Paratec 5.1.12 Documentation

Bernd Pfrommer
David Roundy
Young-Gui Yoon
David Raczkowski

November 25, 2003

Contents

Chapter 1

How to get and run paratec

1.1 Where to find paratec

At the moment, there is no standard repository for paratec. ☹ Interested users can contact David Raczkowski at dbraczkowski@lbl.gov or Andrew Canning at canning@nersc.gov

Paratec documentation can be found on the web at
http://www.nersc.gov/projects/paratec

1.2 How to run paratec

1.2.1 Input files

Required files:

```
paratec             the executable
input               the input file
Si_POT.DAT          the pseudopotential files
O_POT.DAT           ...
```

1.2.2 Output files

```
STRUC_CONVERGE    contains the progress of the structural relaxation
STRUC_LOG         a detailed log about the relaxation steps performed
ELASTO_PHONON     information from the H matrix: Bulk modulus, phonon
frequencies
PW_LOG            a log about all the PW calls done
OUT               output file of the current plane wave job
CHECKPOINT        contains all the information to continue a
                  structural relaxation. Also has the Hessian matrix
*.dx              files for visualization with the IBM Data Explorer
*.kh              files for visualization with Khoros
```

Note: As of paratec 4.5c, the output file `OUT._0` has been changed to `OUT`.

1.2.3 How to compile and run on different machines

Having obtained a gzipped tar file of the source e.g. `paratec.xxx.tar.gz`, unpack the source by typing

```
gzip -d -c paratec.xxx.tar.gz | tar -xf -
```

Then enter the `paratec` directory and run the configuration script by typing

```
cd paratec
./configure
```

For a summary of the options accepted by the global `Makefile`, see Table 1.1 or just type

```
make
```

Option to `make`	Function
`paratec`	Build Paratec
`tools`	Build related tools
`html`	Build HTML documentation
`microbench`	Build microbenchmark for FFTs and MMMs
`clean`	Remove object and test files
`dist`	Change version number and pack into a tar file
`dist-pub`	above plus removes NMR and TDDFT code

Table 1.1: Options for global Paratec `Makefile`

The machines currently supported are listed in Table 1.2 for reference. For each machine, a working directory in the source tree is created e.g. `paratec/src/para/`*machine* where the value of *machine* is given in the second column of Table 1.2. The binary files are placed in `paratec/bin/`*machine*. This facility allows the same source tree to be used by more than one machine simultaneously without any conflicts arising between object files.

1.2.4 Some tips on porting paratec

Due to heavy use of optimized library routines, paratec is not easy to port. Here are some steps on the way to port to a new architecture.

1. go into the config directory, and copy an established sysvars.machine and blurb.machine. Edit to meet your needs. The blurb file is displayed during configuration, and can be empty.

2. Edit the "configure" script to include the new machine.

3. Have a look at the files below...

Machine	*machine*	Description
Sun	`sun`	Sun
IBM RS6000	`rs6k`	Single-node IBM RS6000
IBM SP2	`sp2`	IBM SP2 at Cornell
IBM SP3 (NERSC)	`sp3`	Glen Seaborg at NERSC
IBM SP3 (NPACI)	`sp3`	Blue Horizon in San Diego
Cray T3E	`t3e`	Cray T3E at NERSC
SGI o200	`sgio200`	SGI Origin 200
SGI o2000	`sgio2000`	SGI Origin 2000 at NCSA
SGI PowerChallenge	`sgipc`	SGI PowerChallenge at NCSA
HP/Convex Exemplar	`cvx`	
Linux x86	`I386`	Linux on x86
DEC Alpha	`alpha`	DEC Alpha
Hitachi SR2201	`sr2201`	Hitachi SR2201 at Cambridge
SGI o2000	`hodgkin`	SGI Origin 2000 using new libraries

Table 1.2: Machines currently supported

Files in src/para

File	advice
flibcalls.m4h	Has some system dependencies on the use of blas routines, i.e Cray versus the rest of the world. Will most likely need no changes.
all_to_all.f90p data_gather.f90p gspace_gather.f90p isort_gather.f90p setup_packinfo.f90p symmetrize_tensor_global.f90p	May need some tweeking if there are hickups in the MPI implementation. Look for the machine-specific ifdefs there. Most likely will need no changes.
zdiag_scalapack.fp rdiag_scalapack.fp	Scalapack on the Cray uses Shmem, and that requires some ugly workarounds.

file in src/macros/fft

fft_macros.m4h

This one is quite a pain, especially since it uses m4. The standard quotation characters ‘’ have been replaced by []. The file fourier_transform.f90p gets preprocessed into fourier_transform.p.f90. To check correctness of the macros, look at fourier_transform.p.f90 Look up the man on the fourier transforms (e.g man fft), add new macros accordingly to fft_macros.m4h.

Don’t forget to provide for workspace in fft_workspace.fp90, and for a list

of acceptable FFT lengths in adjust_fft.f90p. Once you have that done, you are through the worst. Have a look at fourier_transform.f90p. If the fourier transform interfaces are awkward, you might have to make some changes there as well. Make use of optimized routines to perform the convolution, rather than using the vanilla loop. Try if the preprocessor option FAST_CONVOLUTE results in any speedup - it usually does for machines where no multiple-FFT routines are available.

At some point, the code was running also on cray vector machines. This is no longer the case, although still all the code is there, and it should not be too much work to get it going again.

Tips on porting FFTW

FFTW (the Fastest Fourier Transform in the West) is a nice highly optimized portable FFT library. Paratec can now be compiled to use FFTW on platforms where it is available with only minimal changes needed. This should make it easier to port paratec.

To make FFTW work on a new platform, make sure the sysvars file includes the path of the FFTW library (for example, see sysvars.sgio200). Secondly, you must make sure that if the platform also supports a native FFT package that M4FFTFLAG is defined (again, see sgio200), and that adjustfft.fp uses that value.

I think that is all that should be needed.

1.2.5 How to change paratec

If you make a change to paratec, please make note of the change you make in the `CHANGES` file. This way you (and others) can tell what has changed in each version.

When you want to archive a version of the code, or give it to someone else, use the `make dist` target of the makefile. This saves only the necesary files, and not all the object files or binaries you may have generated. It then tars up the results and gzips them for your convenience. Use `make dist-pub` if you have the full version (with NMR and the untested td-dft), and you want to strip the code of these parts.

`make dist` also prompts you for a new version number. It modifies the current version number (which is printed out in the `OUT` and `PW_LOG` files, and shows up at the top of this documentation. It also updates the time stamp, which shows up in the OUT and PW_LOG files. Then it names the tarball after the new version number. You can, of course, give it the same name as the old version, but if you change the number you will be able to keep track of which binary originated from which version of the code.

WARNING! `make dist` assumes your source code directory is called "paratec". If you keep it in a directory with any other name, these may end up saving the wrong source code, or doing something else nasty.

If you add a new file to paratec, make sure to add it to the lists in the `Makefile` for the `dist` target.

1.3 Changes in Paratec 5.0

On the whole, paratec 5.0 uses the same input files as earlier versions of paratec. However, because the input code was completely rewritten in FORTRAN 90, several of the input structures *have* changed. This is where you look to find out what changes you will have to make to your input files. A more comprehensive list can be found on the home page

- The file formerly named `OUT._0` is now simply `OUT`. This is only an issue if you have scripts that assume it is called `OUT._0`.
- All block inputs (i.e. inputs that start with `begin blah` now must close with `end blah`. Formerly some of these structures were closed with simply `end`.
- The statement `coordinates absolute` has acquired an underscore, so it is now `coordinates_absolute`.

Chapter 2

Basic paratec usage

2.1 Entering the crystal structure

The crystal structure is specified by the lattice vectors and the atomic coordinates. The lattice vectors are given as rows of coordinates with respect to a cartesian basis. The unit lenght is an atomic length unit, i.e. the Bohr radius.

In addition to the lattice vectors, the volume may be specified. If you provide a volume, the lattice vectors will be rescaled such that they span a unit cell with the given volume. If you leave out the volume statement, the lattice vectors are assumed to be given in atomic units.

The lattice vectors and optionally the volume must be enclosed within a `begin latticevecs` and `end latticevecs` statement.

Example:

```
begin latticevecs
coord       7.22362499411784  -5.04013632834720   0.00000000000000
coord       0.00000000000000   3.95068558900000   2.76187238600000
coord       0.00000000000000  -3.95068558900000   2.76187238600000
volume 157.3848
end latticevecs
```

The coordinates of the atoms must be enclosed by the statements `begin coordinates` and `end coordinates`. Each new element is indicated with a keyword `newtype`. All `coord` statements following a `newtype` statement refer to the new element, until the next `newtype` keyword is encountered.

The `newtype` keyword is followed by the atomic type, e.g. `O`,`Si`,`C`, by the spin polarization of the initial charge configuration for this atom, and the size of the muffin-tin sphere (used only for projected DOS, and, if unspecified, is set to 1.0 a.u by default). For instance

`newtype Si 0.5 2.4`

will define a Si atom. For setting up the initial charge density, a definition of spin polarization of $(\rho_{up} - \rho_{down})/(\rho_{up} + \rho_{down}) = 0.5$ will be used, with a

default of 0.2. The muffin-tin sphere around the silicon atom will have a radius of 2.4 a.u. For non-spin-polarized calculations, set the spin polarization to 0.

