k1,k2). Then solve a fixed boundary condition Poisson equation with zero charge, get dV(r) for values inside the box. This is solved by the Fishpack package. Then $V(r) = V_P(r) + dV(r)$. For dV(r) outside the box, we have used simple extension, from its boundary value.

For the occupation, we have used a spatial dependent Fermi energy $Ef(r) = V(r) - V0(r) + Ef0 + dE$. The occuption is done with with spatial dependent Fermi energy as: $\rho(r) = \sum_i occ((\epsilon_i - Ef(r))/kT)|\psi_i(r)|^2$, here ϵ_i is the eigen energy, and occ is the Fermi-Dirac occupation function. Note, we have used a small shift dE to guarantee we get the exact required total charge. Note, this procedure gaurantee the local Fermi energy is at the same position of the local density of state compared with the neutral charge calculation at the step 1. This local Fermi energy is important. For many device systems, the Fermi energy at the electrode is higher than the conduction band in the substrate etc. So, if a global Fermi energy is used, it can be cause large charge slashing from one side to another, which is not physical. In reality, under the open boundary condition (for the current), the system can maintain an steady state nonequilibrium solution, where spatially dependent Fermi energy exists.

2.1.5.17 MD_SPECIAL*

This includes: **MD_SPECIAL**, **MD_SPECIAL2**, **MD_SPECIAL3**, **MD_SPECIAL4**

In the following, we explain them seperately.

1. **MD_SPECIAL**

   Format:

MD_SPECIAL = iflag, x1, x2, x3, Rcut, dR, dV

or

MD_SPECIAL = iflag, x1 ,x2, x3, Rcut, dR, dV, frac, P, rate

Default: MD_SPECIAL = 0

iflag=1: this is a special constraint in MD. In this option, at t=0, all atoms within Rcut from the center (x1,x2,x3) will be frozen, and all the other atoms will be subjected to a spherical potential with height dV (eV) centered at (x1,x2,x3) (fractional coordinate) with radius cut-off Rcut (ansgtrom) and a buffer dR (in a potential file as: dV*exp(-(r-Rcut)/dR)/(exp(-(r-Rcut)/dR)+1)). This is used to keep the atoms out (dV>0) from one domain, or keep the atoms within one domain (dV<0).

iflag=2: this will not freeze the atoms within Rcut, otherwise, it is the same as in iflag=1.

iflag=22: this is the same as in iflag=2, except, when calculating the distance for one atom position (x1a,x2a,x3a) to the spherical center (x1,x2,x3), one do not do periodic wraping for the atoms position. Instead, the atoms position (x1a,x2a,x3a) are defined within ([0,1],[0,1],[0,1]) range. This can be useful, for example, to restraint the water on top of a slab, where one place the (x1,x2,x3) below the slab, so the water will be inside a halfdome on top of the slab, but it will not affect the water on the other side of the periodic box.

iflag=3: this option requires the frac,P,rate in the input line. frac is the fraction [0,1] of the halfdome to the whole sphere. E.g., if the sphere is a halfdome above a substate (like in iflag=22,33), then the frac will be less than 1. This is used to estimate the surface of the halfdome. P: the desired pressure in the unit of bar (0.987atmosphere). rate is a MD change rate for each MD step. The idea here is that, during MD, one can change Rcut (radius) of the confinement sphere (usually, dV is negative for this), so the pressure on the confinement wall equals to P. This will help to decide what Rcut one should use. Note, P should not be

too small, otherwise it might be difficult to converge. We recommend(e.g.): P = 50 bar (atmosphere), rate=0.02. Note, one should have a reasonably large dR, so the force will not be too large. For example, dV=-2 (eV), dR=1(A). frac is determined from the geometry. If it is a full sphere constraint, then frac=1. For iflag=33, e.g., an halfdome constraint, then it is the solid angle ratio between the halfdome solid angle and the 4π full sphere solid angle (this is used to calculate the halfdome surface area). The actually pressure (P_sph), the input desired pressure P_MD_sp (=P), and the dynamically adjusted Rcut (Rcut_MD_sp) will be reported in the header of MOVEMENT file for each step.

iflag=33: same as for iflag=3, except, when define the constraint, no wrapping, the same as for iflag=22. This is usually used to define a halfdome on top of a slab.

In iflag=1,2,3,22,33, one can use a "Weight_Atom" section in xatom.config file to specify which atom can feel this potential (or how much weight to feel this potential).

More specifically, in the xatom.config file, one can add a section like:

```
Weight_Atom
1
iatom(1), weight(1)
iatom(2), weight(2)
....
iatom(natom),weight(natom)
```

The weight(i), for i=1,natom can be used for many special purposes in the code where a coefficient for an atom is needed. iatom(i) is the atom z-number.

So, as a result, the potential felt by each atom is dV*weight(i). So, one can turn-off some of the atoms to feel this particular potential.

2. **MD_SPECIAL2**

Format: MD_SPECIAL2 = iflag, Rcut, dR, dV

Default: MD_SPECIAL2 = 0

This is typically used together with **MD_SPECIAL=1**, with its center defined using x1,x2,x3 defined in the **MD_SPECIAL** line. It defined another potential using parameters Rcut, dR and dV. However, it will be applied to the atoms with weight(i,2), which is defined in the xatom.config file section Weight_Atom as:

```
Weight_Atom
2
iatom(1), weight(1), weight2(1)
iatom(2), weight(2), weight2(2)
....
iatom(natom),weight(natom), weight2(natom)
```

3. **MD_SPECIAL3**

   Format: MD_SPECIAL3 = iflag, m1, m2

   Default: MD_SPECIAL3 = 0

   This is a very special case for electron phonon coupling calculation. The idea is to input m1 wave function $\phi_1(i)$ for i=1,m1, and m2 wave function $\phi_2(i)$ for i=1,m2 at the beginning of the MD simulation from files IN.WG1_MDSP3, IN.WG2_MDSP3. Then during the MD simulation, at every dt step, it will output: $hh(i,j) =< \phi_1(i)|H(t)|\phi_2(j) >$. Note, the H(t) is Hamiltonian at time t, and $\phi_1(i)$ and $\phi_2(j)$ are not changed during the MD.

   The $m1 \times m2$ double complex matrix $hh(i,j)$ is written inside the binary file: MDSP3.hh.out in an concatenation style (position="append"), i.e, continuesly written at its end. It has the following format:

```
t, m1, m2, nkpt, islda
hh(m1,m2,nkpt,islda)
t, m1, m2, nkpt, islda
hh(m1,m2,nkpt,islda)
....
t, m1, m2, nkpt, islda
hh(m1,m2,nkpt,islda)
```

One can used the small utility file plot_MDSP3.f to plot it out. The IN.WG1_MDSP3, IN.WG2_MDSP3 have the same format as IN.WG (if there are spin=2, then we also need IN.WG1_MDSP3_2, IN.WG2_MDSP3_2). But their number of wave functions m1, m2 can be much smaller than mx in IN.WG. They can be selected from OUT.WG from some previous runs (e.g, defect wave functions) using utility file: Select_WG.f90.

4. **MD_SPECIAL4**

   Format: MD_SPECIAL4 = iflag, Rc

   Default: MD_SPECIAL4 = 0

   This is also usually used together with **MD_SPECIAL=1**, but with solvent model. It excludes the solvent effect from the center of the x1,x2,x3 defined in the **MD_DPECIAL=1** line, with a radius of Rc (A unit). This is done by adding some charges at the center when defining the dielectric constant. This is used to prevent the solvent effects to intrude into the center region.

2.1.5.18 NAMD_SPECIAL

Format: NAMD_SPECIAL = iflag_NAMD_sp, param1, param2

Default: NAMD_SPECIAL = 0

This is an optional input section for special input parameters (param1,param2) for JOB=NAMD calculations. Default iflag_NAMD_sp=0, the parameters are not used.

When iflag_NAMD_sp=1, it will output the dipole moment matrix at every MD step in a file called OUT.NAMD_SP. It is a binary file, with the following format.

Do istep=1,nstep

write(OUT.NAMD_SP) istep,islda,nkpt,param1,param2,mst_win,time

Do iislda=1,islda

DO kpt=1,nkpt

write(OUT.NAMD_SP) PXYZ

ENDDO

ENDDO

ENDDO

Here PXYZ is a complex*16 matrix PXYZ(mst_win,param2-param1+1,3). mst_win=m_2-m_1+1 from NAMD_DETAIL, are the number of states output for NAMD calculation. Thus: $PXYZ(i,j,k) =< \phi_i|P_k|\phi_j >$, here k=1,2,3, and P_k is the momentum operator $i\nabla_k$.

2.1.5.19 CHARGE_DECOMP

CHARGE_DECOMP = T / F, imth

Default: CHARGE_DECOMP = F, 1

If set CHARGE_DECOMP = T, it will create atomic charge on each atom after each SCF iteration. For spin = 2, besides the atomic charge, the magnetic_moment = charge_up - charge_down will also be recorded.

The result of last SCF iteration will be stored in OUT.QDIV. Also, for JOB=RELAX/MD/TDDFT, the result of each SCF iteration will be reported in MOVEMENT.

This option is useful for charge analysis.

imth=1: When imth=1 (default), the atomic charge is generated by using the Hirshfield algorithm.

For the Hirshfield algorithm, the charge on one atom i is defined as: $Q_i = \int \rho(r) \frac{\rho_{atomtype(i)}(r-R_i)}{\sum_j \rho_{atomtype(j)}(r-R_j)} d^3r$, here $\rho_{atomtype(i)}(r)$ is the neutral charge density of the atom type atomtype(i) described in the atom pseudopotential file xxx.upf. Note, this option only works for norm conserving pseudopotential.

Please note, this method is quite expensive for large systems. So, it is time consuming if you want to do charge analysis during atomic relaxation, molecular dynamics, etc. In this situation, you might want to use imth=1, see below.

imth=2: When imth=2, the atomic charge is generated by simply add all the charge which is in the sphere of each atom.

The cutoff radius of each atom will be set to its default covalent radius. You can set covalent radius for each atom types by 'COVALENT_RADIUS', see 2.1.5.20).

This method is cheap and fast, but it only give the qualitative results, and it is hardly possible to determine charge transfer and the atomic magnetic moments. For a more accurate charge analysis, you can use imth=1, see above.

You can search for 'Volume ratio:' in screen output. It is the ratio of atom spheres volume to the cell volume. For mono-atomic bulk systems, you can modulate the cutoff radius util the ratio is close to 1. For systems consisting of more than one atom type, there is no unambiguous way to define the cutoff radius. So you can just use the default the covalent radius, it can not give quantitative results but it relies at least on some physical intuition.

2.1.5.20 COVALENT_RADIUS

COVALENT_RADIUS = value1, value2, value3 ...(number of atom types in total)

Default: None

Specify the covalent radius for each atom types. The order of these values must be consistent to the order of 'IN.PSP'.

Currently, these values are only useful when CHARGE_DECOMP = T, 2.

2.1.5.21 ENERGY_DECOMP*

This includes: **ENERGY_DECOMP**, **ENERGY_DECOMP_SPECIAL**, **ENERGY_DECOMP_SPECIAL1**, **ENERGY_DECOMP_SPECIAL2**, **ENERGY_DECOMP_COULOMB**

In the following, we explain them seperately.

ENERGY_DECOMP **ENERGY_DECOMP = T / F**

Or

ENERGY_DECOMP = T / F, type

Default:

ENERGY_DECOMP = F

type = 1

This will decompose the total DFT (LDA, PBE only, but it also works for IN.SOLVENT=T, as well as Poisson-Boltzmann equation) energies into the energies belong to each atom (atomic energies). The sum of the atomic energies will be equal to the total DFT energy (but differ by a constant, this constant is independent of the position of the atom, as well as the lattice length. Basically, each atom will miss an atom type specific energy constant, an onsite energy term). It rewrites the total energy as an spatial integral of the positive energy density term, and use the Hirshfield algorithm to decompose such energy density into each atom, much like the above decomposition for the charge density. See Ref.[J. Kang, L.W. Wang, Phys. Rev. B 96, 020302(R)(2017)] for details. For JOB = SCF, the decomposed energy will be reported in OUT.ENDIV. For JOB = MD, the decomposed energy will also be reported in MOVEMENT. These decomposed energies can be used to do force field fitting.

type=1,2,11,22 (default 1).

type=1,11: the electrostatic energy density is expressed as: $1/8\pi|E(r)|^2$ (here E is the electric field). This is positive everywhere, but it can have large amplitude even in vacuum region.

type=2,22: the electrostatic energy density is expressed as: $1/2\rho(r)V(r)$, here

$\rho(r)$ is the charge density including both electron and nuclear charge, V(r) is the total electrostatic potential. So, this only has values where $\rho(r)$ is not zero. Note, for all types, for IN.SOLVENT=T, the solvent polarization induced electrostatic energy density is always represented as $\rho_{solute}(r)V_{polarization}(r)$.

type=1,2: a straight forward Hirshfeld spatial partitioning is used to partition the energy density, to yield the energy for each atom. However, for the Hirsheld partitioning, the atomic charge (amplitude and shape) can be altered by the parameter in ENERGY_DECOMP_SPECIAL and ENERGY_DECOMP_SPECIAL2.

type=11,22: an atomic weight watom(iatom) is used, and dynamically adjusted, so when doing charge decomposition, it will yield the atomic charge equal to the neutral atom charge z(atom_type). This watom(iatom) is then used to do the energy decomposition. The hope is that, by getting the fixed charge density, the energy part (especially when using type=22) will have minimum variations. All these are designed to get the minimum fluctuation of the atomic energy. Note, there are additional costs by doing type=11,22, in order to find watom(iatom) for each atomic configuration. The watom(iatom) are listed in the last column in OUT.ENDIV, or the corresponding section in MOVEMENT.

Special note: If ENERGY_DECOMP_COULOMB=T, then type=1/2 are the same, and type=11/22 are the same. The detailed option will be determined by imth in the ENERGY_DECOMP_COULOMB line. Please check that section.

WARNING: ernergy decomposition does not support NUM_BLOCKED_PSI = T .

ENERGY_DECOMP_SPECIAL ENERGY_DECOMP_SPECIAL = w(1), w(2), ..., w(ntype)

Default:

ENERGY_DECOMP_SPECIAL = 1, 1, ..., 1

This is an additional option for ENERGY_DECOMP=T as well as CHARGE_DECOMP=T. This option modifies the weight for each atom type,

not just using $\rho_{atomtype(i)}(r)$, but using $w(atomtype(i)) * \rho_{atomtype(i)}(r)$ as the weight in the Hirshfeld method to calculate the charge and energy. The default values for all w(i) are 1.

ENERGY_DECOMP_SPECIAL1 ENERGY_DECOMP_SPECIAL1 = eta(1), eta(2), ..., eta(ntype)

Default:

ENERGY_DECOMP_SPECIAL1 = 1, 1, ..., 1

This is a very special input, please don't use it if you don't know the technical detail. In order to do the energy decomposition, for each input atom.UPF pseudopotential file, it will generate the corresponding: atom.UPF.ionrhoR, atom.UPF.rhoq, atom.UPF.rhoatom, files. They are the real space $v\_loc(r)$ file before and after fitting; q-space $v_{loc}(q)$ file before and after fitting, and $\rho(r)$ used for spatial partitioning function to generate the atomic quantities. It is worth to plot $v_{loc}(r)$, especially $v_{loc}(q)$. In $v_{loc}(q)$, for $0 < q < q_c/2$, it is the original $v_{loc}(q)$, while for $q_c/2 < q < q_c$ is the fitted one. Make sure the fitted one is close to the original one, there is no big variation. If there are large change, one can modify eta(itype). The default eta value is 1. Larger eta can make the $v_{loc}(q)$ smoother, but could be less accurate in some sense. One can set eta to 1.5 for example if $v_{loc}(q)$ for $q_c/2 < q < q_c$ is large.

This is an additional option for ENERGY_DECOMP=T as well as CHARGE_DECOMP=T. This option modifies the weight for each atom type, not just using $\rho_{atomtype(i)}(r)$, but using $w(atomtype(i)) * \rho_{atomtype(i)}(r)$ as the weight in the Hirshfeld method to calculate the charge and energy. The default values for all w(i) are 1.

ENERGY_DECOMP_SPECIAL2 ENERGY_DECOMP_SPECIAL2 = exp_decomp a(1) a(2) ... a(ntype) b(1) b(2) ... b(ntype)

Default:

ENERGY_DECOMP_SPECIAL2 = 1

This is used to control the atomic charge density $\rho_{atom}(|r-R|)$ to be used in the Hirshfeld formula.

$\rho_{atom}(|r-R|) = \rho_{atom}(|r-R|) + a(atomtype(R) * exp(-(|r-R|/b(atomtype(R)))^2))$

The together with the weight input in ENERGY_DECOMP_SPECIAL, the Hirshfeld partition function for atom R at position r, is given as:

$\rho_{atom}^{exp\_decomp}(|r-R|)w(atomtype(R))/\sum_{R'} \rho_{atom}^{exp\_decomp}(|r-R'|)w(atomtype(R'))$

(Note, there could be an additional weight watom(iatom) for type=11,22 partitioning method). So, higher value of exp_decomp (e.g., 2) can make the partitioning more local, and the interface more abrupt, but it can also be less smooth.

Note, the parameters in ENERGY_DECOMP_SPECIAL1, and ENERGY_DECOMP_SPECIAL2 can also be used to control the CHARGE_DECOMP.

ENERGY_DECOMP_COULOMB ENERGY_DECOMP_COULOMB = T / F, iconstr,imth,fac1,fac2,numG,q2W,iout

Default:

ENERGY_DECOMP_COULOMB = F

This is an option for Coulomb potential charge fitting. It can either do it on the flight (imth=1,2), or use an already fitted spherical atomic charge model (funcq_atom.fit), and recalculate the electrostatic Coulomb interaction energy. This will affect the electrostatic interaction involving both the electron charge and nuclear charge. The idea is to fit the electron charge density, and calculate the $rho_{fit} * rho_{fit}$ interactions analytically (using atom-center pair interaction), and we will have a residuation charge: $rho_{res} = rho - rho_{fit}$, and hopefully this residue is small, and it will not have long range interaction. In terms of fit, we can do on the flight, as in imth=1,2 (this is done for using 1 or 2 Gaussians for each atom, and either we have fitting, or we have no fitting, just use the previously fitted results), or we can have an more extensive fitting, using the vion_coulomb_fit.f90 utility file, to generate the funcq_atom.fit files to be read by the program.

Note, ENERGY_DECOMP_COULOMB = T does not work with Solvent model.

imth=1,2: fitting the charge with Gaussian on the flight. The idea is to use one or two (numG) Gaussians to represent a charge density at a given atom. **iconstr=1**: (this is only used under imth=1,2), one must provide a IN.CONSTRAINT_COULOMB to give information for constrains during the density fitting. This can be used to specified, the sum of the charge of a few atoms (or one atom) must be a given number. The fitted charge will output in: OUT.natom_coulomb_fit.

When ENERGY_DECOMP_COULOMB = T, one "ENERGY_NATOM_COULOMB" section must be provided in atom.config file. This section has the following format (note for imth=3, this section is not really used, but nevertheless, please provide an place holder faked section):

```
........ INSIDE THE ATOM.CONFIG FILE ............
ENERGY_NATOM_COULOMB
 4                        # natom_fit
 151  15  0.20 0.5        # atom order in xatom; zatom, a1, a2 (A)
 152   9  0.15 0.4        # atom order in xatom; zatom, a1, a2 (A)
 153   9  0.15 0.4
 154   9  0.15 0.4
 ------------------------
 150                                  # natom_fix
 1   8  0.3 0.5 -10.674, 11.914       # atom_order,zatom,a1,a2(A),Q1,Q2
 2   8  0.3 0.5 -10.674, 11.914       # atom_order,zatom,a1,a2(A),Q1,Q2
 3   8  0.3 0.5 -10.674, 11.914       # atom_order,zatom,a1,a2(A),Q1,Q2
 4   8  0.3 0.5 -10.674, 11.914       # atom_order,zatom,a1,a2(A),Q1,Q2
 ..... (150 lines)
```

Note, the a1,a2 are the size of the two Gaussian (even if numG=1, only one Gaussian is used, please provide two columns, for the place holder, the format is fixed). The atom_order is the index of the atom in the original xatom.config (Note, if the atoms are not in consequetive order for different atom types, this first index will be changed, to the index in output xatom.config file. The natom_fit is the atoms to be fitted (to get their charge parameters Q1,Q2), while the natom_fix is the atoms which already fitted before, thus already know the Q1,Q2 (the charge on these two Guassian functions).

natom_fit and/or natom_fix can be zero (for imth=3, one can set both to be zero). Note, if desired, it is okay for only fiting the charge for a few atoms, instead of all the atoms. Or one can first fit the Q1,Q2 from some other systems, then use them as natom_fix, and fit some additional atoms in this system.

numG=1, or 2: In above, in the section of ENERGY_NATION_COULOMB, there are always two Gaussians. Actually, one can adjust the number of Gaussian, numG=1 means only use 1 Gaussian (but there should still have two columns in the ENERGY_NATION_COULOMB section, only the first column is used). When numG=2, two Gaussians are used.

iconstr=1, the IN.CONSTRAIN_COULOMB need to be input. It can have the following format:

```
2                       # number of constrains (followed by num lines)
3  1.0 1,2,3            # Num_atom; Qtot, ind1,ind2,ind3,...
1 -1.0 4                # Num_atom; Qtot, ind1,ind2,ind3...
```

Note, each line is one constrant. Qtot is the total charge of these few atoms within this constraint. Ind1,ind2,ind3 are the atom number index of the atoms within the natom_fit sequence in the ENERGY_DECOMP_COULOMB section of the xatom.config file. Note, they are not the index in the original xatom.config atom sequence. They are index within the list of natom_fit (e.g., must be less or equal to natom_fit. For example, the 1, 2, 3 atoms in the above example correspond to atoms 151, 152, 153.

The fitted natom_fit Q1,Q2 results are shown in OUT.natom_coulomb_fit (together with the natom_fix Q1,Q2 results input within the ENERGY_NATOM_COULOMB section of xatom.config).

imth=1: In this method, $\rho_{res}(r) = \rho(r) - \rho_{fit}(r)$, note, $\rho(r) = \rho_{el} - \rho_{nuclear}$. So, the fitting is done for the total charge density (including nuclear), not just for the electron charge density. Then: Electrostatic potential density equals: $E_{elst}(r) = \frac{1}{8\pi}|\nabla V_{res}(r)|^2 + \rho_{fit}V_{res}(r)$ (excluding $\rho_{fit} * \rho_{fit}$ interaction). In the Hirshfeld parititiong, only the first term is partitioned, the second term is directly integrated for each atom, thus we have a $E(Q * V_{res})$ term (Q is the fitted charge), which is listed as one column in OUT.ENDIV.

To get the atom decomposed energy without new fitting(after excluding the fitted charge Coulomb interactions), one can set natom_fit=0 in the ENERGY_NATOM_COULOMB section of xatom.config file.

imth=2: In this option, the energy density (excluding the $\rho_{fit} * \rho_{fit}$ interaction) is expressed as: $\frac{1}{8\pi}[|\nabla V(r)|^2 fac1 - |\nabla V_{fit}(r)|^2 fac2]$. Here V(r) is the Coulomb potential of $\rho(r) = \rho_{el} - \rho_{nuclear}$, and V_{fit} is the Coulomb potential of ρ_{fit}. So, normally, **fac1 and fac2** should be one. They are provided here, just for the purpose of analysis. Note, in this option, there is no $E(Q * V_{res})$ term, and this column in OUT.ENDIV will be zero.

imth=3: In this option, there is no on-the-flight fitting. Instead, a prior fitted atomic charge will be used. In this option, all the atoms need to be fitted, not just for a selected subset. The fitted spherical charge density for each atom type in q-space is input from a file called: funcq_atom.fit. This funcq_atom.fit is obtained from the utility code vion_coulomb_fit.f90, based on the output charge density file OUT.rho_EpN ($\rho_{el} - \rho_{nuclear}$) from a previous ENERGY_DECOMP_COULOMB run. The electrostatic energy denisty is expressed as $\frac{1}{8\pi}|\nabla V_{res}(r)|^2 + \rho_{fit} V_{res}(r)$. But unlike in imth=1, the second term is not integrated for each atom, instead, it is included in the Hirshfeld partitioning. This imth=3 can significantly reduce the total electrostatic interaction energy (e.g, by a factor of 1000 in the case of melted NaCl).

q2w: a parameter (unit 1/A) used for a factor $exp(-(q * q2w)^2/4)$, this factor is used in the on-the-flight fitting of the Gaussian charge density to reduce the electrostatic energy: $\sum_q |\rho_{res}(q)|^2 * exp(-(q * q2w)^2/4) * 4\pi/q^2$. The idea is that, we can use this parameter to emphasize only the small reciprocal vector q components, hence only the long range part of the Coulomb interaction. Note, this factor is also used when generating output "OUT.Coulomb_EpN" and "OUT.Coulomb_residual". It can be used to filter out the high energy components.

iout= 0 or 1: 1 will mean there will be output (every MD step, rewrite) charge density: OUT.rho_EpN ($\rho_{el} - \rho_{nuclear}$) on a double grid; OUT.rho_EpN.fit (ρ_{fit}); OUT.Coulomb_EpN (the Coulomb potential of $\rho_{el} - \rho_{nuclear}$, subject to the $exp(-(q * q2w)^2/4)$ factor; OUT.Coulomb_residual (the COulomb potential of $rho_{res} =$

$(\rho_{el} - \rho_{nuclear}) - \rho_{fit}$ subject to the $exp(-(q * q2w)^2/4)$ factor. Note, these quantitites are double grid $(2n_1 \times 2n_2 \times 2n_3)$ quantities for Hirshfeld partitioning usage. iout=1 can be expensive.

It usually only used for JOB=SCF calculation, and it is can be used to generate OUT.rho_EpN to fit the funcq_atom.fit using vion_coulomb_fit.f90.

2.1.5.22 ATOMIC_ORBITAL_IATOM_OUT

Default:

NO DEFAULT

ATOMIC_ORBITAL_IATOM_OUT contains two parameters atomic_orb_iatom_i and atomic_orb_iatom_e which decide the atom index range(the index is from the IN.ATOM configuration file) for JOB=ATOMIC_ORB.

2.1.6 Corrections & constraints tags

2.1.6.1 VDW

VDW = NONE / DFT-D2 / DFT-D3 /PAIR

Default:

VDW = NONE

This parameter is used to specify the type of Van Der Waals correction.

If use DFT-D2, some variables are optional to be set: **LONDON_S6, LONDON_C6, LONDON_RCUT**. We use the Grimme's empirical vdw functional term. It is okay without setting the LONDON parameters.

LONDON_S6: Global scaling parameter for DFT-D. Default is 0.75.

LONDON_C6: It is an array its dimension is the number of atomic type. Its format is like this: LONDON_C6(1) = ..., LONDON_C6(2) = ...

(1),(2) are the atom types, in accordance with IN.PSP1, IN.PSP2. These are the C6 parameters in the Lennard-Jones potential $1/r^6$ term parameter. Only the attractive $1/r^6$ term is included in VDW. The repulsion part is already in the DFT energy. The

parameter LONDON_S6 determines trunction of this term at small r.

The default value is from the Grimme-D2 values. You can refer to the article: S. Grimme, J. Comp. Chem. 27, 1787(2006).

LONDON_RCUT: The cutoff radius (a.u.) for dispersion interactions calculations. The default is 200.0. For $|R1 - R2|$ larger than this cut-off, the vdW interaction will not be calculated. Note, this default value might be too large.

If use DFT-D3(zero-damping method), some variables are optional to be set: **DFTD3_S6, DFTD3_RS6, DFTD3_S18, DFTD3_RS18, DFTD3_ALPHA6, DFTD3_VERSION, DFTD3_3BODY**. For more information about the DFT-D3, one can refer to this paper [24].

In the D3 correction method, the following vdW-energy expression is used:

$$E_{disp} = -\frac{1}{2}\sum_{i=1}^{Natom}\sum_{j=1}^{Natom}\sum_{L}\left(f_{d,6}(r_{ij,L})\frac{C_{6ij}}{r_{ij,L}^{6}} + f_{d,8}(R_{ij,L})\frac{C_{8ij}}{r_{ij,L}^{8}}\right) \tag{2.11}$$

The dispersion coefficients C_{6ij} are geometry-dependent as they are adjusted on the basis of local geometry (coordination number) around atoms i and j. In the zero-damping method, damping of the following form is used:

$$f_{d,n}(r_{ij}) = \frac{s_n}{1 + 6(r_{ij}/(s_{R,n}R_{oij}))^{-\alpha_n}} \tag{2.12}$$

where $R_{oij} = \sqrt{\frac{C_{8ij}}{C_{6ij}}}$, the parameters $s_{R,8}$ are normally 1. Respectively, the $s_6, s_8, s_{R,6}, \alpha_6$ are adjustable parameters whose values depend on the choice of exchange-correlation functional. Note the default parameter is tested for PBE.

DFTD3_VERSION The parameter for zero-damping DFT-D3 method is 3.

DFTD3_3BODY The parameter controlling whether considering the three body term in DFT-D3 correction method. The default is T (considering). If not considering the term, setting DFTD3_3BODY = F.

DFTD3_CUTOFF The cutoff radius (a.u.) for pair interactions in real space. Default is 94.868 bohr.

DFTD3_CUTOFF_CN The cutoff radius (a.u.) for coordination number in real space. Default is 40.0 bohr.

VDW=PAIR In that case, the file IN.VDWPAIR need to provided. This is a general user provided pair potential in a numerical form, which has the following format:

```
1001, 3, 8.0       ! nr, n_pair, r_cut(angstrom)
31 31 33           ! iatom_1 (npair)
33 31 33           ! iatom_2 (npair)
0.00,  p1,  p2,  p3  ! r, pot(pair1), pot(pair2), pot(pair3) (eV)
...
r,   p1, p2, p3       ! The nr-th line for p1,p2,p3
```

2.1.6.2 COULOMB

Default:

COULOMB = 0

Control the Poisson equation solution (for the Coulomb interaction).

We provide special ways to calculate the Coulomb potential (also called Hartree potential) of the charge density rho(r) during the SCF calculation. This could be particularly helpful for isolated system calculation, or for slab calculation. This is to avoid electrostatic image interaction for isolated systems, or the dipole moment effect for neutral slab calculations.

COULOMB = 0, the periodic boundary condition, the default.

COULOMB = 1, X1, X2, X3: the isolated cluster boundary condition. It can avoid the image interaction in this calculation. The X1, X2, X3 (value: 0∼1) are the fractional coordination values in the unit cell edge vectors 1, 2, 3, used to cut a box for this special Coulomb solution. In the other word, the center of the box is at: (X1+0.5, X2+0.5, X3+0.5).

COULOMB = 11, X1: A slab calculation along the first direction, with the cut at X1. This can avoid the dipole moment interaction between slabs. Note, this method

only works for neutral system. For charged slab system, the energy is infinite for an isolated slab. In that case, please just use COULOMB=0.

COULOMB = 12, X2: A slab calculation along the second direction with the cut at X2. Only for neutral system.

COULOMB = 13, X3: A slab calculation along the third direction with the cut at X3. Only for neutral system.

2.1.6.3 LDAU_PSP

LDAU_PSP1 = LDAU_L(1), Hubbard_U(1) (optional Hubbard_U2(1))
LDAU_PSP2 = LDAU_L(2), Hubbard_U(2) (optional Hubbard_U2(2))

...

Default:

LDAU_PSP1 = -1 0 0

LDAU_PSP2 = -1 0 0

...

If this parameter is set, LDA+U method will be used. When using LDA+U method, one must specify, for each element(i), the atomic orbit to add U, and the value of U. Note the (i) should correspond to the IN.PSP(i) for the pseudopotential input.

LDAU_L(i) = -1/0/1/2/3, 0/1/2/3 means adding a U term to the s/p/d/f orbital. -1 means not to use LDA+U.

HUBBARD_U(i): the U parameter (eV) for species element types i, the default value is 0.0. If HUBBARD_U2(i) is also provided, then the first number is for spin up component, the second number is for spin down component. They can be different. If the second number is not provided, then the spin up and down U parameters will be the same.

Note, there are cases where the same atom type, say Co, needs to use different U parameter depending on its local environment and valence state (e.g., Co^{3+} and Co^{2+}). In such case, one can change the atom number of Co in the atom.config file, say, one is 27, another is 127. Also, one add another Co pseudopotential Co2.xxx.upf, and in

Co2.xxx.upf, change its atomic number from 27 to 127. Now, one can add different U for this new 127 Co type.

Besides, you should set proper LDAU_RCUT_PSP for the LDA+U calculation. The default value is 4 Bohr. Check the LDAU_RCUT_PSP section for more details.

Note, the LDA+U calculation is done by the following Hamiltonian:

$$H = H_{LDA} + \sum_{I,\sigma} \frac{U_{I,\sigma}}{2} Tr[n^{I,\sigma}(I - n^{I,\sigma})]$$

here I denote the atomic site, and σ is for the spin, and $n^{I,\sigma}(m1, m2)$ is an occupation matrix for atomic orbital ϕ_m as:

$$n^{I,\sigma}(m1, m2) = \sum_j < \psi_{j,\sigma}|\phi^I_{m1} >< \phi^I_{m2}|\psi_{j,\sigma} > occ(j, \sigma)$$

Here $\psi_{j,\sigma}$ is the Kohn-Sham orbital for spin σ and $occ(j, \sigma)$ is its occupation. In above $U_{I,\sigma}$ is provided by the LDAU_PSP1 lines.

However, we can also add another term, which is:

$$H = H_{LDA} + \sum_{I,\sigma} \frac{U_{I,\sigma}}{2} Tr[n^{I,\sigma}(I - n^{I,\sigma})] + \sum_{I,\sigma} \lambda_{I,\sigma} Tr[n^{L,\sigma}]$$

The parameters $\lambda_{I,\sigma}$ are provided in a special section LDAU_lambda in the atom.config file:

```
LDAU_lambda
 27  0.5  0.5   ! iatom, lambda_up, lambda_dn
 27  0.0  0.0
 ...........
 27  0.0  0.0   ! must have natom lines
```

Note, in this LDAU_lambda section in atom.config, even if not all atoms have LDA+U, you must provide natom (all atom) lines for every atom. For those atoms which do not has LDA+U (determined by the LDAU_PSPx lines), their corresponding lambda will not be used. So, if you want to use lambda, you must set the LDAU_PSPx

line for that atom type. You can set a very small U parameter for that atom type to reduce the first LDAU term in above H formula.

