PARSEC 1.4 User's Guide

Prof. J. R. Chelikowsky's Research Group
http://real-space.org/

May 29, 2021

Contents

1 What it is

PARSEC is a computer code that solves the Kohn-Sham equations by expressing electron wave-functions directly in real space, without use of explicit basis set. The name stands for "Pseudopotential Algorithm for Real-Space Electronic Calculations". It handles one-dimensional and three-dimensional periodic boundary conditions and confined-system boundary conditions. In the latter, the charge density and wave-functions are required to vanish outside a spherical shell enclosing the electronic system. A finite-difference approach is used

for the calculation of spatial derivatives. Norm-conserving pseudopotentials are used to describe the interaction between valence electrons and ionic cores (core electrons+nuclei). Owing to the sparsity of the Hamiltonian matrix, the Kohn-Sham equations are solved by direct or iterative diagonalization, with the use of extremely efficient sparse-matrix eigensolvers. Besides standard DFT calculations, parsec is able to perform structural relaxation, simulated annealing, Langevin molecular dynamics, and polarizability calculations (confined-system boundary conditions only). It can also be used to provide input for TDLDA (time-dependent density functional theory, in adiabatic local-density approximation), GW approximation, and solution of the Bethe-Salpeter equation for optical excitations.

The development of parsec started in the early 1990's with the work of J.R. Chelikowsky, N. Troullier, and Y. Saad (Phys. Rev. Lett **72**, 1240 ; Phys. Rev. B **50**, 11355). Later, many people have contributed with additional features and performance improvement:

- A. Stathopolous : mathematical subroutines;
- H. Kim : core correction, multipole expansion;
- I. Vasiliev : Polarizability, asymptotically-corrected functionals;
- M. Jain and L. Kronik : GGA, generalized Broyden mixer, "cleaning" of the source code, efficiency improvements, and added functions;
- K. Wu : improvements in the iterative eigensolver *diagla* and the Hartree solver by conjugate-gradient;
- R. Burdick : major rewrite and improvement of multipole expansion;
- M. Alemany and M. Jain : periodic boundary conditions, parallel implementation;
- M. Alemany and E. Lorin : ARPACK solver;
- A. Makmal: dynamical memory allocation;
- M. Tiago : parallelization, symmetry operations, wire (1-D) boundary conditions, this user's guide;
- Y. Zhou : Chebyshev-Davidson eigensolver; thick-restart Lanczos eigensolver; Chebyshev subspace filtering method which serves to avoid diagonalizations;
- A. Natan : non-orthogonal boundary conditions; slab (2-D)
- A. Benjamini: slab (2-D)
- D. Nave : spin-orbit potential;
- H.-W. Kwak : LDA+U;
- S. Beckman : k-point sampling.

2 Getting the code

Public versions of the code can be downloaded from Prof. Chelikowsky's group page http://real-space.org/. There is also a development version, with implementations that are being tested for stability and performance. If you are interested in contributing to development, contact the developers.

PARSEC is distributed under the General Public License (GPL). In a nutshell, these are the conditions implied in using it:

- It should be used for non-commercial purposes only.
- The authors give no warranty regarding its reliability or usefulness to your particular application.
- The authors offer no technical support.

2.1 Compiling PARSEC

At the beginning of the `Makefile` file, edit the include line with the correct machine-dependent compilation file. Existing compilation files are inside subdirectory `config`. Some of the existing files are:

- `make.ibmsp3, make.ibmsp3_mpi` : sp.msi.umn.edu, with/without MPI support.
- `make.intel` : plain i686 machine running linux.

This code has been ported and tested on a variety of different platforms: IBM SP3 and SP4 (AIX compiler), SGI (intel and f90 compilers), Intel/Linux (g95, intel compilers). Once the `config/make.*` file has been edited, build the executable by running make. Upon completing a successful build, the file `parsec.ser` (no MPI support) or `parsec.mpi` (with MPI support) will be found in the src directory.

PARSEC uses several external libraries. For the basic serial build the only necessary external library is LAPACK/BLAS. The best option is to pick machine optimized versions of these libraries such as ESSL (IBM) or MKL (Intel). If these are unavailable you can download LAPACK from netlib. If you want to use periodic boundary conditions then FFTW is needed. Turn this feature on by passing either -DUSEFFTW3 or -DUSEFFTW2 to the preprocessor depending on if you are using the version 2 or 3 of the library. If you want to use the parallel implementation you need MPICH2. To make the parallel version pass -DMPI to the preprocessor. If you want to use the ARPACK diagonalization method, you need to build the ARPACK library. Pass the flag -DUSEARPACK to the preprocessor.

To port PARSEC to new machines you will need to modify the existing config/make.* file for the machine you want to use. In addition you may need to modify the machine specific files `myflush.f90p`, `timing.f90p`, `cfftw.f90p`. If you look in these you will notice that there have been a variety of ports

made and machine specific modifications have been specified by preprocessor calls (e.g. the preprocessor flag `-DIBM` specifies that the IBM "mclock" function be used for timing calls). In the case that the Portland Group compiler is used, the preprocessor flag `-DPGI` must be passed.

In addition to these preprocessor flags there are several other options that can be passed to the preprocessor. These are generally for development or other highly specialized purposes. Their operation can be learned by reading the source code.

2.2 Porting to new platforms

Besides creating a new `make.*` file for the machine you want to use, these are the files you may have to modify in order to port it to other platforms: myflush.F, timing.F, cfftw.F

PARSEC uses the following external libraries:

- ARPACK, diagonalization of sparse matrices. Optional. The interface with arpack is turned on with compilation flag `USEARPACK`.
- FFTW, versions 2.x or 3.x, used for 3-dimensional FFT of real functions. Optional. The interface with FFTW is turned on with compilation flag `USEFFTW2` or `USEFFTW3`. Notice that, without FFTW, PARSEC cannot handle periodic systems.
- LAPACK/BLAS, linear algebra. Alternatively, support for libraries ESSL (IBM) and MKL (Intel Math Kernel Library) is also available.
- MPI : message passing interface, for parallel machines.

3 Running PARSEC

3.1 Basic input

- `parsec.in` : input file with user-provided parameters.
- Pseudopotential files. For the moment, only norm-conserving pseudopotentials are implemented. Currently, PARSEC supports four different formats (see `Pseudopotential_Format` PseudopotentialFormat):
  - `Martins_old` : original format, included for back-compatibility with older versions of PARSEC. Pseudopotential files have name `*_POTR.DAT`. In this format, files do not have atomic parameters such as cut-off radii included.
  - `Martins_new` : format of files created with Prof. J.L. Martins' pseudopotential generator, version 5.69p. The distributors of Parsec have

a version of Prof. Martins' code with minor but important modifications, so that files in the proper format are created. Pseudopotential files have name `*_POTRE.DAT`. Files in this format are compatible with the "Siesta" DFT program, but the opposite is not true: pseudo-wavefunctions are appended to `*_POTRE.DAT` files, but the same does not necessarily happen to formatted pseudopotential files used in Siesta. WARNING: different from `Martins_old` format, a single radial grid is used for pseudopotentials, pseudowavefunctions and densities, and the grid density has a default value which is often too low. In some cases, this can be a problem, especially for the core density. There are at least two solutions for this problem: either modify the default value of grid density (for example, set `bb=160` in file input.f), or specify your own grid density in input file `atom.dat`, or the equivalent in your pseudopotential code. A good sanity check is to compare the results provided by parsec with reliable results obtained by using other DFT codes.