By default, the coordinates following `coord` are understood to be relative to the basis vectors. To facilitate the input of molecular crystals, it is possible to give them in absolute, cartesian coordinates (a.u.) by specifying the keyword `coordinates_absolute` OUTSIDE the begin/end of the coordinates.

NOTE: This is a change from pre-5.0 versions of paratec! Previously one specified `coordinates absolute`, whereas now an underscore is required (i.e. `coordinates_absolute`)!

Example:

```
begin coordinates
newtype O 0.17
coord       0.15693515545994   0.10044534718500  -0.10044534718500
coord      -0.15693515545994  -0.10044534718500   0.10044534718500
newtype  Si 0.0
coord       0.5  0.5  0.5
end coordinates
```

2.2 The job variable

The job variable in the input file determines, what the code does:

```
job relax                   Does a structural relaxation job

relax_from_checkpoint       Continues relaxation from CHECKPOINT file.
                            This helps in case of a crash.

job relax_recycle_H         Relaxes, and uses Hessian matrix from
                            the CHECKPOINT file.  This improves efficiency
                            if you perform several relaxations on
                            similar systems.

job MD                      Does a molecular dynamics run
```

Default: `job relax`

If you don't want to relax, you simply set `relax_max_iter` to be 1.

NOTE: If you specify `job relax_from_checkpoint` and there is no CHECKPOINT available, paratec will abort without performing any calculation.

2.2.1 Deciding how much to relax

If the square modulus of the gradient (composed of forces and stress) falls below `relax_accuracy`, the relaxation stops.

Default: `relax_accuracy 0.01`

Don't choose it to be less then 0.001, since this is generally close to the accuracy with which the forces and stress is computed.

To avoid runaway jobs, the variable `relax_max_iter` determines the maximum number of relaxation steps to be performed. The statement

`relax_max_iter 10`

instructs the code to do at most 10 relaxation steps.

Default: `relax_max_iter 0`

2.2.2 Basic molecular dynamics variables

```
MD_time_step  default = 3.0        Time step for MD in femtoseconds

MD_temp       default = 500.0       Temperature for MD run in Kelvin

MD_max_iter   default = 100         Number of time steps in MD run
```

2.3 Plane wave code parameters

There are several parameters for the plane wave code which you will need to set correctly for each calculation.

2.3.1 Plane wave job

Whenever the plane wave code is called for a given structure, it can perform several jobs. For a simple relaxation, all you will need to do is an scf (self-consistent field) loop:

```
begin pw_jobs
pw_job scf
end pw_jobs
```

Later, you may want to add extra pw_jobs to do things like compute bandstructures and potential plots, at which time you should read Section 5.1.1.

2.3.2 Cutoff Energies

The following two variables for the energy cutoffs are mandatory, and have no default value. All energies are in Rydbergs.

```
energy_cutoff             Cutoff for the Hamiltonian matrix
submatrix_energy_cutoff   Cutoff for the submatrix to be used for the
                          generation of the starting guesses.
```

Example:

```
energy_cutoff           70
submatrix_energy_cutoff 10
```

Your `energy_cutoff` will depend on your pseudopotential, and generally you can simply use a `submatrix_energy_cutoff` of 5 to 10 Ry with no problem.

Advice: take the submatrix cutoff about 1/5 of the energy_cutoff, but usually not larger than about 10 Rydbergs. The generation of the starting guess can become very time consuming if the submatrix is too large. Also, the memory requirements increase rapidly with submatrix_energy_cutoff, because the full submatrix is kept in memory. The amount of time spent should be equal to the time of about 3 CG iterations on the wavefunctions.

Very small values can cause problems. In order to prevent high symmetry local minima, one can add randomization to Fourier coeffficents larger than the submatrix cutoff energy by

```
randomize_diag_start_guess 1.0  Default: 0
```

2.3.3 K-point grid and gaussian smearing

For each crystal you will need to specify what k-point grid you want to use to sample the Brillouin zone.

Example:

```
k_grid  4 4 4
k_grid_shift 0.5 0.5 0.5
```

This specifies a Monkhorst-Pack grid of 4x4x4 points shifted off Γ by $(\frac{1}{8}, \frac{1}{8}, \frac{1}{8})$ in reciprocal lattice vector units. This example is probably sufficient for most semiconductors with a smallish unit cell. If you have a large unit cell, you need fewer k-points, and if you have a metal, you will need a lot more.

If you are calculating a metal, you should use gaussian smearing. This smears out the fermi level, so that more of your k-points will be effectively on the fermi surface. The `gaussian_smearing` is specified in eV. If you are calculating an insulator, you should set `gaussian_smearing` to be some small value.

Example:

```
gaussian_smearing 0.05
```

2.3.4 Number of bands

You need to specify the number of bands to be computed. Of course, this should be greater than or equal to the number of actually filled bands.

Example:

```
number_bands 4
```

For insulators one should be equal to the number of filled bands. For metals, this should be more and is dependent on the system and the smearing method used. For the Grassmann_metal minimization method, 4 bands are added to this value.

2.3.5 Pseudopotential (PP) format

Starting with Paratec 5.1.1 has three different formats for PP input. For all formats the file must reside in chemical_symbol_POT.DAT files. The format is chosen by the line

```
pp_format          1 - original format in Fourier space
                   2 - L-W Wang's version of Martins code
                   3 - FHI98PP code
```

Format 1 requires a binary file format (created with the kbascbin executable created by "make tools"). Formats 2 and 3 are in asci format. Format 1 is incompatible with the pulay_tf mixing option. Formats 2 and 3 are in real space and include the atomic pseudowavefunctions. Format 3 is the suggested format for new users. Information and source can be found at http://www.fhi-berlin.mpg.de/th/fhi98md/fhi98PP.

For format 3, this additional keyword information must be given:

```
begin pseudopotential
pp_data  (l of local PP) + 1, occupation of s shell, p shell, d shell
                  for every atom type in the order they appear
end pseudopotential
```

e.g. For GaAs (Ga appearing as the first newtype in the coordinates and the s potential as local),

```
begin pseudopotential
pp_data   1  2.0 1.0 0.0
pp_data   1  2.0 3.0 0.0
end pseudopotential
```

2.4 Congratulations!

You now know (unless I left something out) the everything you need to perform simple calculations with paratec.

You may want to read the sections on diagonalization methods 5.3.1 and mixing methods 5.3.4, to learn how you may be able to make your calculations run faster.

Chapter 3

Advanced relaxation parameters

3.1 Specifying the crystal structure

3.1.1 Inversion and symmetry

If there is a center of inversion, the code automatically tries to shift the atomic coordinates such that the crystal has inversion symmetry. If you want to inhibit that, put a line saying:

```
no restore_inversion
```

Finally, you can specify whether you want the symmetry-operations to be generated by the code, or if they should be read from the file `SYMMETRYOPS`. If `number_symmetry_ops` is followed by `-1`, then the symmetry operations are determined by the code itself. If `number_symmetry_ops` is > 0, the matrices for the symmetry operation are read in from the file. The file format is the same as the printout in e.g. OUT.

```
Default: {\tt number\_symmetry\_ops -1}
```

If `number_symmetry_ops` is followed by `-2`, then the symmetry operations are determined by the code itself, but nonzero fractional translations are ignored. This could be useful for supercells.

3.1.2 Specifying an initial strain

You may optionally specify an initial deformation tensor using the format...

```
begin deformation
0.00 0.00 0.02
0.00 0.00 0.00
```

```
0.00 0.00 0.00
end deformation
```

This is the deformation tensor applied to your structure. It is defined by the following equation:

$$\mathbf{a} = (1 + \mathbf{D}) \cdot \mathbf{a_o}$$

where $\mathbf{a}$ are the deformed lattice vectors, and $\mathbf{a_o}$ are the lattice vectors specified in your input file. This can be an easy way to apply a shear strain to a cubic crystal, for instance.

There are times when it is useful to ensure that a given direction is held fixed during the relaxation process. This does not involve a constraint on the relaxation, as there are three rotational degrees of freedom. One can also arrange to maintain fixed a plane which contains the fixed direction.

To keep the $\mathbf{z}$ direction fixed, and also keep the zx plane from rotating, you could specify...

```
fixed_vector                0 0 1
other_vector_in_fixed_plane 1 0 0
```

3.2 Relaxing a structure

For the impact of the `job` variable on the relaxation process, see section 2.2.

3.2.1 Controlling the relaxation process

If `job relax` or `job relax_from_checkpoint` or `job relax_recycle_H` is chosen, the variable `relax_what` allows different ways to relax a structure:

`relax_what force`

or

`relax_what stress`

or

`relax_what force_and_stress`

Default: `relax_what force_and_stress`

Another variable allows to manipulate the way the relaxation is done:

Set

`relax_how gradient`

to simply follow along the forces and stress, without updating the inverse of the Hessian matrix.

