RGWBS MANUAL

Contents

1. What it is

RGWBS is a suite of ALDA and GW-BSE codes developed by me (Murilo L. Tiago). Its purpose is to calculate several related quantities:

- electron polarizability,
- self-energy,
- excited states (charged and neutral) of an electronic system,

- quasiparticle wavefunctions,
- linear optical spectra.

It is based on two major theories, and users are expected to have a minimal experience with both: density-functional theory (see *e.g.* Martin's book) and many-body Green functions (see Aulbur *et al.*). The methodology implemented in RGWBS can be found in Tiago and Chelikowsky. Several executables are included in this package in an integrated fashion: output data of one executable is input data of another, some input data is shared among several executables. Development of the package started in 2004. All source files are written in Fortran 95. Below, each portion of the package is discussed in more detail.

---

2. Getting the package

RGWBS is distributed free of charge, under the GNU General Public License (GPLv1). It can be downloaded from my personal web page. Other mirror sites could become available in the future. Users are allowed and encouraged to improve the package to fit their own purposes, and they should leave improvements (and also bug fixes) available to the community. Please contact the author if you want to report bugs, send or receive updated versions of the package. Users should be aware that I cannot offer technical support and that I do not promise to reply to requests in a timely fashion.

---

3. Compiling the package

This package has been tested in several different architectures with different compilers, mostly the compilers distributed by IBM, Intel and PGI. It has been tested with GFortran as well. This is how compilation works step-by-step:

1. Look at the list of available machine-dependent parameter files under subdirectory `config`. Pick the one that best matches your machine and make modifications if necessary. The name of the parameter file defines the machine type. For instance, file `make.gfortran.h` was built for machine type `gfortran`. If needed, create a new file corresponding to your computer.
2. The parent directory has a driver `Makefile`. In that file, you must define the value of `MACH` as the machine type. For example, if you use gfortran in a basic linux box, you probably need this:

   ```
   MACH = gfortran
   ```

3. Type `make` for available options. Type `make all` if you want to compile everything. Most executable files have extension `.mpi` (with MPI) or `.ser` (serial, no MPI).

3.1. Dependencies with external libraries

3.1.1. Message passing

Use MPI to enable parallel computation. This suite is compliant with the MPI-1 standard. In your `make.*.h` file, add macro `-DMPI`.

3.1.2. Linear algebra

Operations with dense matrices is done using BLAS and LAPACK. You must have these libraries available. If necessary, define the location of these libraries in your `make.*.h` file.

3.1.3. Distributed linear algebra

Diagonalization of dense matrices can be done in parallel if ScaLAPACK is available. If necessary, define the location of these libraries in your `make.*.h` file. If you want to use MPI but do not want to use ScaLAPACK, add `-DNOSCALA` to the list of macros in your `make.*.h` file.

3.1.4. Fast Fourier Transform (FFT)

This package makes extensive use of FFTs, even if the electronic system is confined (see Onida *et al.*). FFTs are done by external libraries, so you need one of those. There is existing interface with these libraries:

- FFTW 3: this is arguably the most used free-software FFT library out there. Add macro `-DUSEFFTW3`.
- FFTW 2.*: an older version of FFTW. Add macro `-DUSEFFTW2`.
- MKL (Math Kernel Library): the proprietary math library distributed by Intel. Add macro `-DUSEMKL`.
- ESSL (Engineering Scientific Subroutine Library): another proprietary math library, from IBM. Add macro `-DUSEESSL`.

3.1.5. Exchange-correlation functional

Tools to calculate exchange-correlation functionals are an essential part of DFT codes. Most of them use locally developed tools. Fortunately, there are now GPL libraries. This package uses two libraries developed by the Octopus team: `libxc`, which is the main XC library, and `libstring`, which is a string conversion tool. Contact the Octopus developers (preferred) or myself (backup) if you need a copy of the libraries. You also need to define the location of the libraries in your `make.*.h` file.

3.2. Porting issues

As long as you use a decent fortran compiler and the environment is properly defined, porting to new machines should be painless. There are no major machine-dependent functions in the source code. If you use GFortran, you may have problems with BLAS dot product functions. In that case, add the auxiliary source file `aux_gfortran.f90` (see `make.gfortran.h`). Otherwise, add the generic wrapper `aux_generic.f90` found under subdirectory `shared`.

---

4. Compatibilities

4.1. DFT codes

Although based on first principles theories, RGWBS needs some input information (actually, a lot of it!) from elsewhere. DFT eigenvectors and eigenvalues must be provided by some DFT code. Also, norm-conserving pseudopotentials are necessary. Currently, there is interface with two DFT codes: PARSEC and PARATEC. Support for different DFT code can be built as long as it produces wavefunctions that can be expressed in real space on a regular grid without loss of relevant information (plane-wave codes and regular-grid codes are easy in that aspect, Gaussian basis codes can be challenging). See section on how to built interfaces with other DFT codes for more information.

PARSEC is a real-space DFT code developed at the University of Texas at Austin. It is a convenient code because it generates wavefunctions on a regular grid, which is the same layout used internally in RGWBS. Also, it handles confined electronic systems and systems with mixed periodicity (wires and slabs). RGWBS is compatible with versions 1.1 and newer versions of PARSEC. Some of the 1.3* versions provide support for mixed boundary conditions, but not the older ones. Contact the developers of PARSEC if you have inquiries about their code or want to use it.

PARATEC is a plane-wave DFT code, continuously improved and maintained at the University of California at Berkeley. It is distributed free of charge but download requires registration. Please consult the developers at http://www.nersc.gov/projects/paratec for distribution, usage etc.

4.2. Pseudopotentials

RGWBS uses norm-conserving pseudopotentials out of convenience. With them, the electron wavefunctions can be expressed on a regular grid with a spacing between points that is not too small. Of course, the type of pseudopotential depends on what your DFT code uses. Unfortunately, there is no standard for pseudopotential format yet. It seems that each developer of DFT codes takes pride in defining his/her own format and hope that it becomes the standard. But there are some good pseudopotential generators, for example Opium. Currently, RGWBS has built-in support for three formats:

- Martins: generated by the code of Prof. J. L. Martins. His generator produces non-relativistic, scalar-relativistic and spin-orbit pseudopotentials. This format is the native format used in PARSEC. See more details about it in the PARSEC webpage. Used with the PARSEC and PARATEC interfaces.
- FHI: from the Fritz-Haber-Institut, in Germany. This format is also supported by Abinit with one *caveat*: the header lines needed by Abinit should be removed. RGWBS uses the original FHI format. Used with the PARSEC interface.
- Martins-Wang: format used in the PEtot code. It is very similar to Martins' format. Used with the PARSEC interface.

---

5. The `tdlda` executable: electron polarizability

5.1. Description

This executable calculates the polarizability of the electronic system in energy representation. It follows the energy-orbital formulation popularized by M. Casida. For a set of occupied and unoccupied electronic orbitals given at input, it computes the eigenmodes of neutral excitations in the system. These eigenmodes are the excited states the system can go to when disturbed by an electric field of well defined energy. In a few words, if the system has charge density ρ and is disturbed by an electric potential V, then the response function is the polarizability $\Pi = \delta\rho/\delta V$. Instead of calculating Π directly, the code assumes that Π can be expanded in a sum over normal modes and calculates eigenvalues and eigenvectors of a suitably defined Hermitian matrix (see section II.A of Tiago and Chelikowsky). In a periodic system, `tdlda` calculates the polarizability resolved in crystal momenta, $\Pi_{\mathbf{q}}$, for the list of crystal momenta **q** given at input. More details about the polarizability can be found in Tiago and Chelikowsky and references therein.

The polarizability can be calculated under two approximations, depending on what type of exchange-correlation interactions are included:

- ALDA (adiabatic local density approximation): in this approximation, the exchange-correlation in the polarizability is calculated assuming the LDA kernel. This is the default polarizability.
- RPA (random phase approximation): in this approximation, exchange-correlation in the polarizability is ignored. This is the most popular type of polarizability used in GW calculations. See `no_lda_kernel`.

5.2. Input files

- `rgwbs.in`: this is an ASCII file with input parameters. All the possible input parameters (see below) have default values, so this file is optional. If it is not provided, the code will take all DFT wavefunctions and calculate the polarizability using the ALDA. Add this file if you want to set restrictions such as: ignore some DFT orbitals, ignore some transitions with high energy, or calculate the polarizability at crystal momenta different from **q**=0 (in periodic systems).
- All the input and output of your DFT code. For PARSEC users, that means: pseudopotentials, `parsec.in`, `parsec.dat` (version 1.2* or later) or `wfn.dat` (versions 1.1*). For PARATEC users, that means: pseudopotentials, `input` and `GWC`.
- `pol_diag.dat`: checkpoint file. Use it as input only if you want to restart the calculation from an interrupted run.

5.3 Input parameters in `rgwbs.in`

- `tdlda_cutoff` (*physical*), default = infinitely large, default unit = `eV`

  Defines an energy cut-off in the electronic transitions. With that, the polarizability is calculated taking only transitions from occupied orbitals to unoccupied orbitals such that the difference between DFT eigenvalues of those orbitals is less than the cut-off. This is useful if it happens that some high-energy transitions give very little contribution to the polarizability.