- `Martins_Wang` : format of files created with Martins' code as modified by Lin-Wang Wang. This format is also supported by the DFT codes PEtot and PARATEC. Pseudopotential files have name `*_POTRW.DAT`.
- `FHIPP` : format of files created by the FHIPP code. Pseudopotential files have name `*_FHIPP.DAT`. This format allows calculations within the virtual crystal approximation, but they do not currently allow the inclusion of spin-orbit corrections.

We do not offer any advice to the user about the choice of pseudopotential, although we do distribute the newest version of J.L. Martins pseudopotential generator with PARSEC. For generating pseudopotentials we suggest the user read the webpages:

- http://www.fhi-berlin.mpg.de/th/fhi98md/fhi98PP/,
- http://opium.sourceforge.net/,
- http://bohr.inesc-mn.pt/ jlm/pseudo.html and
- http://hpcrd.lbl.gov/ linwang/PEtot/PEtot.html.

3.2 The parsec.in file

The user-provided input uses the electronic structure data format (ESDF), implemented as a module written by Chris J. Pickard. Comments can be added on any line, and are preceded by any of the symbols "`# ; !`". Keywords and blockwords are case insensitive. Characters "`. : =`" and multiple spaces are ignored. Most parameters have default values. All parameters with true/false value have default value false. Parameter values can be written anywhere in the file, with the exception that information pertinent to the same chemical element

should be given in consecutive lines (sometimes, data from different elements can be mixed, but this should be avoided for the sake of clarity). Keywords are always followed by their corresponding value on the same line. Input values with intrinsic physical unit can be given in different units, with name following the value. Default unit is used if none is specified. For example, the line

```
grid_spacing 0.5 ang
```

sets a grid spacing of 0.5 angstroms. Blocks have the syntax:

```
begin Blockword
.
.
.
end Blockword
```

WARNINGS:

- PARSEC does not watch out for misprints in `parsec.in`. Lines with unrecognized keywords or blockwords are usually ignored.
- The EDSL library sometimes screws up when identifying certain features with certain atom types (more on this in chapter 6).

3.3 Basic output

- `parsec.out` : The unified output file. This file contains most of the relevant messages about the calculation as well as the results.
- `out.*` : The standard output from each processor. It contains some warnings and error messages. Information about the diagonalization as well as other processor specific information is written here. These files are useful for debugging.
- `parsec.dat` : The eigenvalues, density function, effective potential, wavefunctions and other output data, in binary (unformatted) form. This file can sometimes be very lengthy.

3.4 Interrupting calculation

In situations when the user wants to interrupt a calculation, he/she can do it by creating a file with name `stop_scf` (case sensitive). This file can be empty, but it must be created in the working directory, the same one which contains all the input/output files. The sequence of SCF iterations will be aborted as early as possible and output written to files `parsec.out` and `parsec.dat`. The last file can be used then as input to a subsequent calculation. This feature is useful if the user wants to stop the calculation "smoothly" and be able to restart it at a later time.

4 Input options

4.1 The real-space grid

The following parameters define the region of space where the Kohn-Sham equations will be solved, and the appropriate boundary conditions. In parallel environment, grid points are distributed among processors. Symmetry operations are used to reduce the sampled region to an irreducible wedge.

4.1.1 `Boundary_Conditions`

(*word*), default value = `cluster`

Defines the type of boundary conditions to be imposed. Available options are:

- `cluster`. Specifies completely confined system, with boundary conditions such that the electron wavefunctions are zero outside a spherical domain. The discretized grid is laid out following the Cartesian directions. If necessary (and if consistent with other input options), a constant vector is added to all atomic coordinates so that the geometric center of the systems coincides with the origin of the Cartesian system.

- `bulk`. Specifies periodic boundary conditions along all 3 Cartesian (or generalized) directions. If necessary, the input grid spacing may be increased so as to make it commensurable with the given size of periodic cell. Also, atomic coordinates are used as given, with no initial repositioning around the origin of the real-space grid. If atoms are found to be outside the cell, they are replaced by their periodic image inside the cell.

- `wire`. The system is assumed to be periodic along the Cartesian **x** direction and confined within a cylinder centered on the **x** axis. The periodicity length is defined in block `Cell_Shape` CellShape. The cylinder radius is defined by `Boundary_Sphere_Radius` BoundarySphereRadius. This type of boundary conditions can be used together with k-point sampling.

- `slab`. The system is assumed to be periodic along the Cartesian **x** and **y** directions. The cell structure is defined in the block `Cell_Shape` CellShape. The slab width is twice the value that is defined in `Boundary_Sphere_Radius` BoundarySphereRadius. This boundary conditions can be used together with k-point sampling.

4.1.2 `Cell_Shape`

(*block*), default value = `no default`

Specification of unit lattice vectors in Cartesian coordinates. Notice that, in contrast to earlier versions, this version requires the full vectors. With bulk boundary conditions (periodicity along the 3 Cartesian directions), this block must have three lines, for the coordinates of the 3 unit lattice vectors. With

wire boundary conditions (periodicity along the **x** direction only, this block must have one line with a single number: the periodicity length along **x**. With slab boundary conditions (periodicity along the **x** and **y**), this block must have 2 lines with 2d vectors that span the x-y plane, if 3d vevtors are given, the 3rd number is ignored. A global scale of unit lattice vectors can be done with input flag (`Lattice_Vector_Scale`LatticeVectorScale). Relevant only in periodic systems (bulk, slab, or wire).

4.1.3 `Lattice_Vector_Scale`

(*real*), default value = `1` , default unit = *bohr* (Bohr radii)

Constant which multiplies the length of all three unit lattice vectors. This is useful for volumetric expansions or contractions in periodic systems. Relevant only in periodic systems.

4.1.4 `Super_Cell`

(*block*), default value = `1 1 1`

Specification of super-cell with respect to unit lattice vectors of a periodic system. If declared to be (n, m, l), then the actual periodic cell will be a $n \times m \times l$ expansion of the cell specified in input. Atom coordinates are expanded accordingly. This is useful in super-cell calculations where the super-cell is obtained by replicating a minimal cell several times. Not used in non-periodic systems. In wire (1-D) systems, only the first number is used. In slab (2-D) systems, only the first 2 numbers are used.

4.1.5 `Super_Cell_Vac`

(*block*), default value = `0 0 0`

Specification of vacuum in a supercell calculation. If declared to be (n, m, l), the super-cell is expanded by adding n empty cell along the first periodic direction, m empty cells along the second periodic direction and l empty cells along the third periodic direction. If applicable, this expansion is done after the expansion defined by `Super_Cell`SuperCell. Not used in non-periodic systems. In wire (1-D) systems, only the first number is used. In slab (2-D) systems, only the first 2 numbers are used.

4.1.6 `Boundary_Sphere_Radius`

(*real*), default value = `no default` , default unit = *bohr* (Bohr radii)

Radius of enclosing spherical shell. Boundary conditions in non-periodic systems are such that all wave-functions vanish beyond a spherical shell centered in the origin and with given radius. The Hartree potential is also set to vanish on the shell (internally, this is done by solving Poisson's equation with a compensating charge outside the shell). Calculation stops if any atom is found outside the shell. The amount of extra space that should be given between the shell and any inner atom is strongly system-dependent and should be carefully

tested for convergence. With wire boundary conditions (system periodic along the **x** axis and confined along perpendicular directions), this option defines the radius of the confining cylinder. With slab boundary conditions this number is half the slab width in the **z** axis.

4.1.7 `Grid_Spacing`

(*real*), default value = `no default` , default unit = *bohr* (Bohr radii)

Distance between neighbor points in the real-space grid. This is a critical parameter in the calculation and it should be carefully tested for convergence. The grid is assumed to be regular. In periodic systems (3-D and 2-D) the grid can be non-orthogonal according to the lattice vectors definition.