Default: update the Hessian

With the variable `relax_how` the line-minimization can be altered also:

```
relax_how slow        # do 3 steps per line minimization always
relax_how normal      # do up to 3 steps, based on gradient prediction
relax_how fast        # do up to 2 steps, only if energy decreases
```

Default: `relax_how normal`

Note: This can be combined with the `gradient` option with a statement like `relax_how normal_gradient`

The parameter `lambda_limit` allows you to limit the size of relaxation steps to a certain number. Normally, lambda should be on the order of 1. Sometimes, especially in the beginning of the iteration, and if the H matrix is initialized to small, lambda can become to big, and the algorithm bombs out. It is then advisable to limit lambda with a line like that:

`lambda_limit 4.0`

Default: `lambda_limit 1e6` (no limit)

The statement `starting_distortion` followed by nine floating point numbers allows to set an initial distortion of the lattice. This feature is only for debugging, and not recommended for use.

With version 5.1.6, there are two relaxation algorithms with the addition of a limited memory BFGS algorithm. The benefit is more in stability and in general better efficiency (only based on a small test suite of systems) than for memory. The input keyword is

```
relax_method   default=1   original BFGS quasi-netwon algorithm
                  2        limited memory BFGS
```

3.2.2 Pressure

An external pressure (units are GPa) can be applied with:

`relax_pressure 10.0`

If an entry like

`relax_pressure adjust`

is found, the pressure is chosen from the first iteration, taking it to be 1/3 of the trace of the stress.

Default: `relax_pressure 0.0`

3.2.3 Constraining the relaxation

Paratec offers the possibility to fix atoms to their input coordinates, meaning they will not move during the relaxation process. This is simply done by zeroing out the Hellman-Feynman forces. To fix atoms to their original positions, use the `stay_put` structure. The following example fixes the positions of atoms number 1, 3, and 10.

```
begin stay_put
stay_put 1
stay_put 3
stay_put 10
end stay_put
```

Default: all atoms are relaxed.

As of paratec 4.5, paratec also offers the option of fixing certain strain coordinates while allowing the others to change during the relaxation. This is done by using the keyword `stay_put_lattice` within the `stay_put` structure. The following example fixes the ϵ_{xx} and ϵ_{yy} strains, while allowing the z axis to expand or contract.

```
begin stay_put
stay_put_lattice xx
stay_put_lattice yy
end stay_put
```

Default: all strain coordinates are relaxed.

NOTE: The syntax of the stay_put structure has changed in version 5.0. The `end` at the end is now an `end stay_put`.

3.2.4 Initialization of the inverse of the Hessian matrix

The Broyden scheme used for the relaxation of the forces and the stress updates an inverse of the Hessian matrix, which essentially describes the enthalpy parabola around the minimum. There are some open parameters which determine the values to which the matrix is initialized. They basically determine the length of the trial steps during the first few iterations:

```
relax_eps_factor   0.1     initialization perfactor for the stress part
of H
relax_coord_factor 0.1     initialization perfactor for the force part
of H
```

Try with the values above for the first run. Adjust later such that the first few steps give a lambda of about 0.5 to 1.

Default:

```
relax_coord_factor 1.0
relax_eps_factor   assuming the bulk modulus is about 1 Mbar
```

Alternatively, you can specify the physical quantities to which the above two variables correspond. This is the recommended way:

```
estimated_bulk_modulus  1.0                Bulk modulus in MBar
estimated_optical_phonon_frequency 12.0    TO phonon frequency f at
                                           Gamma in THz)
```

Diamond would have a bulk modulus of about 4.4 MBar and around 35 THz for the optical phonon frequency.

For molecular solids, the option

```
decouple_coordinates yes
```

causes a different initialization of the Hessian matrix, which decouples the coordinates of the molecule from the lattice vectors to first order. This means that if the lattice vectors change 10 percent, the cartesian coordinates will change by about 1 percent. IF THIS OPTION IS USED, THE SYMMETRY OF THE CRYSTAL IS NOT PRESERVED!.

Default:

```
decouple_coordinates no
```

Chapter 4

Advanced molecular dynamics parameters

4.1 ensemble type

Users can choose between the micro-canonical (constant electronic energy) and the canonical (constant temperature) ensembles. The micro-canonical ensemble is integrated with the velocity Verlet algorithm. The canonical ensemble via Nose'-Hoover dynamics is integrated with a generalized leap frog method (J. Comp. Phys. 151 p.114 (1999).

```
ensemble_type default=2          1 - constant energy
                                 2 - constant temperature via Nose'-Hoover dynamics
```

4.2 extrapolation methods - for relaxation also

In order to minimize the number of updates to the electronic wavefunctions in order to reach convergence, an extrapolation of the wavefunctions and/or potential is done to obtain better initial guess for each atomic configuration. One can choose between a first order, 2nd order (PRB 45 p1538 (1992)), or an alternating sequence of these two methods.

```
 < 0          No extrapolation.
   0          1st order extrapolation of wavefunctions and
              potential
   1          2nd order extrapolation of wavefunctions and
              potential
   2          alternating extrapolation of wavefunctions and
              potential
  10          1st order extrapolation of wavefunctions and
              potential is created from extrapolated
```

```
                wavefunctions - suggested for direct
                minimization methods
   11           2nd order extrapolation of wavefunctions and
                potential same as 10
   12           alternating extrapolation of wavefunctions and
                potential same as 10

extrapolation_method  (default 2 -MD) (default 1 - for relaxation)
```

4.3 Mass of extended system for canonical ensemble

For the Nose'-Hoover dynamics, a mass is specified for the extended system that acts as a heat bath in order to regulate the temperature of the electronic and ionic system. A very small or large value will result in non-canonical behavior. The mass is entered in units of Rydbergs. An overall energy of the electrons, ions, and the extended system is conserved.

```
MD_Q_mass default(= total thermal energy / 10)   value in Rydbergs
```

Chapter 5

Advanced plane wave code parameters

5.1 Additonal information on basic parameters

5.1.1 Plane wave code jobs

Whenever the plane wave code is called for a given structure, it can perform several jobs, depending on the job list:

```
begin pw_jobs
pw_job   first_job
pw_job   second_job
...
pw_job   last_job
end pw_jobs
```

If you do a structural relaxation, the list of jobs will be executed for every configuration of the relaxation, not just for the final one. Allowed jobs are:

- `scf` perform a self consistent field calculation, and compute force/stress.
- `nmr_shift` compute the diamagnetic susceptibility of insulators for the nmr shift, see section 6.1.
- `nmr_plot` plot the diamagnetic susceptibility computed in a previous calculation, and read from the file `CHI`. It will also plot the induced current, with the B-field aligned along the "ket" direction.
- `band_structure` compute and plot the band structure.
- `pot_plot` If also the output flag `potplot`, is set, this job plots the local part of the effective KS potential as a line plot, and produces a file for the IBM data explorer also. The G=0 component of the potential is set

to 0. The printed out eigenvalues and the fermi level include VXC(G=0), so you must subtract this offset to find the alignment with respect to the potential. `pot_plot` can also be used to plot the charge density, but not in conjunction with the energy window features. To plot the total charge, set the output flag `cdplot`. Also, the electric field along a line and at the atomic positions is plotted if the output flag `efield` is set.

5.1.2 Kpoint grid

Specify

`number_kpoints 0`

to have the kpoints generated, or use a positive integer number, and the corresponding number of kpoints will be read from the file `KPOINTS`, which must be in the current working directory.

Default: `number_kpoints 0`

As a special hack, you can specify the number of kpoints to be negative. This will also generate kpoints, but will NOT REDUCE THEM:

Default: `number_kpoints -1`

Implemented is a k-point generation scheme a la Monkhorst-Pack. The parameters are `k_grid`, followed by 3 integers, which give the number of grid points along the 3 reciprocal lattice vectors for the unreduced k-points. An entry called `k_grid_shift` followed by three real numbers allows to shift the k-grid.

Example:

```
k_grid 4 4 6
k_grid_shift 0.5 0.5 0.5
```

Shifts the 4x4x6 grid by 1/2,1/2,1/2 (of one grid cell)

Default:

```
k_grid 1 1 1
k_grid_shift 0 0 0
```

5.1.3 Number of bands

The entry `number_bands` gives the number of bands that are computed. It should always be slightly more than the minimum number of bands required, unless a wide-gap insulator is studied. If the number is more than the minimum, the Grassmann_metal diagonalization is suggested. For systems with a large number of states at the Fermi surface (such as metallic surfaces), as many as 50% more than the minimum number of bands may be optimum. The computational effort grows linearly with the number of bands, so do not pick the number unreasonably large either. Note: when the optimize insulator option (a direct minimization algorithm) is used the number of bands must be equal to the number of filled bands. When this option is not used the number of bands is padded by 4, and then slowly decreased to the value entered. In the Grassmann_metal diagonalization algorithm an additional 4 bands are added.

Also in the Grassmann_metal method, for each **k**-point the number of bands is decreased such that no more than 15 bands more than any of those with occupations greater than 1d-12 are used for said **k**-point.

Example:
`number_bands 12`
Default: There is no default

5.2 New functional parameters

5.2.1 Exchange correlation functional

Currently three different exchange-correlation functionals are implemented: LDA (Ceperley-Alder) and GGA (Perdew-Wang 91, Perdew Burke Ernzerhof). Spin polarized calculation is supported in all cases. You can set the exchange-correlation functional with:
`exchange_correlation name_of_functional`.

