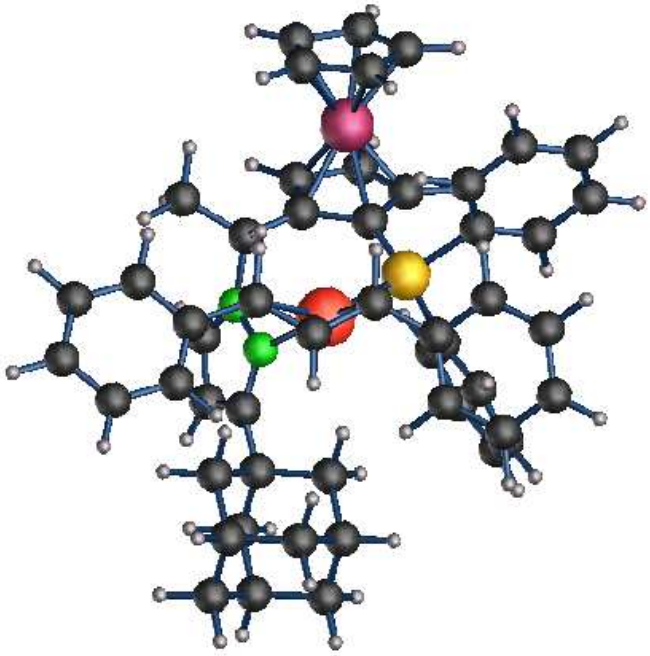
Manual for the Projector Augmented Wave Method

Version 2.0



Peter E. Blöchl, Clausthal University of Technology
(December 9, 2007)

1

[1]The title picture shows the a chiral Pd complex with P,N ligands, a highly enantio-selective catalyst for allylic amination [1].

Contents

1 Initial remarks

I am frequently changing the CP-PAW program. Therefore, it is unavoidable that this description is incomplete in some places, that it may lists options that are no longer supported, or that it contains errors. Please let me know if you find something unclear or even incorrect. Other users will be grateful.

I have attempted to make the program test for inconsistencies of the input data, such as the selection of conflicting options. You will find that the program stops in such cases and tries to advise the user concerning what has gone wrong. New users are particularly creative in the combinations of options they choose. If you run into problems that the program does not detect, for example if it crashes without giving a useful message, please let me know *before* you get used to working around the problem. Your input is particularly valuable in making the the code more secure to use.

Any other suggestions on how to improve the clarity of this description or the code are most welcome.

Programs become outdated nearly as quickly as newspapers. Note that the executable of this code may also have an expiration date. The code will print a warning about a month before it expires. Please make sure that you know the expiration date when you receive an executable. If the license expires, discuss with your contact person about obtaining an extension of the license and a new executable program.

Note that the CP-PAW code including all related material is copyrighted. You require a valid license to execute it.

2 What is the projector augmented wave method?

The projector augmented wave method [3] is an all-electron electronic structure method, which allows accurate electronic structure calculations and ab-initio molecular dynamics simulations on the basis of density functional theory.

What is all that?

Density functional theory (DFT) [4, 5]. Density functional theory describes the ground state of a many-electron system by electrons that do not interact other than through an effective potential that depends on the electron density. It is based on an exact theorem, which specifies that such a description, based on the electron density rather than on the electronic many-particle wave function, be rigorously possible for ground states. In practice the density functional, which also defines

the effective potential as a functional of the density, is not exactly known. However, highly successful approximations have been found. In contrast to Hartree-Fock calculations, density functional theory explicitly treats electron correlation. The accuracy is typically comparable to that of MP2 calculations, i.e. only a few kcal/mol [6, 7].

Ab-initio molecular dynamics (AIMD) [8, 9, 10] is an extension of traditional electronic structure methods which has been invented in 1985 by Roberto Car and Michele Parrinello [8]. The best way to think of it is as a series of electronic structure calculations, one for each time slice, for always different atomic positions. From one time slice to the next, the atomic positions are changed according to Newton's equations of motion $M_i\ddot{R}_i = F_i$. Here M_i is the mass of a nucleus, R_i the position, F_i the force acting on the nucleus as calculated from the electronic structure, and the double-dots stand for the second time derivative. Self-consistent iterations at each time step are avoided by a dynamical evolution of the wave functions, and thus simulations of several picoseconds are possible, which is sufficient to simulate directly chemical reactions and diffusion with low barriers or at high temperatures.

Whereas the basic idea of ab-initio molecular dynamics is to perform real-time and finite temperature simulations, it can be used like a traditional electronic structure method – using a friction to "cool" the temperature to zero – and it has been combined with statistical approaches to study processes with large barriers.

The **projector augmented wave method (PAW)** [3] has been developed by the author in response to the invention of the ab-initio molecular dynamics approach. Whereas the latter was based on the plane wave pseudopotential approach, a new method was needed to enhance the accuracy and computational efficiency of the approach and to provide the correct wave functions, rather than the fictitious wave functions provided by the pseudopotential approach. The PAW method describes the wave function by a superposition of different terms: There is a plane wave part, the so-called pseudo wave function, and expansions into atomic and pseudo atomic orbitals at each atom. On one hand, the plane wave part has the flexibility to describe the bonding and tail region of the wave functions, but used alone it would require prohibitive large basis sets to describe correctly all the oscillations of the wave function near the nuclei. On the other hand, the expansions into atomic orbitals are well suited to describe correctly the nodal structure of the wave function near the nucleus, but lack the variational degrees of freedom for the bonding and tail regions. The PAW method combines the virtues of both numerical representations in one well-defined basis set.

Of course, one does not want to make two electronic structure calculations –

one using plane waves and one with atomic orbitals –, and thus double the computational effort. Therefore, the PAW method does not determine the coefficients of the "atomic orbitals" variationally. Instead, they are unique functions of the plane wave coefficients. It is possible to break up the total energy, and most other observable quantities, into three almost independent contributions: one from the plane wave part and a pair of expansions into atomic orbitals on each atom. The contributions from the atomic orbitals can be broken down furthermore into contributions from each atom, so that strictly no overlap between atomic orbitals on different sites need to be computed.

The PAW method is in principle able to recover rigorously the density functional total energy, if plane wave and atomic orbital expansions are complete. This provides us with a systematic way to improve the basis set errors. The present implementation uses the frozen core approximation, even though the general formalism allows extensions in this respect. It provides the correct densities and wave functions, and thus allows us to calculate hyperfine parameters etc. Limitations of plane wave basis sets to periodic systems (crystals) can easily be overcome by making the unit cell sufficiently large and decoupling the long-range interactions [11]. Thus the present method can be used to study molecules, surfaces, and solids within the same approach.

3 How the code is built

This section is about programming philosophy. I write this up because I myself often wonder when I use other software why something is done in a particular fashion. Therefore, I shall present my reasoning here. Feel free to skip this section if you wish.

The program is written in an **OO (object-oriented)** manner. This means that it consists of agents (objects) that perform certain operations or provide the selected information. Each agent holds the data needed to its job, or it can request the data it needs from other agents. This is a major software strategy, widely used today, which allows the programmer to hide certain details, he need not really worry about at the present level of programming. It also allows him to assemble the code from little "boxes", which are easy to maintain and enhance.

As part of the OO design, the program uses its own low level object library, which customizes a number of common operations, such as interproceessor communication, error handling, file handling, tree-linked-list structures for intuitive IO, periodic table, constants, DFT functionals, tracing, timing, string handling,

and a few more. These low-level objects are rather unspecific to the PAW code, and can be used in combination with self-developed analysis tools. (However, they are still subject to the license agreement for the PAW code.)

The language used is **FORTRAN90**. FORTRAN90 is itself not an OO language such as C++ and Java, and thus limits the possibilities in this respect. However, it has a number of advantages for number crunching, such as good compilers and, for my taste, a natural and easily comprehensible way to write mathematics. Compared to FORTRAN77 it is a significant advance towards the OO features of C++ or Java because it incorporates features such as dynamic memory allocation, derived data types (structures), operator overloding, modules, etc. The option of using templates has been implemented using a self-made preprocessor. FORTRAN90 is largely compatible with FORTRAN77, so you need to pick up only a few new things if you want to work with the existing tools.

The program allows **parallel processing** using MPI (Message Passing Interface) [12]. It is (almost) scalable in central memory and CPU time. The scalar version program is identical to the parallel version with the exception that a dummy interface is used instead of MPI.

The program relies heavily on linear algebra packages such as LAPACK, BLAS and FFTW. These libraries takes care of basic computations such as matrix multiplications and FFTs (Fast Fourier transforms). A large fraction of the total CPU time is spent in these routines. In this way the code development concentrates on algorithmic developments and not, for example, on how to optimize a FFT. The latter will be done by experts who continually adapt this library to modern computer architectures.

One programming principle I try to follow is the German engineering motto **"Fewer Parts!"**. For the user this implies that he will find few instances where two options provide the same functionality. I hope that the limitations will be offset by clarity.

The program uses **Hartree atomic units**, that is $\hbar = e = m_e = 4\pi\epsilon_0 = 1$, and Cartesian coordiantes. Angles are handled in radian ($2\pi \text{ rad} = 360 \text{ deg}$). The conversion to other units is often done during printing. The conversion factors are provided by a particular agent (see section 18), which is based on the values of fundamental physical constants recommended in 1986 by CODATA (Committee on Data for Science and Technology of the International Council of Scientific Unions) [13].

The program divides work into **three steps: simulation – analysis – visualization**. The program consists of the simulation code, which is the core of CP-PAW, and a number of tools used to analyze the results. The simulation code

calculates energies, densities, wave functions, trajectories, etc., and writes the resulting data into files. These files are then read by tools, which collect the desired information, and bring it into the desired form, typically as another file that can be read by your graphics utilities.

Why this three-layer strategy?

Analysis is both an iterative process and an art: when you find something interesting and want to take a closer look. Sooner or later, you will probably want to make your own analysis tools, because you have found a way to understand your data that nobody has thought of before. Or you want examine a property, which nobody has tried to calculate. Or you have a unique visualizer and need an interface for it. And because you want to discuss this particular result with your collegues at the upcoming coffee break, you choose the quick-and-sloppy approach. You do not want to do this inside the simulation code, because you dare not jeopardize its operation.

In contrast to analysis, simulation is computationally expensive. Therefore, it is desirable to archive as much data from the simulation as possible, and be able to go back later and look at it again. Hence the simulation writes most data rather unspecifically in machine-readable form (which saves a considerable amount of disk space). The data are written in a simple format so that it is easy to read them into the analysis tools.

Visualization is also separated from analysis, because many tools exist and the choice is a matter of taste and wealth. The analysis tools that come with the CP-PAW code will write data formats for the visulatization tools that we currently use. (Those are currently the IBM Dataexplorer [2] for 3D representations of the wave functions or trajectories, xmgrace [14] for x-y plots, and CryMolCAD[15] for the analysis of the atomic structure.) However, you can easily change the format to adapt them to your preferred graphics programs.

Of course you can see the most important information, such as total energies and one-particle energies directly while the simulation is proceeding. You may also contact other users about further analysis tools and exchange them. If you wish to write a graphical interface that hides all three steps from the user, feel free to do so.

4 Input data structure

The input data of the simulation code and the tools uses a format that attempts to be both general and intuitive. Logically, the data are arranged in a tree structure

similar to pull-down menues or directory trees in unix. It allows one to hide options of the program that the user may not be interested in, which then can be handled by default values, and it avoids the unnecessary restrictions of formatted input. The PAW library has objects that can handle such structures easily, so that it is widely used for data input for both the simlation code and the analysis tools. The following section shall make you familiar with the general layout of input data.

4.1 Syntax rules for the input data

Input data are structured as nested data blocks, each identified by a key word. Each data block may contain other data blocks and data. Data are again specified by keywords.

The following simple expample shall illustrate the structure

```
!FIRSTBLOCK
  DATA=5
  !SECONDBLOCK OTHERDATA=T !END
  !SECONDBLOCK OTHERDATA=F !END
  !THIRDBLOCK_OFF TEXTDATA='THIS IS A TEXT' !END
!END
!EOB
```

The indentation and the arrangement of the data is arbitrary. The only requirements are that data and keywords be separated by blanks or line breaks, and that their sequence observes the logical tree structure described below.

Every block starts with a block identifier and ends with the string "!END". The identifier starts with an exclamation mark such as !FIRSTBLOCK. (Of course one can use any other name instead of "FIRSTBLOCK". The recognized block key words are described in the later sections of this manual.) The last block must be followed by "!EOB" (End-Of-Buffer) to indicate the end of all data blocks. Each block, together with its data and any contained subblocks, can be made invisible by appending "_OFF" to the block name as done for !THIRDBLOCK in the example. Data blocks may occur multiply such as !SECONDBLOCK, if specified in this manual. The order in which the data blocks are given is irrelevant as long as their hierarchy is observed. An exception are multiple data or datablocks, where the order in which they occur may or may not be significant. Each data refers only to its block. For example, the two occurrences of OTHERDATA= are dif-

ferent even though their key words are identical, because they are within different subblocks.

The general format of input data is a key word, as specified in this manual separated by an "=" sign from the input data. The input data can be simple data or arrays. Higher dimensional arrays are treated like in Fortran with the first dimension incremented first, such as a(1,1) a(2,1) a(1,2) a(2,2). The data are read in free format.

Note that a mistyped key word makes the data or entire branches of the tree structure invisible. There is no way to warn the user if some key words have not been recognized. If the same key word is given several times in a given data block, without being specified as a multiple, only the first occurrence is recognized. The same is true for data blocks. It is therefore recommended that the protocol be checked to determine, whether all data have been used as intended.

The type of data is determined as follows:

- if a data contains a single or double apostrophe, it is assumed to be of character type,
- if a data contains an open parenthesis, it is assumed to be of complex type,
- if a data is either T, F, .true. or .false. it is assumed to be of type logical,
- if a data contains a period it is assumed to be of type real,
- otherwise, the data is assumed to be of type integer.

These rules are checked in the order given here. It is allowed to precede a data item by an integer and a multiplication sign such as $3 * 0.$, which is shorthand for 0. 0. 0. (In some cases, one is allowed to provide both integer and real. Therefore, if you provided a number of the wrong type and, against your expectation, the program does not complain, a conversion has been done by rounding the number to the nearest integer or a simple type conversion from integer to real has occured.)

The description uses the following notation. Data blocks are indicated by a frame. The enclosed name contains the entire hierarchy of data blocks. Only the last one must be specified on the date file. However, this data block must be enclosed by the higher level data blocks. The key words relate to the individual data. If we cross reference data, we use the entire hierarchy of parent blocks, followed by a colon and the data keyword.

4.2 Extended notation for atoms

The PAW code refers to atoms by name. No two atoms must be given the same name. Arbitrary names, given a length limit, are allowed. However, the recommended notation choses the first two characters as the element symbol, with blanks replaced by underscores, and the remainder is a number. Starting the name with the element symbol allows to exploit some default settings. The names are defined in the structure input file and can be referred to in other specifications and by the PAW tools.

In solids, it will be neccesary to distinguish between periodic images of the same atom. An example is a bondlength constraint between two atoms, of which one is a periodic image of a original atom. In that case we use in some cases an extended naming convention.

In the extended notation the atom name has the form "*name*:*ijk*", where *name* is the name of the atom in the original atom cell. and *i*,*j*,*k* are the three translations along the lattice vectors. The latter can be single digit integer numbers with or without preceeding sign. To give a few examples "H_2:100", "H_2:+1+0+0", "H_2:10-1".

Note, that the translations do *not* specify a unit cell that contains that atom. For example, an atom that moves about will never change its lattice translations, despite traversing several unit cells. Rather the atoms are considered first as a initial cluster, which forms a lattice by repetition and lattice translations. There is no restriction that the atoms of the initial cluster are confined in one unit cell.

The extended notation will be used increasingly. However, please refer in all cases to the manual before using the extended notation. It is also mandatory that original atom names are chosen such that no confusion can occur. A simple rule that avoids confusion, is not to use colons in atom names.

5 Before starting... ($PAWDIR and $ROOT)

In this manual, I frequently refer to $PAWDIR and $ROOT. They are explained in the following:

5.1 The PAW directory

$PAWDIR is the name of the directory where the PAW code is installed.

In the directory $PAWDIR you will find the following directories

- `src` contains the source codes. The sources for the analysis tools are located in a subdirectory `Tools`.
- `doc` contains the manual and installation notes
- `bin` will be created if it does not exist. It contains the executable codes
- `dx` contains files required by the dataexplorer to visualize wave functions, densities and atomic structures

If the program is set up properly, you will have an environment variable `$PAWDIR`, which refers to the directory where the PAW code is installed. You can try it with "`echo $PAWDIR`", which will print the PAW directory. Then you can go to the PAW directory using "`cd $PAWDIR`", and see its contents using "`ls -l`".

5.2 The project data structure

All files for a given system should be located in one directory, and it is recommended that this directory contain only that project. All filenames belonging to that system will have a common root, which we denote in a Unix-like fashion by $ROOT (the value of the variable "`ROOT`"). The individual files are distinguished by their extensions. For example, you can look into the directory "`$PAWDIR/Sample`". Here you will find a number of files that begin with "`h2co`". In this case, the root for the filenames ($ROOT) is "`$PAWDIR/Sample/h2co`". Note that te root is always specified by its absolute path. The filenames have extensions of the kind "`.cntl`", "`.strc`", "`.rstrt`", etc. Each extension defines a particular role of the file within the project. For example, in the file "`$ROOT.cntl`" one defines which operations the simulation code shall do, in the file "`$ROOT.strc`" one defines the molecule or crystal to be studied, and the file "`$ROOT.rstrt`" holds, among other data, the instantaneous atomic coordinates and electron wave functions that the simulation code needs to know in order to continue the simulation.

You can make exceptions to this rule and define the filenames explicitly. This may be useful, for example, if you wish to store the bulkiest file elsewhere, because you have run out of space on the disk, where you store the projects.

Note that, unlike $PAWDIR, your environment does not know what $ROOT is. Therefore, you should use $ROOT in this manual as a placeholder for the full root name, unless the variable "`ROOT`" is explicitly defined.

6 The simulation code

6.1 How to perform a simulation

6.1.1 Input files

In order to run the simulation code, two input files have to be prepared:

1. The **structure input file** (or often simply called STRC), describes the system to be studied. Here you provide information about which atoms you wish to simulate, what their specifics are, whether you wish to simulate a molecule or a crystal and so on.

2. The **control input file** (sometimes abbreviated as CNTL) describes what the program shall do, such as optimizing the wave functions, relaxing the atoms or simlating at finite temperature.

3. The **setup files** contain element-specific information about the augmentation, i.e. partial waves and projector functions, and the core density. Currently, these files are supplied by the author. The setups depend via the core density on the functional used in their construction. It is also important to be consistent regarding the division of electrons into core and valence electrons.

After the simlation has finished, the program writes a restart file, which holds the actual wave functions and the positions of the atoms. This file holds all the necessary information to continue a simulation from the point where you finished. It is important to keep the restart file and the structure file if you wish to return later to what you have done.

6.1.2 Execute the simulation code

The simulation code is executed using the command

`paw_fast.x` *controlfile* `1>err 2>&1 &`

where *controlfile* is the name of the control input file described in the next section. With `1>err` one redirects the print statements to the file out. If the file existed before it is overwritten. This information is normally irrelevent and is used for development purposes. However, it is crucial for tracing errors. With `2>&1` one

redirects also error messages into the same file, namely `err`. Thus also error messages be it from PAW or from the system are written in the same directory. The last & simply sends the job into the background. That is you can continue working in the same window while the job is running.

`paw_fast.x` is simply the name of the PAW executable. There will be several executables. If you are using a parallel computer using MPI you will need an executable that starts with `ppaw` instead of `paw`. The executable with `fast` is the one compiled with all reasonable optimization flags for the compiler. For all normal purposes this one should be used. In addition there may be an executable with `dbg` instead of `fast` for debugging purposes and one with nothing, with no special flags, which can be used to find errors induced by optimization.

You can also obtain information about the executable

`--help`	print help information
`-h`	print help information
`?`	print help information
`--version`	print version information
`--parmfile`	print parameter file used for compilation

The version information shall allow to correlation the executable to a well-defined version of the source code. The parameter file in addition provides information about what libraries are linked and which compilation flags have been used etc. This information is important to track bugs in the code.

Sometimes it is convenient to write a little wrapper shell-script for the execution command.

```
#!/bin/sh
#=================================================
# sample doit file to execute the simulation code ==
#=================================================
#                                  define the rootname
ROOT=/home/user/Tree/Testrun/H2CO/h2co
#                                execute paw simulation
paw_fast.x ${ROOT}.cntl 1>${ROOT}.err 2>&1
```

This file would be called `$ROOT.doit`, which is also the command to execute the code. Such a wrapper avoids some unnecessary typing if you run the job repeatedly. It can easily be modified to execute the simulation code several times with different control files.

What does the wrapper do? Let us discuss it line by line. (Comment lines, i.e. those starting with a hatch sign, are not discussed.)

1. The first line `#!/bin/sh` simply specifies that the wrapper is a shell script

2. The second line defines a variable `ROOT`. In the following commands $ROOT is shorthand for `/home/user/Tree/Testrun/H2CO/h2co`. Thus one can use the same wrapper for different simulations by simply changing the value of `ROOT`.

   The last line executes the the simulation code using the control file `$ROOT.cntl`.

In order to track the simulation one can issue the command

```
tail -f $ROOT.prot
```

which continually prints the lines written to the protocoll file also to the screen. Another way to trace the simulation is the command `paw_show` described in section 16.1.

6.1.3 Terminate the simulation

The execution can be stopped before regular completion by creating a so-called exit file

`touch` *exitfile*

where *exitfile* is the name of the exit file. The standard name is $ROOT`.exit`, but this name can be changed in the control input file.

As an alternative to setting the exit file, the CP-PAW code also contains a signal trap. The command

`kill -30` *PID*

where *PID* is the process id number, causes the program to terminate the execution after the next iteration. Note, however, that this command can sometimes crash the code, namely if the code is writing to a file at the moment you issue the command. Its use is therefore not recommended, and it may even be removed from future versions. The process id number can be obtained either from the command `top` or from the command `ps -elf |grep paw`.

6.2 The control input file “CNTL”

The control file is responsible for “what to do” in the simulations. These data are generally unspecific to the system.

6.2.1 Examples for the control input file “CNTL”

The following is a particularly simple example for a control input file. This file can be used for getting started.

```
!CONTROL
  !GENERIC START=T NSTEP=100 !END
  !FOURIER EPWPSI=30 CDUAL=2 !END
  !PSIDYN
    !AUTO FRIC(+)=0.3 FACT(+)=1.
          FRIC(-)=0.3 FACT(-)=0.97 !END
  !END
!END
!EOB
```

The following example illustrates a few more options that give a quick impression of the choices available. The detailed description of all options is given in the next section.

```
!CONTROL
 !GENERIC START=T DT=10 NSTEP=100 NWRITE=100 !END
 !DFT     TYPE=2 !END
 !FOURIER EPWPSI=30 CDUAL=2 !END
 !PSIDYN  STOP=T FRIC=0.005 RANDOM=0.1
          MPSI=1000 MPSICG2=0.5
   !AUTO  FRIC(-)=0.3 FACT(-)=0.97
          FRIC(+)=0.5 FACT(+)=0.97 !END
   !THERMOSTAT_OFF STOP=T FRIC=0
          T[K]=100 FREQ[THZ]=10 !END
 !END
 !RDYN    STOP=T FRIC=0 RANDOM[K]=0
   !AUTO  FRIC(+)=0.5 FACT(+)=0.97
          FRIC(-)=0.3 FACT(-)=0.97 !END
   !THERMOSTAT_OFF STOP=T FRIC=0
```

```
          <EKIN>=0.02 FREQ[THZ]=50 !END
 !END
 !FILES
   !FILE  ID='EXIT' EXT=F NAME='~/EXIT' !END
 !END
 !ANALYSE
   !HYPERFINE ATOM='H_1' EFG=T ISOMERSHIFT=T
              FERMICONTACT=T ANISOTROPIC=T   !END
   !DENSITY FILE='./density.ev' TYPE='TOTAL'
         DR=0.4 OCC=T DIAG=T CORE=F !END
   !WAVE FILE='./wavefunction.wv'
         DR=0.4 B=10 K=1 S=1 IMAG=F !END
   !POTENTIAL FILE='./potential.wv'
         dr=0.4 !END
 !END
!END
!EOB
```

6.2.2 Argument keywords for the control input file "CNTL"

6.2.3 !CONTROL

Rules: optional
Description: defines the operations performed on the system; largely independent of the system

6.2.4 !CONTROL!FILES

Rules: optional
Description: specifies the file names that deviate from the standard values

ROOT

rootname. Files defined as extension will have this name combined with the extension. All files connected as default are defined as extensions.

Type:	character
Rules:	optional
Default:	string preceding the '.cntl' ending of the control input file

6.2.5 !CONTROL!FILES!FILE

Rules:	optional, multiple
Description:	Specifies one file

ID

identifier for the file; options are:

'PROT'	protocol; extension: `.prot` monitors the simulation.
'STRC'	structure input file; extension: `.strc` defines atoms, structure, electron occupations etc.
'CNTL'	control input file; extension: `.cntl` used to control the simulation.
'RESTART_IN'	input restart file; extension `.rstrt` instantaneous coordinates
'RESTART_OUT'	output restart file; extension `.rstrt` see 'RESTART_IN'
'EXIT'	exit file; extension: `.exit` simulation terminates if exit file exists
'PDOS'	projected density of states; extension: `.pdos` used by the paw_dos tool.
'CONSTRAINTS'	constraint report; extension: `_constr.report`
'POSITION_TRAJECTORY'	atomic positions; extension `_r.tra` trajectory used by the paw_tra tool.
'BANDS_TRAJECTORY'	one-particle energies; extension `_r.tra` (not yet used)

Type: character
Rules: mandatory
Default: none

NAME

filename. Can be the relative file name or an extension to the PAW "root". Standard output can be specified by NAME='stdout' and EXT=.false.

Type: character
Rules: mandatory
Default: none

EXT

.true.: NAME specifies the extension only/ .false.: full name
Type: logical
Rules: optional
Default: .false.

6.2.6 !CONTROL!GENERIC

Rules: optional
Description: general data that do not fit into other blocks

START

T: start with random wave functions, and atomic positions from file ”STRC”
F: wave functions and atomic positions are taken from restart file, unless specified otherwise
Type: logical
Rules: optional
Default: F

NSTEP

number of time steps
Type: integer
Rules: optional
Default: 100

DT

time step Δ in a.u. (1 a.u.$\approx$ 0.024 fs)
Type: real
Rules: optional
Default: 10.0

NWRITE

Every “NWRITE” time steps, the program writes detailed information into the protocoll file and it updates the restart file. Note that writing the restart file is time consuming.

Type: integer
Rules: optional
Default: 100

TRACE

provides trace information when entering or leaving subroutines under trace control. Used for debugging purposes.

Type: logical
Rules: optional
Default: .false.

ENDIAN

defines whether unformatted files are written in little endian as typical on intel computers or in big endian as on ibm computers. The value can be ’little’,’intel’,’big’,’ibm’. The first two values have identical meaning and the last two have identical meaning. It is only functional using certain compilers (absoft).

Type: character
Rules: optional
Default: little

RUNTIME

A soft stop is initialized after the given time has elapsed since start of the code. Runtime is specified as a three element vector containing (hours,minutes,seconds). The three elements of the vector are internally converted into seconds and added up.

Type: integer
Rules: optional
Default: ∞

AUTOCONV

The autopilot defines a strategy to vary the friction for various dynamical variables, to test the convergence and to terminate the optimization loop. The decision to terminate is taken if the energy remains within an energy window of 10^{-5} H for autoconv time steps.

Type: integer
Rules: optional
Default: 20

RSTRTTYPE

allows to cut down the information on the restart file. Option RSTRTTYPE='STATIC' only stores one set of wave functions. This reduces the amount of diskspace by nearly a factor of two. This option should not be used in a dynamical simulation, because, the velocity of the wave functions will be set to zero.