- `buffer_size` (*real*), default value = `0.38` (approximately 50,000 matrix elements)

  Maximum amount of CPU memory used to store kernel matrix elements, in MB. This is an optimization parameter. It should be smaller than the cache memory but not too small. Small values cause frequent request of heap memory. Large values reduce the frequency of requests but may lead to unused heap memory.

- `cache_size` (*real*), default value = `4000`

  Size of grid-point blocks used in real-space integrations. This parameter is an optimization parameter. It should be chosen smaller than the actual cache size of your computer. Default value is 4000, which means that around 4x32 kB of cache are used to do integrations.

- `no_lda_kernel` (*logical*), default = absent

  By default, the polarizability is calculated within the adiabatic local approximation (ALDA). If this line is found, the polarizability is calculated within the random-phase approximation (RPA) instead. See Tiago and Chelikowsky or Onida *et al.* for more details about the RPA.

- `tamm_dancoff` (*logical*), default = absent

  Enables the use of the Tamm-Dancoff approximation in the calculation of TDLDA polarizability. The Tamm-Dancoff approximation assumes that excitations from the ground state of an electronic system can only happen by removing electrons from occupied orbitals or adding electrons in unoccupied orbitals. Excitations that involve removing electrons from unoccupied orbitals or adding electrons in occupied orbitals are forbidden. It sounds unusual but actually it is possible to excite an electronic system from its ground state by adding one electron in an occupied orbital, because the ground state of the system is usually not expressible as a single Slater determinant. By default, the Tamm-Dancoff approximation is not used in non-periodic systems but it is used in periodic systems. Actually, the polarizability in periodic systems is *always* calculated using this approximation because it simplifies calculations dramatically and does not affect accuracy significantly.

- `distribute_representations` (*integer*), default = `1`

  Enables the distribution of a specified number of representations among processors. If the number of blocks of representations is not a factor of the number of processors, then the block size is reduced until the new block size is a factor (so that blocks are assigned to groups of processors with the same size). Small values hurt parallelization performance but make checkpointing more frequent. Of course, this is useful only if there is more than one irreducible representation, which can happen only in non-periodic systems.

- `dft_program` (*string*), default = `parsec`

  Choice of DFT program. Available options are `parsec` and `paratec`. For each choice, the user must provide all input and output files. See dft codes.

- `distribute_wavefunctions` (*integer*), default = `1`

  Enables the distribution of wavefunctions over a specified number of processors. Default value is such that wavefunctions are not distributed. If a value *x* > 1 is specified instead, then the values of input wavefunctions on the grid are distributed among *x* processors. Each processor handles a small number of grid points. The value of *x* must be a factor of the total number of processors divided by the number of representation groups, specified in distribute_representations. If that is not true, then *x* is reduced until the new value is a factor. Small values of *x* reduce the amount of MPI communication but increase memory usage. A rule of thumb is that the size of the wavefunction file (`parsec.dat` or `wfn.dat` etc.) divided by *x* should be less than half the amount of CPU memory available per processor. If the wavefunction file is too large because the grid spacing is small or because there are too many orbitals, then you should better distribute wavefunctions to avoid memory shortage.

- `tdlda_valence` (*block*), default = empty

  Defines the set of occupied orbitals (or valence bands in a periodic system) used to calculate the polarizability calculation. Relevant only in spin-unpolarized systems. By default, all existing occupied orbitals are used. Inside the block, you can either list orbitals one-per-line or give a range:

  ```
  begin tdlda_valence

  1

  range 3 5

  end tdlda_valence
  ```

  The example above means that orbitals 1, 3, 4 and 5 are defined.

- `tdlda_valence_up` (*block*), default = empty

  Defines the set of occupied orbitals in the majority spin channel ("spin up") of a spin-polarized system. It is similar to tdlda_valence and follows the same convention. Relevant only in spin-polarized systems.

- `tdlda_valence_down` (*block*), default = empty

Similar to tdlda_valence_up but now for the minority spin channel ("spin_down"). Relevant only in spin-polarized systems.

- `tdlda_conduction` (*block*), default = empty

  Defines the set of unoccupied orbitals (or conduction bands in a periodic system) used to calculate the polarizability calculation. By default, all existing unoccupied orbitals are used. Inside the block, you can either list orbitals one-per-line or give a range, just like in tdlda_valence. Relevant only in spin-unpolarized systems.

- `tdlda_conduction_up` (*block*), default = empty

  Similar to tdlda_conduction but now for the majority spin channel of a spin-polarized system. Relevant only in spin-polarized systems.

- `tdlda_conduction_down` (*block*), default = empty

  Similar to tdlda_conduction but now for the minority spin channel of a spin-polarized system. Relevant only in spin-polarized systems.

- `qpoints` (*block*), default value = empty

  List of crystal momenta ("**q** points") for which the polarizability is calculated. Coordinates are given in units of reciprocal lattice vectors only. This is relevant only in periodic systems. Each line has 5 floating point numbers. The first 3 numbers are the coordinates of each momentum (the last coordinates are not used in partially periodic systems but they must be input). The next number is an overall divisor. The last number is either 0 or 1: 0 for non-zero momenta, 1 for zero crystal momentum. For example:

  ```
  begin qpoints

  0.0 0.0 0.0 1.0 1

  0.0 0.0 0.5 1.0 0

  0.0 0.0 1.0 3.0 0

  end qpoints
  ```

  The above block defines three momenta: the first one is (0,0,0) and flagged as having zero length, the second one is (0,0,0.5) and the last one is (0,0,1⁄3). The last two momenta are flagged as having non-zero length. The divisor is useful if you want to input coordinates that are infinite fractions like 1⁄3 or 2⁄7. The zero-length flag is necessary because the polarizability at zero momentum has special behavior. See Hanke's article.

- `scissors` (*block*), default = empty

  Define a scissors operator in input orbital energies. This could be useful for example if you want to modify the DFT eigenvalues so that the band gap is not underestimated. Each line in this block defines a different operator. This block should have 6 numbers on each line with this format:

  ```
  spin band_min band_max const ref slope
  ```

where

- `spin` : spin channel of orbitals to which this operator is applied (value `1` or `2`). In non-polarized systems, value `1` applies to both spin channels.
- `band_min` : order of lowest orbital to which this operator is applied. Integer.
- `band_max` : order of highest orbital to which this operator is applied. Integer.
- `const` : constant value added to the energy of orbitals included in this operator. Real (float), units are `eV`.
- `ref` : reference energy used to add a "stretch" to the energy of orbitals. Real (float), units are `eV`.
- `slope` : amount of "stretch" applied to energy of orbitals. Real (float) value.

One example is the pair of operators below:

```
begin scissors

1 1 4 0.0 10 -0.1

1 5 10 1.0 11 0.2

end scissors
```

It specifies that the energy of bands 1 through 4 is changed according to the rule $E_{new} = E_{old} - 0.1\times(E_{old} - 10\text{ eV})$ and the energy of bands 5 through 10 is changed according to the rule $E_{new} = E_{old} + 1\text{ eV} + 0.2\times(E_{old} - 11\text{ eV})$.

- `point_group_tables` (*block*), default = empty

  Specifies additional point groups to be used. Each line in this block contains the name of an input file with the specifications of the point group. Usually, additional point groups lead to modest gains in performance. This is more useful to calculate BLIP transformed wavefunctions.

- `rpa_spectrum_only` (*logical*), default = absent

  This flag actually forces the skip of polarizability calculation almost entirely. With that, only oscillator strengths of uncorrelated transitions are calculated and file `eigenvalues_rpa` is printed out. This is useful if you want to study the convergence of sum rule. See example_1.

- `tdlda_triplet_kernel` (*logical*), default = absent

  Calculates the polarizability for spin-triplet excitations instead of spin-singlet excitations (the ones that are actually accessible in linear optics). This flag is useful only in non-spin polarized systems. See Vasiliev *et al.* for more details about spin-triplet excitations.

- `no_exchange` (*logical*), default = absent

  Removes the Hartree term (sometimes called "exchange" or "Coulomb") term of the exchange-correlation kernel. This term is the $\mathbf{K}^x$ of Equation 4 in Tiago and Chelikowsky. If this input flag is used with `bsesolv`, the same Hartree term (K^x of Equation 36 in Tiago and Chelikowsky) is removed from the BSE.

- `truncate_coulomb` (*logical*), default = absent

  Remove the long-wavelength portion of the Hartree term when the polarizability for zero crystal momentum is calculated. This is relevant in periodic systems only. Hanke shows that the polarizability with long-wavelength Hartree is the "full polarizability" (jargon from quantum field theory) and it leads to the inverse dielectric function. The polarizability without long-wavelength Hartree is the "reduced polarizability" and it leads to the dielectric function. If you wish to calculate the dielectric function of a solid, you probably want to use this flag.

5.4. Output files

- Standard output sent to screen: among other things, it contains the first few polarizability eigenvalues, sum rule and static susceptibility.
- `eigenvalues_lda`: list of eigenvalues of the polarizability (in eV) for each eigenmode, followed by the corresponding oscillator strengths along the three Cartesian directions (or along the three major crystalline directions, in periodic systems). The oscillator strength is defined in Equation 9 of Tiago and Chelikowsky. It can be used to calculate absorption spectra.
- `eigenvalues_rpa`: contrary to what the name seems to indicate, this is just the set of uncorrelated excitation energies (difference between energy of occupied orbitals and unoccupied orbitals for each excitation), followed by the corresponding oscillator strengths.
- `pol_diag.dat`: binary file with all polarizability eigenvectors. The format is compatible with codes `sigma` and `bsesolv` and it can be used as input to them. It could also be useful for post-processing.