In the LDAU calculation, we will have output: OUT.LDAU_NS, which writes out the $n^{I,\sigma}$ matrix for each atoms which has the LDAU_PSPx line. One can use this $n^{I,\sigma}$ and $\lambda_{I,\sigma}$ to calculate U with the linear-response method (please check the corresponding PWmat module).

2.1.6.4 LDAU_RCUT_PSP1,2

LDAU_RCUT_PSP1= rcutu1
LDAU_RCUT_PSP1= rcutu2

The Rcut of each element type for the LDA+U calculation for the atomic wave function orbital. The unit is in Bohr, not Angstrom. This is like the IN.PSP_RCUTi, but instead of for the nonlocal projector, it is for the full atomic wave function used for LDA+U calculation. Note, usually, these rcutui should be bigger than rcuti. If no explicit input for LDAU_RCUT_PSPi is provided, the default value of 4 Bohr is used (which is rather large). Note, for LDA+U calculation, it might be necessary to use a large rcutui, e.g., 6, to get a fully converged and smooth force (e.g., for better RELAX convergence). However, if Ecut2 is very large, then a slightly smaller rcuti can be used. Specifically, if you find that the distance inter atom is smaller than LDAU_RCUT_PSP, you should decrease LDAU_RCUT_PSP, large LDAU_RCUT_PSP will cause the unreasonable results.

2.1.6.5 STRESS_CORR

STRESS_CORR = $num\_pw_1, energy_1, num\_pw_2, energy_2$

No default

As we know, the number of plane wavefunction has respond to the size of the lattice, i.e. different lattice will give different number of plane wavefunctions in the same cutoff. When running cell relaxation, the lattice will change. Correspondingly, the number of plane wavefunctions should also change. However, the change of the plane wavefunctions

will disturb the procedure of the relaxation. As a result, in DFT calculations, we keep the number of plane wavefunctions all the same. In order to overcome the problem in cell relaxation, we implement an stress correction. One can refer to the paper [23] for more details.

The steps to carry out the stress correction: Before running the cell relaxation, you should do two SCF calculations with different cutoff. Each calculation will give the *num_pw* (See "Weighted average num_of_PW for all kpoint" in REPORT), the *energy* (See "E_tot" in REPORT). Then continue doing the cell relaxation with these parameters setting STRESS_CORR.**WARNING:the difference of the two cutoff should not be too large, about 1~2Ryd.**

2.1.6.6 FIX_FERMI

FIX_FERMI = T/F, E_Fermi, mix_Q, drho_pulay

Default:

FIX_FERMI = F

This control indicates whether to use fix Fermi energy (fix electrode potential) calculation. The default is F (use fixed total charge). Note, for T, this is usually used together with IN.SOLVENT=T, and with POISSON_BOLTZMANN = T (for Poisson-Boltzmann screening) in the IN.SOLVENT file for electrochemistry grand cannonical calculations. The Poisson-Boltzmann screening is used, because in that case, the potential at far away place is defined as zero, so the absolute Fermi energy is well defined. This option should be used with care. The E_Fermi is the Fermi energy in the unit of eV (usually is negative). Note, in this case, the total number of electron NUM_ELECTRON will be adjusted automatically, and its input value will only be used as an initial value. mix_Q is a charge mixing parameter for the total charge. Small value will be more stable, but slower. Suggest to try 0.02. The drho_pulay decides when switching between pulay mixing and no mixig and its detail meaning is decided by the FIX_FERMI_PULAY_MIXING. Default value is 0, which means doing pulay mixing all the time. It is stable but might be slow when doing relax or md calculation

and we suggest to set it to be 0.5. Also, one might increase the smearing of Fermi-Dirac occupation function (in the line of SCF_ITER0) to make the calculation stable (e.g., 0.25 eV).

2.1.6.7 FIX_FERMI_PULAY_MIXING

FIX_FERMI_PULAY_MIXING = 1/2

Default:

FIX_FERMI_PULAY_MIXING = 1

It contrals the charge mixing type when doing fix fermi calculation (FIX_FERMI=T). If FIX_FERMI_PULAY_MIXING=1, pulay mixing will turn on in the beginning and then turn off when scf charge denstity error $\Delta\rho < RHO\_ERROR * (1 + drho\_pulay * nscf)$, where nscf means the current scf iterations. This charge mixing strategy will speedup the scf convenience when doing relax or md calculation and a proper drho_pulay will be 0.5. If FIX_FERMI_PULAY_MIXING=2, pulay mixing will turn off in the beginning and then turn on when scf charge denstity error $\Delta\rho < drho\_pulay$.

2.1.6.8 CONSTRAINT_MAG

CONSTRAINT_MAG = 0 / 1

Default:

CONSTRAINT_MAG = 0

This input line is used to put constraint on magnetic moment at each atom for spin=2 calculations. If CONSTRAINT_MAG=1, this will be turned on, if CONSTRAINT_MAG=0, this will not be used. Default is CONSTRAINT_MAG=0. Currently, it only works for spin=2. This is to add an energy penalty term: $\sum_{iatom} alpha\_mag(iatom)(M(iatom) - M_{input}(iatom))^2$ in the total energy expression. Note, M(iatom) is the magnetic moment calculated from spin up and down charge density using Hirshfield method, much like in the CHARGE_DECOMP. M_input(iatom) are the input magnetic moment in atom.config file, under a section

name: CONSTRAINT_MAG. alpha_mag(iatom) is also input from the atom.config from that CONSTRAINT_MAG section. Note, if this section is not provided in atom.config, the default value for M_input(iatom) is zero, and alpha_mag(iatom) is also zero. The unit of alpha_mag is in eV.

Alpha_mag should not be set to be too large. Otherwise the SCF iteration will be difficult to converge. One recommended alpha_mag is 0.01 eV. One can also set different alpha_mag for different atoms. Also, note that one can use the above additional energy to add an effective magnetic field H at each atom (with different amplitude etc). To do that, one can use a large M_input(iatom) in the direction one wants, in the mean time, uses a very small alpha_mag(iatom) which is proportional to 1/M_input(iatom). That will provide an effective H field on each atom.

2.1.6.9 SPIN222_MAGDIR_STEPFIX

SPIN222_MAGDIR_STEPFIX = N

Default:

SPIN222_MAGDIR_STEPFIX = 0 (for XCFUNCTIONAL = LDA)

SPIN222_MAGDIR_STEPFIX = 1000 (for everything else)

This input line is used to fix the direction of magnetic moment after SPIN222_MAGDIR_STEPFIX self-consistent iterations. If SPIN222_MAGDIR_STEPFIX = 30, it means that after 30 self-consistent interations, the direction of magnetic moment will be fixed. But for XCFUNCTIONAL = LDA, default SPIN222_MAGDIR_STEPFIX is 0, the direction of magnetic moment will not be fixed. If you set SPIN222_MAGDIR_STEPFIX = 1, initial direction of magnetic moment will be fixed, and will not change with self-consistent interations. Please note that SPIN222_MAGDIR_STEPFIX is only used for SPIN = 222.

2.1.6.10 E_FINITE

E_FINITE = T / F Ex Ey Ez

Default:

E_FINITE = F 0 0 0

This parameter is used to set the homogeneous electric field in the electric enthalpy functional.[31] If the first parameter is T, it will compute the self-consistent response to finite electric fields. Ex,Ey,Ez in unit eV/Angstrom.

In the output of screen, one can search for "Pel", the electronic dipole moment, in unit e*Angstrom.The first "Pel" is the result with fields turned off. The second "Pel" is the result with fields turned on.

An example of AlAs, the file atom.config:

```
2
LATTICE
4.0543775558          0.0000000000          0.0000000000
2.0271887779          3.5111939599          0.0000000000
2.0271887779          1.1703979866          3.3103854121
POSITION
13 0.000000000        0.000000000         0.000000000   1 1 1
33 0.749999998        0.750000011         0.749999994   1 1 1
```

the file etot.input:

```
4  1
JOB = scf
IN.PSP1 = Al.SG15.PBE.UPF
IN.PSP2 = As.SG15.PBE.UPF
IN.ATOM = atom.config

Ecut    = 50
Ecut2   = 200
MP_N123 = 4 4 4  0 0 0 2

e_finite  = T 0.00 0.00 0.001

precision = double
e_error   = 0.0
wg_error  = 0.0
rho_error = 1.d-6
```

```
out.force = t
```

The OUT.FORCE=T is needed to calculate born effective charge, for accurate forces, Ecut2=4*Ecut is recommended.

Kpoints are set by MP_N123 without symmetry(E_FINITE=T can not use symmetry). Kpoints grid should be large enough to get converged results.

PRECISION=DOUBLE is set for more accurate results, much more here we set e_error=0, wg_error=0, rho_error=1.d-6 to ensure the wavefunctions in good convergency.

For systems with small gap, one need to use smaller FremidE, also need to use much smaller strength of field in case of no local minimum, like following example, the file atom.config:

```
2
LATTICE
     4.06599283     0.00000000     0.00000000
     2.03299642     3.52125308     0.00000000
     2.03299642     1.17375103     3.31986925
POSITION
 31     0.00000000     0.00000000     0.00000000 1 1 1
 33     0.25000000     0.24999999     0.25000000 1 1 1
```

the file etot.input:

```
4  1
JOB = scf
IN.PSP1 = Ga.SG15.PBE.UPF
IN.PSP2 = As.SG15.PBE.UPF
IN.ATOM = atom.config

Ecut    = 50
Ecut2   = 200
# for correct and converged results, maybe need to use more kpoints
MP_N123 = 4 4 4  0 0 0 2
e_finite  = T 0.00 0.00 0.0001
```

```
precision = double
e_error   = 0.0
wg_error  = 0.0
rho_error = 1.d-6

out.force = t

#use FermidE=0.001eV
SCF_ITER0_1 =    6   4    3    0.0000     0.0010    1
SCF_ITER0_2 =   94   4    3    1.0000     0.0010    1
```

2.1.6.11 RVV10_DETAIL

RVV10_DETAIL = b, c

Default:

RVV10_DETAIL = 6.3, 0.0093

The current code implemented the RVV10 dispersion interaction (G. Roman-Perez, J.M. Soler, Phys. Rev. Lett. 103, 096102 (2009); R. Sabatini, T. Gorni, S. de Gironcoli, Phys. Rev. B, 87, 041108 (2013)). The RVV10 use a nonlocal integral of charge densities at different points, r, r' derived from RPA formalism. The kernel of this integral is

$$\phi^{VV10}(r,r') = -\frac{3e^4}{2m^2}\frac{1}{gg'(g+g')}$$

Here $g = \omega_0(r)(r-r')^2 + k(r)$, $g' = \omega_0(r')(r-r')^2 + k(r')$. Furthermore, $\omega_0(r) = \sqrt{\omega_g^2(r) + \omega_p^2(r)/3}$. $\omega_p^2(r) = 4\pi n(r)e^2/m$ is the plasma frequency.

$\omega_g^2(r) = c(\hbar^2/m^2)|\frac{\nabla n(r)}{n(r)}|^4$ and $k(r) = 3\pi b(\frac{n(r)}{9\pi})^{\frac{1}{6}}$, Here b and c are parameters. For default, b=6.3, c=0.0093. But one can use RVV10_DETAIL to specify different b and c values. This is only used when XCFUNCTIONAL contains RVV10.

2.1.6.12 RELATIVITY

RELATIVITY = T/F

Default:

RELATIVITY = F

This option controls whether relativistic effects are considered. When RELATIVITY=T, relativistic effects are taken into account, and in this case, PRECISION=DOUBLE must be used. This option is experimental, so use it with caution.

2.1.6.13 QIJ_DETAIL (obsoleted)

QIJ_DETAIL = QIJ_PD, QIJL0_GS

Default:

QIJ_DETAIL = 0 1

Only relevant for ultrasoft pseudopoential. This is a bit obsolete, usually don't use it.

Note: normally, should not be used by user (unless the user knows exactly what he is doing), just take the default values (or not to write this line in etot.input).

QIJ_PD = 0/1: It controls whether to include the l=p and d angular momentum core charge density QIJ(r,l) in the ultrasoft pseudopotential calculation. The default is 0, which means only the s angular momentum core charge density is used. We strongly recommend the use of 0. The difference is usually very small, but the including of p and d components of the QIJ(r,l) can make the energy manifold unsmooth, thus makes the JOB=RELAX difficult.

QIJL0_GS = 1/2: This parameter is only used by ultrasoft pseudopotential calculations. It controls the core charge implementation in ultrasoft pseudopotential. 1 means the s-component of the core Qij charge is used and implemented in G-space; 2 means the s-component of the core Qij charge is implemented in real-space. The default is 1. We strongly recommend the use of default value since QIJL0_GS=2 might introduce some jitter in the energy curve.

2.1.7 I-O tags

2.1.7.1 IN.ATOM

Format: IN.ATOM = atom.config

Default: None

IN.ATOM is used to read the atomic positions file, this file contains the lattice geometry and ionic positions, optional tags – force, velocity, magnetic, constraint_mag, magnetic_xyz, langevin_atomfact_tg, stress_mask, et al. Its specification is described in the section 2.2 of this manual.

2.1.7.2 IN.PSP

Format: IN.PSP1/2/3/... = PSPFilename

IN.PSP1 = H.NCPP.UPF

IN.PSP2 = C.NCPP.UPF

...

Default: None

The names of the pseudopotential files. 'IN.PSP1' is the first atom type, 'IN.PSP2' is the second atom type, and the rest can be deduced by analogy. The order of different element types is arbitrary.

Please see section 2.3 for a discussion of different pseudopotentials.

2.1.7.3 IN.KPT

IN.KPT = T / F

Default:

IN.KPT = F

IN.KPT = T, PWmat will use the k-points from file "IN.KPT" which contains the k-points and their weights. Note, IN.KPT, IN.SYMM usually work together. IN.KPT has a higher priority than MP_N123. If IN.KPT = F, PWmat will not use the file

"IN.KPT". PWmat will always output koints in "OUT.KPT" file. Please check the IN.KPT(OUT.KPT) subsession for more detail format about this file.

2.1.7.4 IN.SYMM

IN.SYMM = T / F

Default:

IN.SYMM = F

IN.SYMM = T, PWmat will use the file "IN.SYMM" (the name is fixed) to perform symmetry operations. The PWmat supports space group symmetry for crystals. "IN.SYMM" should contain space group symmetry operations. Usually, symmetry can be generated automatically by using MP_N123 line. However, one can also copy over the previous OUT.SYMM into IN.SYMM for explicit input (e.g., one can even delete some symmetry operations). The IN.SYMM usually should work together with IN.KPT (for the reduced k-points). Note, if both IN.SYMM=T, IN.KPT=F are specified, and also MP_N123 are also specified, PWmat will generate Kpoints using the symmetry operations provided in file "IN.SYMM". If IN.SYMM=T, IN.KPT=T and also MP_N123 are also specified, the IN.KPT=T has a higher priority, MP_N123 will not be used. If IN.SYMM = F, PWmat will not use file "IN.SYMM", This is the default value. PWmat will always output symmetry operations in "OUT.SYMM" file. Please check the IN.SYMM(OUT.SYMM) subsession for more detail format about this file.

2.1.7.5 IN.OCC

IN.OCC = T / F, iproj (iproj = 0 / 1 / 2 / 22 / 3 / 33)

Default:

IN.OCC = F

Related items: PROJ3_DETAIL, IN.iproj3_CC_2spin, IN.OCC_T, IN.CC. In all these options, the PWmat will try to use special ways to determine the occupation of wave functions, or eigen states, instead of using the conventional Fermi-Dirac distribution from the SCF_ITER0_1, SCF_ITER0_2, SCF_ITER1_1 lines. Note, in general, not

necessarily the eigen states will be occupied, instead an linear combination of the eigen states can be occupied. These options will be particularly useful for either constraint DFT (with some excited electron states, or empty hole states) SCF or RELAX jobs, or TDDFT simulations. For example, it can be used to prepare some special initial excited state in TDDFT.

In this option, PWmat will read a file called "IN.OCC". This is to specify the occupation for each Kohn-Sham orbital ϕ_j for a constraint DFT calculation. **WARNING:"IN.OCC=T" is equivalent to: "IN.OCC=T,0"**

In the following, we will use ϕ_i to denote the adiabatic eigen state of the Kohn-Sham Equation: $H\phi_i = \epsilon_i\phi_i$. In the meantime, we can input another set of wave function: $\{\psi_j\}$, e.g, through the IN.WG=T option. This $\{\psi_j\}$ is a bit like the time evolving wave functions in TDDFT, and they can be used to help the occupation of states to generate the charge density.

The following are the different options for iproj.

iproj=0: (IN.OCC = T, or IN.OCC = T, 0): The SCF calculation charge density will be generated as: $\rho(r) = \sum_i o(i)|\phi_i(r)|^2$, and the occupation number $o(i)$ will be input from the IN.OCC file (see the explanation below). Note, for iproj=0, the o(i) is fixed (for which state is which). For example, if the third state in IN.OCC is empty (o(3)=0), then during the SCF calculation, or atomic relaxation, it is always the third adiabatic state which is unoccupied. This is okay for simple cases, but for more complicated cases, it can cause problem, since during the SCF iteration, or atomic relaxation, the order (which state is the third) according to the adiabatic state eigen energies can often change (re-ordered). As a result, the third state might not be the physical state you like to keep it empty. One option in that case is to use iproj=1.

iproj=1: In this option, the index of which state is which is determined by a projection between the adiabatic eigen state ϕ_i and the input state ψ_j from IN.WG. So, in order to use this, one has to have IN.WG. More specifically, map(i) equals the j which maximizes $| < \phi_i|\psi_j > |^2$. Basically, this identifies which ϕ_i is the input $\psi_{map(i)}$. As a result, the charge density is generated as: $\rho(r) = \sum_i o(map(i))|\phi_i(r)|^2$. Note, ψ_j

is input from IN.WG. Once again, o(j) is input from IN.OCC, which has the following form (for iproj=0,1,3):

```
o1,o2,o3,.....o_mx        ! for kpt=1
o1,o2,o3,.....o_mx        ! for kpt=2
........
o1,o2,o3,.....o_mx        ! for kpt=nkpt
```

Here mx is the num_band, and nkpt is the number of reduced kpoint. Thus, in each line, there are mx number, and there are nkpt line. Note, one can write something like: 4*1,2*0 to replace: 1,1,1,1,0,0. Note, o(j,kpt) is the occupation number discussed above (it must be between 0 and 1, even for SPIN=1, should not be 2). Note, the full occupation is o(j,kpt)=1. So, if SPIN=1, o(j,kpt)=1 means this orbital will occupy 2 electron, and o(j,kpt)=0.5 means this orbital will occupy 1 electron.

If SPIN=2, then one has also to provide an file: IN.OCC_2, which has the same format, but specify the spin down component occupation. In that case, o(j,kpt)=1 means this orbital (up or down spin) will occupy one electron, and o(j,kpt)=0 means this orbital of this spin will occupy zero electron.

iproj=2 (or 22): in the above case of iproj=0 and 1, the Fermi-Dirac distribution function will be ignored (not used, or it is like Fermi-Dirac=0). But even for iproj=1, there could be cases where it is difficult to identify which ϕ_i is ψ_j, for example, if several states have the similar amplitude overlaps (e.g., around 0.3-0.5). In that case, even if we select and occupy one ϕ_i, the results might not be good either. What we really need is to construct one ψ_j-like wave function ψ'_j from a linear combination of ϕ_i, then occupy ψ'_j. More specifically, we can have: $\psi'_j = \sum_i f(i,j) < \phi_i|\psi_j > \phi_i$. Here f(i,j) is a selection and orthonormalization factor. For iproj=22, there is no selection, and f(i,j) is just for orthonormalization (among all the $\{\psi'_j\}$). For iproj=2, a selection weight factor is used for different i. More specifically, $f(i,j) = exp(-((1-x)/0.7)^6)$, where $x = | < \phi_i|\psi_j > |^2$. Thus, effectively, it only select the ϕ_i states with overlap larger than 0.3. After this selection factor, it is orthonormalized. The idea here is only to select a few ϕ_i, and use their linear combination to construct a state resemble that of the original

ψ_j state, but not to completely reconstruct the ψ_j state. This will be useful to deal with the case where two states anticross each other, so the ϕ_i character has changed, but a linear combination can reconstruct the original ψ_j.

In terms of occupation, it uses a different strategy than iproj=0,1,3. For iproj=2, the Fermi-Dirac distribution is still used, then on top of it, an exception for some band is used to add or substract some states. The idea is to use this to add one excited electron or subtract one hole (on ψ_j) (called exception states) from the otherwise Fermi-Dirac calculation. Note, when decide the Fermi energy, a total charge of NUM_ELECTRON - dcharge is used, and dcharge is the charge from ψ_j as specified in the IN.OCC to be explained below. So, the actual total charge is NUM_ELECTRON. Also note, not only the charge density of ψ_j is contributed, also its kinetic and nonlocal potential part of the energy. The total charge density can be specified as: $\rho(r) = \rho_{FD}(r) + \sum_j o(j)|\psi'_j(r)|^2$, here $\rho_{FD}(r)$ is the Fermi-Dirac charge density, and o(j) is the occupation from the IN.OCC, which has the following form in the case of iproj=2(or 22):

```
o1,o2,o3,.....o_mx        ! for kpt=1
o1,o2,o3,.....o_mx        ! for kpt=2
........
o1,o2,o3,.....o_mx        ! for kpt=nkpt
-------- (above nkpt lines are not used, this dashed line must be here)
nump                          ! The number of exception states
iband(1,1),od(1,1)            ! for indx=1,   kpt=1
iband(2,1),od(2,1)            ! for indx=2,   kpt=1
......
iband(nump,1),od(nump,1)      ! for indx=nump,kpt=1
iband(1,2),od(1,2)            ! for indx=1,   kpt=2
iband(2,2),od(2,2)            ! for indx=2,   kpt=2
......
iband(nump,2),od(nump,2)      ! for indx=nump,kpt=2
......                        ! repeat for different kpoint
......
iband(1,nkpt),od(1,nkpt)            ! for indx=1,   kpt=nkpt
iband(2,nkpt),od(2,nkpt)            ! for indx=2,   kpt=nkpt
......
```

```
iband(nump,nkpt),od(nump,nkpt)      ! for indx=nump,kpt=nkpt
```

Note, in this file, there must be nump*nkpt lines after the first nump+1 lines. The nump is the number of exception states in each kpoints. Each of the nump*nkpt line should have two numbers. The first number is the band index, the second number is an occupation (can be positive, or negative (hole)).

Note, for SPIN=2, one needs to have the same format for IN.OCC_2.

Note, iproj=0,1,2 are often used for both SCF calculation, as well as RELAX. iproj=0 is also often used for TDDFT calculation.

For the cases of iproj=1 for RELAX, the ψ_j for next RELAX step is updated from the $\phi_{map(j)}$ from the previous relaxation step. For the case of iproj=2 for RELAX, ψ_j for next RELAX step is replaced with ψ'_j from previous RELAX step. Doing this will allow one to relax the system even if there is a state crossing for a hole state (or electron state), while tracking and keeping the same hole or excited electron.

The iproj=2 is usually more stable than iproj=1 for both SCF and RELAX. So, it should be used if iproj=0 and 1 are unstable, and do not converge.

iproj=3 (or 33): this is used for SCF or TDDFT calculations, not for RELAX. In iproj=2, the exception is used to calculate the charge density. It is mostly designed to deal with RELAX, and conceptually, we like to eventually make the ψ_j the adiabatic eigen states. There are cases where we are interested in occupy specific input wave function ψ_j, but they are not the adiabatic eigen states ϕ_i. More importantly, the occupied state ψ'_j should be completely represented by ϕ_i. So, the occupied state is not the exact ψ_j input from IN.WG. If the fixed ψ_j is to be occupied, one can use JOB=WKM. Here, we will occupy $\psi'_j = \sum_i f(i) < \phi_i|\psi_j > \phi_i$, then orthonormalized following this formula. Thus, $\rho(r) = \sum_j o(j)|\psi'_j(r)|^2$, here o(j) is input from IN.OCC. In the formula, f(i) is used to provide some possible truncation (so there is no high energy ϕ_i components for ψ'_j. This can be provided by the following input line:

PROJ3_DETAIL = Ecut_proj3, dEcut_proj3 (in unit of eV)

Then: $f(i) = 1/(exp((\epsilon(i) - Ecut\_proj3)/dEcut\_proj3) + 1)$

If the PROJ3_DETAIL does not exist, then f(i)=1

Also, in the IN.OCC, if one o(j) is negative, then $\psi'_j = \phi_j$, and $o(j) = |o(j)|$.

This can be used to provide the initial state for TDDFT calculation (can be used together with JOB=TDDFT). So, the wanted initial TDDFT $\psi_j(t = 0)$ wave function (whatever wave function, not necessarily eigen states) can be input from IN.WG. But for our TDDFT implementation, we need to gaurantee the ψ_j can be represented by the set of eigen states $\{\phi_i\}$. This IN.OCC and the first step in TDDFT (or the SCF calculation) can just gaurantee this through a self-consistent iteration. Note, ψ_j can be some localized state, or some ionic states for the ion far away from a colliding surface. One can use other means to pre-construct IN.WG. Note, this iproj=3 will automatically output a file OUT.iproj3_CC_2spin, it contains the coefficient: $CC(i,j) =< \phi_i|\psi_j >$ with orthonormalization. For iproj=33, this OUT.iproj3_CC_2spin must be copied into IN.iproj3_CC_2spin, and inside etot.input, one has to place one line: IN.iproj3_CC_2spin=T. In this case, the IN.WG input is actually the eigen state ϕ_i instead of ψ_j, and ψ_j is constructed as: $\psi_j = \sum_i CC(i,j)\phi_i$. In this way, one can input both the eigen state and the ψ_j, thus continue a iproj=3 calculation.

Comparison with other options: Note, for TDDFT calculation, one can compare IN.OCC=T,3 calculation with IN.OCC_T=T calculation. In IN.OCC_T, the occupation of each state o(j) can be changed with time, specified inside IN.OCC_T. That is a powerful tool to actually change the occupation (e.g., to describe one electron gradually disappear due to some other physical process). It can also be used to prepare the initial state in a TDDFT calculation, e.g, quickly remove one state. But this is less general than IN.OCC=T,3, it can also has some stability issues. Also note, if one very quickly remove one electron through IN.OCC_T, the physical meaning might be different from the initial condition propered using IN.OCC=T,0. In IN.OCC=T,0, the initial condition is in a equilibirium condition, but that is not the case for IN.OCC_T calculation.

Lastly, one can also compare IN.OCC=T,3 with IN.CC. IN.CC can also be used to prepare a mixing state as the initial state for TDDFT calculation. It is easy to construct

the initial wave function, but it is less general, and less powerful than the IN.OCC=T,3 procedure.

2.1.7.6 IN.OCC_T

If we have:

IN.OCC_T = T

Then, it will read in file 'IN.OCC_T' (and 'IN.OCC_T_2' if spin=2), and it will ignore the imth_Fermi flag In the SCF_ITERx_x line.

The explanation of 'IN.OCC_T' file is in section 2.4.4.

2.1.7.7 IN.NONSCF

IN.NONSCF= T / F

Default:

IN.NONSCF = F

This parameter is used to set optional NONSCF parameter in file IN.NONSCF.

See more in 2.4.5.

2.1.7.8 IN.RELAXOPT

IN.RELAXOPT= T / F

Default:

IN.RELAXOPT = F

This parameter is used to set optional RELAX parameter in file IN.RELAXOPT.

See more in 2.4.6.

2.1.7.9 IN.MDOPT

IN.MDOPT= T / F

Default:

IN.MDOPT = F

This parameter is used to set optional MD parameter in file IN.MDOPT.

If the method of MD is 2,3,4 or 5, one can use the file IN.MDOPT to set detailed parameters by setting IN.MDOPT=T. See more in 2.4.7.

2.1.7.10 IN.TDDFTOPT

IN.TDDFTOPT= T / F

Default:

IN.TDDFTOPT = F

This parameter is used to set optional TDDFT parameter in file IN.TDDFTOPT. See more in 2.4.11.

2.1.7.11 IN.EXT_FORCE

IN.EXT_FORCE= T / F

Default:

IN.EXT_FORCE = F

This parameter is used to provide an external force (unit eV/amstrong) for each atom during MD simulation. See more in MD_DETAIL. If IN.EXT_FORCE=T, a file IN.EXT_FORCE will be provided, it has the following format:

```
natom
iatom, fx, fy, fz    ! unit eV/Amstrong
.................
iatom, fx, fy, fz    ! There will be natom lines
```

2.1.7.12 IN.SOLVENT

IN.SOLVENT = T / F

Default:

IN.SOLVENT = F

When calculating the energy of a solute molecule in a liquid solvent, e.g., in electric chemistry study, there are two possible approaches. One is to use explicit solvent molecule (e.g., water molecules) and carry out molecular dynamics simulations, another is to use implicit solvent models. The implicit solvent model represents the effect of the solvent with a continuum mediate, mostly includes its effects of electric static polarization. Compared with the explicit solvent molecules and molecular dynamics, the implicit solvent model is much faster. We have followed the work of self-consistent continuum solvation (SCCS) model [25], as well as similar formalism in Ref.[26]. In quantum chemistry, this is also called: polarizable continuum model (PCM). We have also implemented the linearized Poisson-Boltzmann screening for the effects of free ions (salt, or H+, OH- in low or high pH value situations) [27]. These models use a continuum mediate and a space variation dielectric constanr $\epsilon(r)$ to represent the solvation effects, mostly the polarization effects. There are three energy terms: the polarization solvation energy, the cavity energy (the surface tension, or can also be considered as surface van der Waals energy), and volume energy (pressure energy, PV). If Poisson-Boltzmann equation is used to describe the ion screening, there is another term which describes the ion energy within an electric static potential. When the Poisson-Boltzmann equation is used, the absoulte potential zero is defined at the far away place (there is no ambiguity of the absolute potential position). This also allows us to define the absolute potential of the electrode (for example, the standard hydrogen electrod, SHE, potential in water is at -4.42 eV). In this situation, we can use a fixed potential (fixed Fermi energy calculation). This is controlled using Fix_Fermi = T in etot.input (please see the corresponding item in the manual.

IN.SOLVENT = T, will use solvent model. Note, all the other input in etot.input will be the same. In other words, solvent model can be used to do single SCF, RELAX, MD and TDDFT calculations. However, one has to prepare a "IN.SOLVENT" file in the running directory, which control the parameters for the solvent model. See more in 2.4.9.

When IN.SOLVENT=T, after the PWmat run, a OUT.SOLVENT file will be

generated which lists all the options used for the solvent model.

2.1.7.13 IN.A_FIELD

IN.A_FIELD= T / F, a_field1, a_field2, a_field3

Default:

IN.A_FIELD= F 0.0 0.0 0.0

Note: for PWmat version later than 20200824, it has a new format:

IN.A_FIELD_LIST1= a_field1, a_field2, a_field3 IN.TDDFT_TIME1

IN.A_FIELD_LIST2= a_field1, a_field2, a_field3 IN.TDDFT_TIME2

...

the maximum support is 20 rows. This can be used to add a circularly polarized light. IN.TDDFT_TIME1,IN.TDDFT_TIME2... is the name of TDDFT_TIME file. You need to prepare same number of TDDFT_TIME files, it has the same format as IN.TDDFT_TIME,

```
0 ftddft(0)
1 ftddft(1)
...
N ftddft(N)
```

This controls the G-space external potential input for tddft calculation. (only used when TDDFT_SPACE=-1,...)

The tddft hamiltonian,

$$H = 1/2(-i\nabla_x + a\_field1)^2 + 1/2(-i\nabla_y + a\_field2)^2 + 1/2(-i\nabla_z + a\_field3)^2 \tag{2.13}$$

The values of $a\_field1, 2, 3$ are all in units of 1/Bohr.

2.1.7.14 IN.WG

IN.WG = T / F

Default:

IN.WG = F

IN.WG = T, PWmat will read in the initial wave functions in G-space from the file "IN.WG" (e.g., from previous calculation, copied over from OUT.WG, but note the node1 from the current calculation must be the same as in the previous calculation to generate OUT.WG). When SPIN = 2, an extra file "IN.WG_2" will also be read in. Note, IN.WG, OUT.WG can be plotted using utility program "plot_wg.x", which can be used to view each wave function. For people like to see the format of IN.WG, OUT.WG, one can check the plot_wg.f90 file which is source code of "plot_wg.x" utility program. If IN.WG = F, the PWmat will start from random wave function.

2.1.7.15 IN.RHO

IN.RHO = T / F

Default:

IN.RHO = F

IN.RHO = T, PWmat will read in the initial charge density from file "IN.RHO", stored in the real space grid (N1L, N2L, N3L). This can be copied over from OUT.RHO of previous calculation. Note, the node1 in current calculation, and previous calculation to generation OUT.RHO must dividable from one way or the other. Note, if both IN.VR and IN.RHO are set to T, the program will use the read-in potential to start the calculation. If SPIN = 2, PWmat will read an extra file "IN.RHO_2". IN.RHO=T is also needed for JOB=POTENTIAL. If SPIN = 22, only a single IN.RHO will be needed. If SPIN = 222, besides IN.RHO, a IN.RHO_SOM (a complex 2x2 spin matrix density) will be needed. If IN.RHO = F, not input the charge density.

One can use utility program: convert_rho.x to plot OUT.RHO or IN.RHO. One can also check convert_rho.f90 for the format of IN.RHO, OUT.RHO.

2.1.7.16 IN.RHO_ADD

IN.RHO_ADD = T / F

Default:

IN.RHO_ADD = F

IN.RHO_ADD = T, PWmat will read in an additional charge density ρ_{add} from input file: IN.RHO_ADD, and this ρ_{add} will be added to the total charge density calculated from the wave functions. In another word, $\rho(i) = \sum_i |\psi_i(r)|^2 occ(i) + \rho_{add}(r)$. Note, this ρ_{add} will not be counted as part of NUM_ELECTRON when determing $occ(i)$. This function can be used for many different algorithms. In particular, JOB=SCFEP is one particular case of this (but please continue to use JOB=SCFEP). Note, one can use the utility file: convert_wg2rho.f to general IN.RHO_ADD file from the output wave function file OUT.WG. When spin=2, one also needs to provide an IN.RHO_ADD_2 file for spin down additional charge density. If IN.RHO_ADD = F, no additional charge density is used.