Type: character
Rules: optional
Default: NONE

6.2.7 !CONTROL!DFT

Rules: optional
Description: selects density functional parameterization; default is Perdew-Zunger parameterization of the Ceperley Alder quantum Monte Carlo calculation

TYPE

density functional parameterization. possible values are:

(1) Perdew-Zunger parameterization [16] of Ceperley-Alder [17];
(2) Perdew-Wang 91 parameterization [18] of Ceperley-Alder [17];
(3) Local exchange (X_α with α=2/3);
(4) X-alpha (alpha=0.7);
(6) Local exchange and Becke gradient correction [19] for exchange;
(7) Perdew-Wang LSD [18] and Becke gradient correction for exchange [19];
(71) Perdew-Zunger[16] and Becke-88 gradient correction[19];
(8) like (7) + Perdew-86 gradient correction for correlation [20];
(81) Perdew-Zunger[16] + Becke gradient correction[19] + Perdew86 gradient correction for correlation[20];
(9) like (7) + Perdew-Burke Ernzerhof correlation [21];
(10) Perdew-Burke-Ernzerhof (PBE)exchange and correlation [21]; Almost identical to PW91 GGA[22] but simpler formulation.
(11) RPBE exchange and correlation, includes Hammer's correction[23] to Perdew-Burke-Ernzerhof exchange and correlation [21];

Type: integer
Rules: optional
Default: 1

VDW

Adds van der Waals Interactions[24]. The van der Waals interaction is described by an interatomic pair-potential that has a long-ranged r^{-6} behavior and which is cut of smoothly at short distances.

Type: logical
Rules: optional (Untested)
Default: F

6.2.8 !CONTROL!FOURIER

Rules: optional

Description: Plane wave cutoffs $E_{PW} = \frac{1}{2}G^2_{max}$ for wave functions and charge density. A cutoff of 30 Ry=15 H for the wave function and CDUAL=2 is sufficient for most applications. In order to account for all plane wave components in the density, the plane wave cutoff for the density should in prinicple be four times the cutoff for the wave function. (For low cutoffs, even CDUAL$>$4 may be required since the exchange and correlation functional is nonlinear, and therefore the energy may not necessarily decrease monotonically as the basis set size is increased with a fixed CDUAL. In practice this is rarely a problem.)

EPWPSI

plane wave cutoff for the wave functions in Rydberg (1 Ry=0.5 a.u.). All plane waves up to a maximum kinetic energy equal to the plane wave cutoff are considered. Note that the plane wave cutoff refers only to the plane wave part of the basis functions.

Type: real
Rules: optional
Default: 30.

EPWRHO

plane wave cutoff for the charge density in Rydberg (1 Ry=0.5 a.u.)

Type: real
Rules: optional
Default: 4*EPWPSI

CDUAL

EPWRHO=CDUAL*EPWPSI; The default cdual=4 produces the correct result; cdual=2 is often a good choice.

Type: real

Rules: optional; this value is overwritten by any occurance of “EPWRHO”
Default: 4

EPWBUCKET

Used mostly for cell dynamics. See section 21.7 for an explanation of the sawtooth behavior. Parameter E_B in Ry specified as follows: If specified an additial potential energy term $\sum_{G,n} f_n \langle \tilde{\Psi}_n | G \langle B(G) \langle G | \tilde{\Psi}_n \rangle$, is added top the total energy where $B(G) = c_B \theta(\frac{1}{2}G^2 - E_B)(G - \sqrt{2E_B})^2$. When this term is set to a fixed value, the plane wave convergence is artificially accelerated, which is important to evaluate stresses. The energy converged this way corresponds however not to equal to the energy converged without this term, but corresponds to the total energy obtained with a plane wave cutoff slightly higher than EPWBUCKET.
Type: real
Rules: optional;mandatory of BUCKETPAR is specified
Default: none

BUCKETPAR

Parameter c_B see also description of EPWBUCKET above.
Type: real
Rules: optional; mandatory, if EPWBUCKET is specified
Default: 0.

6.2.9 !CONTROL!PSIDYN

Rules: optional; default uses default values for the electron dynamics.
Description: Parameters used to control the dynamics of the wave functions. The wave function dynamics is governed by the equation

$$m_\Psi|\ddot{\tilde{\Psi}}_n\rangle = -\frac{\partial E}{\partial\langle\tilde{\Psi}_n|} - m_\Psi|\dot{\tilde{\Psi}}_n\rangle\alpha - \sum_m |\tilde{\Psi}_m\rangle\Lambda_{mn},$$

where α can be tuned by a Nosé thermostat or the parameters below. The equation of motions are integrated using the Verlet algorithm [**?**]. The friction parameter α is converted into a parameter $c_\alpha = \alpha\Delta/2$, which can range from 0 to 1. A value of $c_\alpha = 0$ indicates frictionless dynamics, whereas a value of $c_\alpha = 1$ indicates steepest descent. Δ is the time step. A discussion of the virtues of friction dynamics can be found in Ref. [25]

STOP

if STOP=true start with zero velocity of the wave functions
Type: logical
Rules: optional
Default: F

MPSI

mass m^0_Ψ for the wave function dynamics. The wave function mass is an operator of the form

$$\hat{m}_\psi = \sum_{\vec{G}} |\vec{G}\rangle m^0_\psi (1 + c\vec{G}^2)\langle\vec{G}|$$

The coefficient c is set in MPSICG2.
The G-dependent wave function mass is used to reduce the otherwise very large frequency of the wave function components with large $|\vec{G}|$. For the large G-components of the auxiliary wave functions we can approximate the effective potential by a constant, which makes the Hamiltonian diagonal in $|\vec{G}\rangle$. This allows to estimate the frequency from

$$\hat{m}^0_\psi (1 + c\vec{G}^2)\ddot{\Psi}_G = \left[\frac{\vec{G}^2}{2} + v_{eff}\right]\Psi_G$$

Type: real
Rules: optional
Default: 10 Δ^2 (for Δ see "!CONTROL!GENERIC:DT")

MPSICG2

high-G enhancement c for the wavefunction mass. The fictitious electron mass for the wave function dynamics is G-dependent $m_\Psi(G) = m_\Psi(1 + cG^2)$. A recommended value is $c = 0.5$. A reasonable value is $\frac{1}{2}(\frac{x}{2\pi})^2\frac{\Delta^2}{m_\Psi}$, where x is the fraction of shortest oscillation period for the electron dynamics and the time step. The stability criterion of the Verlet algorithm requires $x > 3$. The default uses this expression with $x = 10$. For proper mass renormalization (discounting the nuclear mass by the effective mass of the wave function cloud) the values of !STRUCTURE!SPECIES:PS<G2> and !STRUCTURE!SPECIES:PS<G4> need to be specified.
Type: real
Rules: optional
Default: $\frac{1}{2}(\frac{10}{2\pi})^2\frac{\Delta^2}{m_\Psi}$

FRIC

constant friction c_α for the wave function dynamics
Type: real
Rules: optional, not used if "!CONTROL!PSIDYN!AUTO" is selected.
Default: 0.0

SAFEORTHO

chooses the way the orthogonality constraints for the wave functions is enforced. If SAFEORTHO=T is selected the traditional way is used, which results in a strictly energy-conserving dynamics. If SAFEORTHO=F a new method is used which converges at eigenstates of the Hamiltonian, but is energy conserving only if the states are sufficiently close to eigenstates. This option allows to calculate excited states or to use the Mermin functional[26] in order to introduce Fermi occupation numbers. It must be used in connection with dynamical occupations such as the Mermin functional and the tetrahedron method for Brillouin-zone integration.
Type: logical
Rules: optional
Default: T

SWAPSTATES

Only used with SAFEORTHO=F. For SWAPSTATES=F, the approximate eigenstates are ordered in increasing energy expectation values. This is used for optimizing electronic and atomic structures. Thus the given occupations refer to increasing one-particle energies in the final state.
If, in a photochemical reaction with a state crossing with fixed occupations, the system shall remain on the upper branch of the energy surface, choose SWAPSTATES=F. If the system shall follow the crossing to the lower branch of the energy surface, choose SWAPSTATES=T.
In combination with the Mermin functional the following happens at a band crossing at the Fermi level.

- SWAPSTATES=F: the wave functions change their character, which results in wave function heating. The time scale of the conversion of the state depends on matrix elements and on the fictitious mass of the wave functions. This process is therefore rather uncontrolled. The occupations remain approximately constant. The crossing is treated like an avoided crossing.

- SWAPSTATES=T: the wave function maintains its character, but the occupations dynamically change from occupied to unoccupied and vice versa. If the croosing is an avoided crossing, the wave functions are reordered only if the if the conversion of the character of the wave functions is sufficiently retarded, that a band crossing of the one-particle energies takes place.

Type: logical
Rules: optional, not tested
Default: F

6.2.10 !CONTROL!PSIDYN!AUTO

Rules: optional

Description: Automatic annealing procedure for wave function dynamics. (overwrites setting of !CONTROL!PSIDYN:FRIC.) For optimizing the electronic structure the default set has been proven very useful. The friction switches between a lower and an upper value depending on whether the energy decreases or increases. The upper and lower friction values are each scaled in each time step. With this option, the program will stop if the total energy changes during 20 iterations by less than 10^{-5} Hartree. This does not necessarily imply convergence of the annealing step. It can happen, in particular near convergence, that the friction jumps from step to step between the lower and the upper value. In this case the system is not converged, even if the total energy is virtually constant, and a constant friction should be used.

FRIC(+)

start value for the friction factor c_α used for increasing energy

Type: real
Rules: optional
Default: 0.3

FACT(+)

factor multiplied with the friction factor FRIC(+) in each step that reduces the energy

Type: real
Rules: optional
Default: 1.0

FRIC(-)

start value for the friction factor c_α used for decreasing energy

Type: real
Rules: optional
Default: 0.3

FACT(-)

factor multiplied with the friction factor FRIC(-) in each step that reduces the energy

Type: real
Rules: optional
Default: 0.97

MINFRIC

minimum friction used

Type: real
Rules: optional
Default: 0.

6.2.11 !CONTROL!PSIDYN!THERMOSTAT

Rules: optional

Description: (presently in test phase:
Thermostat for the wave functions. The thermostat acts like a Nose-Hoover thermostat[27, 28, 29] , but it does not aim at constructing a canonical ensemble. Instead it starts acting only when the wave function kinetic exceeds the target temperature estimated for Born-Oppenheimer kinetic energy.[30] The latter obtained from the instantaneous atomic velocities, the wave function masses and the values of !STRUCTURE!SPECIES:PS<G2> and !STRUCTURE!SPECIES:PS<G4> provided the structure input file. Do not use this thermostat without providing PS<G2> and PS<G4>. If the thermostat kicks in, it reshuffles kinetic energy from the wave functions into the atomic motion. The target kinetic energy is subtracted from the wave function kinetic energy in the protocol. A friction on the thermostat shall be used to avoid instabilities. The thermostat shall be used only if the atoms are moving.
Old version: Nosé thermostat for constant temperature ensemble for the electrons [31]. The target kinetic energy should be a little (about 50%) larger than

$$\langle E_{kin}^{\Psi} \rangle_T = \sum_R 2 \frac{m_\Psi}{M_R} \tilde{T}_R k_B T, \tag{1}$$

where $\tilde{T}_R = \sum_n \langle \tilde{\Psi}_n | - \nabla^2/2 | \tilde{\Psi}_n \rangle$ is the kinetic energy of the pseudo wave functions of the corresponding isolated atom and T is the physical temperature of the nuclei.

<**EKIN**>

average kinetic energy of the wave functions. Do not use in the new version.

Type: real
Rules: optional
Default: 0.01

T[K]

New version: atom temperature in Kelvin.
Old version: average kinetic energy of the wave functions divided by Boltzmann's constant.

Type: real
Rules: optional
Default: see <EKIN>

PERIOD

Oscillation period of the Nosé variable in a.u.

Type: real
Rules: optional
Default: see !CONTROL!PSIDYN!THERMOSTAT:FREQ[THz]

FREQ[THZ]

Frequency of the Nosé variable in THz.

Type: real
Rules: optional; not used if !CONTROL!PSIDYN!THERMOSTAT:PERIOD present
Default: 100.

FRIC

Friction acting on the Nosé variable. Can have values between zero and one. The largest value before overdamping is $\frac{4\pi}{period}$, where period is that of the thermostat variable defined in this section.

Type: real
Rules: optional
Default: 0.

STOP

set velocity for Nosé variable to zero in the first iteration

Type: logical
Rules: optional
Default: F

OLD

selects the “old” implementation of the Nose thermostat
Type: logical
Rules: optional
Default: F

6.2.12 !CONTROL!RDYN

Rules: optional; default is fixed atomic positions
Description: parameters used to control the dynamics of the atomic positions. The atomic dynamics is governed by the equation

$$M_R\ddot{R} = -\frac{\partial E}{\partial R} - M_R\dot{R}\alpha,$$

where α can be tuned by a Nosé thermostat or the parameters below. The friction parameter α is converted into a parameter $c_\alpha = \alpha\Delta/2$, which can range from 0 to 1. A value of $c_\alpha = 0$ indicates frictionless dynamics, whereas a value of $c_\alpha = 1$ indicates the steepest descent. Δ is the time step.
Remark: Do not use this option before the wave functions are optimized, because unreasonable forces will result from an incorrect electron distribution.

STOP

start with zero velocity for the atomic positions
Type: logical
Rules: optional
Default: F

START

Take initial structure from structure control file instead of the restart file.

Type: logical
Rules: optional
Default: F

RANDOM[K]

adds random velocities at the start; the velocity distribution corresponds to a temperature in Kelvin. Can be used to bring the system quickly to a finite temperature. Note, however, that the electron wave functions cannot immediately follow this sudden kick. They will start to lag behind, and deviate from the Born Oppenheimer surface. A small friction acting on the wave functions will allow the wave functions to recover.
Type: real
Rules: optional
Default: 0

FRIC

constant friction c_α
Type: real
Rules: optional, not used if "AUTO" is selected.
Default: 0.0

NONEGEFRIC

Experimental option! Avoids negative friction on the atoms. It is recommended for structure optimizations. It must not be used in connection with thermostats. The friction can become negative via the compensation for the drag by the wave function cloud. A friction on the wave functions implicitely causes an effective friction on the atoms. This effective friction is compensated by adding a negative friction on the atoms. In this case the total friction on some atoms can become negative so that the atoms cannot come to rest. [Future developents shall automatically exclude simultaneous use together with thermostats.]
Type: logical
Rules: optional
Default: false

USEOPTFRIC

Experimental option! Estimates the optimum friction for optimization of the atomic structure. as

$$a_{opt} = \Delta \sqrt{\frac{\dot{\vec{R}}\dot{\vec{F}}}{\dot{\vec{R}}\mathbf{M}\dot{\vec{R}}}}$$

In addition a floating average of this friction value is used. The formula is derived from a projection os the Lagrangian onto a one-dimensional motion and the assumption of a harmonic oscillator. The force constant can then be estimated from the change of the forces between two time steps. See Sec. 21.2 for further information.

Type: logical
Rules: optional
Default: false

6.2.13 !CONTROL!RDYN!AUTO

Rules: optional
Description: Automatic annealing procedure for atomic motion (overwrites setting of !CONTROL!RDYN:FRIC). See the description for !CONTROL!PSIDYN!AUTO. Note that frequent switching between the lower and upper friction indicates a problem that can even result in a heating up of the electrons. In this case, switch to constant friction.

FRIC(+)

start value for the friction factor c_α used for increasing energy

Type: real
Rules: optional
Default: 0.8

FACT(+)

factor multiplied with the friction factor FRIC(+) in each step that reduces the energy

Type: real
Rules: optional
Default: 1.0

FRIC(-)

start value for the friction factor c_α used for decreasing energy

Type: real
Rules: optional
Default: 0.3

FACT(-)

factor multiplied with the friction factor FRIC(-) in each step that reduces the energy

Type: real
Rules: optional
Default: 0.97

MINFRIC

minimum friction used

Type: real
Rules: optional
Default: 0.

6.2.14 !CONTROL!RDYN!WARMUP

Rules: optional, not tested
Description: Heating procedure for QM nuclei. Applies a series of sinusoidal heat pulses in orthogonal directions to provide optimal excitation of the atomic system without dislodging the wavefunctions from the BO surface. This is an approximate procedure.

TARGET_TEMP

Target temperature that should be achieved in Kelvin. Due to approximate nature of the algorithm, it will not be reached exactly in most cases.

Type: real
Rules: mandatory
Default: none

NPULSES

The required temperature increase specified in TARGET_TEMP will be applied in NPULSES steps.

Type: integer
Rules: mandatory
Default: none

PULSE_LENGTH

Each of the NPULSES heat pulses will stretch over PULSE_LENGTH time steps.

Type: integer
Rules: mandatory
Default: none

6.2.15 !CONTROL!RDYN!THERMOSTAT

Rules: optional

Description: Nosé thermostat [27, 28, 29] for creating a constant temperature ensemble for the ions. When using this thermostat, the friction acting indirectly on the atoms, and that results from a friction on the wavefunctions, is corrected for by an opposing force $C_i \dot{R}_i f_\Psi$ acting on the atoms. f_Ψ is the friction acting on the wave functions, which is either fixed or dynamically tuned by a wave function thermostat

This thermostat has the added option of a friction for faster equilibration. The friction must be zero to obtain a canonical ensemble.

T[K]

temperature of the ions in Kelvin
Type: real
Rules: optional
Default: 293.15 (=room temperature=20^{o}C)

<**EKIN**>

average kinetic energy of the atoms
Type: real
Rules: optional
Default: see T[K]

PERIOD

Oscillation period of the Nosé variable in a.u.
Type: real
Rules: optional
Default: see !CONTROL!RDYN!THERMOSTAT:FREQ[THz]

FREQ[THZ]

frequency of the Nosé variable in THz.
Type: real
Rules: optional; not used if !CONTROL!RDYN!THERMOSTAT:PERIOD is present
Default: 10.

FRIC

friction on the Nosé variable. Sensible values lie between 0 and 1; FRIC=0 will give the original Nosé thermostat.
Without friction, the thermostat tends to induce large oscillations when the temperature is changed. This oscillations may break your system in the high-temperature peaks, and they decrease only slowly to their physical amplitude. In this case it is recommended to use a friction of about 0.01 until the system is about stationary at the new temperature. Once the stationary state has been reached, the friction must be removed, so that the system can approach the canonical ensemple

Type: real
Rules: optional
Default: 0

STOP

velocity for Nosé variable is set to zero in the first iteration

Type: logical
Rules: optional
Default: F

6.2.16 !CONTROL!MERMIN

Rules: optional

Description: Describes the treatment of variable occupations of the one-electron energy levels. It allows to describe the occupations as dynamical variables in the framework of the Mermin functional[26], which includes finite temperature effects. The total energy in this case is the free energy and includes an entropic term $-TS$ for the partially occupied orbitals at finite temperatures. The occupations will converge to the Fermi distribution function: $f_i = (1 + e^{-(\epsilon_i - \mu)/kT})^{-1}$. The eigenvalues are the $\epsilon_n = \langle \Psi_n | H | \Psi_n \rangle$.
In the adiabatic mode of operation also the improved tetrahedron method [32]can be used, which is the recommended option for static ground state calculations of metallic systems.
Use this option (variable occupations) only with eigenstates of the Hamitlonian, i.e. with !CONTROL!PSIDYN:SAFEORTHO=.false. Otherwise the results are unpredictable.
It is important to initialize the occupations once using START=.TRUE.. (If occupations are close to zero or one, they will take a very long time to move away from these values.) if Start=.false., total charge and total spin is obtained from the restart file and not from the structure input file
Remark the dynocc object has been largely rewrittem 23-28.12.06. Please report any suspicious observations.

T[K]

Temperature of the electrons in Kelvin (enters the Fermi distribution function)
Type: real
Rules: optional
Default: 1000

M

mass for the occupation dynamics. Should be larger than 200.
Type: real
Rules: optional

Default: 300

ADIABATIC

chooses quasi-adiabatic mode for occupations. New energy levels are introduced that approach the energy levels $\epsilon = \frac{\partial E}{\partial f_n}$.in a retarded fashion. The occupations are determined for these retarded energies. Note that in the previous formula the energy also includes the fictitious kinetic energy. This mode of operation does not produce an energy conserving dynamics. It is recommended for static calculations. The current implementation of the tetrahedron method for Brillouin zone integration only works with adiabatic=T.

Type: logical
Rules: optional
Default: .false.

RETARD

The energy levels used for the Brillouin-zone integration in the adiabatic mode approach the actual enegy levels as $\sim \exp(-\frac{1}{r}\frac{t}{\Delta})$.

Type: real
Rules: optional, used only if !ADIABATIC=T
Default: 0.

TETRA+

if true, the Brillouin zone integration is performed with the improved version of the tetrahedron method[32]. If false, the Brillouin zone integration is done by sampling and the Mermin functional. The tetrahedron method is the recommended method for metallic systems. However it is not fully variational and restricted to the quasi-adiabatic mode. The tetrahedron method usually causes problems when only 1 k-point is used, because this results in delta-function peaks of the density of states. In that case it is better to use TETRA=false and a finite temperature.

Type: logical
Rules: optional, requires ADIABATIC=T
Default: .false.

STOP

set velocity for occupation dynamics to zero in the first iteration

Type: logical
Rules: optional
Default: F

STARTTYPE

defines how the initial occupations are determined.

'X': Occupations are read from restart file.

'E': energies are read from restart file. Occupations are constructed using the fermi distribution function.

'N': States are filled with $T = 0$ assuming absolutely flat bands ordered according to increasing energy.

Type: character
Rules: optional; 'X' incompatible with ADIABATIC=T
Default: 'N'

START

Restart occupations, total spin and total charge from Lagrange multipliers for wave function orthogonalization, which approximate the one-particle energies. If start=.false., the information is read from the restart file.

Type: logical
Rules: optional, not ised if !CONTROL!MERMIN:STARTTYPE is set.
Default: F

MOVE

Propagate occupations. Otherwise the occupations are kept frozen.

Type: logical
Rules: optional
Default: T

FRIC

friction for the occupation dynamics
Type: real
Rules: optional
Default: 0.0

FIXQ

conserve total charge
Type: logical
Rules: optional
Default: .true.

FIXS

conserve total spin
Type: logical
Rules: optional
Default: .false.

EFERMI[EV]

chemical potential of the elctrons in eV; only used if fixq=.false.
Type: real
Rules: optional
Default: 0.0

MAGFIELD[EV]

external magnetic field in eV; only used if fixs=.false.
Type: real
Rules: optional
Default: 0.0

6.2.17 !CONTROL!MERMIN!DIAL

Rules: optional
Description: allows to linearly vary the thermodynamic variables of the Mermin functional such as temperature, (not implemented: total charge,total spin, chemical potential, magnetic field). The initial Value is taken from the actual setting of the object

NAME

Specifies the variable to be modified. Can be TEMPERATURE[K]; CHARGE[E]; SPIN[HBAR].
Type: character
Rules: mandatory
Default: none

RATE

Specifies the rate of change of the variabble. The unit of the variable is that indicated by the name. The time scale unit is a.u.
Type: real
Rules: mandatory
Default: none

FINAL

Specifies the final value of the modified quantity. After this value is reached, the quantity remains constant.
Type: real
Rules: mandatory
Default: none

6.2.18 !CONTROL!CELL

Rules: optional; default is fixed unit cell, and no stress calculation
Description: This option is experimental!. Invokes pressure calculation, and Parrinello-Rahman dynamics of the unit cell, if requested.

MOVE

Dynamical evulution of the unit cell.
Type: logical
Rules: optional
Default: T

STOP

reset velocities to zero
Type: logical
Rules: optional
Default: F

FRIC

friction parameter
Type: real
Rules: mandatory if !control!cell:move=.true.
Default: none

M

mass for unit cell dynamics
Type: real
Rules: mandatory if !control!cell:move=.true.
Default: none

P

external pressure
Type: real
Rules: optional
Default: 0.0

CONSTRAINTTYPE

Constraints on the cell dynamics. Allowed values are the following:

- 'ISOTROPIC' only allows isotropic expansions and contractions.
- 'NOSHEAR' allows anisotropic expansions and contractions but no shearing.
- 'FREE' does not restrict the dynamics other than to remove the rotation from the stress tensor.
- 'NOSTRESS' sets the stresses to zero. (Only used for tests).

Type: character
Rules: optional
Default: 'FREE'

6.2.19 !CONTROL!QM-MM

Rules: optional
Description: controls parameters for the dynamics of the molecular mechanics environment specified in the structure file.
Remark: Switch this option on only after the wave functions are optimized, because unreasonable electrostatic forces will act on the environment atoms if the electron density is incorrect

STOP

sets initial velocities of the environment to zero.
Type: logical
Rules: optional
Default: false

FREEZE

keeps environment atoms except link atoms frozen

Type: logical
Rules: optional
Default: false

ADIABATIC

Relaxes the environment in every time step to zero (in contrast to damped or undamped dynamics)
Type: logical
Rules: optional
Default: true

FRIC

Friction (FRIC=1 corresponds to steepest decent; FRIC=0 to zero friction)
Type: real
Rules: optional
Default: 0.0

RANDOM[K]

randomizes the atomic positions (not yet implemented)
Type: real
Rules: optional
Default: 0.0

MULTIPLE

makes environment multiple times faster (multiple time steps). For very large values the QM reaction center experiences free energy forces from the environment.
Type: integer
Rules: optional
Default: 1

6.2.20 !CONTROL!QM-MM!AUTO

Rules: optional
Description: Automatic annealing procedure for the dynamics of the MM-environment. See also remarks for !CONTROL!PSIDYN!AUTO

FRIC(+)

start value for the friction factor c_α used for increasing energy
Type: real
Rules: optional
Default: 0.3

FACT(+)

factor multiplied with the friction factor FRIC(+) in each step that reduces the energy
Type: real
Rules: optional
Default: 1.0

FRIC(-)

start value for the friction factor c_α used for decreasing energy
Type: real
Rules: optional
Default: 0.3

FACT(-)

factor multiplied with the friction factor FRIC(-) in each step that reduces the energy
Type: real
Rules: optional
Default: 0.97

MINFRIC

minimum friction used
Type: real
Rules: optional
Default: 0.

6.2.21 !CONTROL!QM-MM!THERMOSTAT

Rules: optional
Description: Nosé thermostat [27, 28, 29] to for constant temperature ensemble for the QM-MM environment atoms. It has the added option of a friction for faster equilibration and faster acceleration from small velocities. To regain the original thermostat leave the default FRIC=0.

T[K]

temperature of the ions in Kelvin
Type: real
Rules: optional
Default: 293.15 (=room temperature=20^{o}C)

<**EKIN**>

average kinetic energy
Type: real
Rules: optional
Default: see T[K]

FREQ[THZ]

frequency of the Nosé variable in THz. This is an alternative to !CONTROL!RNOSE:PERIOD
Type: real
Rules: optional; is overwritten by PERIOD
Default: see PERIOD

FRIC

friction on the Nosé variable. Sensible values lie between 0 and 1; FRIC=0 will give the original Nosé thermostat, while FRIC=1 is used for equilibration and does not produce a canonical ensemble.
Type: real
Rules: optional
Default: 0

STOP

velocity for Nosé variable is set to zero in the first iteration
Type: logical
Rules: optional
Default: F

6.2.22 !CONTROL!COSMO

Rules: optional
Description: controls parameters for the dynamics of the continuum description of solvent. Essentially identical to the COSMO method by Klamt and Schuurmann, with modifications that allow consistent forces[**?**].

STOP

sets initial velocities of the environment to zero
Type: logical
Rules: optional
Default: false

START

restarts screening charges with value zero
Type: logical
Rules: optional
Default: false

M

Mass for the surface charge dynamics
Type: real
Rules: optional
Default: 1000.

MULTIPLE

Number of time steps of the surface charge dynamics per PAW timestep
Type: integer
Rules: optional
Default: 1

ADIABATIC

chooses optimizations of screening charges in each time step
Type: logical
Rules: optional
Default: false

ETOL

energy convergence criterion for ADIABATIC=T. Inner loop terminates, if estimated energy deviation from minimum is smaller than etol.
Type: logical
Rules: if ADIABATIC=T either ETOL or QTOL or both must be specified. Not used for ADIABATIC=F.
Default: 10^{-7} if qtol is not specified

QTOL

charge convergence criterion for ADIABATIC=T. Inner loop terminates, if estimated charge deviation from minimum , i.e. $|\vec{q} - \vec{q}_{min}|$ is smaller than etol.

Type: logical
Rules: if ADIABATIC=T either ETOL or QTOL or both must be specified. Not used for ADIABATIC=F.
Default: charge criterion is not used

FRIC

friction for the dynamics of the screening charges
Type: real
Rules: optional
Default: 0.

OPTFRIC

If true, the optimum friction scheme used for the relaxation of the screening charges. See Sec. 21.2 for further information. Is only used in combination with option ADIABATIC.
Type: logical
Rules: optional, only used in combination with ADIABATIC=T
Default: .false.

RETARD

Only used in combination with OPTFRIC=T. The running average of the optimum friction scheme adjusts within RETARD time steps to one-half of the original deviation.
Type: real
Rules: optional, only used in combination with OPTFRIC=T
Default: 0.d0

6.2.23 !CONTROL!ANALYSE

Rules: optional
Description: Prepares special data for analysis

OPTIC

This option is currently disconnected. Writes data required for the optic package of M. Alouani. Attention! Many files are written starting with extension ".optics...". Does not work on parallel computers, and not for non-collinear spin density.