By default:
`exchange_correlation ceperley_alder`

The GGA can be switched on with:
`exchange_correlation perdew_wang_91` or

`exchange_correlation perdew_burke_ernzerhof`

The shortened forms `pw91` and `pbe` are also recognised options to `exchange_correlation`.

5.2.2 spin polarized calculations

Note that for spin-polarized calculation you should put the following line in "input" file
`number_of_spins 2`

For non-spin-polarized calculation you should put
`number_of_spins 1` (This is default)

For a magnetic insulator one can also fix the number of up and down spins with
`number_of_alpha` (number of spin up electrons)

`number_of_beta` (number of spin down electrons)

These commands must be used in conjunction with the keyword
`occupy_levels from_input` .

If the desired number of bands to be used for each spin differs from these input values one must use the commands

`number_of_bands_alpha`

`number_of_bands_beta.`

The number of bands for each band should be chosen judiciously such that the largest gap exists between the eigenvalue for the highest band used and the subsequent eigenvalue of the next band, which is not used.

5.2.3 Occupation numbers a.k.a. smearing

After the energy levels at the various kpoints have been computed, they are occupied with electrons according to their energy. Paratec allows several methods for occupying the orbitals by

```
smearing_method  default=1
1   Gaussian -  C-FU AND K-M HO, PHYS. REV. B 28, 5480 (1983).
2   Fermi-Dirac -  D.N.MERMIN, PHYS. REV 137, A1441 (1965)
3   Hermite-delta - METHFESSEL AND PAXTON, PHYS. REV.B 40, 3616 (1989)
4   Gaussian spline - J.M. Holender et.al., Phys. Rev. B \textbf{52}, 967 (1995).
5   COLD SMEARING I - N. Marzari
6   COLD SMEARING II - N. marzari
```

You must specify the width of the exponential-type functions in electron volts that are used to calculate the occupations of the energy levels. Example:

```
smearing_energy 0.1   Default = 0.05
```

For insulators, choose a very small number, say 0.001 eV. For metals, ideally the smearing should give an entropic energy term of about 1meV per atom. gaussian_smearing is also acceptable for backward compatibility of input files for version 5.1.5 and earlier.

With the variable `occupy_levels` you can control the way the occupation numbers are computed.

```
occupy_levels   normal        occupy up to fermi level, which is
                              common to all kpoints or

occupy_levels   fixed         fill minimum number of bands

occupy_levels   from_input    fill by prescribed input of
                              number_of_alpha and number_of_beta

Default: occupy_levels normal
```

5.2.4 Options for improved stress

The stress generally converges slower than the total energy, because the stress is normally evaluated assuming constant number of plane waves, which is a bad approximation.

The code has two options to improve that. The first trick is due to Michel Cote and goes as follows. After the selfconsistent calculation has been performed at normal cutoff, a second calculation with higher cutoff is done, but WITHOUT allowing the charge density to relax. This turns out to be still a very good approximation, but the second calculation is somewhat expensive. The cutoff energy (in Rydberg) for the second run is set with the keyword

`polished_energy_cutoff`.

Example:

`polished_energy_cutoff 50`

Default: switched off

The second trick is to modify the kinetic energy according to Bernasconi et al. You can set the height A, position E_0, and width σ (all in Rydberg) of the step function that gets added:

$$|\vec{k}+\vec{G}|^2 \rightarrow |\vec{k}+\vec{G}|^2 + A(1+\mathrm{erf}((|\vec{k}+\vec{G}|^2 - E_0)/\sigma))$$

with the keyword `modify_kinetic_energy` A E_0 σ.

Example:

`modify_kinetic_energy 20.0 10.0 1.0`.

If you just do

`modify_kinetic_energy on`

a reasonable setting is assumed: $A = 2E_{cut}$, $E_0 = E_{cut}$, $\sigma = E_{cut}/10$.

Default: `modify_kinetic_energy off`

5.2.5 simple option for creating super cells

This option creates a large unit cell that is a multiple number of the primitive unit cell given in the input file. The keyword

```
super_cell 2 2 2
```

creates a super cell with lattice vectors twice the primitive lattice vector length and with 8 times the number of atoms. Note: the user must still input the proper number of bands.

```
Default:  super_cell 1 1 1  ie. no super cell
```

For a super cell that extends the size of the unit cell without adding atoms (e.g. adds vacuum layers for a surface calculation) use

```
super_cell_vac 2 0 0          Default: 0 0 0  - no vacuum
```

This creates a unit cell with vacuum layers of twice the length of the first primitive lattice vector.

5.2.6 External potential

Paratec has an option to apply an external (electric) potential of the form:

$$V = V_0 \sin(\vec{G} \cdot \vec{r}). \tag{5.1}$$

This is done with an entry like:
`electric_potential_kvec 1 0 0 1`
The last number switches the field on (=1) or off (=0), the first 3 INTEGER numbers specify the wave number in reciprocal lattice coordinates.
The amplitude (in Rydbergs) is entered as:
`electric_potential_strength 0.3`

5.2.7 Real space nonlocal pseudopotential projectors

For very large systems (at least 100 atoms if not more), it is advantageous to implement the action of the nonlocal pseudopotential projectors in real space. In Fourier space the calculation scales as $O(N^3)$. Since the projectors are local in real space, this method scales as $O(N^2)$. The Fourier method has a smaller prefactor so there is a crossover point for when the real space implentation is faster. The projectors are made more local by use of mask function (developed by LW Wang). The keyword is

```
NLPP_rspace .true.    Default=.false.
```

One also needs to specify the cutoff radius for the projectors

```
NLPP_rcut  4.2  4.7 adding radii for each new type of atom
Default=4.7 for all
```

For the force and stress to be calculated one uses

```
NLPP_rspace_force .true. Default=.false.
```

The diagonal elements of the stress field are quite long ranged requiring a radius of 6.7. For now it is suggested to NOT do the force and stress in real space.

5.2.8 Number of bands for FFT - Parallelization

On parallel machines which use shared memory processors (SMP) as nodes that are combined to make a large distributed machine, long latency in the communications of the FFT can cause very poor scaling to large number of processors. PC clusters also also in this category.
As a solution, one has the control to do the FFT of multiple bands at a time so latency effects are minimized since larger packets of data are being passed. One must be careful though. If too many bands are used then the data can no longer fit in the cache and the result will be slower. Some examples of optimal values can be found in the analysis section on PARATEC's home page.

```
number_bands_fft 4 Default=1
```

5.3 Methods for minimizing the energy

5.3.1 Diagonalization Methods

The iterative method for the diagonalization to obtain the electronic wavefunctions is chosen by

```
diagonalization_method    Mauri, Grassmann, or Grassmann_metal
```

The default is Grassmann_metal. For insulators and semiconductors at low temperature, Mauri (the original algorithm in versions of Paratec) and Grassmann gives similar results. Comparisons can be found at http://www.nersc.gov/projects/paratec/METHODS/comp_
Current tests have show that with a max_iter_diag of 5 and less that Grassmann tends to perform better.

For metallic surfaces and long skinny cells, both of these algorithms suffer problems. The Grassmann_metal algorithm was developed to handle these systems. It uses the occupations of the electronic wavefunctions to facilitate the diagonalization. A write-up on this method can be found at http://www.nersc.gov/projects/paratec/METHODS/scf_me
Note that for the fist SC step, the Grassmann method is used as the occupations are not available.

For systems with a gap, one can also choose a direct minimization via the **optimize insulator** option. This is just as fast or faster for systems with a large gap and few k-points. In the direct minimization method, the wavefunction is minimized with a fixed hamiltonian until the `(norm of the gradient)/(# of bands)` is less than 0.1. After this point is reached, the hamiltonian is calculated directly from the updated wavefunction. This process ensures a good initial guess before the potential is updated

For metals, one can choose **optimize metal**, but this was found to be always inferior to the self-consistent method, which uses potential mixing.

5.3.2 Diagonalization accuracy

The accuracy of the diagonalization scheme is determined by the keyword `accuracy_diag`. The diagonalization stops when the gradient (i.e. the residual) is below `accuracy_diag`. The accuracy of the eigenvalues is normally at least as good, often an order of magnitude better.

Example:

```
accuracy_diag  1e-10  Default: accuracy_diag 1e-12
```

Advice: The energy is normally converged to 1e-6 only, so an accuracy of 1e-7 is often sufficient. However, to get the forces accurate to n digits, one should require `accuracy_diag` to be less than 1e-(2n). For example if you set `relax_accuracy` to 1e-4, you must set `accuracy_diag` to 1e-9 or smaller.

5.3.3 Number of CG iterations

The method for determining the number of iterations for the electronic minimization by one of the conjugate gradient methods is hierarchical. The number of iterations is not a fixed amount but varies depending on the convergence at a particular **k**-point for a given SC step. The conjugate gradient minimization process is always stopped if a absolute convergence is achieved as set by `accuracy_diag`.