6. The `sigma` code: self-energy in the family of GW approximations

6.1. Description

This executable calculates the electron self-energy and quasiparticle energies. The purpose of this code is to determine electronic orbitals (both wavefunctions and energies) that are more realistic than the DFT eigenvectors and eigenvalues. One strategy is to replace the DFT eigenvalue equation with a quasi-particle eigenvalue equation, where the exchange-correlation functional is replaced with a self-energy (the formalism is presented in great detail for example in Aulbur *et al.*). Inside the code, the self-energy $\Sigma(\mathbf{r},\mathbf{r}',E)$ is never calculated explicitly. Instead, what is calculated are matrix elements involving the self-energy and pairs of orbitals, $\langle i|\Sigma|j\rangle$, or pairs of Bloch functions $\langle i\mathbf{k}|\Sigma|j\mathbf{k}\rangle$ (since the self-energy has crystal symmetry, matrix elements involving Bloch functions with different crystal momenta are zero). The user can specify in `rgwbs.in` which matrix elements are computed and how they are computed (which approximation, which parameters etc.). After the self-energy is calculated, the quasi-particle eigenvalue equation is diagonalized using the input electron orbitals as basis functions, and the resulting eigenvalues are saved in disk. The main output of this code is self-energy and quasi-particle eigenvalues. Other output information is: quasi-particle wavefunctions, a breakdown of self-energy matrix elements (useful for benchmarking or debugging), electron total energy, polarizability.

Normally, the self-energy is calculated within the G_0W_f (sometimes referred to as $G_0W_{LDA}\Gamma_{LDA}$) approximation, non-self-consistent. In that approximation, the screened Coulomb interaction W is obtained within the time-dependent ALDA, a vertex insertion is calculated also within the ALDA, and G is simply the Kohn-Sham Green's function. Other types of self-energies can be calculated as well: self-consistent GW_f, G_0W_{RPA} (RPA screened Coulomb interaction, no vertex insertion) or its self-consistent counterpart. At a lower level of approximation, one can calculate exchange and correlation functions that are approximations to the self-energy. There are two classes of approximations of that type: Hartree-Fock (equivalent to removing correlation for the self-energy), or model DFT functionals. See `exchange_correlation` for a list of available options.

In order to calculate the GW self-energy, one must have the polarizability calculated (notice that many GW codes calculate the dielectric matrix instead, which is anyway related to the polarizability). Users have two options: they can run the `tdlda` code before `sigma` and use file `pol_diag.dat` as input; or they can run `sigma` right away. If file `pol_diag.dat` is not present, then the code will calculate the polarizability before actually calculating the self-energy. Be careful if you prefer to run `tdlda` beforehand and the electronic system has crystal periodicity: `tddla` must calculate the polarizability at a full Monkhorst-Pack grid (see `qpoints`), one of the **q** points must have zero length *and* the Coulomb potential must not be truncated (do *not* use `truncate_coulomb`).

Regarding numerical complexity, calculating the self-energy is not trivial. It is hard to quantify the computational cost, but one can easily expect self-energy calculations to be anything from 10 to 1000 times more demanding than DFT calculations for the same system. Fortunately, `sigma` makes good use of parallelization even in machines with hundreds of processors. Also, memory is distributed with little communication overhead. A `sigma` run can be done using information from previous, incomplete calculations (see `read_checkpoint`).

6.2. Input

- `rgwbs.in`: this is an ASCII file with input parameters. All the possible input parameters (see below) have default values, so this file is optional. If it is not provided, the code will calculate self-energy matrix elements for all wavefunctions found in input file. Add this file only if you want to set restrictions in the matrix elements to be calculated, enable flags, or specify your own values for input parameters.
- All the input and output of your DFT code. For PARSEC users, that means: pseudopotentials, `parsec.in`, `parsec.dat` (version 1.2* or later) or `wfn.dat` (versions 1.1*). For PARATEC users, that means: pseudopotentials, `input` and `GWC`.
- `pol_diag.dat`: if available, the `sigma` code will skip the calculation of polarizability. See `read_checkpoint`. Optional.
- `sigma.chkpt.dat`: checkpoint file. This file is usually created and updated several times during a calculation. Leave it available if you wish to restart from an incomplete previous calculation. See `read_checkpoint`. Optional.
- `wpol0.dat`: file with the COHSEX screened interaction. The GW self-energy here is corrected with a static remainder, as discussed in Appendix B of Tiago and Chelikowsky. This file has the potential W(**r**) defined in equation B2 and a similar potential for the vertex self-energy, both of them computed on all points in the real-space grid. If this file is available, the static remainder is calculated from the contents of this file instead of calculated from scratch. Optional.
- group table files: see `point_group_tables`. Optional.
- `occup.in`: see `read_orbital_occupancies`. Optional.

6.3 Input parameters in `rgwbs.in`

- `tdlda_cutoff`
- `buffer_size`
- `cache_size`
- `no_lda_kernel` (*logical*), default = absent

In `sigma` calculations, this flag also removes the LDA kernel, Equation 30 of Tiago and Chelikowsky. With this flag, the self-energy is calculated in the so-called G_0W_{RPA} (without self-consistency) or GW_{RPA} approximation (with self-consistency). By default, the self-energy is calculated under $G_0W_{LDA}\Gamma_{LDA}$ (without self-consistency) or $GW_{LDA}\Gamma_{LDA}$ approximations (with self-consistency).

- `tamm_dancoff`
- `distribute_representations`
- `dft_program`
- `distribute_wavefunctions`
- `tdlda_valence`
- `tdlda_valence_up`
- `tdlda_valence_down`
- `tdlda_conduction`
- `tdlda_conduction_up`
- `tdlda_conduction_down`
- `qpoints`

  List of crystal momenta for which the polarizability is calculated. For the `sigma` code, this list must contain a complete Monkhorst-Pack grid (*i.e.* the **q** points must form a regular mesh covering the entire first Brillouin zone, including points related to each other by symmetry operations).

- `scissors`
- `point_group_tables`
- `max_number_states` (*integer*), default value = maximum possible

  Highest orbital included in Green's function summation. This parameter defines where to stop the sum over n in Equations 24 and 30 of Tiago and Chelikowsky. If not specified, use all orbitals in the wavefunction file.

- `max_number_states_cohsex` (*integer*), default value = maximum possible

  Highest state for which the self-energy is computed under the COHSEX approximation. States below this order have self-energy calculated within the COHSEX approximation unless self-energy matrix elements are requested explicitly with blocks "`diag`" and "`offdiag`".

- `energy_range` (*physical*), default value = `20`, default unit = `eV`

  Energy range used to calculate self-energy. Used together with `energy_data_points`. If user inputs "`energy_range = Δ`" and "`energy_data_points = N`", then the self-energy is evaluated at `N` values of energy regularly spaced between $E_{in} - \Delta/2$ and $E_{in} + \Delta/2$ (E_{in} is the input energy eigenvalue), including the bounds.

- `energy_data_points` (*integer*), default value = `21`

  Number of data points used to calculate self-energy. Used together with `energy_range`. If user inputs `energy_range = Δ` and `energy_data_points = N`, then the self-energy is evaluated at `N` values of energy regularly spaced between $E_{in} - \Delta/2$ and $E_{in} + \Delta/2$ (E_{in} is the energy in input wavefunction file), including the bounds. As shown for example in Equation 35 of Hybertsen and

Louie, the self-energy should be calculated at the quasi-particle energy, which is not known beforehand. This code determines the actual quasi-particle energy by solving Equation 35 using a Newton-Raphson method. The self-energy is calculated in between points of the energy grid using spline interpolation. The energy range should be such that the final quasi-particle energy is inside the energy range, $|E_{qp} - E_{in}| < \Delta/2$. The number of data points should be small enough so that the spline interpolation is numerically stable. If anything goes wrong, you will see some warning in standard output like "`ERROR!! Newton-Raphson exceeded maximum number of iterations`".

- `renormalize_sumrule` (*logical*), default = absent

  With this flag, the polarizability eigenvalues are rescaled so that the sum rule is set to one and the static susceptibility is kept fixed. This is an *ad hoc* trick to improve the accuracy of the polarizability but there is no guarantee it actually improves anything. Use it with caution.

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

  Maximum number of self-consistent iterations to be performed. Setting a value greater than 0 enables self-consistency in the calculation of self-energy. In general, every iteration consumes the same amount of memory and CPU time.

- `convergence_potential` (*physical*), default value = `0.001`, default unit = `eV`

  Maximum difference between old and new interaction potential ($= \Sigma_{out} - \Sigma_{in}$) at convergence, where Σ_{out} is the self-energy at current iteration and Σ_{in} is the self-energy at the previous iteration (or V_{xc} if there is no previous iteration). Used only in self-consistent GW calculations.