Type: logical
Rules: optional, currently disconnected
Default: false

6.2.24 !CONTROL!ANALYSE!TRA

Rules: optional
Description: select trajectories to be written.

R

positions an point charges.

Type: logical
Rules: optional
Default: .true.

FORCE

forces acting on the atoms.

Type: logical
Rules: optional
Default: .false.

E

total energy contributions.

Type: logical
Rules: optional
Default: .false.

BANDS

one-particle energies.(not yet implemented)
Type: logical
Rules: optional
Default: .false.

NSKIP

number of time slices to be skipped between two written ones.
Type: integer
Rules: optional
Default: 0

6.2.25 !CONTROL!ANALYSE!HYPERFINE

Rules: optional,multiple
Description: calculates hyperfine parameters for a selected atom. Note that hyperfine parameters are extremely sensitive quantities, and therefore plane wave convergence should be checked carefully. In some cases it is also necessary to use more partial waves in the augmentation than normally used for total energy calculations.
The magnetic hyperfine field B_i^N describes the magnetic field produced at the nuclear site by the electron spin distribution. The total hyperfine field depends on the direction of the electron spin, which is imposed by the external magnetic field. Hence the anisotropic hyperfine field is provided as a matrix that needs to be multiplied by a normalized vector pointing in the direction of the external magnetic field to give the contribution of the anisotropic part to the total hyperfine field. The total hyperfine field is obtained by adding the Fermi-contact field and the anisotropic hyperfine field. The hyperfine splitting ΔE is obtained by multiplication of the total hyperfine field B_i^N with $2\hbar\frac{m^N}{S_N}$, where m^N is the nuclear magnetic moment and S^N is the nuclear spin.

ATOM

atom name consistent with structure input file
Type: character
Rules: mandatory
Default: none

EFG

select electric field gradient calculation. Result will be documented in the protocol. The electric field gradient is related to the second derivatives of the potential at the nuclear site
Type: logical
Rules: optional
Default: .false.

ISOMERSHIFT

select isomer shift calculation. Result will be documented in the protocol. The isomer shift is related to the charge density at the nuclear site.
Type: logical
Rules: optional
Default: .false.

FERMICONTACT

calculates the Fermi contact field. It is the magnetic field acting on the nucleusResult will be documented in the protocol. The Fermi contact term is related to the spin density at the nuclear site.
Type: logical
Rules: optional
Default: .false.

ANISOTROPIC

selects anisotropic hyperfine parameter calculation. Result will be documented in the protocol. The anisotropic hyperfine parameter is related to the $\ell = 2$ component of the spin density near the nucleus

Type: logical
Rules: optional
¡ Default: .false.

6.2.26 !CONTROL!ANALYSE!CORELEVELS

Rules: optional, not tested
Description: calculates the positions of the core levels (absolute and relative to the isolated atom (atomic calculation)). Note that core relaxation effects, which contribute typically to 15 %, are not included. The absolute position of the core levels is only meaningfull for isolated molecules. For extended systems, the zero of the electrostatic potential is not the vacuum level, so that only relative shifts within a single calculation are meaningful. For Tests see Pasquarello et al., Phys. Rev. B 53, 10942 (1996).
Note does not work yet for parallel execution. Minor changes are required

DEFAULT

Selects if core-levels shall be calculated for atoms, which are not in the list of selected atoms.(see below).

Type: logical
Rules: optional
Default: .false.

ATOMS

List of atoms, for which the core levels shall be calculated or avoided. If default=F the core levels of specified are calculated. If default=T, the core levels of all atoms not specified are calculated.

Type: character,array of arbitrary length

Rules: optional
Default: none

6.2.27 !CONTROL!ANALYSE!WAVE

Rules: optional,multiple
Description: Writes a wave function. The file created can then be processed by the paw_wave tool to produce an input file for OPENDX[2]

TITLE

title of the image. Currently it is not used other than in the print-out.
Type: character
Rules: optional
Default: none

FILE

full file name of the file to be produced, which can be converted into a input file for OPENDX[2] using the paw_wave tool.
Type: character
Rules: mandatory
Default: none

DR

grid spacing for the output file. rounded to get an integer factor relative to the real-space grid used in the calculation.
Type: real
Rules: optional
Default: 0.4

B

band index
Type: integer

Rules: mandatory
Default: none

K

k-point index
Type: integer
Rules: mandatory
Default: 1

S

spin index
Type: integer
Rules: mandatory
Default: 1

IMAG

use imaginary part
Type: logical
Rules: optional
Default: F

6.2.28 !CONTROL!ANALYSE!DENSITY

Rules: optional,multiple
Description: Writes a density. The file created can then be processed by the paw_wave tool to produce an input file for OPENDX[2].

TITLE

title of the image. Currently it is not used other than in the print-out.
Type: character
Rules: optional
Default: none

FILE

full file name of the file to be produced, which can be converted into a input file for OPENDX[2] using the paw_wave tool.
Type: character
Rules: mandatory
Default: none

DR

grid spacing for the output file. rounded to get an integer factor relative to the real-space grid used in the calculation.
Type: real
Rules: optional
Default: 0.4

TYPE

can be 'TOTAL', 'SPIN', 'UP' or 'DOWN'. Determines the weights of the states in the density plots. 'TOTAL' takes the actual occupations and k-point or uniform weights (depending on 'OCC'). 'SPIN' is like 'TOTAL', but counts states for spin=2 negative. 'UP' and 'DOWN' give the spin up and down densities respectively.
Type: character
Rules: optional
Default: 'TOTAL'

OCC

use acutal occupations
Type: logical
Rules: optional
Default: T

DIAG

use eigenstates in the subspace of the dynamic wave functions.
Type: logical
Rules: optional
Default: T

CORE

include core density.
Type: logical
Rules: optional
Default: F

EMIN[EV]

lowest eigenenergy to be included.
Type: real
Rules: optional
Default: -1^{-10}

EMAX[EV]

highest eigenenergy to be included.
Type: real
Rules: optional
Default: 1^{-10}

BMIN

lowest band to be included.
Type: integer
Rules: optional
Default: 1

BMAX

highest band to be included.
Type: integer
Rules: optional

Default: 10000000

6.2.29 !CONTROL!ANALYSE!POTENTIAL

Rules: optional
Description: Writes the hartree potential. The file created can then be processed by the paw_wave tool to produce an input file for OPENDX[2].

TITLE

title of the image. Currently it is not used other than in the print-out.
Type: character
Rules: optional
Default: none

FILE

full file name of the file to be produced, which can be converted into a input file for OPENDX[2] using the paw_wave tool.
Type: character
Rules: mandatory
Default: none

DR

grid spacing for the output file. It is rounded to get an integer factor relative to the real-space grid used in the calculation.
Type: real
Rules: optional
Default: 0.4

6.3 The structure input file “STRC”

The structure input file defines the molecule or crystal to be studied.

6.3.1 Example for the structure input file "STRC"

This is an example of a structure input file for formaldehyde in a fcc supercell with lattice constant of 10 Å.

```
!STRUCTURE
  !GENERIC LUNIT=1.8897259926 !END
  !OCCUPATIONS NBAND=8 NSPIN=1 !END
  !LATTICE T= 0.00000  5.00000  5.00000
              5.00000  0.00000  5.00000
              5.00000  5.00000  0.00000 !END
  !SPECIES NAME= 'H_' ZV=1. M=1.9380
           FILE='/u/blo.2/PAW/Setups/BP86/001H/h1.out'  !END
  !SPECIES NAME= 'C_' ZV=4. M=18.9984
           FILE='/u/blo.2/PAW/Setups/BP86/006C/c1.out'  !END
  !SPECIES NAME= 'O_' ZV=6. M=15.99946
           FILE='/u/blo.2/PAW/Setups/BP86/008O/o1.out'  !END
  !ATOM    NAME= 'H_1'  R= -0.50000  0.00000 -0.86600 !END
  !ATOM    NAME= 'H_2'  R= -0.50000  0.00000  0.86600 !END
  !ATOM    NAME= 'C_3'  R=  0.00000  0.00000  0.00000 !END
  !ATOM    NAME= 'O_4'  R=  1.50000  0.00000  0.00000 !END
  !ISOLATE NF=3 RC=0.5 RCFAC=1.5 GMAX2=3.0 DECOUPLE=T !END
!END
!EOB
```

The following is not a working example. It is aimed at illustrating the syntax of the input file. It is more complex than a typical case because it incorporates (almost) all options at the same time.

```
!STRUCTURE
  !GENERIC LUNIT=1.0 !END
  !LATTICE T=0. 10. 10. 10. 0. 10. 10. 10. 0. !END
  !KPOINTS K= 0 0 0 !END
  !SPECIES  NAME='H_' ZV=1.  M=1.008 PSEKIN=0.0
           FILE=~/PAW/Setups/006H/h1.out !END
  !SPECIES NAME='C_' ZV=4. M=12.011
           FILE=~/PAW/Setups/006C/c1.out !END
  !ATOM NAME='alpha-hydrogen'
                  R= 1.0 1.0 1.0 SP='H_' !END
```

```
  !ATOM NAME='H_2' R=-1.0000 -1.0000  1.0000 !END
  !ATOM NAME='H_3' R=-1.0000  1.0000 -1.0000 !END
  !ATOM NAME='H_4' R= 1.0000 -1.0000 -1.0000 !END
  !ATOM NAME='C_1' R=0.0 0.0 0.0 M=5.0 !END
  !OCCUPATIONS
    NBAND=10 NSPIN=2 CHARGE[E]=0. SPIN[HBAR]=1.
    !STATE K=1 S=1 B=5 F=1.000 !END
    !STATE K=1 S=1 B=6 F=1.000 !END
  !END
  !ISOLATE NF=2 RC=1.0 RCFAC=1.5 GMAX2=3.
           DECOUPLE=T !END
  !GROUP NAME='BOND1'  ATOMS='C_1' 'H_1' !END
  !GROUP NAME='BOND2'  ATOMS='C_1' 'H_2' !END
  !CONSTRAINTS
    !RIGID GROUP='BOND1' !END
    !FREEZE GROUP='BOND2' !END
    !FREEZE ATOM='H_3'  !END
    !TRANSLATION GROUP='ALL' !END
    !TRANSLATION GROUP='BOND1' DIR= 1. 0. 0. !END
    !ORIENTATION  GROUP='BOND1' AXIS= 1. 0. 0. !END
    !LINEAR
      !LINE ATOM='H_1' VEC= 0.5 0.0  0.0 !END
      !LINE ATOM='H_2' VEC= 0.5 0.0 0.0 !END
    !END
  !END
!END
!EOB
```

6.3.2 Argument description for the structure input file "STRC"

6.3.3 !STRUCTURE

Rules: mandatory

Description: Defines the system to be studied. Specifies geometry, atom types, electronic occupations, atomic masses. etc...

6.3.4 !STRUCTURE!GENERIC

Rules: optional
Description: data that do not fit into other blocks.

LUNIT

specifies the length unit (in atomic units) for structural input in this file. All data are converted into atomic units (1 a.u.=0.529167$\times 10^{-10}$ m) by multiplication with LUNIT
Type: real
Rules: optional
Default: 1.0

6.3.5 !STRUCTURE!LATTICE

Rules: mandatory
Description: lattice translation vectors

T

Type: real(3,3)
Rules: mandatory
Lattice vectors in the order; (x,y,z of first vector; x,y,z of second vector...)
Default: none

6.3.6 !STRUCTURE!KPOINTS

Rules: optional
Description: specifies the k-point grid. Default is Γ-point sampling.

R

defines the density of k-points for the automatic generation of k-points. $\Delta k = 2\pi/R$.

Type: real
Rules: optional
Default: 12.

DIV

fractions of reciprocal lattice vectors defining the reciprocal sublattice for the k-points.If K is not specified, the k-points are equally spaced an set compatible with the inversion symmetry. Caution with non-orthogonal lattice vectors: $\vec{G}_1 = \frac{1}{V}\vec{T}_2 \times \vec{T}_3$, $\vec{G}_2 = \frac{1}{V}\vec{T}_3 \times \vec{T}_1$, $\vec{G}_3 = \frac{1}{V}\vec{T}_1 \times \vec{T}_2$, where $\vec{G}_i$ are the reciprocal space lattice vectors and $\vec{T}_i$ are the real space lattice vectors and $V = \vec{T}_1(\vec{T}_2 \times \vec{T}_3)$ is the volume of the real space unit cell.
Type: integer(3)
Rules: optional; mutually exclusive with R
Default: 2 2 2

SHIFT

shift is displaces the K-point grid such that the Γ-point is avoided. The three components of SHIFT specify the shift directions into the three lattice vectors. Each component may have the values 0 or 1. For zero, the grid is not shifted along the specified direction. For a value of 1, the grid is shifted by one-half of the grid spacing along the corresponding reciprocal lattice vector. While the density of the k-point grid is not affected by shifting the k-point grid, the k-point convergence is apparently faster. Shifting the grid avoids the high-symmetry points. Thus, one does not see the band edges, which usually lie on high symmetry points. The computational effort is not much affected by shifting the grid.
Type: integer(3)
Rules: optional
Default: 0 0 0

K

k-point coordinates (i_1, i_2, i_3) in units of fractions of reciprocal lattice vectors: $k_i = 1/2 \sum_j G_{ij} * i_j/n_j$, where G_{ij} is defined by the real space lattice vectors T_{ij} by $\sum_k G_{ki}T_{kj} = 2\pi\delta_{ij}$. The fractions are provided by `DIV`. The k-point integration weights are equally distributed. This option is not recommended for general use. The safe approach is to set either `R` or `DIV`. If `K` is not specified, the k-points are set equally spaced according to `R` or `div`

Type: integer(3)
Rules: multiple, optional
Default: 0 0 0

6.3.7 !STRUCTURE!SPECIES

Rules: Mandatory, multiple
Description: Specifies an atom type (Element).

NAME

Name of the atom type. It is recommended to begin the name with the two-character element symbol, where a whitespace as second character can be replaced by an underscore. This allows the program to select certain default values form the periodic table.

Type: Character
Rules: Mandatory
Default: None

ZV

Number of valence electrons

Type: real
Rules: Mandatory
Default: None

Z

Atomic number. This number is used only to set the default value for the mass.

Type: real
Rules: optional
Default: atomic number as deduced from !STRUCTURE!SPECIES:NAME, if the latter starts from an element symbol

M

Atomic mass in mass units (u=1822.89 a.u.=$\frac{1}{12}m(^{12}\mathrm{C}) \approx \mathrm{m}_p$).

Type: real
Rules: Optional, if either !STRUCTURE!SPECIES:Z is specified, or if !STRUCTURE!SPECIES:NAME begins with the element symbol
Default: Atomic mass, as obtained from !STRUCTURE!SPECIES:Z or, if not present, from !STRUCTURE!SPECIES:NAME

PSEKIN

[Do not use this option. It works but may be removed in later versions. Please use "PS⟨G2⟩" and "PS⟨G4⟩" instead]. Kinetic energy of the pseudo wave functions of the atom. This term is used to renormalize the atomic masses to account for the effective mass of the wave function cloud. When this term is set the estimate of the Born-Oppenheimer wave function kinetic energy is subtracted from the kinetic energy of the wave functions in the protocoll.

Type: real
Rules: Optional
Default: None

PS<G2>

Parameter used for determining the effective mass of the electron cloud at an atom.

$$\sum_n f_n \int d^3GG^2|\tilde{\Psi}_n^{at}(G)|^2$$

The value is twice the pseudo-kinetic-energy of the atom.
Type: real
Rules: Optional
Default: 0

PS<G4>

Parameter used for determining the effective mass of the electron cloud at an atom.

$$\sum_n f_n \int d^3GG^4|\tilde{\Psi}_n^{at}(G)|^2$$

This value is needed only in combination with a G-dependend electron mass.
Type: real
Rules: Optional
Default: 0

LRHOX

One center expansion for the density is limited to a maximum angular momentum of lrhox. If the value of LRHOX is larger $2\ell_{max}$, where ℓ_{max} is the maximum angular momentum in the set of partial waves, LRHOX is reset internally to $2\ell_{max}$.
Type: integer
Rules: Optional
Default: 2

NPRO

Array which determines the maximum number of projector functions per angular momentum – together with its all-electron and pseudo partial waves – used in the calculation. The maximum possible value is determined by the setup file. Each element refers to the main angular momentum ℓ, in increasing order. The length of the array determines the maximum angular momentum for the augmentation.

Type: integer array
Rules: Mandatory
Default: None

FILE

name of the file containing the partial waves and projector functions etc. for this atom.

Type: Character
Rules: Mandatory
Default: First two letters of the atom name

6.3.8 !STRUCTURE!SPECIES!LDAPLUSU

Rules: optional
Description: sets the parameters for lda+U calculation[33, 33].. Only used when !CONTROL!DFT:LDA+U=T.

NCORROFL

number of correlated orbital shells per angular momentum. The array elements correspond to s,p,d,f shells. For one shell of correlated d-orbitals use (0,0,1,0). The number of correlated orbitals is reduced to the number of partial waves for this angular momentum minus one. If there is only one partial wave, this one is used as correlated orbital. This is however not recommended.

Type: integer(*)
Rules: Mandatory
Default: none

RCUT

Correlated orbitals are constructed so that they have a note at the radius RCUT.

Type: real
Rules: optional
Default: covalent radius plus 15%

DIEL

relative dielectric constant. The interaction matrix elements are scaled with $1/\epsilon_r$.

Type: real
Rules: optional, incompatible with UPAR and JPAR
Default: none

UPAR[EV]

U-parameter for the orbital shell specified by MAINLN. The entire interaction matrix is scaled so that the U-parameter of the specified shell has the value UPAR. (The U-parameter is identical to the Slater integral F^0

Type: real
Rules: optional, incompatible with DIEL
Default: none

JPAR[EV]

J-parameter for the orbital shell specified by MAINLN. The Slater integrals $F^2, F^4, \ldots$ are rescaled. The U and J-parameters are defined as

$$\begin{aligned} U &= \frac{1}{(2\ell+1)^2} \sum_{m_1,m_2} U_{m_1,m_2,m_1,m_2} \\ J &= U - \frac{1}{2\ell(2\ell+1)} \sum_{m_1,m_2} (U_{m_1,m_2,m_1,m_2} - U_{m_1,m_2,m_2,m_1}) \end{aligned}$$

and the U-tensor matrix elements are defined from the correlated orbitals $\chi_i(\vec{r})$ as

$$U_{i,j,k,l} = \int d^3r \int d^3r' \, \frac{e^2 \chi_i^*(\vec{r}) \chi_j^*(\vec{r'}) \chi_k(\vec{r}) \chi_l(\vec{r'})}{4\pi\epsilon_0 |\vec{r} - \vec{r'}|}$$

This expression defines the default value.
Type: real
Rules: optional, incompatible with DIEL
Default: none

F4/F2

changes the Slater integral F^4 of the main shell relative to F^2. The Slater integrals are rescaled to leave the J-parameter untouched.
Type: real
Rules: optional, incompatible with DIEL
Default: none

F6/F2

Changes the Slater integral F^6 of the main shell relative to F^2. The Slater integrals are rescaled to leave the J-parameter untouched.
Type: real
Rules: optional, incompatible with DIEL
Default: none

MAINLN

Specify the main correlatd orbital shell by angular momentum ℓ (first value) and the index (second) of this shell. Example: MAINLN= 3 1 specifies the first correlated orbital shell in the f-angular momentum channel.

Type: integer(2)
Rules: optional, mandatory with UPAR and JPAR
Default: none

6.3.9 !STRUCTURE!SPECIES!hybrid

Rules: optional
Description: sets the parameters for lda+U calculation. Only used when !CONTROL!DFT:LDA+U=T. **Experimental option!**

NCORROFL

number of correlated orbital shells per angular momentum. The array elements correspond to s,p,d,f shells. For one shell of correlated d-orbitals use (0,0,1,0). The number of correlated orbitals is reduced to the number of partial waves for this angular momentum minus one. If there is only one partial wave, this one is used as correlated orbital. This is however not recommended.

Type: integer(*)
Rules: Mandatory
Default: none

RCUT

Correlated orbitals are constructed so that they have a note at the radius RCUT.

Type: real
Rules: optional
Default: covalent radius plus 15%

HFweight

weight of HF fock exchange.

Type: real
Rules: optional
Default: 1/4

6.3.10 !STRUCTURE!ATOM

Rules: mandatory, multiple
Description: Specifies one atom.

NAME

Atom name. If the Keyword “SP” is not specified, the first two characters have to be identical with the “NAME” of the corresponding species. It is recommended that the name start with the element symbol, with a underscore instead of a blank for one-letter symbols. All atoms must have different names.
Type: character
Rules: Mandatory
Default: None

R

Atomic coordinates in Cartesian coordinates and units of ”!STRUCTURE!GENERIC:LUNIT”. Note that the coordinates need not be updated, because the instantaneous coordinates are held in the restart file.
Type: real(3)
Rules: Mandatory; must follow atom name
Default: None

M

Mass of this atom. Overwrites value specified in block ”SPECIES”.
Type: real
Rules: optional

Default: Value specified for the corresponding species

SP

Name of the atom type (species) to which this atom belongs.
Type: character
Rules: optional
Default: First two letters of the atom name

6.3.11 !STRUCTURE!OCCUPATIONS

Rules: optional
Description: defines number of bands and the initial occupations of the one-particle states. The default occupations are chosen according to the assumption of zero temperature and that energies of the states increase with band index and that spin directions and k-points are degenerate. These default occupations can be changed, first by specifying the occupation of each state in !STRUCTURE!OCCUPATIONS!STATE and second when using the Mermin functional in !CONTROL!MERMIN.

NSPIN

Number of spin directions. (1 for spin-restricted calculations and 2 for spin-polarized calculations, 3 for noncollinear spins)
Type: integer
Rules: optional
Default: 1

NBAND

Number of bands. A band is the number of states per k-point, where a state can hold two electrons in a spin-restricted calculation and one electron otherwise.
Type: integer
Rules: optional
Default: calculated from the number of valence electrons and the highest specified band in block ”OCCUP”(ations).

EMPTY

Number of unoccupied bands. (see also keyword “NBAND”)
Type: integer
Rules: optional
Default: calculated from the number of valence electrons and the highest specified band in block ”OCCUP”(ations).

CHARGE[E]

Ionization state of the entire system. An electron has the charge CHARGE[E]=-1. May be overwritten by “!STRUCTURE!STATE”
Type: real
Rules: optional
Default: 0

SPIN[HBAR]

Total spin in $\hbar$. A single electron has SPIN[HBAR]=0.5 $\hbar$. May be overwritten by “!STRUCTURE!STATE”
Type: real
Rules: optional
Default: 0

6.3.12 !STRUCTURE!OCCUPATIONS!STATE

Rules: optional
Description: allows one to change the number of electrons for certain states from the standard occupation. Standard occupation: states are completely filled beginning with the first, until the system is neutral; the highest occupied state may be partially occupied; all k-points and spin directions are occupied equally. Note that the occupation must decrease monotonically with the band index, unless the option !CNTL!PSIDYN!SAFEORTHO=F is specified.

K

selects a k-point. Number refers to the order in which the k-points appeared.
Type: integer
Rules: optional
Default: all k-points are selected

S

selects a spin (1 or 2). S=2 is allowed only for spin polarized calculations (see !STRUCTURE!GENERIC)
Type: integer
Rules: optional
Default: all spin directions are selected

B

selects band
Type: integer
Rules: mandatory
Default: none

F

occupation
Type: real
Rules: mandatory
Default: none

6.3.13 !STRUCTURE!ISOLATE

Rules: optional
Description: Decouples the system electrostatically from its periodic images [11]. Recommended isolated molecules. Isolate is also required to calculate approximate point charges for the QM-MM coupling.

NF

Number of Gaussians used to fit the charge density. Should contain at least two.
Type: integer

Rules: optional
Default: 2

RC

Smallest decay constant for the Gaussians.
Type: real
Rules: optional
Default: 1.

RCFAC

Scaling factors for the decay constants. $r_c(i) = r_c(1) * \mathrm{RCFAC}^{i-1}$.
Type: real
Rules: optional
Default: 1.5

GMAX2

Cutoff for the plane waves of the pseudo charge density contributing to the fit
Type: real
Rules: optional
Default: 3.

DECOUPLE

switches electrostatic decoupling of periodic images on.
Type: logical
Rules: optional
Default: .true. (.false. if the block !STRUCTURE!ISOLATE is not present)

6.3.14 !STRUCTURE!CONFINE

Rules: optional

Description: Places a confining potential around the molecule which mimics the Pauli repulsion of a solvent. The potential starts to raise at a radius $r_{in,R}$, chosen equal to the van der Waals radius of the respective atom and reaches the maximum value V_0 at a rsadius $r_{out,R}$ equal to twice the Van der Waals radius. The confining potential has the form

$$v(\vec{r}) = V_0 \prod_R f\left(\frac{|\vec{r} - \vec{R}| - r_{in,R}}{r_{out,R} - r_{in,R}}\right)$$

where $f(x)$ is a differentiable step-like function, that serves to switch off the potential inside the cavity

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 3x^2 - 2x^2 & \text{for} \quad 0 < x < 1 \\ 1 & \text{for} \quad 1 < x \end{cases}$$

POT

Height V_0 of the confinement potential
Type: real
Rules: optional
Default: 20.

6.3.15 !STRUCTURE!QM-MM

Rules: optional, under development

Description: Describes the quantum-mechanical molecular-mechanical coupling. The coupling involves three different systems. (1) The quantum mechanical (QM) molecule, (2) The molecular mechanical (MM) shadow of the QM molecule and (3) the MM embedding environment, which also includes the molecule. The atoms of the QM molecule are repeated both in the MM shadow and in the MM environment, and their positions are constrained to be identical in all three parts. The total energy is $E_1 + E_3 - E_2$. The total energy $E_3 - E_2$ decribes the external forces and potentials acting from the environment on the QM molecule.

Bonds are linked in the following way: Suppose there is a bond from atom X to atom Y, where X is part of the QM molecule, but Y is only part of the environment. The bond of the QM molecule is saturated, for example by a hydrogen attached to X. An image of this hydrogen atom and its bond to X is introduced into the shadow. By specifying the option ONLY='SHADOW', this atom is (a) not constrained to be on the same position as its QM image, and (b) it is not introduced into the MM environment. (Its initial positions, however, are taken from the QM molecule.) Atom Y and its bond to X are introduced into the MM environment. As a result the two dummy hydrogen atoms move independently of each other. The bond to X is introduced twice, first linking X-H and then linking X-Y.

6.3.16 !STRUCTURE!QM-MM!ATOM

Rules: mandatory, multiple

Description: Describes the quantum-mechanical molecular-mechanical coupling.

NAME

Atom name. The atom names in the QM-MM block can be chosen independently from the choice in !STRUCTURE!ATOM.

Type: Character
Rules: Mandatory
Default: None

FFTYPE

Atom type name describing the setting of the parameters. The naming convention must be consistent with the force field chosen.
Type: Character
Rules: Mandatory
Default: None

QMATOM

Links the atomic position of this atom to the QM atom with the specified name (the name refers to that specified in !STRUCTURE!ATOMS. No atomic positions, masses or charges need to be specified for link atoms.
Type: Character
Rules: optional
Default: None

R

atomic position.
Type: real(3)
Rules: mandatory if QMM is not specified
Default: None

Q

charge in electron charges (i.e. a positive value indicates a negatively charged atom.
Type: real
Rules: optional
Default: 0.0

M

Mass in proton masses
Type: real
Rules: optional
Default: interprets the first two letters of FFTYPE (a underscore in place of a blank is removed) as element symbol for lookup in the periodic table

ONLY

specifies whether an atom is only in the MM shadow of the QM part but not in the MM part including the environment (ONLY='SHADOW')
Type: character
Rules: optional
Default: Atom is part of the MM environment

6.3.17 !STRUCTURE!QM-MM!BOND

Rules: optional,multiple
Description: specifies a bond for the classical force field

ATOM1

name of the first atom in the bond, corresponding to !STRUCTURE!QM-MM!ATOM
Type: character
Rules: mandatory
Default: none

ATOM2

name of the second atom in the bond, corresponding to !STRUCTURE!QM-MM!ATOM
Type: character
Rules: mandatory

Default: none

BO

Bond order of the bond. BO=1 for a single bond, BO=2 for a double bond. BO=3 for a triple bond.
Type: real
Rules: optional
Default: 1.

6.3.18 !STRUCTURE!QM-MM!LINK

Rules: optional,multiple
Description: specifies a link-bond between the MM and QM parts.

MMJOINT

Name of the atom common to MM, QM, and Shadow atoms. The name refers to the MM part. An atom referred as MMJOINT, must have the attribute !QMMM!ATOM:QMATOM
Type: character
Rules: mandatory
Default: none

MMATOM

Name of the MM atom bonded to MMJOINT. This atom must not have the attribute !QMMM!ATOM:QMATOM
Type: character
Rules: mandatory
Default: none

QMATOM

Name of the QM atom bonded to MMJOINT. This atom must not be used as attribute !QMMM!ATOM:QMATOM to any MM atom.