2.1.7.17 IN.VR

IN.VR = T / F

Default:

IN.VR = F

IN.VR = T, PWmat will read in the initial potential from file "IN.VR" in real space grid: (N1L, N2L, N3L). Note, if both IN.VR and IN.RHO are set to T, the program will use the read-in potential to start the calculation. The format, and requirement of IN.VR are the same as that for IN.RHO. When SPIN=2, an extra file "IN.VR_2" will also be read. When SPIN=22, just a single IN.VR will be read (no IN.VR_2). When SPIN=222, besides IN.VR, IN.VR_SOM (complex 2x2 spin matrix potential), and IN.VR_DELTA (a single real up-down potential) need to be read in. If IN.VR = F, not read in the file.

2.1.7.18 IN.VEXT

IN.VEXT = T / F

Default:

IN.VEXT = F

IN.VEXT = T, PWmat will read in an external potential from file "IN.VEXT" in real space grid (N1L, N2L, N3L). Both the total energy and forces are calculated using this external potential. This can be useful to calculate the influence of an external potential (e.g., an electric field), for JOB=SCF, RELAX or MD. Note, the IN.VEXT has the same format as that in IN.RHO and IN.VR. Its unit is Hartree. One can try (and check, modify) the utility programs: calculate_Vext.f90 and gen_external_efield.f90 to generate such IN.VEXT. One can also write such IN.VEXT by self-made codes. If IN.VEXT = F, no external potential is used.

2.1.7.19 IN.LDAU

IN.LDAU = T / F

Default:

IN.LDAU = F

This is mostly for LDA+U JOB=NONSCF calculation. But for JOB=SCF calculation, and for a continued run, this can also be used (together with IN.WG and IN.RHO) to input the initial LDAU occupation number, so a smooth continue run can be carried out. PWmat will read in the initialization of LDA+U from file "IN.LDAU" (Note this is not the U parameters, instead they are the LDA+U occupation values calculated from JOB=SCF calculation). This setting is only required when JOB=NONSCF. One needs to copy OUT.LDAU to IN.LDAU after the JOB=SCF. If SPIN = 2, PWmat will read an extra file "IN.LDAU_2", also there will be OUT.LDAU_2 after JOB=SCF. If IN.LDAU = F, not input the initialization of LDA+U.

2.1.7.20 OUT.WG

OUT.WG = T / F

Default:

OUT.WG = T

If OUT.WG = T, PWmat will output a file "OUT.WG", which stores the final wave

functions in G-space. When SPIN = 2, an extra file “OUT.WG_2” will also be output. This is the default value. Use utility program “plot_wg.x" to plot the wave functions. More details about “plot_wg.x”, please refer to PWmat website http://www.pwmat.com/utility-download.

If OUT.WG = F, will not output the wave function file.

2.1.7.21 OUT.RHO

OUT.RHO = T / F

Default:

OUT.RHO = T

If OUT.RHO = T, PWmat will output a file “OUT.RHO”, the final charge density in real space grid (N1L, N2L, N3L). This is the default value. If SPIN = 2, PWmat will write out an extra file “OUT.RHO_2”. If SPIN=222, PWmat will also output OUT.RHO_SOM, a 2x2 complex spin matrix density.

Use utility program “convert_rho.x” to plot the charge density (unit in e/Bohr^3. More details about “convert_rho.x”, please refer to PWmat website http://www.pwmat.com/utility-download.

If OUT.RHO = F, not output the charge file.

2.1.7.22 OUT.VR

OUT.VR = T / F

Default:

OUT.VR = T

If OUT.VR = T, PWmat will output the total potential in file “OUT.VR”, stored in real space grid (N1L, N2L, N3L). This is the default value. When SPIN=2, an extra file “OUT.VR_2” will be output. When SPIN=222, OUT.VR, OUT.VR_SOM (a 2x2 complex spin matrix potential), and OUT.VR_DELTA (a real up-down potential) will be output. Use utility program “convert_rho.x” to plot the potential (unit in Hartree).

if OUT.VR = F, not output the potential file.

2.1.7.23 OUT.HSEWR

OUT.HSEWR = T / F Default: **OUT.HSEWR = T** If OUT.HSEWR = T and XCFUNCTIONAL = HSE / B3LYP, PWmat will output the real space wave functions for the Fock exchange kernel for all the extended k-points on every GPU in 'OUT.HSEWR(i)' files, i is the index of GPUs.

2.1.7.24 OUT.ELF

OUT.ELF = T / F ELF_RHO_TOL

Default:

OUT.ELF = F 1.D-4

This parameter is used to control whether to output OUT.ELF file. If charge density rho(r)<ELF_RHO_TOL, set rho(r)=0 for ELF calculation. The file OUT.ELF is the electron localization function, you can convert it to xsf format by using convert_rho.x utility. Usually you should set Ecut2=4*Ecut in etot.input.

2.1.7.25 OUT.REAL.RHOWF_SP

OUT.REAL.RHOWF_SP = IFLAG, KPT1, KPT2, ISPIN1, ISPIN2, IW1, IW2, E1, E2

IFLAG = 0 / 1 / 11 / 12 / 2 / 21 / 22

Default:

OUT.REAL.RHOWF_SP = 0

Output the charge density (or wave function) in real space. This is a special option allow user to selectively output some charge density and wave functions.

Controls the output of partial charge density (or wave function without square) in real space grid (N1, N2, N3) from selected eigen orbitals within the intervals: k-points: [KPT1, KPT2], spins: [ISPIN1, ISPIN2], bands: [IW1, IW2] in the file: "OUT.REAL.RHOWF_SP". For partial charge density, one can also set [E1,E2] the eigen energy range. If [E1,E2] exists, the bands range will not be used. The unit of E1

and E2 is eV. This is different from OUT.RHO, since it can select which wave function to be included in the charge density. For:

IFLAG=0, not output the density or wavefunctions, default setting.

IFLAG = 1/11/12, output the density or wavefunctions **at the end of other calculations**.

1. **IFLAG=1**, output charge density;
2. **IFLAG=11**, output the wavefunctions, one after the other without the e^{-ikr} phase;
3. **IFLAG=12**, output the wavefunctions, one after the other with the e^{-ikr} phase.

IFLAG = 2/21/22, output the density or wavefunctions **before doing any other calculations, then stop PWmat**.

1. **IFLAG=2**, output charge density;
2. **IFLAG=21**, output the wavefunctions, one after the other without the e^{-ikr} phase;
3. **IFLAG=22**, output the wavefunctions, one after the other with the e^{-ikr} phase.

```
DO IK = KPT1, KPT2
   DO IS = ISPIN1, ISPIN2
      DO IW = IW1, IW2
         REAL PART:
         DO INODE = 1, NNODE
            WRITE (11) (REAL(PSI(IR+(INODE-1)*NR_N)), IR = 1, NR_N)
         END DO
         IMAG PART:
         DO INODE = 1, NNODE
            WRITE (11) (IMAG(PSI(IR+(INODE-1)*NR_N)), IR = 1, NR_N)
         END DO
      END DO
   END DO
END DO
```

In above PSI(IR) is the wave function in the real space grid (N1,N2,N3), it first runs through N3, then N2, then N1. In another word, for a given point (i,j,k), for i within [1,N1], j within [1,N2], k within [1,N3] then: IR=(i-1)*N2*N3+(j-1)*N3+k.

Note, the wave function can also be viewed (perhaps more conveniently) one by one using the utility function plot_wg.x, using OUT.WG.

2.1.7.26 OUT.SOLVENT_CHARGE

OUT.SOLVENT_CHARGE = T / F

Default:

OUT.SOLVENT_CHARGE = F

Output several files for viewing when IN.SOLVENT=T.

If OUT.SOLVENT_CHARGE=T, several files (in the same format as OUT.RHO) will be ouput: OUT.RHO_4DIELECTRIC (the rho_e to be used to generate the dielectric function. One can plot the isosurface plots using RHOMAX_DIELECTRIC and RHOMIN_DIELECTRIC values described in IN.SOLVENT to view where are the turn-in/off surfaces of the dielectric constant); OUT.RHO_POLARIZE (the solvent induced polarization charge); OUT.V_POLARIZE (the polarized potential generated by the polarization charge OUT.RHO_POLARIZE); OUT.RHOP_VHION (the polarizaion charge multiplied by the electric static potential of the solute molecule. The integrate of this density is the polarization energy. One can use this to see where the polazation energy comes from). One usually use VESTA to view these files.

2.1.7.27 OUT.FORCE

OUT.FORCE = T / F

Default:

OUT.FORCE = T (JOB=RELAX or JOB=MD)

OUT.FORCE = F (everything else)

If OUT.FORCE = T, the PWmat will calculate the atomic force, and output the force in file "OUT.FORCE". This is for one shot (one atomic position snap shot) JOB=SCF

calculation only. For JOB=RELAX, or JOB=MD, PWmat will always calculate the force and output them. Also note, this will not work for JOB=NONSCF, since there the total energy and SCF charge density will not be calculated. The force unit is eV/Amstrong. if OUT.FORCE = F, the default, not calculate the force and output the file.

2.1.7.28 OUT.STRESS

OUT.STRESS = T / F

Default:

OUT.STRESS = T (JOB = RELAX)

OUT.STRESS = F (everything else)

Calculate and output the stress tensor.

Definition of stress tensor(eV), ϵ_{ij} is stain:

$\sigma_{ij} = \frac{\partial E_{tot}}{\partial \epsilon_{ij}}$

Definition of pressure(GPascal):

$P = -\frac{1}{3\Omega}(\sigma_{11} + \sigma_{22} + \sigma_{33})$

If OUT.STRESS=T, the stress tensor will be calculated and written in file OUT.STRESS.

If do cell relaxation, the stress is automatically calculated. When JOB = SCF, if OUT.STRESS = T, the stress will be calculated; if OUT.STRESS = F, the stress will not be calculated.

2.1.7.29 OUT.VATOM

OUT.VATOM = T / F

Default:

OUT.VATOM = F

If setting OUT.VATOM = T, it will output the atom center potential for SCF or MD simulation. The default is F. This can be used for energy level alignment etc. The output unit is eV. This does an average using atomic charge density dot-product with

the total potential. We do not have core level potential. So, for band alignment, we usually use this atomic potential.

2.1.7.30 OUT.HSEWR

OUT.HSEWR = T / F

Default:

OUT.HSEWR = T

This parameter is used to control whether to output OUT.HSEWR* file while XCFUNCTIONAL=HSE. The files OUT.HSEWR* are the real space wavefunctions of all unreduced kpoints, usually they are in big size and take long time to be written on disk. If you're sure that you do not need these files, you can set OUT.HSEWR=F. (For example, files OUT.HSEWR* will be used to run NONSCF with XCFUNCTIONAL=HSE.)

2.1.7.31 OUT.TDDFT

OUT.TDDFT = T_1, T_2, n_1, T_3, n_2

Default:

OUT.TDDFT = F F 1.0 F 1.0

The output files can be used to restart TDDFT and show the process of TDDFT.

$T1, T2, n1$	$T1 = T/F$	eigen energy, $occ(i)$ per $n1$ fs. The output will be in file OUT.TDDFT1. One can use plot_TDDFT.f90(ref. util) to read and output OUT.TDDFT1.
	$T2 = T/F$	C_{ij} per n1 fs
$T3, n2$	$T3 = T/F$	output all the wavefunctions and charge densities per $n2$ fs for restart. The output will be in file OUT.TDDFT and directory TDDOS/. This can be very expensive, so use large $n2$.

2.1.7.32 OUT.MLMD

OUT.MLMD = T / F

Default:

OUT.MLMD = F

This parameter is used to control whether to output OUT.MLMD file during JOB = SCF. The format of OUT.MLMD is the same as MOVEMENT, including atomic postion and atomic force setctions.

2.1.7.33 OUT.GAUSSIAN

OUT.GAUSSIAN = T/F

Default: **OUT.GAUSSIAN = F**

If USE_GAUSSIAN = T and OUT.GAUSSIAN = T, the program will output file OUT.GAUSSIAN_H, OUT.GAUSSIAN_S, OUT.GAUSSIAN_H_T, OUT.GAUSSIAN_S_T, OUT.GAUSSIAN_BASIS_INDEX.

2.1.7.34 OUT.GTH2UPF

OUT.GTH2UPF = T/F

Default: **OUT.GTH2UPF = F**

If USE_GAUSSIAN = T and OUT.GTH2UPF = T, the program will convert the GTH pseudopoentials to UPF format. The output files are with prefix "GTH".

2.1.7.35 PWSCF_OUTPUT

PWSCF_OUTPUT = T/F

Default:

PWSCF_OUTPUT = F

This parameter controls whether to output pwscf compatible output files (wave function, charge density, and potential), so the result can be run subsequently on pwscf. For T, it will output those files into the directory "prefix.save". For F, it will not output.

Some recommendations: If PWSCF_OUTPUT=T, please use the setting: **ECUT2L = ECUT2, N123L = N123, ECUT2 = 4*ECUT**. Because PWmat implement a different FFT from PWSCF.

2.1.7.36 PULAY_IN_OUT

PULAY_IN_OUT= 0/1 0/1

Default:

PULAY_IN_OUT= 0 0

This parameter controls whether to read-in or write-out the pulay mixing matrix in the file "dwdR_out.*". The first number controls whether to read-in the pulay mixing matrix from the file "dwdR_out.*", 0 - not read , 1 - read; The second number controls whether to write-out the pulay mixing matrix to the file "dwdR_out.*", 0 - not write , 1 - write.

2.2 Structure file (atom.config)

The structure file can be named arbitrarily, but its name must be specified by the tag 'IN.ATOM' in 'etot.input' file. In most of our examples and tutorials, the name of structure file is 'atom.config'. The contents in this file describe the lattice vector of the supercell, and how many atoms are in the supercell, their atomic number, fractional coordinates and whether they can move. Some optional data blocks can also be written in the structure file: such as the force of the atom, the velocity of the atom, the initial magnetic moment of the atom (including collinear or non-collinear magnetic moments), etc. It has the following format:

```
64
LATTICE
0.1084993850E+02 0.0000000000E+00 0.0000000000E+00
0.0000000000E+00 0.1084993850E+02 0.0000000000E+00
0.0000000000E+00 0.0000000000E+00 0.1084993850E+02
POSITION
```

```
30  0.952534560  0.363594470  0.382027650 1 1 1
30  0.540553000  0.850230410  0.966359450 1 1 1
...
16  0.242857140  0.140553000  0.684331800 1 1 1
FORCE # optional
30 -0.060040948  0.097096690  0.063013193
30  0.001068674 -0.002521614  0.000147553
...
16 -0.007955164 -0.008758074  0.029047748
VELOCITY # optional
30  0.02339881  -0.287387433 -0.109339839
30 -0.23878474  -0.210836551  0.049311111
...
16  0.53761771  -0.023987172  0.288399911
MAGNETIC # optional, initial collinear magnetic moment
30 2
30 2
...
16 0
CONSTRAINT_MAG # optional, to constraint magnetic moment
30 2 0.01      # mag,alpha (mag: desired magnetic moment)
30 2 0.01      # mag,alpha (alpha: penalty coeff, eV)
...
16 0 0.00
MAGNETIC_XYZ # optional, initial non-collinear magnetic moment
30 2 0 0
30 2 0 0
...
16 0 0 0
LANGEVIN_ATOMFACT_TG # optional, special Lagenvin MD, atomic temperature and gamma
30  1.0   1.0
30  0.5   1.0
...
16  0.5   0.5
STRESS_MASK # optional
1 0 0
0 1 0
0 0 1
STRESS_EXTERNAL # optional
```

```
0.1 0.0 0.0
0.0 0.1 0.0
0.0 0.0 0.1
PTENSOR_EXTERNAL # optional
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
DIMER_DIR_N # optional
30   0.000001    0.522103   -0.000009
30  -0.000006    0.530068    0.000000
...
16   0.000001   -0.111442    0.000001
```

They have the following meanings:

Natom: the number of atoms in the system, as a result, Position, Force, Velocity, Magnetic sections will all have Natom lines, each atom per line, and the sequence must be consistent.

LATTICE: The header of the lattice vector AL(3,3) section. There will be three lines following Lattice vector:

AL(1,1), AL(2,1), AL(3,1) (the 1st vector of the supercell axis in Å)

AL(1,2), AL(2,2), AL(3,2) (the 2nd vector of the supercell axis in Å)

AL(1,3), AL(2,3), AL(3,3) (the 3rd vector of the supercell axis in Å)

POSITION: the header of the atomic positions of the system. There will be Natom lines, each line describes one atom, including its atomic number, fractional coordinates and degree of freedom, in the following form:

```
Zatom  x1      x2       x3       imv1  imv2  imv3
30     0.2293  0.59822  0.44444  1     1     1
```

ZATOM is the atomic number of this atom, **x1, x2, x3** are the fractional coordinates of this atom in the supercell. **imv1, imv2, imv3** indicates whether the cartesian coordinates of this atom can be changed when JOB = RELAX / MD / TDDFT / NAMD / NEB / DIMER. If Imv1 (2, 3) = 1, the coordinate of direction x (y, z) can be changed; If Imv1 (2, 3) = 0, the coordinate of direction x (y, z) can not be changed.

Note the x, y, z coordinates of this atom can be calculated as:

$$X = AL(1,1) * x_1 + AL(1,2) * x_2 + AL(1,3) * x_3 \tag{2.14}$$

$$Y = AL(2,1) * x_1 + AL(2,2) * x_2 + AL(2,3) * x_3 \tag{2.15}$$

$$Z = AL(3,1) * x_1 + AL(3,2) * x_2 + AL(3,3) * x_3 \tag{2.16}$$

FORCE: The header of the force section. This section is optional. It will be followed by natom lines in the following form:

```
zatom, f_x,   f_y,     f_z
30     0.0372 0.01112 -0.1021
```

They are the x, y, z direction atomic forces in $eV/\text{Å}$.

VELOCITY: The header of the velocity section. This section is optional. If will be followed by natom lines in the following form:

```
zatom, v_x,    v_y,       v_z
30     0.39292 -0.222933 0.28211
```

They are the x, y, z direction atomic velocity in $Bohr/fs$.

MAGNETIC: The header of the magnetic section. This tag specify the initial collinear magnetic moment for each atom when SPIN = 2. It will be followed by natom lines in the following format:

```
zatom spin
30    2
```

2 is the magnetic moment: 0 is no spin, the negative is spin down, the positive is spin up.

CONSTRAINT_MAG: The header of the constraint magnetic moment section. This tag specifies the desired magnetic moment, and a penalty coefficient to enforce the system to have such magnetic moment. This only works for spin=2. It will be followed by natom lines in the following format:

```
zatom spin,alpha
30    2    0.01
```

2 is the desired magnetic moment to force the atom to have: 0.01 is the alpha coefficient (in the unit of eV) for the penalty term. Smaller this alpha, less the enforcement. Note, if alpha is too large, the SCF iteration might not converge.

MAGNETIC_XYZ: The header of the non-collinear magnetic systems section when spin = 222. It specifify the initial magnetic moment for each atom. If will be followed by natom lines in the following format:

```
zatom spin_x spin_y spin_z
30    2      0      0
```

LANGEVIN_ATOMFACT_TG: The multiplication scaling factors for the atomic temperatute and the gamma parameter in the Langevin molecular dynamics. Here, in Langevin MD, we provide an option which one can tune the temperature of each atom by providing these atomic scaling factor. The temperature for each atom is determined by the overall desired temperature defined by the MD_DETAIL line, multiplied by the atomic scaling factor provided here. Similarly, the gamma parameter in the Langevin dynamics is also determined by the global gamma factor defined in IN.MDOPT (or its default values) multiplied by the Gamma scaling factor provided here. Note, if this section is not provided, the default scaling factors are 1.d0 The local temperature is maintained by providing atomic temperature proportional atomic random force. The rate of the local temperature dissipation is controlled by the gamma(i) term defined in dV/dt= F(i)-gamma(i)*V(i)+F_random(i). Larger the gamma(i), faster it reaches its desired temperature. However, note, the heat injection from the random force will be diffused into nearby atoms. So, one has to adjust fact_temp(i), and fact_gamma(i) to get the desired results.

This section title will be followed by natom lines in the following format:

```
zatom fact_temp, fact_gamma
30    0.8         0.9
```

STRESS_MASK: used to multiply to the stress tenser for cell relaxation(and constant pressure molecular dynamics), so some directions of the cell can be fixed. For example, if you want to optimize the lattice of two-dimensional materials, and vacuum direction is along z, STRESS_MASK can be set as follows:

```
STRESS_MASK
1 1 0
1 1 0
0 0 0
```

STRESS_EXTERNAL: used to add an external stress tensor for cell relaxation, its unit is eV. Definition of stress tensor is $\sigma_{ij} = \frac{\partial E_{tot}}{\partial \epsilon_{ij}}$. After cell relaxation: $STRESS + STRESS\_EXTERNAL \rightarrow 0$

PTENSOR_EXTERNAL: used to add an external stress tensor for cell relaxation, its unit is Gpa. Definition is $\sigma_{ij} = \frac{1}{VolumeofCell}\frac{\partial E_{tot}}{\partial \epsilon_{ij}}$.After cell relaxation: $STRESS + PTENSOR\_EXTERNAL * (VolumeofCell) \rightarrow 0$

DIMER_DIR_N:Used to specify the initial direction along the dimer when JOB =DIMER. First column is the atomic number of each atom, other columns are fractional coordinates of the initial search direction.

How to convert the format of the structure file: By using some utilities, we can convert other crystal formats to PWmat format or convert it back like:

VASP format to PWmat format: 'poscar2config.x POSCAR';

CIF format to PWmat format: 'cif2cell XXX.cif –p pwmat', please refer to CIF2CELL for more details;

XSF format to PWmat format: 'xsf2config.x POSCAR'.

PWmat format to VASP format: 'config2poscar.x atom.config';

PWmat format to CIF format: 'atomconfig2cif atom.config', please refer to CIF2CELL for more details;

PWmat format to XSF format: 'convert_from_config.x atom.config'.

Check structural information of PWmat structure file: ‘atominfo.x atom.config’.

2.3 Pseudopotential files (*.UPF)

Pseudopotential files can be named arbitrarily, but must be specified in ‘etot.input’ by the tag ‘IN.PSP’. The pseudopotential supported by PWmat is in unified pseudopotential format (UPF). At present, PWmat only supports the use of norm-conserving pseudopotential (NCPP) or ultrasoft pseudopotential (USPP), PWmat does not support the "projector augmented wave" (PAW) pseudopotential. We strongly recommend you to use NCPP because most of the USPP functions in PWmat are no longer maintained. Besides, the recent "norm conserving vanderbilt pseudopotential" released by D.R. Hamann made the norm conserving pseudopotential very reliable.

Some online libraries provide available pseudopotential files. Now you can also download it from PWmat website. The current release include the following pseudopotential sets: ONCV-PWM, NCPP-SG15, NCPP-PD03,NCPP-PD04, NCPP-FHI. They are all NCPP sets. PWmat allows to use pseudopotentials from different sets in one calculation but it is recommended to use pseudopotentials from same set.

NCPP-SG15, NCPP-PD03 and NCPP-PD04 are very accurate. However, due to the consideration of semi-core valence electrons for most elements, the calculation speed is relatively slow and requires a large amount of memory. Comparing to SG15, ONCV-PWM has less valence electrons, and the plane wave cutoff energy for wavefunction can be relatively greatly reduced (our recommended cutoff is 45Ry). The error of ONCV-PWM and NCPP-FHI may be relatively large. If you want to calculate quickly, you can use ONCV-PWM or NCPP-FHI. If you want to calculate accurately, you can use NCPP-SG15, NCPP-PD03, or NCPP-PD04.

The calculation of spin-orbit coupling (SPIN=22/222) requires SOC pseudopotential sets, such as NCPP-SG15-PBE-SOC. However, if one has a NCPP UPF with SOC, one can use our utility ‘upf2upfSO.x’ to convert it into our special UPF format for our SOC

calculation.

For NCPP-PD03, NCPP-PD04, NCPP-SG15 (these two are rather similar), we recommend the user to use Ecut=50 Ryd (if it is not converged, one can even use 60, or 80 Ryd). The most challenging calculation is for the atomic relaxation where smooth energy surface is required. For that purpose, for these two sets of pseudopotentials, we recommend: Ecut2=4Ecut, and Ecut2L=4Ecut2 (e.g., N123L=2N123). In other words, choose Accuracy=high. On the other hand, for NCPP-FHI, one can use Ecut=40,50, while Ecut2=2Ecut, Ecut2L=Ecut2 (e.g., choose Accuracy=norm). For USPP-SOFT, in most cases, one can choose: Ecut=30, Ecut2=2Ecut, Ecut2L=4Ecut2. FOr USPP-GBRV, Ecut=40 Ryd is recommended, along with, perhaps, Ecut2=2Ecut, Ecut2L=4Ecut2.

Note, the above requirement is only true for JOB=RELAX. For other jobs, including JOB=MD, or bandstructure, one can relax the requirement, perhaps using Ecut2=2Ecut, and Ecut2L=Ecut2 (Accuracy=norm) even for NCPP-PD03 and NCPP-SG15.

Note, the pseudopotential file also contains information for Ecut, and rcut. That will be the default Ecut, and rcut when they are used.

2.4 Optional input files

2.4.1 IN.KPT file

If one set IN.KPT = T in 'etot.input', it will read the file "IN.KPT", which contains the k-point vectors and their weights.

The format of 'IN.KPT' is same as 'OUT.KPT'.

2.4.2 IN.SYMM file

If one set IN.SYMM = T in 'etot.input', it will read the file "IN.SYMM", which contains the symmetry operations.

The format of 'IN.SYMM' is same as 'OUT.SYMM'.

2.4.3 IN.OCC file

Check 2.1.7.5 for more details.

2.4.4 IN.OCC_T file

If we have IN.OCC_T = T. Then, it will read in file 'IN.OCC_T' (and 'IN.OCC_T_2' if spin=2), and it will ignore the imth_Fermi flag In the SCF_ITERx_x line.

Inside 'IN.OCC_T', we currently have the following format:

```
nline, nkpt
**** kpt=1 *****
Istate, iformula, a1,a2,a3,a4,a5
Xxxxx
nlines
**** kpt= 2 ******
Istate,iformula,a1,a2,a3,a4
Xxxxx
```

One example is:

```
2,  2
*** kpt=1 ****
1,  1,  0.1, -1., 0.,0,0
3,  1,  0.1, -1., 0., 0., 0.
****kpt=2 ****
1,  1,  0.1, -1., 0.,0,0
3,  1,  0.1, -1., 0., 0., 0.
```

The nline indicate how many states (bands) one need the time-dependent occ(t) correction. Nkpt is the number of k-points.

The istate is the index for one band (psi_istate) which needs time dependent modification for occ(t).

Iformula is the flag for formula. A1,a2,a3,a4,a5 are five parameter.

We have the following two formula:

Iformula=1

For time < a1: Occ(t,istate)=occ(0,istate)+(time/a1)*a2

For time >= a1: Occ(t,istate)=occ(0,istate)+a2

Iformula=2

For time <a1, Occ(t,istate)=occ(0,istate)+(1-cos(pi*time/a1/2)**2)*a2

For time >=a1: Occ(t,istate)=occ(0,istate)+a2

So, at this moment, a3,a4,a5 are never used.

This change of occ(i) is used to describe some electron state is suddenly removed (or added) to the system from some specific states. It is different from the use of IN.OCC. In IN.OCC, a selfconsistent solution is achieved making some state empty or occupied not according to Fermi-Dirac distribution. But the solution of IN.OCC is the occupation of pure eigen states in the H(N-1) (remove one electron) or H(N+1) (add one electron) Hamiltonian. For the use of IN.OCC_T, it could be that the original H(N) eigen state psi_istate is removed, so at H(N-1), the states which is removed (or still occupied) might not be the eigen states of H(N-1) (but they might be close to the eigen states of H(N), if parameter a1 is short enough)

2.4.5 IN.NONSCF file

When JOB = NONSCF, some specific parameters can be set in the file 'IN.NONSCF'. In this way, you also have to set 'IN.NONSCF = T' in etot.input.

The settings available in the file 'IN.NONSCF' are as follows:

```
NONSCF_METH = 0
    ! 0 – the conventional NONSCF calculation
    ! 1 – calculate minimal eigen energy
    !     must set PRECISION=DOUBLE in etot.input
```

! -1 – calculate maximal eigen energy
! must set PRECISION=DOUBLE in etot.input
! 2 – the escan calculation based on $(H - FSM\_EREF)^2$, using folded spectrum method
! must set PRECISION=DOUBLE in etot.input
! must set NUM_BAND explicitly in etot.input
! 3 – the DOS calculation using generalized moments method
! must set PRECISION=DOUBLE in etot.input
! 4 – the OAS(optical absorption spectrum) calculation using generalized moments method
! must set PRECISION=DOUBLE in etot.input
! 5 – the chebyshev filter method to calculate eigen energies within specified range
! must set PRECISION=DOUBLE in etot.input
FSM_EREF = 0.0
! the reference energy used in $(H - FSM\_EREF)^2$ for NONSCF_METH=2
! unit eV.
GMM_DOS_EMIN = 0.0
! the minimal energy of DOS range, must be smaller than the mininal eigen energy calculated by NONSCF_METH=1
! only used by NONSCF_METH=3
! unit eV.
GMM_DOS_EMAX = 0.0
! the maximal energy of DOS range, must be larger than the maxinal eigen energy calculated by NONSCF_METH=-1
! only used by NONSCF_METH=3
! unit eV.
GMM_DOS_IN_PSI0 = T/F FILE_NAME_PSI0
! whether to use an initial wavefunction file instead of random initialization
! only used by NONSCF_METH=3
GMM_DOS_MMAX = 3000
! the total number of moments for the moments method. The energy resolution is roughly $(EMAX - EMIN)/MMAX$
GMM_DOS_IRANDOM = 2019
! random number seed
GMM_DOS_mx_ab = 20

```
! used by nonscf_meth=5, the number of wave functions to be stored in the memory.
! larger mx_ab will require more memory, but it will reduce the number of I/O
GMM_DOS_ipxyz = 0
! used by nonscf_meth=5, the absorption polarization in x, y, z
! ipxyz=1, x polarization only, output Tm.store.1
! ipxyz=2, y polarization only, output Tm.store.2
! ipxyz=3, z polarization only, output Tm.store.3
! ipxyz=0, x, y, z polarizations, output Tm.store.1, Tm.store.2, Tm.store.3
ESCAN_DETAIL = E_window_start, E_window_end, degree_cheb, niter_lanczos
! used by nonscf_meth=5, unit of E_window_start and E_window_end is eV.
! PWmat well calculate eigen states with energy in range (E_window_start, E_window_end)
! default is 0,0,1000,20; Please check ref paper for last two parameters. [32]
! E_window_start should not be equal to E_window_end for actrual use.
```

For NONSCF_METH=5, the chebyshev filter method[32], one should use just one nonscf with larger nline by setting SCF_ITER0 as following (NITER0=1,NLINE0=10), you can always increase NLINE0 to get more converged results:

```
SCF_ITER0_1 = 1 10 3 0.0 0.025 1
```

For NONSCF_METH=5, you also need to estimate the NUM_BAND, especially for large systems.

2.4.6 IN.RELAXOPT file

When JOB = RELAX / NEB / DIMER, some specific parameters can be set in the file 'IN.RELAXOPT'. In this way, you also have to set 'IN.RELAXOPT = T' in etot.input.

The IN.RELAXOPT is as follows:

```
PSTRESS_EXTERNAL= 0.0
    ! external hydrostatic pressure. unit GPascal – for cell relaxation
    ! the energy P*V (i.e. pressure*volume) will be written in file REPORT with flag "Energy PV".
RELAX_MAXMOVE = 1.0
    ! max move distance. unit bohr – for JOB=RELAX/NEB and method=1,5,6.
    ! This can be used to enforce small steps, for convergence.
LBFGS_MEMORY = 30
    ! LBFGS storage size – for JOB=RELAX/NEB and method=5.
FIRE_DT = 1.0
    ! initial time step. unit fs – for JOB=RELAX/NEB and method=6.
    ! the max time step for FIRE method is 10*FIRE_DT.
RHOWG_INTER_TYPE = 1
    ! interpolation type for JOB=NEB: 0–both rho and wave function; 1–rho only.
    ! default = 1, save time by not writing wavefunction to disk
NSTEP_OUTPUT_RHO=100
    ! step interval to output the charge density; for JOB=RELAX
    ! It is used to output more charge density for later use and analysis.
CELL_FIX = 0 1 1 1
    ! the first number is the flag for cell optimization(JOB=RELAX), 0 - not fix shape and angle, 1 - fix angle, 2 - fix shape.
    ! the following 3 numbers are the flags for each lattice vector (i.e. a,b,c) just used when the first number is 1,
    ! 0 means that lattice vector is fixed, 1 means that lattice vector can be optimized.
    ! when the first number is 2, the following 3 numbers are not used, and only the volume of the cell is optimized.
UNIT_TOL_STRESS = eV
    ! unit of stress tolerance, can be eV, Gpa, kbar.
```

Following parameters can be set for JOB=DIMER in file IN.RELAXOPT, one also should set IN.RELAXOPT=T in etot.input if needed.

DIMER_DR
! the dimer seperation, unit Bohr; for JOB=DIMER
DIMER_NMAX_ROTS
! the maximum number of rotation for each translation step; for JOB=DIMER
DIMER_NMAX_STEPS
! the maximum number of translation steps; for JOB=DIMER
DIMER_TOL_FORCE
! force tolerance to judge the convergency, unit eV/Angstrom; for JOB=DIMER
DIMER_MAX_STEPSIZE
! the maximum step size of each translation step, unit Bohr; for JOB=DIMER

2.4.7 IN.MDOPT file

When JOB = MD, some specific parameters related to MD methods can be set in the file 'IN.MDOPT'. In this way, you also have to set 'IN.MDOPT = T' in etot.input.