- `read_checkpoint` (*logical*), default = absent

  Enables the reading of checkpoint files (`pol_diag.dat`, `sigma.chkpt.dat`, `wpol0.dat`). If they do not exist, calculation proceeds normally. If they exist, their contents is stored in memory and not re-calculated. This is useful if you need to continue a `sigma` calculation that was interrupted for some reason. By default, checkpoint files are not read between one self-consistent GW iteration and the subsequent one. Also, checkpoint files are never read if the exchange-correlation type is not `gw` (for example, some model DFT exchange-correlation).

- `write_qp_wavefunctions` (*logical*), default = absent

  With this flag, the code prints out quasi-particle wavefunctions to file `parsec_qp.dat`. This file has the same format as `parsec.dat`, but it contains quasiparticle orbitals instead of DFT orbitals. In addition, this flag forces the write out the self-energy matrix elements to file `sigma_mtxel_qp.dat`. By default QP wavefunctions are never printed out. In self-consistent GW calculations, the QP wavefunctions are printed at each iteration. In self-consistent calculations with model exchange-correlations, the QP wavefunctions are printed only in the last iteration.

- `read_vxc_matrix_elements` (*logical*), default = absent

  With this flag, the code reads matrix elements of the exchange-correlation operator from file `sigma_mtxel.dat`, instead of calculating them. This is useful if self-consistency is imposed and we want to restart a calculation from previous runs.

- `read_orbital_occupancies` (*logical*), default = absent

  Forces the reading of orbital occupancies from file `occup.in` (compatible with PARSEC, see its documentation). This is useful in spin-polarized systems where DFT produces fractional or otherwise incorrect occupancies.

- `cohsex_approximation` (*logical*), default = absent

  With this flag, the GW correlation is always calculated using the COHSEX approximation. This flag is useful for debugging purposes.

- `exchange_correlation` (*string*), default value = `gw`

  Defines different exchange-correlation types. By default, the GW approximation is used to calculate both exchange and correlation. Alternatively, one can also use simpler approximations such as the Hartree-Fock approximation (no correlation, exact exchange), or DFT with some type of exchange-correlation functional. In that case, the electron wavefunctions are calculated for the specified exchange-correlation type using the existing set of input wavefunctions to define a "Hilbert space". This is equivalent to solving the Hartree-Fock equations or the DFT equations in an explicit basis set. Of course, the accuracy of the calculation depends critically on the ability of input wavefunctions to span a sufficiently large Hilbert space. Exchange-correlation types other than GW are usually used with some sort of self-consistency. Available options are:

  - `gw`: GW exchange-correlation (default).
  - `hartree_fock`: Hartree-Fock exchange-correlation.
  - `b3lyp`: hybrid exchange-correlation functional, see Martin's book.
  - `blyp`: generalized-gradient exchange-correlation functional, see Martin's book.
  - `lda_ca`: local-density functional parametrized by Perdew and Zunger (1981), see Martin's book.
  - `gga_pbe`: generalized-gradient functional parametrized by Perdew, Burke and Ernzerhof (1996), see Martin's book.

- `scratch_disk_size` (*physical*), default value = as large as necessary, default unit = `MB`

  Maximum amount of disk space used to store real-space potentials, in MB. By default, this code uses all the necessary disk space such that 99.99% of the electron density is accounted for. The motivation behind this parameter is that quantities associated to the static polarizability (see Appendix B of Tiago and Chelikowsky) are calculate in real space and they often involve dumping a huge amount of data to disk. If your computer happens to have small amount of disk space available, then you may need to specify an upper bound. Otherwise, let it use as much disk space as needed. The amount of data written to disk is printed in standard output around line beginning with "`Calculating W_pol in static limit with`".

- `dynamic_energy_resolution` (*physical*), default value = `0.1d0`, default unit = `Ry`

  Energy resolution used in energy denominators. The GW correlation has energy denominators (see *e.g.* Equations 24 and 30 of Tiago and Chelikowsky) that can diverge if there is a match between polarizability eigenvalues and DFT eigenvalues. In order to prevent singularities, each energy denominator $1/E$ is replaced by the function $E/(E^2 + ecut^2)$, which has no singularity at E=0. For self-energy calculations, this parameter is expected to strongly affect quasi-particle energies of deep occupied orbitals. Orbitals around the gap are less affected. Values of the order of `1 eV` are typical.

Large values lead to unphysically weak energy dependence of the self-energy. Small values lead to sharp divergences in the self-energy. See example 2 for more details about this parameter.

- `qp_mixing_param` (*real*), default value = `1`

  Amount of new self-energy correction (= $\Sigma_{out} - \Sigma_{in}$) to be added to the input Hamiltonian. Its value should be chosen between zero and one. By default, the new self-energy replaces completely the old one (`qp_mixing_param` = 1). Using mixing parameters less than one may improve the stability of self-consistency cycles.

- `self_energy_cutoff` (*physical*), default value = `0`, default unit = `eV`

  If there is a large amount of numerical noise in the calculation of self-energy (for example because of underconverged numerical parameters), then the self-consistency cycles may become unstable. With a self-energy cut-off greater than zero, matrix elements of the operator $\Sigma_{out} - \Sigma_{in}$ with absolute value less than the specified cut-off are neglected.

- `sigma_energy` (*string*), default value = `left`

  Since the self-energy is an energy-dependent operator, one cannot add it to a Hamiltonian and start diagonalizing the Hamiltonian before some approximations are assumed. Usually, the diagonal part of the self-energy is evaluated at the quasi-particle energy (see Equation 35 of Hybertsen and Louie) but there is no consensus about how to handle the off-diagonal part. By default, the off-diagonal part is evaluated at the energy of the orbital on the left side: $\langle i|\Sigma|j\rangle = \langle i|\Sigma(E = E_i)|j\rangle$. Alternatively, one can evaluate it at the energy of the right-side orbital or at the average of the two energies. Possible choices are:

  - `left`: default
  - `right`: use the right orbital, $\langle i|\Sigma|j\rangle = \langle i|\Sigma(E = E_j)|j\rangle$.
  - `average`: use the average energy, $\langle i|\Sigma|j\rangle = \langle i|\Sigma(E = [E_i+E_j]/2)|j\rangle$.

- `no_hqp_symmetrize` (*logical*), default = absent

  When the self-energy is evaluated at specified energies, it may no longer be Hermitian, which makes the quasi-particle Hamiltonian non-Hermitian. By default, the off-diagonal part is explicitly symmetrized: $\langle i|\Sigma|j\rangle = [\ \langle i|\Sigma|j\rangle + \text{complex conjugate}\ ]\times 1/2$. With this flag, symmetrization is suppressed.

- `sigma_kpoints` (*block*), default = absent

  List of crystal momenta ("**k**-points") at which the self-energy is computed. This block is relevant only in periodic systems. If not found, only the self-energy at the Γ point (**k** = 0) is computed. If found then more **k**-points are used. The format is similar to qpoints but the zero-length flag is not necessary:

  ```
  begin sigma_kpoints

  0.0 0.0 0.0 1.0 # Γ point

  0.0 0.0 0.5 1.0 # point (0, 0, 1/2)
  ```

```
0.0 0.0 1.0 3.0 # point (0, 0, 1/3)

end sigma_kpoints
```

- `diag` (*block*), default = absent

  List of electronic orbitals (or bands) for which the diagonal part of the self-energy is computed. If not found, the code calculates the self-energy for all orbitals found in the wavefunction file. Like tdlda_valence block, one can either list separate orbitals or give the lower/upper bounds of a range of orbitals.

- `offdiag` (*block*), default = absent

  List of electronic orbitals for which off-diagonal matrix elements of self-energy are computed. If not found, then calculate the self-energy for all orbitals found in the wavefunction file. Like tdlda_valence block, one can either list separate pairs of orbitals or give the lower/upper bounds of a range of orbitals.

6.4. Output

A regular run generates output printed out to standard output (screen). The user may want to save that output. It contains information about input data (parameters from the DFT calculation, timings, estimate of memory usage etc.). The same output files produced by `tdlda` are generated as well: `eigenvalues_rpa`, `eigenvalues_lda`, `pol_diag.dat`. Additional output is written in these files:

- `hmat_qp`: quasiparticle eigenvalues. These are the eigenvalues obtained by diagonalizing the quasiparticle Hamiltonian (Equation 32 of Tiago and Chelikowsky). Its format is appropriate to use by the `bsesolv` code. Data written in this file are:

  - 1st column: order of orbital `i`.
  - 2nd column: order of orbital `j`.
  - 3rd column: matrix element $\langle i\mathbf{k}|\Sigma|j\mathbf{k}\rangle$, in units of eV.
  - 4th column: spin orientation of orbitals `i` and `j`.
  - 5th column: **k**-point index of orbitals `i` and `j` (relative to the list of **k**-points present in the input wavefunction file).
  - 6th column: energy eigenvalue of orbital `i` as given in the input wavefunction file (column printed only for diagonal matrix elements, `i` = `j`).
  - 7th column: energy given in 6th column plus a scissors operator (if given).