Type: character
Rules: mandatory
Default: none

SHFFTYPE

Atom type name describing the setting of the parameters for the shadow dummy atom. The naming convention must be consistent with the force field chosen.
Type: Character
Rules: Mandatory
Default: None

6.3.19 !STRUCTURE!COSMO

Rules: optional
Description: Here, data is provided that pertains to the representation of the solute surface. For each atom a radius that specifies the distance of closest approach between solute and solvent for this atom.

TESSFILE

name of the tesselation file with the full path.
Type: Character
Rules: optional
Default: Default tesselation with 60 points.

EPSILON

Dielectric constant of the solvent
Type: real
Rules: optional
Default: 10^12 (metallic screening)

VPAULI

Only for testing purposes!

Type: real
Rules: optional
Default: 0.d0

6.3.20 !STRUCTURE!COSMO!ATOM

Rules: mandatory, multiple
Description: ThereHere, data is provided that pertains to the representation of the solute surface. For each atom a radius that specifies the distance of closest approach between solute and solvent for this atom.

NAME

Atom name. Must be correspond to !STRUCTURE!ATOM.
Type: Character
Rules: Mandatory
Default: None

RAD

Distance of closest approach between solute and solvent. Usually close to the vdW radius.
Type: Real
Rules: Mandatory
Default: None

6.3.21 !STRUCTURE!GROUP

Rules: Optional, multiple
Description: Groups a number of atoms together and defines a name for the group. A few groups are predefined: The group “ALL” containing all atoms and one group for all atoms for a given atom type with the name equal to species name

NAME

Group name
Type: Character
Rules: Mandatory
Default: None

ATOMS

Names of atoms in the group
Type: array of type character
Rules: mandatory
Default: None

6.3.22 !STRUCTURE!VEXT

Rules: Optional
Description: Applies an external potential to the atoms. Note that the total energy is changed by the external potential.

6.3.23 !STRUCTURE!VEXT!UNBIND

Rules: Optional, multiple
Description: Applies a pair potential between the selected atoms, which vanishes beyonmd the sum of the van der Waals radii, and increases in a parabolic way inwardly. This option can be used in order to avoid the formation of covalent bond between certain atoms.

atom1

atom id according to !STRUCTURE!ATOM:NAME
Type: Character
Rules: ATOM1 or GROUP1 must be specified
Default: None

GROUP1

GROUP id according to !STRUCTURE!GROUP:NAME
Type: Character
Rules: ATOM1 or GROUP1 must be specified
Default: None

E

Energy of the potential at the sum of covalent radii
Type: real
Rules: mandatory
Default: None

6.3.24 !STRUCTURE!CONSTRAINTS

Rules: Optional
Description: Lists all constraints acting on atomic positions. The constraints are enforced using the scheme proposed by Ryckaert, Cicotti and Berendsen [34], which conserves the total energy in a MD simulation. Constraints are used to search for transition states, to force a system adiabatically over a reaction barrier and, to calculate free energies of reaction at finite temperatures, or to map out the total energy in a subspace of the total phase space [35]. If not specified otherwise the value and velocities are given in atomic units. `LUNIT` is NOT used.

6.3.25 !STRUCTURE!CONSTRAINTS!RIGID

Rules: optional, multiple
Description: constrains a group of atoms as a rigid body. Should not be used for linear molecules.

GROUP

Name of the group to be kept rigid

Type: Character
Rules: Mandatory
Default: None

6.3.26 !STRUCTURE!CONSTRAINTS!FREEZE

Rules: optional, multiple
Description: Keeps the specified atom or group fixed in space.

GROUP

Name of the atom or group to be fixed in space.
Type: Character
Rules: either "GROUP" or "ATOM" must be specified
Default: None

ATOM

Name of the atom or group to be fixed in space.
Type: Character
Rules: Optional
Default: None

6.3.27 !STRUCTURE!CONSTRAINTS!TRANSLATION

Rules: optional, multiple
Description: suppresses a translational momentum of a group of atoms in a given direction.

GROUP

Name of group to be fixed in space.
Type: character
Rules: Optional

Default: 'ALL'

DIR

Name of group to be fixed in space.
Type: real(3)
Rules: Optional
Default: Translation into all three space direction is suppressed

6.3.28 !STRUCTURE!CONSTRAINTS!ROTATION

Rules: optional,multiple

Description: Supresses the angular momentum $\vec{L}$ of a group of atoms in a given direction $\vec{e}$ in the reference frame of the center of gravity $\vec{R}_C(t)$ of the specified group.Thus the constraint enforces

$$\vec{e}\vec{L} = \vec{e}\sum_R M_R \left(\vec{R}_R(t) - R_C(t)\right) \times \left(\dot{\vec{R}}_R - \dot{\vec{R}}_C\right) = 0$$

Note, that this constraint cannot be represented as a hypersurface in phase space. It should not be used for phase space sampling or transition state searches. For those purposes, use the ORIENTATION constraint.

GROUP

Name of group to be fixed in space.
Type: Character
Rules: Optional
Default: 'ALL'

AXIS

Axis for which the angular momentum shall be constrained.
Type: real(3)

Rules: Optional
Default: Angular momentum into all three space directions is suppressed

6.3.29 !STRUCTURE!CONSTRAINTS!ORIENTATION

Rules: Not tested, optional, multiple
Description: Fixes the orientation of a group of atoms in a given direction. It is to be used together with a saddle search for isolated molecules. Note that this constraint does not imply zero angular momentum.

GROUP

Name of group to be fixed in space.
Type: Character
Rules: Optional
Default: 'ALL'

AXIS

Axis for which the rotation shall be constrained.
Type: real(3)
Rules: Optional
Default: translation into all three space directions is suppressed

6.3.30 !STRUCTURE!CONSTRAINTS!BOND

Rules: optional, multiple
Description: Fixes the distance between two atoms.

ATOM1

Name of the first atom. (Extended notation)
Type: Character
Rules: mandatory

Default: none

ATOM2

Name of the second atom. (Extended notation)
Type: Character
Rules: mandatory
Default: none

MOVE

Defines whether constrained value shall be moved to a new location or whether it should be given a specified constant velocity. The method uses a lowest order polynomial adjusted to the initial value and the specified final value and/or velocity to specify the time evolution of the constraint values.
Type: logical
Rules: optional
Default: .false.

NSTEP

number of time steps after which the new constraint values shall be satisfied
Type: integer
Rules: mandatory if MOVE=.true.
Default: none

VALUE

specified new value for the constraint.
Type: REAL
Rules: optional
Default: current value of the constraint.

VELOC

specified new value for the velocity of the constraint.

Type: REAL
Rules: optional
Default: 0.0

SHOW

print value and force of the constraint in the "CONSTRAINTS" file (standard extension: _constr.report)
Type: LOGICAL
Rules: optional
Default: .FALSE.

FLOAT

defines this constraint as floating constraint.
Type: logical
Rules: optional
Default: .false.

M

Mass of the constraint in a.u. A negative value results in an energy maximizing evolution for saddle point determination.
Type: real
Rules: optional
Default: 1.0=m_e

FRIC

Friction coefficient for the constraint dynamics.
Type: real
Rules: optional
Default: 0.0

6.3.31 !STRUCTURE!CONSTRAINTS!ANGLE

Rules: optional, multiple
Description: Fixes the bond angle between three atoms.

ATOM1

Name of the first atom.
Type: Character
Rules: mandatory
Default: none

ATOM2

Name of the second, central atom.
Type: Character
Rules: mandatory
Default: none

ATOM3

Name of the third atom.
Type: Character
Rules: mandatory
Default: none

MOVE

Defines whether constrained value shall be moved to a new location or whether it should be given a specified constant velocity. The method uses a lowest order polynomial adjusted to the initial value and the specified final value and/or velocity to specify the time evolution of the constraint values.
Type: logical
Rules: optional
Default: .false.

NSTEP

number of time steps after which the new constraint values shall be satisfied

Type: integer
Rules: mandatory if MOVE=.true.
Default: none

VALUE[DEG]

specified new value for the constraint in degrees.

Type: REAL
Rules: optional
Default: current value of the constraint.

VELOC

specified new value for the velocity of the constraint.

Type: REAL
Rules: optional
Default: 0.0

SHOW

print value and force of the constraint in the “CONSTRAINTS” file (standard extension: _constr.report)

Type: LOGICAL
Rules: optional
Default: .FALSE.

FLOAT

defines this constraint as floating constraint.

Type: logical
Rules: optional
Default: .false.

M

Mass of the constraint in a.u. A negative value results in an energy maximizing evolution for sassle point determination.
Type: real
Rules: optional
Default: 1.0=m_e

FRIC

Friction coefficient for the constraint dynamics.
Type: real
Rules: optional
Default: 0.0

6.3.32 !STRUCTURE!CONSTRAINTS!TORSION

Rules: optional, multiple

Description: Fixes the torsion along a bond.

ATOM1

Name of the first and first terminal atom, connected to ATOM2.
Type: Character
Rules: mandatory
Default: none

ATOM2

Name of the second, the first central atom.
Type: Character
Rules: mandatory
Default: none

ATOM3

Name of the third, the second central atom.
Type: Character
Rules: mandatory
Default: none

ATOM4

Name of the fourth, the second terminal atom, connected to ATOM3.
Type: Character
Rules: mandatory
Default: none

MOVE

Defines whether constrained value shall be moved to a new location or whether it should be given a specified constant velocity. The method uses a lowest order polynomial adjusted to the initial value and the specified final value and/or velocity to specify the time evolution of the constraint values.
Type: logical
Rules: optional
Default: .false.

NSTEP

number of time steps after which the new constraint values shall be satisfied
Type: integer
Rules: mandatory if MOVE=.true.
Default: none

VALUE

specified new value for the constraint. (A torsion angle of zero corresponds to a planar trans configuration)
Type: REAL
Rules: optional

Default: current value of the constraint.

VELOC

specified new value for the velocity of the constraint.
Type: REAL
Rules: optional
Default: 0.0

SHOW

print value and force of the constraint in the "CONSTRAINTS" file (standard extension: _constr.report)
Type: LOGICAL
Rules: optional
Default: .FALSE.

FLOAT

defines this constraint as floating constraint.
Type: logical
Rules: optional
Default: .false.

M

Mass of the constraint in a.u. A negative value results in an energy maximizing evolution for sassle point determination.
Type: real
Rules: optional
Default: 1.0=m_e

FRIC

Friction coefficient for the constraint dynamics.
Type: real
Rules: optional
Default: 0.0

6.3.33 !STRUCTURE!CONSTRAINTS!COGSEP

Rules: optional, multiple
Description: Fixes the distance between the center of gravities of two atom groups.

GROUP1

First group of atoms
Type: Character
Rules: mandatory
Default: none

GROUP2

second group of atoms.
Type: Character
Rules: mandatory
Default: none

MOVE

Defines whether constrained value shall be moved to a new location or whether it should be given a specified constant velocity. The method uses a lowest order polynomial adjusted to the initial value and the specified final value and/or velocity to specify the time evolution of the constraint values.
Type: logical
Rules: optional
Default: .false.

NSTEP

number of time steps after which the new constraint values shall be satisfied
Type: integer

Rules: mandatory if MOVE=.true.
Default: none

VALUE

specified new value for the constraint.
Type: REAL
Rules: optional
Default: current value of the constraint.

VELOC

specified new value for the velocity of the constraint.
Type: REAL
Rules: optional
Default: 0.0

SHOW

print value and force of the constraint in the “CONSTRAINTS” file (standard extension: _constr.report)
Type: LOGICAL
Rules: optional
Default: .FALSE.

FLOAT

defines this constraint as floating constraint.
Type: logical
Rules: optional
Default: .false.

M

Mass of the constraint in a.u. A negative value results in an energy maximizing evolution for sassle point determination.
Type: real
Rules: optional

Default: 1.0=m_e

FRIC

Friction coefficient for the constraint dynamics.
Type: real
Rules: optional
Default: 0.0

6.3.34 !STRUCTURE!CONSTRAINTS!MIDPLANE

Rules: optional, multiple
Description: Constrains the ATOM2 into a plane perpendicular to the vector from ATOM1 to ATOM3. The plane is attached at a point $R_1 + x(R_3 - R_1)$, where x is the value of the constraint. In other words $x = (R_2 - R_1)(R_3 - R_1)/|R_3 - R_1|^2$.

ATOM1

Name of the first atom (Extended notation).
Type: Character
Rules: mandatory
Default: none

ATOM2

Name of the second atom (Extended notation).
Type: Character
Rules: mandatory
Default: none

ATOM3

Name of the third atom (Extended notation).
Type: Character
Rules: mandatory

Default: none

MOVE

Defines whether constrained value shall be moved to a new location or whether it should be given a specified constant velocity. The method uses a lowest order polynomial adjusted to the initial value and the specified final value and/or velocity to specify the time evolution of the constraint values.

Type: logical
Rules: optional
Default: .false.

NSTEP

number of time steps after which the new constraint values shall be satisfied

Type: integer
Rules: mandatory if MOVE=.true.
Default: none

VALUE

specified new value for the constraint.

Type: REAL
Rules: optional
Default: current value of the constraint.

VELOC

specified new value for the velocity of the constraint.

Type: REAL
Rules: optional
Default: 0.0

SHOW

print value and force of the constraint in the “CONSTRAINTS” file (standard extension: _constr.report)
Type: LOGICAL
Rules: optional
Default: .FALSE.

FLOAT

defines this constraint as floating constraint.
Type: logical
Rules: optional
Default: .false.

M

Mass of the constraint in a.u. A negative value results in an energy maximizing evolution for sassle point determination.
Type: real
Rules: optional
Default: 1.0=m_e

FRIC

Friction coefficient for the constraint dynamics.
Type: real
Rules: optional
Default: 0.0

6.3.35 !STRUCTURE!CONSTRAINTS!LINEAR

Rules: optional, multiple
Description: Specifies a linear constraints of the form $\sum_R c_R R_R = a$. This very flexible type of constraint can be used to represent a complex constraint approximately in the neighborhood of a given structure. Hereby the nonlinear constraint is expanded into a Taylor expansion, and the first two terms are retained.

MOVE

Defines whether constrained value shall be moved to a new location or whether it should be given a specified constant velocity. The method uses a lowest order polynomial adjusted to the initial value and the specified final value and/or velocity to specify the time evolution of the constraint values.

Type: logical
Rules: optional
Default: .false.

NSTEP

number of time steps after which the new constraint values shall be satisfied

Type: integer
Rules: mandatory if MOVE=.true.
Default: none

VALUE

specified new value for the constraint.

Type: REAL
Rules: optional
Default: current value of the constraint.

VELOC

specified new value for the velocity of the constraint.

Type: REAL
Rules: optional
Default: 0.0

SHOW

print value and force of the constraint in the “CONSTRAINTS” file (standard extension: _constr.report)

Type: LOGICAL

Rules: optional
Default: .FALSE.

FLOAT

defines this constraint as floating constraint.
Type: logical
Rules: optional
Default: .false.

M

Mass of the constraint in a.u. A negative value results in an energy maximizing evolution for sassle point determination.
Type: real
Rules: optional
Default: 1.0=m_e

FRIC

Friction coefficient for the constraint dynamics.
Type: real
Rules: optional
Default: 0.0

6.3.36 !STRUCTURE!CONSTRAINTS!LINEAR!ATOM

Rules: mandatory, multiple
Description: Specifies the contribution of one atom to the constraint vector

NAME

name of the atom.
Type: character
Rules: mandatory
Default: none

R

vector defining c_R
Type: REAL(3)
Rules: mandatory
Default: none

6.3.37 !STRUCTURE!ORBPOT

Rules: optional
Description: applies external potentials acting on orbitals. The relation $|\Psi\rangle \approx \sum_i |\phi_i\rangle\langle\tilde{p}_i|\tilde{\Psi}\rangle$ between the true wave function and its one-center expansion is used to represent the energy of a potential u that is localized well withing the augmentation region as

$$\sum_n \langle\Psi_n|u|\Psi_n\rangle \approx \sum_{i,j} D_{i,j}U_{j,i} \ ,$$

where $D_{i,j} = \sum_n \langle\tilde{p}_i|\tilde{\Psi}_n\rangle\langle\tilde{\Psi}_n|\tilde{p}_j\rangle$ is the one-center density matrix and $U_{i,j} = \langle\phi_i|u|\phi_j\rangle$.
The special angular dependent form

$$u(r,r') = U_0 \sum_m \mathrm{e}^{-(r/r_c)^q}\delta(|r|-|r'|)Y_L(r)Y_L(r')$$

is used, where $L = (\ell, m)$ and U_0, r_c, q, ℓ and the atomic site R, which is used as origin in the above equation, are input parameters.

6.3.38 !STRUCTURE!ORBPOT!POT

Rules: optional, multiple
Description: describes external potentials acting on orbitals

ATOM

atom name or species name of the atom on which the potential should act, referring to name of !STRUCTURE!ATOMS, if a species name is given the potential will be applied to all atoms of that species.
Type: character
Rules: mandatory
Default: none

VALUE

value of the external potential in Hartree
Type: real
Rules: mandatory
Default: none

TYPE

orbital type, can be 'S', 'P', 'D', 'ALL'
Type: character
Rules: mandatory
Default: none

rc

cutoff radius r_c as defined above
Type: real
Rules: mandatory
Default: none

pwr

exponent q for the cutoff radius as defined above
Type: real
Rules: optional
Default: 2.0

S

restricts the spin on which the potential act. If this parameter is ommitted the potential acts on all electrons irrespective of their spin direction. For a collinear spin-polarized potential only $S = 1$ is allowed and defines a potential acting on the z-component of the spin. For a non-collinear calculation (NSPIN=3 in !STRUCTURE!GENERIC) the values of $S = 1, S = 2, S = 3$ refer to the x,y,z components of the spin respectively.

Type: integer
Rules: optional
Default: 0

7 The Energy Grab Tool: paw_grab

The tool paw_grab is useful to collect energies from different systems and compile a set of reaction energies. It looks for the last (long) printout of energies in a given protocol. This total energy is then attributed to a given molecule name. Arbitrary reaction energies between the selected molecules can automatically be compiled. Particularly useful is that a number of reaction energies can be updated with the latest computed energies using a single command.

7.1 Command

The calling sequence is

`paw_grab` *controlfile*

where *controlfile* is the file name of the control input file for the paw_grab tool. I recommend the extension ".gcntl".

7.2 Example for the control input file

```
!GCNTL
  !GENERIC UNIT(E)='KJ/MOL' !END
  !SUBSTANCE ID='H2' FILE='~/PAW/Sample/h2.prot' !END
  !SUBSTANCE ID='CO' FILE='~/PAW/Sample/co.prot' !END
  !SUBSTANCE ID='H2CO' FILE='~/PAW/Sample/h2co.prot' !END
  !REACTION
    !FROM ID='H2'   FAC=1.  !END
    !FROM ID='CO'   FAC=1.  !END
    !TO   ID='H2CO' FAC=1.  !END
  !END
!END
!EOB
```

7.3 Argument description for the control input file

7.3.1 !GCNTL

Rules: mandatory
Description: defines location of the protocol files to be searched and the reaction energies to be calculated.

7.3.2 !GCNTL!GENERIC

Rules: optional
Description: general definitions

UNIT(E)

Identifier of the energy unit. Can be “HARTREE”, “KJ/MOL” “KCAL/MOL”, “RY”, “EV”, “JOULE”
Type: character
Rules: optional
Default: “HARTREE”

7.3.3 !GCNTL!SUBSTANCE

Rules: optional
Description: defines a reactand or reaction product through a protocol file of the simulation code

ID

Identifier of the substance
Type: character
Rules: mandatory
Default: none

FILE

File name of the protocol file
Type: character
Rules: mandatory
Default: none

7.3.4 !GCNTL!REACTION

Rules: optional, multiple
Description: defines the reaction

ID

Identifier of the reaction.
Type: character
Rules: optional
Default: A stochiometric formula of the reaction created from reactands and products and their stochiometries

7.3.5 !GCNTL!REACTION!FROM

Rules: optional, multiple
Description: defines the reactand

ID

Identifier of the reactand
Type: character
Rules: mandatory
Default: none

FAC

defines the stochiometry with which the the reactand takes part in the reaction

Type: real
Rules: mandatory
Default: none

7.3.6 !GCNTL!REACTION!TO

Rules: optional, multiple
Description: defines the reaction product

ID

Identifier of the product
Type: character
Rules: mandatory
Default: none

FAC

defines the stochiometry with which the product is produced by the reaction
Type: real
Rules: mandatory
Default: none

8 The Atomic Structure Analysis Tool: "paw_strc"

8.1 Command

The tool paw_strc prepares a cssr structure file which can be read, by some molecular modeling software packages. This tool uses the *rootname*.strc_out file of the simulation code, and writes a file *rootname*.cssr.

The calling sequence is

`paw_strc` *rootname*

where rootname is the name of the protocol file without the ".prot" ending and option is one of the following:

-c creates output with a unit cell for crystals. Relative coordinates and lattice constants and angles are used. The default produces molecular output with absolute Cartesian coordinates in Å.

9 The Wave Function and Density Analysis Tool: “paw_wave”

The paw_wave analysis tool converts a waveplot file into a data file readable by OPENDX[2]. The paw_wave analysis tool requires an input file which is created after setting the options “!CONTROL!ANALYSE!WAVE” in the control input file of the simulation code and the ’strc_out’ file of the simulation code. It can be inspected by the Datexplorer program “wave.net”.

9.1 Command

The calling sequence is

`paw_wave` *controlfile*

where *controlfile* is the file name of the control input file for the paw_dos tool described below. I recommend the extension “.wcntl” for the control file of the density of states analysis tool.

9.2 Example for the control input file

```
!WCNTL
  !FILES
    !FILE ID='WAVE'
          NAME='~/PAW/Sample/h2co-homo.wv' !END
    !FILE ID='STRC'
          NAME='~/PAW/Sample/h2co.strc_out' !END
    !FILE ID='WAVEDX'
          NAME='~/PAW/Sample/h2co-w.dx' !END
  !END
  !VIEWBOX O=-5. -5. -5.
           T=10. 0. 0. 0. 10. 0. 0. 0. 10. !END
!END
!EOB
```

9.3 Argument description for the control input file

9.3.1 !WCNTL

Rules: mandatory
Description: defines the operations done on the system; largely independent of the system

9.3.2 !WCNTL!FILES

Rules: optional
Description: defines the energy grid and energy broadening

9.3.3 !WCNTL!FILES!FILE

Rules: optional, multiple
Description: defines the energy grid and energy broadening

ID

Identifier of the file. Can be 'STRC', 'WAVE', 'WAVEDX'.
Type: character
Rules: mandatory
Default: none

EXT

rootname flag: if true, the filename is interpreted as an extension of the rootname. The rootname is the filename of the wavecontrol file with out the last dot and the following characters.
Type: logical
Rules: optional
Default: .false.

NAME

File name
Type: character
Rules: mandatory
Default: none

9.3.4 !WCNTL!VIEWBOX

Rules: optional
Description: Sets the view box. Only the data within the view box are shown. The view box is defined by an origin o and three vectors t_1, t_2, t_3. The eight corners of the view box are $o, o+t_1, o+t_2, o+t_3, o+t_1+t_2, o+t_2+t_3, o+t_1+t_3, o+t_1+t_2+t_3$

O

Origin (lower left, forward point of the view box) in a.u.
Type: real(3)
Rules: optional
Default: 0. 0. 0.

T

vectors defining the edges of the view box in a.u.
Type: real(3,3)
Rules: optional
Default: lattice vectors of the calculation

9.4 View data using OPENDX[2]

Start the Dataexplorer using

```
dx -image &
```

1. Select "**Load Macro**" from the "**file**" pulldown menu. In the file selector, which pops up, check that you are in the DX subdirectory of the PAW directory and click on "**Load all macros**".

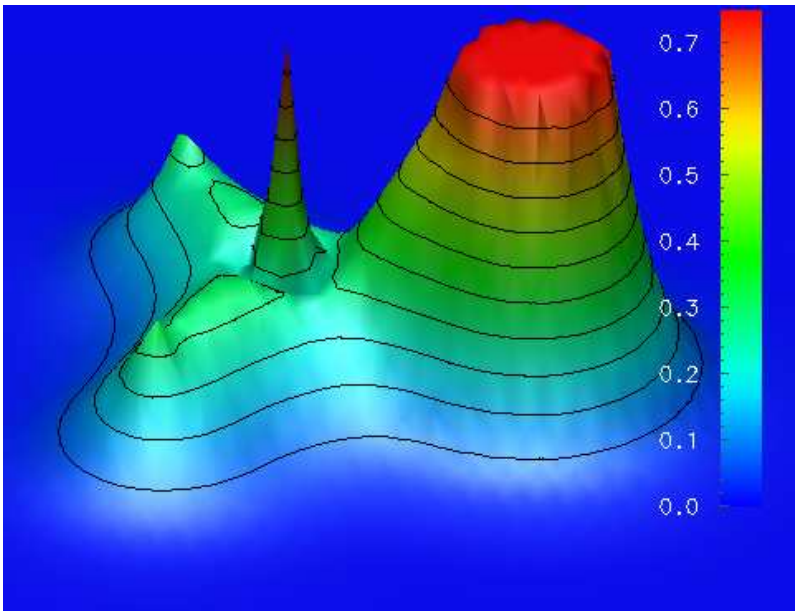
2. Select "**open**" from the "**file**" pulldown menu. In the file selector, which pops up, check that you are in the DX subdirectory of the PAW directory and select the dx program "**dx_wave**".

3. select "**open all control panels**" from the "**windows**" pulldown menu.

4. In the "**Main Control Panel**" that opens, select the input data file. You can do this by typing the file name into the text field of the file selector, or by pressing the button to the right of the text field and by maneuvering towards the right file in the file selector menu. After selecting the correct file, press OK.

5. Now select the the representation of your data. You can select

   - a ball-stick model of your atomic structure,
   - a surface of constant density or wave function amplitude (there is a golden surface for the value provided and a blue surface for the same value with opposite sign)
   - a rubber sheet representation of the density on a given plane. The plane is defined by a point through which the plane passes and a vector normal to the surface. You can, for example, select an atom position of interest as the point, and then adjust the orientation of the plane by modifying the vector. The plane will then rotate about the atom position selected.



If your viewing direction is straight onto the plane, you will obtain a (color) contour plot.

By adjusting the maximum value, you can cut off very large values, and thus obtain more detail for the lower density values.

10 Wave function converter for CryMolCAD: “paw_cmcwave”

The tool converts the wave function file .wv produced by the simultation code into an input file for the CrymolCAD viewer. The wave function can be a general field such as wave function, density or potential.

paw_cmcwave.x case.wv

A new file is case.cmcv created, which can be read by CryMolCAD. The file contains only the information for the field. The structure information must be supplied separately.

11 Band-Offset and Work-Function Tool: "paw_1davpot"

The tool "paw_1davpot" produces the one dimensional potential obtained by averaging the potential over lattice planes. By comparing with the respective bulk calculations one can determine the band offsets or work functions.

The tool uses the electrostatic potential produces by the CP-PAW code using the option !control!analyse!potential. If the file of the resulting potential file is "case_pot.wv", the calling sequence is

paw_1davpot.x case_pot.wv

Three files are produced, namely AVPOT100.DAT, AVPOT010.DAT and AVPOT001.DAT. The first is an xy file of the averaged potential along the normal of the plane spanned by the lattice vectors $\vec{T}_2$ and $\vec{T}_3$. The other files contain the corresponding information along the two other plane normals.

In order to obtain a band offset from a supercell calculation determine first the one-dimensional potential $v_{sup}(z)$ averaged over planes parallel to the surface or interface. Next perform a calculation of the bulk material using exactly the same geometry (including any lateral strain), and determine the averaged potentials $v_{bulkA}(z)$ and $v_{bulk,B}$ for the same lattice planes. Determine furthermore the Fermi-level or the valence band top in the bulk materials $e_{vbt,A}$ and $e_{vbt,B}$.

By forming the corresponding differences, one obtains a spatially dependent valence band top $e'_{vbt,A}$ and $e'_{vbt,B}$ for the slab calculation.

$$e'_{vbt,A}(z) = e_{vbt,A} - v_{bulk,A}(z + \Delta z_A) + v_{sup}(z)$$
$$e'_{vbt,B}(z) = e_{vbt,B} - v_{bulk,A}(z + \Delta z_B) + v_{sup}(z)$$

The shift $\Delta z_{A/B}$ are needed to account for the difference in z-coordinates for the lattice planes in the bulk and in the supercell.

If the spatially dependent valence band tops converge to a constant value in the interior of the slab, this value can be taken as the position of the valence band top in the supercell.