4.1.8 `Expansion_Order`

(*integer*), default value = `12`

Order used in the finite-difference expansion of gradients and Laplacians. It corresponds to twice the number of neighbor points used on each side and for each direction. It must be an even number.

4.1.9 `Ignore_Symmetry`

(*boolean*), default value = `false`

Cancels all symmetry operations. if this flag is true, symmetry is not going to be used in the calculation.

4.1.10 `Double_Grid_Order`

(*integer*), default value = `1`

Order for the double-grid used in the calculation of pseudopotentials in real space. In order to account for short-range variations in pseudopotentials (which can break translational symmetry in periodic systems), a double grid is defined following the method of Ono and Hirose (*Phys. Rev. Lett.* **82**, 5016 (1999)). A double-grid order N implies that $N-1$ additional grid points will be added between two neighbor points in the original grid, and Kohn-Sham eigenvectors interpolated between them. The default value is equivalent to original setting (no double grid).

4.1.11 `Cluster_Domain_Shape`

(*word*), default value = `sphere`

This option is relevant for the 0-D case only. It defines the shape of the boundary conditions of the grid. Implemented options are:

- `sphere`: Spherical boundary conditions are used. The domain is defined by all the points that fulfill $(x^2+y^2+z^2) < R^2$ where R is defined by the `Boundary_Sphere_Radius` BoundarySphereRadius

- `ellipsoid`: Ellipsoid boundary conditions are used. The Ellipsoid paramaters are defined by the `Domain_Shape_Parameters` DomainShapeParameters block, this block MUST be defined.
- `cylinder`: Cylindrical boundary conditions are used. The cylinder parameters are defined by the `Domain_Shape_Parameters` DomainShapeParameters block, this block MUST be defined.
- `box`: Box boundary conditions defines a rectangular cuboid box. The box parameters are defined by the `Domain_Shape_Parameters` DomainShapeParameters block, this block MUST be defined.

4.1.12 `Domain_Shape_Parameters`

(*block*), default value = `no default`

This block defines the paramers for the differnet shapes of the grid. A single line with 3 numbers is supplied. The meaning of the numbers depend on the domain definition:

- ellipsoid: The numbers are simply the a,b,c of the ellipsoid.
- cylinder: The numbers are R, L, d. R is the cylinder radius, L is the cylinder length and d is the orientation: d=1 for x, d=2 for y and d=3 for Z.
- box: The numbers are simply a,b,c - the box dimensions in x,y,z respectively.

4.1.13 `Full_Hartree`

(*boolean*), default value = `false`

This flag is set for the case of an isolated shape that is extended in one dimension relative to the others (e.g. a cylinder where $L >> R$). In such shapes the usual multipole approximation is not converging well and so an exact calculation of the Hartree potential is performed for the boundary conditions of the Poisson equation. Note that such calculation is much slower than the usual multipole approximation.

4.1.14 `Boundary_Mask`

(*boolean*), default value = `false`

Option available in confined systems only. Allows for flexibility in the definition of the real-space grid. If enabled, the grid will be constructed according to function `My_Mask` built in the source. See source for more information. WARNING: function `My_Mask` must be tailored to the specific system.

4.2 Diagonalization parameters

Many DFT codes solve the Kohn-Sham equations by minimizing the total energy functional. Parsec uses a different approach: it performs iterative diagonalization of the Kohn-Sham equations. Expressed in real space, the Hamiltonian is a sparse matrix, and efficient algorithms can be used to obtain the low-energy eigenstates with minimal memory usage and parallel communication.

4.2.1 `Eigensolver`

(*word*), default value = `chebdav`
Implemented eigensolvers are:

- `arpack`: Implicitly restarted Lanczos eigensolver as implemented in ARPACK. Reference: http://www.caam.rice.edu/software/ARPACK/
- `diagla`: Block preconditioned Lanczos. Reference: A. Stathopoulos, Y. Saad, and K. Wu, SIAM J. on Scientific Computing **19** 229 (1998).
- `trlanc`: Locally modified thick-restart Lanczos package developed by K. Wu and H. Simon. Reference: http://crd.lbl.gov/ kewu/trlan.html http://crd.lbl.gov/ kewu/trlan.html
- `chebdav`: Chebyshev-Davidson eigensolver algorithm. References: Y. Zhou, Y. Saad, M.L. Tiago, and J.R. Chelikowsky, J. Comput. Phys. **219**, 172 (2006); Phys. Rev. E **74**, 066704 (2006).
- `chebff`: Chebyshev-filtered subspace iteration eigensolver. References: Y. Zhou, J. R. Chelikowsky, Y. Saad, J. Comput. Phys. **274**, 770–782 (2014).

The last two eigensolvers use significantly less memory, and `diagla` occasionally fails to converge in periodic systems.

4.2.2 `Diag_Tolerance`

(*real*), default value = `1.d-4`
Tolerance in the residual norm of eigen-pairs.

4.2.3 `Maximum_Matvec`

(*integer*), default value = `max(30000,N*neig)`
Maximum number of matrix-vector operations to be allowed on each diagonalization. This is useful if diagonalization fails to converge. The ideal value depends on the actual number of rows in the Hamiltonian (`N`) and the number of desired eigenvalues (`neig`). Calculations with hundreds of atoms or more may require this parameter to be increased from its default value.

4.2.4 `Subspace_Buffer_Size`

(*integer*), default value = `5`

Number of additional eigenvalues to be computed for each representation. Must be some positive value, so that the full set of eigenvalues can be reconstructed from the sum of representations without missing levels.

4.2.5 `Chebyshev_Filtering`

(*boolean*), default value = `false`

After the first few diagonalizations, this option replaces the standard diagonalization with a polynomial filtering of the approximate eigen-subspace. This technique is usually much more efficient than any of the eigensolvers above. It is enabled automatically if the eigensolver `chebdav` is used. Reference for the subspace filtering method is:

Y. Zhou, Y. Saad, M.L. Tiago, and J.R. Chelikowsky, *Self-consistent-field calculation using Chebyshev polynomial filtered subspace iteration*, J. Comput. Phys. **219**, 172 (2006)

4.2.6 `Chebdav_Degree`

(*integer*), default value = `20`

Degree of Chebyshev polynomial used in the Chebyshev-Davidson eigensolver. The degree is recommended to be within 15 to 30, usually a degree within 17 to 25 should work well.

4.2.7 `Chebyshev_Degree`

(*integer*), default value = `15`

Average degree of Chebyshev polynomial used in the subspace filtering method, this degree generally is smaller than the degree used in the Chebyshev-Davidson method. Note that the subspace filtering method serves to avoid doing diagonalizations, while the Chebyshev-Davidson method is used for diagonalization.

This polynomial degree is recommended to be within 7 to 20. If slow convergence is observed for some hard problems, one may use a higher degree, e.g., 21 to 25.

4.2.8 `Chebyshev_Degree_Delta`

(*integer*), default value = `1` (if `Chebyshev_Degree` < 10), `3` otherwise

During subspace filtering, the energy spectrum is divided in two windows of equal width: the low-energy window is filtered with polynomial of degree $p - \delta$ and the high-energy window is filtered with polynomial of degree $p + \delta$, where p is the average degree (`Chebyshev_Degree`) and δ is the degree variation (`Chebyshev_Degree_Delta`). Large values of the degree variation usually give shorter runtime, but they may make the filtering process unstable if the ratio δ/p is much larger than around 0.30.

4.3 Self-consistency and mixing parameters

4.3.1 `Max_Iter`

(*integer*), default value = `50`

Maximum number of iterations in the self-consistent loop.