The file and parameters(and the parameters' default values) are as follows,

IN.MDOPT:

MD_CELL_TAU = 400*DT !(for 4,5,8, LV,NH-cell)
! DT is the MD time step (fs).
! characteristic time for cell oscillations (fs).
! Larger MD_CELL_TAU results in a longer time to reach equilibrium (both temperature and pressure) for the cell

MD_ION_TAU = 40*DT !(for 2,5, NH-ion)
! characteristic time for particles oscillations (fs).
! This is for the NH algorithm, the time scale for the ion movement to reach equilibrium for a given temperature. Longer MD_ION_TAU, slower the particles to reach equilibrium.
! For more accurate simulation, one should use larger MD_ION_TAU, but the fluctuation will also be larger.

MD_LV_GAMMA = 0.01 !(for 3,4, LV-ion)

! friction coefficient for particle movement in LV (fs^{-1}).

! Larger MD_LV_GAMMA, faster it reaches equilibrium, but then larger is the random noise in LV algorithm

! For more accurate, more realistic calculation, one needs smaller MD_LV_GAMMA, but then the fluctuation will also be bigger.

MD_NPT_GAMMA= 0.01 !(for 4, LV-NPT)

! friction coefficient for cell (fs^{-1}).

! Larger MD_NPT_GAMMA, faster it reaches cell equilibrium.

! For accurate calculation, one should use smaller MD_NPT_GAMMA, but the fluctuation will also be larger.

MD_NPT_PEXT= 0.0 !(for 4,5,7, NPT)

! This is the applied external hydrastatic pressure (GPa).

MD_NPT_PEXT_XYZ= 0.0 0.0 0.0 !(for 4,5,7, NPT)

! external x,y,z pressure (GPa), over-write(higher priority than) MD_NPT_PEXT.

MD_BERENDSEN_TAU= 500*DT !(for 6, 7, BR-ion)

! the atom velocity rescaling time in Berendsen method (fs).

! Larger MD_BERENDSEN_TAU will be more accurate, but the fluctuation will also be larger.

MD_BERENDSEN_TAUP= 500*DT !(for 7, BR-cell)

! the cell rescaling time in Berendsen method for MD=7 (fs).

! Larger MD_BERENDSEN_TAUP will be more accurate, but the fluctuation will also be larger.

MD_BERENDSEN_CEL_STEPS= nstep !(for 7, BR-cell)

! the number of MD steps for one stress calculation.

! The default value is one. One can increase nstep, so there are less stress calculation.

! The stress calculation is a bit expensive.

MD_SEED= 12345 !(for all)

! random seed for initializing the velocities

! when no velocities are specified in IN.ATOM file.

! if MD_SEED=-1, the random seed will be set using the system_clock().

! if this is not set, a default 12345 will be used.

MD_AVET_TIMEINTERVAL= 100*DT !(for all)

! time interval to calculate average temperature and pressure (fs).
! This is not for the instantaneous temperatur and pressure, but the average values within the MD_AVET_TIMEINTERVAL time.
MD_NPT_ISOSCALEV= 0 !(for all)
! 1–overall scaling of the box; default=0
NSTEP_OUTPUT_RHO= 100
! step interval to output the charge density
MD_MSST_VS = 0.0
! velocity of shock wave (bohr/fs)
MD_MSST_DIR = 0
! direction of shock wave (0–x, 1–y, 2–z)
MD_ZERO_TOTMOMENT = F
! if MD_ZERO_TOTMOMENT=T, make system's total momentum to zero; default = F

2.4.8 IN.EXT_FORCE file

If IN.EXT_FORCE=T, a file IN.EXT_FORCE will be provided, it has the following format:

```
natom
iatom, fx, fy, fz    ! unit eV/Amstrong
.................
iatom, fx, fy, fz    ! There will be natom lines
```

2.4.9 IN.SOLVENT file

IN.SOLVENT = T, will use solvent model. Note, all the other input in etot.input will be the same. In other words, solvent model can be used to do single SCF, RELAX, MD and TDDFT calculations. However, one has to prepare a "IN.SOLVENT" file

in the running directory, which control the parameters for the solvent model. When IN.SOLVENT=T, after the PWmat run, a OUT.SOLVENT file will be generated which lists all the options used for the solvent model. The "IN.SOLVENT" has the following contents:

SOLVENT_TYPE = predefined_solvent_type

predefined_solvent_type can be: WATER, NONE.

If SOLVENT_TYPE=WATER, the program will use the default parameters for water, but the other explicit input parameters can overwrite the default parameters.

DIELECTRIC_CONST = epsilon0

epsilon0 is a real number: the dielectric constant of the solvent.

SURFACE_TENSION = E

E is the surface tension (dyn/cm) of the cavity,

used to calculate E*S term, S is the surface area of the cavity.

positive/negative E (default is 50 for water) means it will have positive/negative surface energy.

Can be used to represent the solute/solvent van der Walls interaction.

Can be used as a fitting parameter.

PRESSURE = P

P is the pressure (GPa) of the cavity,

used to calculate the P*V term, V is the volume of the cavity.

Default is -0.35 for water.

Can be used as a fitting parameter.

RHOMAX_DIELECTRIC = cut1

Default, 0.005 ($electron/Bohr^3$),

used to control epsilon(rho_e).

When rho_e > RHOMAX_DIELECTRIC, epsilon=1

RHOMIN_DIELECTRIC = cut2

Default, 0.0001, used to control epsilon(rho_e).

When rho_e < RHOMIN_DIELECTRIC, epsilon=epsilon0 input above.

Basically, the dielectric constant changes from 1 to epsilon0 when rho_e changes from cut1 to cut2.

The choice of rho_e will be controlled by DIELECTRIC_MODEL below.

DIELECTRIC_MODEL = dielectric_model_type

The parameter controls what to be used as the rho_e (together with cut1 and cut2) to control the dielectric function profile.

The dielectric_model_type can be SCF_CHARGE, ATOM_CHARGE, EXP_CHARGE, AEXP_CHARGE.

SCF_CHARGE: use the SCF calculated electron charge density (default)

ATOM_CHARGE: use the sum of neutral atomic charge density from upf file (each atom multiplied by a prefactor param1). We recommend this option for most calculations due to stability.

EXP_CHARGE: for each atom, use an exponential param2*exp(-r/param3) form.

AEXP_CHARGE: use the sum of neutral atomic charge and the exponential charge, equals: param1*rho_atom(r)+param2*exp(-r/param3)

Note, for stability, especially for JOB=RELAX, we suggest to use ATOM_CHARGE, EXP_CHARGE or AEXP_CHARGE. There is not much benefit to use SCF charge density.

AEXP_CHARGE is used to fill the possible hole at r=0 for ATOM_CHARGE. If ATOM_CHARGE is used, the program will output an atom.UPF.rhoatom file for each atom, which has: r, rho(r) (with unit Bohr, electron/$Bohr^3$). One can use these files to fit the param2, param3, then to use EXP_CHARGE to replace ATOM_CHARGE (if one wants to do that). Or to find the proper param2,param3 for AEXP_CHARGE. One can also use that to find out whether the RHOMAX_DIELECTRIC value will put some dielectric constant at r=0, i.e. whether RHOMAX_DIELECTRIC is larger than rho_atom(r) at r=0 (which is bad, then AEXP_CHARGE or EXP_CHARGE are better to be used).

PARAM_CHARGE.1 = param1,param2,param3

PARAM_CHARGE.2 = param1,param2,param3

The number, 1,2,.. must be the same as the atom types the pseudopotential types. There must always be three numbers for each line, even if some are not used. These parameters are used to control the charge expression for ATOM_CHARGE, EXP_CHARGE, and AEXP_CHARGE. param1 is a prefactor for ATOM_CHARGE. For the EXP_CHARGE, it is: $param2 * exp(-r/param3)$. Here param2 unit is electron/$Bohr^3$, and param3 unit is Bohr.

RHOMAX_CAVITY = cut11

Default, 0.005 ($electron/Bohr^3$), used to control the cavity of the solute molecule.

RHOMIN_CAVITY = cut22

Default, 0.0001, used to control the cavity of the solute molecule. The cavity is created inside cut11 (when rho > cut11). The thickness of the surface is controlled by cut11 to cut22 (the cavity disappears for rho < cut22. Note, the cavity is always judged by the SCF calculated valence electron charge density. Also, the cavity for the cavity and pressure term can be different from the effective cavity of the dielectric function.

POISSON_BOLTZMANN = T/F

Used to control whether to do linearized Poisson-Boltzmann equation. If true, we need the following parameter: RHOMAX_DEBY, RHOMIN_DEBY, AKK0_DEBY.

DEBY_AKK0 = akk_0

The inverse deby length square in the linearized Poisson-Boltmann equation. Unit: $1/Bhor^2$, default 0.

Note, $akk_0 = akk_b^2 = e^2 * \sum_i N_i Z_i^2/kT/epsilon0$.

Here, N_i is the concentration of the free ion i in the solvent, and Z_i is the free ion charge, and kT are the temperature energy. For example, at room temperature, in water with epsilon0=80, when ion concentration is $0.1Mol$, $akk_0 = 0.036/Bohr^2$. Large akk_0, there will be strongly ionic screening. Small value corresponds to weaker (longer length) screening. Note, when $akk_0 = 0$, there will be no Poisson-Boltzmann equation. When $akk_0 > 0$, there will be Poisson_Boltzmann equation: $\nabla[\epsilon(r)\nabla\phi(r)] - \epsilon_0 * k^2(r)\phi(r) = -4\pi(rho_{solute} - rho_{ion})$.

RHOMAX_DEBY = cut111

Default, 0.005 ($electron/Bohr^3$), used to control the turn on of $k^2(r)$ in the Poisson-Boltzmann equation. When rho_e > cut111, $k^2(r) = 0$.

RHOMIN_DEBY = cut222

Default, 0.0001 ($electron/Bohr^3$). cut111 and cut222 are used to control the turn on of $k^2(r)$ in the Poisson-Boltzmann equation. When rho_e < cut222, $k^2(r) = akk_0$. If one likes to put the free ion screening effects further away from the surface or solute molecule, we can set RHOMAX_DEBY, RHOMIN_DEBY smaller than RHOMAX_DIELECTRIC, RHOMIN_DIELETRIC. Note, here rho_e as described by DIELECTRIC_MODEL is used.

POISSON_MIX_SCHEME = LINEAR/PULAY (must be all capital)

The default is LINEAR. This is used to control which scheme is used to solve the Poisson equation with a spatially variation dielectric function. We have used an iterative scheme, which generates a polarization charge, and mix the polarization charge with previous steps. LINEAR, PULAY mixing schemes can be used. But

unless the iteration does not converge, one should use the simple LINEAR (default) mixing scheme. Usually this option is not needed.

POISSON_MIX_COEFF = param

Default 0.5. The mixing parameter used in the above mixing schemes. Smaller the value, more stable, but slower. Usually this option is not needed.

POISSON_ERROR = error

Default 1.E-10. The tolerance and stopping error for the Poisson equation iteration. Usually this option is not needed.

For the water, here is the recommended settings (according to: J. Chen. phys. 136, 064102.):

```
dielectric_const=78
surface_tension=50
pressure=-0.35
rhomax_dielectric=0.005
rhomin_dielectric=0.0001
```

2.4.10 IN.TDDFT_TIME file

File IN.TDDFT_TIME format,

```
0 ftddft(0)
1 ftddft(1)
...
N ftddft(N)
```

2.4.11 IN.TDDFTOPT file

In file IN.TDDFTOPT one can set:

1. OUT.MDDIPOLE.RSPACE=T/F, default=T.

If T, write file MDDIPOLE.RSPACE each time step.

2. OUT.MDDIPOLE.KSPACE=T/F, default=F.

If T, write file MDDIPOLE.KSPACE each time step.

3. TDDFT_SEED, default=12345.

Random seed for initializing the wavefunctions and velocities. If TDDFT_SEED=-1, the random seed will be set using the system_clock(). If this is not set, a default 12345 will be used.

4. Besides the TDDFT_SPACE and TDDFT_TIME, One can use file IN.TDDFTOPT to input external potential by setting TD_EFIELD or TD_EFIELD_LIST_*. The details are as follows.

(unit: energys–Hartree, coordinates—fractional in [0,1], time — fs)

```
TD_EFIELD=efield_type num_pars pars_list
TD_EFIELD_LIST_1=efield_type num_pars pars_list
TD_EFIELD_LIST_2=efield_type num_pars pars_list
TD_EFIELD_LIST_3=efield_type num_pars pars_list
...
TD_EFIELD_LIST_20=efield_type num_pars pars_list

---E(r,t)-------#---[efield_type num_pars pars_list]---------#
 E(r)=(x-x0)*Ex     [nontd_linear_x 4 x0,y0,z0,Ex]
 E(r)=(y-y0)*Ey     [nontd_linear_y 4 x0,y0,z0,Ey]
 E(r)=(z-z0)*Ez     [nontd_linear_z 4 x0,y0,z0,Ez]
 E(r)=Er*sqrt((x-x0)^2+(y-y0)^2+(z-z0)^2)^order,E(r)=Emax if E(r)>Emax
                    [nontd_well_poly 6 x0,y0,z0,order,Er,Emax]
-------------------------------------------------------------#
 E(r,t)=(x-x0)*Ex*delta(t-t0)
                    [td_kick_x 5 x0,y0,z0,Ex,t0]
 E(r,t)=(y-y0)*Ey*delta(t-t0)
                    [td_kick_y 5 x0,y0,z0,Ey,t0]
 E(r,t)=(z-z0)*Ez*delta(t-t0)
                    [td_kick_z 5 x0,y0,z0,Ez,t0]
-------------------------------------------------------------#
 E(r,t)=(x-x0)*Ex*exp(-(t-t0)**2/sigma**2)
                    [td_gaussian_x 6 x0,y0,z0,Ex,sigma,t0]
 E(r,t)=(y-y0)*Ey*exp(-(t-t0)**2/sigma**2)
```

```
                    [td_gaussian_y 6 x0,y0,z0,Ey,sigma,t0]
 E(r,t)=(z-z0)*Ez*exp(-(t-t0)**2/sigma**2)
                    [td_gaussian_z 6 x0,y0,z0,Ez,sigma,t0]
-------------------------------------------------------------#
 E(r,t)=(x-x0)*Ex*exp(-(t-t0)**2/sigma**2)*sin(w*t+k*z+phi)
                    [td_gaussian_sin_xz 9 x0,y0,z0,Ex,sigma,t0,w,k,phi]
 E(r,t)=(y-y0)*Ey*exp(-(t-t0)**2/sigma**2)*sin(w*t+k*z+phi)
                    [td_gaussian_sin_yz 9 x0,y0,z0,Ey,sigma,t0,w,k,phi]
 E(r,t)=(x-x0)*Ex*exp(-(t-t0)**2/sigma**2)*sin(w*t+k*y+phi)
                    [td_gaussian_sin_xy 9 x0,y0,z0,Ex,sigma,t0,w,k,phi]
 E(r,t)=(z-z0)*Ez*exp(-(t-t0)**2/sigma**2)*sin(w*t+k*y+phi)
                    [td_gaussian_sin_zy 9 x0,y0,z0,Ez,sigma,t0,w,k,phi]
 E(r,t)=(y-y0)*Ey*exp(-(t-t0)**2/sigma**2)*sin(w*t+k*x+phi)
                    [td_gaussian_sin_yx 9 x0,y0,z0,Ey,sigma,t0,w,k,phi]
 E(r,t)=(z-z0)*Ez*exp(-(t-t0)**2/sigma**2)*sin(w*t+k*x+phi)
                    [td_gaussian_sin_zx 9 x0,y0,z0,Ez,sigma,t0,w,k,phi]
-------------------------------------------------------------#
 E(r,t)=(x-x0)*Ex*cos(w*t+k*z+phi)
                    [td_cos_xz 7 x0,y0,z0,Ex,w,k,phi]
 E(r,t)=(y-y0)*Ey*cos(w*t+k*z+phi)
                    [td_cos_yz 7 x0,y0,z0,Ey,w,k,phi]
 E(r,t)=(x-x0)*Ex*cos(w*t+k*y+phi)
                    [td_cos_xy 7 x0,y0,z0,Ex,w,k,phi]
 E(r,t)=(z-z0)*Ez*cos(w*t+k*y+phi)
                    [td_cos_zy 7 x0,y0,z0,Ez,w,k,phi]
 E(r,t)=(y-y0)*Ey*cos(w*t+k*x+phi)
                    [td_cos_yx 7 x0,y0,z0,Ey,w,k,phi]
 E(r,t)=(z-z0)*Ez*cos(w*t+k*x+phi)
                    [td_cos_zx 7 x0,y0,z0,Ez,w,k,phi]
```

2.4.12 IN.WANNIER_*

When XCFUNCTIONAL = LDAWKM / LDAWKM2 and JOB = SCF (or other MD, but not JOB=WKM), one should prepare input file 'IN.WANNIER_PARAM' and

input wannier functions 'IN.WANNIER_*.u/d'.

IN.WANNIER_PARAM The parameters λ_k are input from the file 'IN.WANNIER_PARAM' It has a form like:

```
144 72 72      : n1w,n2w,n3w
5   1          : num_wannier_site, num_site(to repeat the Wannier)
-----------------------------------
1   -0.5       : lambda eV, up
2    0.0       : lambda eV, up
3    0.0       : lambda eV, up
4    0.0       : lambda eV, up
5    0.0       : lambda eV, up
-----------------------------------
1    0.0       : lambda eV, dn
2    0.0       : lambda eV, dn
3    0.0       : lambda eV, dn
4    0.0       : lambda eV, dn
5    0.0       : lambda eV, dn
-----------------------------------
1   0, 0, 0    : isite,ish1,ish2,ish3
-----------------------------------
```

Note, if there are num_site > 1, then there should be so many lines in the last section (isite,ish1,ish2,ish3), which spell out each site, how much shift. This feature is used to avoid to input too many Wannier functions if they are the same (just by a shift).

IN.WANNIER_*.u/d The Wannier functions are input from files: IN.WANNIER_0001.u, IN.WANNIER_0002.u, IN.WANNIER_0001.d, IN.WANNIER_0001.d etc (each file has one Wannier function, with the same format as charge density files) just like for the JOB=WKM calculations.

IN.WANNIER_PARAM3 When XCFUNCTIONAL = LDAWKM3 (or PBEWKM3) and JOB=SCF (or other MD, but not JOB=WKM), instead of inputting real space localized Wannier functions, one should input a IN.WG0 (IN.WG style, Bloch

state) with the same kpoints as the one in current JOB. The parameters λ_k are the shifts on these Bloch states, and they are input from the file 'IN.WANNIER_PARAM3' It has a form like:

```
2, 40, 1        : nkpt,mx0,islda
1   1           : iislda,kpt
1   -0.5        : im,lambda eV
2   -0.1        : im,lambda eV
3    0.0        : im,lambda eV
5    0.0        : im,lambda eV
.......
40   0.0        : im,lambda eV
-----------------------------------
1   2           : iislda,kpt
1   -0.3        : im,lambda eV
2   -0.2        : im,lambda eV
3    0.0        : im,lambda eV
5    0.0        : im,lambda eV
.......
40   0.0        : im,lambda eV
-----------------------------------
```

2.4.13 wavefunction files

If you want PWmat to read wavefunctions, you have to set 'IN.WG = T' in etot.input, and prepare wavefunction files.

Usually, the input wavefunction files are copied from previous PWmat calculations:

SPIN = 1 / 22 / 222 : copy 'OUT.WG' to 'IN.WG'

SPIN = 2 : copy 'OUT.WG' to 'IN.WG'; copy 'OUT.WG_2' to 'IN.WG_2'

You can check output wavefunction files for more details.

2.4.14 charge density files

If you want PWmat to read charge densities, you have to set 'IN.RHO = T' in etot.input, and prepare charge density files.

Usually, the input charge density files are copied from previous PWmat calculations:

SPIN = 1 / 22 : copy 'OUT.RHO' to 'IN.RHO'

SPIN = 2 : copy 'OUT.RHO' to 'IN.RHO'; copy 'OUT.RHO_2' to 'IN.RHO_2'

SPIN = 222 : copy 'OUT.RHO' to 'IN.RHO'; copy 'OUT.RHO_SOM' to 'IN.RHO_SOM'

You can check output charge density files for more details.

2.4.15 potential files

If you want PWmat to read potentials, you have to set 'IN.VR = T' in etot.input, and prepare potential files.

Usually, the input potential files are copied from previous PWmat calculations:

SPIN = 1 / 22 : copy 'OUT.VR' to 'IN.VR'

SPIN = 2 : copy 'OUT.VR' to 'IN.VR'; copy 'OUT.VR_2' to 'IN.VR_2'

SPIN = 222 : copy 'OUT.VR' to 'IN.VR'; copy 'OUT.VR_SOM' to 'IN.VR_SOM'; copy 'OUT.VR_SOM' to 'IN.VR_SOM'

You can check output potential files for more details.

Chapter 3

Output files

3.1 ASCII (human-readable) files

3.1.1 Standard output

Standard (on-screen) output contains the verbose information of each SCF calculation. Standard output has almost all the information for PWmat, but it might be messy to read. If we use: ">mpirun –np num PWmat > out &" to run our job, the standard output will be stored in the file "out".

3.1.2 REPORT

The "REPORT" file contains the most useful information in a concise way for the run. It might have the following format, with the corresponding explanation given in blue.

The REPORT header gives an explicit form for most the parameters to be used in the calculation (generated by defaults). An experienced user can change some of these parameters according to the need. However, one can also just use the short version etot.input to run PWmat, using the default values for most parameters.

```
1 4
PRECISION = AUTO
JOB = RELAX
IN.PSP1 = Si.SG15.PBE.UPF
IN.ATOM = atom.config
CONVERGENCE = EASY
ACCURACY = NORM
RELAX_DETAIL = 1  200  0.10000E-01  0  0.00000E+00   -0.10000E-02
VFF_DETAIL = 1 500 0.50000000E-02 30.00000 4.00000 0.00000 1.00000
EGG_DETAIL =            3           3           3
IN.RELAXOPT = F
SPIN = 1
CONSTRAINT_MAG = 0
QIJ_DETAIL = 0 1
PWSCF_OUTPUT = F
Ecut = 50.0000000000000
Ecut2 = 200.000000000000
Ecut2L = 200.000000000000
EcutP = 50.0000000000000
N123      =    36    36    36
NS123     =    70    70    70
N123L     =    36    36    36
MP_N123 =8 8 8 0 0 0
STRESS_CORR =   0.100000E+01  0.000000E+00  0.200000E+01  0.000000E+00
XCFUNCTIONAL = PBE
HSE_DETAIL    =      1.00000000    1     0.00000000    6 1 1
RELAX_HSE    =    20   0.50000E-01     2
VDW = NONE
LONDON_S6 = 0.000000000000000E+000
LONDON_C6(1) = -1.00000000000000
LONDON_RCUT = 0.000000000000000E+000
DFTD3_S6 = 1.00000000000000
DFTD3_RS6 = 1.21700000000000
DFTD3_S18 = 0.722000000000000
DFTD3_RS18 = 1.00000000000000
```

```
DFTD3_ALPHA6 = 14.0000000000000
DFTD3_VERSION = 3
DFTD3_3BODY = T
COULOMB = 0
IN.WG = F
OUT.WG = T
IN.RHO = F
OUT.RHO = T
IN.VR = F
OUT.VR = T
IN.VEXT = F
OUT.VATOM = F
OUT.REAL.RHOWF_SP= 0
OUT.FORCE = T
OUT.STRESS = F
IN.SYMM = T
NUM_KPT = 29
CHARGE_DECOMP = F
ENERGY_DECOMP = F
IN.SOLVENT = F
NUM_ELECTRON = 8.00000000000000
IN.NONSCF = F
NUM_BAND = 14
WG_ERROR = 1.000000000000000E-004
E_ERROR = 2.176910881600000E-005
RHO_ERROR = 5.000000000000000E-005
RHO_RELATIVE_ERROR = 0.00
FORCE_RELATIVE_ERROR = 0.00
SYS_TYPE = 1
IN.OCC = F -1
IN.CC = F
IN.OCC_ADIA = F
SCF_ITER0_1 =    6   4    3    0.0000     0.02500    1
SCF_ITER0_2 =   94   4    3    1.0000     0.02500    1
SCF_ITER1_1 =   40   4    3    1.0000     0.02500    1
NONLOCAL = 2
```

RCUT = 3.20000000000000
IN.PSP_RCUT1 = 3.20000000000000
MD_VV_SCALE = 100
LDAU_PSP1 = -1 2.721138638331097E-009 2.721138638331097E-009 eV
NUM_BLOCKED_PSI= F
WF_STORE2DISK = 0
NUM_DOS_GRID = 4000
NMAP_MAX = 50000
KERK_AMIN = 0.300000000000000
KERK_AMIX = 0.400000000000000
KERK_AMIX_MAG = 1.00000000000000
KERK_BMIX = 0.500000000000000
LDAU_MIX = 0.700000000000000
PULAY_WEIGHT_SPIN = 1.00000000000000
PULAY_WEIGHT_NS = 1.00000000000000
OUT.MLMD = F
NUM_MPI_PER_GPU = 1
---
total number of K-point: 29
the position of kpoint in G-space in unit 1/bohr and their weights.

0.00000	0.00000	0.00000	0.00195
-0.07653	0.07653	0.07653	0.01562
-0.15305	0.15305	0.15305	0.01562
-0.22958	0.22958	0.22958	0.01562
-0.30611	0.30611	0.30611	0.00781
0.00000	0.00000	0.15305	0.01172
-0.07653	0.07653	0.22958	0.04688
-0.15305	0.15305	0.30611	0.04688
-0.22958	0.22958	0.38263	0.04688
-0.30611	0.30611	0.45916	0.04688
-0.38263	0.38263	0.53568	0.04688
-0.45916	0.45916	0.61221	0.02344
0.00000	0.00000	0.30611	0.01172
-0.07653	0.07653	0.38263	0.04688
-0.15305	0.15305	0.45916	0.04688
-0.22958	0.22958	0.53568	0.04688

```
 -0.30611     0.30611     0.61221     0.02344
0.00000     0.00000     0.45916     0.01172
-0.07653     0.07653     0.53568     0.04688
-0.15305     0.15305     0.61221     0.02344
0.00000     0.00000     0.61221     0.00586
0.00000     0.15305     0.30611     0.04688
-0.07653     0.22958     0.38263     0.09375
-0.15305     0.30611     0.45916     0.04688
0.00000     0.15305     0.45916     0.04688
-0.07653     0.22958     0.53568     0.09375
-0.15305     0.30611     0.61221     0.04688
0.00000     0.15305     0.61221     0.02344
0.00000     0.30611     0.61221     0.01172
-----------------------------------------------
*********************************************
********** end of etot.input report ********
the above are etot.input.long, can be copied into etot.input
minimum n1,n2,n3 from Ecut2
     32.668       32.668       32.668
minimum n1L,n2L,n3L from Ecut2L
     32.668       32.668       32.668
*********************************************
Weighted average num_of_PW for all kpoint=           1614.291
***********************************
E_Hxc(eV)         -50.3360515430143
E_ion(eV)         -70.7758020324849
E_Coul(eV)         15.7159812114520
E_Hxc+E_ion(eV)  -121.111853575499
NONSCF     1           AVE_STATE_ERROR= 0.2502E+01
NONSCF     2           AVE_STATE_ERROR= 0.1139E+00
NONSCF     3           AVE_STATE_ERROR= 0.7470E-02
NONSCF     4           AVE_STATE_ERROR= 0.2787E-03
NONSCF     5           AVE_STATE_ERROR= 0.1178E-04
iter= 7 ave_lin= 3.5 iCGmth= 3
iter: SCF iter num; ave_line: CG line min num;
Ef(eV) = 0.7661625E+01 Fermi energy
```

err of ug = 0.5951E-07 Avery wave function (ug) error: $|(H-e)u_g|$.
dv_ave, drho_tot = 0.0000E+00 0.1150E+00
E_tot = -.21438186533335E+03 -.2144E+03

...
---
iter= 13 ave_lin= 2.0 iCGmth= 3
iCGmth: the method for wave func. Solver: 3, CG, 2: DIIS
Ef(eV) = 0.7534412E+01
err of ug = 0.3069E-05
dv_ave, drho_tot = 0.7833E-03 0.8271E-04
$|V_{in}-V_{out}|$ and $|rho_{in}-rho_{out}|$ errors in SCF (a.u)
E_tot = -.21450877234593E+03 -.6187E-05
Total energy (in eV), $E_{thisstep}-E_{laststep}(eV)$

E_Fermi(eV)= 7.53441246287097

Ef(eV) = 0.7534412E+01
dvE, dvE(n)-dvE(n-1) = 0.6136E-06 -.6254E-06
$dvE=\int|V_{in}-V_{out}|*rho(r)dr$ (a.u)
dv_ave, drho_tot = 0.7833E-03 0.8271E-04
err of ug = 0.3069E-05
---
ending_scf_reason = tol Etot_err 2.176910881600000E-005
Ewald = -.22851918665832E+03 Ewald energy (eV)
Alpha = -.82589820281155E+01 Pseudopotential Alpha energy (eV)
E_extV = 0.00000000000000E+00 0.0000E+00
energy due to ext potential: $\int V_{ext}(r)*rho(r)dr(eV)$
E_NSC = 0.17584653226505E+02 -.1047E-01 $\sum_i occ(i)*eigen(i)$, (eV)
E[-rho*V_Hxc]= 0.55177198633373E+02 0.1047E-01
$\int V_{Hxc}(r)*rho(r)dr$, V_{Hxc}: hartree, exchange, correction (eV)
E_Hxc = -.50492455446920E+02 -.5492E-02 The sum of Hartree, exchange and correlation energies (eV)
-TS = -.72446553375796E-07 -.3101E-09 occupation entropy term (eV)
E_tot(eV) = -.21450877234593E+03 -.6187E-05
total energy, and $E_{tot}(thisstep)-E_{tot}(lastSCFstep)(eV)$.

E_tot(Ryd) = -.15766104099826E+02 -.2274E-06

---
occup for: kpt=1,spin=1,m=(totN/2-2,totN/2+2) 1.000 1.000 1.000 0.000 0.000
eigen(eV) for: kpt=1,spin=1,m=(totN/2-2,totN/2+2) 7.245 7.245 7.245 9.802
9.802 ----------

E_Hart,E_xc,E_ion =0.15028296881420E+02
-.65520752328340E+02 -.62726190797084E+02
E_Hart: Coulomb interaction energy (eV)
E_Hxc+E_ion =-.11321864624400E+03
E_kin+E_nonloc =0.13548804265696E+03
E_rhoVext,E_IVext =0.00000000000000E+00 0.00000000000000E+00
$E_{rhoVext} = \int V_{ext} * rho * dr$, $E_{IVext} = Ion - V_{ext}$ energy (eV)
E_psiV,E_dDrho =-.11790779330926E+03 0.00000000000000E+00
$E_{dDrho} = \sum_i D_{j1j2} * < \beta_{j1} | \psi_i >< \psi_i | \beta_{j2} > (eV)$
ave(vtot):v0 =-.89820491030616E+01 v_0: (eV). $E_{\psi V} = \int rho * V_tot * dr(eV)$
ave(V_ion_s(or p,d))=ave(V_Hatree)=0; ave(Vtot)=ave(V_xc)=v0

**
**
RESULT: atom_move_step, E_tot: 0 -0.214508772345929E+03
**** finished input atom config calc. ***
**** following are atomic relaxation ***
**
**
force_max,stress .lt. tolforce,tol_stress finished
****
Eigen energies are values after setting ave(Vtot)=0"
For Vtot=V_ion+V_Hartree+V_xc, and
ave(V_ion+V_Hatree)=0, ave(V_xc).ne.0: E=E+v0
****
iislda,kpt= 1, 1 0.000000 0.000000 0.000000 kpoint in xyz unit
err of each states, A.U
0.573674E-04 0.215818E-04 0.218771E-04 0.234885E-04 0.133999E-04
0.135147E-04 0.137877E-04 0.312014E-04 0.322240E-04 0.308460E-04

```
0.941030E-04  0.297856E-04  0.262598E-04  0.298091E-04
eigen energies, in eV
-4.699596      7.245389      7.245391      7.245395      9.801827
9.801834      9.801837     10.593119     14.861113     14.861115
15.189885     18.458619     18.458622     18.458625
*********************************
*********************************
...
*********************************
*********************************
iislda,kpt=   1, 29  0.000000  0.306106  0.612211  kpoint in xyz unit
err of each states, A.U
0.471671E-04  0.471350E-04  0.449899E-04  0.449497E-04  0.463574E-04
0.412857E-04  0.499375E-04  0.499620E-04  0.315688E-04  0.404333E-04
0.960614E-04  0.962922E-04  0.470290E-03  0.637003E-03
eigen energies, in eV
-0.391756     -0.391756      3.413886      3.413891     11.472384
11.472387     12.224273     12.224277     17.739918     17.739925
20.202989     20.202993     23.588241     23.588244
*********************************
total computation time (sec)=           9
```

3.1.3 final.config

The final atom configuration when JOB = RELAX / MD / TDDFT / NAMD / DIMER will be stored in file 'final.config'.

When JOB = NEB, the final atom configuration of all images will be stored in file 'final.config'.

3.1.4 MOVEMENT

This is the file generated in RELAX, NEB and MD. It outputs the atom.config of every atomic movement steps (including the correction steps in the line minimization

of RELAX) in a single file, one after another. It contains the atomic position, atomic force sections. For MD, it also contains the velocity section. So, it can be copied to atom.config, to continue the run of MD. It can also be converted to other format for visualization (e.g., as animation), by using: ">convert_from_config.x MOVEMENT".

For JOB=NEB, the MOVEMENT contains the configurations for all the image points. Note, inside MOVEMENT, the new configuration is appended on the old ones already in the file. The format of MOVEMENT is the same as in atom.config.

3.1.5 RELAXSTEPS

The "RELAXTEPS" is the file concisely reports the atomic relaxation steps, and the energy and atomic forces at each atom, as well as the SCF convergence for each ab initio step calculations. A typical "RELAXSTEPS" file will look like:

```
It= 0 NEW E= -0.7526919500493E+03 Av_F= 0.17E+00 M_F= 0.32E+00 dE=.4E-04 dRho=.4E-03 SCF= 4
dL=-.70E-01 p*F =-0.38E-01 p*F0=-0.77E-01 Fch= 0.10E+01
It= 1 CORR E= -0.7527130487491E+03 Av_F= 0.18E+00 M_F= 0.37E+00 dE=.3E-04 dRho=.2E-03 SCF= 3
dL=-.14E+00 p*F=-0.23E-02 p*F0=-0.77E-01 Fch= 0.10E+01
It= 2 NEW E= -0.7527421363137E+03 Av_F= 0.10E+00 M_F= 0.20E+00 dE=.5E-04 dRho=.9E-03 SCF= 2
dL=0.49E-01 p*F =-0.19E-01 p*F0=-0.51E-01 Fch= 0.10E+01
It= 3 CORR E= -0.7527473988358E+03 Av_F= 0.12E+00 M_F= 0.23E+00 dE=.7E-05 dRho=.3E-03 SCF= 2
dL=0.78E-01 p*F= 0.80E-03 p*F0=-0.51E-01 Fch= 0.10E+01
```

It: The index of total line-minimization number (iteration, or step index).