12 The Trajectory Analysis Tool: “paw_tra”

This trajectory analysis tool is used to inspect atomic trajectories. It allows one to create a movie input file for OPENDX[2], or print the time evolution of selected bond lengths, angles, torsions or any combination thereof.

The time scale is measured in psec. (If you cannot view the ball stick model in OPENDX[2], please convert the dx file globally to lower case.)

12.1 Command

The calling sequence is

`paw_tra` *controlfile*

where *controlfile* is the file name of the control input file for the paw_tra tool. I recommend the extension “.tcntl”.

12.2 Example for the control input file

```
!TCNTL
  !FILES
    !FILE ID='STRC'
          NAME='~/PAW/Sample/h2co.strc_out' !END
    !FILE ID='TRA'
          NAME='~/PAW/Sample/h2co_r.tra' !END
  !END
  !SEQUENCE T1[PS]=0. T2[PS]=2. DT[FS]=1. !END
  !MOVIE skip=2
    !VIEWBOX O=3*-5.  T=10. 3*0. 10. 3*0. 10. !END
    !FILE  EXT=T NAME='_movie.dx' !END
  !END
  !temperature retard[ps]=0.1
    !select atoms= 'H_1' 'H_2' !END
    !FILE  EXT=T NAME='.h.temp.dx' !END
  !end
  !spaghetti
    !FILE  EXT=T NAME='.temp.pasta' !END
  !end
```

```
  !MODE ID='H-C-BOND'
    !BOND SCALE=1.0 ATOM1='H_1' ATOM2='C_3' !END
  !END
  !MODE ID='H-ANTI-STRETCH'
    !MODE SCALE=1.0 ID='H1-C-BOND' !END
    !BOND SCALE=-1.0 ATOM1='H_2' ATOM2='C_3' !END
  !END
  !MODE ID='H-C-H-ANGLE'
    !ANGLE SCALE=1.0 ATOM1='H_1' ATOM2='C_3'
           ATOM3='H_2' !END
  !END
  !OUTPUT ID='H-ANTI-STRETCH' TYPE='VELOCITY' !END
!END
!EOB
```

12.3 Argument description for the control input file "tcntl"

12.3.1 !TCNTL

Rules: mandatory
Description:

12.3.2 !TCNTL!FILES

Rules: optional
Description: specifies the file names that deviate from standard file names

ROOT

root name. Files defined as extension will have this name combined with the extension. All files connected as default are defined as extensions.

Type: character
Rules: optional
Default: string preceding the '.tcntl' ending of the control input file

12.3.3 !TCNTL!FILES!FILE

Rules: optional, multiple
Description: define the file name:

- **PROT** protocol. Standard extension: '.tprot'
- **CNTL** control input file for the paw_tra tool. The name of this file is mandatory input and cannot be reset.
- **STRC** structure output file produced by the simulation and used as input. standard extension: '.strc_out'
- **TRA** trajectory file produced by the simulation and used as input. standard extension: '_r.tra'

ID

Identifier of the file.
Type: character
Rules: mandatory
Default: none

EXT

rootname flag: if true, the filename is interpreted as an extension of the rootname. The rootname is the filename of the wavecontrol file without the last dot and the following characters.
Type: logical
Rules: optional
Default: .false.

NAME

File name
Type: character

Rules: mandatory
Default: none

12.3.4 !TCNTL!SEQUENCE

Rules: optional
Description: specifies the beginning and ending time and the sampling frequency of the sequence to be analyzed.

T1[PS]

Beginning time of the sequence in picoseconds
Type: real
Rules: optional
Default: time of the first frame on the trajectory file

T2[PS]

ending time of the sequence in picoseconds
Type: real
Rules: optional
Default: time of the last frame on the trajectory file

DT[FS]

sampling frequency in femtoseconds
Type: real
Rules: optional
Default: spacing on the trajectory file

12.3.5 !TCNTL!MOVIE

Rules: optional
Description: writes a input file for OPENDX[2] to show the moving atoms as a ball-stick model. Bonds are drawn for every pair of atoms that are closer than 1.2 times the sum of the covalent radii.

FORMAT

descriptor for the format of the movie file to be created. can be 'DX' (OPENDX[2]), 'XYZ' (xyz format)
Type: character
Rules: optional
Default: 'DX'

SKIP

number of frames on the input trajectory files that are skipped between movie frames
Type: integer
Rules: optional
Default: 0.

12.3.6 !TCNTL!MOVIE!VIEWBOX

Rules: optional
Description: Sets the view box. Only the data within the view box are shown. The view box is defined by an origin o and three vectors t_1, t_2, t_3. The eight corners of the view box are $o, o+t_1, o+t_2, o+t_3, o+t_1+t_2, o+t_2+t_3, o+t_1+t_3, o+t_1+t_2+t_3$

O

Origin (lower left, forward point of the view box) in a.u.
Type: real
Rules: optional; incompatible with "!TCNTL!MOVIE!VIEWBOX:C"
Default: 0. 0. 0.

C

Center of viewbox
Type: real
Rules: optional; incompatible with "!TCNTL!MOVIE!VIEWBOX:O"
Default: 0. 0. 0.

T

vectors defining the edges of the view box
Type: real(3,3)
Rules: optional
Default: lattice vectors of the calculation

12.3.7 !TCNTL!MOVIE!FILE

Rules: optional
Description: specify the file where the movie plot is written. The file is then used with an appropriate visualizer.

EXT

switch to the rootname appended by the extension provided, versus using the filename (including its path).
Type: logical
Rules: optional
Default: .false.

NAME

file name
Type: character
Rules: optional
Default: the rootname appended by ".movie.dx" for !TCNTL!MOVIE:FORMAT='DX' and by ".movie.xyz" for !TCNTL!MOVIE:FORMAT='XYZ'

12.3.8 !TCNTL!MOVIE!SELECT

Rules: optional
Description: specify the atoms to shall be used. Unless specified, all atoms are plotted.

ATOMS

atom names consistent with the “strc” file. An arbitrary number of atoms can be included
Type: character
Rules: mandatory
Default: none

12.3.9 !TCNTL!CORRELATION

Rules: optional
Description: evaluates pair correlation functions for distances and angles

Type

select distance or angle correlation. (Angle correlations not yet implemented!)
Type: character(1); allowed values are ’D’ for distance and ’A’ for angle
Rules: otional
Default: ’D’

DEP

selects correlation over time as spaghetti plot or the time averaged histogram as function of angle or distance.
Type: character; allowed values are ’SPAGHETTI’ and ’HISTOGRAM’
Rules: optional

Default: 'SPAGHETTI'

12.3.10 !TCNTL!CORRELATION!FILE

Rules: optional
Description: specify the file where cssr file is written to.

EXT

switch to the rootname appended by the extension provided, versus using the filename (including its path).
Type: logical
Rules: optional
Default: .false.

NAME

file name
Type: character
Rules: optional
Default: the rootname appended by ".tra.cssr"

12.3.11 !TCNTL!CORRELATION!CENTER

Rules: optional
Description: specify the atoms to be used. Unless specified, all atoms are plotted.

12.3.12 !TCNTL!CORRELATION!CENTER!SELECT

Rules: optional
Description: specify the atoms to be used, as reference atoms. Unless specified, all atoms are SELECTED.

ATOMS

atom names consistent with the “strc” file. An arbitrary number of atoms can be included

Type: character
Rules: mandatory
Default: none

12.3.13 !TCNTL!CORRELATION!PARTNER

Rules: optional
Description: specify the atoms to be used as partner atoms. Unless specified, all atoms are selectes.

12.3.14 !TCNTL!CORRELATION!PARTNER!SELECT

Rules: optional
Description: specify the atoms to be used. Unless specified, all atoms are selected.

ATOMS

atom names consistent with the “strc” file. An arbitrary number of atoms can be included

Type: character
Rules: mandatory
Default: none

12.3.15 !TCNTL!SNAPSHOT

Rules: optional
Description: picks the unit cell and the atomic position for a given snapshot in a trajectory. The data are written to the protocoll file in a pseudo format for the structure input file and as cssr file to the file specified.

T[PSEC]

time at which the snapshot shall be taken out of the trajectory. The code selects the last timestep before the given time.

Type: real
Rules: either T[PSEC] or STEP must be selected
Default: none

STEP

time step number at which the snapshot shall be taken out of the trajectory.

Type: integer
Rules: either T[PSEC] or STEP must be selected
Default: none

12.3.16 !TCNTL!SNAPSHOT!FILE

Rules: optional
Description: specify the file where cssr file is written to.

EXT

switch to the rootname appended by the extension provided, versus using the filename (including its path).

Type: logical
Rules: optional
Default: .false.

NAME

file name

Type: character
Rules: optional
Default: the rootname appended by ".tra.cssr"

12.3.17 !TCNTL!SPAGHETTI!SELECT

Rules: optional
Description: specify the atoms to be used. Unless specified, all atoms are plotted.

ATOMS

atom names consistent with the "strc" file. An arbitrary number of atoms can be included
Type: character
Rules: mandatory
Default: none

12.3.18 !TCNTL!SPAGHETTI

Rules: optional
Description: performs a spaghetti plot ($|R(t) - R(0)|$) for a number of atoms

12.3.19 !TCNTL!SPAGHETTI!FILE

Rules: mandatory
Description: specify the file where the spaghetti plot is written. The file is formatted and to be used with an x-y plotting tool. Each line containes the time in ps and the distance in Angstrom

EXT

switch to the rootname appended by the extension provided, versus using the filename (including its path).
Type: logical
Rules: optional
Default: .false.

NAME

file name
Type: character
Rules: optional
Default: the rootname appended by ".tra.pasta"

12.3.20 !TCNTL!SPAGHETTI!SELECT

Rules: optional
Description: specify the atoms to be used. Unless specified, all atoms are plotted.

ATOMS

atom names consistent with the "strc" file. An arbitrary number of atoms can be included
Type: character
Rules: mandatory
Default: none

12.3.21 !TCNTL!TEMPERATURE

Rules: optional, multiple
Description: prints the average temperatures of the atoms and optionally a plot for the temperature versus time for individual atoms.

RETARD[PS]

The values are averaged over previous values with exponential decay.

$$x(t) = \frac{1}{t_0}\int_{-\infty}^{t} dt' x(t') \exp(-\frac{t-t'}{t_0})$$

This value is the decay time t_0 in pico seconds.
Type: real
Rules: optional

Default: 0.

12.3.22 !TCNTL!TEMPERATURE!SELECT

Rules: optional
Description: specify the atoms to be used. Unless specified, all atoms are plotted.

ATOMS

atom names consistent with the “strc” file. An arbitrary number of atoms can be included
Type: character
Rules: mandatory
Default: none

12.3.23 !TCNTL!TEMPERATURE!FILE

Rules: mandatory
Description: specifies the file where the plot is written. The file is formatted and to be used with an x-y plotting tool. Each line containes the time in ps and the temperature in Kelvin

EXT

switch to the rootname appended by the extension provided, versus using the filename (including its path).
Type: logical
Rules: optional
Default: .false.

NAME

file name
Type: character
Rules: optional

Default: the rootname appended by ".tra.temp"

12.3.24 !TCNTL!SOFT

Rules: optional
Description: produces the mass weighted path-length versus time

$$s(t) = \int_0^t \sqrt{\sum_i M_i \dot{R}_i^2}$$

. This mapping can be used to transform a time scale to a length scale, using the definitions of the intrinsic reaction coordinate (IRC)[**?**]. The unit of the mass weighted path length is $\text{Å}\sqrt{\text{u}}$, where u is the "atomic mass unit", i.e. one twelfth of the mass of ^{12}C.

12.3.25 !TCNTL!SOFT!SELECT

Rules: optional
Description: specify the atoms to be used. Unless specified, all atoms are used.

ATOMS

atom names consistent with the "strc" file. An arbitrary number of atoms can be included
Type: character
Rules: mandatory
Default: none

12.3.26 !TCNTL!SOFT!FILE

Rules: optional
Description: specify the file where the mass-weighted path length versus time is written. The file is formatted and to be used with an x-y plotting tool. Each line contains the time in ps and the distance in angstrom

EXT

switch to the rootname appended by the extension provided, versus using the filename (including its path).
Type: logical
Rules: optional
Default: .false.

NAME

file name
Type: character
Rules: optional
Default: the rootname appended by ".tra.soft"

12.3.27 !TCNTL!NEIGHBORS

Rules: optional
Description: print a neighborlist at the beginning and the end of a trajectory and report all bon-breaking and bond-forming events. The report will be written into a file with extension ".tra.neighbors"

12.3.28 !TCNTL!MODE

Rules: optional
Description: defines a vibrational mode of the system such as a bond length, an angle etc. Note that all bond vectors that define bonds, angles or torsions are mapped onto the minimum image.

ID

Identifier of this mode to be used in further operations can be any string, but different from all other mode-identifierss.
Type: character
Rules: mandatory
Default: none

12.3.29 !TCNTL!MODE!BOND

Rules: optional
Description: adds a bond length as a contribution defining a mode. The bond is defined as the distance between two atoms R_1, R_2.

SCALE

Scales the parameter before adding it to the mode. (Used for example to define symmetric and antisymmetric bond stretch modes.)
Type: real
Rules: mandatory
Default: none

ATOM1

Name of the first atom R_1 defining the bond. The name is chosen consistent with the structure input file of the simulation code.
Type: character
Rules: mandatory
Default: none

ATOM2

Name of the second atom R_2 defining the bond. The name is chosen consistent with the structure input file of the simulation code.

Type: character
Rules: mandatory
Default: none

12.3.30 !TCNTL!MODE!ANGLE

Rules: optional
Description: adds a bond angle as a contribution defining a mode. The bond angle is the angle between the two bonds (R_1, R_2) and (R_2, R_3) The unit for angles is such that 2π is a full turn.

SCALE

Scales the parameter before adding it to the mode.

Type: real
Rules: mandatory
Default: none

ATOM1

Name of the first terminal atom R_1 defining the bond angle. The name is chosen consistent with the structure input file of the simulation code.

Type: character
Rules: mandatory
Default: none

ATOM2

Name of the central atom R_2 defining the bond angle. The name is chosen consistent with the structure input file of the simulation code.

Type: character
Rules: mandatory
Default: none

ATOM3

Name of the second terminal atom R_3 defining the bond angle. The name is chosen consistent with the structure input file of the simulation code.

Type: character
Rules: mandatory
Default: none

12.3.31 !TCNTL!MODE!TORSION

Rules: optional

Description: Warning! this option does not seem to work! adds a bond torsion as a contribution defining a mode. The unit for angles is such that 2π is a full turn. A torsion is defined by four atoms (R_1, R_2, R_3, R_4). The torsion about the central bond (R_2, R_3) is the angle between the two planes defined by two triples (R_1, R_2, R_3) and (R_2, R_3, R_4).

SCALE

Scales the parameter before adding it to the mode.

Type: real
Rules: mandatory
Default: none

ATOM1

Name of the atom R_1 defining the bond torsion consistent with the structure input file of the simulation code.

Type: character
Rules: mandatory
Default: none

ATOM2

Name of the atom R_2 defining the bond torsion consistent with the structure input file of the simulation code.
Type: character
Rules: mandatory
Default: none

ATOM3

Name of the atom R_3 defining the bond torsion consistent with the structure input file of the simulation code.
Type: character
Rules: mandatory
Default: none

ATOM4

Name of the atom R_4 defining the bond torsion consistent with the structure input file of the simulation code.
Type: character
Rules: mandatory
Default: none

12.3.32 !TCNTL!MODE!MODE

Rules: optional
Description: adds a predefined mode as a contribution defining a mode.

SCALE

Scales the parameter before adding it to the mode.
Type: real
Rules: mandatory
Default: none

ID

ID of the mode to be included.
Type: character
Rules: mandatory
Default: none

12.3.33 !TCNTL!OUTPUT

Rules: optional, multiple
Description: writes the time dependence of a mode to file

ID

identifier of the mode to be printed
Type: character
Rules: mandatory
Default: none

FILENAME

file name of the file to which the result is written
Type: character
Rules: mandatory
Default: none

TYPE

Defines the property to be written,

- default: the mode itself
- 'VELOCITY' the time derivative of the mode

Type: character
Rules: optional

Default: position

12.4 View data using OPENDX[2]

Start the Dataexplorer using

```
dx -image &
```

1. Select "**Load Macro**" from the "**File**" pulldown menu. In the file selector, which pops up, check that you are in the DX subdirectory of the PAW directory and click on "**Load all macros**".

2. Select "**Open**" from the "**File**" pulldown menu. In the file selector, which pops up, select the dx program "**tra.net**".

3. select "**open all control panels**" from the "**windows**" pulldown menu.

4. In the "**Main Control Panel**" that opens, select the input data file. You can do this by typing the file name into the text field of the file selector, or by pressing the button to the right of the text field and by maneuvering towards the right file in the file selector menu. After selecting the correct file, press OK.

5. Select "**Sequencer**" from the "**Execute**" Pulldown menu.

6. Use the sequencer like a VCR. (In the existing version, an error frequently occurs while reading the data file. The message says that it cannot read beyond a certain number of time slices. Remember the number, click on the right upper field of the sequencer, and type the number into the field for max. Then click on the same field of the sequencer, and continue.)

13 The Density of States Analysis Tool: paw_dos

The paw_dos tool allows one to analyze binding locally and energy resolved. It can tell you which states contribute mostly to a particular σ, π, δ bond. For example, you can learn whether either bonding and antibonding states are close to the Fermi level (HOMO-LUMO gap), in which case a structural deformation or a charge transfer can affect such a bond strongly. This analysis is similar in spirit to the Crystal-Orbital-Overlap-Population (COOP) analysis of R. Hoffmann. It is of tremendous value to obtain a chemical (local) description of binding in complex systems, where states are delocalized and closely spaced in energy.

13.1 Command

The calling sequence is

`paw_dos` *controlfile*

where *controlfile* is the file name of the control input file for the paw_dos tool. I recommend the extension ".dcntl".

13.2 Theoretical basis for the paw_dos tool

The density of states matrix is defined as follows

$$D_{\chi,\chi'}(\epsilon) = \sum_n \langle\chi|\Psi_n\rangle\delta(\epsilon-\epsilon_n)\langle\Psi_n|\chi'\rangle \tag{2}$$

from the Kohn-Sham orbitals Ψ_n and their one-particle energies ϵ_n. The orbitals χ and χ' pick out certain matrix elements of the density-of-states operator. The possible choices for the orbitals χ, χ' are described in more detail below.

The density of states is most useful to obtain a qualitative understanding of chemical binding, both local and energy-resolved. Consider, for example, a complete, orthonormal set of orbitals χ_{RL} similar to atomic orbitals.

The band structure energy $\sum f(\epsilon_n)\epsilon_n$, where $f(\epsilon)$ is the Fermi distribution function and ϵ_n are the one-particle energies, can be decomposed into bond contributions.

$$\sum_n f(\epsilon_n)\epsilon_n = \sum_{i,j,k,l}\int d\epsilon f(\epsilon) D_{i,j}(\epsilon) O^{-1}_{j,k} H_{k,l} O^{-1}_{k,i} \tag{3}$$

where

$$H_{i,j} = \langle\chi_i| -\frac{1}{2}\nabla^2 + v|\chi_j\rangle \tag{4}$$

is the Hamilton operator and

$$O_{i,j} = \langle \chi_i | \chi_j \rangle \tag{5}$$

is the overlap matrix for a complete set of functions χ_i.

Two sum rules connect the overlap matrix element with the zeroth energy moment of the projected density of states

$$\begin{aligned}
\langle \chi_i | \chi_j \rangle &= \sum_{n,m} \langle \chi_i | \Psi_n \rangle \langle \Psi_n | \Psi_m \rangle \langle \Psi_m | \chi_j \rangle \\
&= \sum_{n} \langle \chi_i | \Psi_n \rangle \langle \Psi_n | \chi_j \rangle \\
&= \sum_{n} \int_{-\infty}^{\infty} d\epsilon \, \langle \chi_i | \Psi_n \rangle \delta(\epsilon - \epsilon_n) \langle \Psi_n | \chi_j \rangle \\
&= \int_{-\infty}^{\infty} d\epsilon \, D_{i,j}(\epsilon)
\end{aligned}$$

and the Hamilton matrix element with the first moment:

$$\begin{aligned}
\langle \chi_i | H | \chi_j \rangle &= \sum_{n,m} \langle \chi_i | \Psi_n \rangle \langle \Psi_n | H | \Psi_m \rangle \langle \Psi_m | \chi_j \rangle \\
&= \sum_{n} \langle \chi_i | \Psi_n \rangle \epsilon_n \langle \Psi_n | \chi_j \rangle \\
&= \sum_{n} \int_{-\infty}^{\infty} d\epsilon \, \epsilon \langle \chi_i | \Psi_n \rangle \delta(\epsilon - \epsilon_n) \langle \Psi_n | \chi_j \rangle \\
&= \int_{-\infty}^{\infty} d\epsilon \, \epsilon D_{i,j}(\epsilon)
\end{aligned}$$

Let us now make a simplifying assumption: assume that the local functions χ_i have only diagonal overlap matrix elements, in which case the bond contributions have the form

$$\sum_n f(\epsilon_n)\epsilon_n = \sum_{i,j} \frac{\left[\int d\epsilon f(\epsilon) D_{i,j}(\epsilon)\right]\left[\int d\epsilon \, \epsilon D_{j,i}(\epsilon)\right]}{\left[\int d\epsilon D_{i,i}(\epsilon)\right]\left[\int d\epsilon D_{j,j}(\epsilon)\right]}$$

This equation allows us to estimate the covalent bond strength from the peak positions and peak strengths of the projected density of states. Note that this analysis is only qualitative. As a qualitative tool, however, this analysis has been invaluable.

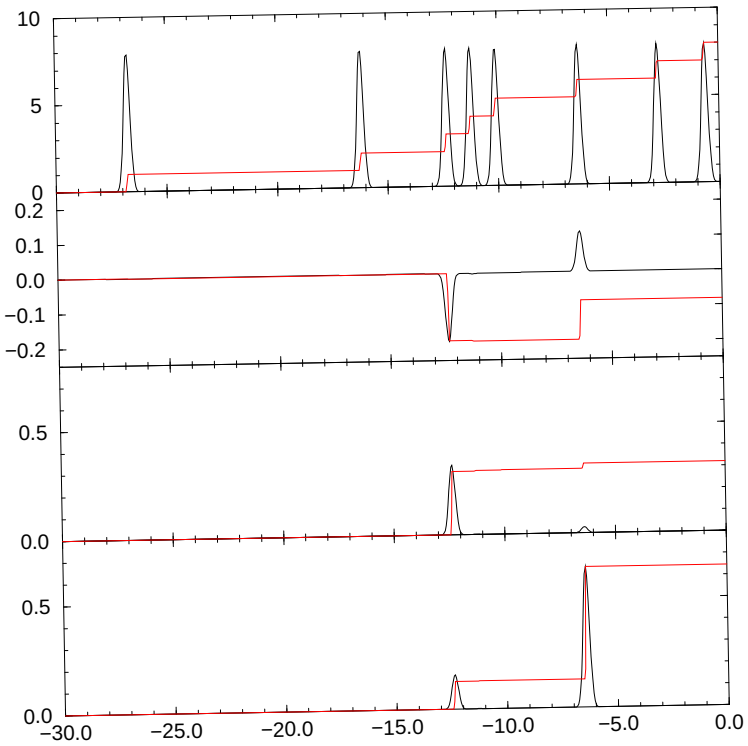


Figure 1: From top to bottom (1) total density of states of the formaldehyde molecule (H_2CO). (2) COOP between the p-orbitals lying in the plane of the molecule but perpendicular to the CO axis. (3) density of states of the p-orbital contributing to the COOP on the carbon atom, (4) density of states of the p-orbital contributing to the COOP on the oxygen atom

13.2.1 Orbitals

The paw_dos tool works with elementary orbitals, identified by the atom name, and an abbreviation of an angular momentum such as

s An s-orbital with angular dependence $Y_s = \sqrt{\frac{1}{4\pi}}$

px A p-orbital with angular dependence $Y_{p_x} = \sqrt{\frac{3}{4\pi}}\frac{x}{|r|}$

py A p-orbital with angular dependence $Y_{p_y} = \sqrt{\frac{3}{4\pi}}\frac{y}{|r|}$

pz A p-orbital with angular dependence $Y_{p_z} = \sqrt{\frac{3}{4\pi}}\frac{z}{|r|}$

dxy A d-orbital with angular dependence $Y_{d_{xy}} = \sqrt{\frac{60}{16\pi}}\frac{xy}{|r|^2}$

dxz A d-orbital with angular dependence $Y_{d_{xz}} = \sqrt{\frac{60}{16\pi}}\frac{xz}{|r|^2}$

dyz A d-orbital with angular dependence $Y_{d_{yz}} = \sqrt{\frac{60}{16\pi}}\frac{yz}{|r|^2}$

d3z2-r2 A d-orbital with angular dependence $Y_{d_{3z^2-r^2}} = \sqrt{\frac{5}{16\pi}}\frac{3z^2-r^2}{|r|^2}$

dx2-y2 A d-orbital with angular dependence $Y_{d_{x^2-y^2}} = \sqrt{\frac{15}{16\pi}}\frac{x^2-y^2}{|r|^2}$

sp An sp^1 orbital pointing in z-direction: $\sqrt{\frac{1}{2}}Y_s\phi_s(|r|) + \sqrt{\frac{1}{2}}Y_{p_z}\phi_p(|r|)$

sp2 An sp^2 orbital pointing in z-direction: $\sqrt{\frac{1}{3}}Y_s\phi_s(|r|) + \sqrt{\frac{2}{3}}Y_{p_z})\phi_p(|r|)$

sp3 An sp^3 orbital pointing in z-direction: $\sqrt{\frac{1}{4}}Y_s\phi_s(|r|) + \sqrt{\frac{3}{4}}Y_{p_z})\phi_p(|r|)$

where the spherical harmonics are expressed relative to the Cartesian coordinate system used in the calculation. Instead of projecting on atomic orbitals as in a Mulliken population analysis, we project here onto partial waves, that are truncated at the atomic sphere radius. The atomic sphere radius is approximately 10% larger than the covalent radius. (The factor corresponds to the ratio between the radius of a volume-filling-sphere radius in an fcc crystal and the corresponding touching-sphere radius.)

More complex orbitals such as bond orbitals can be constructed from these elementary orbitals by using the block "!DCNTL!ORBITAL". The orbitals are named and can be referred to later by their name.

In practice we define the orbital as $\sum_i |\chi_i\rangle = \sum_j |\phi_j\rangle c_{j,i}$, and

$$\begin{aligned}\langle A\rangle &= \sum_{n,i,j}\langle\tilde{p}_i|\Psi_n\rangle f_n\langle\Psi_n|\tilde{p}_j\rangle\langle\phi_j|A|\phi_l\rangle \\ &= \sum_{n,i,j,k,l}\langle\tilde{p}_k|\Psi_n\rangle f_n\langle\Psi_n|\tilde{p}_l\rangle\frac{c_{l,A}c^*_{i,A}}{\sum_m c^*_{m,A}c_{m,A}}\langle\phi_i|A|\phi_j\rangle\frac{c_{j,B}c^*_{k,B}}{\sum_m c^*_{m,B}c_{m,B}}\end{aligned}$$

which is divided up into into a density matrix and and a matrix element according to

$$\begin{aligned}D_{B,A} &= \sum_{n,i,j,k,l}\frac{c^*_{k,B}}{\sum_m c^*_{m,B}c_{m,B}}\langle\tilde{p}_k|\Psi_n\rangle f_n\langle\Psi_n|\tilde{p}_l\rangle\frac{c_{l,A}}{\sum_m c^*_{m,A}c_{m,A}} \\ A_{A,B} &= \sum_{i,j} c^*_{i,A}\langle\phi_i|A|\phi_j\rangle c_{j,B}\end{aligned}$$

Thus if we choose a complete set of orthogonal vectors $\{c_{i,u}\}$, the total expectation value A is recovered by summing the individual contributions. For an particular atom and orbital, only the first partial wave is taken as default.

There is one important remark. The projection done here differs from the projection

$$\begin{aligned}P &= \frac{|\chi_u\rangle\langle\chi_u|}{\langle\chi_u|\chi_u\rangle} \\ &= |\phi_i\rangle\frac{\sum_{i,j} c^*_{i,u}c_{j,u}}{\sum_{k,l} c^*_{k,u}\langle\phi_k|\phi_l\rangle c_{l,u}}\langle\phi_j|\end{aligned}$$

that may be anticipated. In this case we would need a set of vectors normalized with the overlap between partial waves to obtain the complete result.