The variable `min_iter_diag` actually determines the minimum number of iterations on the wavefunctions per SC step, if no convergence according to accuracy_diag is achieved. Example:

```
min_iter_diag 5  Default: 3
```

The input used to be `max_iter_diag`. This keyword serves the same function as is retained in order to work with old inputs.

Generally for metallic systems 3 should be sufficient. If there are problems then the number_of_bands can be increased. For insulators 3 is also suitable with the minimum number of bands. For difficult semiconductor systems (such as surfaces) with a small gap and a minimum number of bands, 5 might be better. In this case one might use more bands with the Grassmann_metal algorithm.

If the norm of gradient does not decrease to 30% of its original value for a SC cycle then additional iterations are done. This is set by the keyword

```
iter_diag_add 5  Default: 5
```

for the Grassmann method and

```
iter_diag_add_metal 5  Default: 3
```

for the Grassmann_metal method.

If the norm of the gradient divided by the number of bands is greater than 1d-3 for the Grassmann method, then this number of additional CG iterations are done. This typically happens at the beginning of a calculation. Note that for the Grassmann_metal method, the Grassmann algorithm is used for the first SC step. For a difficult layered magnetic surface, it is beneficial to set iter_diag_add to between 20-40, so the wavefunctions are relatively accurate after the first SC step.

For a band structure calculation, one only does one scf loop and thus one wants fully converged wavefunctions at the end of the first scf loop. It is convenient to have a separate variable for the number of CG iterations for band structure calculation.

```
max_iter_diag_band Default: 137
```

5.3.4 Self-consistent mixing methods

The potential mixing method that obtains a new input potential for the next self-consistent step is chosen by

```
mix_method              broyden, pulay_kerker, or pulay_tf
Default = pulay_kerker
```

If broyden, the following additional parameter is used.

```
mixing_energy_cutoff      Sets the broyden mixing energy cutoff.
Default = 5 Ryd
```

If pulay_tf, the following additional parameters are used

```
PTF_num_init_PK      gives the number of pulay_kerker mixing
                     steps before the pulay_tf mixing is used.
Default = 2          best performance for Al 20 layer surface

PTF_max_iter_cg      gives the number of conjugate gradient
                     iterations for solving the TFW equations
default = 40
```

The Broyden method should probably never be used unless someone has a sentimental attachment. For any size relatively homogeneous system, pulay_kerker should be used. For large inhomogeneous systems, pulay_tf should be used; especially for systems with a regions of vacuum such as sufaces, nanotubes, and molecules. Additional information can be obtained at http://www.nersc.gov/projects/paratec/analysis.htm.

Broyden mixing

The `mixing_parameter` variable determines the initialization of the inverse of the Hessian matrix. A small mixing parameter will lead to smaller movements of the charge density during the first few iterationss. Later, when the inverse of the Hessian matrix has been constructed, the mixing parameter does not change the algorithm much.

```
mixing_parameter 0.33       initialize H to 0.33 of standard size
Default:    {\tt  mixing\_parameter 0.77}
```

For smaller wave vectors, a Broyden type mixing is used, for larger wave numbers a linear mixing is employed. The energy cutoff for this Broyden mixing may be set with the parameter:

```
mixing_energy_cutoff   5.0   Sets Broyden mixing cutoff to 5.0Ry
Default:    5 Ry
```

Often a small fraction of the total cutoff is sufficient, say 5 Ry.

Linear mixing

In some cases, Broyden mixing is too aggressive. In this case, you can switch to linear mixing with a line

`linear_mixing a1 a2 a3`

which mixes the potentials according to the formula:

$$a_{mix} = a_1 + a_2 e^{-a_3 |\vec{G}|^2} \tag{5.2}$$

$$v_{new}(\vec{G}) = a_{mix} v_{in}(\vec{G}) + (1 - a_{mix}) v_{out}(\vec{G}) \tag{5.3}$$

That way, the higher wavenumbers are mixed stronger, while the tricky small G components are mixed less.

`Defaults: {\tt linear\_mixing 0 0.6 0.05}`

To switch linear mixing on, a1 has to be > 0. This switches off Broyden mixing. `a1=0.2` seems to be a good value to start with.

5.3.5 Self-consistent iteration accuracy

The accuracy with which the self-consistent potential is computed can be set with the variable `potential_convergence_criterion` or with `energy_convergence_criterion`, or both..

`potential_convergence_criterion 1e-4 Default: 1e-6`

Requires that the largest difference between the current and the previous potential (in G-space) be 1e-4. A value of 1e-4 is generally sufficient for energy convergence and relaxation of atomic positions. For optimization of strain on the unit cell or phonon calculations, smaller values should be used. Values smaller than 1e-6 probably are never needed.

`energy_convergence_criterion 1e-8 Default: 1e-9`

If the change in energy from the last self-consistent cycle is less than the prescribed value for two consecutive self-consistent cycles then convergence is achieved. For simple semiconductor systems this corresponds approximately with the default for potential convergence.

If no convergence is achieved within `max_iter_scfloop`, the self-consistent iteration terminates, and goes on to compute the stress and forces.

`max_iter_scfloop 50 Default: 40`

5.3.6 Screening (inital guess for the charge density)

For the first iteration of each plane wave programm call, the ionic pseudo cores have to be screened with an initial guess for the charge density. With

`screening_type atomic`

the initial screening charge density is calculated from the valence charge density of the atoms.

Setting `screening_type previous` screens with the charge density from the file `CD` if that is available, or with the atomic charge otherwise. This means that the charge density from the previous call to the plane wave code is reused. The mixed potential itself is stored in the file `VMIX` and if available, may be re-used by setting `screening_type vmix`.

Default: `screening_type previous`

Chapter 6

Getting extra outputs and other interesting stuff

6.1 NMR shift

If a `pw_job nmr_shift` entry is encountered, paratec computes the local diamagnetic susceptibility. The variable

`nmr_q 0.001`

sets the $\vec{q}$ (in cartesian coordinates), which determines the periodicity of the applied external field.

Default: `nmr_q 0.01`

Set the output flag `nmrshift` to get a plot of the susceptibility (section 6.9).

If a line

`checkpoint nmr`

is entered, the code will checkpoint the NMR shift calculation after each kpoint. You can recover from a crash by giving an `input_flags` of `chkptchi`.

IMPORTANT: Right now, the NMR shift can only be computed for INSULATORS. You have to set `number_bands` to contain only the VALENCE BANDS, or else the result will be wrong.

6.2 Band structure computation

An entry `pw_job band_structure` computes the band structure. You should set `screening_type` to `previous`, and provide a converged CD file for the starting density. THERE WILL BE NO SELFCONSISTENT ITERATION, i.e. the bandstructure is plotted corresponding to the CD file, or to the atomic valence charge if no intact CD file is found. Furthermore, YOU HAVE TO USE THE FERMILEVEL OF A PREVIOUS SCF RUN. You do so by giving a line like:

`fermi_level fermi_energy_in_Ryd`

Default: `fermi_level 0`

If you want to use a different number of bands from that used in an earlier SCF loop, you may specify this number using:

```
number_alt_bands <number of bands for the bandstructure>
```

You also have to determine the lines in the BZ along which you wish to plot the bandstructure. This is done by specifying a sequence of lines in the following way:

```
begin bandstructure
label left_middle_right
kpoint startx  starty  startz   endx  endy  endz   number_of_bins
label left_middle_right
kpoint startx  starty  startz   endx  endy  endz   number_of_bins
..
label left_middle_right
kpoint startx  starty  startz   endx  endy  endz   number_of_bins
end bandstructure
```

NOTE: The syntax of this structure has changed in version 5.0. Previously this ended with just `end`, and this has changed, so that now you need to end with `end bandstructure`.

All coordinates are given with respect to the reciprocal lattice vectors. The labels are mandatory; they have to be triples, separated by an underscore. Notice that xmgr allows greek characters using escape sequences (see example below).

The kpoints will be generated into the file `KPOINTS`. The output of the calculation can be found in the file `BANDSTRUC`. Energies are in eV, and are plotted against an x-coordinate that corresponds to the physical distances in cartesian k-space, i.e. 1/a.u. The file can be displayed with the popular graphics program "xmgr".

Example:

```
begin bandstructure
label \8G\4_\8\D\4_X
kpoint 0.0 0.0 0.0         0.0 0.0 0.5            10
label _V_W
kpoint 0.0 0.0 0.5         0.0 0.5 0.5            10
end bandstructure
```

6.3 Density of states (DOS)

To compute the density of states, a self-consistent calculation has to be done first to get the wave functions. To generate a DOS, add to the `output_flags` either `dos` or `angdos`. This will produce a file "DOS" in a suitable format for xmgr (it is an ascii text file, and has comments in there). A tetrahedron interpolation

scheme is used, where each of the Monkhorst-Pack grid cells is divided into six tetrahedra. If you compute the angular momentum resolved DOS, you must also specify the radius of the muffin-tin spheres (see section

The density of states is output in units of states/eV/unit cell. It does not count both spin states. 2.1).

6.4 Momentum density

To compute the momentum density, first compute the selfconsistent charge density, and then do a band structure calculation. There, you specify those lines in the BZ along which you want to know the momentum density. To get the momentum density, you must add the flag `momdens` to the list of `output_flags`. Example:

```
begin pw_jobs
pw_job scf
pw_job band_structure
end pw_jobs
output_flags momdens
```

You also need to specify the Fermi level according to a previous scf run: `fermi_level fermi_energy_in_Ryd`

If you don't specify this correctly, you will get garbage!

