GWL Manual

September 30, 2013

1 What is GWL

GWL is a code for performing first-principles GW calculations according, at the moment, to the G_0W_0 approximation[1]. It is based on plane-waves for representing wave-functions and on pseudopotentials for describing the interactions between valence and core electrons. At the moment, only norm-conserving pseudo-potentials are supported.

GWL is distributed, along the GNU-GPL license, as a package of the Quantum-Espresso (*www.quantum-espresso.org*) suite (QE) of ab-initio codes. GWL can be downloaded with the development version of the Quantum-Espresso (*www.qe-forge.org*) or through the GWL website (*www.gwl-code.org*).

GWL requires the use of the *pw.x* code of the QE and requires the installation of the *ph.x* package.

GWL avoids in the GW formalism any explicit sum over empty Kohn-Sham states yielding in this way converged results[2]. Moreover, for specifying the basis set used for representing polarizability operators, two choices are implemented: either the use of a conventional plane-waves basis or the use of an *optimal basis*. As this is smaller, a significant speed-up can be achieved, especially for non-homogenous systems, without loosing accuracy[2, 3]. With GWL calculations are performed on the imaginary energy axis. Then, the expectation values of the self-energy operators are analytically continued on the real energy axis through multipole expansion. In this way, the plasmon-pole approximation is avoided.

GWL supports hybrid (exact-exchange) exchange and correlation functionals.

As the target systems are (large) non-homogeneous systems, only Γ-point sampling of the Brillouin's zone has been implemented. In the case of isolated systems, a truncated Coulomb interaction can be used. In the more general case of extended systems, head and wings of the symmetric dielectric matrix are calculated. Only for this calculation general **k**-point sampling of the Brillouin's zone is supported together with a treatment of the point symmetries of the primitive crystal cell.

In the development version of GWL other features have been already implemented and will be released soon after having completed the test phase. These

include the use of contour integration instead of analytic continuation, a scheme for the approximate treatment of semicore orbitals[4], evaluation of off-diagonal elements of the self-energy operators, some levels of self-consistency.

2 How to install GWL

The GWL code can be download with the development version of the Quantum-Espresso through *svn* from the *www.qe-forge.org* website. It is easily installed as the other programs of the package.The GWL programs will be installed in the subdirectory GWW of the main Quantum-Espresso directory. Just digit "*make gwl*" in the main Quantum-Espresso directory.

The GWL code is written in fortran 90 and is parallelized using the MPI protocol. It uses intensively BLAS (mainly DGEMM) and LAPACK calls so it is strongly suggested to make use of external proprietary libraries appropriate for the chose computer architecture. Such libraries are the same used by the Quantum-Espresso as the same *make.sys* file is used. Note that using internal BLAS libraries, compiled by the user, can lead to a strong deterioration of the performances.

As BLAS and LAPACK libraries are also available in versions supporting the OPENMP parallelization protocol, we suggest to compile the GWL (and the QE) package enabling the mixed MPI/OPENMP parallelization. With this choice, we have reach good, almost linear, scaling up to 2048 computing cores.

3 How to run a GW calculation with GWL

A GW calculation will require at least three or four different runs. The simplest case is when a truncated coulomb interaction is used. First, a DFT self-consistent calculation with the *pw.x* code must be performed. In this case a k-point sampling of the Brillouin's zone can be used. Second, a DFT non self-consistent calculation must be run specifying the total number of Kohn-Sham states for which we want to calculate the GW quasi-particle energies. In this case Γ-only sampling is required in order to impose the KS wave-functions to be real. In the third step, the *pw4gww.x* code of the GWL package is used in order to obtain the polarizability basis, the Lanczos'chain terms and all the other quantities needed by the GW calculation. This is then performed using the *gww.x* code of the GWL package which will required to define a grid on the imaginary frequency axis and on the imaginary time axis. The *gww.x* code will calculate the expectation values of the correlation part of the self-energy operator and write them on files, named re_on_imXXXXX and im_on_imXXXXX for the real and imaginary part, respectively (with XXXXX the number of the corresponding Kohn-Sham state), which can be plotted. The *gww.x* code performs also fits of such lines with a multipole expression in order to analytically continue the expectation values on the real frequency axis for obtaining the quasi-particle energies. Such fits can be also performed again later using the *gww_fit.x* code.

This code just requires the re_on_imXXXXX and im_on_imXXXXX files together with the DFT eigenenergies and the exchange energies. Therefore, it needs few computational resources (e.g. one laptop).