- `hmat_qp_nostatic`: contains data similar to `hmat_qp` with the difference that the static remainder (see Appendix B of Tiago and Chelikowsky) is not included. This is important only for debugging purposes.
- `eqp_*_*_*`: these files also contain quasiparticle energies. Each file corresponds to a particular spin orientation (`up` or `down`, or blank for spin-unpolarized systems), **k**-point and iteration, indicated in that sequence in the file name. The contents is almost self-explanatory. Each line corresponds to a quasiparticle orbital with these data: orbital order, representation, occupancy, quasiparticle energy (`E_0`), $\Sigma - \delta V_{\mathrm{Hartree}}$ (`V_xc + delta_Vh`), $\Delta V_{\mathrm{Hartree}}$ (`delta_Vh`), imaginary part of the self-energy (`Im{Sigma}`). All energy quantities are given in electron-volts. Notice that, since the quasiparticle equation has been diagonalized, the code computes a new electron density using the quasiparticle wavefunctions. As a result, a new Hartree potential should be used with the new electron

density. $\delta V_{Hartree}$ is the difference between new and old Hartree potentials.

- `sigma_*_*_*`: these files contain a breakdown of the self-energy, both diagonal and off-diagonal parts. Name convention is equal to `eqp_*_*_*` files. All energy quantities are given in eV. The first few lines correspond to diagonal matrix elements:

  - 1st column: order of orbital `i`.
  - 2nd column: occupancy of orbital `i`.
  - 3rd column: DFT energy eigenvalue (*i.e.* read from input wavefunction file).
  - 4th column: exchange-correlation matrix element $\langle i|V_{xc}|i\rangle$(or previous self-energy matrix element if this is part of a self-consistent calculation), where `i` is a DFT (input) orbital.
  - 5th column: exact (Fock) exchange $\langle i|\Sigma_x|i\rangle$.
  - 6th column: screened exchange portion of self-energy. Printed out for debugging purposes.
  - 7th column: correlation self-energy $\langle i|\Sigma_c|i\rangle$ (omitting vertex contributions), as defined in Equation 24 of Tiago and Chelikowsky.
  - 8th column: vertex self-energy $\langle i|\Sigma_f|i\rangle$, as defined in Equation 30 of Tiago and Chelikowsky.
  - 9th column: $\langle i|\Sigma - V_{xc}|i\rangle$ matrix element.
  - 9th column: renormalization factor, $z^{-1} = 1 - d\Sigma/dE$.
  - 10th column: DFT (input) energy eigenvalue of orbital `i` after applying any scissors operator defined in `rgwbs.in`.
  - 11th column: first approximation to quasiparticle energy, defined as the energy E such that $E_i = E^{DFT}_i + \langle i|\Sigma(E_i) - V_{xc}|i\rangle$ (self-energy evaluated at energy E_i). This is the quantity present in the diagonal part of the quasiparticle Hamiltonian (Equation 32 of Tiago and Chelikowsky).

  The last lines have the off-diagonal part:

  - 1st column: order of `i`-th orbital.
  - 2nd column: order of `j`-th orbital.
  - 3rd column: DFT exchange correlation, $\langle i|V_{xc}|j\rangle$ matrix element.
  - 4th column: Fock exchange, $\langle i|\Sigma_x|j\rangle$.
  - 5th column: correlation, $\langle i|\Sigma_c|j\rangle$, evaluated at the energy specified by `sigma_energy`.
  - 6th column: correlation, $\langle j|\Sigma_c|i\rangle$, evaluated at the energy specified by `sigma_energy`.
  - 7th column: vertex, $\langle i|\Sigma_f|j\rangle$, evaluated at the energy specified by `sigma_energy`.
  - 8th column: vertex, $\langle j|\Sigma_f|i\rangle$, evaluated at the energy specified by `sigma_energy`.
  - 9th column: operator $\langle i|\Sigma - V_{xc}|j\rangle$, evaluated at the energy specified by `sigma_energy`.
  - 10th column: operator $\langle j|\Sigma - V_{xc}|i\rangle$, evaluated at the energy specified by `sigma_energy`. This and the previous matrix element are the off-diagonal part of the quasiparticle Hamiltonian, Equation 32 of Tiago and Chelikowsky.

  If Hartree-Fock or a model exchange-correlation is used instead of GW (see exchange_correlation), then some of the columns above are omitted.
- `sigma_nostatic_*_*_*`: similar to the previous but without the static remainder correction.
- `wpol0.dat` and `wpol_rho.dat`: these files contain the COHSEX screened Coulomb interaction. See wpol0.dat.

7. The `bsesolv` executable: Bethe-Salpeter equation

7.1. Description

This code diagonalizes the Bethe-Salpeter equation and prints out excitation energies. The formalism behind this code is described is good detail in these articles: Onida *et al.*, Rohlfing and Louie and Tiago and Chelikowsky. Just like the `tdlda` code, `bsesolv` calculates the polarizability of the electronic system. But this polarizability is calculated within the many-body framework of the electron-hole Green's function and its dynamical equation (the Bethe-Salpeter equation).

7.2. Input

- `rgwbs.in`: this is an ASCII file with input parameters. All the possible input parameters (see below) have default values, so this file is optional. If it is not provided, the code will diagonalize the Bethe-Salpeter equation using all wavefunctions found in input file, using default values for input parameters (which are safe for most cases). Add this file only if you want to set restrictions in the BSE, enable flags, or specify your own values for input parameters.
- All the input and output of your DFT code. For PARSEC users, that means: pseudopotentials, `parsec.in`, `parsec.dat` (version 1.2* or later) or `wfn.dat` (versions 1.1*). For PARATEC users, that means: pseudopotentials, `input` and `GWC`.
- `pol_diag.dat`: if available, the `bsesolv` code will skip the calculation of polarizability. Notice that this polarizability is the one used in the direct Kernel, *not* the one used to calculate excited states. Optional.
- `bse_chkpt.dat`: checkpoint file. This file is usually created and updated several times during a calculation. Leave it available if you wish to restart from an incomplete previous calculation. Optional.
- group table files: see `point_group_tables`. Optional.
- `occup.in`: see `read_orbital_occupancies`. Optional.
- `hmat_qp`: file with the quasiparticle Hamiltonian calculated in the basis of input wavefunctions. Generated during a `sigma` calculation. Optional.

7.3 Input parameters in `rgwbs.in`

- `tdlda_cutoff`
- `buffer_size`
- `cache_size`
- `no_lda_kernel` (*logical*), default = absent

  In `bsesolv` calculations, this flag also removes the LDA kernel, Equation 30 of Tiago and Chelikowsky. With this flag, the self-energy is calculated in the so-called G_0W_{RPA} (without self-consistency) or GW_{RPA} approximation (with self-consistency). By default, the self-energy is calculated under $G_0W_{LDA}\Gamma_{LDA}$ (without self-consistency) or $GW_{LDA}\Gamma_{LDA}$ approximations (with self-consistency). The self-energy is related to the BSE kernel through equation 35 of Tiago and Chelikowsky.

- `tamm_dancoff`
- `distribute_representations`
- `dft_program`

- distribute_wavefunctions
- tdlda_valence
- tdlda_valence_up
- tdlda_valence_down
- tdlda_conduction
- tdlda_conduction_up
- tdlda_conduction_down
- qpoints

  List of crystal momenta for which the polarizability is calculated. For the bsesolv code, this list must contain a complete Monkhorst-Pack grid (*i.e.* the **q** points must form a regular mesh covering the entire first Brillouin zone, including points related to each other by symmetry operations).

- scissors
- point_group_tables
- energy_reference (*physical*), default value = undefined, default unit = eV

  Energy reference for calculation of dynamical screened interaction. If absent, then the static screened interaction is calculated. That is the approximation discussed in section II.C of Tiago and Chelikowsky. In reality, the screened interaction has dynamical effects (see Equation 23 of Rohlfing and Louie. If the energy reference is defined (Ω_s) then screening is calculated at that fixed energy.

- bse_cutoff (*physical*), default value = maximum possible, default unit = eV

  Defines an energy cut-off in the BSE transitions. This parameter is similar to tdlda_cutoff but it removes high energy transitions from the BSE polarizability only.

- no_eigenvectors (*logical*), default = absent

  Suppresses the printout of BSE eigenvectors. By default, eigenvectors of the BSE equation are written in file bse_diag.dat. If this flag is present, the file is not written.

- bse_triplet_kernel (*logical*), default = absent

  If this line is present, spin-triplet excitations are calculated instead of spin-singlet (as normally done). This is similar to tdlda_triplet_kernel in that it enables the calculation of polarizability for spin-triplet excitations. This is relevant only in spin-unpolarized systems.

- truncate_coulomb
- use_mixing (*logical*), default = absent

  This flag disables the Tamm-Dancoff approximation in BSE. Normally, the BSE is diagonalized assuming the Tamm-Dancoff approximation. With this flag present, the BSE is diagonalized with mixing between positive and negative energy transitions added (that is, no Tamm-Dancoff). See Onida *et al.* or Rohlfing and Louie for more details about the impact of the Tamm-Dancoff approximation in the Bethe-Salpeter polarizability.

- no_exchange
- renormalize_sumrule
- read_orbital_occupancies

- `exchange_correlation`

  In `bsesolv`, the implemented options are only `gw`(default) and `Hartree_Fock`.

- `bse_energy_resolution` (*physcial*), default value = `0`, default unit = `eV`

  Energy resolution used in energy denominators. Each energy denominator 1⁄E is replaced by the function E⁄(E^2 + ecut^2), which has no singularity at E=0. For self-energy calculations, this parameter is expected to strongly affect quasi-particle energies of deep occupied levels; levels around the gap are less affected. This resolution is applied to the static and dynamic polarizability insertions in the BSE kernel.