6.5 GW Interface and Usage

GW-interfaced Version of Paratec with f-states was implemented by Eric Chang. Further development was done by Xavier Blase, Gian-Marco Riganese and others.

6.5.1 GW Interface

To calculate dielectric function, we need 2 wavefunctions from paratec. They are called CWFE and CWFEq. CWFE requires lots of conductance bands, but CWFEq only requires wavefunction of bands (valence bands + 2).

Procedure:
(1) Generate a converged charge density `CD`, save this `CD` to a file `CD.conv`. You must use the same charge density each time when you perform the following jobs.
(2) Compute CWFE. In the input file of paratec, use the options

```
number_bands 90 # this number means the number of conductance
bands used in output_flags gwr # the summation of dielectric
function calculation
```

If the system does not have inversion symmetry, you need to run the complex version:

```
number_bands 90 # this is a large number compared to the number of
valence bands output_flags gwc
```

After run paratec, rename the file `GWR` (or `GWC` for complex version) to `CWFE`. You may need to use fewer or more bands in the option `number_bands`.

If one wants to choose only selected wavefunctions for the GW calculation then the following procedure may be implemented. For selected bands around the Fermi energy one uses

```
num_gwout 4
```

in order to choose 4 bands on either side of the Fermi energy. If one wants a different energy than the Fermi energy to obtain wavefunctions near then one uses

```
gwout_mid_energy -0.45
```

If instead of defining a number of states in the above manner, one wants to obtain all wavefunctions within an energy window then

```
gwout_low_energy   -1.0
gwout_high_energy   2.0
```

is employed.

(3) Compute CWFEq. In the input file of paratec, use the options

```
number_bands 5         # this number is (number of valence bands +
2) output_flags gwr        # use gwc if you want to run complex
version of the GW code gwshift 0.005 0.005 0.005
```

You should use `number_bands` slightly more than the number of valence bands in the system. `gwshift` gives a wavefuction at slightly shifted grid. Rename the file `GWR` to `CWFEq` after paratec computation.

GWC or GWR → CWFE for the shifted grid (need many bands)
GWC or GWR → CWFEq for slightly shifted+q grid (need only valence bands,need to read k-points from a file)

(4) Compute `VXC` and `CD95`. `VXC` is the information of exchange-correlation functional; CD95 is the charge density. In the input file of paratec, use the option

```
output_flags gwscreening
```

(5) Compute `wfn0`. In the input file of paratec, use the option

```
k_shift 0 0 0
number_bands 90        # test the convergence of this number
output_flags gwr       # use gwc if you want to run complex version of the GW code
```

Rename `GWR` or `GWC` to `wfn0`.

(6) Now you are ready to use `CWFE` and `CWFEq` to compute eps0mat and epsmat. eps0mat, epsmat, wfn0, VXC, CD95 are required to do GW calculation.

6.5.2 Example: Si

```
If you want to run the code for Si, do the following:

(1) mkdir Si
(2) cd Si
(3) mkdir paratec; mkdir xi; mkdir sig
(4) if you type pwd you should have the subdirectories:

 paratec  sig      xi

paratec -> has the LDA/pseudopotential/planewave code
sig -> has self-energy program
xi -> has screening program

 (ignore subdirectory epsilon)

(2) in your paratec directory you should have:

the following input files:

input.cd
input.cwfe
input.cwfeq
input.wfn0

the pseudopotential:

Si_POT.DAT
```

```
and the executable:

paratec.mpi

--- to make the executable, paratec.mpi, do the following:

   cd ~
   mkdir source
   cp paratec.tar source/
   cd source
   tar -xvf paratec.tar (untar the tar file)
   cd paratec
   make paratec
   cd ~/Si/paratec
   ln -s ~/source/paratec/bin/paratec.mpi . ! paratec.mpi is the
                                              executable

----To convert the ascii pseudopotential to the binary form in which
it is read, do the following:

   cd ~/source/paratec ! same directory as above
   make tools
   cd ~/Si/paratec
   ln -s ~/source/paratec/bin/kbascbin .
   ./kbascbin Si_POT.ASC.DAT Si_POT.DAT


Now we are ready to run paratec for four times, one time for each
of the four input files:

input.cd ----> compute the ground state density, and screening.
               use output_flags gwscreening, and save CD.
input.cwfe -----> compute the wavefunctions on the
                  4x4x4 grid, shifted 0.5 0.5 0.5. In this
                  case we compute 5-10 times the number
                  of valence bands. Notice the option in the input:
                  output_flags gwr. This means, produce real
                  wavefunctions to be read by the program in directory
                  Xi. Also, make sure that you converge the wavefunctions
                  to a high accuracy.
                  By the way, for a system with inversion symmetry,
                 you need to say, output_flags gwc.
input.cwfeq -----> compute the wavefunction on the
                   4x4x4 grid, shifted 0.5 0.5 0.5 plus some small
                   q-vector, specified by the option in the input,
                   e.g. gw_shift 0.00 0.00 0.001. Compute only one more
```

```
                  than the number of valence bands in this case. Use
                  option output_flags gwr.
input.wfn0  ---> computes wavefunctions on unshifted grid, 5-10 number
                 of valence wavefunctions, to be read by



In summary, we have the following new features in paratec
which enable the interface between paratec and GW:

gw_shift 0.00 0.00 0.001
output_flags  gwr,gwc, or gwscreening

gwr -> produces real wavefunctions
gwc -> produces complex wavefunctions
gwscreening -> produces CD95 and VXC (for the sigma code)

So do the following:


cd ~/Si/paratec  ! go to the right directory

cp input.cd input  ! compute ground state charge density
mpprun -n 8 ./paratec.mpi ! run paratec
cp CD CD.save             ! save ground state charge density
mv CD95 ../sig  ! CD95 is the charge density read by self-energy program
mv VXC ../sig   ! VXC is the exchange corr

e. read by self-energy program

cp input.cwfe input ! computes wavefunctions on 4x4x4 0.5 0.5 0.5 k grid
                    ! computes about 5-10 times number of valence bands
cp CD.save CD
mpprun -n 8 ./paratec.mpi ! run paratec
mv GWR ../xi/CWFE         ! move wavefunctions (GWR or GWC) to xi

cp input.cwfeq input ! computes wavefunctions on 4x4x4 0.5 0.5 0.5 +
                                          small q  k grid for xi
                    ! only need one more than number of valence bands
cp CD.save CD
mpprun -n 8 ./paratec.mpi ! run paratec
mv GWR ../xi/CWFEq

cp input.wfn0 input   ! computes wavefunctions for self-energy
                      ! 4x4x4 unshifted
                      ! 5-10 times number of valence bands
```

```
cp CD.save CD
mpprun -n 8 ./paratec.mpi
mv GWR ../sig/wfn0


(2) calculate static screening matrix:

  q is an element of 4x4x4 unshifted grid (8 q-points)
  k,k' are elements of 4x4x4 shifted grid (10 k-points)

  q is {k-k'| k,k' shifted}

  RPA screening (xi polarization)

   xi(G,G';q) =  Sum(cvk) [ M(cvkqG) M*(cvkqG') /(E_ck - E_vk+q) ]

     M(cvkqG) = <ck|exp(-i(G+q).r)|vk+q>

  eps(G,G';q) = delta_GG' - [8pi/(q+G)^2 ]xi(G,G';q)

  if G=0  and q=0

  then we let

   eps(0,G';q) = delta_0G' - lim(q->0) [8pi/(q)^2] xi(0,G';q)


   program takes inverse of eps(G'G;q) for every q:

      -> epsmat (for q not equal to gamma)
      -> eps0mat (for q equal to gamma)


cd xi  ! you should have the following files:
           CWFE
           CWFEq
           xi0.inp input file
           xi0 executable

xi0.inp looks like

epsilon_cutoff          6.25  ! compute eps matrix up to G^2,G'^2 < 6.25 Ry)
number_bands          44  ! number of bands in summation (cond.+valence)
wavefunction_cutoff         12.0 ! same as in paratec
number_qpoints  8   ! number of q-points to compute eps(GG';q)
band_occupation 4*1 40*0 ! band occupation
```

```
begin qpoints
  0.00      0.00      0.001       1.0 1   ! q-point,divisor, 1 or 0
  0.00      0.00      0.25       1.0 0    !  copy from paratec
  0.00      0.00      0.50       1.0 0
  0.00      0.25      0.25       1.0 0
  0.00      0.25      0.50       1.0 0
  0.00      0.25      0.75       1.0 0
  0.00      0.50      0.50       1.0 0
  0.25      0.50      0.75       1.0 0
end

#Values for the 1st step
evs     0.0000
evdel   0.0000
ev0     0.0000
ecs     0.0000
ecdel   0.0000
ec0     0.0000

mpprun -n 8 ./xi0    ! run xi
mv eps0mat ../sig
mv epsmat ../sig

always check Xi(0,0;q->0) and eps^(-1)(0,0;q->0)
in xi0.log

1/eps^(-1)(0,0;q->0) should be the macroscopic epsilon,
                    local field effects
1/(1-8*pi*Xi(0,0;q->0) should also be (roughly) the macroscopic epsilon,
                     non local field effects

(3)  Look at Hybertsen/Louie, PRB 34,5390.

formulas:

  (34a) we calculate Sigma_(SEX)

          has two terms: delta_GG' + Omega^2/(DeltaE^2-wp^2)

              first term -> Gv (small v is bare coulomb interaction)
              second term -> is part of G(W-v)

  (34b) we calculate Sigma_(COH)

               other part of G(W-v)
```