It should be noted that the non-self consistent DFT calculation with the *pw.x* code could be avoided if we specify all the KS states of interest in the scf DFT calculation and if we use for this Γ-only sampling of the Brillouin's zone.

When, we want to perform a calculation on an extended system a slightly more involved formalism must be used. This requires the calculation of the head (element $\mathbf{G} = 0, \mathbf{G} = 0$) and wings (elements $\mathbf{G} = 0, \mathbf{G} \neq 0$) of the symmetric dielectric matrix. The *head.x* code of the GWL package fulfills this purpose. It requires k-point sampling of the Brillouin's zone and can take advantage of the point symmetries of the model structure. We must use the same grid on imaginary frequency that will be then used in the *gww.x* code. The computational cost of this calculation is smaller than those with the *pw4gww.x* and *gww.x* codes.

The *pw4gww.x* and *gww.x* codes offer several possibility of restart. In particular, the self_energies can be calculated by groups of KS states re-running first the *pw4gww.x* code and then the *gww.x* code. In this case, we have introduced an option (called *l_big_system* in both cases) which permits to have higher flexibility in the choice of the parameters involved in such restart (e.g. length of Lanczos chains).

4 pw4gww.x

The *pw4gww.x* code prepares all the required data (such as Wannier's transform matrices, Lanczos's chains, exchange terms,..) for the GW calculation that will be performed with the *gww.x* code. It uses the same environment than the *Quantum-Espresso* and its is parallelized in the same way on the plane-waves. It also uses a good number of subroutines from the QE and in particular from the *pw.x* code. At the moment parallelizations other than on plane-waves are not supported.

The code is structured in this way: the main routine *pw4gww.f90* reads the input namelist and then calls the subroutine *produce_wannier_gamma.f90*. This calls all the subroutines which calculate all the required data for the *gww.x* code. These are divided in several point and the input parameter *restart_gww* determines the starting point, from 0 to 4 . For the moment no ending point can be specify. In point 2, the Lanczos chains are calculated for the polarizability and the self-energy. This point is further divided in 4 sub-points, from 0 to 3. The starting one is specified by the input parameter *lanczos_restart*.

These are the tasks performed by each point, of *produce_wannier_gamma.f90,* given by the corresponding *restart_gww* number:

- *restart_gww=0* : calculate Wannier's transform, expectation values of DFT exchange and correlation potential
- *restart_gww=1*: calculate the polarizability basis

- *restart_gww=2*: calculate the Lanczos chains
- *restart_gww=*4: calculate the projections of the wings of the symmetric dielectric matrix on the polarizability basis vectors (if required) and calculate the expectation values of the bare exchange operator.

The point 2 is divided in 4 sub-points; the starting one can be specified by the input option *lanczos_restart*:

- *lanczos_restart=0*: calculate the local t vectors for the calculation of the polarizability
- *lanczos_restart=1*: calculate the global t vectors and the Lanczos chains for the calculation of the polarizability
- *lanczos_restart=*2: calculate the local s vectors for the calculation of the self-energy
- *lanczos_restart=*3: calculate the global s vectors and the Lanczos chains for the calculation of the self-energy

While a calculation is proceeding, the code reports on output, after having completed a point, the *restart_gww* and the *lanczos_restart* parameters which could be used for restarting the calculation from that point. As the point given by *restart_gww=*1 (calculation of the polarizability basis) can be quite time-consuming, we have implemented the possibility of resuming a partially completed calculation. This is done automatically by reading the tiny text file XYZ.restart_fk0_status (where XYZ is the QE *prefix* parameter which specifies the name of the calculation). If this cannot be found or if it starts with "-1" the program will start from beginning. If point 1 is completed a "-1" is written in order not to affect newer calculations. If a calculation has not been completed and if you want to discard the data obtained you must delete this file.

The code is parallelized using the MPI protocol over the **G** points in the same way as the *pw.x* code. Note that for small systems it would be useless to use a large number of MPI tasks as some task would have no **G** points at all resulting even in longer calculations. The *pw4gww.x* strongly relies on linear algebra operation which are performed using the BLAS and LAPACK libraries. As usually such libraries can efficiently take advantage of the OPENMP parallelization, a good choice (especially for large systems) is to use a mixed MPI-OPENMP parallelization, distributing each MPI task over a number of OPENMP threads. Only in the routines *pola_basis_lanczos* and *self_basis_lanczos* ,which calculate the *local* t and s vectors, matrix diagonalization are further distributed over a certain number of MPI tasks. We plan in a future version of the code to develop further such strategy of MPI parallelism.