13.2.2 Matrix elements

Once the orbitals have been defined, we need to choose the matrix elements of the PDOS operator. Each matrix element is given a name, which allows us to use it in further operations. The matrix elements can be selected in two different ways:

- To select **diagonal elements** or sums of diagonal elements we use the block "!DCNTL!WEIGHT". This option allows us to plot the total density of states or the angular momentum weights of a particular orbital.

- We can also obtain **off-diagonal elements** using "!DCNTL!COOP" in order to analyze the bond order or overlap populations.

13.2.3 Operations

Projected charges, spin and eventually spin directions are printed to the protocol file. In order to access the data we have to define how they are to be represented.

- We can map the density of states on an energy grid and write the date to a file, so that they can be plotted.
- The values at a given energy can be printed to the protocol.
- The values of a given state can be written to a protocol.

13.3 Example for the control input file

```
!DCNTL
  !FILES
    !FILE ID='PDOS' EXT=F NAME='~/xx.pdos' !END
    !FILE ID='PROT' EXT=F NAME='stdout'  !END
  !END
  !ORBITAL NAME='SP3(C1-H2)'
    !ORB ATOM='C_1' TYPE='SP3'
         NNZ='H_2' FAC=1.0 !END
  !END
  !WEIGHT ID='TOTAL' TYPE='TOTAL' !END
  !WEIGHT ID='W(1)'
     !ORB NAME='SP3(C1-H2)' !END
  !END
  !WEIGHT ID='W(2)'
     !ATOM NAME='C_1' TYPE='ALL'!END
  !END
  !COOP   ID='W(3)'
    !ORB1  NAME='SP3(C1-H2)' !END
    !ORB2  ATOM='H_2' TYPE='S' !END
  !END
  !GRID  EMIN[EV]=-18. EMAX[EV]=0. DE[EV]=0.1
         BROADENING[EV]=0.1 !END
```

```
  !OUTPUT ID='W(1)' FILE='~/xx.sp2a' !END
  !OUTPUT ID='W(2)' FILE='~/xx.sp2b' !END
  !OUTPUT ID='W(3)' FILE='~/xx.sp2c' !END
!END
```

13.4 Argument description for the control input file

13.4.1 !DCNTL

Rules: optional
Description: defines the operations done on the system; largely independent of the system

13.4.2 !DCNTL!GRID

Rules: optional
Description: defines the energy grid and energy broadening

EMIN[EV]

lower bound in eV of the energy interval for density of number of states
Type: real
Rules: mandatory
Default: none

EMAX[EV]

upper bound in eV
Type: real
Rules: mandatory
Default: none

DE[EV]

spacing of the grid points
Type: real
Rules: mandatory
Default: none

BROADENING[EV]

density of states are broadened by a Gaussian of the form $\exp -(e/b)^2$, where b is the broadening.
Type: real
Rules: optional
Default: none

SCALEY

Allows to scale either the Number of States (NoS) or the Density of States (DoS) to the same height as the corresponding other function. Possible values are 'NONE', 'DOS', and 'NOS'. For 'NONE' no rescaling takes place. with 'DOS' the DoS is rescaled and for 'NOS', the NoS is rescaled.
Type: character
Rules: optional
Default: 'none'

13.4.3 !DCNTL!FILES

Rules: optional
Description: Specifies the file names that deviate from the standard values

13.4.4 !DCNTL!FILES!FILE

Rules: optional, multiple
Description: Specifies one file

ID

identifier for the file; options are:

'PROT' protocol file
Standard extension:'.dprot'

'ERR' error file
Standard extension:'.derr'

'PDOS' data file produced by the simulation code. Input for paw_dos tool
Standard extension:'.pdos'

'PDOSOUT' data files ready for plotting.
Standard extension:'.pdosout'

Type: character
Rules: mandatory
Default: none

NAME

filename. Can be the relative file name or an extension to the PAW "root". Standard output can be specified by NAME='stdout' and EXT=.false.
Type: character
Rules: mandatory
Default: none

EXT

.true.: NAME specifies the extension only/ .false.: full name
Type: logical
Rules: optional
Default: .false.

13.4.5 !DCNTL!ORBITAL

Rules: optional
Description: defines an orbital

NAME

names the newly defined orbital !DCNTL!ORBITAL!ORB
Type: character
Rules: mandatory
Default: none

13.4.6 !DCNTL!ORBITAL!ORB

Rules: optional, multiple
Description: select a predefined orbital from which the orbital is built

NAME

specifies an orbital previously specified by another block !DCNTL!ORBITAL!ORB
Type: character
Rules: optional; incompatible with atom and type
Default: none

ATOM

atom name on which the orbital resides
Type: character
Rules: optional; incompatible with name
Default: none

TYPE

orbital type. Can be one of the following. 'S', 'PX', 'PY', 'PZ', 'DXY', 'D3Z2-R2', 'DXZ', 'DYZ', 'DX2-Y2', 'SP', 'SP2', 'SP3'. The meaning of these orbitals is described in the preceding introduction to the paw_dos tool.

Type: character
Rules: optional; incompatible with name; mandatory if atom is specified.
Default: none

IMAG

selects the imaginary part of the wave function

Type: logical
Rules: optional
specified. Default: .false.

FAC

Orbital coefficient, with which it contributes the orbital to be built.

Type: real
Rules: optional
Default: 1.0

Z

vector defining the new z-direction to which TYPE refers

Type: real(3)
Rules: optional (overwritten by NNZ)
Default: 0.0,0.0,1.0

NNZ

atom name of the atom to which the vector defining the new z-direction points

Type: character
Rules: optional; overwrites Z
Default: see Z

X

vector defining the new xz-plane together with z to which TYPE refers
Type: real(3)
Rules: optional (overwritten by NNX)
Default: 1.0,0.0,0.0

NNX

atom name of the atom to which the vector defining the new x-direction points
Type: character
Rules: optional; overwrites X
Default: see X

13.4.7 !DCNTL!WEIGHT

Rules: optional,multiple
Description: defines orbital weights; three ways to specify the weights are possible

- specify “TYPE”; rather unspecific selections such as total density of states.
- specify one or several sub-blocks “!ATOM”; selects contributions from different atoms.
- specify one or several sub-blocks “!ORB”. Most specific version, identifying individual orbitals

The selection type is not compatible with the other two possibilities.

ID

defines the name for this set of matrix elements
Type: character

Rules: mandatory
Default: none

TYPE

specifies how the weights are defined; can be

“ALL” sum over all projected density of states

“EMPTY” weight of the wave function not considered in any projection

“TOTAL” total density of states

Type: character
Rules: optional; if type is specified, no blocks !atom or !orbs must be present
Default: none

13.4.8 !DCNTL!WEIGHT!ATOM

Rules: optional
Description: select a predefined orbital from which the orbital is built

NAME

atom name from which the contribution to this weight is selected
Type: character
Rules: mandatory
Default: none

TYPE

selects the weights from this atom. Can be

“ALL” sum over all projected density of states on this atom

“S” angular momentum weight for $\ell = 0$

“P” angular momentum weight for $\ell = 1$

“D” angular momentum weight for $\ell = 2$

Type: character
Rules: optional
Default: none

SPIN

selects the spin directions in noncollinear calculations. Can be

“TOTAL” total density

“X” spin density in x-direction

“Y” spin density in y-direction

“Z” spin density in z-direction

“MAIN” spin density in the main spin direction of this atom

Type: character
Rules: optional
Default: TOTAL

13.4.9 !DCNTL!WEIGHT!ORB

Rules: optional
Description: usage is same as !DCNTL!ORBITAL!ORB

13.4.10 !DCNTL!COOP

Rules: optional

Description: selects a sum of off-diagonal elements of the density-of-states operator. The off-diagonal elements are defined by all pairs of orbitals specified in !DCNTL!COOP!ORB1 on the one hand and in !DCNTL!COOP!ORB2 on the other hand.

ID

defines the name for this set of matrix elements

Type: character

Rules: mandatory

Default: none

13.4.11 !DCNTL!COOP!ORB1

Rules: optional, multiple

Description: selects an orbital for which the off-diagonal elements of the density of states operator with the orbitals selected by !DCNTL!COOP!ORB2 are to be calculated; usage is same as !DCNTL!ORBITAL!ORB

13.4.12 !DCNTL!COOP!ORB2

Rules: optional, multiple

Description: selects an orbital for which the off-diagonal elements of the density of states operator with the orbitals selected by !DCNTL!COOP!ORB1 are to be calculated; usage is same as !DCNTL!ORBITAL!ORB

13.4.13 !DCNTL!OUTPUT

Rules: optional,multiple

Description: selects output of the data; if neither E[EV] nor B are given, the density of states and the number of states are mapped onto the energy grid.
The format of the file produced is as follows: Each file have 5 columns with the following content.

1. energy in eV.
2. density of states (scaled fitting to the sum of states)
3. integrated density of states (number of states below a given energy).
4. density of states multiplied with actual occupations
5. integrated density of states multiplied with actual occupations (number of occupied states below a given energy).

In spin-polarized calculations, columns 2-5 refer to spin up. There are corresponding columns 6-9 for spin down electrons. In non-collinear calculations there are only 5 columns, which refer to the total density of states, unless a spin direction has been explicitely specified.

ID

defines the name for this set of matrix elements; refers to ID in block !DCNTL!WEIGHT or !DCNTL!COOP

Type: character
Rules: mandatory
Default: none

FILE

defines the file to which the result shall be written

Type: character
Rules: optional

Default: the file identified by 'pdosout'

B

band index
Type: integer
Rules: optional; incompatible with E[EV]
Default: none

E[EV]

energy
Type: real
Rules: optional; incompatible with B
Default: none

S

spin index
Type: integer
Rules: optional
Default: all spins

K

k-point index
Type: integer
Rules: optional
Default: all k-points

14 The Atomic-Setup program: “paw_atom”

The atomic-setup program performs a calculation for an isolated atom and determines the augmentation-transformation for this atom.

14.1 Command

The calling sequence is

`paw_atom` *name*

where *name* is a specifier for the setup.

14.2 Example for the control input file

```
!ACNTL
 !GENERIC ELEMENT='N ' RAUG=2.D0 RBOX=20.D0 !END
 !DFT     TYPE=1 !END
 !GRID    R1=1.056D-4 DEX=0.05 NR=250 !END
 !AECORE
   !STATE L=0 N=1 F=2 !END
 !END
 !VALENCE
   !STATE L=0 N=2 F=2. !END
   !STATE L=1 N=2 F=3. !END
   !STATE L=0 N=3 F=0. !END
 !END
 !VTILDE  TYPE='POLYNOMIAL' RC=1.0 POWER=3 POT(0)=-2.58 !END
 !COMPENSATE RC=0.3 !END
 !PSCORE  TYPE='POLYNOMIAL' RC=1.0 POWER=3 RHO(0)=0.05 !END
 !WAVE L=0 N=2       PSTYPE='HBS' RC=1.0 LAMBDA=6 !END
 !WAVE L=1 N=2       PSTYPE='HBS' RC=1.0 LAMBDA=6 !END
 !WAVE L=0 N=3       PSTYPE='HBS' RC=1.0 LAMBDA=6 !END
!END
!EOB
```

14.3 Argument description for the control input file ACNTL

14.3.1 !ACNTL

Rules: mandatory
Description:

14.3.2 !ACNTL!GENERIC

Rules: optional
Description:

ELEMENT

element symbol.
Type: character
Rules: mandatory
Default: none

RAUG

maximum radius where augmentation operates.
Type: real
Rules: mandatory
Default: none

RBOX

the atom sits in a spherical box with hard walls at r=RBOX.
Type: real
Rules: mandatory
Default: none

14.3.3 !ACNTL!DFT

Rules: optional
Description:

TYPE

functional type index. see “!CNTL!DFT”
Type: integer
Rules: mandatory
Default: none

14.3.4 !ACNTL!GRID

Rules: optional
Description: Specifies the logarithmic radial grid. It is recommended that the default parameters be left as they are, because the simulation program can only work with one grid for all setups.

R1

innermost grid point
Type: real
Rules: optional
Default: 1.056×10^{-4}

DEX

logarithmic spacing $\ln(r(i+1)/r(i))$.
Type: real
Rules: optional
Default: 0.05

NR

number of grid points

Type: integer
Rules: optional
Default: 250

14.3.5 !ACNTL!AECORE

Rules: optional
Description: Defines core states

14.3.6 !ACNTL!AECORE!STATE

Rules: optional
Description: Defines one state.

L

main angular momentum quantum number.
Type: integer
Rules: mandatory
Default: none

N

main radial quantum number.
Type: integer
Rules: mandatory
Default: none

F

Occupation. number of electrons in this state
Type: real
Rules: mandatory

Default: none

14.3.7 !ACNTL!VALENCE

Rules: optional
Description: Defines valence states

14.3.8 !ACNTL!VALENCE!STATE

Rules: optional
Description: Defines one state.

L

main angular momentum quantum number.
Type: intger
Rules: mandatory
Default: none

N

main radial quantum number.
Type: integer
Rules: mandatory
Default: none

F

Occupation. number of electrons in this state
Type: real
Rules: mandatory
Default: none

14.3.9 !ACNTL!VTILDE

Rules: optional
Description: Defines pseudo-protential. (Do not confuse with the pseudopotential of the pseudopotential method.)

TYPE

pseudization type. Can be “Polynomial” or ’HBS’.

- Polynomial uses a polynomial $\tilde{v}_{at}(r < r_c) = Ar^n + Br^{n+1}$ and $\tilde{v}_{at}(r > r_c) = v_{at}(r)$ so that it is differentiable and the value at the origin has the value specified by POT(0).
- HBS uses a cutoff function $k(r) = e^{-(\frac{r}{r_c})^\lambda}$ to obtain the pseudopotential as $\tilde{v}_{at}(r) = \tilde{v}_0 k(r) + [1 - k(r)]v_{at}(r)$.

Type: character
Rules: mandatory
Default: none

RC

for POLYNOMIAL: matching radius. For HBS: parameter r_c.
Type: real
Rules: mandatory
Default: none

POWER

for POLYNOMIAL: Order of the polynom used for pseudization. For HBS: Parameter λ.
Type: integer
Rules: mandatory
Default: none

POT(0)

Value of the pseudo potential at the origin

Type: real
Rules: mandatory
Default: none

14.3.10 !ACNTL!COMPENSATE

Rules: optional
Description: Defines compensation density.

RC

Gaussian decay parameter $g = \exp(-(\frac{r}{r_c})^2)$.
Type: real
Rules: optional
Default: 0.3

14.3.11 !ACNTL!PSCORE

Rules: optional
Description: Defines pseudo core.

TYPE

pseudization type. Can be “Polynomial”.
Type: character
Rules: mandatory
Default: none

RC

matching radius.
Type: real
Rules: mandatory

Default: none

POWER

Order of the polynom used for pseudization
Type: integer
Rules: mandatory
Default: none

POT(0)

Value of the pseudo potential at the origin
Type: real
Rules: mandatory
Default: none

14.3.12 !ACNTL!WAVE

Rules: optional
Description: Defines pseudo wave function. The order of the states is relevant. (Increasing n and within each n increasing ℓ)

L

main angular momentum quantum number.
Type: integer
Rules: mandatory
Default: none

N

main radial quantum number. Use only for bound states.
Type: integer
Rules: mandatory if E is not specified. Must not be specified if E is specified.
Default: none

E

Energy at which the partial waves are obtained.
Type: integer
Rules: mandatory if N is not specified. Must not be specified if N is specified.
Default: none

PSTYPE

pseudization type. Can be "HBS". The potential is varied by a potential $\exp(-(r/r_c)^\lambda$ with a variable scaling factor.
Type: character
Rules: mandatory
Default: none

RC

matching radius.
Type: real
Rules: mandatory
Default: none

LAMBDA

Type: integer
Rules: mandatory
Default: none

15 The Structure Pre-Optimization Tool: paw_preopt

The paw_preopt tool allows to perform a fast but inaccurate structure pre-optimization based on force-field molecular dynamics. It reads the structure input file (`case.strc`) and tries to find bonds in the given structure using the covalent radii of the atoms. Then it tries to find suitable parameterizations for the atoms using the universal force field (UFF) [36]. (See also section 20.) Bonds and force field parameterizations can also be provided via the input files. Atoms to be frozen can be specified.

Using the parameterization it optimizes the geometry (currently) using the conjugate gradient line search algorithm which is also used in QM-MM coupling in the main simulation program. Atom positions after optimization are printed to the protocol file and may be used as starting point for a PAW simulation.

Atoms can be frozen during the optimization. No other constraints can be applied. Freezing can either be done via the keywords !PCNTL!FREEZEONLY and !PCNTL!MOVEONLY or via the keyword FREEZE=T in the !ATOM block of the structure input file. The latter will be ignored by the main PAW simulation program.

15.1 Command

The calling sequence is

`paw_preopt.x` *controlfile*

where *controlfile* is the file name of the control input file for the paw_preopt tool. I recommend the extension ".pcntl".

15.2 Example for the control input file

```
!PCNTL
  !FILES
    !FILE ID='xyz' NAME='optimized.xyz' !END
  !END
  !GENERIC
    TRACE=F
    TOL=0.0001
    NSTEPS=10000
  !END
```

```
  !OUTPUT
     PRINTATOMS=T
     PRINTBONDS=T
     WARNFF=T
     XYZOUT=T
  !END
  !FREEZEONLY
    COMMENT='Not used if MOVEONLY is present.'
    !ATOM NAME='MO1' !END
    !ATOM PART='FE' !END
  !END
  !MOVEONLY
     !ATOM_OFF PART='H_nh' !END
     !ATOM NAME='N_nh41' !END
     !ATOM NAME='H_nh43' !END
     !ATOM NAME='H_nh44' !END
     !ATOM NAME='H_nh45' !END
  !END
!END
!EOB
```

Please note that a keyword COMMENT is ignored by all PAW programs. Thus it can be used to add comments to the input files.

15.3 Argument description for the control input file

15.3.1 !PCNTL

Rules: mandatory
Description: defines the operations done on the system; largely independent of the system

15.3.2 !PCNTL!GENERIC

Rules: optional
Description: defines optimization parameters (and use of the trace)

TRACE

writes information on the calls of subroutines to standard output
Type: logical
Rules: optional
Default: F

TOL

tolerance of the maximal component of the force on atoms in the convergence cycles
Type: real
Rules: optional
Default: 10^{-4}

NSTEPS

maximal number of steps for the convergence. After that number the structure optimization is stopped no matter if the forces have become smaller than TOL. A statement on the convergence will be written to the protocol.
Type: integer
Rules: optional
Default: 1000

15.3.3 !PCNTL!FILES

Rules: optional
Description: specifies the file names that deviate from the standard values

15.3.4 !PCNTL!FILES!FILE

Rules: optional, multiple
Description: Specifies one file

ID

identifier for the file; options are:

'PROT' protocol file
Standard extension:'.pprot'

'STRC' The structure file as used by the simulation code. Used for the input structure.
Standard extension:'.strc'

'XYZ' data file with the resulting optimized structure
Standard extension:'.xyz'

Type: character
Rules: mandatory
Default: none

NAME

filename. Can be the relative file name or an extension to the PAW "root". Standard output can be specified by NAME='stdout' and EXT=.false.
Type: character
Rules: mandatory
Default: none

EXT

T: NAME specifies the extension only. F: full name
Type: logical
Rules: optional
Default: F (false)

15.3.5 !PCNTL!OUTPUT

Rules: optional
Description: specifies which information should be written to the protocol

PRINTATOMS

information on the used atom parameterization and freezing is written to the protocol
Type: logical
Rules: optional
Default: T

PRINTBONDS

information on the used bonds is written to the protocol
Type: logical
Rules: optional
Default: T

WARNFF

paw_preopt tries to find best-suiting force fields for the atoms. However, in some cases it cannot decide which parameterization will be the best. If T in these cases a warning will be printed to the protocol.
Type: logical
Rules: optional
Default: T

XYZOUT

should an xyz-file of the resulting structure be created?
Type: logical
Rules: optional
Default: T

15.3.6 !PCNTL!FREEZEONLY

Rules: optional, not used if !PCNTL!MOVEONLY is present
Description: sets freezing of all atoms but those specified in this block to false. May be overwritten for single atoms by !STRC!ATOM:FREEZE

NAME

specifies an atom name
Type: character
Rules: optional, multiple
Default: none

PART

Specifies a part of an atom name. All atoms containing this string are used. Case is ignored.
Type: character
Rules: optional, multiple
Default: none

15.3.7 !PCNTL!MOVEONLY

Rules: optional, overwrites !PCNTL!FREEZEONLY
Description: sets freezing of all atoms but those specified in this block to true. May be overwritten for single atoms by !STRC!ATOM:FREEZE

NAME

specifies an atom name
Type: character
Rules: optional, multiple
Default: none

PART

Specifies a part of an atom name. All atoms containing this string are used. Case is ignored.

Type:	character
Rules:	optional, multiple
Default:	none

15.4 Argument description for the structure input file

The atomic structure is read from a structure input file which has the same syntax as for the main simulation code. However, a few keywords are used which are ignored by the main simulation code. The keyword !STRC!GENERIC:LUNIT is mandatory.

15.4.1 !STRC!ATOM

Rules: mandatory,multiple

Description: usage is same as in the main simulation code. Keywords R, SP and NAME are used. Additional keywords are:

FFTYPE

Force field type used for this atom. This keyword can be used to overwrite the suggestion of the program. Possible force fields can be found in section 20 on page **??**.

Type:	character
Rules:	optional
Default:	depending on the atom type and the number of bonds to that atom

FREEZE

freeze this atom

Type:	logical
Rules:	optional
Default:	F

15.4.2 !STRC!BOND

Rules: optional,multiple

Description: Overwrite the suggestion of the program for bonds between the atoms. Not yet implemented.

16 Tools

16.1 The Energy Analysis Tool: “paw_show”

The tool paw_show allows one to inspect the protocol file. It plots the total energy, the instantaneous temperature of the atoms, the fictitious kinetic energy of the wave functions, the conserved energy versus the time step in picoseconds. This tool requires the xmgr plot library[14].

16.1.1 Command

The calling sequence is

`paw_show` *option rootname file*

where *rootname* is the name of the protocol file without the “.prot” ending and *option* is one of the following:

-f fictitious kinetic energy of the wave functions

-e the static total energy

-c the conserved energy

-t temperature of the atoms in Kelvin

-ar friction parameter for the atoms

-ap friction parameter for the wave functions

-h prints information about usage of the tool

? prints information about usage of the tool

The optional parameter *file* the file to which the data are written. If *file* is absent, the data are viewed.

16.2 Copy a project: “paw_copy”

paw_copy makes a copy of a cp-paw project with a new rootname.

16.2.1 Command

The calling sequence is

`paw_copy` *root1 root2*

where *root1* is the root name of the project to be copied and root2 is the target foot name.

Specifically, it takes all files beginning with *root1* in the current directory and copies the files with the same name but *root1* replaced by *root2*.

16.3 collect energies “paw_collect”

paw_collect collects the last total energies from all protocoll files in the current directory.

16.3.1 Command

The calling sequence is

```
paw_collect
```

17 Specifications of formatted and unformatted output data files

17.1 Protocoll

One-particle energies: The eigenvalues of the Hamiltonian in the basis of the wave function are printed, unless dynamical occupations are selected (i.e. `!CONTROL!MERMIN`). In this latter case, the diagonal elements of the Hamiltonian are printed. The one-particle energies are printed in eV. The occupations are printed without the geometric weight, but include spin degeneracy, so that for a non-spin polarized calculation the maximum allowed occupation is two.

17.2 waveplot

The file waveplot is used to provide a wave function or a density on a real space grid. From there it has to be converted into a format ready for visualization, which can be done using the "waveplot" tool. A waveplot file is created by selecting "!CNTL!ANALYSE!WAVE" in the control input file.

The file is written as follows:

```
CHARACTER(*) :: TITLE     ! user defined comment
INTEGER(4)   :: NAT       ! number of atoms
REAL(8)      :: RBAS(3,3) ! lattice vectors (xyzxyzxyz)
REAL(8)      :: POS(3,NAT)! atomic positions
REAL(8)      :: Z(NAT)    ! atomic numbers
REAL(8)      :: Q(NAT)    ! point charges
CHARACTER(32):: NAME(NAT) ! atom name (see file strc_out)
                          ! number of grid points along
INTEGER(4)   :: NR1       !        first direction
INTEGER(4)   :: NR2       !        second
INTEGER(4)   :: NR3       !        third
REAL(8)      :: WAVE(NR1,NR2,NR3) ! density or wave function
                                  ! on the real space grid
WRITE(UNIT)'WAVEPLOT',LEN(TITLE)
WRITE(UNIT)TITLE
WRITE(UNIT)RBAS,NAT
WRITE(UNIT)NR1,NR2,NR3
WRITE(UNIT)NAME
WRITE(UNIT)Z
```

```
WRITE(UNIT)POS
WRITE(UNIT)Q
WRITE(UNIT)WAVE
WRITE(UNIT)'END OF FILE'
```

The coordinates of the density or wave with (i_1, i_2, i_3) are

$$x_i = \sum_{i=1}^{3} \mathrm{RBAS}_{i,j} \frac{i_i - 1}{\mathrm{NR}i} \tag{6}$$

17.3 Projected density of states files

```
INTEGER(4)  :: NAT   !#(ATOMS)
INTEGER(4)  :: NSP   !#(ATOM TYPES)
INTEGER(4)  :: NKPT  !#(K-POINTS)
INTEGER(4)  :: NSPIN !#(SPINS)
INTEGER(4)  :: NDIM  !#(SPINOR COMPONENTS)
INTEGER(4)  :: NPRO  !#(PROJECTIONS PER STATE)
INTEGER(4)  :: LNXX  !MAX#(PARTIAL WAVES PER ATOM (L,N))
INTEGER(4)  :: LNX(NSP)      !#(PARTIAL WAVES PER ATOM (L,N))
INTEGER(4)  :: LOX(LNXX,NSP) !MAIN QUANTUM NUMBER
INTEGER(4)  :: ISPECIES(NAT) !ATOM TYPE
REAL(8)     :: RBAS(3,3)     !LATTICE VECTORS
REAL(8)     :: R0(3,NAT)     !ATOMIC POSITIONS
INTEGER(4)  :: IZ            !ATOMIC NUMBER
REAL(8)     :: RAD           !''ASA RADIUS''
REAL(8)     :: VAL(LNXX)     ! AEPHI(RAD)
REAL(8)     :: DER(LNXX)     ! DAEPHI/DR(RAD)
REAL(8)     :: OV(LNXX,LNXX) ! <AEPHI|AEPHI> WITHIN RAD
REAL(8)     :: XK(3,NKPT)    ! K-POINT IN RELATIVE COORDINATES
INTEGER(4)  :: NB            ! #(BANDS)
REAL(8)     :: EIG(NB,NSPIN,NKPT)              ! EIGENVALUES
COMPLEX(8)  :: PROJ(NDIM,NPRO,NB,NSPIN,NKPT)   ! PROJECTIONS
!*************************************************************
WRITE(NFIL)NAT,NSP,NKPT,NSPIN,NDIM,NPRO,LNXX
WRITE(NFIL)LNX(:),LOX(:,:),ISPECIES(:)
WRITE(NFIL)RBAS(:,:),R0(:,:)
DO ISP=1,NSP
   WRITE(NFIL)IZ,RAD,VAL(1:LNX(ISP)),DER(1:LNX(ISP)) &
&                                   ,OV(1:LNX(ISP),1:LNX(ISP))
ENDDO
```

```
DO IKPT=1,NKPT
  DO ISPIN=1,NSPIN
    WRITE(NFIL)XK(:,IKPT),NB
    DO IB=1,NB
      WRITE(NFIL)EIG,PROJ(:,:)
    ENDDO
  ENDDO
ENDDO
```

17.4 Trajectory files

Trajectory files are used to monitor variables step by step.

The file is written as follows:

```
INTEGER(4)  :: ISTEP1    !
INTEGER(4)  :: ISTEP2
INTEGER(4)  :: LEN
REAL(8)     :: TIME(ISTEP1:ISTEP2)
REAL(8)     :: ARRAY(LEN,ISTEP1:ISTEP2)
!
DO ISTEP=ISTEP1,ISTEP2
  WRITE(UNIT)ISTEP,TIME(ISTEP),LEN,ARRAY(:,ISTEP)
ENDDO
```

Here "istep" is the number of the time step, consistent with the protocol file.

17.4.1 Position Trajectory

In the position trajectory "*root*_r.tra". The length of each item is $LEN = 9+8*N$, where N is the number of atoms. The data on "ARRAY" are arranged as follows:

1. $\mathbf{T}$: lattice vectors (3×3 data)
2. $\vec{R}$: atomic positions ($3 \times N$ data)
3. Q: point charges (N) as obtained from Gaussian fit (N data)
4. $q, \vec{m}$ Charges and moments as obtained from integrating the one-center density over a "atomic sphere" radius. ($4 \times N$ data)

18 Units and constants

The program works in Cartesian coordinates using Hartree atomic units

$$\hbar = e = m_e = 4\pi\epsilon_0 = 1\ . \tag{7}$$

Angles are given in radian (2π radian $= 360$ deg).