We compute

<nk|Sigma(E)|nk>, Sigma = i G_0 W

cd sig

in your directory you should have:

```
eps0mat
epsmat
CD95
VXC
wfn0
sig.inp
sigma
```

looking at sig.inp:

```
screened_coulomb_cutoff   6.25  ! same as in xi0.inp
bare_coulomb_cutoff       12.0  ! never less than wavefunction_cutoff
wavefunction_cutoff       12.0  ! same as in paratec
number_bands               44   ! These two lines have the same meaning as
band_occupation      4*1  40*0  ! in xi0.inp
emergency_cutoff           6

#Define the bands for which we want to compute sigma
band_index_min   1       ! band_index_min and band_index_max fix the range
band_index_max   10      ! of calculation in terms of bands

#finite_difference_spacing   0.1

number_kpoints  1
begin kpoints
  0.0000  0.0000  0.0000   1.0   !  Gamma
end

number_offdiag  3      ! choose off-diagonal entries of Epsilon to look at

begin offdiag
    1  3              <1k|Sigma(E)|3k>
    1  2
    2  1
end

evs     0.0000               ! scissor shift (energy unit is eV)
```

```
evdel   0.0000
ev0     0.0000
ecs     0.0000
ecdel   0.0000
ec0     0.0000

mpprun -n 8 ./sigma    ! run Sigma


output looks like this (sig.log) :

           k=   0.000   0.000   0.000
  i    elda      e0       x      sx      ch     sig      v0    efso    enew (ev) spin=1
  1   3.884   3.884 -17.107  13.287  -7.228 -11.048 -10.429   3.265   3.467
  2  15.775  15.775 -12.700   9.382  -8.410 -11.727 -11.229  15.277  15.383
  3  15.775  15.775 -12.700   9.382  -8.410 -11.727 -11.229  15.277  15.383
  4  15.775  15.775 -12.700   9.382  -8.410 -11.727 -11.229  15.277  15.383
  5  18.356  18.356  -5.672   3.732  -7.827  -9.767 -10.051  18.640  18.579
  6  18.356  18.356  -5.672   3.732  -7.827  -9.767 -10.051  18.640  18.579
  7  18.356  18.356  -5.672   3.732  -7.827  -9.767 -10.051  18.640  18.579
  8  19.096  19.096  -5.770   4.011  -8.825 -10.584 -10.804  19.316  19.268

 1  1 --real-       -17.107  13.287  -7.228 -11.048 -10.429 del=  -0.620

 1  2 --real-        -0.075   0.122  -0.013   0.034  -0.000 del=   0.034

 2  1 --real-        -0.075   0.060   0.002  -0.012  -0.000 del=  -0.012



elda -> E_LDA
e0 -> E_LDA scissor shifted (linear interpolation)
x -> <nk|Gv|nk>, Fock operator
sx +ch -> <nk|G(W-v)|nk>
sig =  x + sx + ch
v0 -> <nk|v_xc|nk>
efs0 = e0 + sig - v0
enew = e0 + Z(sig - v0), where Z is close to 1


     Sigma depends on E

E^QP = E_0 + <nk|Sigma(E^QP) - vxc|nk>

 Assuming Sigma(E^QP) = Sigma(E_LDA) + Sigma'(E_LDA)(E^QP-E^LDA)
```

```
then we can show that

E^QP = E_0 + Z <nk|Sigma(E^LDA) - vxc|nk>

 where Z = 1/(1-Sigma'(E_LDA))

-----------------------------------------------------------------

We calculate

Sigma(E) = G({Psi_LDA},{E_0}) * W({Psi_LDA,{E_0'})

{E_0} is given by E_LDA with scissor shift specified in sig.inp
{E_0'} is given by E_LDA with scissor shift specified in xi0.inp

{E_0} --> some approximation to E^QP.


How scissor shift is defined:

eval --> eval+(evs+evdel*(eval-ev0))

econd --> econd+(ecs+ecdel*(econd-ec0))


Updating Xi : epsilon -> 1 + wp^2/E_av^2
doesn't matter.         E_av = average gap (which doesn't change much)
```

6.6 Visualization

By setting the appropriate `output_flags`, various quantities can be visualized. Paratec can produce line plots (filename ??.line.number) and files for the IBM Data Explorer (filename ??.dx). If you are using Khoros instead of the Data Explorer, set the `output_flags khoros`, which will produce a file with suffix .kh.

6.6.1 Tensor Data

Often, you want to plot only a certain component of a field or tensor field. You use a statement like:

```
tensor_bra 0 1 0
tensor_ket 0 0 1
```

to select the yz component (in cartesian coordinates) of a tensor. The "tensor" input is also used for specifying the alignment of the external B-field for plotting

purposes.
Default:

```
tensor_bra 1 0 0
tensor_ket 1 0 0
```

6.6.2 Line plots

The `line_plot` structure determines along which lines there should be plotted:
Example:

```
begin line_plot
line 0.0 0.0 0.0  1.0 1.0 1.0   100    plot data along (111) on 100 mesh
points
end line_plot
```

The start and end coordinates of `line_plot` are relative to the realspace lattice vectors.

6.6.3 Visualizing wave functions

Paratec can produce Data Explorer and Khoros files that visualize wavefunctions at particular kpoints, bands. The square of the wavefunctions is plotted in realspace. Wave function plotting is switched on by setting `pw_job scf` and `output_flags waveplot`.

The wave functions to be plotted are specified as pairs of the irreducible kpoint number and band number.

Example:

```
begin plot_wave_function
wavefunction 1 12                  irreducible kpoint 1, band 12
wavefunction 2 10                  irreducible kpoint 2, band 10
end plot_wave_function
```

6.6.4 Visualizing the charge density

To produce a simple three-dimensional charge density plot, one simply adds the flag `cdplot` to the list of output flags, and runs an scf calculation. If an energy window is specified with the energy window structure:

```
begin energy_window
window (start_energy in eV) (end_energy in eV)
end energy_window
```

then the charge density for the plot is generated only by using wave functions with eigenvalues within the energy window (specified in eV). To find suitable values for start_energy and end_energy, look at the eigenvalues printed out in previous runs. A gaussian smearing is applied, and the occupation numbers are

ignored. This feature only works if the wave functions have been computed, i.e. it does not work with the pot_plot plane wave job.

Warning: the energy window specification does not include an adjustment for the fermi level. Also, unoccupied bands will be included in the charge density if your energy window includes energies above the fermi level.

6.6.5 Plotting the potential

Both line plotting and three-dimensional plotting of the effective Kohn-Sham potential (without the nonlocal pseudopotential part) are available via the `pot_plot` plane-wave job. For more see section 5.1.1.

6.6.6 Ball and stick model

A ball-and-stick file `BALLS.dx` or `BALLS.kh` can be generated and viewed with the IBM Data Explorer or Khoros. This is done by adding the output flag `ballnstick` to the `output_flags` field (section 6.9). To use the data explorer, please refer to the README file in /usr/local/codes/paratec/dx. To use Khoros instead, set `output_flags khoros`.

6.7 Spin polarized calculations and magnetic field

The number of spins to be used can be set with the variable `number_of_spins`:

`number_of_spins 1`

(for non-spin polarized LDA)

or

`number_of_spins 2`

(for spin-polarized LSDA)

Default: `number_of_spins 1`

To get a spin-polarized starting configuration for the self-consistent charge, use the spin polarization factors where you input the crystal structure.

6.7.1 Applying a magnetic field

A magnetic field

$$H = H_0 \cos(\vec{G} \cdot \vec{r}) \tag{6.1}$$

can be applied to stabilize for example an antiferromagnetic phase. The magnetic field acts only on the spin, AND ALTERS THE ENERGY AND WAVEFUNCTIONS!! This is done with an entry like:

`magnetic_field_kvec 1 1 0 1`

The last number switches the field on (=1) or off (=0), the first 3 INTEGER numbers specify the wave number in reciprocal lattice coordinates.

The amplitude (normalized such that $\mu_b H_0 = 1$ Ryd) is entered as:

`magnetic_field_strength 0.001`

6.8 Reading additional input

The option `input_flags` enables you to read additional input from various files. The flags for the various inputs you want to read are concatenated by underscore.

`input_flags flag1_flag2_flag3` (sets 3 input flags)
Implemented flags:

- `wavefn` reads wave functions from a previous calculation as a starting guess for the iterative diagonalization. It is generally safer not to use this option, because the eigenvectors can be really bad if the structure is different, causing the diagonalization routine to bomb out.
- `chkptchi` reads the intermediate results from a previous nmr shift calculation. The data must be in a file CHKPTCHI.x, where x is the number of the processor. Make sure you don't read in garbage from the checkpoint file!