The main contribution to memory usage is due to the storage of the polarizability basis and in point 1 to the storage of the vectors used for calculating the polarizability basis.

4.1 Input parameters

Required parameters:

prefix (character*) Title of the calculation, must be the same as that used in *pw.x* .

l_truncated_coulomb (logical) if .true. truncates the coulomb interaction at a distance R (see below), if .false. a calculation with *head.x* must have been performed before calling *pw4gww.x* (default: .false.)

truncation_radius (real*8) the truncation radius R (see above) in Bohr.

pmat_type (integer) Type of basis for the polarizability matrices, at the moment the available options in the released version of the code are: 5 for a plane-waves basis set and 4 for the optimal polarizability basis (default: 4)

pmat_cutoff (real*8) If pmat_type=5 it specifies the cutoff in Rydberg for the plane-waves, if pmat_type=4 it specifies the cutoff E^* (default: 3.d0)

numw_prod (integer) In pmat_type=5 gives the length of the polarizability basis

num_nbndv (integer(2)) number of occupied states (single or double) for each spin channel. In the case of spin unpolarized calculations only num_nbndv(1) is required

num_nbnds (integer) Total number of DFT states for both spin channels for which GW correction will be calculated, must be the same as the parameter nbnd in *pw.x*

Optional parameters:

restart_gww (integer) restart point see above (default: 0)

restart_lanczos (integer) restart sub-point see above (default: 0)

s_pmat (real*8) threshold for the calculation of the temporary basis from which the actual polarizability basis is obtained (default: 0.01d0)

dual_pb (real*8) defines the ratio between the cutoff used for obtaining grids in real spaces with respect to the cutoff use for representing wave-functions in **G** space. It must be $\leqslant 4$. It affects the calculation of the polarizability basis. (default: 1.d0)

dual_vt (real*8) defines the ratio between the cutoff used for obtaining grids in real spaces with respect to the cutoff use for representing wave-functions in **G** space. It must be $\leqslant 4$. It affects the calculation of the **t** vectors. (default: 1.d0)

dual_vs (real*8) defines the ratio between the cutoff used for obtaining grids in real spaces with respect to the cutoff use for representing wave-functions in **G** space. It must be $\leqslant 4$. It affects the calculation of the **s** vectors. (default: 1.d0)

n_pola_lanczos (integer) dimension of the local **t** vectors for the calculation of the irreducible polarizability matrices (default: 400)

s_pola_lanczos (real*8) threshold for obtaining from the local **t** vectors the global **t** vectors . Note it has dimension Bohr^3 and for a given choice of the polarizability basis parameters it scales as the volume of the simulation cell. (default: 0.5d0).

nsteps_lanczos_pola (integer) length of Lanczos chains for the calculation of the irreducible polarizability matrices (default: 20)

n_self_lanczos (integer) dimension of the local **s** vectors for the calculation of the expectation values of the self-energy (default: 600)

s_self_lanczos (real*8) threshold for obtaining from the local *s* vectors the global *s* vectors. (default: 1.d-12). Note for larger systems a value of 1.d-13 is more adequate

nsteps_lanczos_self (integer) length of Lanczos chains for the calculation of the irreducible polarizability matrices (default: 40) Note values of 100 - 200 give converged results in large systems.

l_big_system (logical) in large systems can be convenient to calculate the self-energy separately each state at one time, this corresponds to several calculations where in each one the global **s** vectors will coincide with the local one. Although obtaining the whole spectrum with this strategy, in this way it is easier to check the convergence with respect to n_self_lanczos and nsteps_lanczos_self (s_self_lanczos will not be use). It is also the best choice if only few states must be calculated. (default: .false.)

s_first_state (integer) if l_big_system==.true. self_energy will be calculated only from_s_first_state till s_last_state (default: 1)

s_last_state (integer) if l_big_system==.true. self_energy will be calculated only from_s_first_state till s_last_state (default: num_nbnds)

l_verbose (logical) if .true. a much larger output file is produced. Useful for debugging (default: .false.)

4.2 Output file

We describe the output file, which is directed on standard output, according to the points specified by the parameter restart_gww.