- `bse_valence` (*block*), default = absent

  Defines the set of occupied orbitals (or valence bands in periodic systems) in the Bethe-Salpeter equation. If this block is absent, then all orbitals found in wavefunction file are included in the BSE. Just like `tdlda_valence`, orbitals can be declared one at a line or a range of orbitals can be input.

- `bse_valence_up` (*block*), default = absent

  Defines the set of occupied orbitals (or valence bands in periodic systems) in the Bethe-Salpeter equation, majority spin channel. Follows the same convention as `bse_valence`.

- `bse_valence_down` (*block*), default = absent

  Defines the set of occupied orbitals (or valence bands in periodic systems) in the Bethe-Salpeter equation, minority spin channel. Follows the same convention as `bse_valence`

- `bse_conduction` (*block*), default = absent

  Defines the set of unoccupied orbitals (or conduction bands in periodic systems) in the Bethe-Salpeter equation. If this block is absent, then all orbitals found in wavefunction file are included in the BSE. Just like `tdlda_conduction`, orbitals can be declared one at a line or a range of orbitals can be input.

- `bse_conduction_up` (*block*), default = absent

  Defines the set of unoccupied orbitals (or conductionbands in periodic systems) in the Bethe-Salpeter equation, majority spin channel. Follows the same convention as `bse_conduction`.

- `bse_conduction_down` (*block*), default = absent

  Defines the set of unoccupied orbitals (or conduction bands in periodic systems) in the Bethe-Salpeter equation, minority spin channel. Follows the same convention as `bse_conduction`.

- `qpoints_bse` (*block*), default = absent

  Defines the **q** point (crystal momentum) for which the BSE polarizability is calculated. Just like `qpoints`, each line has 5 numbers: the three coordinates of **q** point, a common divisor, and the zero-length flag. Contrary to `qpoints`, only one point can be defined.

7.4. Output

A large amount of data is written in standard output (screen), and it is useful to save it for future reference. This code generates a TDLDA (or RPA) polarizability and the associated output files: `eigenvalues_rpa`, `eigenvalues_lda`, `pol_diag.dat`. Additional output files are:

- `eigenvalues_bse`: eigenvalues and associated oscillator strengths obtained by diagonalizing the Bethe-Salpeter equation, Equation 36 of Tiago and Chelikowsky.
- `bse_diag.dat`: eigenvectors of the BSE. This file is written in a format identical to `pol_diag.dat`, which makes post-processing and data analysis easier. Input flag `no_eigenvectors` suppresses the print out of this file.

---

8. Post-processing tools

8.1 `absp`

This code reads excitation energies and oscillator strengths from any of these files: `eigenvalues_rpa`, `eigenvalues_lda`, `eigenvalues_bse`. The user must rename the input file as `eigenvalues` and add a second number to the first line of that file. That number is the energy resolution η (in eV) used in the widening of absorption lines. This code does not calculate line widths, which are important because all measured absorption lines have some finite width even if they are "monochromatic". Several quantities are printed in output: the absorption cross section (useful if the electronic system is confined), absorption spectrum (imaginary part of the dielectric function, or the polarizability). The coding is very simple, and any user should be able to read it if he/she wishes to know how each quantity is calculated.

8.2 `proj_pol`

In many situations, it is important to characterize absorption lines. If some absorption line is dominated by a single transition (meaning: one electron being promoted from an occupied orbital to an unoccupied orbital), then one often wants to know what is the dominant transition and its weight. This code prints out the weight of specified transitions in specified excitations. In other words, for specified occupied and unoccupied orbitals `v` and `c`, this code prints out the weight $|F_{vc}^{i}|^2$ in `i`-th excitation. Input file is `pol_diag.dat`. One can also give a `bse_diag.dat` file as input but it must be renamed. The choices of orbitals are typed in standard input (keyboard in most cases). Once you compile the code, just run `proj_pol` in a shell window and type in the quantities it asks for.

8.3 `chkpt_bin_asc`

This code transcribes the contents of file `sigma.chkpt.dat` into ASCII format. Input is file `sigma.chkpt.dat`. Output is file `sigma.chkpt.dat.asc`. Output file contains a detailed breakdown of self-energy components: DFT exchange-correlation matrix elements, Fock exchange, correlation part of self-energy in a wide range of energy values, real and imaginary parts of self-energy. This is useful if one wishes to study the imaginary part of the self-energy or identify satellites in the spectral function.

---

9. Examples

9.1. Example 1: convergence in TDLDA

Working directory = `tutorial_1`.

Estimated run time on a standard linux box = 20 min.

Often, the low-energy end is the most important portion of the absorption spectrum, because it defines the absorption edge. Fortunately, it is also easier to ensure numerical accuracy for low-energy excitations rather than for high-energy excitations. One just needs to include electronic transitions with low energy. In this example, I show how to study convergence when we calculate the absorption spectrum of the silane molecule, SiH_4.

In the working directory, file `rgwbs.in` has a specified energy cut-off in TDLDA transitions. When you run `tdlda`, you should obtain an output on screen similar to the one in file `tdlda.out_10eV`. Rename file `eigenvalues_lda` as `eigenvalues`, add an energy resolution on the first line of that file and run tool `absp`. Important output is file `across`, which should be similar to `across_10eV` in the working directory. When you plot the first two columns of the file, you see that the absorption cross section has a first peak at 8.7 eV, which is around the absorption edge of silane.

Now, increase the energy cut-off in `rgwbs.in` to 15 eV and run codes `tdlda` and `absp` just like you did in the previous step. Make sure you delete file `pol_diag.dat` so that the second run uses the new input parameter. Now, when you visualize the contents of file `across`, you see new absorption lines at high energy. They come from the transitions with energy between 10 eV and 15 eV. You can now repeat the procedure a third time with an even higher cut-off, 20 eV, or remove the cut-off altogether and see how the absorption lines change. Since there is a finite number of unoccupied orbitals in file `parsec.dat`, the spectrum is always truncated at some threshold energy. In the present example, there are missing absorption lines starting at 26 eV, which is the difference in energy between the highest unoccupied DFT orbital and the highest occupied DFT orbital. The absorption cross section without energy truncation is depicted in file `across.gif`.

Another important issue regarding the absorption spectrum of finite systems is that unbound orbitals are very sensitive to the size of the real-space domain. The energy, density of states and spatial distribution of unbound orbitals change a lot when you change the boundary sphere radius. Oscillator strengths and excitation energies are often less sensitive, but it always important to ensure convergence of the final results with respect to the boundary sphere radius.

Instead of testing convergence by visualizing the absorption cross section, one can quickly assess the converge of the TDLDA polarizability by looking at the static susceptibility and the *f*-sum rule. See Equations 8 and 10 of Tiago and Chelikowsky to know how these quantities are calculated. Both parameters are printed in standard output on every `tdlda` run. Predictably, the numeric value of the *f*-sum rule approaches its exact value as more and more transitions (and more unoccupied orbitals) are added in the calculation. Since we use pseudopotentials, there is a non-local correction missing in the sum rule, but one should expect to see the numeric sum rule converge to within a few percent of the exact sum rule, either above or below it.

9.2. Example 2: analyzing the self-energy

Working directory = `tutorial_2`.

Estimated run time on a standard linux box = 16 min.

In this example, I show how to compute the self-energy in Na_3 and how to visualize satellites in the spectral function.

You should find in the working directory the input files for codes `parsec` and `sigma`. Run those codes in sequence. You can also run `tdlda` between `parsec` and `sigma`. Nothing will change in the final results. Notice that `rgwbs.in` specifies a TDLDA cut-off but the numeric sum rule is already close to 100%. You can remove the cut-off and see how the final self-energies are affected. They should change by no more than a fraction of eV, which is typically the accuracy of GW self-energies. Also, the self-energy is calculated for only the low energy orbitals, and a large energy range is used. That is because we want to know the energy dependence of *Σ(E)* for orbitals around the HOMO and LUMO. At the end of these calculations, you should see files `sigma_up_001_0000`, `sigma_down_001_0000` and `sigma.chkpt.dat`. In `sigma_up_001_0000` and `sigma_down_001_0000`, you see that the HOMO-LUMO gap increases from 0.525 eV (DFT-LDA) to 3.342 eV (GW). The last number does not change much after we diagonalize the quasiparticle Hamiltonian, as you see in files `eqp_up_001_0000` and `eqp_down_001_0000`.

Now, run tool `chkpt_bin_asc` and visualize the data in `sigma.chkpt.dat.asc`. More specifically, we want to plot the self-energy at the HOMO, ⟨i|Σ|i⟩ with *i = 2↑*. Search for line `"Sigma - V_xc + E0 diagonal, eV, spin 1"` in that file and scroll down until you find the self-energy for *i=2* (in the example file the working directory, that self-energy starts at line number 15271). In the table you see, columns 3 and 4 are the real and imaginary parts respectively of the operator $E_{DFT} + \Sigma - V_{xc}$. When you plot those quantities as functions of energy, you see that the quasiparticle energy printed in `sigma_up_001_0000` (3.649 eV) is such that $E = E_{DFT} + \Sigma - V_{xc}$. That is how it is defined. The slope of the self-energy at *E* is around -0.33, which is a typical value. The renormalization factor is usually around 0.7 to 0.9. You can also compute the spectral function (see Aulbur *et al.*) as $A(E) = (1/\pi)\times Im\{ G(E) \}$ where $G(E) = 1/\{E - [E_{DFT} + \Sigma(E) - V_{xc}]\}$ is the Green's function evaluated at the HOMO, as a function of energy. The spectral function has a sharp peak at the quasiparticle energy, with half-width around 0.05 eV. The integral under this peak is around 0.66, very close to the renormalization factor. Aulbur *et al.* explain the meaning of this renormalization factor and what the spectral function is. The spectral function also shows satellites at energies -6.4 eV and -8.4 eV. They are extra quasiparticle modes created when a hole at the HOMO is scattered by neutral excitations (optical modes).