6.9 Getting additional output

For the direct energy minimization (optimize insulator), it is often useful to checkpoint the wave functions and charge density during the iteration. This can be done with

```
checkpoint_emin frequency
```

Example:
`checkpoint_emin 50` writes the wave functions to disk every 50 iterations. By default, no checkpointing is done.

Checkpointing can also be controlled for the SCF method. By default the charge density and potential at each SCF cycle are not written to disk as this is too costly. They are written at convergence or when the maximum number of cycles is reached. If one want them checkpointed then use

```
io_scf true
```

The option `output_flags` enables you to get additional numbers out of the code. The flags for the various outputs you want to obtain are concatenated by underscore.

`output_flags flag1_flag2_flag3` (sets 3 output flags)
Implemented flags:

- `diagperf` gives the performance of the diagonalization routine, resolved into the FFT part and the matrix-matrix multiplies.
- `vqmc` writes out a data file for the variational quantum monte carlo code.
- `nmrshift` plots the diamagnetic susceptibility χ in realspace. If requested, line plots are done also.

- `timing` prints timing information for various parts of the code. The time printed is system and user time together, per processor.
- `memusage` prints out memory consumption of large arrays in various subroutines. Not all subroutines print out their memory usage currently.
- `cdplot` produces charge density plots after a self-consistent field calculation.
- `neighbors` prints nearest distances in a.u. between the atoms in the unit cell.
- `ballnsticks` generates a file for DX or Khoros which contains the positions of the balls.
- `ballswrapped` same as ballnsticks, but all balls are within the first unit cell. This is useful for volumetric plots.
- `angles` prints angles between atoms, e.g. bond angles.
- `dos` computes the density of states with the tetrahedron method. The output goes to the file "DOS". The job variable has to be set to `pw_job scf`. You can inhibit the selfconsistent iterations my setting `max_iter_scfloop 1`.
- `angdos` Same as `dos`, but also computes the local and angular momentum resolved density of states for each atom. The output goes to the file "DOS". The job variable has to be set to `pw_job scf`. You can inhibit the selfconsistent iterations my setting `max_iter_scfloop 1`.
- `project` Outputs the angular projections of the wave functions. Currently this calls two separate codes, one of which supports f-states, while the other supports projection onto multiple atoms. One outputs to 'project.dat' while the other outputs to 'projections'.
- `waveplot` Writes the square modulus of particular wave functions into a .dx or .kh file.
- `potplot` Generates a graphical output file of the Kohn-Sham effective potential (without the nonlocal part) into a .dx or .kh file. Plot is done for both spins if applicable.
- `eigvals` Writes the eigenvalues, the weights of the kpoints, and (if computed) the angular momentum decomposition for each kpoint and band into a file "EIG"
- `efield` Compute the electric field due to the ionic potential and the charge density. In order for this to work, you must *also* specify a `pw_job pot_plot`. For more information, see Section 5.1.1.

- `wavefn` tells the code to write a file with the eigenvectors at the end of a self-consistent field calculation (`pw_job scf`).

  WARNING: paratec names the wavefn files using only one digit for the k points, so if you plot the same band at multiple k points (which have an index greater than 9), the wavefunctions will overwrite all but the first. This is a bug, but you can work around it, and I (David Roundy) don't feel like fixing it right now. Sorry.

- `momdens` write a file "MOMDENS" with the momentum density information. See section 6.4.

- `khoros` write all graphical output files for Khoros (.kh) rather than DX (.dx).

- `xyz` Output structural data in xyz format, for use in the xmol visualization program.

- `diagconv` Gives the keyword rho then the residual$\hat{2}$, iteration number, and the trace of the Hamiltonian.

- `fermisurf` Outputs the eigenvalues in k space, so you can plot the fermi surface.

- `noioscf` Prevents the output of CD, potneital, or wavefunctions

- `nofrcstr` Prvents code from calculating or outputing force and stress calculations.

Chapter 7

Funky extra flags

7.1 Optimization flags

`optimize` allows you to optimize for different resources, e.g. disk IO, memory usage, etc. The flags for the various optimizations you want to obtain are concatenated by underscores.

`optimize flag1_flag2_flag3` (sets 3 optimization flags)
Implemented flags:

- `insulator` does a direct minimization of the energy functional rather than the usual scfloop. Only works for insulators! Also, the number of bands must be ONLY THE NUMBER OF OCCUPIED BANDS.
- `metal` uses the ensemble DFT formulation of Marzari and Vanderbilt. It can be very slow and is not recommended.
- `memory` optimizes for memory in the calculation of NMR shifts, at a small expense in additional CPU time.
- `nostartguess` Recalculates the starting guesses in the NMR subroutine every time. Using this option will generally result in SUBSTANTIALLY SLOWER execution, but will reduce memory cost.
- `nocgsolveguess` switches OFF the better starting guesses for the conjugate gradient solver in the NMR subroutine. Using this option will generally result in SUBSTANTIALLY SLOWER execution, but will reduce memory cost.
- `nowfnreuse` when relaxing with a fixed lattice, DOES NOT reuse the wave functions from the previous relaxation step.
- `noprecond` do not precondition in the direct energy minimization. Use this option only if you have convergence problems, since it will generally result in 2-3 times slower convergence.

- `noadjustshift` keeps the number of bands fixed during the calculation and should be used if you have trouble with complex charge densities.

The computational cost is generally dominated by the FFTs, and on parallel computers, a significant fraction of the FFT time is involved in communication between processors. Poor scaling results when latency dominates these communications, but the latency may be reduced (with an associated memory cost on some machines) by setting
`number_bands_fft` n
where n is the number of bands to FFT simultaneously (this option is disabled when using the SHMEM library).

Chapter 8

Helpful tools and stuff

8.1 Paratec Perl Library

The paratec perl library is currently under construction. It is a set of perl modules designed to make it much easier to parse and modify paratec input and output files. Actually, it is designed to parse paratec input and output files, and only modify input files! ☺

If you want to use these modules in your perl scripts, you just have to set the environment variable $PERL5LIB to be the path to the paratec/src/perllib directory (so that perl will know where to find the modules), and then follow the instructions below to actually use the code.

Here are the manuals for the paratec modules, which are automatically generated from Plain Old Documentation embedded in the module itself:

- input.pm
- OUT.pm
- PW_LOG.pm

The library also has a few utility modules to make it easier to deal with things like vectors and tensors. Here is the documentation to the utility files:

- vector.pm
- tensor.pm

8.2 tools

8.2.1 gaussdos

This program uses the `EIG` file, which is produced when the output flag `eigvals` is set. It computes the density of states with a gaussian smearing. The input file has the following form:

```
0.2      gaussian smearing in eV
1000     number of mesh points for DOS
```

8.2.2 bsfix

In case you (or paratec!) screwed up with the fermi level for the band structure plot, you can use this awk script to shift the bandstructure plot around. Just look at the script, modify the fermi energy there, and run it with

```
bsfix >NEWBANDSTRUC
```

It will read the file BANDSTRUC, shift the bands according to the new fermi level, and write the output to NEWBANDSTRUC.

Chapter 9

Erratum etc.

9.1 Known difficulties

1. Matrix diagonalization bombs out. Increase diagsafety (section 5.3.3).

2. The double unit cell problems: If you run a supercell of twice the size of the real cell, i.e. if there are NONPRIMITIVE translations within the unit cell left, you will encounter `complex charge density` problems. This is due to shortcomings in the routine that finds the symmetry operations. Workaround: Only use the primitive unit cell, or break the symmetry completely by slightly perturbing the atomic positions. Alternatively, use the `noadjustshift` option to the `optimize` entry.

3. Loss of symmetry: After many structural relaxation steps (say 15 or so), roundoff errors break the symmetry of the system, and lead to strange results. Often, the forces are symmetrized incorrectly, leading to termination of the code. Workaround: Restart from last step after explicitly symmetrizing the structural parameters.

4. Bad charge density: If the program bombs out, it is often because a bad initial charge density has been used. There is no sanity checks being done on the CD file! Workaround: Remove old charge density (the CD file), and retry.

5. Structural relaxation scheme does not converge: There can be several reasons. Often, the system is about to undergo a phase transition, and is just intrinsically instable. Another possibility is that the accuracy of the diagonalization is not good enough (see the corresponding section), or the potential convergence criterion is too large.

6. Negative charge density. Should only appear when the initial charge is set up from the atomic charge density, or if a nonlinear core correction is used.

7. Complex charge density. Most of the times, this is because the coordinates of the atoms somewhat match the symmetry (say to 1e-8), but not quite. Fix: input lattice parameters and basis coordinates with very high accuracy, e.g. 1e-20. The gnu utility "bc" is very useful for this case, as it has arbitrary precision arithmetic.

8. NMR q problem: If the perturbation `nmr_q` for the NMR calculation is set too small, you will get wrong results unless the accuracy for the wave function is also cranked up. This affects mostly the susceptibility (G=0). An accuracy of 1e-12 for the wave function and a `nmr_q` of 0.01 is normally a good choice. In doubt, check if results are stable when `nmr_q` is varied.