The following list is produced by the CONSTANTS object. The data are based on a values recommended by The Commitee on Data for Science and Technology (CODATA) [13].

```
UNITS AND CONSTANTS IN ATOMIC HARTREE UNITS:
--------------------------------------------
     NAME           VALUE        :DESCRIPTION
PI............=    3.141593     :PI
HBAR..........=    1.000000     :A.U. FOR ANGULAR MOMENTUM
E.............=    1.000000     :ELEMENTARY CHARGE
ME............=    1.000000     :ELECTRON MASS
EPSILON0.....=    7.957747E-02 :PERMITIVITY OF VACUUM
ALPHA.........=    7.297353E-03 :FINE STRUCTURE CONSTANT
ABOHR.........=    1.000000     :BOHR RADIUS
TAU0..........=    1.000000     :A.U. FOR TIME
HARTREE.......=    1.000000     :HARTREE ENERGY UNIT
C.............=  137.035989     :SPEED OF LIGHT
MU0...........=    6.691764E-04 :PERMEABILITY OF VACCUM
BOHRMAGNETON...=    0.500000     :BOHR MAGNETON
NUCLEARMAGNETON=    2.723085E-04 :NUCLEAR MAGNETON
GE............=    2.002319     :ELECRON GYROMAGNETIC RATIO
KG............=    1.097768E+30 :KILOGRAMM
SECOND........=    4.134137E+16 :SECOND
METER.........=    1.889726E+10 :METER
AMPERE........=  150.974819     :AMPERE
MOL...........=    6.022137E+23 :MOLE
KB............=    3.166679E-06 :BOLTZMANN CONSTANT IN A.U.
NEWTON........=    1.213779E+07 :NEWTON
JOULE.........=    2.293710E+17 :JOULE
COULOMB.......=    6.241506E+18 :COULOMB
VOLT..........=    3.674931E-02 :VOLT
TESLA.........=    4.254381E-06 :TESLA
GAUSS.........=    4.254381E-10 :GAUSS
RY............=    0.500000     :RYDBERG ENERGY UNIT
```

```
EV.............=    3.674931E-02 :ELECTRON VOLT
ANGSTROM.......=    1.889726     :ANGSTROM
KJ/MOL.........=    3.808798E-04 :KILOJOULE PER MOLE
KCAL/MOL.......=    1.593601E-03 :KILOCALORIE PER MOLE
U..............=    1.822889E+03 :atomic MASS UNIT
DEBYE..........=    0.393430     :DEBYE


CONVERSION FACTORS AN CONSTANTS:
--------------------------------
--------- ATOMIC UNITS ---------------------------------
QE.....................=    1.000000     :A.U.
HBAR...................=    1.000000     :A.U.
ME.....................=    1.000000     :A.U.
HARTREE................=    1.000000     :A.U.
ABOHR..................=    1.000000     :A.U.
1/ALPHA................=  137.035989     :
EPSILON0...............=    7.957747E-02 :
MU0....................=    6.691764E-04 :
C......................=  137.035989     :ABOHR/TAU0
KB.....................=    3.166679E-06 :HARTREE
--------- CONVERSION FROM SI TO ATOMIC UNITS -----------
MOL....................=    6.022137E+23 :
METER..................=    1.889726E+10 :ABOHR
SECOND.................=    4.134137E+16 :TAU0
JOULE..................=    2.293710E+17 :HARTREE
KG.....................=    1.097768E+30 :ME
C......................=    2.997925E+08 :METER/SECOND
--------- CONVERSION FROM ATOMIC TO SI UNITS -----------
ABOHR..................=    5.291772E-11 :METER
TAU0...................=    2.418884E-17 :SECOND
ME.....................=    9.109390E-31 :KG
E......................=    1.602177E-19 :COULOMB
HBAR...................=    1.054573E-34 :JOULE*SECOND
--------- LENGTH CONVERSIONS ---------------------------
ANGSTROM...............=    1.889726     :ABOHR
ABOHR..................=    0.529177     :ANGSTROM
ABOHR..................=    5.291772E-11 :METER
METER..................=    1.889726E+10 :ABOHR
--------- TIME CONVERSIONS -----------------------------
```

```
TAU0..................=    2.418884E-17 :SECOND
TAU0..................=    2.418884E-02 :FEMTO SECOND
PICO SECOND...........=    4.134137E+04 :TAU0
--------- ENERGY CONVERSIONS --------------------------
273.15 KB.............=    2.353727E-02 :EV
EV....................=   96.485309     :KJ/MOL
HARTREE...............=    2.625500E+03 :KJ/MOL
EV....................=   23.060542     :KCAL/MOL
HARTREE...............=  627.509556     :KCAL/MOL
KJ/MOL................=    1.036427E-02 :EV
KCAL/MOL..............=    4.336411E-02 :EV
HARTREE...............=   27.211396     :EV
RY....................=    0.500000     :HARTREE
--------- MASS CONVERSIONS ----------------------------
U.....................=    1.660540E-27 :KG
U.....................=    1.822889E+03 :ME
--------- CHARGE CONVERSIONS --------------------------
QE....................=    1.602177E-19 :COULOMB
COULOMB...............=    6.241506E+18 :QE
--------- OTHER CONVERSIONS ---------------------------
DEBYE.................=    0.393430     :E*ABOHR
E*ABOHR...............=    2.541748     :DEBYE
E_ZPV/T[PSEC].........=    7.599149E-05 :HARTREE
E_ZPV/T[PSEC].........=    2.067835E-03 :EV
E_ZPE/(1/LAMBDA)[CM**-1=    6.199212E-05 :EV
E_ZPE/(1/LAMBBDA)[CM**-=    2.278168E-06 :HARTREE
(1/LAMBDA)............=   33.356410     :CM**-1/T[PSEC]

Scale factors :
---------------
FEMTO.........=    1.000000E-15 :FEMTO=1.D-15
PICO..........=    1.000000E-12 :PICO=1.D-12
NANO..........=    1.000000E-09 :NANO=1.D-9
MICRO.........=    1.000000E-06 :MICRO=1.D-6
MILLI.........=    1.000000E-03 :MILLI=1.D-3
KILO..........=    1.000000E+03 :KILO=1.D+3
MEGA..........=    1.000000E+06 :MEGA=1.D+6
GIGA..........=    1.000000E+09 :GIGA=1.D+9
TERA..........=    1.000000E+12 :TERA=1.D+12
PETA..........=    1.000000E+15 :PETA=1.D+15
```

18.1 Vibrations

Vibrational freqencies and optical absorption energies are frequenty measured via the frequency or wave-length of the absorbed light. The frequency or energy is measured in wave-numbers $\bar{\nu} = 1/L$ defined as the inverse of the wavelength L. (The wave number is the number of times an oscillation of a light wave fitswithin a given length interval.) It is measured in units of cm^{-1}. Here we provide the conversion of wave numbers into energies, frequencies, etc.

First we convert the wave number into a frequency $\omega = 2\pi/T$ where T is the time of a full oscillation.

We use the dispersion relation of light $\omega = c|k|$, where c is the speed of light. $\omega = \frac{2\pi}{T}$ is the frequency and $|\vec{k}| = \frac{2\pi}{L}$ is the absolute value of the wave vector. T is the time for one full period at a given position and L is the length for one full period at a given time.

From the dispersion relation we calculate the the frequency $\omega = 2\pi/T$ and the

$$
\begin{aligned}
|k| &= \frac{2\pi}{\bar{\nu}} && (8)\\
L &= \frac{1}{\bar{\nu}} && (9)\\
\omega &= c|k| = c2\pi/L = 2\pi c\bar{\nu} && (10)\\
T &= \frac{2\pi}{\omega} = \frac{1}{c\bar{\nu}} && (11)
\end{aligned}
$$

The connection to the energy is given by $E = \hbar\omega$, where E is the energy of the absorbed or emmited photon with frequency ω. It is the energy difference between the energy levels of the corresponding transition.

$$
E = \hbar\omega = 2\pi\hbar c\bar{\nu} \qquad (12)
$$

Finally we would like to know the force constant C for a harmonic oscillator producing these level spacings.

$$
\begin{aligned}
M\ddot{x} &= -CX \qquad \Rightarrow \qquad \omega = \sqrt{\frac{C}{M}}\\
C &= M\omega^2 = M4\pi^2c^2\bar{\nu}^2 && (13)
\end{aligned}
$$

Let us now evaluate the conversion factors

$$
\begin{aligned}
T[ps] &\overset{\text{Eq. 11}}{=} \frac{1}{c[a_0/\tau_0]\bar{\nu}[cm^{-1}]}\frac{\tau_0}{ps}\cdot\frac{cm}{a_0}\\
&= \left(\frac{1}{137.035989}\right)\cdot\left(\frac{1}{10^{-12}\cdot 4.134137\times 10^{16}}\right)\cdot\left(10^{-2}\cdot 1.889726\times 10^{10}\right)\frac{1}{\bar{\nu}[cm^{-1}]}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1.889726}{1.37035989 \cdot 4.134137} \cdot 10^{-2+12-16-2+10} \frac{1}{\bar{\nu}[cm^{-1}]} \\
&= 0.3335640485 \times 10^{2} \frac{1}{\bar{\nu}[cm^{-1}]} = 33.35640485 \frac{1}{\bar{\nu}[cm^{-1}]} \\
T[a.u.] &= 1.379 \times 10^{6} \frac{1}{\bar{\nu}[cm^{-1}]}
\end{aligned}
$$

Note that we can read $T[ps]$ as T/ps if we ps is the value of a pico-second in the unit system of choice.

The energy is obtained from Eq. 12 as

$$
\begin{aligned}
E[a.u.] &= \hbar\omega = 2\pi\hbar[a.u.]c[a.u.]\frac{a_0}{m}10^2\bar{\nu}[cm^{-1}] \\
&= 2\pi \times 137.035989 \frac{1}{1.889726 \times 10^{10}} 10^2\bar{\nu}[cm^{-1}] \\
&= 2\pi \times 137.035989 \frac{1}{1.889726 \times 10^{10}} 10^2\bar{\nu}[cm^{-1}] \\
&= 4.556335 \times 10^{-6}\bar{\nu}[cm^{-1}] \\
E[eV] &= 1.23984242 \times 10^{-4}\bar{\nu}[cm^{-1}]
\end{aligned}
$$

As crosscheck we may use that the ground state of the hydrogen atom lies at $-\frac{1}{2}$ H=-13.6 eV corresponds to $\bar{\nu}$=-109678 cm^{-1}. This number is the Rydberg constant. (see Demtröder, Experimentalphysik 3; Springer Verlag p.102)

Here i summarize the main vibrational modes

Vibr. Mode	$\bar{\nu}[cm^{-1}]$
H-H Stretch	$\approx$ 4000
C-H Stretch	2800-3999
C-H Bend	1400-1500
C-C Stretch	1500-1750

The hydrogen stretch vibration has a period of about 350 τ_0 or 8.3 fs. The optical phonon in bulk silicon has a frequency of about 14 THz, which corresponds to $\bar{\nu} = 450 - 500$ cm^{-1}.

18.2 Magnetic hyperfine parameters

The hyperfine parameters measure the splitting of several transitions, each of which switches the spin of an electron, and preserves the spin of the nucleus. The energy of a nucleus in the magnetic field of the electronic spin density is

$$E = m^N B^N$$

where m^N is the magnetic moment of the nucleus, and B^N is the hyperfine field created by the electronic spin density. (The hyperfine field is a kind of effective magnetic field). When the electron spin is flipped by an electronic transition, the transition energy is $\hbar\omega = \hbar\omega_0 + 2m^N B^N$. For different spin quantum numbers S_z^N, we obtain transitions

$$\hbar\omega(S_z^N) = \hbar\omega_0 + 2\frac{m^N}{S^N}B^N S_z^N \tag{14}$$

from which follows the splitting of neighboring levels

$$\Delta E = \Delta\hbar\omega(S_z^N) = 2\hbar\frac{m^N}{S^N}B^N \tag{15}$$

with S^N being the nuclear spin with values $0, 1/2\hbar, \hbar, ..$, depending on the nucleus.

The quantity $\gamma_N = m^N/S^N$ is called the gyromagnetic ratio of the nucleus. The nuclear magnetic moment is typically given units of nuclear magnetons $\mu_N = (m_e/m_p)\mu_B$, where m_e is the electron mass, m_p is the proton mass, and $\mu_B = \frac{e\hbar}{2m_e}$ is the Bohr magneton.

In the literature the hyperfine splitting is often converted into a frequency ν using the relation $\Delta E = h\nu$. The frequency is expressed typically in MHz. Another term used for MHz is Mc/s, i.e. megacycles per second.

$$\begin{aligned}
\nu &= \frac{\Delta E}{2\pi\hbar} = \frac{m^N B^N}{\pi S^N} \\
&= \frac{m^N[\mu_N]B^N[\mathrm{T}]}{S^N[\hbar]} \times \frac{\mu_N \mathrm{T}}{\pi\hbar \mathrm{MHz}}\ \mathrm{MHz} \\
&= \frac{m^N[\mu_N]B^N[\mathrm{T}]}{S^N[\hbar]} 15.24518048\ \mathrm{MHz}
\end{aligned}$$

The frequency is often converted in wavenumbers using the relation $\frac{1}{l} = \frac{\nu}{c}$.

$$\begin{aligned}
\frac{1}{l} &= \frac{m^N[\mu_N]B^N[\mathrm{T}]}{S^N[\hbar]} \times \frac{\mu_N \mathrm{T} 10^{-2}\mathrm{m}}{\pi c\hbar}\ \mathrm{cm}^{-1} \\
&= \frac{m^N[\mu_N]B^N[T]}{S^N[\hbar]} \times 5.085135181 \times 10^{-4}\ \mathrm{cm}^{-1}
\end{aligned}$$

The energy of the magnetic hyperfine splitting is often converted into a effective magnetic field B^e acting on the electrons via

$$\Delta E = g_e \mu_B B^e$$

where μ_B is the Bohr magneton, g_e is the g-factor of the free electron. While the g-factor of the electron is slightly larger than 2, its value is set exactly to 2. The term $g_e\mu_B$ is twice

the magnetic moment of the electron.

$$\begin{aligned} B^e &= \frac{\Delta E}{g_e \mu_B} = \frac{2\hbar m_N B_N}{s_N g_e \mu_B} \\ &= \frac{m_N[\mu_N] B_N[\mathrm{T}]}{s_N[\hbar]} \times \frac{2\hbar \mu_N T}{\hbar 2 \mu_B \mathrm{T}}\ \mathrm{T} \\ &= \frac{m_N[\mu_N] B_N[\mathrm{T}]}{s_N[\hbar]} \times 0.544617\ \mathrm{mT} \end{aligned}$$

The magnetic field is either given in milliTesla, in Oersted or in Gauss (1 Oe=1 gauss=10^{-4} T).

19 Periodic Table

Atomic masses according to IUPAC 1985 (Pure Appl. Chem. 1985, 58, 1677-1692). Length units are in atomic units. "R(COV)" is the covalent radius. "R(ASA)" is the covalent radius scaled up by about 10%, consistent with the scaling between touching and volume filling spheres in an fcc solid. "R(VDW)" is the Van der Waals radius [36].

SY	Z	MASS[U]	R(COV)	R(ASA)	R(VDW)	CONFIGURATION	#(NODES)
H	1	1.008	0.605	0.668	5.454	[0]S1	
HE	2	4.003	1.757	1.943	4.464	[0]S2	
LI	3	6.941	2.324	2.569	4.632	[HE]S1	S1
BE	4	9.012	1.701	1.880	5.187	[HE]S2	S1
B	5	10.811	1.550	1.713	7.716	[HE]S2P1	S1
C	6	12.110	1.455	1.608	7.277	[HE]S2P2	S1
N	7	14.007	1.417	1.567	6.916	[HE]S2P3	S1
O	8	15.999	1.380	1.525	6.614	[HE]S2P4	S1
F	9	18.998	1.361	1.504	6.357	[HE]S2P5	S1
NE	10	20.180	1.342	1.483	6.128	[HE]S2P6	S1
NA	11	22.990	2.910	3.217	5.637	[NE]S1	S2P1
MG	12	24.305	2.570	2.841	5.709	[NE]S2	S2P1
AL	13	26.982	2.230	2.465	8.502	[NE]S2P1	S2P1
SI	14	28.086	2.098	2.319	8.116	[NE]S2P2	S2P1
P	15	30.974	2.003	2.214	7.837	[NE]S2P3	S2P1
S	16	32.066	1.928	2.131	7.625	[NE]S2P4	S2P1
CL	17	35.453	1.871	2.068	7.459	[NE]S2P5	S2P1
AR	18	39.948	1.852	2.047	7.309	[NE]S2P6	S2P1
K	19	39.098	3.836	4.240	7.204	[AR]S1	S3P2
CA	20	40.078	3.288	3.634	6.423	[AR]S2	S3P2
SC	21	44.956	2.721	3.008	6.227	[AR]S2D1	S3P2
TI	22	47.880	2.494	2.757	6.000	[AR]S2D2	S3P2
V	23	50.942	2.305	2.548	5.941	[AR]S2D3	S3P2
CR	24	51.996	2.230	2.465	5.713	[AR]S1D5	S3P2
MN	25	54.938	2.211	2.444	5.595	[AR]S2D5	S3P2
FE	26	55.847	2.211	2.444	5.503	[AR]S2D6	S3P2
CO	27	58.933	2.192	2.423	5.427	[AR]S2D7	S3P2
NI	28	58.340	2.173	2.402	5.355	[AR]S2D8	S3P2
CU	29	63.546	2.211	2.444	6.605	[AR]S1D10	S3P2
ZN	30	65.390	2.362	2.611	5.221	[AR]S2D10	S3P2
GA	31	60.723	2.381	2.632	8.283	[AR]S2P1D10	S3P2
GE	32	72.610	2.305	2.548	8.088	[AR]S2P2D10	S3P2

SY	Z	MASS[U]	R(COV)	R(ASA)	R(VDW)	CONFIGURATION	#(NODES)
AS	33	74.922	2.268	2.507	7.994	[AR]S2P3D10	S3P2
SE	34	78.960	2.192	2.423	7.946	[AR]S2P4D10	S3P2
BR	35	79.904	2.154	2.381	7.916	[AR]S2P5D10	S3P2
KR	36	83.800	2.116	2.339	7.825	[AR]S2P6D10	S3P2
RB	37	85.468	4.082	4.512	7.774	[KR]S1	S4P3D1
SR	38	87.620	3.609	3.990	6.880	[KR]S2	S4P3D1
Y	39	88.906	3.061	3.384	6.321	[KR]S2D1	S4P3D1
ZR	40	91.224	2.740	3.029	5.904	[KR]S2D2	S4P3D1
NB	41	92.906	2.532	2.799	5.981	[KR]S1D4	S4P3D1
MO	42	95.940	2.457	2.715	5.767	[KR]S1D5	S4P3D1
TC	43	97.907	2.400	2.653	5.665	[KR]S2D5	S4P3D1
RU	44	101.070	2.362	2.611	5.599	[KR]S1D7	S4P3D1
RH	45	102.906	2.362	2.611	5.535	[KR]S1D8	S4P3D1
PD	46	106.420	2.419	2.674	5.478	[KR]D10	S4P3D1
AG	47	107.868	2.532	2.799	5.949	[KR]S1D10	S4P3D1
CD	48	112.411	2.797	3.091	5.382	[KR]S2D10	S4P3D1
IN	49	114.818	2.721	3.008	8.434	[KR]S2P1D10	S4P3D1
SN	50	118.710	2.665	2.945	8.300	[KR]S2P2D10	S4P3D1
SB	51	121.757	2.646	2.924	8.353	[KR]S2P3D10	S4P3D1
TE	52	127.600	2.570	2.841	8.447	[KR]S2P4D10	S4P3D1
I	53	126.904	2.513	2.778	8.504	[KR]S2P5D10	S4P3D1
XE	54	131.290	2.476	2.736	8.322	[KR]S2P6D10	S4P3D1
CS	55	132.905	4.441	4.909	8.536	[XE]S1	S5P4D2
BA	56	137.327	3.742	4.136	6.998	[XE]S2	S5P4D2
LA	57	138.906	3.194	3.530	6.656	[XE]S2D1	S5P4D2
CE	58	140.115	3.118	3.446	6.720	[XE]S2D1F1	S5P4D2
PR	59	140.908	3.118	3.446	6.814	[XE]S2F3	S5P4D2
ND	60	144.240	3.099	3.426	6.756	[XE]S2F4	S5P4D2
PM	61	145.000	3.080	3.405	6.703	[XE]S2F5	S5P4D2
SM	62	150.360	3.061	3.384	6.652	[XE]S2F6	S5P4D2
EU	63	151.965	3.496	3.864	6.601	[XE]S2F7	S5P4D2
GD	64	157.250	3.042	3.363	6.365	[XE]S2F8	S5P4D2
TB	65	158.925	3.005	3.321	6.521	[XE]S2F9	S5P4D2
DY	66	162.500	3.005	3.321	6.478	[XE]S2F10	S5P4D2
HO	67	164.930	2.986	3.300	6.442	[XE]S2F11	S5P4D2
ER	68	167.260	2.967	3.279	6.408	[XE]S2F12	S5P4D2
TM	69	168.934	2.948	3.259	6.376	[XE]S2F13	S5P4D2

SY	Z	MASS[U]	R(COV)	R(ASA)	R(VDW)	CONFIGURATION	#(NODES)
YB	70	173.040	3.288	3.634	6.340	[XE]S2F14	S5P4D2
LU	71	174.967	2.948	3.259	6.879	[XE]S2D1F14	S5P4D2
HF	72	178.490	2.721	3.008	5.936	[XE]S2D2F14	S5P4D2
TA	73	180.948	2.532	2.799	5.990	[XE]S2D3F14	S5P4D2
W	74	183.840	2.457	2.715	5.800	[XE]S2D4F14	S5P4D2
RE	75	186.207	2.419	2.674	5.582	[XE]S2D5F14	S5P4D2
OS	76	190.230	2.381	2.632	5.896	[XE]S2D6F14	S5P4D2
IR	77	192.220	2.400	2.653	5.367	[XE]S2D7F14	S5P4D2
PT	78	195.080	2.457	2.715	5.204	[XE]S1D9F14	S5P4D2
AU	79	196.967	2.532	2.799	6.223	[XE]S1D10F14	S5P4D2
HG	80	200.590	2.816	3.112	5.112	[XE]S2D10F14	S5P4D2
TL	81	204.383	2.797	3.091	8.215	[XE]S2P1D10F14	S5P4D2
PB	82	207.200	2.778	3.071	8.120	[XE]S2P2D10F14	S5P4D2
BI	83	208.980	2.759	3.050	8.258	[XE]S2P3D10F14	S5P4D2
PO	84	208.982	2.759	3.050	8.899	[XE]S2P4D10F14	S5P4D2
AT	85	209.987	2.740	3.029	8.976	[XE]S2P5D10F14	S5P4D2
RN	86	222.018	2.721	3.008	9.005	[XE]S2P6D10F14	S5P4D2
FR	87	223.020	4.724	5.222	9.260	[RN]S1	S6P5D3F1
RA	88	226.025	3.779	4.178	6.949	[RN]S2	S6P5D3F1
AC	89	227.028	3.118	3.446	6.572	[RN]S2D1	S6P5D3F1
TH	90	232.038	3.118	3.446	6.418	[RN]S2D2	S6P5D3F1
PA	91	231.036	3.118	3.446	6.470	[RN]S2D1F2	S6P5D3F1
U	92	238.029	2.683	2.966	6.416	[RN]S2D1F3	S6P5D3F1
NP	93	237.048	3.080	3.405	6.470	[RN]S2D1F4	S6P5D3F1
PU	94	244.064	3.061	3.384	6.470	[RN]S2F6	S6P5D3F1
AM	95	243.061	3.496	3.864	6.389	[RN]S2F7	S6P5D3F1
CM	96	247.070	3.042	3.363	6.285	[RN]S2D1F7	S6P5D3F1
BK	97	247.070	3.005	3.321	6.310	[RN]S2F9	S6P5D3F1
CF	98	251.080	3.005	3.321	6.261	[RN]S2F10	S6P5D3F1
ES	99	252.083	2.986	3.300	6.234	[RN]S2F11	S6P5D3F1
FM	100	257.095	2.967	3.279	6.210	[RN]S2F12	S6P5D3F1
MD	101	258.099	2.948	3.259	6.187	[RN]S2F13	S6P5D3F1
NO	102	259.101	3.288	3.634	6.138	[RN]S2F14	S6P5D3F1
LR	103	260.105	2.948	3.259	6.115	[RN]S2D1F14	S6P5D3F1
RF	104	261.109	2.721	3.008	6.614	[RN]S2D2F14	S6P5D3F1
HA	105	262.114	2.532	2.799	6.614	[RN]S2D3F14	S6P5D3F1
CP	106	1.000	0.000	0.000	0.000	[0]	

20 The Universal Force Field (UFF)

The "Universal Force Field"(UFF)[36] is used in the QM-MM coupling and in the pre-optimization tool paw_preopt. It has been published in [36]. Parts of that paper most important for calculations are given here.

The original UFF has 126 atom types, however in the CP-PAW implementation there are currently 141 and occasionally some more may be added. In case of uncertainty have a look into the code: subroutine `UFFTABLE_INI` in the file `paw_classical.f`. A five-character mnemonic label is used to describe the atom types. The first two characters correspond to the chemical symbol; an underscore appears in the second column if the symbol has one letter. The third column describes the hybridization or geometry: 1=linear, 2=trigonal, R=resonant, 3=tetrahedral, 4=square planar, 5=trigonal bipyramidal, 6=octahedral. Thus N_3 is tetrahedral nitrogen, while Rh6 is octahedral rhodium. The fourth and fifth columns are used as indicators for alternate parameters such as formal oxidation state: Rh6+3 indicates an octahedral rhodium formally in the +3 oxidation state. H___B indicates a bridging hydrogen as in B_2H_6. Some parameters of the implementation of UFF in the CP-PAW code are given in the following table. `bond` is the bond radius in Å and `angle` the bond angle in degrees.

FFTYPE	BOND	ANGLE	FFTYPE	BOND	ANGLE
H_	0.354	180.000	AG1+1	1.386	180.000
H___B	0.460	83.500	CD3+2	1.403	109.471
HE4+4	0.849	90.000	IN3+3	1.459	109.471
LI	1.336	180.000	SN3	1.398	109.471
BE3+2	1.074	109.471	SB3+3	1.407	91.600
B_3	0.838	109.471	TE3+2	1.386	90.250
B_2	0.828	120.000	I_	1.382	180.000
C_3	0.757	109.471	XE4+4	1.267	90.000
C_R	0.729	120.000	CS	2.570	180.000
C_2	0.732	120.000	BA6+2	2.277	90.000
C_1	0.706	180.000	LA3+3	1.943	109.471
N_3	0.700	106.700	CE6+3	1.841	90.000
N_R	0.699	120.000	PR6+3	1.823	90.000
N_2	0.685	111.300	ND6+3	1.816	90.000
N_1	0.656	180.000	PM6+3	1.801	90.000
O_3	0.658	104.510	SM6+3	1.780	90.000
O_3_Z	0.528	145.500	EU6+3	1.771	90.000
O_R	0.680	110.300	GD6+3	1.735	90.000
O_2	0.634	120.000	TB6+3	1.732	90.000
O_1	0.639	180.000	DY6+3	1.710	90.000
F_	0.668	180.000	HO6+3	1.696	90.000
NE4+4	0.920	90.000	ER6+3	1.673	90.000
NA	1.539	180.000	TM6+3	1.660	90.000
MG3+2	1.421	109.471	YB6+3	1.637	90.000
AL3	1.244	109.471	LU6+3	1.671	90.000
SI3	1.117	109.471	HF3+4	1.611	109.471
P_3+3	1.101	93.800	TA3+5	1.511	109.471

```
P_3+5  1.056  109.471   W_6+6  1.392   90.000
P_3+Q  1.056  109.471   W_3+4  1.526  109.471
S_3+2  1.064   92.100   W_3+6  1.380  109.471
S_3+4  1.049  103.200   RE6+5  1.372   90.000
S_3+6  1.027  109.471   RE3+7  1.314  109.471
S_R    1.077   92.200   OS6+6  1.372   90.000
S_2    0.854  120.000   IR6+3  1.371   90.000
CL     1.044  180.000   PT4+2  1.364   90.000
AR4+4  1.032   90.000   AU4+3  1.262   90.000
K_     1.953  180.000   HG1+2  1.340  180.000
CA6+2  1.761   90.000   TL3+3  1.518  120.000
SC3+3  1.513  109.471   PB3    1.459  109.471
TI3+4  1.412  109.471   BI3+3  1.512   90.000
TI6+4  1.412   90.000   PO3+2  1.500   90.000
V_3+5  1.402  109.471   AT     1.545  180.000
CR6+3  1.345   90.000   RN4+4  1.420   90.000
MN6+2  1.382   90.000   FR     2.880  180.000
FE3+2  1.412  109.470   RA6+2  2.512   90.000
FE6+2  1.335   90.000   AC6+3  1.983   90.000
CO6+3  1.241   90.000   TH6+4  1.721   90.000
NI4+2  1.164   90.000   PA6+4  1.711   90.000
CU3+1  1.302  109.471   U_6+4  1.684   90.000
ZN3+2  1.193  109.471   NP6+4  1.666   90.000
GA3+3  1.260  109.471   PU6+4  1.657   90.000
GE3    1.197  109.471   AM6+4  1.660   90.000
AS3+3  1.211   92.100   CM6+3  1.801   90.000
SE3+2  1.190   90.600   BK6+3  1.761   90.000
BR     1.192  180.000   CF6+3  1.750   90.000
KR4+4  1.147   90.000   ES6+3  1.724   90.000
RB     2.260  180.000   FM6+3  1.712   90.000
SR6+2  2.052   90.000   MD6+3  1.689   90.000
Y_3+3  1.698  109.471   NO6+3  1.679   90.000
ZR3+4  1.564  109.471   LW6+3  1.698   90.000
NB3+5  1.473  109.471   CPR    0.551   90.000
MO6+6  1.467   90.000   CPR_B  0.340   90.000
MO3+6  1.484  109.471   CIR    0.616   90.000
TC6+5  1.322   90.000   PIR    0.616   90.000
RU6+2  1.478   90.000
RH6+3  1.332   90.000
PD4+2  1.338   90.000
```

21 Methods

21.1 Friction dynamics and annealing schedules

The time evolution of most quantities is done as friction dynamics. This friction itself may be time dependent for example in optimization procedures or via a thermostat. Here we shall describe the main features, and provide guidelines for the selction of the free parameters.