4.3.2 `Convergence_Criterion`

(*real*), default value = `2.d-4` , default unit = *Ry* (rydbergs)

The density function is considered converged when the self-consistent residual error (SRE) falls below this value. The SRE is an integral of the square of the difference between the last two self-consistent potentials, weighted by electron density and taken squared root. This parameter should not be chosen less than the typical diagonalization accuracy.

4.3.3 `Convergence_Criterion_Approach`

(*real*), default value = `100 * Convergence_Criterion` , default unit = *Ry* (rydbergs)

If the SRE becomes less than this value and if the polynomial order during subspace filtering is more than 10, the polynomial order will be reduced by one unity after each SCF cycle until the order becomes equal to 10. Beyond that point, the order is kept fixed. This is a performance parameter, and it exploits the fact that subspace filtering with low order is as efficient as filtering with high order if the eigenvectors are close enough to the converged solution.

4.3.4 `Mixing_Method`

(*word*), default value = `Anderson`

Defines the scheme used in the mixing of old and new potentials. Implemented schemes are `Broyden`, `Anderson`, `msecant1`, `msecant2`, `msecant3`. The first two methods were proved to be equivalent by V. Eyert, 1996. In practice, there is no significant difference between them. Broyden mixing uses disk for input/output of internal data. The last three methods are multi-secant methods. They were developed and coded by Haw-ren Fang and Yousef Saad, at the University of Minnesota. `Msecant1` is a class of multi-secant methods with Type-I update to minimize the change of Jacobian. `Msecant2` is a class of multi-secant methods with Type-II update to minimize the change of inverse Jacobian. `Msecant3` is a hybrid method, that minimizes the change of Jacobian or inverse Jacobian depending on the secant errors. This last method was inspired by the work of J.M. Martínez.

4.3.5 `Mixing_Param`

(*real*), default value = `0.30`

Choice for the diagonal Jacobian used in the mixing of potentials. Typical range is between `0.1` and `1.0`. Small values reduce the amount of mixing and

result in slower but more controlled convergence. For improved performance, the user may try reducing this parameter in cases where the energy landscape is too corrugated (e.g.: magnetic systems, or systems with many degrees of freedom).

4.3.6 `Memory_Param`

(*integer*), default value = `4`

Number of previous SCF steps used in mixing. Ideal values range from 4 to 6. Larger values may actually worsen convergence because it includes steps far from the real solution.

4.3.7 `Restart_Mixing`

(*integer*), default value = `20`

Maximum number of iterations in the self-consistent loop before mixing is restarted. This should improve SCF convergence in cases when mixing becomes "stuck" and the SRE stops decreasing.

4.3.8 `Mixing_Group_Size`

(*integer*), default value = `1`

Sets the size of groups of secant equations. To set infinite size of groups, use `Mixing_Group_Size: 0`.

4.3.9 `Mixing_EN_Like`

(*integer*), default value = `0`

Indicates using the Broyden-like update of the approximate (inverse) Jacobian. For EN-like update, set `Mixing_EN_Like: 1`. Possible values are `0` and `1`.

4.3.10 `Mixing_Preferred_Type`

(*integer*), default value = `1`

Relevant only for hybrid multi-secant methods (`msecant3`). Possible values are `0` and `1`. The preferred type of update should be specified when the secant error is undefined (i.e., the first group of update).

4.3.11 `Mixing_Restart_Factor`

(*real*), default value = `0.10`

This parameter defines the restarting factor (ratio between the the norm of old function versus new function). If $||f_{new}||$ is too large relative to $||f_{old}||$, then the linear model or its approximation is not reliable, so restart should be performed. More precisely, if $||f_{old}|| <$ restart_factor $* ||f_{new}||$ then restart should be performed. In particular, if restart_factor is 0, then no restart is done.

4.3.12 `Mixing_Memory_Expand_Factor`

(*real*), default value = `2.0`

This line means that whenever the memory for mixing is insufficient, it will expand by a factor of 2.0, or by the value defined at input.

The last five parameters are only effective for multi-secant methods. Anderson and Broyden methods are special cases of multi-secant methods. One can reduce to these methods by choosing carefully the multi-secant parameters. For example, Broyden's first method is obtained by setting:

```
Mixing_Group_Size: 1
Mixing_EN_Like: 0
Mixing_Method: msecant1
```

whereas Broyden's second method is obtained by replacing `msecant1` above with `msecant2`. Anderson mixing is obtained by setting:

```
Mixing_Method: msecant2
Mixing_Group_Size: 0
Mixing_EN_Like: 0
```

The very first nonlinear Eirola-Nevanlinna (EN)-like algorithm proposed by Yang can be obtained by setting:

```
Mixing_Method: msecant1
Mixing_Group_Size: 1
Mixing_EN_Like: 1
```

4.4 Global atom and pseudopotential parameters

4.4.1 `Atom_Types_Num`

(*integer*), default value = `no default`

Number of different chemical elements present in the system.

4.4.2 `Coordinate_Unit`

(*word*), default value = `Cartesian_Bohr`

By default, positions of atoms are given in cartesian coordinates and units of Bohr radius. Other implemented options are:

- `Cartesian_Ang` atomic coordinates given in angstroms, cartesian.
- `Lattice_Vectors` atomic coordinates given in units of lattice vectors. Available only with periodic boundary conditions.

4.4.3 `Old_Interpolation_Format`

(*boolean*), default value = `false`

This is a compatibility flag. Normally, the local component of pseudopotentials V at some arbitrary point r is evaluated by linear interpolation of the function $r*V$. Due to the Coulomb-like behavior of V far away from the atom, this interpolation is exact beyond the cut-off radius. This flag reverts to the old method: linear interpolation of V itself. It should be used for debugging purposes only.

4.5 Parameters specific to each atom type

These parameters should be given in sequence in `parsec.in`, and they refer to a particular chemical element in the system.

4.5.1 `Atom_Type`

(*word*), default value = `no default`

Chemical symbol of the current element. THIS IS CASE SENSITIVE!

4.5.2 `Pseudopotential_Format`

(*word*), default value = `Martins_new`

Specification for the format of input pseudopotentials. Available choices are: `Martins_old`, `Martins_new`, `Martins_Wang` and `FHIPP`. See also input of pseudopotential filesInputPseudopotentials.

4.5.3 `Potential_Num`

(*integer*), default value = `no default`

Number of available angular components. Relevant for pseudopotential formats `Martins_old` and `FHIPP`.

4.5.4 `Core_Cutoff_Radius`

(*real*), default value = `no default` , default unit = *bohr* (Bohr radii)

Maximum cut-off radius used in the generation of pseudopotential. Relevant for pseudopotential formats `Martins_old`, `Martins_Wang` and `FHIPP`.

4.5.5 `Electron_Per_Orbital`

(*block*), default value = `no default`

Number of electrons per atomic orbital used in the generation of pseudopotential. Relevant for pseudopotential formats `Martins_old` and `FHIPP`.

4.5.6 `Local_Component`

(*integer*), default value = `no default`

Choice of angular component to be considered local in the Kleinman-Bylander transformation. No ghost-state check is done in parsec. Therefore, check if the chosen component does not have ghost states. Relevant for pseudopotential formats `Martins_old`, `Martin_new` and `FHIPP`.

4.5.7 `Nonlinear_Core_Correction`

(*boolean*), default value = `false`

This flag should be enabled if the pseudopotentials for this element have non-linear core correction. Relevant for pseudopotential format `FHIPP` only.

4.5.8 `Move_Flag`

(*integer*), default value = `0`

Type of movement to be applied to atoms. Relevant only if structural relaxation is performed. Value `0` makes all atoms movable in all directions. Value n positive makes the first n atoms movable. A negative value makes selected atoms movable (the selection is made inside block `Atom_Coord` AtomCoord).