NEW: this is a new line-minimization direction. The search direction p has changed.

CORR: this is a middle step in the line-minimization process (correction step). Its search direction p is the same as in previous steps (all the way to the last NEW step). Note the energy of this trial step can be higher than previous step. So, to see the convergence, only the energies for the NEW steps should be used.

E: total energy of this step in eV;

Av_F, M_F: average and maximum atomic forces (eV/Å);

Av_e: average stress ($eV/NATOM$), Natom is the total number of atoms;

dE: the SCF iteration $E(n) - E(n-1)(eV)$. This is used to judge whether the SCF iteration is converged. Note, this is not the dE between this relaxation step and previous relaxation step!

dRho: the SCF iteration $|rho(n) - rho(n-1)|$ relative error. This is used to judge whether the SCF is converged.

SCF: the SCF iteration number for this step.

dL: the movement $|R - R(new\_initial)|$ of this step (in atomic unit $Bohr$). $R(new\_initial)$ is the initial atomic position of this line-minimization direction. Note, for the NEW step, there is already one dL. In another word, the dL shown in the NEW line is actually the $|R - R(new\_initial)|$ for the R in the following CORR line (i.e, it is the length of the first trial step). Similarly, the dL of one CORR line, is the dL of the R in the following line. The dL's in the NEW, and subsequent CORR (before the next NEW) lines are in the same search direction, and all measured from the beginning point of this new line direction.

p*F: the force of the current step project to the search direction. Note, the purpose of the line-minimization is to make $p * F$ zero (it uses linear interpolation of $p * F$) to predict the next step size (dL) in this line-minimization.

p*F0: the same as $p * F_0$, however, not use the force of this configuration, but use the force of $R(new\_initial)$. Thus this $p * F_0$ is the same throughout one line-minimization direction.

Fch: the force check, calculated as $dL * (F + F_0)/2/dE$, dL is the displacement for this step from the $R(new\_initial)$, $F, F0$ are the forces are the two ends of this step, (F is the force at the current position, F_0 is the initial force at $R(new\_initial)$. dE is the total energy difference of this step (the current energy minus the initial energy at the beginning of the new search direction). $Fch = 1$ indicates all the calculations are accurate. Note, for single precision GPU calculation, the dE is usually less accurate than $dL * (F + F_0)/2$, so Fch not equaling 1 can still be fine, since the force is good, and the relaxation algorithms are based on force, not the total energy. When using HSE functional to do relaxation, the RELAXSTEPS will like this:

```
1 hse= 1 HSE E= -0.3021519099504E+05 Av_F= 0.49E-01 M_F= 0.20E+00 dE=.3E-01 dRho=.5E-03 SCF=40
2 It= 0 TRIAL E= -0.3021525446280E+05 Av_F= 0.22E-01 M_F= 0.12E+00 dE=.2E-02 dRho=.1E-02 SCF= 6
dL=0.30E-01 p*F=-0.10E-01 p*F0=-0.27E-01 Fch= 0.10E+01
3 It= 1 CORR E= -0.3021527108036E+05 Av_F= 0.20E-01 M_F= 0.84E-01 dE=.6E-03 dRho=.2E-02 SCF= 3
dL=0.49E-01 p*F= 0.42E-03 p*F0=-0.27E-01 Fch= 0.11E+01
```

```
...
15 It= 13 TRIAL E= -0.3021529742117E+05 Av_F= 0.21E-02 M_F= 0.66E-02 dE=.4E-04 dRho=.5E-04 SCF= 3
dL=-.48E-02 p*F=-0.15E-03 p*F0=-0.14E-02 Fch= 0.12E+01
17 hse= 2 HSE E= -0.3021523919362E+05 Av_F= 0.57E-02 M_F= 0.35E-01 dE=.3E-02 dRho=.3E-04 SCF=26
18 It= 0 TRIAL E= -0.3021525066203E+05 Av_F= 0.42E-02 M_F= 0.18E-01 dE=.8E-02 dRho=.2E-02 SCF= 3
dL=0.10E-01 p*F= 0.46E-04 p*F0=-0.31E-02 Fch= 0.82E+01
19 It= 1 CORR E= -0.3021523932495E+05 Av_F= 0.43E-02 M_F= 0.19E-01 dE=.2E-03 dRho=.3E-03 SCF= 3
dL=0.98E-02 p*F=-0.10E-04 p*F0=-0.31E-02 Fch= 0.94E-01
...
```

The new item is HSE. That means in this iteration, PWmat will use HSE functional to do the relaxation.

3.1.6 MDSTEPS

The "MDSTEPS" is the file concisely describes the steps of a molecular dynamics simulation. It can have the following format:

For NVE and NVT systems:

```
Iter(fs)= 0.100000E+01 Etot,Ep,Ek(eV)= -0.1698558736E+05 -0.1699177467E+05
0.6187317043E+01 Temp(K)= 598.34048 aveTemp(K)= 598.34048 dE= -.46E-03
dRho= 0.35E-03 SCF= 8 dL= 0.14E-01 Fcheck= 0.106E+01
Iter(fs)= 0.200000E+01 Etot,Ep,Ek(eV)= -0.1698558872E+05  -0.1699171678E+05
0.6128058892E+01 Temp(K)= 592.60996 aveTemp(K)= 595.47522 dE= 0.37E-04
dRho= 0.42E-03 SCF= 5 dL= 0.14E-01 Fcheck= 0.103E+01
Iter(fs)= 0.300000E+01 Etot,Ep,Ek(eV)= -0.1698559089E+05  -0.1699161871E+05
0.6027821480E+01 Temp(K)= 582.91656 aveTemp(K)= 591.28900 dE= -.16E-03
dRho= 0.51E-03 SCF= 5 dL= 0.14E-01 Fcheck= 0.102E+01
...
```

Iter is the simulation time in fs.

Etot is the total energy (DFT energy plus kinetic energy) in eV.

Ep is the potential energy (here, DFT energy) in eV.

Ek is the kinetic energy in eV.

Temp is the temperature calculated from the E_k in K

aveTemp is the average temperature in K

dE is the $E(n) - E(n-1)$ in the SCF iteration (eV).

drho $= |rho_{in} - rho_{out}|$ relative error, in the SCF calculation.

SCF is the number of SCF iterations for this MD step.

For NPT systems:

```
Iter(fs)= 0.100000E+01 Etot,Ep,Ek(eV)= -0.1698570268E+05  -0.1699177647E+05
0.6073782319E+01 Temp(K)= 587.36118 aveTemp(K)= 587.36118 Press(Hartree/bohr^3)=
0.75287E-05 aveP(Hartree/bohr^3)= 0.75287E-05 dE= -.52E-03 dRho= 0.35E-03 SCF= 8
dL= 0.14E-01 Fcheck= 0.116E+01
Iter(fs)= 0.200000E+01 Etot,Ep,Ek(eV)= -0.1698575833E+05  -0.1699172623E+05
0.5967897923E+01 Temp(K)= 577.12169 aveTemp(K)=582.24144 Press(Hartree/bohr^3)=
0.72836E-05 aveP(Hartree/bohr^3)= 0.74061E-05 dE= -.89E-03 dRho= 0.82E-03 SCF= 4
dL= 0.14E-01 Fcheck=0.116E+01
Iter(fs)= 0.300000E+01 Etot,Ep,Ek(eV)= -0.1698579068E+05  -0.1699163917E+05
0.5848493662E+01 Temp(K)= 565.57478 aveTemp(K)=576.68589 Press(Hartree/bohr^3)=
0.82208E-05 aveP(Hartree/bohr^3)= 0.76777E-05 dE= 0.33E-03 dRho= 0.44E-03 SCF= 5
dL= 0.14E-01 Fcheck=0.112E+01
...
```

Iter is the simulation time in fs.

Etot is the total energy (DFT energy plus kinetic energy) in eV.

Ep is the potential energy (here, DFT energy) in eV.

Ek is the kinetic energy in eV.

Temp is the temperature calculated from the E_k in K

aveTemp is the average temperature in K

Press is the pressure in $Hartree/bohr^3$

aveP is the average pressure in $Hartree/bhor^3$

dE is the $E(n) - E(n-1)$ in the SCF iteration (eV).

drho = $|rho_{in} - rho_{out}|$ relative error, in the SCF calculation.

SCF is the number of SCF iterations for this MD step.

3.1.7 DIMERSTEPS

DIMERSTEPS can be used to check the convergency during dimmer calculation, datas in DIMERSTEPS:

STEP: the index of current transition step.
FORCE MAX: the maximum of current transition step, unit eV/Angstrom.
FORCE ROT/dR: the rotational force / dimer_dR, unit eV/Angstrom$\hat{2}$,
used to judge the convergency of rotation.
ENERGY: the energy of the dimer center, unit eV.
CURVATURE: the curvature along the dimer.
ANGLE ROT: the angle rotated in rotational steps, unit rad.
NSTEPS ROT: the number of rotational steps in current transition step.
DISTANCE TRANS: the step size of current transition step, unit Bohr.

If the dimer converged, the FORCE MAX should be less then DIMER_TOL_FORCE, the CURVATURE negative, and CURVATURE and FORCE ROT/dR absolutely small.

3.1.8 NEB.BARRIER

The "NEB.BARRIER" is the file concisely report the energies along the images for different relaxation iteration steps. One can yield the barrier height and profiles from NEB.BARRIER. It has the following format:

```
iter= 19 Etot(eV),dist(Bohr),angle(cos(th))
 0 -0.75306186045042E+03 0.504486E+00 0.000000E+00
 1 -0.75305820517778E+03 0.520270E+00 0.944578E+00
 2 -0.75304052843358E+03 0.530724E+00 0.846617E+00
 3 -0.75296036069356E+03 0.526520E+00 0.355627E+00
 4 -0.75266754347227E+03 0.517507E+00 0.883061E+00
 5 -0.75234053674623E+03 0.512514E+00 0.961894E+00
 6 -0.75234044035416E+03 0.517438E+00 0.961928E+00
 7 -0.75266732167841E+03 0.526413E+00 0.883176E+00
 8 -0.75296021410969E+03 0.530651E+00 0.356206E+00
 9 -0.75304050727950E+03 0.520291E+00 0.846314E+00
10 -0.75305820226225E+03 0.504589E+00 0.944466E+00
11 -0.75306185743092E+03 0.000000E+00 0.000000E+00
```

This means Nimage=10. The $Etot(eV)$ indicates the total energy of this image. Dist

is the distance between the neighboring images (between image and image+1). For good NEB run, the distance should be roughly the same. Angle is the $cos\theta$ of the angle theta between two $R(image+1) - R(image)$, and $R(image) - R(image-1)$. For good NEB run, $cos\theta$ should be close to 1 (especially around the barrier height). In practice, it should be fine as long as the $cos\theta$ is close to 1 near the barrier height. Also, for a good NEB run, the distance between the images should be roughly equal. Iter=19 means this is the 19th line minization result. In NEB.BARRIER, it writes out the results for every relaxation iterations.

3.1.9 ORIGIN.INDEX

The index of atoms of input and output structure might be different, because PWmat will reorder the atoms, and store the corresponding relationship in 'ORIGIN.INDEX'.

An example input structure (partial):

```
Position, move_x, move_y, move_z
   8    0.431149725170    0.375645979880    0.608062611377   1   1   1
   1    0.490833544711    0.442692872224    0.649907571525   1   1   1
   1    0.491412376841    0.301306183455    0.582602305387   1   1   1
   8    0.395349708974    0.690647389021    0.505432691750   0   0   0
   1    0.327545006230    0.620310889932    0.489396818540   0   0   0
   1    0.477944633359    0.655553298285    0.463640966751   0   0   0
   8    0.667084990716    0.439013253319    0.395398851505   0   0   0
   1    0.732866540515    0.463073730048    0.325441623924   0   0   0
   1    0.580819484671    0.432493539466    0.347228414125   0   0   0
```

After running PWmat, the output structure (partial):

```
 Position, move_x, move_y, move_z
   8    0.431149725    0.375645980    0.608062611     1  1  1
   8    0.395349709    0.690647389    0.505432692     0  0  0
   8    0.667084991    0.439013253    0.395398852     0  0  0
   1    0.490833545    0.442692872    0.649907572     1  1  1
   1    0.491412377    0.301306183    0.582602305     1  1  1
   1    0.327545006    0.620310890    0.489396819     0  0  0
```

```
1    0.477944633    0.655553298    0.463640967    0  0  0
1    0.732866541    0.463073730    0.325441624    0  0  0
1    0.580819485    0.432493539    0.347228414    0  0  0
```

And the 'ORIGIN.INDEX':

```
  1  1 1 1      1   8
  4  0 0 0      2   8
  7  0 0 0      3   8
  2  1 1 1      4   1
  3  1 1 1      5   1
  5  0 0 0      6   1
  6  0 0 0      7   1
  8  0 0 0      8   1
  9  0 0 0      9   1
```

1st column: the original index in input structure of this atom

2nd-4th column: the degrees of freedom of this atom in the x, y, z direction

5th column: the reordered index of this atom

6th column: the atomic number of this atom

The above example shows that the atom index has been reordered.

3.1.10 OUT.KPT

'OUT.KPT' contains the k-point vectors and their weights used in PWmat calculation. It has the following format:

```
2                       # nkpt
2 1.0000                # iflag, a0
0.250 0,250 0.250 0.25 # ak1, ak2, ak3, weight
0.250 0.250 0.750 0.75
```

nkpt: The number of k-points.

iflag:

1. iflag = 1, the k-points are in x, y, z directions (which is defined by the x, y, z in AL(3,3) in "atom.config").

2. iflag = 2, the k-points are in the reciprocal lattice of the super cell AL(3,3). "a_0" will not be used.

$\mathbf{a}_0$: only used when iflag = 1. (in atomic unit *Bohr*)
ak1,ak2,ak3:

1. iflag = 1, the k-points are defined as:

$$k_x = 2 * \pi * ak_1/a_0 \tag{3.1}$$

$$k_y = 2 * \pi * ak_2/a_0 \tag{3.2}$$

$$k_z = 2 * \pi * ak_3/a_0 \tag{3.3}$$

2. iflag = 2, the k-points are defined as:

$$k = G_1 * ak_1 + G_2 * ak_2 + G_3 * ak_3 \tag{3.4}$$

Here G_1, G_2, G_3 are the reciprocal lattice vector of lattice AL(3,3).

weight: This is the weight of this reduced k-point (it can represent several symmetric k-points). This is used for SCF calculations, and the total weight as the sum of individual k-points should be 1.

3.1.11 OUT.SYMM

'OUT.SYMM' contains all the symmetry operations of input structure. It has the following format:

```
12 24 | nsym, nrot
"identity and corresponding fractional translation "
1   0   0
0   1   0
0   0   1
0.000 0.000 0.000
"180 deg rotation - cart. axis [0,0,1]..."
```

```
-1  0   0
0 -1   0
0  0  -1
0.000 0.000 -0.500
...
180 deg rotation - cryst. axis [1,1,0]
-1  1  -1
 0  1   0
 0  0  -1
```

The first line is the two variables: "nsym" and "nrot". "nsym" is the number of the crystal symmetries operations (space group) and "nrot" is the number of the crystal Bravais lattice symmetries (only for the lattice, not considering the atoms, thus nrot is always larger than nsym). For PWmat, only nsym is used. For the rest of the file, there will be "nsym" operations, each has this following format:

```
"180 deg rotation - cart. axis [0,0,1]..."
-1  0  0            # s(1,1), s(2,1), s(3,1)
 0 -1  0            # s(1,2), s(2,2), s(3,2)
 0  0 -1            # s(1,3), s(2,3), s(3,3)
 0.000 0.000 -0.500 # l(1), l(2), l(3)
```

The first line is the explanation of this symmetry operation. The following three line defines the point group rotation matrix $s(3,3)$ around the origin (x_1, x_2, x_3 = 0, 0, 0) point. The rotation $s(3,3)$ will convert a real space point (x_1, x_2, x_3) (in lattice cell fractional coordination) into another point following:

$$y_1 = s(1,1) * x_1 + s(2,1) * x_2 + s(3,1) * x_3 \tag{3.5}$$

$$y_2 = s(1,2) * x_1 + s(2,2) * x_2 + s(3,2) * x_3 \tag{3.6}$$

$$y_3 = s(1,3) * x_1 + s(2,3) * x_2 + s(3,3) * x_3 \tag{3.7}$$

The next line defines the fractional translation in the space group. Thus we have:

$$y_1 = y_1 + l(1) \tag{3.8}$$

$$y_2 = y_2 + l(2) \tag{3.9}$$

$$y_3 = y_3 + l(3) \tag{3.10}$$

3.1.12 OUT.IND_EXT_KPT

'OUT.IND_EXT_KPT' contains the relationship between the reduced kpoint and the extended kpoint, and the symmetry operation which rotates the reduced kpoint to the extended kpoint.

3.1.13 OUT.OCC

The occupation and eigen enengy of eigen states. The unit of eigen energy is eV. It has the following format:

```
KPOINTS      1:    0.0000    0.0000    0.0000    # the first kpoint
 NO.   ENERGY(eV) OCCUPATION    # index, eigen energy, occupation
      1     -4.6120   0.00274
      2      7.3497   0.00274
      3      7.3497   0.00274
...
KPOINTS      2:    0.0680    0.0680   -0.0680    # the second kpoint
 NO.   ENERGY(eV) OCCUPATION
      1     -4.4513   0.02195
      2      5.9974   0.02195
      3      7.1073   0.02195
...
```

3.1.14 OUT.VATOM

The atom center potential for SCF or MD simulation, which is an average using atomic charge density dot-product with the total potential. The unit of potential is eV.

It has the following format when SPIN = 1:

```
   27   average Vtot at each atom in eV   # There will be Natom lines
25 0.297 0.193 0.300 -58.452  # atomic number, coordinate, potential
25 0.297 0.527 0.300 -58.448
   ...
```

It has the following format when SPIN = 2:

```
   27   average Vtot at each atom in eV   # There will be Natom lines
25 0.297 0.193 0.300 -58.452 -58.453 # atomic number, coordinate, spinup/dn potential
25 0.297 0.527 0.300 -58.448 -58.448
   ...
```

3.1.15 OUT.FERMI

'OUT.FERMI' stores the FERMI energy in SCF calculation, which can be used in plotting bandstructure or density of states. The unit of fermi energy is eV.

It has the following format:

```
 Fermi Energy =    7.64127414615752       eV
```

3.1.16 OUT.FORCE

'OUT.FORCE' stores the calculated force of all atoms. The unit of force is $eV/$Å.

It has the following format:

```
 ****** force (eV/A) ******************
zatom, fx, fy, fz
...
 ****** total force ******************
   0   tx, ty, tz
 ****** force after remove total force (eV/A) ***
zatom, fax, fay, faz
...
```

WARNING: The force felt by the atom is -(fx/y/z).

3.1.17 OUT.STRESS

The stress tensor output file.

3.1.18 OUT.QDIV

The atomic charge on each atom.

3.1.19 OUT.ENDIV

The decomposed atomic energy on each atom.

3.1.20 OUT.ATOMSPIN

The local charge and magnetic moment in spin-polarized calculations when SPIN = 222.

3.1.21 OUT.BORN_CHARGE

The effective born charge when E_FINITE = T.

3.1.22 MDDIPOLE.RSPACE

$$P_\sigma(t) = \int \rho(r,t) * \sigma d^3 r, \sigma \in x, y, z$$

3.1.23 MDDIPOLE.KSPACE

$$\frac{\partial P(t)}{\partial t}$$

3.1.24 MDINT.RHOVEXT

$$\int \rho(r,t) * V_{external}(r,t) d^3 r$$

3.1.25 DOS.totalspin

The density of states results when JOB = DOS.

3.2 BINARY files

3.2.1 OUT.WG*

The wave function output file when OUT.WG = T. When SPIN = 1 / 22 / 222, PWmat will only output 'OUT.WG'; when SPIN = 2, PWmat will output 'OUT.WG' and 'OUT.WG_2', which are spin up and spin down components, respectively.

For files like wave function file, they have the following format (can be written or read out like this):

```
OPEN (11, FILE = 'OUT.WG', FORM = 'UNFORMATTED')
READ (11) N1, N2, N3, MX
ALLOCATE (UG(N1*N2*N3,MX))
READ (11) ECUT
READ (11) AL ! READ LATTICE AL(3,3)
```

```
READ (11) NNODES
DO KPT = 1, NKPT
   DO IWAVEFUN = 1, NBLOCK_BAND_MX
       READ (11) UG(:,IWAVEFUN)
   END DO
END DO
```

You can check the source code 'plot_wg.f90' in the directory 'src_utils/plot_wg/' of PWmat package for more details.

3.2.2 OUT.RHO*

OUT.RHO, OUT.RHO_2 & OUT.RHO_SOM The charge density output file when OUT.RHO = T. When SPIN = 1 / 22, PWmat will only output 'OUT.RHO' for the total charge density; when SPIN = 2, PWmat will output 'OUT.RHO' and 'OUT.RHO_2', which are spin up and spin down components, respectively; when SPIN = 222, PWmat will output 'OUT.RHO' and 'OUT.RHO_SOM', which are total charge density and a complex 2x2 spin matrix density, respectively.

For files like charge density file, they have the following format (can be written or read out like this):

```
OPEN (11, FILE = 'OUT.RHO', FORM = 'UNFORMATTED')
REWIND (11)
! READ THE FFT GRIDS
READ (11, IOSTAT = IERR) N1, N2, N3, NNODES, NSTATE
IF (IERR /= 0) THEN
    REWIND (11)
    READ (11) N1, N2, N3, NNODES
    NSTATE = 1
END IF
READ (11) AL ! READ LATTICE AL(3,3)
NR = N1 * N2 * N3
NR_N = NR / NNODES
ALLOCATE (VR_TMP(NR_N), VR(N1,N2,N3))
DO IST = 1, NSTATE
    DO IREAD = 1, NNODES
```

```
      READ (11) VR_TMP
      DO II = 1, NR_N
         JJ = II + (IREAD-1)*NR_N
         I = (JJ-1)/(N2*N3) + 1
         J = (JJ-1-(I-1)*N2*N3)/N3 + 1
         K = JJ - (I-1)*N2*N3 - (J-1)*N3
         VR(I,J,K) = VR_TMP(II)
      END DO
   END DO
END DO
```

You can check the source code 'convert_rho.f90' in the directory 'src_utils/convert_rho/' of PWmat package for more details.

OUT.RHO_4DIELECTRIC When IN.SOLVENT = T and OUT.SOLVENT_CHARGE = T, PWmat will output the rho_e in 'OUT.RHO_4DIELECTRIC'. The rho_e is used to determine the position dependent dielectric function. One can plot the isosurface plots using RHOMAX_DIELECTRIC and RHOMIN_DIELECTRIC values described in 'IN.SOLVENT' file to view where are the turn-in/off surfaces of the dielectric constant.

OUT.RHO_POLARIZE When IN.SOLVENT = T and OUT.SOLVENT_CHARGE = T, PWmat will output the solvent induced polarization charge in 'OUT.RHO_POLARIZE'.

OUT.RHOP_VHION When IN.SOLVENT = T and OUT.SOLVENT_CHARGE = T, PWmat will output the polarization charge multiplied by the electric static potential of the solute molecule in 'OUT.RHOP_VHION'. The integrate of this density is the polarization energy. One can use this to see where the polazation energy comes from.

3.2.3 OUT.V*

The potential files are in the charge density format.

OUT.VR, OUT.VR_2, OUT.VR_SOM & OUT.VR_DELTA The potential output file when OUT.VR = T. When SPIN = 1 / 22, PWmat will only output 'OUT.VR' for the total potential; when SPIN = 2, PWmat will output 'OUT.VR' and 'OUT.VR_2', which are spin up and spin down components, respectively; when SPIN = 222, PWmat will output 'OUT.VR' 'OUT.VR_SOM' and 'OUT.VR_DELTA', which are total potential, a complex 2x2 spin matrix potential and a real up-down potential, respectively.

OUT.VR_hion & OUT.VR_hion_solvent When OUT.VR = T, PWmat will output the electrostatic potential (Hartree + Vion, without XC potential) in 'OUT.VR_hion'. When IN.SOLVENT = T, will also output electrostatic + polarized potential in 'OUT.VR_hion_solvent'.

OUT.V_POLARIZE When IN.SOLVENT = T and OUT.SOLVENT_CHARGE = T, PWmat will output the polarized potential generated by the polarization charge in 'OUT.V_POLARIZE'.

It has the charge density format.

OUT.VEXT_TDDFT The external potential in real space when set parameter 'itype_space=1/2/3' in tag 'TDDFT_SPACE'.

3.2.4 OUT.HSEWR(i)

When XCFUNCTIONAL = HSE / B3LYP and OUT.HSEWR = T, PWmat will output the real space wave functions for the Fock exchange kernel for all the extended k-points on every GPU in 'OUT.HSEWR(i)' files, i is the index of GPUs.

3.2.5 OUT.REAL.RHOWF_SP

When the first value of OUT.REAL.RHOWF_SP is not 0, PWmat will output the real space charge density or wave function in file 'OUT.REAL.RHOWF_SP'.

3.2.6 OUT.GKK

The momentum operator $i * k_x, i * k_y, i * k_z$.

3.2.7 bpsiiofil10000(x)

When JOB = DOS, PWmat will output the wave function to atomic orbital projection of every kpoint in 'bpsiiofil10000(x)' files, x is the index of kpoints.

3.2.8 OUT.SPIN_X/Y/Z

When SPIN = 222, PWmat will output the spin charge density in x/y/z direction at every r point in 'OUT.SPIN_X/Y/Z' files.

It has the charge density format.

3.2.9 OUT.EIGEN

The eigen energies output file.

3.2.10 OUT.overlap_uk*

OUT.overlap_uk The Bloch state overlap between different k-points when **DOS_DETAIL = 1 ...**

OUT.overlap_uk.2 The Bloch state overlap between different k-points when **DOS_DETAIL = 2 ...**

3.2.11 OUT.momentK.(x)

When JOB = MOMENT, PWmat will output the resulting momentum matrix for every kpoint in 'OUT.momentK.(x)' files, x is the index of extended (expanded) kpoints set.

It has the following format (M_x(mx,mx,kpt,spin) is a complex*16 matrix) :

```
num_ext_kpt, mx, islda
do kpt=1,num_ext_kpt
do spin=1,islda
M_x(:,:,kpt,spin)
M_y(:,:,kpt,spin)
M_z(:,:,kpt,spin)
enddo
enddo
```

Chapter 4

The basic calculations

4.1 Self-consistent calculations(JOB=SCF)

This is a one-shot DFT calculation for a fixed atomic position. It can be used to study the total energy, the magnetic moment, the charge density, the electronic structure, etc. It will not move the atoms. The charge density will be iteratively calculated, until it converges (input charge density equals the output charge density). If subsequent runs (e.g., for NONSCF or DOS) are expected, one should set: OUT.WG = T, OUT.RHO = T, OUT.VR = T (they are all defaults for SCF runs), so they will output files: OUT.WG, OUT.RHO, OUT.VR for subsequent uses. For bandstructure plot and DOS plot, remember to copy OUT.FERMI (so the Fermi energy can be read out).

4.2 None self-consist calculations(JOB=NONSCF)

This is usually used, following a SCF calculation, to study the electronic structure of the system, in particular the bandstructure. It can use the OUT.VR from the previous calculation (copy them to IN.VR, and set IN.VR to T in etot.input), but with different k-points in IN.KPT, to study the bandstructure. It can also be used to study some special cases (e.g., with patched together potential, or charge density). Note, in NONSCF, it can still generate the potential from an input charge density. One needs to set IN.VR=T

(to input potential from IN.VR), or IN.RHO=T (to input charge density from IN.RHO, then generate the potential). It will then simply calculate the eigen energies (e.g., for different k-points listed in IN.KPT).

To do a bandstructure calculation, it usually run PWmat in the following procedure: Set Monkhorst-Pack line in etot.input: "MP_N123 = nk1, nk2, nk3, sk1, sk2, sk3", then run a SCF calculation, get OUT.VR, and copy OUT.VR to IN.VR. edit the high symmetry kpoints by hand, then run a NONSCF calculation using this IN.KPT (IN.KPT=T, IN.VR=T). The bandstructure information will then be reported in "REPORT". One can use "plot_band_structure.x" for post-process to view the bandstructure.

4.3 Density of States Calculations(JOB=DOS)

This usually also follows from one SCF calculation (just like for NONSCF), then to calculate the DOS in this step. In other words, there need to have three calculations in order to get DOS. As for NONSCF, one first gets OUT.VR from SCF calculation. Then in DOS calculation, copy OUT.VR as IN.VR, set IN.VR = T (read this IN.VR). Also, one might want to use more k-points for a nice DOS. To do that, one can use a larger Monkhorst-Pack grids in etot.input (MP_N123 = nk1, nk2, nk3, sk1, sk2, sk3), to generate a new IN.KPT. Then, one needs to do a JOB = NONSCF calculation to get the eigen energies (stored in OUT.EIGEN) and eigen wave functions (OUT.WG). (If one doesn't want to use more k-points, then one can use the SCF calculation's eigen energies and wave functions, thus skip the JOB = NONSCF calculation step). After this, one can copy OUT.WG to IN.WG, and do a JOB = DOS calculation. The PWmat will output a file: DOS.totalspin (if SPIN=2, there will also be DOS.spinup, DOS.spindown). The format in DOS.totalspin is (each line) (example):

```
Energy Total Zn-s Zn-p Zn-d O-s O-p O-d
```

One can plot this file for graphics. The default energy smoothing/broadening parameter is 0.1 eV. If one wants to have different broadening parameter, or have partial

DOS for different atoms, one can use the postprocessing utility code: plot_dos.x. In order to have atom selective partial DOS, one needs to provide a modified atom.config file, with the position section looks like:

```
30 0.952534560 0.363594470 0.382027650 1 1 1 w1
30 0.540553000 0.850230410 0.966359450 1 1 1 w2
...
16 0.242857140 0.140553000 0.684331800 1 1 1 w3
```

Here w1, w2, w3 are the weights for this atom in the partial DOS. The plot_dos.x also uses bpsiiofil10000x, which are the eigen wavefunciton to atomic orbital projection coefficients for different k-points x, which is output from the JOB = DOS run.

4.4 Atomic Relaxation(JOB=RELAX)

This is to relax the atomic positions following the DFT energy and force. It will generate the RELAXSTEPS, and MOVEMENT files. If it is not fully relaxed, the MOVEMENT can be copied to atom.config (remove all the other iterations, except the last one), then run the PWmat again. Pay attention to FORCE_RELATIVE_ERROR. This parameter is used to stop the SCF iterations. It takes the last iteration average force, and the estimated force error (estimated from $|V_{in} - V_{out}|$ in SCF calculations), if the estimated_force_error is less than last_iteration_force*FORCE_RELATIVE_ERROR, then it will jump out the SCF loop. So, smaller FORCE_RELATIVE_ERROR (and larger niter1), more SCF loop might be carried out, and more accurate will be the force. This might be particularly critical if very accurate final atomic positions are needed. The default FORCE_RELATIVE_ERROR for RELAX is 0.003. There are two relaxation methods: imth=1, conjugated gradient method; imth=2, BFGS method (the imth is specified in line RELAX_DETAIL). Their performances are similar. More advanced methods will be introduced in later version of PWmat. When JOB = RELAX is used, one can also include an etot.input line: "RELAX_DETAIL = imth, nstep, force_tol". If the

max_force becomes smaller than force_tol($a.u$), the relaxation step will stop (before nstep).

4.5 Nudged Elastic Band Calculations(JOB=NEB)

Nudged Elastic Band (NEB) method is often used to calculate the potential barrier from one local minimum configuration to another local minimum configuration. In order to do NEB calculation, the two local minimum must be known already. They can be calculated by JOB=RELAX, with their atomic configurations being atom1.config, atom2.config, and energies being E_1,E_2. The idea of NEB is to use a string of images (configuration points) between the two end points (atom1.config, atom2.config). This can guarantee the path can go from one configuration to the other configuration. To avoid the force from the potential (DFT energy) to move the images away from the barrier saddle point (which lower the potential energy), the tangent component of the potential force will be removed. To avoid the force of the elastic string from moving the string away from the saddle point (corner cutting, since an elastic string like to have the minimum length), the perpendicular (to the string tangent) component of the string force will be removed. So the force of the string will only maintain equal distances between the images. This is the essence of the NEB. However, after such modification, there is no guarantee the remaining total force can be written down as a gradient of a potential (i.e, the vortices of the force might not be zero). This can cause significant difficulty in the relaxation procedure to make the force zero (e.g., if one just follows the force to make atomic movement, it is possible that the relaxation iteration can end up in an infinite loop). Another common problem is that string is not smooth, with the vector $R(image+1)-R(image)$ and $R(image)-R(image-1)$ having an angle not close to 180 degree. Thus, there are several points need to be considered when doing a JOB=NEB calculation. First, it is better to use imth=3 (steepest decent for the atomic relaxation method). This is because the CG or BFGS method, which assumes a parabolic potential and the related force might no longer work in NEB. Second, we have

implemented two types of string (to deal with the string force). type_string=1 means the original NEB string (remove the perpendicular string force); while type_string=2 means the conventional string (the perpendicular string force is not removed). If the problem failed to converge, one possibility is to first use type_string=2, and a relative large string constant ak. Then, after that is converged, one can do a second NEB calculation (copy MOVEMENT to atom2.config, and use itype_at2=2) using either itype_string=1, or a smaller ak while still use type_string=2. All these choices are to increase the flexibility in NEB calculations.

A normal NEB calculation should have the following steps:

1. using JOB=RELAX to calculate the two local minimum, their atomic positions atom1.config, atom2,config and energies E_1, E_2. Note, the atomic orders in atom1.config and atom2.config must be the same.