The general equation of motion is

$$m\ddot{x} = F(x) - m\alpha\dot{x}$$

$m\alpha$ is the friction coefficient. The mass has been integrated to be consistent with the formulation of the thermostats.

In practice this equation is discretized.

$$m\frac{x(t+\Delta) - 2x(t) + x(t-\Delta)}{\Delta^2} = F(x) - m\alpha\frac{x(t+\Delta) - x(t-\Delta)}{2\Delta}$$

which we resolve for $x(t+\Delta)$ as

$$x(t+\Delta) = \frac{2}{1+\frac{\alpha\Delta}{2}}x(t) - \frac{1-\frac{\alpha\Delta}{2}}{1+\frac{\alpha\Delta}{2}}x(t-\Delta) + \frac{1}{m}F(x)\frac{\Delta^2}{1+\frac{\alpha\Delta}{2}}$$

We define a new friction coefficient $a = \frac{\alpha\Delta}{2}$, which is the parameter specified as input to the program, so that

$$x(t+\Delta) = \frac{2}{1+a}x(t) - \frac{1-a}{1+a}x(t-\Delta) + \frac{1}{m}F(x)\frac{\Delta^2}{1+a}$$

There are two important special cases:

- Undamped dynamics results from $a = 0$. Here the friction is zero, and we obtain the well known Verlet algorithm

  $$x(t+\Delta) = 2x(t) - x(t-\Delta) + \frac{1}{m}F(x)\Delta^2$$

  for a second order differential equation. Over long times, the dynamics is energy conserving independent of the step size as result of the time inversion symmetry of the verlet algorithm.

- steepest descent is obtained with the choice $a = 1$.

$$x(t + \Delta) = x(t) - \frac{1}{m} F(x) \frac{\Delta^2}{2}$$

  Steepest descent corresponds to a dynamics with infinite friction. The motion does not come to rest even in this limit because the the time step is scaled inversely proportional to the friction. The product $\Delta^2/(2m)$ plays the role of the mixing parameter.

During optimization, that is with the options !AUTO switched on, the system can choose between two friction values. If the total energy is lowered the lower fiction value is chosen, and if the energy goes up the higher. Both are scaled by a factor, which allows to start with a high friction in order to take out energy from high frequency, that is high energy, modes. The reduced friction allows then to work on the low frequency modes.

If we know the frequencies $\omega = \frac{2\pi}{T}$, where T is the period of one oscillation we can provide some guidelines for the friction parameters.

- The dynamics becomes unstable for a time step $\Delta \geq 2/\omega_0 \approx T/3$. For timesteps $\Delta \leq T/10$ the error in the frequency is less than 1%. The frequency ω of the discretized undamped dynamics increases with the time step.

- The most rapid convergence rate is obtained for a friction lying just at the boundary between damped and overdamped dynamics, namely $a = 2\pi \frac{\Delta}{T}$. For frequencies with a shorter period than $\frac{2\pi\Delta}{a}$ the the variables converge exponentially with a rate $2\Delta/a$, which makes a large friction a appear favorable. However at the same time, oscilations with a period $T > 2\pi\Delta/a$ do not converge due to overdamping. (In order to converge bond stretch frequencies, which typically lie above 500 cm^{-1}, the friction parameter should not be higher than $a = 0.025$. This also indicates the inefficiency of the steepest descent approach which uses $a = 1$.)

21.2 Optimum friction scheme

The optimum friction is estimated from the forces and velocities of two subsequent time steps. These data are used to estimate the effective curvature and the effective mass along the line of propagation.

The optimum friction factor is estimated according to

$$a_{opt} = \Delta \sqrt{\frac{\dot{R}\dot{F}}{\dot{R}\mathbf{m}\dot{R}}}$$

Numerator and Denominator can oszillate rapidly. Therefore we use a running average, which is calculated accoring to

$$\langle a_{opt} \rangle = \beta a_{opt} + (1-\beta)\langle a_{opt}$$

The running average approaches the actual value exponentially. The deviation approaches $\frac{1}{2}$ of the original value after n time steps of

$$\beta = 1 - 2^{-\frac{1}{n}}$$

21.3 Wave function dynamics

Here we start with a total energy

$$E_{tot} = \sum_n f_n \langle \dot{\tilde{\Psi}}_n | m_\Psi | \dot{\tilde{\Psi}}_n \rangle + E_{DFT} - \sum_{n,m} \langle \tilde{\Psi}_n | \tilde{O} | \tilde{\Psi}_m \rangle \Lambda_{n,m} \tag{16}$$

The $\tilde{\Psi}$ are the pseudowave functions as defined by the PAW method, $\Lambda_{n,m}$ are the Lagrange parameters corresponding to the constraint of orthonormal wave functions and

$$\begin{aligned} E_{DFT} &= \sum_n \langle \Psi_n | -\frac{1}{2}\nabla^2 | \Psi_n \rangle \\ &+ \frac{1}{2} \int dr \int dr' \frac{(n(r)+n^Z(r))(n(r')+n^Z(r'))}{|r-r'|} \\ &+ \int dr n(r) \epsilon_{xc}(n_\sigma(r), \nabla n_\sigma(r)) \end{aligned}$$

Here we denote with $n^Z(r) = -\sum_R Z_R \delta(r-R)$ the density of the point charges of the nuclei.

$$m_\Psi |\ddot{\tilde{\Psi}}_n\rangle = -\tilde{H}|\tilde{\Psi}_n\rangle + \sum_m \tilde{O}\Lambda_{m,n} - m_\Psi |\dot{\tilde{\Psi}}_n\rangle f_\Psi \tag{17}$$

21.4 Nuclear dynamics

In order to move also the nuclei we add the kinetic energy of the atoms

$$\Delta E_{tot} = \sum_R \frac{1}{2} M_R \dot{R}^2 - \frac{1}{2} C_R \dot{R}^2 \tag{18}$$

Here the M_R are the true nuclear masses and C_R are the effective masses of the wave functions. The second term aims at subtracting the fictitious kinetic energy of the wave functions in a parameterized form.

$$(M_i - C_i)\ddot{R}_i \;=\; F_i - (M_i - C_i)\dot{R}_i f_R$$

21.5 Thermostats

Thermostats can be linked to both atoms and wave functions by adding

$$\Delta E_{tot} = \frac{1}{2} Q_\Psi \dot{x}_\Psi^2 + \frac{1}{2} Q_R \dot{x}_R^2 + g k_B T x_R \tag{19}$$

The equations of motion do not directly follow from the total energy as there is no Lagrangian formulation of the two thermostat method.

$$\begin{aligned}
m_\Psi |\ddot{\tilde{\Psi}}_n\rangle &= -\tilde{H}|\tilde{\Psi}_n\rangle + \sum_m \tilde{O}|\tilde{\Psi}_n \Lambda_{m,n} - m_\Psi |\dot{\tilde{\Psi}}_n\rangle \dot{x}_\Psi \\
(M_i - C_i)\ddot{R}_i &= F_i - (M_i - C_i)\dot{R}_i \dot{x}_R + C_i \dot{R}_i \dot{x}_\Psi \\
Q_\Psi \ddot{x}_\Psi &= 2\sum_n \langle \dot{\tilde{\Psi}}_n | m_\Psi | \dot{\tilde{\Psi}}_n \rangle - \sum_i C_i \dot{R}^2 \\
Q_R \ddot{x}_\Psi &= \sum_i (M_i - C_i)\dot{R}^2 - g k_B T
\end{aligned}$$

As an option one can also use only the atom thermostat x_R and replace the wave function thermostat friction $\dot{x}_\Psi$ by a constant friction f_Ψ, in the equation for atom and wave function thermostat.

21.6 Occupations

$$\begin{aligned}
\Delta E_{tot} &= \sum_{n,\sigma} m_X \dot{X}_{n,\sigma}^2 + k_B T \sum_{n,\sigma} \Big[f_{n,\sigma} \ln[f_{n,\sigma}] + (1 - f_{n,\sigma}) \ln[1 - f_{n,\sigma}] \Big] \\
&+ \mu \Big[\sum_{n,\sigma} f_{n,\sigma} - N \Big] + \mu \Big[\sum_{n,\sigma} f_{n,\sigma} (\delta_{\sigma,\uparrow} - \delta_{\sigma,\downarrow}) - S \Big] B
\end{aligned}$$

Here B is the magnetic Field acting on the electron spins, μ is the electron chemical potential, S is the total spin and N is the total number of electrons.

21.7 Sawtooth behavior of the total energy.

If we change the volume of the unit cell, the number of basis functions changes abruptly always when a new star of G-vectors enters or leaves the sphere defined by the plane wave cutoff. This abrupt change of the basis set results in abrupt changes of the total energy. Thus we obtain a typical saw-tooth behavior of the energy as function of volume as shown in Fig. 21.7.

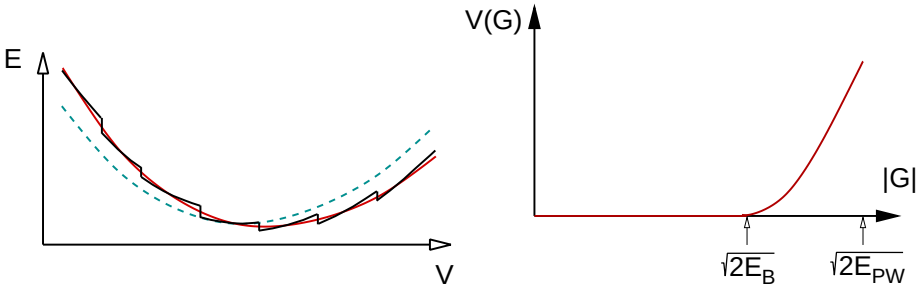


Figure 2: Left: Sawtooth behavior of the total energy as function of volume with a fixed plane wave cutoff (black) with a fixed number of plane waves (green dashed) and the correct result with a fully converged calculation (red). By adding a G-dependent potential the plane wave convergence can be artyificially accelerated.

The steps in the total energy vanish if the calculation is fully converged in the basis set size, because additional basis functions do not lower the total energy. If the function is not converged, the best result is obtained by keeping the same plane wave cutoff. However, if a the energy obtained at a few points is then fitted to, for example, a parabola or the Murnaghan equation of state[37], one has to be careful with the steps. A step can considerably mess up the fit. In these cases it is better to use a larger step size in the lattice constant.

This sawtooth behavior can be suppressed by including an additional G-dependent bucket potential that ensures that the wave function coefficients at the plane wave cutoff vanish strictly.

21.8 Constraints acting on atoms

21.9 Electrostatic decoupling and coupling of point charges

22 Benchmarks

Among the best calculations to compare energies are the calculations using Numol by Becke[38, 6, 7, 39]. Zero-point vibration energies and experimental data have been tabulated by Pople[40, 41].

23 Troubleshooting

23.1 Errors in the input data

- A common mistake is to enter data with the wrong data type. In this case you will receive a message as follows, which should help to locate the problem.

  ```
  TYPE INCONSISTENT
  VARIABLE ID HAS THE VALUE DT
  VARIABLE NTH HAS THE VALUE          1
  VARIABLE TYPE EXPECTED HAS THE VALUE R(8)
  VARIABLE ACTUAL TYPE HAS THE VALUE I(4)
  STOP IN LINKEDLIST_GETGENERIC
  ```

  The first line says that there is a type mismatch. The last line says that the problem happened in method of the linkedlist object, which is also responsible for linked list/tree data structures such as block-structured input files. In particular the program attempts to collect a data item from this tree structure. The second line gives you the name of the variable, in this example "DT", which is the key word for the time step. It cannot tell in which file or which block the data is located. The third line tells you that it is the first data with this name in the block. The next two lines say that the program expects a data with type "real(8)", but finds one of type "integer(4)". It appears that the program has mistaken your data for DT as an integer, because the user forgot to type the decimal point.

- The program does not do what you want it to. You may have mistyped keywords, or mixed bup the tree structure. This renders a lot of data invisible, and the program uses the default values. Try to consult the protocol of the program. It should report all options selected. If options are not listed, the option is not selected, or the corresponding value is zero. (Of course, it can

also be that it is not reported. In this case see whether you get a statement in the protocol if you select the option. If not, please report this to the author.)

- If you did not provide, or mistype, a keyword that does not have a default, you may receive a message like this:

```
KEYWORD T= IS MANDATORY
STOP IN STRCIN; !STRUCTURE!LATTICE
```

23.1.1 Changes of Input data

Here a list of syntactic changes for the input data. Although I am trying to keep this list complete, please consult the description of the input data to make sure.

1. !CONTROL!GENERIC:STOREPSIR
   → !CONTROL!PSIDYN:STOREPSIR
2. !CONTROL!PSIDYN:AUTO
   → !CONTROL!PSIDYN!AUTO
3. !CONTROL!RDYN:AUTO
   → !CONTROL!RDYN!AUTO
4. !CONTROL!FILES!FILE:IDENT
   → !CONTROL!FILES!FILE:ID
5. !STRUCTURE!GENERIC:NBAND
   → !STRUCTURE!OCCUPATIONS:NBAND
6. !STRUCTURE!GENERIC:NSPIN
   → !STRUCTURE!OCCUPATIONS:NSPIN
7. !STRUCTURE!GENERIC:CHARGE[-QEL]
   → !STRUCTURE!OCCUPATIONS:CHARGE[E]
8. !STRUCTURE!GENERIC:SPIN[-QEL]
   → !STRUCTURE!OCCUPATIONS:SPIN[HBAR]
9. !STRUCTURE!OCCUP
   → !STRUCTURE!OCCUPATIONS!STATE

10. !CNTL!PSIDYN:SAVEORTHO
    → !CNTL!PSIDYN:SAFEORTHO

11. !CNTL!ANALYSE!BOX disabled

12. !CONTROL!RNOSE
    → !CONTROL!RDYN!THERMOSTAT

13. !CONTROL!PSINOSE
    → !CONTROL!PSIDYN!THERMOSTAT

23.2 Runtime errors

This is a list of nontrivial user errors from which the program is not sufficiently protected and loose ends of the code where particular caution is needed. It is my goal to adjust the program to keep this list as short as possible.

1. Sometimes the program stops with an error message saying "LOOP FOR ORTHOGONALIZATION NOT CONVERGED". This is an indication of extremely large changes of the wave functions, which can no longer be orthogonalized. There are several possible problems that can lead to this, and several possible remedies:

    - The atomic positions have been chosen with unreasonably short distances, or with atoms lying on top of each other.
    - The number of valence electrons in the .strc file are not consistent with the setup files.
    - Some setups produce this behavior in the first few time steps. In this case reduce the plane wave cutoff to a small value such as 5, 10 or 20 Ry. If this alone does not help, also reduce the time step, but leave the masses constant. It is suffient to do this for the first few (about five or ten) time steps. If you started a new job with gradient corrected DFT functionals, you should try "!CONTROL!FOURIER:CDUAL=4" or reach convergence furst for a truly local functional before switching gradient corrections on. If the kinetic energy for the wave functions decreases to a reasonable value of a few Hartree, one can restart with the original values.
    - If all this does not help, you are in trouble!

2. The number of valence electrons have not been chosen consistent with the atomic setup. The number of valence electrons for each atom is not printed in the protocol.

3. If the program produces a coredump, increase first the stack size using the "ulimit" command. You can also control the size of the stack array using the compiler option -bmaxstack.

4. The code has a limited size of the send-receive buffer. The program is not guarded against a buffer overflow, even though the MPI Library should give a useful message (affects only the parallel version).

5. Errors occur if the operating system, Fortran runtime environment or SP2 Software are not on the same level, used during compilation.

6. If you find that the program crashes without providing an error message, and if you have the source code, you can set "TOFF=.false" in the source file paw_trace.f, and recompile. The program will then continuously write information on subroutines entered and left, which often allows one to locate the problem. Note that, in order to avoid producing files too large to be useful, not all routines result in a trace message.

7. The routine "MADELUNG" chooses itself the range of real and reciprocal space summations. It has not been sufficiently tested whether these choices are accurate. It may affect the result of the object "Isolate" that separates isolated molecules from its periodic images.

8. The Becke88 gradient correction[19]for exchange produces unreasonable results if the total density or the spin density is vanishes. In this case small numerical values for the gradient pick up a singularity in the functional form. This may happen for a fully spin polarized system. The cure is to pick another density functional, which does not contain the Becke88 functional.

9. In the compiler xlf5.1.1 the code creates a segmentation fault in a parallel-synchonization routine. In that case an empirical solution is to reduce the optimization from O3 to O. There is no problem of this type in the scalar version.

23.3 Porting

- Your system may not have the Korn shell available. In that case change the first line of any shell scripts, namely `#!/bin/ksh`, to `#!/bin/bash`. Most shell scripts are in the main paw directory. There may be some commands that do not work with bash yet.
- Adaptions of the preprocessor `F90PP/f90pp`. (This is a self made preprocessor.
- use the option **-arch** when compiling the code with `paw_compile` to use settings for a compiler different from the IBM xlf compiler. To see if your compiler is included and what value to set use `paw_compile ?` to obtain the command line syntax.

24 Installation

This section is under construction!

- unpack the tarball of the CP-PAW distribution. We will refer to it in the following as "PAW"
- create a configure script in the PAW directory

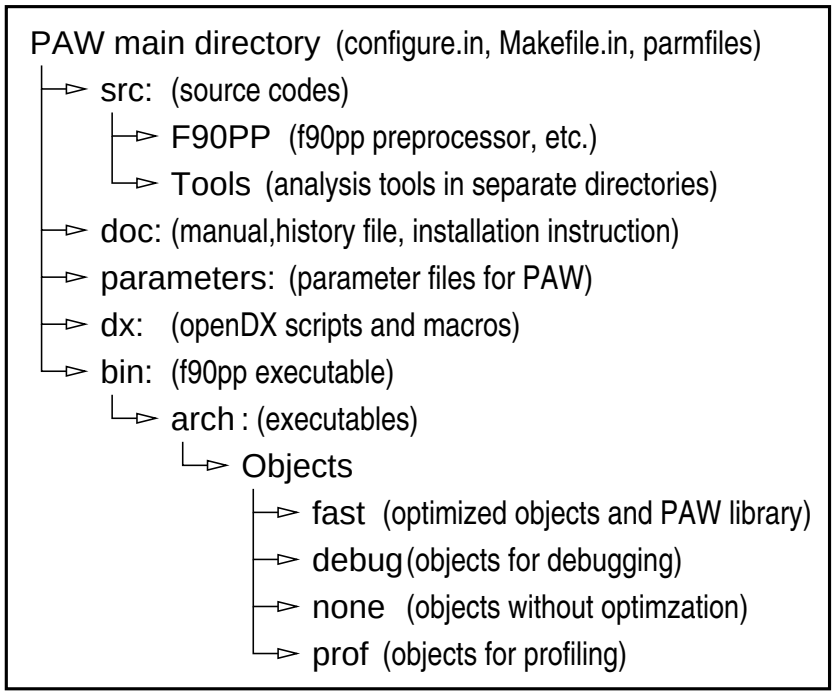
```
autoconf configure.in >configure
```

- create a parameter file *parmfile*. There are generic parameter files named for example `parms.g95`, `parms.ifc`, `parms.xlf`, which are compiler specific. Choose the appropriate one and create a copy by appending the name of your computer or environment. For example `parms.g95_mycomputer`. Specify ARCH as for example as `''g95_mycomputer''` and define FFTWDIR,MPICHDIR,LAPACKDIR as the locations of the FFTW library for Fourier transforms, the MPI library, and the library with algebraic subroutines (lapack and blas).
- Create Makefiles by running

```
configure -with-parmfile=parmfile
```

- include the directory with the executables into your search path.

The directory structure will have the following form



25 Suggestions

- print the creation date of the executable. This could be done by small subroutine which simply prints the time. This subroutine is on a separate file paw_date.f, which "depends" on the executable. A small shell script modifies paw_date.f first by replacing a characteristic string by the current date and then compiles is. It is always compiled before the program is linked.

- replace the covalent radii by the bond radii of UFF. give the reference in the manual.

- make criterion for drawing bonds in the paw_tra and the paw_wave file user defineable. Instead of fixing it to 1.2 times the covalent radius, read the factor in.

- check plotting eigenstates, versus wave functions

- make selection for PDOS scaling in the paw_dos tool
- check if paw_dos tool treats spin-polarized calculations properly

References

[1] P.E. Blöchl and A. Togni. First-principles investigation of enantioselective catalysis: Asymmetric allylic amination with pd-complexes bearing p,n-ligands. *Organometallics*, 15:4125, 1996.

[2] Open data explorer (opendx). http://www.opendx.org/. a general-purpose 3D visualization software derived from the IBM Dataexplorer.

[3] P.E. Blöchl. Projector augmented-wave method. *Phys. Rev. B*, 50:17953, 1994. original paper on PAW.

[4] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:864, 1964.

[5] W. Kohn and L.J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:1133, 1965.

[6] A.D. Becke. Density functional thermochemistry. ii. the effect of the perdew-wang generalized-gradient correlation correction. *J. Chem. Phys*, 97:9173, 1992.

[7] R.M. Dickson and A.D. Becke. Basis-set ffree local density functional calculations of geometries of polyatomic molecules. *J. Chem. Phys.*, 99:3898, 1993.

[8] R. Car and M. Parrinello. Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett*, 55:2471, 1985. Original paper on ab-initio molecular dynamics.

[9] R. Car and M. Parrinello. *Simple Molecular Systems at Very High Density*, chapter A Unified Approach for Molecular Dynamics and Density Functional Theory, page 455. Plenum, NY, 1989.

[10] G. Pastore, E. Smargiassi, and F. Buda. *Phys. Rev. A*, 44:6334, 1991.

[11] P.E. Blöchl. Electrostatic decoupling of periodic images of plane wave expanded densities and derived atomic point charges. *J. Chem. Phys.*, 103:7422, 1995.

[12] Message passing interface. hhttp://www.osc.edu/Lam/lam.html#MPI. Software for interprocessor communication on parallel computers.

[13] P.J. Mohr and B.N. Taylor. Codata recommendet values of the fundamental physical constants: 1989. *Rev. Mod. Phys.*, 72:351, 2000.

[14] P.J. Turner. Xmgrace, an xy-plotting tool. http://plasma-gate.weizmann.ac.il/Xmgr/.

[15] B.Lüken. Crymolcad. Planned to be available soon at http://orion.pt.tu-clausthal.de/paw/. Software package for the visualization, analysis and manipulation of Crystal and Molecular Structures as well data-fields.

[16] J.P. Perdew and A. Zunger. Self interaction correction to density-functional approximations for many-electron systems. *Phys. Rev. B*, 23:5048, 1981.

[17] D.M. Ceperley and B.J. Alder. Ground state of the electron gas by a stochastic method. *Phys. Rev. Lett.*, 45:566, 1980.

[18] J. P. Perdew and Y. Wang. Accurate and simple analytic representation of the electron gas correlation energy. *Phys. Rev. B*, 45:13244, 1992.

[19] A.D. Becke. Density-functional exchange energy with correct asymptotic behavior. *Phys. Rev. A*, 38:3098, 1988.

[20] J. P. Perdew. Density functional approximation for the correlation energy of the inhomogeneous electron gas. *Phys. Rev. B*, 33:8822, 1986.

[21] J.P. Perdew, K. Burke, and M. Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett*, 77:3865, 1996.

[22] J.P. Perdew, J.A. Chevary, S.H. Vosko, K.A. Jackson, M.P. Pederson, D.J. Singh, and C. Fiolhais. Atoms, molecules solids and surfaces: applications of the generalizd gradient approximation for exchange and correlation. *Phys. Rev. B*, 46:6671, 1992.

[23] B. Hammer, L.B. Hansen, and J.K. Norskov. Improved adsorption energetics within density-functional theory using revised perdew-burke-ernzerhof functionals. *Phys. Rev. B*, 59:7413, 1999.

[24] S. Grimme. Accurate description of van der waals complexes by density functional theory including empirical corrections. *J. Comp. Chem*, 25:1463, 2004.

[25] F. Tassone, F. Mauri, and R. Car. Acceleration schemes for ab-initio molecular dynamics simulations and electronic structure calculations. *Phys. Rev. B*, 50:10561, 1994.

[26] N.D. Mermin. Thermal properties of the inhomogeneous electron gas. *Physical Review*, 1965.

[27] S. Nose. A molecular dynamics method for the simulation in the canonical ensemble. *Mol. Phys.*, 52:255, 1984.

[28] S. Nose. A unified formulation of the constant temperature molecular dynamics methods. *J. Chem. Phys.*, 81:511, 1984.

[29] W.G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31:1695, 1985.

[30] P.E. Blöchl. Second-generation wave-function thermostat for ab-initio molecular dynamics. *Phys. Rev. B*, 65:104303, 2002.

[31] P.E. Blöchl and M. Parrinello. Adiabaticity in first-principles molecular dynamics. *Phys. Rev. B*, 45:9413, 1992.

[32] P.E. Blöchl, O. Jepsen, and O.K. Andersen. Improved tetrahedron method for brillouin-zone integrations. *Phys. Rev. B*, 49:16223, 1994.

[33] V.I. Anisimov, J. Zaanen, and O.K. Andersen. Band theory and mott insulators: Hubbard u instead of stoner i. *Phys. Rev. B*, 44:943, 1991.

[34] J.-P. Ryckaert, G. Cicotti, and H. J. C. Berendsen. Numerical integration of cartesian equations of motion of a system with constraints: molecular dynamics of alkenes. *J. Comp. Phys.*, 23:327, 1977.

[35] First-Principles Calculations of Diffusion Coefficients: Hydrogen in Silica. P.e. blöchl and c.g. van de walle and s.t. pantelides. *Phys. Rev. Lett*, 64:1401, 1990.

[36] A. K. Rappe, C.J. Casewit, K.S. Colwell, III W.A. Goddard, and W.M. Skiff. Uff, a fully periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.*, 114:10024, 1992.

[37] F.D. Murnaghan. The compressibility of media under extreme pressures. *Proc. Natl. Acad. Sci. U.S.A.*, 30:244, 1944.

[38] A.D. Becke. Density functional thermochemistry. i. the effect of exchange-only gradient correction. *J. Chem. Phys*, 96:2155, 1992.

[39] A.D. Becke. Density functional thermochemistry. v. systematic optimization of exchange-correlation functionals. *J. Chem. Phys*, 107:8554, 1997.

[40] J.A. Pople, M. Head-Gordon, D.J. Fox, K. Raghavacchari, and L.A. Curtiss. Gaussian-1 theory: A general procedure for prediction of molecular energies. *J. Chem. Phys.*, 90:5622, 1989.

[41] L.A. Curtiss, K. Raghavacharir, G.W. Trucks, and J.A. Pople. Gaussian-2 theory for molecular energies of first- and second-row compounds. *J. Chem. Phys.*, 94:7221, 1991.

Index