4.5.9 `Atom_Coord`

(*block*), default value = `no default`

Coordinates of atoms belonging to the current chemical species. See `Coordinate_Unit` CoordinateUnit. If structural relaxation is done with selected atoms (`Move_Flag` MoveFlag < 0), each triplet of coordinates must be followed by an integer: 0 if this atom is fixed, 1 if it is movable.

4.5.10 `Alpha_Filter`

(*real*), default value = `0.0`

Cut-off parameter in Fourier filtering of pseudopotentials, in reciprocal space. If value is non-positive, pseudopotentials are not filtered. Fourier filtering is implemented according to Briggs *et al.* Phys. Rev. B **54**, 14362 (1996), but no second filtering in real space has been implemented so far.

4.5.11 `Beta1_Filter`

(*real*), default value = `0.0`

Cut-off parameter in Fourier filtering of pseudopotentials, in reciprocal space. Referenced only if `alpha_filter` is positive.

4.5.12 `Core_Filter`

(*real*), default value = equal to `Alpha_Filter`

Cut-off parameter in Fourier filtering of non-linear core density. If non-positive, then core density is not filtered.

4.5.13 `Initial_Spin_Polarization`

(*real*), default value = `0.1` ,

Value of spin polarization to be used in the initial guess of charge density. An initial spin polarization of 0.1 means that the charge ratio $(n^{\uparrow}-n^{\downarrow})/(n^{\uparrow}+n^{\downarrow})$ is 0.1. Warning: if you specify this parameter for one chemical element, then you also must specify it for all the other ones (use zero for non-spin polarized atoms).

4.5.14 `LDAplusU_U`

(*real*), default value = `0.0` , default unit = Ry (rydbergs)

Value of the U parameter (on-site Coulomb interaction) for this chemical element. A positive value for this quantity or for the J parameter below will add a on-site Coulomb interaction in the DFT Hamiltonian, following the "LDA+U" approach, see: Anisimov *et al.*, Phys. Rev. B **44**, 943 (1991).

4.5.15 `LDAplusU_J`

(*real*), default value = `0.0` , default unit = Ry (rydbergs)

Value of the J parameter (exchange part of on-site Coulomb interaction) for this chemical element.

4.5.16 `SO_psp`

(*boolean*), default value = `false`

Enables the inclusion of spin-orbit components in the pseudopotentials for this chemical element. Notice that `FHIPP` format does not currently have support for spin-orbit components. Spin-orbit components for pseudopotential format `Martins_old` require one additional file, with name `*_SO.DAT`. Notice that the inclusion of spin-orbit components require spin-polarized calculation, flag `Spin_Polarization`SpinPolarization.

4.5.17 `SCF_SO`

(*boolean*), default value = `false`

Enables the inclusion of spin-orbit components as perturbation. Spin-orbit components are included in a SCF calculation only after the solution is obtained without spin-orbit components. This can lead to easier convergence.

4.5.18 `Cubic_Spline`

(*boolean*), default value = `false`

Enables cubic spline interpolation in the pseudopotentials. This is useful if the radial grid where pseudopotentials are computed is coarse, especially in the vicinity of the cut-off radius.

4.6 Electronic parameters

4.6.1 `Restart_Run`

(*boolean*), default value = `false`

By default, the initial charge density in a parsec run is built from the superposition of atomic charge densities. If you want instead to start from a previous run, choose value true. For confined systems, having `restart_run = true` prevents the initial centering of atoms. If this flag is used but an adequate file `parsec.dat` is not found, the calculation is interrupted.

4.6.2 `States_Num`

(*integer*), default value = `no default`

Number of energy eigenvalues to be computed. In spin-polarized calculations, this is the number of eigenvalues per spin orientation.

4.6.3 `Net_Charges`

(*real*), default value = `0.0` , default unit = e (electron charge)

Net electric charge in the system, in units of electron charge. It can also be used in periodic systems. In that case, a background charge is used and the divergent component of the Hartree potential is ignored.

4.6.4 `Fermi_Temp`

(*real*), default value = `80.0` , default unit = K (kelvins)

Fermi temperature used to smear out the Fermi energy. If a negative value is given, then the occupancy of Kohn-Sham orbitals is read from file `occup.in` and kept fixed throughout the calculation.

4.6.5 `Correlation_Type`

(*word*), default value = `CA` (without spin polarization), `PL` (with spin polarization)

Choice of exchange-Correlation functional to be used. Available choices and their variants are:

- `CA, PZ` : Ceperley-Alder (LDA), with spin-polarized option. D.M. Ceperley and B.J. Alder, Phys. Rev. Lett. **45**, 566 (1980); J.P. Perdew and A. Zunger, Phys. Rev. B **23**, 5048 (1981).

- `XA`: Slater's x-alpha (LDA). J.C. Slater, Phys. Rev. **81** 385 (1951).
- `WI`: Wigner's interpolation (LDA). E. Wigner, Phys. Rev. **46**, 1002 (1934).
- `HL`: Hedin-Lundqvist (LDA). L. Hedin and B.I. Lundqvist, J. Phys. C **4**, 2064 (1971).
- `LB`: Ceperley-Alder with Leeuwen-Baerends correction (asymptotically corrected). R. van Leeuwen and E. J. Baerends, Phys. Rev. A **49**, 2421 (1994).
- `CS`: Ceperley-Alder with Casida-Salahub correction (asymptotically corrected). M.E. Casida and D.R. Salahub, J. Chem. Phys. **113**, 8918 (2000).
- `PL, PWLDA`: Perdew-Wang (LDA), with spin-polarized option. J.P. Perdew and Y. Wang, Phys. Rev. B **45**, 13244 (1992).
- `PW, PW91, PWGGA`: Perdew-Wang (GGA), with spin-polarized option. J. Perdew, *Electronic Structure of Solids* (ed. P. Ziesche and H. Eschrig, Akademie Verlag, Berlin, 1991).
- `PB, PBE`: Perdew-Burke-Ernzerhof (GGA), with spin-polarized option. J.P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).

4.6.6 `Ion_Energy_Diff`

(*real*), default value = `0.0`, default unit = *Ry* (rydbergs)

Energy difference used in the construction of the Casida-Salahub functional. Relevant only if that functional is used.

4.6.7 `Spin_Polarization`

(*boolean*), default value = `false`

Enables spin-polarized densities and functional.

4.6.8 `Symmetrize_Charge_Density`

(*boolean*), default value = `false`

Forces charge density to be resymmetrized after each SCF step. In some cases (for example, if the Fermi temperature is too small), the charge density may lose symmetry after some steps due to round-off error accumulation. This option is valid only in confined (totally non-periodic) systems. For periodic systems, the charge density is always symmetrized.

4.6.9 `Add_Point_Charges`

(*boolean*), default value = `false`

Enables embedding of the system in electrostatic point charges. This feature can be used when the actual system is surrounded by additional ions which do not take part in electronic processes and can be taken into account as classical point charges, without a full quantum-mechanical treatment. Also, the system must be confined: `Boundary_Conditions cluster`BoundaryConditions.

4.6.10 `Point_Typ_Num`

(*integer*), default value = `0`

Number of classical point charge types. Referenced only if point charge embedding is enabled.

4.6.11 `Pt_Chg`

(*real*), default value = `0.0`, default unit = e (electron charge)

Charge of each point charge type.

4.6.12 `Point_Coord`

(*block*), default value = `no default`

Coordinates of each point charge. They can also be rescaled with a coordinate unit. See `Coordinate_Unit` CoordinateUnit.

4.6.13 `External_Mag_Field`

(*real*), default value = `0.0`, default unit = *Ry/mu_Bohr* (rydbergs per Bohr magneton)

Strength of the external magnetic field applied on the system, along the **z** direction. This magnetic field acts only on the spin magnetic dipole, and not on orbital magnetic dipole (orbital angular momentum is very small for electrons).