2. Use JOB=NEB, and write NEB_DETAIL as:

   IN.ATOM=atom1.config

   NEB_DETAIL=imth, nstep, force_tol, Nimage, ak, type_string, E0, En, itype_at2, atom2.config

   Make sure imth=3, nstep, force_tol can be similar as in RELAX_DETAIL. Nimage is the number of images between the first and last images. Typically Nimage can be 5 to 10. Choose a string constant. Typically, ak=0.1$\sim$1 ($eV/\mathring{A}^2$) sounds reasonable. If the relaxation is difficult (to reduce the atomic force, as reported in RELAXSTEPS), one can use type_string=2, otherwise, just use type_string=1. Place the E_0,E_N from previous RELAX calculations to the line NEB_DETAIL. Use itype_at2=1. Then do the JOB=NEB calculation.

   If type_string=2 was used, and ak is a bite large, one can do another calculation:

3. copy MOVEMENT into atom.continue (remove all the previous iterations). Replace "1, atom2.config" in NEB_DETAIL by "2, atom.continue". Now, either use type_string=1 (true NEB method), or still use type_string=2, but with a

much smaller ak. Do the calculation again. Check "NEB.BARRIER", make sure the images are roughly in equal distance, and the angle between $R(image+1) - R(image)$, and $R(image) - R(image-1)$, especially around the saddle point, is close to 0 degree ($cos(\theta)$ close to 1).

4.6 Molecular Dynamics(JOB=MD)

This is for ab initio molecular dynamics simulations. There are three methods:

1. md = 1, Verlet algorithm (for energy conserved true Newton dynamics);
2. md = 2, Nose-Hoover (stochastic methods for given temperature control simulation).
3. md = 3, Langevin dynamics (with a viscosity and a thermos bath, for given temperature control simulation);

We recommend to use either md=1, or md=3. This simulation will output MDSTEPS, and MOVEMENT. One can continue the simulation by copying MOVEMENT to atom.config (remove all the other iterations except the last one), then restart the calculation (only retype the running commands in the terminal). The PWmat will automatically detect whether there is a velocity section in atom.config. If yes, then it will use it as the initial velocity. If no, it will use temp1 to randomly generate an initial velocity. When using "JOB = MD", one has to include an etot.input line: "MD_DETAIL = md, mstep, dt, temp1, temp2".

"md" specifies the method. For md=1 (verlet), only when it is start from scratch, temp1 will be used. It is used to generate the initial random velocity according to this temperature. For md=1, temp2 is not used. For continued run, the initial velocity is read-in from the atom.config file, so temp1 is not used. For md=2, 3, if start from scratch (initial atom.config does not provide the velocity), then temp1 is used to generate the initial velocity. The md=2,3 dynamics will scale the temperature linearly with steps from temp1 to temp2.

4.7 Noncollinear magnetic moment

This is the calculation with SPIN=222. It includes the SPIN-ORBIT coupling. Note, one should usually specify the initial magnetic moment in atom.config, or if there is already an initial input from previous runs. The output charge density is no longer just a density, but a density matrix.

4.8 f-states

For some heavy elements, the f-states electrons play an important role in some poperties calculations. If one wants to consider these effects, just use the corresponding pseudopotentials which contain f electrons. The pseudopotential with f electron has the tag "lmax = 3" in the description part. One must check this on himself/herself.

4.9 optical spectrum

If one want to calculate optical absorption spectrum (using Fermi Gordon rule, no excition effects), the details can be found in Frequency dependent dielectric function calculations using RPA method).

4.10 Compatible runs with PWSCF

The PWmat can be run compatibly with the open source code PWSCF. Mostly, the PWmat can generate the wave function, charge density, and potential files, which can be read by the PWSCF program, or the PWSCF compatible programs (e.g., Wannier90 function generator, or GW calculations). These programs can be run on CPU. To run those programs, the user is responsible to prepare their control input files. One should also copy the corresponding PWSCF pseudopotential files from our library.

With the compatibility of PWmat and PWSCF, one can fully take the advantages of the wide functionalities of the PWSCF, while enjoy the speed of PWmat for some

of the key calculations. The available PWSCF capabilities include: Wannier function generation; linear-response phonon bandstructure calculation; linear-response TDDFT calculation; GW calculation. We refer the user to consult the open source PWSCF manual for how to calculate these properties.

To generate the PWSCF compatible wave function and potential files, set PWSCF_OUTPUT = T in etot.input.

4.10.1 Wannier function

Interface between PWmat and wannier90 are available now. To use the wannier90 program, please follow the next descriptions.

1. Run "scf"/"nonscf" calculations with **PWmat**. If setting **PWSCF_OUTPUT = T**, PWmat will output files with QE format into the directory **prefix.save**.

2. Run **wannier90.x** with **postproc_setup = .true.** to generate **seedname.nnkp**

3. Run **pw2wannier90.x** (from Quantum Espresso package "PP"). First it reads an input file e.g., **seedname.pw2wan**, which defines **prefix** and **outdir** for the underlying "scf" calculation, as well as the name of the file **seedname.nnkp**, and does a consistency check between the direct and reciprocal lattice vectors read from **seedname.nnkp** and those defined in the files specified by **prefix**. **pw2wannier90** generates **seedname.mmn**, **seedname.amn** and **seedname.eig**

4. Run **wannier90** with **postproc_setup = .false.** to disentangle bands (if required), localise MLWF, and use MLWF for plotting, bandstructures, Fermi surfaces etc.

Note, more infomation about using QE and wannier90 programs can be found on the websites (the websites are listed in page 231.)

Chapter 5

utility programs

Our utility programs, for which the source codes are available to the users, are an important part of our package. It provides many post-processing functions, and also tools for how to manipulate the output of PWmat. These tools include: how to calculate density of state and optical absorption spectrum; how to calculate the optical coupling constant (oscillator strength) between different states; how to carry out the non-adiabatic MD using the NAMD results; how to plot out the charge density; how to plot the wave functions. For more detailed information, please refer to PWmat website (http://www.pwmat.com/utility-download).

poscar2config.x
cell2config.x
xsf2config.x
pwscf2config.x
convert_from_config.x
config2poscar.x
atominfo.x
vwr2upf.x
uspp2upf.x
upf2upfSO.x
convert_rho.x
convert_realwg.x
convert_wg2rho.x

plot_band_structrure.x
plot_DOS.py
plot_DOS_interp.x
absorption_spec_K2step.x
RPA_absorb.x
plot_wg.x
plot_TDDFT.x
plot_fatband_structure.x
plot_electrical_conductivity.x
plot_tddft_absorp.x
split_kp.x
add_field.x
NAMD_psi.x
NAMD_Boltzman.x
ug_moment.x
vacuum.x
Gap_Read
nonradiative.x

5.1 Format conversion

5.1.1 poscar2config.x

convert VASP POSCAR file to atom.config

Syntax: poscar2config.x POSCAR

or **Syntax**: poscar2config.x < POSCAR

5.1.2 cell2config.x

convert CASTEP CELL format file to atom.config

Syntax: cell2config.x input.cell

or **Syntax**: cell2config.x < input.cell

5.1.3 xsf2config.x

convert xsf format file to atom.config

Syntax: xsf2config.x input.xsf

or **Syntax**: xsf2config.x < input.xsf

5.1.4 pwscf2config.x

convert pwscf input file to atom.config

Syntax: pwscf2config.x pwscf.in

or **Syntax**: pwscf2config.x < pwscf.in

5.1.5 convert_from_config.x

convert atom.config or final.config or MOVEMENT to .xsf and .xyz format file , so they can be used by different visualization tools for viewing.

Syntax: convert_from_config.x atom.config

or **Syntax**: convert_from_config.x < atom.config

5.1.6 config2poscar.x

convert atom.config or final.config to VASP POSCAR format

Syntax: config2poscar.x atom.config

or **Syntax**: config2poscar.x < atom.config

5.1.7 atominfo.x

show structure information in atom.config or final.config

Syntax: atominfo.x atom.config

or **Syntax**: atominfo.x < atom.config

5.1.8 vwr2upf.x

convert vwr format pseudopotential file to upf format pseudopotential file

Syntax: vwr2upf.x input.vwr

5.1.9 uspp2upf.x

convert uspp format pseudopotential file to upf format pseudopotential file

Syntax: uspp2upf.x input.uspp

5.1.10 upf2upfSO.x

convert spin-orbital pseudopotential file of pwscf to spin-orbital pseudopotential file of PWmat

Syntax: upf2upfSO.x < input

5.1.11 convert_rho.x

convert potential file OUT.VR or charge density file OUT.RHO or OUT.REAL.RHOWF_SP to RHO.xsf which can be visualized with VESTA

Syntax: convert_rho.x OUT.RHO **or** convert_rho.x OUT.VR

5.1.12 convert_realwg.x

convert the wave function file (OUT.REAL_RHOWF_SP) in real space. Like the convert_rho.x, it will output a file in XSF type which will be read by VESTA.

Syntax: convert_realwg.x OUT.REAL.RHOWF_SP01

5.1.13 convert_wg2rho.x

convert one or more wave function in OUT.WG file to charge density, stored in OUT.WG2RHO

Syntax: convert_wg2rho.x

5.2 Data visualization

5.2.1 plot_band_structure.x

Before run the plot_band_structure.x, please prepare REPORT and OUT.FERMI (this file is copied from the SCF calculation). Then it will generate the following files: bandstructure.eps, bandstructure.png, bandstructure.pdf and bandstructure_1.txt (the data file of bandstructure), which can be used to plot band with specified scale and regions. Note, for "spin=2", another data file bandstructure_2.txt will be generated.
Syntax: plot_band_structure.x

5.2.2 plot_DOS.py

After running PWmat JOB = DOS or program 'plot_DOS_interp.x', one can use 'plot_DOS.py' to get pictures. Besides, if "OUT.FERMI" file is presented, it will also output DOS with Fermi energy set to zero in file 'DOS.*_ShiftFermi'.
Syntax: plot_DOS.py

5.2.3 plot_DOS_interp.x

If one wishes to change Gaussian broadening or number of energy grid points of DOS, or more importantly, plot the angular momentum projected DOS or the selected atoms partial DOS, one can use this plot_DOS_interp.x.

To do that, one has to provide an input file: DOS.input, with the following contents:

```
0
1
0.05 4000
8 8 8
0
```

The **1st** line: if setting 0, it means plotting DOS for all atoms; if setting 1, for partial atoms. Note, when doing for partial atoms, one need to add the 8th column to set the weights for all the atoms.

The **2nd** line: if setting 1, it means using interpolation for DOS plotting (must use DOS_DETAIL with IDOS_interp=1); if setting 0, it keeps the old method, not doing interpolation, just using a Gaussian broadening.

The **3rd** line: energy smearing, in eV; number of energy grid points, default is 4000.

The **4th** line: NM_1, NM_2, NM_3:, the interpolation grid, with in each grid in NQ_1, NQ_2, NQ_3. The larger of NM_1, NM_2, NM_3, the smoother of DOS, but also slower to run.

The **5th** line: only useful for TDDFT DOS plotting. If setting 1, it means reading "IN.OCC_ADIA" (occupation file from TDDOS directory, see section B.11.2) and output DOS/PDOS of occupied states; if setting 0, do nothing and output DOS/PDOS of all the states. Default is 0.

The format of structure file when doing partial DOS:

```
30 0.952534560 0.363594470 0.382027650 1 1 1 w1
30 0.540553000 0.850230410 0.966359450 1 1 1 w2
...
16 0.242857140 0.140553000 0.684331800 1 1 1 w3
```

Here w1, w2, w3 are the weight for this atom in the partial DOS.

Note, the plot_DOS_interp.x will rewrite the file “DOS.totalspin”.

Syntax: plot_DOS_interp.x

5.2.4 absorption_spec_K2step.x

This program is used to plot optical absorption spectrum, it reads PWmat output files: MDDIPOLE.KSPACE, OUT.TDDFT_TIME. In addition, one has to provide a input file: absorp_K.input, with the following contents:

```
5.0                    !broading fact
200 0.001              !MD_step1, dt1(fs)
1000 0.01              !MD_step2, dt2(fs)
1.8, 20                !w_cut_min, w_cut_max (eV)
8                      !N_electron
```

```
0.002                   !E_field
2.7155 2.7155 0.0       !AL(:,1)
2.7155 0.0    2.7155    !AL(:,2)
0.0    2.7155 2.7155    !AL(:,3)
```

For more detail information about absorption_spec_K2step.x and absorp_K.input, please refer to "Frequency dependent dielectric function calculations for bulk systems using rt-TDDFT method" on PWmat website.

Syntax: absorption_spec_K2step.x

5.2.5 RPA_absorb.x

This program is used to plot optical absorption spectrum with RPA method, one could use LDA, PBE, or HSE functions for this calculation. Please refer to Frequency dependent dielectric function calculations using RPA method) for more details.

5.2.6 plot_wg.x

This program is used to plot OUT.WG, and the output file is written in PSI.xsf, which can be visualized by VESTA.

Syntax: plot_wg.x

5.2.7 plot_TDDFT.x

This program is used to plot RT-TDDFT output, it reads PWmat output file: OUT.TDDFT1. Here we give some descriptions of the code about how to read the "OUT.TDDFT1".

```
open(23,file="OUT.TDDFT1",form="unformatted")
write(23) islda,nkpt,mx,nref_tot_8,natom,nnodes
write(23) mst_win0,mst_win,mst_td,mst
write(23) isTddftOut1_Cmat,dtTddftout1
do iislda=1,islda
    do kpt=1,nkpt
```

```
        write(23)
        dt_time,iislda,kpt,weighkpt_2(kpt),akx_2(kpt),aky_2(kpt),akz_2(kpt)
        write(23) E_st(1:mst,kpt,iislda)*Hartree_ev
        write(23) dos_adiabatic(1:mst,kpt,iislda)
        if(isTddftOut1_Cmat) then
            write(23) Cmat(1:mst,1:mst_td,kpt,iislda)
        endif
    enddo
enddo
```

Syntax: plot_TDDFT.x

5.2.8 plot_fatband_structure.x

Before run the plot_fatband_structure.x, please prepare REPORT and OUT.FERMI (this file is copied from the SCF calculation). Then it will generate the output file: fatbandstructure_1.txt (the data file of bandstructure), which can be used to plot band with specified scale and regions. Note, for "spin=2", another data file fatbandstructure_2.txt will be generated. For more detail information, please refer to "PDOS & fatband structure" on PWmat website.

Syntax: plot_fatband_structure.x

5.2.9 plot_electrical_conductivity.x

This program is used to calculate electrical conductivity using Kubo-Greenwood electrical conductivity formulation. In addition, one has to provide a input file: DOS.input, with the following contents:

```
1
0.05 7.66881742637123 0.2
3 3 3
OUT.SYMM atom.config
1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0
```

The 1st line: 1, use interpolation; 0, no interpolation. The 2nd line: Gaussian broadening, unit eV; Fermi energy, unit eV (comes from OUT.FERMI); kBT in Fermi-Dirac formula, unit eV, used to calculate the electron occupations. The 3th line: The interpolation grid, with in each grid in NQ1, NQ2, NQ3. The 4th line: Name of symmetry operation file; Name of structure file. The 5th-7th line: polar(:,1), polar(:,2), polar(:,3). The polarization usedto calculate the results (e.g., for circular polarization). For x, y, z, just use the example one. Need to be normalized to one.

The **1st** line: if setting 1, it means using interpolation; if setting 0 , no interpolation.

The **2nd** line: Gaussian broadening, unit eV; Fermi energy (comes from OUT.FERMI), unit eV; equivalent temperature $k_b T$ in Fermi-Dirac formula to calculate the electron occupations, unit eV.

The **3rd** line: The interpolation grid, with in each grid in NQ1, NQ2, NQ3.

The **4th** line: Name of symmetry operation file; Name of structure file.

The **5-7th** line: polar(:,1), polar(:,2), polar(:,3). The polarization used to calculate the results (e.g., for circular polarization). For x, y, z, just use the example one. Need to be normalized to one.

For more detail information about plot_electrical_conductivity.x, please refer to Electrical conductivity on PWmat website.

Syntax: plot_electrical_conductivity.x

5.2.10 plot_tddft_absorp.x

This program is used to calculate RT-TDDFT optical absorption spectrum for isolated systems, such as clusters and molecules, please refer to Absorption Spectrum for non-periodic systems on PWmat website.

5.2.11 split_kp.x

To generate the k-points between the specific points of the Brillouin zone, one can use “split_kp.x”. One should prepare an input file for “split_kp.x”. Note the file name

can be arbitrary except for "IN.KPT", because "split_kp.x" will output the k-points file which PWmat will use for bandstructure calculation. The input file naming "gen.kpt":

```
BAND    # COMMENT line
10        # number of k-points between G and X
0.0 0.0 0.0 G        #reciprocal coordinates, label 'G' for Gamma point
0.5 0.0 0.0 X
15
0.5 0.0 0.0 X
0.5 0.5 0.5 R
10
0.5 0.5 0.5 R
0.5 0.5 0.0 M
```

After running "split_kp.x gen.kpt", "split_kp.x" will generate 10 k-points between "G" and "X", 15 k-points between "X" and "R" ..., and write all the k-points in "IN.KPT". The coordinates of the k-points should be fractional coordinates in reciprocal lattice. Note the words start with # is the comment line, not essential.

After running "split_kp.x gen.kpt", "split_kp.x" will also generate high-symmetry points information in "HIGH_SYMMETRY_POINT":

```
Label       Index       Coordinate
G               1          0.00000
X              12          0.53022
R              28          1.14246
M              39          1.67268
```

"Index" indicates which k-points are high-symmetry points in "IN.KPT", "Coordinate" is the abscissa value of high-symmetry point in bandstructure results. It could be useful for bandstructure post-processing.

5.3 post processing

5.3.1 add_field.x

This program is used to add external electric field, and it will generate IN.VR.EXT file. You can copy IN.VR.EXT to IN.VEXT, which can be read by PWmat. One should prepare two input files: IN.VR and gen.vext. IN.VR is a binary file, which can be obtained by PWmat. Here is a way: set "OUT.REAL.RHOWF_SP=2" in etot.input, then PWmat will generate output file: OUT.REAL.RHOWF_SP. The data structure of OUT.REAL.RHOWF_SP is similar to OUT.VR and OUT.RHO. One can copy OUT.REAL.RHOWF_SP to IN.VR. The other input file is gen.vext, with the following contents:

Parameters	Illustration
VR_CENTER	VR_CNETER = a1 a2 a3, Center of external electric field, default values: 0.5 0.5 0.5 (fractional coordinates in x, y, z drection)
VR_TYPE	VR_TYPE supports three type of external electric field, please refer to VR_DETAIL
VR_DETAIL	VR_TYPE = 1, VR_DETAIL= a4 a5 a6 (the units of a3, a4, a5 are $Hartree/Bohr$), Vext(r)=(x-a1)*a4+(y-a2)*a5+(z-a3)*a6 ; VR_TYPE = 2, VR_DETAIL= a4 a5 a6 a7 a8 a9 (the units of a4, a8 are $Hartree/Bohr$; the units of a5, a6, a7, a9 are $Hartree/Bohr^2$), Vext(r)=(x-a1)*a4+$(x-a1)^2$*a5+(y-a2)*a6+$(y-a2)^2$*a7+(z-a3)*a8+$(z-a3)^2$*a9; VR_TYPE = 3, VR_DETAIL= a4 a5 (the unit of a4 is Hartree, the unit of a5 is Bohr), Vext(r)=a4*exp-$[(x-a1)^2+(y-a2)^2+(z-a3)^2]/a5^2$
ADD_VR	ADD_VR = T, add the external electric file to the original potential file (for example, IN.VR), and generate EXT file; ADD_VR = F, only import the external electric field to EXT file.

For more detail information, please refer to utility on PWmat website.

Syntax: add_field.x IN.VR

5.3.2 NAMD_psi.x

This program is used to plot NAMD output wave functions.

Syntax: NAMD_psi.x

5.3.3 NAMD_Boltzman.x

This program is used to calculate boltzmann constant for NAMD calculation.

WARNING: We'd highly recommend you use "namd_dm.x" to simulate Boltzman NAMD. Please refer to Boltzman-NAMD for more information. In this module, we developed a new NAMD simulation method by modifying the conventional density matrix, it can incorporate the detailed balance and decoherence.

Syntax: NAMD_Boltzman.x

5.3.4 ug_moment.x

This program is used to calculate angular momentum, it reads PWmat outputs: OUT.GKK and IN.WG, and generate moment.matrix file. moment.matrix is a binary file, which save px, py, pz. You can read moment.matrix by the following code:

```
program read_moment_matrix
    !
    integer :: nkpt, ispin, mx
    complex(kind=8), allocatable, dimension(:,:) :: cdot_gkx, cdot_gky, cdot_gkz
    open (12, file = 'moment.matrix', form = 'unformatted')
    read (12) nkpt, ispin, mx
    allocate (cdot_gkx(mx,mx), cdot_gky(mx,mx), cdot_gkz(mx,mx))
    read (12) cdot_gkx
    read (12) cdot_gky
    read (12) cdot_gkz
    !
end program
```

Syntax: ug_moment.x

5.3.5 vacuum.x

This program is used to calculate vacum level, please refer to utility on PWmat website.

Syntax: vacuum.x

5.3.6 Gap_Read

This program is used to read band gap from REPORT file.

Syntax: Gap_Read

5.3.7 nonradiative.x

This program is used to calculate defect nonradiative decay, please refer to Defect Nonradiative Decay on PWmat website.

Syntax: nonradiative.x

Appendix A

Work Flow and Websites

A.1 Work Flow

A.1.1 Pre-process

1. prepare xsf format file
2. xsf2config.x
3. prepare etot.input

A.1.2 Run PWmat

1. MD
2. RELAX
3. SCF
4. NONSCF
5. DOS
6. TDDFT
7. NAMD

A.1.3 Post-process

1. plot_band.x
2. plot_dos.x
3. convert_rho.x
4. convert_from_config.x
5. Post-pwscf calculation

A.2 Useful Websites

1. PWmat: http://www.pwmat.com/
2. Quantum Espresso: http://www.quantum-espresso.org/
3. Wannier90: http://www.wannier.org/
4. American Mineralogist Crystal Structure Database: http://rruff.geo.arizona.edu/AMS/amcsd.php
5. NIST Chemistry WebBook: http://webbook.nist.gov/chemistry/
6. Nvidia Cuda Zone: https://developer.nvidia.cn/
7. XCrySDen: http://www.xcrysden.org/
8. VESTA: http://www.jp-minerals.org/vesta/en/
9. VMD: http://www.ks.uiuc.edu/Research/vmd/

Appendix B

TDDFT and NAMD Manual and Examples

B.1 JOB=TDDFT

carry out real time TDDFT (rt-TDDFT) simulations. The It supports,

1. xcfunctional=lda/pbe
2. norm-conserving pseudopotential

Note, JOB=TDDFT requires TDDFT_DETAIL, MD_DETAIL.

B.2 TDDFT_DETAIL

TDDFT_DETAIL = m_1 m_2 **mstate**

Default: TDDFT_DETAIL = 1,NUM_BAND,NUM_BAND.

This is mostly required for JOB=TDDFT. We strongly encourage to include this line. In the TDDFT simulation, we will calculate time dependent state $\psi_j(t)$, j=1,mstate. Note, mstate is different from NUM_BAND. The NUM_BAND is the number of calculated adiabatic state $\phi_i(t)$. For the first $m_1 - 1$ $\psi_j(t)$ state (j=1,$m_1 - 1$), $\psi_j(t) = \phi_j(t)$, e.g., they are just the adiabatic states (just like in Born-Oppenheimer

MD). However, for the rest, mstate-m_1+1 $\psi_j(t)$ states, they are expanded by the adiabatic state window [m_1,m_2]. The occupation number of $\psi_j(t)$ are fixed. They are either given by the first SCF iteration Fermi-Dirac distribution, or given by input. Thus, in total, there will be m_2 state to be used, with the first $m_1 - 1$ to be the same as $\psi_j(t)$ to be occupied, and the rest of the state, the [m_1,m_2] as the window to expand the rest of wave function $\psi_j(t)$.

Expand $\psi_j(t)$ in terms of the adiabatic eigenstates $\phi_i(t)$

$$\psi_j(t) = \sum_i C_{ji}\phi_i(t) \tag{B.1}$$

Define the Adiabatic window $[m1, m2]$:

$$\psi_j(t) = \phi_j(t), j = 1, m1 - 1 \tag{B.2}$$

$$\psi_j(t) = \sum_i C_{ji}(t)\phi_i(t), j = m1, mstate; i = m1, m2 \tag{B.3}$$

$[m1, m2]$	Adiabatic window $\phi_{i,i=m1,m2}$. The $[1, m1 - 1]$ will always be occupied by the first $\psi_{j,j=1,m1-1}$ states. $m2 \in [m1, NUM\_BAND]$, usually $m2$ is smaller than NUM_BAND by a few states, cause the last few states maybe not converge well.
$[1, mstate]$	Wavefunction index. $\psi_{j,j=1,mstate}$. $mstate \in [m1, m2]$

B.2.1 example B.2.1: default settings

atom.config:

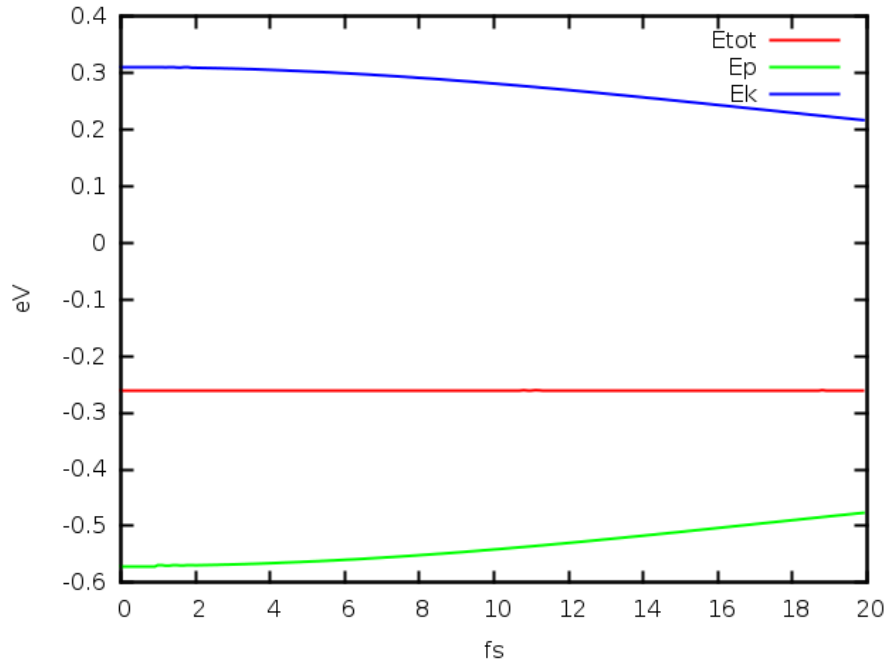
```
8
Lattice vector
5.65      0.00      0.00
0.00      5.65      0.00
```

```
0.00      0.00      5.65
Position, move_x, move_y, move_z
31    0.0010   0.0000   0.0000  1  1  1   1.0
31    0.0000   0.5010   0.5020  1  1  1   1.0
31    0.5000   0.0000   0.5000  1  1  1   1.0
31    0.5000   0.5000   0.0000  1  1  1   1.0
33    0.2500   0.2500   0.2500  1  1  1   0.0
33   -0.2500  -0.2500   0.2500  1  1  1   0.0
33   -0.2500   0.2500  -0.2500  1  1  1   0.0
33    0.2500  -0.2500  -0.2500  1  1  1   0.0
```

etot.input:

```
1           1
IN.ATOM    = atom.config
JOB        = TDDFT
MD_DETAIL = 1, 20, 0.1, 300,300
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
```

MDSTEPS–Etot,Ep,Ek plot:



B.2.2 example B.2.2: adiabatic window

from the output file OUT.OCC of example1,

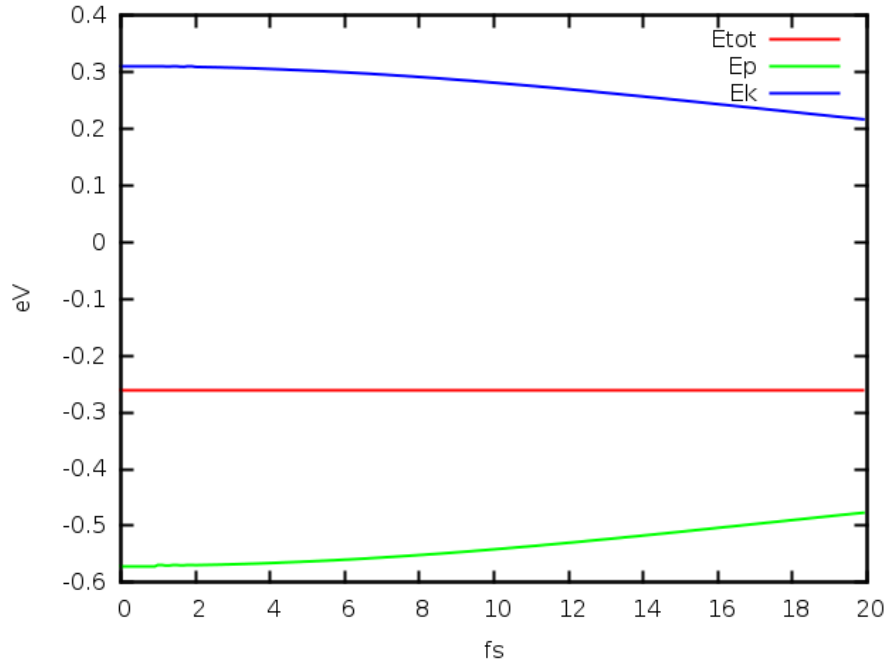
```
KPOINTS   1:    0.0000    0.0000    0.0000
NO.   ENERGY(eV) OCCUPATION
 1    -10.7422   2.00000
 2     -8.3784   2.00000
 3     -8.2272   2.00000
 4     -8.1217   2.00000
 5     -5.0553   2.00000
 6     -5.0473   2.00000
 7     -5.0042   2.00000
 8     -0.8431   2.00000
 9     -0.8061   2.00000
10     -0.7368   2.00000
11     -0.7011   2.00000
12     -0.6666   2.00000
13     -0.6165   2.00000
14      1.8319   1.99983
15      2.0285   1.99966
16      2.1978   1.99956
17      2.4204   0.00095
18      3.0161   0.00000
19      3.0985   0.00000
20      3.2698   0.00000
21      3.3809   0.00000
22      3.4191   0.00000
23      3.5045   0.00000
24      5.4035   0.00000
25      5.5223   0.00000
26      5.6578   0.00000
```

we know that the $[1, 16]$ states are occupied, and the total num of band is 26. Then we can set the TDDFT_DETAIL=m1 m2 mstate,$m1 \in [1, 16]$, $m2 \in [m1, 26]$, $mstate \in [m1, m2]$

```
1           1
IN.ATOM   = atom.config
```

```
JOB        = TDDFT
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_DETAIL=6,26,23
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
convergence = difficult
```

MDSTEPS–Etot,Ep,Ek plot:



B.3 OUT.TDDFT

OUT.TDDFT = T_1, T_2, n_1, T_3, n_2

DEFAULT:= F F 1.0 F 1.0

The output files can be used to restart TDDFT and show the process of TDDFT.

$T1, T2, n1$	$T1 = T/F$	eigen energy, $occ(i)$ per $n1$ fs. The output will be in file OUT.TDDFT1. One can use plot_TDDFT.f90(ref. util) to read and output OUT.TDDFT1.
	$T2 = T/F$	C_{ij} per n1 fs
$T3, n2$	$T3 = T/F$	output all the wavefunctions and charge densities per $n2$ fs for restart. The output will be in file OUT.TDDFT and directory TDDOS/. This can be very expensive, so use large $n2$.

B.3.1 example B.3.1: output files

```
1           1
convergence=difficult
IN.ATOM    = atom.config
JOB        = TDDFT
MD_DETAIL = 1, 200, 0.1, 300,300
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
OUT.TDDFT = T T 1.0 T  5.0
```

>ls

./	OUT.TDDFT1	update per 1.0 fs used by plot_TDDFT.f90
./	OUT.TDDFT	update per 5.0 fs used by restarting TDDFT
TDDOS/	OUT.WG.* OUT.EIGEN.* OUT.RHO.*	update per 5.0 fs used by ploting DOS

B.4 TDDFT_SPACE

TDDFT_SPACE = itype_space, N, a(1), ..., a(N)

DEFAULT:= 0 ...

In the TDDFT calculation, we often need to have external potential to perturb the system (e.g., for optical absorption, or plasmon excitation). The descriptions of these time dependent, spatial dependent external potential are controlled by the tags: TDDFT_SPACE, TDDFT_TIME, or input file IN.VEXT_TDDFT, IN.TDDFT_TIME.

The TDDFT_SPACE controls the real space Vext_tddft(r). Vext_tddft(r) refers to the external potential in real space for tddft calculation.

itype_space	
0	No external input term.
1	Read vext_tddft from file IN.VEXT_TDDFT(all capital, same format as in IN.VEXT).
2	$Vext\_tddft(r) = (x - a(1))a(4) + (x - a(1))^2 a(5) + (y - a(2))a(6) + (y - a(2))^2 a(7) + (z - a(3))a(8) + (z - a(3))^2 a(9)$, a(1),a(2),a(3) in fractional coordinates, a(4)-a(8) in unit of Hartree/Bohr. output file OUT.VEXT_TDDFT.
3	$Vext\_tddft(r) = a(4)e^{-[(x-a(1))^2+(y-a(2))^2+(z-a(3))^2]/a(5)^2}$, a(1),a(2),a(3) in fractional coordinates, a(4) in unit of Hartree, a(5) in unit of Bohr. output file OUT.VEXT_TDDFT.
-1	Not use real space format, but use G-space,it wil use IN.A_FIELD

The 'IN.VEXT_TDDFT' file can be copied from other TDDFT calculation output file 'OUT.VEXT_TDDFT', or generated by utility programs add_field.x.