The width of quasiparticle peak is somewhat arbitrary, as well as the width of peaks in the imaginary part of the self-energy. They are defined by the `dynamic_energy_resolution` given as input to `sigma`. Change the values of parameters `energy_data_points`, `energy_range` and `dynamic_energy_resolution` to get a feeling of how the spectral function changes. With them, you have control over the resolution of the spectral function, and the energy range where used to calculate the spectral function. Typical profiles of the self-energy and spectral function for the HOMO are depicted in file `sigma.gif`.

9.3. Example 3: macroscopic dielectric function in solids

Working directory = `tutorial_3`.

Estimated run time on a standard linux box = 31 min.

Linear optical absorption in solids is often quantified by the imaginary part of the macroscopic dielectric function. One can compute it by first computing the inverse dielectric matrix in plane-wave representation and inverting the long-wavelength component afterward (see Hanke). Since this package does not compute the inverse dielectric matrix directly, we need to do some data analysis. In this example, I describe how to calculate the linear absorption spectrum in bulk silicon using the TDLDA (one can also do it with BSE, but that is more time consuming).

In the working directory, you see input files for codes `parsec` and `tdlda`. Run these codes using the provided input files. They are set to calculate excited states with **q**=0, with a Monkhorst-Pack grid 5×5×5 and non-commensurate shift. The non-commensurate shift is important because it improves convergence of the absorption spectrum with respect to size of the **k**-point grid. Also, notice that there is a scissors operator defined in `rgwbs.in`. The operator leaves conduction bands unchanged but shifts valence bands to lower energy, and also stretches them by 10%. The purpose of this scissors operator is that, since we know that the energy gap is underestimated, we want to blue-shift the TDLDA absorption lines so that they fall more or less where they should (strictly speaking, this is not TDLDA absorption spectrum anymore but that is a terminology issue). The scissors operator does not correct the height of peaks though. Local field effects are included.

Since this is a periodic system, you can also use a plane-wave DFT code to generate the wavefunction file. By now, you should be able to get the necessary input for a `tdlda` run. After you run `tdlda`, rename file `eigenvalues_lda` as `eigenvalues` and run `absp`. Use some energy resolution of maybe 0.1 eV to 0.5 eV. You should get file `absorp`. This file has the real and imaginary parts of the polarizability at zero photon momentum, with an additional volume factor. You obtain the macroscopic dielectric function from this polarizability by using the relationship $\varepsilon = 1 + 8\pi \underline{P}/(N_{\boldsymbol{k}}V)$, where V is the volume of the unit cell, $N_{\boldsymbol{k}}$ is the number of **k**-points (125 in this example) and $\underline{P}$ is the truncated polarizability. The factor 8π includes spin degeneracy. You should get something similar to the data shown in file `si_epsilon.gif`. Rohlfing and Louie explain what type of many-body effects are missing in this type of calculation and how to recover them.

9.4. Example 4: role of different kernels

Working directory = `tutorial_4`.

Estimated run time on a standard linux box = 60 min.

When written in the form of an eigenvalue equation (for example Equation 36 of Tiago and Chelikowsky), the BSE contains an effective Hamiltonian for electron-hole pairs. This Hamiltonian has several terms. The first one is a diagonal quasiparticle term, which is essentially the transition energy between a hole orbital (occupied quasiparticle orbital) and a quasielectron orbital (unoccupied quasiparticle orbital). We call it D. The second term is commonly referred to as "exchange kernel", K_x. That is a misnomer because it actually originates from propagation of neutral excitations in the material. It is a response of the electron system upon application of an electromagnetic field. It creates plasmon modes, which can be seen both in extended and in confined systems. The third term is usually called "direct kernel", K_d. It describes electrostatic attraction between electrons and holes. This term depends on polarizability because the surrounding medium will screen the electrostatic attraction between electron and hole. The fourth term on the left-hand side of Equation 36 of Tiago and Chelikowsky is a vertex contribution. In this example, I show how all these terms manifest themselves in the BSE absorption spectrum of silane.

As usual, run parsec in order to get a wavefunction file. Now, run `bsesolv` using as input the files `parsec.dat` and `rgwbs.in_hf` (renamed as `rgwbs.in`). In this calculation, the energies of electrons and holes are taken from DFT plus a scissors operator. The input file also has two flags: `no_exchange`, which removes the "exchange kernel" from the BSE, and `exchange_correlation hartree_fock`, which removes screening from the direct kernel. When you run `bsesolv` with this input, the low-energy excitation modes have high oscillator strength. What happens is that the optical response of the electron distribution is removed. What remains is the response from electron-hole pairs, which is strong at low energy only. You recover the strong absorption lines at high energy by removing flag `exchange_correlation hartree_fock` from input and re-running `bsesolv` (delete checkpoint file `bse_chkpt.dat` if you have it).

Another problem with this run is that the resulting excitation energies are low. The lowest excitation energy is much smaller than the HOMO-LUMO gap (10.45 eV, after applying a scissors operator). That is because electrons and holes are binding with each other very strongly. There is no screening to weaken the electric field. What we need to do is include screening. For example, we can include the TDLDA polarizability. For that, run `bsesolv` using file `rgwbs.in_tdlda` as input. Please do not forget to delete any `bse_chkpt.dat` file before anything. If you look at the output, you see that many TDLDA excitations are now being computed. In the BSE, both the "exchange kernel" and "direct kernel" are included. The resulting excitation energies, shown in file `eigenvalues_bse_tdlda` are now higher.

One can also use RPA screening instead of TDLDA. You just need to add flag `no_lda_kernel` in `rgwbs.in`. Use input file `rgwbs.in_rpa`, remove files `pol_diag.dat` and `bse_chkpt.dat`, and run `bsesolv` again. There is not much difference in the results. The BSE excitation energies are close to what TDLDA screening produced. Oscillator strengths are also similar and *f*-sum rule is about the same (the sum rule can be more than 100% if a scissors operator is used because this operator adds a non-local term in the Hamiltonian, that modifies the exact value of the *f*-sum rule). But you see that the run time is shorter. That is because matrix elements involving the LDA kernel are not being calculated anymore. This is a typical behavior.

Finally, one can remove the Tamm-Dancoff approximation from the BSE. The input files are now `parsec.dat` and `rgwbs.in_mix`. Remove any files `bse_chkpt.dat` and `pol_diag.dat`. This input uses flag `no_mixing`.

All the runs in this example were done using a scissors operator to approximate the quasiparticle energies. I chose that just to reduce runtime. For more accurate results, you can calculate quasiparticle energies from first principles, which is actually the recommended procedure. You can do this calculation by running `sigma` before `bsesolv`. These are the steps:

1. Run `PARSEC` using the input pseudopotential and file `parsec.in` in the working directory.
2. Run `sigma` using as input the file `parsec.dat` (or `wfn.dat`, for older versions of PARSEC) and the PARSEC input files.
3. Run `bsesolv` using as input these files as input: `hmat_qp`, `pol_diag.dat` plus all the input files from the previous step.

The BSE spectra obtained using files `rgwbs.in_hf`, `rgws.in_tdlda` and the calculation without scissors operator are depicted in file `bse.gif`.

9.5. Example 5: extra point group symmetries, BLIP transformation

Working directory = `tutorial_5`.

Estimated run time on a standard linux box = 14 min.

The purpose of this example is to explain how BLIP coefficients for electron wavefunctions are calculated with this package. This feature of RGWBS is not directly related to many-body calculations but it could be useful in applications where BLIP transformed wavefunctions are used instead of the actual wavefunctions.

BLIP coefficients are calculated on real space on a regular grid. Executable `w_blip` is the executable that calculates these coefficients. In the working directory, you see the input files necessary to run PARSEC and `w_blip`. The structure is the benzene molecule. File `rgwbs.in` specifies that BLIP coefficients will be calculated for all bound orbitals plus the lowest 12 unbound orbitals. Besides the usual D_{2h} point group, the point group I_h is used to classify wavefunctions. You should be able to read file `Ih` and recognize the rotation matrices that define this group, characters of irreducible representations, and representation product table.

Output of the `w_blip` code is file `bwfn.data`, which contains all BLIP coefficients for the specified orbitals. Standard output also has some information that could be useful:

- Projections of the input wavefunctions on all irreducible representations. These projections should be always zero or one, apart from loss of spatial resolution caused by the finite grid spacing.
- Kinetic energy of the input wavefunctions, calculated using BLIP coefficients. This is useful for debugging purposes.