4.7 Wave functions and k-points

The parameters here affect the wave functions and the k-point sampling.

4.7.1 `Complex_Wfn`

(*boolean*), default value = `true`

By default use complex value for the wavefunctions. This is necessary for sampling of k-space and for other instances. For the case that only the Γ point is sampled the calculation will run much faster and uses fewer resources if this parameter is turned to false.

4.7.2 Kpoint_Method

(*string*), default value = none

The method to sample k-space. There are two options

- none: No additional k-point is used besides the Γ (0,0,0) point. Also, wave-functions are computed as real functions (null imaginary part).
- mp: Sample by the Monkhorst-Pack algorithm (see H.J. Monkhorst and J.D. Pack, Phys. Rev. B **13**, 5188 (1976)) This will require including a Monkhorst_Pack_Grid in a block.
- manual: Manual sampling of k-space. This will require including a kpoint_list in a block.

4.7.3 Monkhorst_Pack_Grid

(*block*), default value = none

Grid to sample kpoints over via the Monkhorst-Pack algorithm. Relevant only if Kpoint_MethodKpointMethod = mp.

4.7.4 Monkhorst_Pack_Shift

(*block*), default value = none

Shift of Monkhorst-Pack grid. Kpoints in this grid have coordinates given by: $k_i = (i + s)/n$ in units of reciprocal lattice vectors, where n is a grid order (see Monkhorst_Pack_Grid MonkhorstPackGrid), s is a shift and i is an integer with value between 0 and $n - 1$. Relevant only if Kpoint_MethodKpointMethod = mp.

4.7.5 Kpoint_List

(*block*), default value = none

List of k-points to sample and the sampling weight of each k-point. Relevant only if Kpoint_MethodKpointMethod = manual.

4.7.6 Kpoint_Unit

(*string*), default value = Cartesian_Inverse_Bohr

Defines the type of coordinates used to input k-points. Available options are Cartesian_Inverse_Bohr (Cartesian coordinates, units of inverse Bohr radius) and Reciprocal_Lattice_Vectors (units of reciprocal lattice vectors).

4.8 Band structure and Density of States (DOS) calculation

Special commands are given for band structure calculation and density of states (DOS) calculation. Both calculations can be carried out at the end of an SCF run - so no need for a special run.

4.8.1 `bandstruc`

(*block*), default value = `no default`

bandstruc is a block containing the path in k-space to be calculated for the band structure. The block is composed of lines. Each line defines a segment in the band path and contains 7 numbers and a label, the first number is a serial number. the next 6 numbers contain the starting k-point of the segment and final k-point of the segment, given in *relative* coordinates. the last part of each line is a label, describing the segment.

An example is shown below for the 2d case of graphene:

```
begin bandstruc
1 0.0 0.0 0.0 0.5 0.0 0.0 gamma-M
2 0.5 0.0 0.0 0.333333333 -0.333333333 0.0 M-K
3 0.333333333 -0.333333333 0.0 0.0 0.0 0.0 K-gamma
end bandstruc
```

In the case of 2d - the 3rd number of each k-point should be 0.0. The band structure calculation is done after the SCF calculation has ended. At the end of calculation a text file that is called "bands.dat" is created. The file is arranged a line of number. Each line is arranged as follows:

```
spin, line_num, kpoint_num, kpt_x, kpt_y, kpt_z, eigs(kpt,spin,1:n)-efermi
```

spin - is simply 1 or 2, line_number is the path segment number, kpoint_num is the k-point number, kpt_x, kpt_y and kpt_z are the k-point components, note that they are given in *cartesian* coordinates and not in relative. eigs are simply the eigenvalues, they are given relative to the fermi level.

4.8.2 `bandstruc_points`

(*integer*), default value = `20`

This parameter defines the number of points in the *shortest* segment in the band path that is given. The calculation is done so that the points density along the path is kept constant.

4.8.3 `create_dos`

(*boolean*), default value = `false`

Density of States (DOS) is calculated when this flag is true. DOS calculation can be performed only for periodic (3d) or partially periodic (1d and 2d) systems.

4.8.4 `dos_pnum`

(*integer*), default value = `1000`

Number of points in the energy axis for the DOS calculation.

4.9 Non-self consistent calculations and BerkeleyGW interface

Special commands are given for non-self consistent calculations and interface to the `BerkeleyGW` code http://www.berkeleygw.org. Unlike band structure and density of states (DOS) calculation, the non-self consistent calculations has to be carried out after the end of an SCF run. The code reads `parsec.dat` file for charge density and potential information. This file is NOT overwritten. It should be kept in mind that one has to specify the parameters for the scf calculation as well. The code uses these when reading `parsec.dat` to ensure that it is the intended file. The non-self consistent code is independent of the BerkeleyGW interface and can be used separately as well.

4.9.1 `nscf_kpoints`

(*block*), default value = `no default`

nscf_kpoints is a block containing the points in k-space and their weights to be calculated in the non-self consistent calclation. The weights are not used unless one is outputing information for BerkeleyGW calculation. The format for each line in this block is the same as `Kpoint_list`.

4.9.2 `Nscf_Kpoint_Unit`

(*string*), default value = `Cartesian_Inverse_Bohr`

Defines the type of coordinates used to input `nscf_kpoints`. Available options are `Cartesian_Inverse_Bohr` (Cartesian coordinates, units of inverse Bohr radius) and `Reciprocal_Lattice_Vectors` (units of reciprocal lattice vectors).

4.9.3 `nscf_states_num`

(*integer*), default value = `none`

This parameter defines the number of states to be calculated in the non-self consistent calculation.

4.9.4 `nscf_fermi_level`

(*real*), default value = `none`

Because no self-consistent calculation is done in the same run, the code does not have any idea about the Fermi Level. So a Fermi level has to be provided from the previous scf calculation. Default units are Ry.

4.9.5 `output_gw`

(*boolean*), default value = `false`

This flag determines whether output for `BerkeleyGW` is written. In order to use this flag, the code must be compiled with -DBGW flag. It also has

to be linked properly to BerkeleyGW code libraries which contain subroutines for input/output in BerkeleyGW format. When true, this flag will write WFN, VXC, and RHO files. These files contain the wavefunction information (in g-space), exchange-correlation potential (in g-space) and charge density (in g-space) respectively. These can be used as input to the BerkeleyGW code.

4.9.6 nscf_kgrid

(*block*), default value = none

Grid used to sample kpoints via the Monkhorst-Pack algorithm for non-self consistent calculation. This information is just passed to BerkeleyGW. The calculation is performed on the nscf_kpointsnscfkpoints. Relevant only if output_gwoutputgw = true.

4.9.7 nscf_kgrid_shift

(*block*), default value = none

Shift of Monkhorst-Pack grid for non-self consistent calculations. Kpoints in this grid have coordinates given by: $k_i = (i + s)/n$ in units of reciprocal lattice vectors, where n is a grid order (see nscf_kgridnscfkgrid), s is a shift and i is an integer with value between 0 and $n - 1$. This information is just passed to BerkeleyGW. The calculation is performed on the nscf_kpointsnscfkpoints. Relevant only if output_gwoutputgw = true.

4.10 Structural relaxation

4.10.1 Minimization

(*word*), default value = none

Choice of method for structural relaxation. Available options are: none (no relaxation), simple (steepest-descent algorithm), BFGS (Broyden-Fletcher-Goldfarb-Shanno, the best), and manual (manual movement). Steepest-descent should be used as last resort. With manual movement, the sequence of atomic coordinates must be provided by the user in file manual.dat and it is used "as is". The last choice is useful for debugging. Symmetry operations are ignored if manual movement is performed

4.10.2 Movement_Num

(*integer*), default value = 500

Maximum number of iterations during structural relaxation.