B.4.1 example B.4.1: itype_space=1 or 2

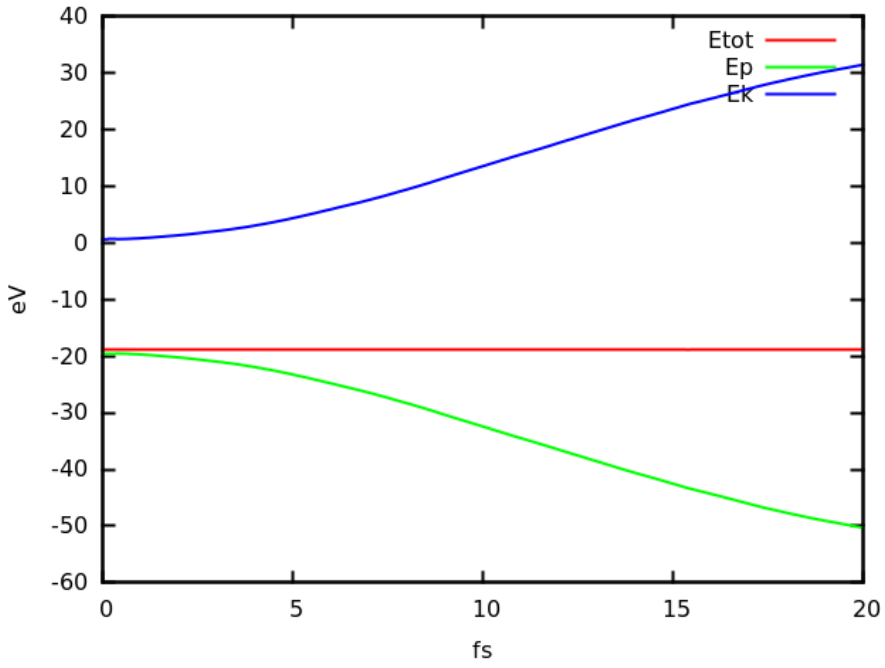
First we can get IN.VEXT_TDDFT by set itype_space=2.

```
1            1
IN.ATOM    = atom.config
convergence=difficult
JOB        = tddft
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_SPACE = 2, 9,0.5,0.5,0.5,0.0,0.01,0.0,-0.02,0.0,0.01
```

> cp OUT.VEXT_TDDFT IN.VEXT_TDDFT

```
1            1
IN.ATOM    = atom.config
precision = double
convergence=difficult
JOB        = tddft
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_SPACE = 1
```

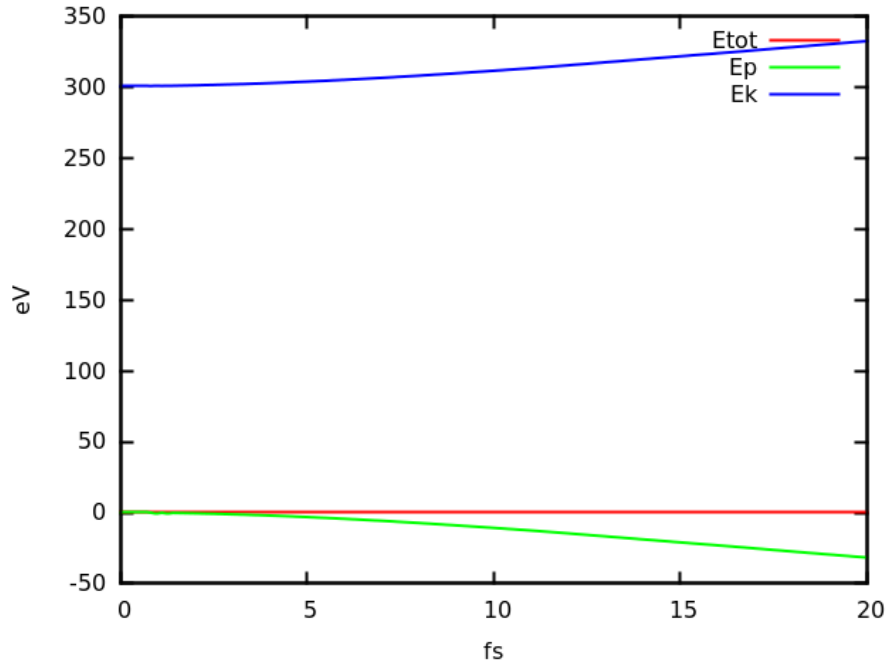
MDSTEPS–Etot,Ep,Ek plot:



B.4.2 example B.4.2: itype_space=3

```
1           1
IN.ATOM    = atom.config
convergence=difficult
JOB        = tddft
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_SPACE = 3, 5, 0.5,0.5,0.5, 1.0, 5
```

MDSTEPS–Etot,Ep,Ek plot:



B.5 IN.A_FIELD

IN.A_FIELD= T / F, a_field1, a_field2, a_field3

Default:

IN.A_FIELD= F 0.0 0.0 0.0

Note: for PWmat version later than 20200824, it has a new format:

IN.A_FIELD_LIST1= a_field1, a_field2, a_field3 IN.TDDFT_TIME1

IN.A_FIELD_LIST2= a_field1, a_field2, a_field3 IN.TDDFT_TIME2

...

the maximum support is 20 rows. This can be used to add a circularly polarized light. IN.TDDFT_TIME1,IN.TDDFT_TIME2... is the name of TDDFT_TIME file. You need to prepare same number of TDDFT_TIME files, it has the same format as IN.TDDFT_TIME,

```
0 ftddft(0)
1 ftddft(1)
...
N ftddft(N)
```

This controls the G-space external potential input for tddft calculation. (only used when TDDFT_SPACE=-1,...)

The tddft hamiltonian,

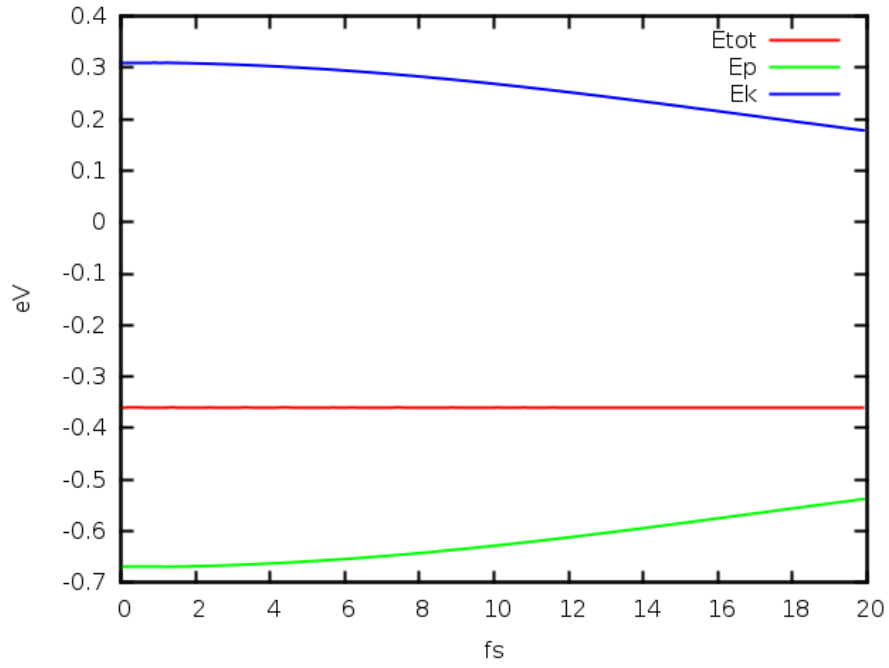
$$H = 1/2(-i\nabla_x + a\_field1)^2 + 1/2(-i\nabla_y + a\_field2)^2 + 1/2(-i\nabla_z + a\_field3)^2 \tag{B.4}$$

The values of $a\_field1, 2, 3$ are all in units of 1/Bohr.

B.5.1 example B.5.1: itype_space=-1

```
1           1
IN.ATOM   = atom.config
precision = double
convergence=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_SPACE = -1
IN.A_FIELD = T  0.1 0.2 0.3
```

MDSTEPS–Etot,Ep,Ek plot:



B.6 TDDFT_TIME

TDDFT_TIME = itype_time, N, b(1), ..., b(N)

DEFAULT:= 0 ...

This is used to control the time dimension of the external function fTDDFT(i).

itype_time	
0	$ftddft(t) = 1.0$
1	read in $ftddft(i)$ from IN.TDDFT_TIME
2	$ftddft(t) = b(1)e^{-(t-b(2))^2/b(3)^2)}\sin(b(4)t + b(5))$. $b(2)$,$b(3)$ in unit of fs; $b(4)$ in unit of rad/fs, $b(5)$ in unit of rad; $b(1)$ no unit. output file OUT.TDDFT_TIME
22	$ftddft(t) = \int_0^t [b(1)e^{-(t-b(2))^2/b(3)^2)}\sin(b(4)t + b(5))]dt$. $b(2)$,$b(3)$ in unit of fs; $b(4)$ in unit of rad/fs, $b(5)$ in unit of rad; $b(1)$ no unit. output file OUT.TDDFT_TIME

File IN.TDDFT_TIME format,

```
0 ftddft(0)
1 ftddft(1)
...
N ftddft(N)
```

For TDDFT Hamiltonian, we have,

itype_space	
$\neq -1$	$H(t) = H_0 + Vext\_tddft(r)ftddft(t)$
-1	$H(t) = 1/2(-i\nabla_x + A_x * ftddft(t))^2 + 1/2(-i\nabla_y + A_y * ftddft(t))^2 + 1/2(-i\nabla_z + A_z * ftddft(t))^2$

B.6.1 example B.6.1: itype_space=2,itype_time=1 or 2

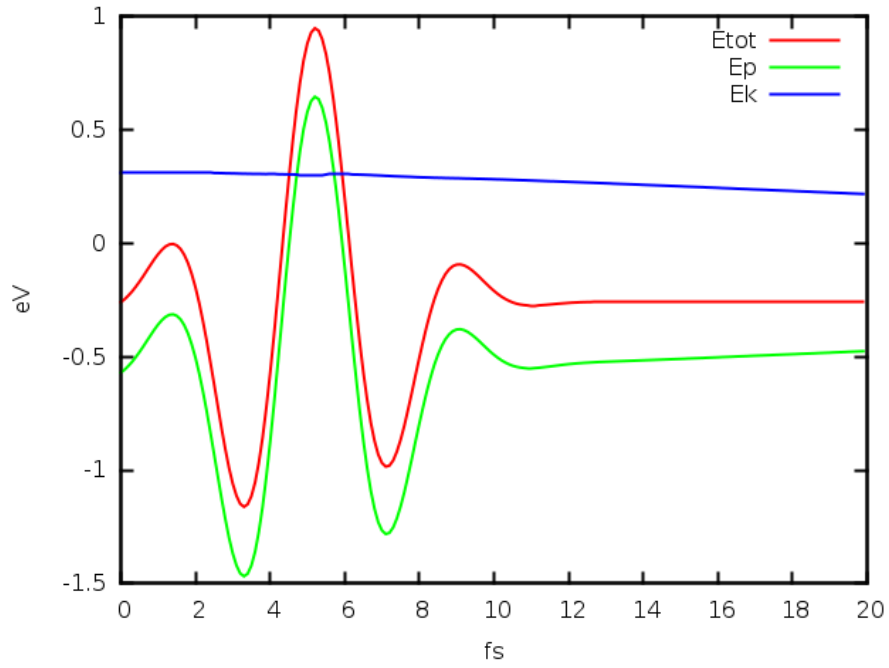
First set itype_time=2,we can get OUT.TDDFT_TIME,

```
1           1
IN.ATOM   = atom.config
convergence=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_SPACE = 2, 9,0.5,0.5,0.5,0.002,0,0.,0, 0., 0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
```

> cp OUT.TDDFT_TIME IN.TDDFT_TIME

```
1           1
IN.ATOM   = atom.config
convergence=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_SPACE = 2, 9,0.5,0.5,0.5,0.002,0,0.,0, 0., 0
TDDFT_TIME = 1
```

MDSTEPS–Etot,Ep,Ek plot:



B.7 IN.OCC/IN.OCC_2

IN.OCC=T/F

The files are used to set the occupation of adiabatic eigenstates when FERMI-DIRAC=0 in line "SCF_ITER0_X". This initial adiabatic states ($\phi_i(t=0)$) are used as the initial time dependent state ($\psi_j(t=0) = \psi_j(t=0)$ for all j=1,mstate.

spin=1, use IN.OCC. spin=2, use both IN.OCC and IN.OCC_2.

Files IN.OCC, IN.OCC_2 format,

```
1.0 1.0 1.0 0.6 0.0 0.0 0.0 ...
#occupations for k-point1 (this line should have NUM_BAND number)
1.0 1.0 1.0 0.6 0.0 0.0 0.0 ...
#occupations for k-point2 (this line should have NUM_BAND number)
```

or

```
3*1.0 0.6 0.0 0.0 0.0 ...
3*1.0 0.6 0.0 0.0 0.0 ...
```

B.7.1 example B.7.1: IN.OCC

```
1           1
IN.ATOM   = atom.config
convergece=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_DETAIL = 1,26,17
TDDFT_SPACE = 1,9,0.5,0.5,0.5, 0.002,0,0.,0, 0., 0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
IN.OCC  = T
```

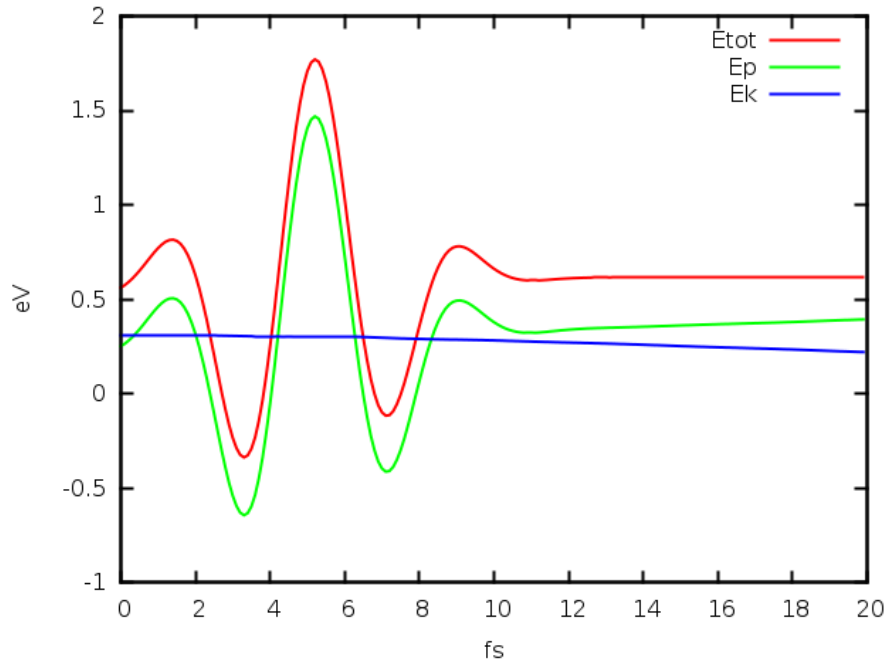
IN.OCC:

```
1 1 1 1 1 1 1 1 1 1
1 1 1 0.66666666666666
0.66666666666666 0.66666666666666 1
0 0 0 0 0 0 0 0 0 0
```

or

```
13*1 3*0.66666666666666 1 9*0
```

MDSTEPS–Etot,Ep,Ek plot:



B.8 IN.CC/IN.CC_2

IN.CC=T/F

The files are used to initialize the C_{ij} for TDDFT, which is used as $\psi_j(t) = \sum_i C_{ji}(t)\phi_i(t)$.

spin=1, use IN.CC. spin=2, use both IN.CC and IN.CC_2.

Files IN.CC, IN.CC_2 format,

```
1  1  1.0
1  2  1.0
1  3  1.0
2  4  0.8  5 0.2
1  5  1.0
....
```

Line j specify the $\psi_j, j = 1, mstate$. Define pair (i,CC), i is the index of adiabatic states, CC is the value of C_{ji}. Each line corresponds to one j state: ψ_j in the consecutive order of j. The first column number n_pair specifies the number of pairs described in

this line. If m, one index of adiabatic states, is not specified within a given line, then $C_{jm} = 0$.

B.8.1 example B.8.1: IN.CC

```
1           1
IN.ATOM   = atom.config
convergece=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 200, 0.1, 300,300
TDDFT_DETAIL = 1, 26, 26
TDDFT_SPACE = 1,9,0.5,0.5,0.5,0.002,0,0.,0, 0., 0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
NUM_BAND = 30
IN.CC=T
```

IN.OCC:

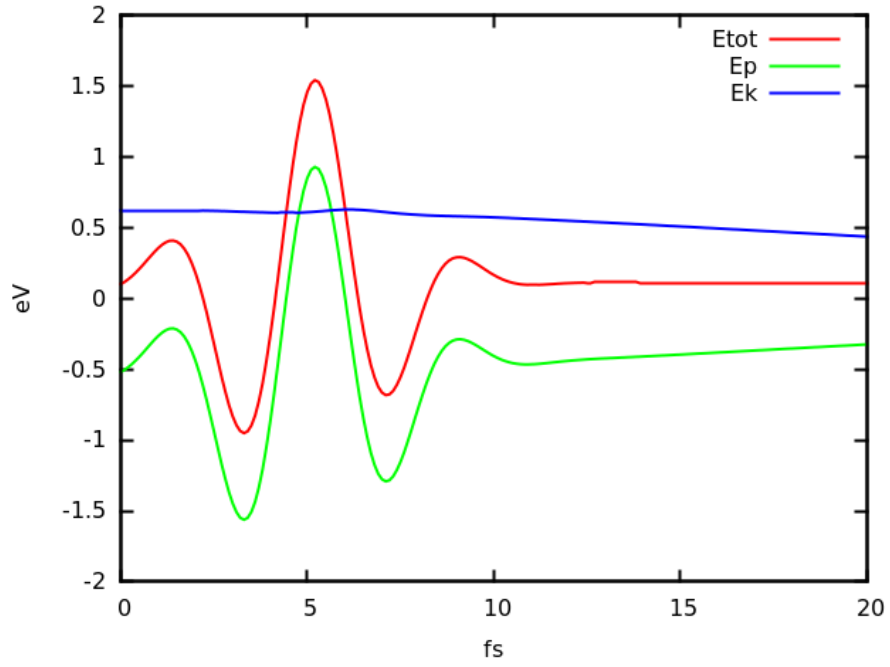
```
1 1 1 1 1 1 1 1 1 1
1 1 1 0.66666666666666
0.66666666666666 0.66666666666666 1
0 0 0 0 0 0 0 0 0 0
```

IN.CC:

```
1  1 1.0
1  2 1.0
1  3 1.0
1  4 1.0
1  5 1.0
1  6 1.0
1  7 1.0
1  8 1.0
1  9 1.0
1  10 1.0
1  11 1.0
```

```
1   12 1.0
1   13 1.0
1   14 1.0
2   15 0.8 16 0.2
1   16 1.0
1   17 1.0
1   18 1.0
1   19 1.0
1   20 1.0
1   21 1.0
1   22 1.0
1   23 1.0
1   24 1.0
1   25 1.0
1   26 1.0
```

MDSTEPS–Etot,Ep,Ek plot:



B.9 MD_DETAIL = MD, MSTEP, DT, TEMP1, TEMP2

Note: this is a required line for JOB=MD, JOB=TDDFT and JOB=NAMD. (ref. PWmat manual 2.1.6.)

B.10 RESTART

Needed settings:

```
MD_DETAIL=11,...
```

Needed files:

```
OUT.TDDFT
TDDOS/*
```

B.10.1 example B.10.1: RESTART

If you want to restart TDDFT from a certain time, the first thing is to set 'OUT.TDDFT' at the beginning. Here is a TDDFT example which will be terminated at 1 fs. By setting 'OUT.TDDFT = T T 1.0 T 0.1', it will output the wavefunctions, charge density and OUT.TDDFT files into TDDOS directory every 0.1 fs for restart. Note, such small output interval can be very expensive, so use you should use large time interval in practice.

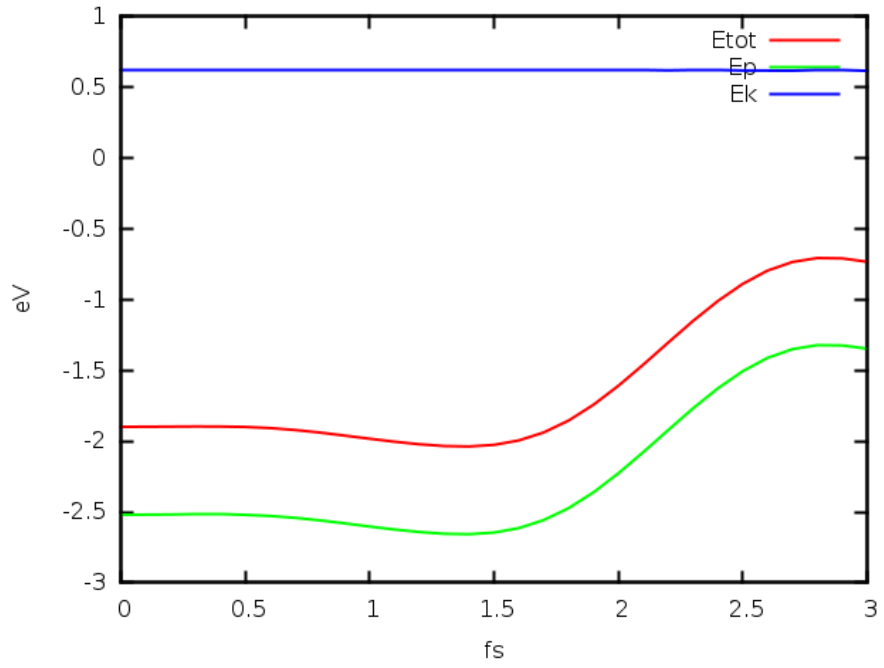
```
1           1
IN.ATOM   = atom.config
precision = double
convergece=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
```

```
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 10, 0.1, 300,300
TDDFT_SPACE = -1
IN.A_FIELD = T  0.1 0.0 0.0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
OUT.TDDFT = T T 1.0 T 0.1
```

When the previous calculation is completed without any error, if we change the 'MD_DETAIL' tag like below then run PWmat, the program will read the last output files and continue to run 2 fs. So the total simulation time will be 3 fs.

```
1           1
IN.ATOM   = atom.config
precision = double
convergece=difficult
JOB       = TDDFT
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
MD_DETAIL = 11, 20, 0.1, 300,300
TDDFT_SPACE = -1
IN.A_FIELD = T  0.1 0.0 0.0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
OUT.TDDFT = T T 1.0 T 0.1
```

MDSTEPS–Etot,Ep,Ek plot:



But, if we want to restart the job from 0.5 fs, except changing the 'MD_DETAIL' tag, we have to copy the OUT.TDDFT and structure file of 0.5 fs. So the total simulation time will be 2.5 fs.

> cp TDDOS/OUT.TDDFT.0.500000E+00 OUT.TDDFT

> cp TDDOS/restart.config.0.500000E+00 TDDOS/restart.config

B.11 SHOW_RESULTS

B.11.1 example B.11.1: plot_tddft

The file plot_tddft.f90 is in util/. One can check the Module TDDFT Carrier Cooling for details.(http://www.pwmat.com/module-download)

```
1            1
IN.ATOM    = atom.config
convergece=difficult
JOB        = TDDFT
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
```

```
MD_DETAIL = 1, 100, 0.1, 300,300
TDDFT_SPACE = -1
IN.A_FIELD = T  0.1 0.0 0.0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
OUT.TDDFT = T T 0.1 T 1.0
```

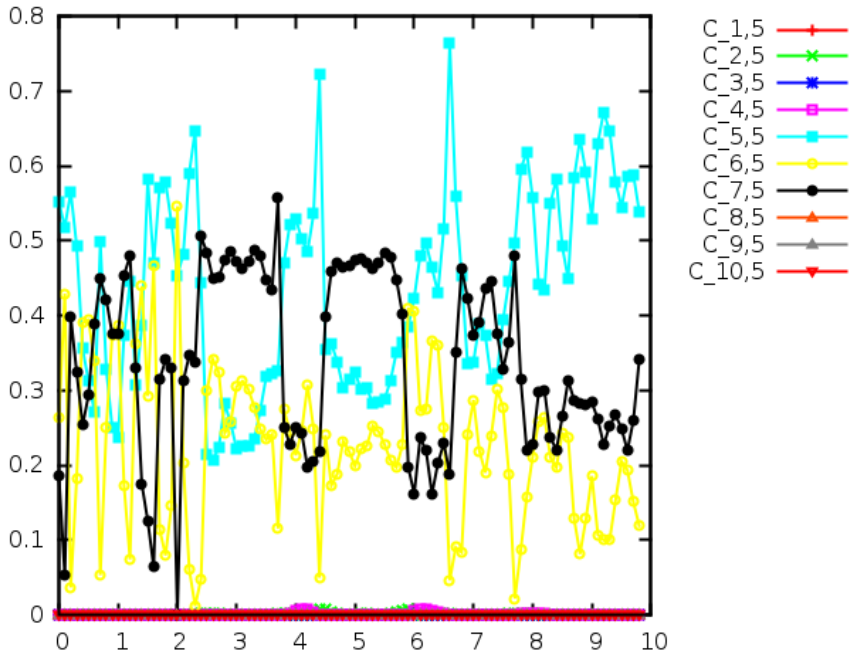
plot_TDDFT.f90 & OUT.TDDFT1:

TDDFT/example B.11.1:ifort plot_TDDFT.f90 -o plot_TDDFT.x

TDDFT/example B.11.1:./plot_TDDFT.x

```
there is Cmat, plot E,DOS (1) or Cmat(2)
2
there are nkpt,islda spin           1           1
input ikpt,iislda to plot
1 1
there are           26  psi_j(t) in Cmat(i,j)
input one j to plot
5
there are           26  adiabatic state phi_i(t)in Cmat(i,j)
input a window [mst1,mst2] to plot
1 10
Cmat is written in plot.TDDFT.Cmat
```

plot plot.TDDFT.Cmat



B.11.2 example B.11.2: TDDOS/*

we can use OUT.EIGEN.* , OUT.WG.* to run JOB=DOS with PWmat. OUT.EIGEN.* and OUT.WG.* are adiabatic eigen energys and adiabatic wavefunctions.

```
1           1
IN.ATOM    = atom.config
precision = double
convergece=difficult
JOB        = TDDFT
IN.PSP1    = 31-Ga.LDA.fhi.UPF
IN.PSP2    = 33-As.LDA.fhi.UPF
MD_DETAIL = 1, 10, 0.1, 300,300
TDDFT_SPACE = -1
IN.A_FIELD = T  0.1 0.0 0.0
TDDFT_TIME = 2, 5, 1.d0,5.,3., 1.5, 0.0
OUT.TDDFT = T T 1.0 T 1.0
```

>ls TDDOS/

```
OUT.EIGEN.0.100000E+01  OUT.RHO.0.100000E+01  OUT.WG.0.100000E+01
OUT.OCC_ADIA.0.100000E+01
```

> cp OUT.EIGEN.0.100000E+01 OUT.EIGEN

> cp OUT.WG.0.100000E+01 IN.WG

> cp OUT.OCC_ADIA.0.100000E+01 IN.OCC_ADIA

etot.input for DOS [the job=dos will read OUT.EIGEN implicitly]:

```
1           1
IN.ATOM   = atom.config
precision = double
convergece=difficult
JOB       = dos
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
in.wg=t
```

we get file DOS.totalspin.

> cp DOS.totalspin DOS.totalspin.tot

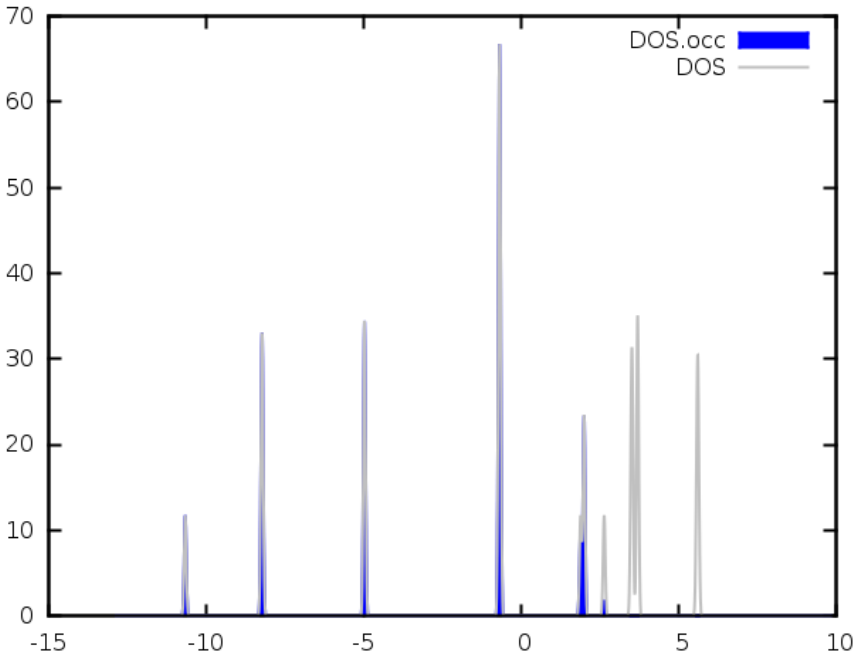
Use IN.OCC_AIDA = T, PWmat will read file IN.OCC_ADIA and we will get occupied desity of adiabatic states.

```
1           1
IN.ATOM   = atom.config
precision = double
convergece=difficult
JOB       = dos
IN.PSP1   = 31-Ga.LDA.fhi.UPF
IN.PSP2   = 33-As.LDA.fhi.UPF
in.wg=t
in.occ_adia=t
```

> cp DOS.totalspin DOS.totalspin.occ

plot DOS.totalspin.occ DOS.totalspin.tot,

```
gnuplot> plot "DOS.totalspin.occ" u 1:($2) w filledcurve lc rgb "blue"
         title "DOS.occ","DOS.totalspin.tot" u 1:2 w l lc rgb "grey"
         title "DOS"
```



B.12 Stability

One can try to adjust the CONVERGENCE, PRECISION, DT(of MD_DETAIL) to get more stable result. DT for TDDFT is recommended to be $\leq 0.1fs$

B.12.1 Energy diverge Problem

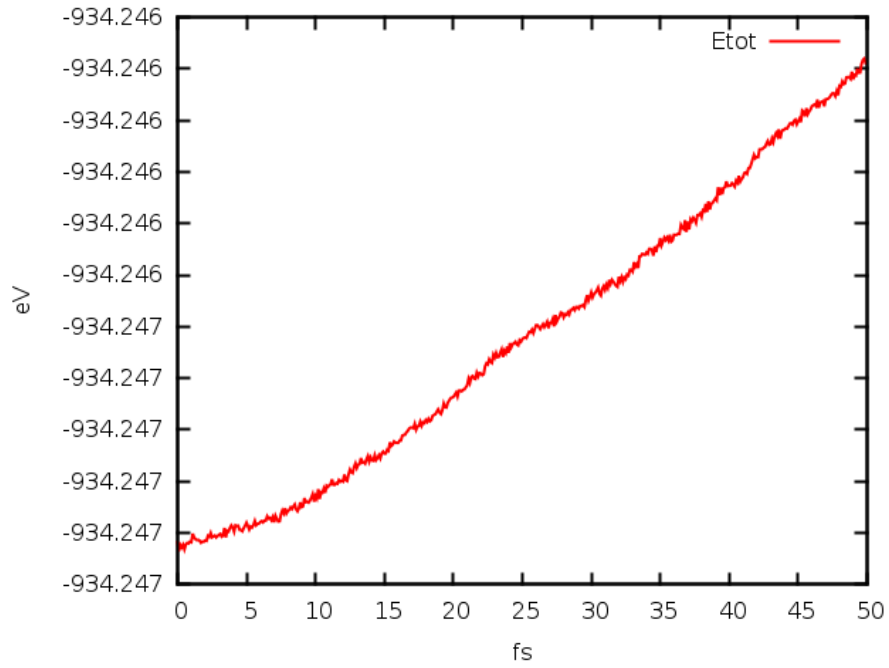
Without external potential, the total energy may blow up in the process of TDDFT.

```
8
Lattice vector
5.65      0.0000000000      0.0000000000
0.0000000000      5.65      0.0000000000
0.0000000000      0.0000000000      5.65
```

```
Position, move_x, move_y, move_z
31   0.010000000000   0.000000000000   0.000000000000   1   1   1
31   0.000000000000   0.501000000000   0.502000000000   1   1   1
31   0.500000000000   0.000000000000   0.500000000000   1   1   1
31   0.500000000000   0.500000000000   0.000000000000   1   1   1
33   0.250000000000   0.250000000000   0.250000000000   1   1   1
33  -0.250000000000  -0.250000000000   0.250000000000   1   1   1
33  -0.250000000000   0.250000000000  -0.250000000000   1   1   1
33   0.250000000000  -0.250000000000  -0.250000000000   1   1   1
```

```
1            1
JOB        = tddft
MD_DETAIL  = 1, 500, 0.1, 300,300
IN.ATOM    = atom.config
in.psp1    = 31-Ga.LDA.fhi.UPF
in.psp2    = 33-As.LDA.fhi.UPF
```

MDSTEPS–Etot plot:

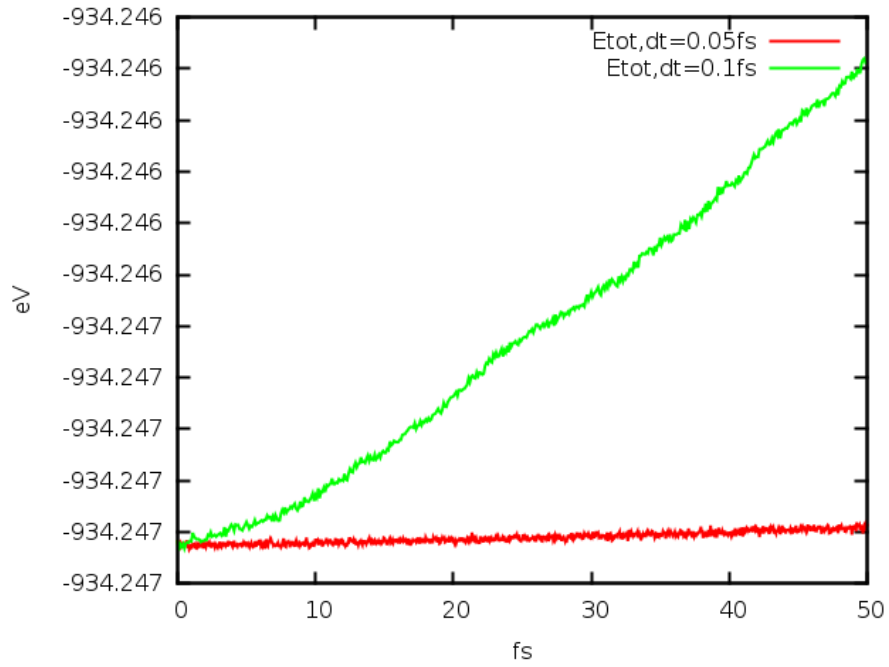


Methods to fix this are as follows.