---

10. Frequently asked questions

1. Can I calculate the polarizability using the GGA, or some non-local approximation?

   Not with this package. But if you are really serious about it, you need to calculate exchange-correlation kernels for the functional you have in mind (GGA or whatever) and implement them in the source code. Notice that most DFT software calculate the exchange-correlation functional and exchange-correlation potential (the first derivative of the functional with respect to density), but they do not normally calculate the kernel (the *second* derivative of the functional with respect to density).

2. Can I study systems with non-collinear spin, or systems where spin-orbit interactions are included at DFT level?

   Not with this package. This package was designed for two classes of systems: non-polarized systems, and spin-polarized systems where the spin orientation is decoupled from real-space coordinates.

3. How expensive is a calculation of TDLDA polarizability? How does it scale with system size?

   There are two major tasks involved in TDLDA. The first one is setting up the eigenvalue problem. That involves computing all matrix elements of the Hermitian matrix. Of course, the number of matrix elements to be computed is N_s^2 where N_s is the number of single-electron transitions (usually the product between the number of occupied orbitals and the number of unoccupied orbitals, if no energy cut-off is defined). Matrix elements involving the Coulomb potential, like the ones in Equation 6 of Tiago and Chelikowsky, scale as $N_r log(N_r)$ where N_r is the number of real-space grid points. Matrix elements involving the LDA kernel, Equation 7 of Tiago and Chelikowsky, are linear in N_r. As a very rough estimate, the calculation of matrix elements scales as N^5 with system size.

   The second task involved in TDLDA is diagonalizing the eigenvalue problem. This is a straightforward diagonalization of a dense matrix, which scales as N_s^3, or approximately N^6 with system size.

4. Can I simplify the calculation of TDLDA polarizability?

Yes. One strategy is by exploring symmetry properties. If the symmetry operations in the system form an Abelian group (which does not need to be the largest point group), then the code does not calculate matrix elements that are zero by selection rule. The scaling in calculation of matrix elements will become approximately N^5/N_g where N_g is the order of the Abelian group. Symmetries are also used in the diagonalization of the TDLDA equation: the TDLDA matrix is rewritten in block form, with one block for each irreducible representation. The diagonalization step then scales as N^6/N_g^2.

Another strategy is parallel computation. Both the calculation of matrix elements and diagonalization are fairly well parallelized. Users should see good scaling with number of processors up to the range of several hundreds of processors. Of course, each calculation has its own particularities, so "your mileage may vary".

5. How does the `sigma` and `bsesolv` codes scale with system size?

   In `sigma`, matrix elements involving electron transitions are calculated. If the system has N_s transitions, N_n DFT orbitals (used in the sum over poles of Green's function, for example Equation 24 of Tiago and Chelikowsky), and we are computing self-energies for N_i orbitals, then the calculation of matrix elements scales as $N_s \times N_n \times N_i \times N_r / N_g$. Again, the scaling is approximately N^5 with system size. The `bsesolv` code has a similar N^5 scaling.

6. Can I calculate quasiparticle energy using a desktop?

   You may be able to run isolated atoms but not much more than that. Electron self-energy of macromolecules or nanostructures with thousands of atoms have not been calculated from first principles because it is not trivial, at least for the supercomputers we have now. But confined systems with a few tens of atoms, or periodic systems with around that many atoms in the unit cell, can be studied using this package in a parallel computer with moderate size.

7. I am running `sigma` with a Hartree-Fock exchange-correlation type and I am not reproducing the Gaussian basis benchmarks. What is happening?

   These calculations use the input wavefunctions as basis. Therefore, the quality of output data is limited by the completeness of the basis.

8. I calculated the TDLDA polarizability some time ago but lost part of the input files. Now, I recalculated the wavefunction file (`parsec.dat`) but I am obtaining unreasonable self-energies. What is wrong?

   The wavefunction file defines the basis. If you recalculated wavefunctions, then they could differ from the previous ones by global phases. The extra phases destroy coherence between orbitals. A single wavefunction file should be used to run codes `tdlda`, `sigma` and `bsesolv`.

9. Can I use Hartree-Fock orbitals instead of DFT orbitals as input wavefunctions in a `sigma` calculation?

   In principle yes. You need two things: develop an appropriate interface to read in the HF data (look at the existing interfaces to codes `PARSEC` and `PARATEC`); and calculate the exchange-correlation operator in the input wavefunctions properly, and plug it in the V_{xc} registers. The exchange-correlation operator is just the Fock exchange, of course.

10. The source files are distributed over several directories. Why?

    Since this package contains different executables, the source files are correspondingly distributed over several subdirectories. This is a brief description of the contents of each subdirectory:

    - `BSE` : source files specific to `bsesolv`.
    - `SIGMA` : source files specific to `sigma`.
    - `TDLDA` : source files specific to `tdlda`.
    - `W_BLIP` : source files specific to `w_blip` (BLIP transformation).
    - `bin` : location of all executable files.
    - `shared` : source files shared by multiple executables.
    - `config` : location of machine-dependent parameter files.
    - `utils` : set of post-processing tools and scripts for specialized self-consistent schemes. Each file has its own inline documentation.
    - `example_*` : subdirectories with example runs. Users who are building this package on a new machine and/or want to get acquainted with this package are advised to try and reproduce some of these runs.

11. How to build interfaces with other DFT codes?

    Look at source files for the existing interfaces. This is a brief description:

    - `parsec_wfn` : reads wavefunctions on a regular real-space grid and auxiliary data: energy eigenvalues, specification of the grid, numerical parameters used in the DFT calculation.
    - `parsec_atoms` : reads atom coordinates from file `parsec.in`.
    - `parsec_xc` : reads specification of the exchange-correlation functional from `parsec.in`.
    - `paratec_wfn` : reads wavefunctions on a plane-wave basis, energy eigenvalues and various numerical parameters (see source file).
    - `paratec_atoms` : reads atom coordinates from file `input`.
    - `paratec_xc` : reads specification of the exchange-correlation functional from `input`.

    In addition, you may need to look at subroutine `read_psp` in module `psp_module.F90z`. This subroutine reads the pseudopotential files. You need to modify that module if you wish to define a new pseudopotential format.

12. How are the RGWBS codes parallelized?

    RGWBS uses multilayer parallelization. In the outermost layer, processing units are distributed in "representation groups" (no relationship with irreducible representations of point groups) which I call *rgp*. Each *rgp* has the same number of processors and is assigned a set of irreducible representations in such a way that each *rgp* has the same number of representations. That ensures load balance. The number of representation groups is controlled by input flag `distribute_representations`. Grid points are distributed in the next layer of parallelization. In that layer, the processors belonging to each *rgp* group are distributed in "wavefunction groups", *wgp*. Again, all *wgp* groups have the same number of processors. The number of *wgp* groups is controlled by input option `distribute_wavefunctions`. Each *wgp* group is assigned a number of grid points, which is not necessarily the same for all groups. Internally, all these groups of processors are handled with the use of MPI groups. Module `mpi_module.F90z` contains the definitions of groups, layouts and most communication-related functions. Currently, there is no parallelization over k-points or q-points. Most input/output is done by one processor only.

13. What are those files with extension .F90z?

    They are pre-processable Fortran 95 source files, written in such a way that they can generate source codes with and without support for complex algebra. Capital letter "Z" has special meaning in those files. File `mycomplex.h` has the replacement rules. Since operator overloading is not very advanced in Fortran, sometimes programmers are forced to build unusual workarounds.

References

R. W. Martin, *Electronic structure : basic theory and practical methods* , Cambridge University Press (Cambridge, UK, 2004).

W.G. Aulbur and L. Jonsson and J.W. Wilkins, *Quasiparticle calculations in solids* , in Solid State Physics (ed. F. Seitz, D. Turnbull, and H. Ehrenreich, Academic Press, New York, USA) **54**, page 1 (2000).

M. L. Tiago and J. R. Chelikowsky, *Optical excitations in organic molecules, clusters, and defects studied by first-principles Green's function methods* , Phys. Rev. B **73**, 205334 (2006).

G. Onida, R. Reining, R. W. Godby, R. Del Sole and W.Andreoni, *Ab Initio Calculations of the Quasiparticle and Absorption Spectra of Clusters: The Sodium Tetramer* , Phys. Rev. Lett. **75**, 818 (1995).

I. Vasiliev, S. Ogut and J. R. Chelikowsky, *First-principles density-functional calculations for optical spectra of clusters and nanocrystals* , Phys. Rev. B **65**, 115416 (2002).

M. E. Casida, in *Recent Advances in Density-Functional Methods, Part I*, page 155 (editor D. P. Chong, World Scientific, Singapore, 1995).

M. S. Hybertsen and S. G. Louie, *Electron correlation in semiconductor and insulators: Band gaps and quasiparticle energies*, Phys. Rev. B **34**, 5390 (1986).

G. Onida, L. Reining and A. Rubio, *Electronic excitations: density-functional versus many-body Green's-function approaches*, Rev. Mod. Phys. **74**, 601 (2002).

M. Rohlfing and S. G. Louie, *Electron-hole excitations and optical spectra from first principles*, Phys. Rev. B **62**, 4927 (2000).

W. Hanke, *Dielectric theory of elementary excitations in crystals* , Adv. Phys. **27**, 287 (1978).

Author of this manual: Murilo Tiago .

Last modification: Monday, March 16, 2009.