4.10.3 Force_Min

(*real*), default value = 1.0d-2 , default unit = *Ry/bohr* (rydbergs/Bohr radii)

Maximum magnitude of the force on any atom (that was allowed to move) in the relaxed structure. This should not be set less than about 0.01 *Ry/a.u.*, which is the usual accuracy in the calculation of forces.

4.10.4 `Max_Step`

(*real*), default value = `0.05` (steepest descent), `-1` (BFGS), default unit = *bohr* (Bohr radii)

If steepest-descent minimization is performed, this parameter sets the maximum displacement of atoms between two steps in relaxation. With BFGS, this defines the maximum displacement of atoms along each Cartesian component from their original position (if negative, no constraint in the movement of atoms is imposed). Ignored in all other circumstances.

4.10.5 `Min_Step`

(*real*), default value = `1.d-4` , default unit = *bohr* (Bohr radii)

Minimum displacement of atoms between two steps in steepest-descent relaxation. Relevant only if steepest descent is performed.

4.10.6 `BFGS_Number_Corr`

(*integer*), default value = `7`

Number of variable metric corrections used to define the limited-memory matrix. Relevant only if BFGS relaxation is performed.

4.10.7 `Relax_Restart`

(*boolean*), default value = `false`

Flag for the restart of a relaxation run. If true, the user must provide information about previous run in file `relax_restart.dat`. No need to provide input file `parsec.dat`. If this option is used, the atomic coordinates in `parsec.in` are replaced with the ones in `relax_restart.dat`. Appropriate warnings are written to output. If the restart file is not found, this flag is ignored and no information from previous relaxations is used.

4.11 Molecular dynamics

4.11.1 `Molecular_Dynamics`

(*boolean*), default value = `false`

Enables molecular dynamics. This feature can be used both for simulated annealing and for direct simulation of hot systems. Both canonical and microcanonical ensembles are implemented. Symmetry operations are ignored if molecular dynamics is being performed.

4.11.2 `Cooling_Method`

(*word*), default value = `stair`

Type of cooling (i.e.: change of ensemble temperature) used during molecular dynamics. Possible choices are: `stair` (stair-case), `log` (logarithmic cooling), and `linear` (linear cooling). As the molecular dynamics simulation proceeds,

the ensemble temperature changes according to the cooling method and to the choices of initial and final temperatures. At each step, the actual temperature is calculated from equipartition theorem and the kinetic energy (difference between the total energy and potential energy). In the first steps, the actual and ensemble temperatures may be very different, particularly if there are few atoms, when thermal fluctuations are enormous.

4.11.3 `Tinit`

(*real*), default value = `500.0` , default unit = K (kelvins)

Initial ensemble temperature in the system.

4.11.4 `T_final`

(*real*), default value = `500.0` , default unit = K (kelvins)

Final ensemble temperature.

4.11.5 `T_Step`

(*real*), default value = `150.0` , default unit = K (kelvins)

Step temperature used in the cooling. Used only with stair-case cooling method.

4.11.6 `Time_Step`

(*real*), default value = `150.0` , default unit = $\hbar/hartrees$ ($0.0242fs = 2.42 \times 10^{-17}$ seconds)

Time interval between two steps in molecular dynamics. The dynamical equations of motion are numerically integrated using a modified Verlet algorithm.

4.11.7 `Friction_Coefficient`

(*real*), default value = `0.0` default unit = $hartrees/\hbar$ ($41.3PHz = 4.13 \times 10^{16}Hz$)

Viscosity parameter in molecular dynamics, in atomic units (inverse time). If less than 10^{-6} $hartrees/\hbar$ (approximately 41 GHz), then a micro-canonical ensemble is used: total energy is assumed constant, and the atoms start moving with a velocity distribution consistent with the initial temperature. If more, then a canonical ensemble is used and true Langevin molecular dynamics is performed.

4.11.8 `Step_num`

(*integer*), default value = `250`

Number of steps (iterations) in molecular dynamics. A successful simulation ends only after reaching the requested number of steps.

4.11.9 `Restart_mode`

(*boolean*), default value = `false`

Enables the restart of an aborted molecular dynamics simulation. With this feature, the last two sets of positions, velocities and accelerations stored in input file `mdinit.dat` are used. The number of steps in that file is included in the current run as well. WARNING: make sure that the initial/final temperatures and other MD parameters are correct in the restarted run.

4.12 Polarizability

4.12.1 `Polarizability`

(*boolean*), default value = `false`

Enables calculation of polarizability using a finite-field method. Available only with non-periodic boundary conditions. An uniform electric field is applied successively along the +x,-x,+y,-y,+z,-z directions, and dipole moments are calculated and collected and printed in output file `polar.dat`. From dipole moments, the diagonal part of polarizability tensor is obtained. Symmetry operations are ignored if this flag is used.

4.12.2 `Electric_Field`

(*real*), default value = `0.001`, *default unit = ry/bohr/e* (rydbergs/Bohr radii/electron charge)

Magnitude of external electric field to be applied.

4.13 MPI parallelization options

4.13.1 `MPI_Groups`

(*integer*), default value = `maximum valid value`

Number of groups (or subsets) of processors. PARSEC allows for parallelization with respect to number of k-points N_k (periodic systems only), of spin channels N_s (spin-polarized systems only) and of symmetry representations N_r, besides the standard parallelization with respect to grid points. Each triplet (k, s, r) of k-point, spin channel and representation is assigned to a single group. Processors belonging to the same group work together in the calculation of eigenvalues for the Hamiltonian $H_{k,s,r}$ assigned to this group. The number of groups must be a factor of the total number of processors (N_p) and of $N_k * N_s * N_r$, so that all groups have the same number of processors. By default, the number of groups is chosen to be the highest common factor between N_p and $N_k * N_s * N_r$. If a lower number is specified at input, the actual number of groups may be reduced so that proper load balance is present.

4.14 Flags for additional input/output

4.14.1 `Output_All_States`

(*boolean*), default value = `false`

Forces output of all wave-functions up to requested number of states to file `parsec.dat`, instead of only the occupied ones (default).

4.14.2 `Save_Wfn_Bands`

(*block*), default value = `no default` (optional)

By default, file `parsec.dat` contains wave-functions of all occupied energy levels. Alternatively, the user can select specific wave-functions to be printed by giving a list of energy bands (levels, for non-periodic systems) inside the block. For example, this:

```
begin save_wfn_bands
1
2
4
end save_wfn_bands
```

will result in only bands 1,2, and 4 being printed out. The user can also specify a range of bands by giving the initial and final levels. This:

```
begin save_wfn_bands
2 8
end save_wfn_bands
```

will result in all levels between 2 and 8 being printed. If no wave-functions are to be printed, leave the block empty:

```
begin save_wfn_bands
end save_wfn_bands
```

This option is ignored if the input `Output_All_States .true.` is used.

4.14.3 `Save_Intermediate_Charge_Density`

(*boolean*), default value = `false`

Causes wave-functions to be written to output after each self-consistent iteration, instead of only after self-consistency is reached. Previous wave-functions are over-written. This is useful for debugging purposes or if diagonalization is too demanding.

4.14.4 `Output_Eigenvalues`

(*boolean*), default value = `false`

Saves eigenvalues to file `eigen.dat` after self-consistency in density function is achieved. This is useful for debugging purposes.

4.14.5 `Save_Intermediate_Eigenvalues`

(*boolean*), default value = `false`

Saves eigenvalues after each SCF iteration, instead of after the last one. Useful for debugging purposes.