B.12.1.1 a. use smaller dtMD

```
1            1
JOB        = tddft
MD_DETAIL = 1, 1000, 0.05, 300,300
IN.ATOM    = atom.config
in.psp1    = 31-Ga.LDA.fhi.UPF
in.psp2    = 33-As.LDA.fhi.UPF
```

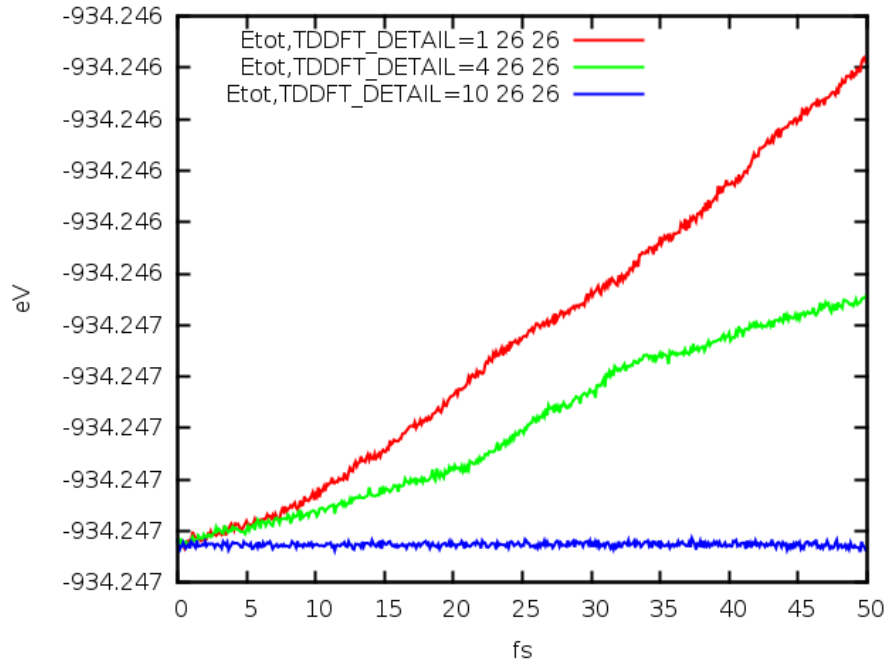
MDSTEPS–Etot plot:



B.12.1.2 b. use proper adiabatic window

```
1            1
JOB        = tddft
MD_DETAIL = 1, 1000, 0.1, 300,300
IN.ATOM    = atom.config
in.psp1    = 31-Ga.LDA.fhi.UPF
in.psp2    = 33-As.LDA.fhi.UPF
TDDFT_DETAIL =  4      26      26
```

MDSTEPS–Etot plot:



B.12.1.3 c. use high accuracy(or convergence=difficult)

55 Ag atoms.

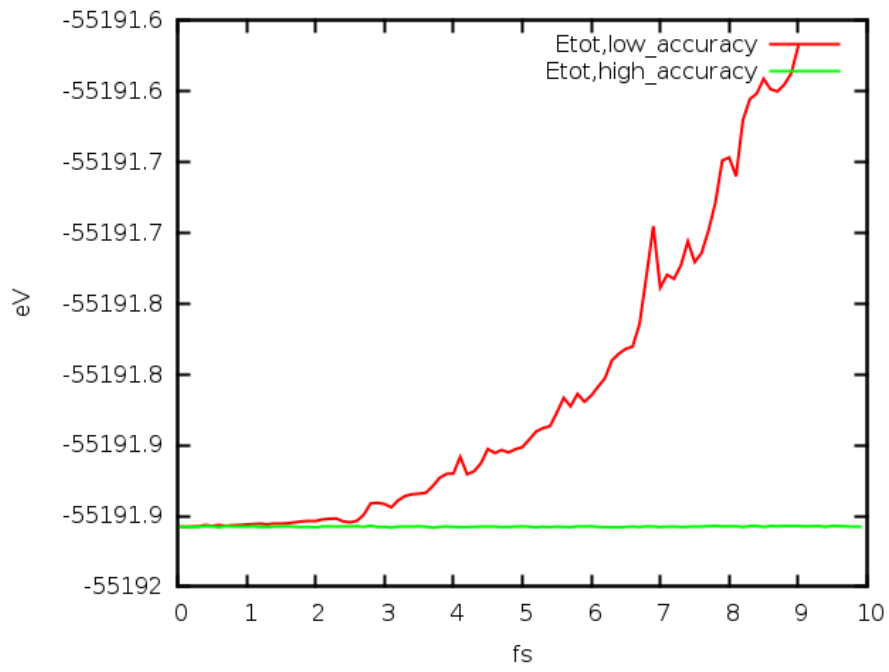
etot.input.low_accuracy

```
4   1
IN.ATOM          =  xatom
JOB              =  TDDFT
MD_DETAIL        =  1, 100, 0.1, 300, 300
IN.PSP1          =  vwr.Ag+U.UPF
```

etot.input.high_accuracy

```
4   1
IN.ATOM          =  xatom
JOB              =  TDDFT
MD_DETAIL        =  1, 100, 0.1, 300, 300
IN.PSP1          =  vwr.Ag+U.UPF
E_ERROR          =   1.00E-005
RHO_ERROR        =   1.00E-005
```

MDSTEPS–Etot plot:



B.13 JOB=NAMD

Non-adiabatic molecular dynamics (NAMD). Like TDDFT, NAMD is a way to simulate the carrier dynamics. It is more suitable for large systems where it is difficult to carry out a full TDDFT simulation. It ignores the feedback from the carrier movement to the nuclear movement, or the electron structure change of other electrons. The idea is that, the rest of the electron (besides this carrier) can be described by the occupied valence states (as in a usual Born-Oppenheimer MD). So, the nuclear movement, as well as the electronic structure during the movement, are described exactly as BO-MD. Such a BO-MD movement can be described by a time dependent Hamiltonian H(t). Thus, NAMD is a by-product of the usual BO-MD. The purpose is to study a carrier dynamics of wave function $\psi(t)$, for the carrier riding on the BO-MD Hamiltonian H(t). This is equivalent to say the occupation of $\psi(t)$ is zero (so there is no feedback from $\psi(t)$ to nuclear movement and H(t)). $\psi(t)$ satisfies the time dependent Schrodinger's equation $i\partial\psi(t)/\partial\psi(t) = H(t)\psi(t)$, or its variance including effects like surface hopping,

Boltzmann correction, or wave function collapsing. There are many problems where we are only interested in the dynamics of this additional electron, but not the nuclear movement or the response from the other electron, then this can be the method to study such problem.

To carry out one NAMD study, one first does JOB=NAMD, which is exactly the same as JOB=MD, also using MD_DETAIL to carry out the simulation. However, the JOB=NAMD will output files OUT.NAMD and ugioallxxxxx, as controlled by NAMD_DETAIL. It can also use external potential etc, as input from TDDFT_SPACE, TDDFT_TIME, TDDFT_STIME, TDDFT_AFIELD. In "MD_DETAIL=iMD,...", it can only use iMD=1 (Verlet algorithm), but it can also use iMD=11, as a RESTART. For the RESTART, OUT.WG from previous run must be copied to IN.WG, and set IN.WG=T in etot.input. Also, OUT.RHO can be copied into IN.RHO. Secondly, final.config from last NAMD run should be used (or copied to) as input IN.ATOM file. Lastly, a proper TDDFT_STIME (in the unit of fs) should also be set to the end time of last NAMD run (e.g., TDDFT_STIME=1000) in etot.input. **(Note)**: For the NAMD RESTART job, the time length DT in MD_DETAIL can not be changed! In the rerun, the output will be appendix into the original OUT.NAMD and ugioallxxxxx.

After the PWmat JOB=NAMD is finished, one can use "namd_dm.x" in Boltzman-NAMD to carry out the actual Non-adiabatic dynamics. Multiple runs can be carried out to explore different scenarios.

B.13.1 NAMD_Boltzman.x

WARNING: We'd highly recommend you use "namd_dm.x" to simulate Boltzman NAMD, instead of "NAMD_Boltzman.x". Please refer to Boltzman-NAMD for more information. In this module, we developed a new NAMD simulation method by modifying the conventional density matrix, it can incorporate the detailed balance and decoherence.

This utility code is a open-source utility code to carry out NA-MD using the JOB=NAMD output OUT.NAMD. The input file NAMD.input has the following

format:

```
100     ! MDstep: should be the same as from MDstep
        !         in "MD_DETAIL=iMD,MDstep,dtMD,T1,T2"
        !         in etot.input
21      ! nstates: mst2-mst1+1, from "NAMD_DETAIL =
        !          mst2,mst1,mout" in etot.input
1.0     ! dtMD (fs): should be the same as dtMD
        !            in etot.input
1  300  0.3  !  e(1)/h(-1), temperature(kevin),
             !  win_Boltz(eV), the temperature
             !  can be diff from T1,T2
5       ! mout: should be the same as ‘‘NAMD_DETAIL
        !       = mst1,mst2,mout’’ in etot.input
```

The win_Boltz is to define a window, when |E_i1-E_i2| smaller than this window, then the transition between these two states will be multiplied by a Boltzman factor to restore the detail balance between these two states. If this wind_Boltz is zero, then no Boltzman factor is added.

Like in the TDDFT simulation, the wave function $\psi(t)$ will be expanded by the adiabatic states $\phi_i(t)$ (from the BO-MD eigen states) $\psi(t) = \sum_{i=m_1,m_2} C_i(t)\phi_i(t)$. The initial coefficient $C_i(t=0)$ will be input from NAMD.cc_init. It has a format like:

```
21     # nstate=mst2-mst1+1
0.707,0.0  # C_1(0)
0.0,0.0  # C_2(0)
0.707,0.0  # C_3(0)
0.0,0.0  # C_4(0)
.......
```

The output of the NAMD_Boltzman.x run is reported in NAMD.graph.aveE, NAMD.graph.eigen, NAMD.graph.cc. While the NAMD.graph.eigen reports the individual adiabatic state $\phi_i(t)$ eigen energies $E_i(t)$, NAMD.graph.aveE reports the energy of $\psi(t)$ as a function of time t. NAMD.graph.cc reports the expansion coefficients

$C_i(t)$ as a function of time. These data can be plotted, or plotted using gnuplot with plot.db or plot2.db (> gnuplot plot.db)

Finally, the wave function $\psi(t)$ can also be plotted in real space $\psi(r)$ with an interval mout*dtMD (defined in NAMD.input). This can be done running NAMD_psi.x.

B.13.2 example B.14.2: NAMD

step1. run PWmat

etot.input:

```
4           1
IN.ATOM    = atom.config
JOB        = NAMD
MD_DETAIL = 1, 100, 1.0, 1000, 500
NAMD_DETAIL = 295,315,5
NUM_BAND   = 335
TDDFT_TIME  = 2, 4, 1, 40, 20, 0.5
TDDFT_SPACE = 3, 5,0.5,0.5,0.5, 0.02, 5.
TDDFT_STIME = 0.0
IN.PSP1    = Ga.nc.pbe.UPF
IN.PSP2    = As.nc.pbe.UPF
IN.PSP3    = Cu.nc.pbe.UPF
```

step2. run NAMD_Boltzman.x

WARNING: We'd highly recommend you use "namd_dm.x" to simulate Boltzman NAMD, instead of "NAMD_Boltzman.x". Please refer to Boltzman-NAMD for more information. In this module, we developed a new NAMD simulation method by modifying the conventional density matrix, it can incorporate the detailed balance and decoherence. The following steps are reference for using "NAMD_Boltzman.x" only!

NAMD.input:

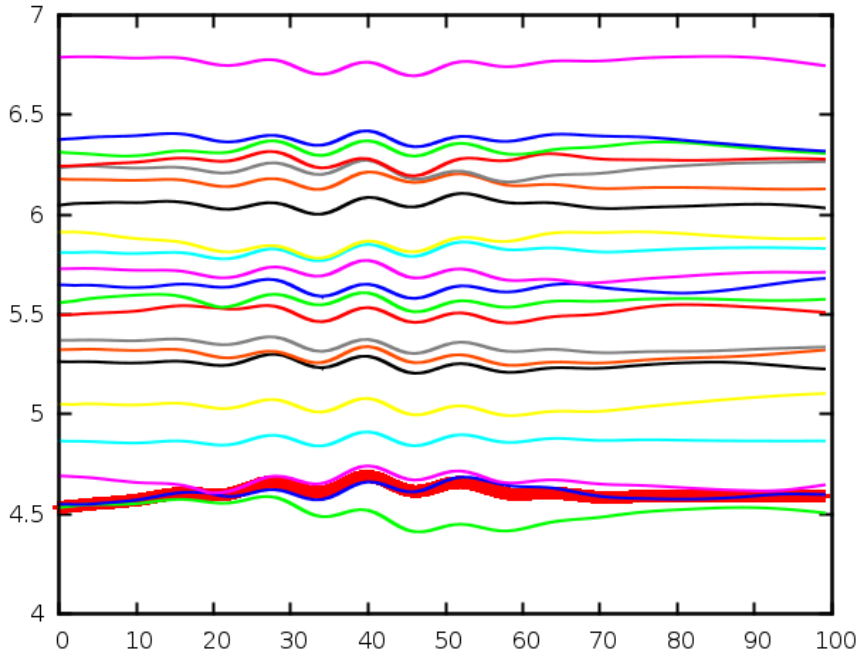
```
100      ! nstep
21       ! nstates
1.0      ! dt (fs)
```

```
1  300  0.3  !  e(1)/h(-1), temperature, win_Boltz(eV)
5       ! mout
```

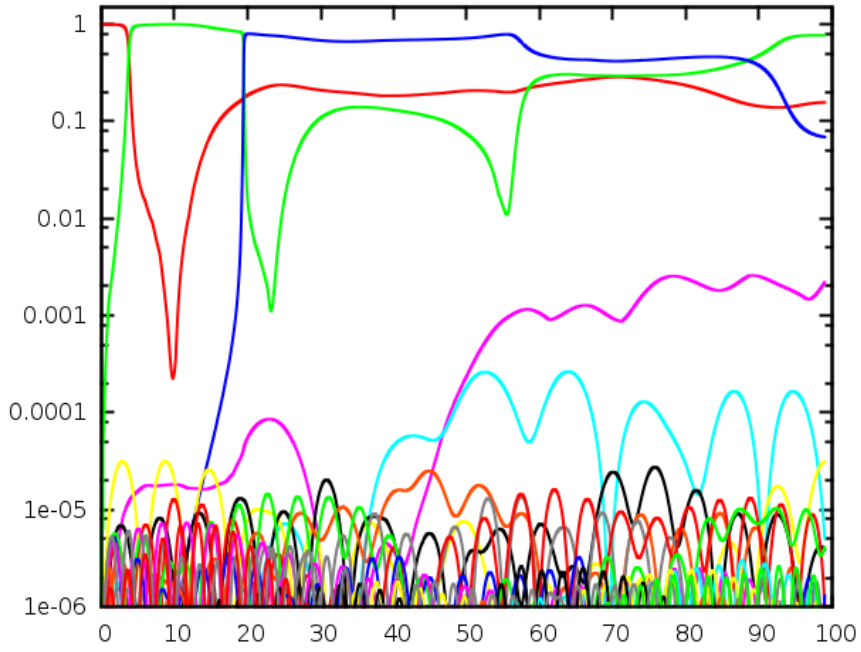
NAMD.cc_init

```
21
1.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
0.0,0.0
```

plot NAMD.graph.eigen,NAMD.graph.avE

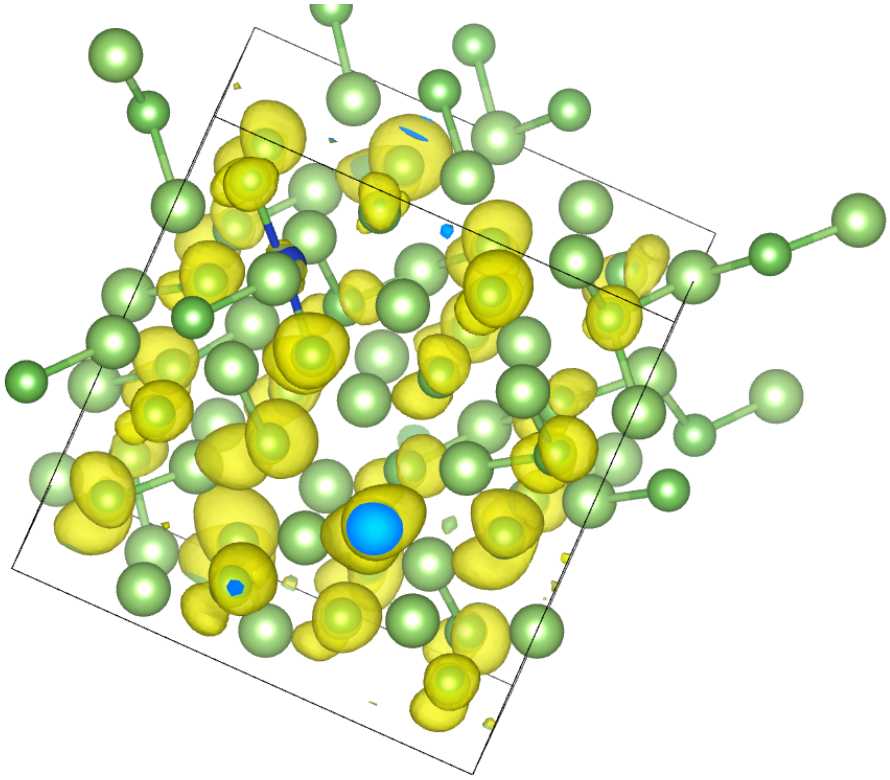


plot NAMD.graph.cc



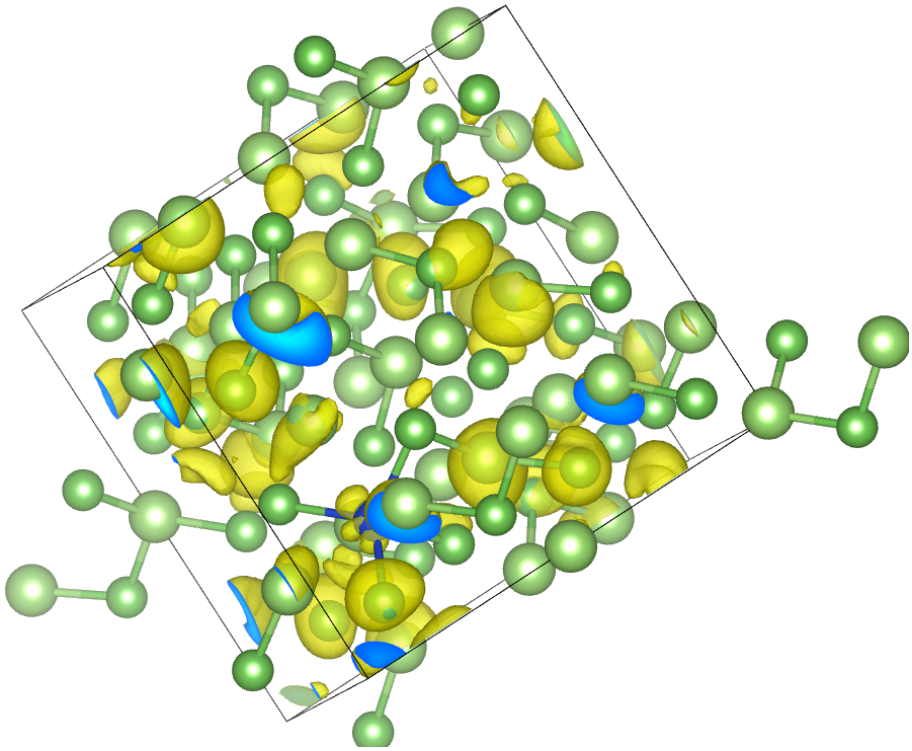
plot wavefunction use NAMD_psi.x

> NAMD_psi.x



plot wavefunction use plot_wg.x

> plot_wg.x



Bibliography

§ **PWmat Reference**

[1] Jia, W., Fu, J., Cao, Z., Wang, L., Chi, X., Gao, W. & Wang, L. W. (2013). *Fast plane wave density functional theory molecular dynamics calculations on multi-GPU machines. Journal of Computational Physics. 251, 102-115.*

[2] Jia, W., Fu, J., Cao, Z., Wang, L., Chi, X., Gao, W. & Wang, L. W. (2013). *The analysis of a plane wave pseudopotential density functional theory code on a GPU machine. Computer Physics Communications. 184(1), 9-18.*

[3] Zhi Wang, Shu-Shen Li, and Lin-Wang Wang. *Effcient Real-Time Time-Dependent Density Density Functional Theory Method and its Application to a Collision of an Ion with a 2D Material. Physical Review Letters. 114, 063903(2015).*

[4] Verlet, Loup. *Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard Jones Molecules. Physical Review. 159, 98-103(1967)).*

[5] A. Brünger, C. L. Brooks III, M. Karplus. *Stochastic boundary conditions fro molecular dynamics simulations of ST2 water. Chem. Phys. Letters, 1984, 105 (5) 495-500.*

[6] Nose, S. *A unified formulation of the constant temperature molecular-dynamics methods. Journal of Chemical Physics. 81, 1(1984).*

[7] Hoover, William G. *Canonical dynamics: Equilibrium phase-space distribution. Phys. Rev. A. 31, 3(1985).*

[8] Berendsen *to be filled later*

[9] Berendsen *to be filled later, for cell scaling*

[10] M. Parrinello, A. Rahman. *Polymorphic transitions in single crystals: A new molecular dynamics method. Journal of Applied Physics 52, 7182 (1981).*

[11] D.Quigley, M.I.J. Probert. *Constant pressure Langevin dynamics: theory and application. Computer Physics Communications 169(2005) 322-325.*

[12] G.J.Martyna, J.T. Tobias, M.L. Klein. *Constant pressure modecular dynamics algorithms, J. Chem. Phys. 101(5)(1994) 4177-4189.*

[13] O. Andreussi, I. Dabo and N. Marzari. *Revised self-consistent continuum solvation in electronic-structure calculations. J. Chem. Phys. 136, 064102(2012).*

[14] Reed, Fried, and Joannopoulos. *Phys.Rev.Lett., 90, 235503 (2003).*

[15] K. Letchworth-Weaver, T.A. Arias *Joint density functional theory of the electrode-electolyte interface: Application to fixed electrode potentials, interfacial capacitances, and potentials of zero charge. Phys. Rev. B 86, 075140 (2012).*

[16] H. Jonsson, G. Mills and K. W. Jacobsen. *'Nudged Elastic Band Method for Finding Minimum Energy Paths of Transitions' in 'Classical and Quantum Dynamics in Condensed Phase Simulations', ed. B. J. Berne, G. Ciccotti and D.F.Coker(World Scientific, 1998).*

[17] G. Henkelman and H. Jonsson, *Improved Tangent Estimate in the NEB method for Finding Minimum Energy Paths and Saddle Points, J. Chem. Phys. 113, 9978 (2000)*

[18] G. Henkelman, B. P. Uberuaga and H. Jonsson, *A Climbing-Image NEB Method for Finding Saddle Points and Minimum Energy Paths, J. Chem. Phys. 113, 9901 (2000)*

[19] EsbenL.Kolsbjerg,MichaelN.Groves,and Bjork Hammer, *An automated nudged elastic band method, J. Chem. Phys. 145, 094107 (2016)*

[20] Erik Bitzek,Pekka Koskinen,Franz Gahler,Michael Moseler,and Peter Gumbsch , *Structural Relaxation Made Simple,PRL 97, 170201 (2006)*

[21] Miguel A. L. Marques, Micael J. T. Oliveira, Tobias Burnus. *LIBXC: a library of exchange and correlation functionals for density functional theory. Comput. Phys. Commun. 183, 2272(2012).*

[22] Susi Lehtola, Conrad Steigemann, Micael J. T. Oliveira, Miguel A. L. Marques. *Recent developments in LIBXC-A comprehensive library of functional for density functional theory. Software X, 7, 1(2018).*

[23] P Gomes Dacosta, O H Nielsen and K Kunc. *Stress theorem in the determination of static equilibrium by the density functional method. J. Phys. C: Solid State Phys. 19, 3163-3172(1986)*

[24] S. Grimme, J. Antony, S. Ehrlich, and S. Krieg. *A consistent and accurate ab initio parametrization of density functional dispersion correction (dft-d) for the 94 elements H-Pu. J. Chem. Phys. 132, 154104 (2010).*

[25] Kiran Mathew, Ravishankar Sundararaman, Kendra Letchworth-Weaver, T. A. Arias, and Richard G. Hennig.*Implicit solvation model for density-functional study of nanocrystal surfaces and reaction pathways. The Journal of Chemical Physics 140, 084106 (2014).*

[26] Oliviero Andreussi, Ismaila Dabo, and Nicola Marzari. *Revised self-consistent continuum solvation in electronic-structure calculations. The Journal of Chemical Physics 136, 064102 (2012).*

[27] G. Fisicaro, L. Genovese, O. Andreussi, N. Marzari, S. Goedecker. *A generalized Poisson and Poisson-Boltzmann solver for electrostatic environments. The Journal of Chemical Physics 144, 014103 (2016).*

[28] J.Ma, L.-W. Wang. *Using Wannier functions to improve solid band gap predictions in density functional theory. Sci. Rep. 6, 24924 (2016).*

[29] M. Cococcioni, S. de Gironcoli. *Linear response approach to the calculation of the effective interaction Parameters in the LDA+U method, Phys. Rev. B, 71, 035105 (2005).*

[30] Fan Zheng, Hieu H. Pham, Lin-Wang Wang. *Effects of the c-Si/a-SiO2 interfacial atomic structure on its band alignment: an ab initio study, Phys. Chem. Chem. Phys., 19, 32617 (2017).*

[31] I. Souza, J. Íñiguez, and D. Vanderbilt. *First-Principles Approach to Insulators in Finite Electric Fields, Phys. Rev. Lett, 89, 117602 (2002).*

[32] A. Pieper, M. Kreutzer, A. Alvermann, M. Galgon, H. Fehske, G. Hager, B. Lang, G. Wellein. *High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations, J. Comp. Phys. 325 (2016) 226–243*

[33] G. Henkelman and H. Jónsson, *A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives, J. Chem. Phys. 111, 7010(1999).*

[34] A. Heyden, A. T. Bell, and F. J. Keil, *Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method, J. Chem. Phys. 123, 224101, (2005).*

[35] D. Porezag and T. Frauenheim and T. Köhler and G. Seifert and R. Kaschner. *Construction of tight-binding-like potentials on the basis of density-functional theory: Application to carbon, PRB. 51 12947 (1995).*

[36] M. Elstner and D. Porezag and G. Jungnickel and J. Elsner and M. Haugk and T. Frauenheim and S. Suhai and G. Seifert. *Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties, PRB 58, 7260 (1998).*

[37] B. Aradi and B. Hourahine and Th. Frauenheim, *DFTB+, a Sparse Matrix-Based Implementation of the DFTB Method, jpca.26, 5678 (2007).*

[38] Hourahine, B. and Sanna, S. and Aradi, B. and Köhler, C. and Niehaus, T. and Frauenheim, Th., *Self-Interaction and Strong Correlation in DFTB, jpca, 26, 5671 (2007)*

[39] Yang, Y. and Yu, H. and York, D. and Cui, Q. and Elstner, M., *Extension of the Self-Consistent-Charge Density-Functional Tight-Binding Method: Third-Order Expansion of the Density Functional Theory Total Energy and Introduction of a Modified Effective Coulomb Interaction, jpca, 111, 10861 (2007).*

[40] Zhechkov, L. and Heine, Th. and Patchkovskii, S. and Seifert, G. and Duarte, H. A., *An Efficient a Posteriori Treatment for Dispersion Interaction in Density-Functional-Based Tight Binding, jctc, 1, 841-847 (2005).*

[41] Gaus, M. and Cui, Q. and Elstner, M. *DFTB3: Extension of the Self-Consistent-Charge Density-Functional Tight-Binding Method, jctc, 7, 931-948 (2011).*

[42] Grimme, S. and Antony, J. and Ehrlich, S. and Krieg, H., *A consistent and accurate ab initio parametrization of density functional dispersion correction for the 94 elements H-Pu, jcp, 132, 154104 (2010).*

[43] Grimme, S. and Ehrlich, S. and Goerigk, L.,*Effect of the Damping Function in Dispersion Corrected Density Functional Theory, jcp,32,1456-1465 (2011).*

[44] Caldeweyher,Eike and Ehlert,Sebastian and Hansen,Andreas and Neugebauer,Hagen and Spicher,Sebastian and Bannwarth,Christoph and Grimme,Stefan, *A generally applicable atomic-charge dependent London dispersion correction, jcp, 15, 154122 (2019).*

[45] Tkatchenko, Alexandre and Scheffler, Matthias, *Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data, PRL,102,073005 (2009).*

[46] Aradi, B. and Niklasson, A. M. N. and Frauenheim, T., *Extended Lagrangian Density Functional Tight-Binding Molecular Dynamics for Molecules and Solids, jctc, 11, 3357-3363 (2015).*

[47] Hourahine,B. and Aradi,B. and Blum,V. and Bonafé,F. and Buccheri,A. and Camacho,C. and Cevallos,C. and Deshaye,M. Y. and Dumitrică,T. and Dominguez,A. and Ehlert,S. and Elstner,M. and van der Heide,T. and Hermann,J. and Irle,S. and Kranz,J. J. and Köhler,C. and Kowalczyk,T. and Kubař,T. and Lee,I. S. and Lutsker,V. and Maurer,R. J. and Min,S. K. and Mitchell,I. and Negre,C. and Niehaus,T. A. and Niklasson,A. M. N. and Page,A. J. and Pecchia,A. and Penazzi,G. and Persson,M. P. and Řezáč,J. and Sánchez,C. G. and Sternberg,M. and Stöhr,M. and Stuckenberg,F. and Tkatchenko,A. and Yu,V. W.-z. and Frauenheim.T, *DFTB+, a software package for efficient approximate density functional theory based atomistic simulations, jcp, 12, 124101(2020).*

[48] Bannwarth, Christoph and Ehlert, Sebastian and Grimme, Stefan, *GFN2-xTB—An accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions, jctc, 15, 1652-1671 (2019).*

[49] Grimme, Stefan and Bannwarth, Christoph and Shushkov, Philip, *A robust and accurate tight-binding quantum chemical method for structures, vibrational frequencies, and noncovalent interactions of large molecular systems parametrized for all spd-block elements (Z=1–86), jctc, 13, 1989 (2017).*

[50] Bitzek, Erik and Koskinen, Pekka and Gähler, Franz and Moseler, Michael and Gumbsch, Peter,*Structural Relaxation Made Simple, PRL, 97, 170201 (2006).*

[51] Rappe, A. K. and Casewit, C. J. and Colwell, K. S. and Goddard III, W. A. and Skiff, W. M.,*UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations, jacs, 114, 10024-10035 (1992).*

Release Note

2024.05.07

1. Fix Bug of DFTD3 with some initiallization of variables.
2. Fix Bug of HSE stress calculation when using multiple kpoints and processors.
3. Add support of MD_SLOWGROWTH in JOB=MD.
4. Add new method for SCF_SPECIAL with iflag = 3.
5. Fix bug of JOB=NEB when atoms not move in some steps.
6. Remove max_move settings for JOB=RELAX when using CG method.
7. Fix bug of stress synchronization when using multiple kpoints and processors.
8. Update HSE relaxation, now the parameter HSE_PBE_SCF can control the leading PBE SCF in HSE relaxation.

2024.04.01

1. Fix bug of plumed force calculation.
2. Fix bug of stress calculation when with f-electron.
3. Fix bug of inconsistent wait and isend, in invfft, when with large number of node1.
4. Speed up DFT-D3 calculation.
5. Add output of CHARGE_DECOMP and ENERGY_DECOMP in file MOVEMENT when JOB=RELAX.
6. Update error infos for input parameters.
7. Add new method for CHARGE_DECOMP.

2024.02.26

1. Add DFTB support.
2. Add new license server support.

3. Update DCU profiled codes.

4. Fix bug in stress calculation of JOB=RELAX, wrong initialization of dftd3 stress, in some cases may cause strange results.

5. Add new XCFUNCTIONAL=LDAWKM3/PBEWKM3.

6. Fix bug in emotif calculation.

2024.01.25

1. Fix bug in JOB=RELAX when using gaussian basis.

2. Update subroutines for NEGF support.

3. Profile the performance of gaussian basis, still need further improvement.

2023.12.28

1. Added DFTB interface via linking DFTB+ library

2023.12.25

1. Fixed a bug related to nonlocal=3.

2. Added support for nonlocal=3 in LDA+U.

3. Updated the memory usage of nonlocal projectors, now using MPI shared memory.

4. Fxied a bug related to JOB=MD with Gaussian basis.

2023.11.24

1. Updated the default value of LDAU_RCUT_PSP to 4.0 for LDA+U. If the distance between atoms is smaller than LDAU_RCUT_PSP, decrease its value.

2023.11.17

1. Fixed a bug related to FFT when using a large number of Node1.

2. Fixed a bug related to Ewald stress when using K-points parallelization.

2023.11.15

1. Updated the parameters of the Gaussian basis. Refer to the section USE_GAUSSIAN for details.

2023.11.06

1. Updated the automatically generated N123 to ensure efficient FFT. N1, N2, and N3 are now chosen as good numbers, but still require N1*N2*N3 to be divisible by NODE1.

2023.10.25

1. Updated the automatically generated N123 to ensure efficient FFT. N1, N2, and N3 are now chosen as good numbers, but still require N1*N2*N3 to be divisible by NODE1. The update also takes into consideration the general symmetry of the lattice.

2. Added support for PAW pseudopotential.