4.14.6 `Output_Level`

(*integer*), default value = `0`

Output flag for additional info in parsec.out. With default value, only the basic output is written. Positive values cause additional information to be printed out:

```
> 0 : print out local/non-local components of force
> 1 : print out all symmetry data and additional eigensolver output
> 2 : print out more BFGS output
> 3 : print out all BFGS output
> 4 : print out all g-stars (warning: _lots_ of data)
> 5 : print out local potentials (warning: _lots_ of data)
```

4.14.7 `Output_File_Name`

(*word*), default value = `parsec.out`

Choice of output filename. Any name is valid, but spaces must not be included, as well as some special symbols.

4.14.8 `Restart_From_Wfndat`

(*boolean*), default value = `false`

Makes possible to reuse an old file `wfn.dat` in a PARSEC run. Notice that file `parsec.dat` is still being written, and file `wfn.dat` is not modified.

5 Post-processing tools

5.1 Other tools

There are additional tools that read parsec output and perform specific tasks. Some of them are: `angproj`, for calculation of angle-projected density of states; `plotdx`, interface with Data Explorerhttp://www.opendx.org for visualization of charge density, wave-functions, etc.

6 Problems, unfinished work, etc.

7 Troubleshooting

1. Diagla occasionally shows bad performance if periodic boundary conditions are used. If you get error messages related to this eigensolver, change

it.

2. Missing symmetry operations: only symmetry operations belonging to the cubic system are tested for existence. That is because the finite grid spacing breaks all other symmetries (for example, the five-fold rotational symmetry in icosahedral molecules). In the Hamiltonian, only some of the present operations are used, and the selection is very stringent. If you feel not all possible symmetry operations are included in the Hamiltonian, try rotating or translating the whole system.

3. Unrecognized chemical element: Parsec has an internal periodic table (file `ptable.f`), and it checks for the existence of input elements. If the calculation stops with "unknown element", edit the periodic table.

4. Current pseudopotential generator does not support asymptotically corrected functionals (LB, CS).

5. The EDSL library sometimes screws up when identifying certain features with certain atom types. All the relevant information about pseudopotentials is written to file `parsec.out`. The user should search that file for inconsistencies in input data.

8 Modifications

8.1 Modifications in version 1.2v1

Version 1.2 is unstable and should be replaced with 1.2v1. This is a list of modifications in version 1.2v1 compared to version 1.1:

1. Two new eigensolvers are available: thick-restart Lanczos and Chebyshev-Davidson filtering. Eigensolvers implemented by Yunkai Zhou and tested by Yunkai and Murilo Tiago.

2. Chebyshev subspace filtering has been implemented. This feature should be always used since it improves considerably the solution of Kohn-Sham equations at each SCF iteration. This was also implemented by Yunkai Zhou, and tested by Yunkai and Murilo Tiago.

3. Numerical tolerance in the search for symmetry operations has increased. Previous tolerance was causing crashes.

4. The output on file parsec.out has been modified in various places, with minor changes.

5. Usage of "stop" statements was replaced with calls to exit_err. This avoids sudden crashes without any meaningful error message.

8.2 Modifications in version 1.2.7.2

Compared to versions 1.2v1 and 1.2.7.1, this version has minor bug fixes and some features added. The most important ones are:

1. Embedding of classical point charges (`Add_Point_Charges` AddPointCharges.
2. Initial spin polarization for each chemical element (`Initial_Spin_Polarization` InitialSpinPolarization.

In addition, support for complex algebra is added to all eigensolvers except TRLANC (version 1.2.7.1 also has it). This is a first step towards full support of k-point sampling.

8.3 Modifications in version 1.2.8

Version 1.2.8 is an early merge of versions 1.2.4, 1.2.7.1 and 1.2.7.2. Compared to version 1.2.7.2, these are the most important modifications:

1. Support for non-orthogonal unit cells (periodic systems) is implemented. Non-orthogonal unit vectors can be defined at input with the block `Cell_Shape` CellShape (notice that three lines must be given in the block, instead of one!). Some additional testing must be done for this feature.
2. Option for name of output file. See `Output_File_Name` OutputFileName.
3. Wavefunctions are now printed out in file `parsec.dat` instead of `wfn.dat`. This modification is necessary because additional information must be saved, such as coordinates of unit lattice vectors. The user can still restart a calculation from a previous `wfn.dat` file by using the input option `Restart_From_Wfndat` RestartFromWfndat.
4. Choice of printing out some wave-functions to file `parsec.dat`. See `Save_Wfn_Bands` SaveWfnBands.

Support for k-point sampling is not yet finished, but input variables are already defined. See section "Wave functions and k-points".

Unresolved issue: the ARPACK eigensolver seems to occasionally miss eigenvalues if complex algebra is turned on (`Complex_Wfn .true.` ComplexWfn).

8.4 Modifications in version 1.2.8.2

Version 1.2.8.2 is a bug-fix branch that was created from 1.2.8. Modifications from version 1.2.8:

1. Double grid is implemented. See `Double_Grid_Order` DoubleGridOrder.
2. Support for pseudopotential formats `Martins_Wang` and `FHIPP` is added. See input of pseudopotential files InputPseudopotentials.

3. Support for the "LDA+U" approximation is added. See `LDAplusU_U` `LDAplusU_J` LDAplusUU.

4. Support for spin-orbit calculations. This is still in development stage.

8.5 Modifications in version 1.2.9

Version 1.2.9 is the first version of this code with support for k-points different from the Γ point. More details about the input of k-points is found in section "Wave functions and k-points" KpointMethod. Support for spin-orbit calculations, multiple k-points and LDA+U is still under tests.

8.6 Modifications in version 1.2.9.2

Version 1.2.9.2 has more stable support for spin-orbit calculations, multiple k-points and LDA+U. In addition, it has two new input options: `Kpoint_Unit`KpointUnit and `Boundary_Mask`BoundaryMask.

8.7 Modifications in version 1.2.9.3

Version 1.2.9.3 has an alternative choice for the parallelization: besides the standard parallelization with respect to grid points, this version can also work with parallelization with respect to number of k-points, of spin channels and of symmetry representations. See `MPI_Groups`MPIGroups. In addition, the option of interrupting "smoothly" a calculation is added. See section "Interrupting calculation" StopSCF.

8.8 Modifications in version 1.2.9.4

Wire boundary conditions are implemented. See `Boundary_Conditions`BoundaryConditions. Also, all source files were converted to free form, except for the LBFGS library.

8.9 Modifications in version 1.3

- Inclusion of multi-secant mixing methods, as developed by Haw-ren Fang and Yousef Saad, from EECS, Univ. of Minnesota.

- Addition of restart from previous spin-orbit calculation. Fixed a bug in the write-out of spin-orbit wavefunctions. Older versions do not write correct spin-orbit wavefunctions.

- Removal of old/unused input flags: skip_rr_rotation, skip_force, spinorbit, old_pseudopotential_format.

- Addition of a gfortran makefile. Removal of unused variables and commented-out old coding.

- Inclusion of time-reversal symmetry in Monkhorst-Pack k-grid.

8.10 Modifications in version 1.3.4

- A number of bugs were fixed:

  1. Definition of non-orthogonal directions in periodic systems has been modified. This is an attempt to fix the problem with incorrect ground state in BCC crystals.
  2. Bugs related to the definition of grid spacing in periodic systems, reading of old charge density (it used to crash in wires), calculation of total energy in wires, and array-out-of-bound errors in BFGS relaxation were all fixed.

- A cubic spline interpolation was implemented. See 4.5.18.

- Functional Perdew-Zunger spin-polarized (LSDA) was added. For the sake of back-compatibility, the default LSDA functional is still Perdew-Wang.