4.2.1 Point 0

- First the parameters of the simulation cell and of the pseudopotentials are reported as for the *pw.x* code.
- **KS energy: I X** DFT Kohn-Sham energy X in eV for the I-th state
- **energies_xc : I X** DFT exchange and correlation contribution in eV to the KS energy of the I-th state
- **energies_h: I X** DFT Hartree contribution in eV to the KS energy of the I-th state
- **LOCALIZING WANNIER FUNCTIONS:** now the unitary transform for obtaining the Wannier's functions, of the valence manifold only, is computed
- **Spread X1 X2** during successive optimization steps a quantity which is proportional to the inverse of the spread is printed
- **Center Wannier: X1 X2 X3** coordinates of the center of the i-th Wannier's function, only for valence states, in Bohr

4.2.2 Point 1

- **Number of projected orthonormalized plane waves: I** total number of plane-waves defined by E^* , parameter pmat_cutoff, is printed
- **FK state: I** the algorithm for obtaining the *temporary* polarizability basis is iterative. Now the product terms relative to the I-th valence states are computed
- **FK GS J** at this iteration J vectors have been added to the basis
- **POLARIZABILITY eigen: I X** the eigenvalues of the polarizability-like operator defining the polarizability basis are reported in ascending order in units of Bohr^2

4.2.3 Point 2

- **USE RESTART: 2 LANCZOS RESTART:0**
- **EIGEN T LOCAL: I 1 X** for the i-th Wannier's function (states are distributed among MPI tasks) report the n_pola_lanczos lowest eigenvalue X of the overlaps of $P_c \left|\phi\mu w_v\right\rangle$
- **EIGEN T LOCAL: I N_POLA_LANCZOS X** for the i-th Wanier's function (states are distributed among MPI tasks) report the largest eigenvalue X of the overlaps of $P_c \left|\phi_\mu w_v\right\rangle$
- **USE RESTART: 2 LANCZOS_RESTART:1**

- **EIGEN GLOBAL: 1 X** the global basis for the calculation of the polarizability is now being constructed iteratively step by step, for each state it is reported the smallest and the largest eigenvalue, only terms with eigenvalue larger than S_POLA_LANCZOS will be taken
- **TOTAL NUMBER OF GLOBAL T VECTORS: N** the final dimension of the basis is N
- **lanczos_state:** now the lanczos chains are being built , it will write this for every MPI task
- **USE RESTART: 2 LANCZOS_RESTART:2**
- **EIGEN S LOCAL: I 1 X** for the i-th KS state (states are distributed among MPI tasks) report the n_self_lanczos lowest eigenvalue X of the overlaps of $|(v\phi_\mu)\,\psi_i\rangle$
- **EIGEN S LOCAL: I N_SELF_LANCZOS X** for the i-th KS state (states are distributed among MPI tasks) report the largest eigenvalue X of the overlaps of $|(v\phi_\mu)\,\psi_i\rangle$
- **USE RESTART: 2 LANCZOS_RESTART:3**
- **EIGEN GLOBAL: 1 X** the global basis for the calculation of the self-energy is now being constructed iteratively step by step, for each state it is reported the smallest and the largest eigenvalue, only terms with eigenvalue larger than S_SELF_LANCZOS will be taken
- **TOTAL NUMBER OF GLOBAL S VECTORS: N** the final dimension of the basis is N
- **lanczos_state:** now the Lanczos chains are being built , it will write this for every MPI task

4.2.4 Point 3

- note that to perform the calculation of this point the parameter lanczos_restart must be different from 2 or 3
- **Routine calculate_wing** the projections of the polarizability basis on the wings of the symmetric dielectric matrix are computed
- **Exchange energy I IS X** it reports the expectation value X of the Fock operator for the I-th KS-state of the spin-channel IS

4.2.5 Point 4 -only in development version

- prepares the data for the semicore treatment during the calculation with semicore states in valence, active only if l_semicore = .true.

4.2.6 Point 5-only in development version

- prepares the data for the semicore treatment during the calculation without semicore states in valence, active only if l_semicore_read = .true.

5 gww.x

The actual GW calculation is then performed by the gww.x code. This code reads the input from standard input in the form of a namelist which must have the following form:

&inputgww
ggwin%name_of_parameter=its_value
/

The code requires the definition of a grid on the positive imaginary frequency half-axis. Two possible kinds of discretization have been implemented. The simplest choice,**ggwin%grid_freq=3**, is an equally spaced grid consisting of **ggwin%n** grid points plus the point at zero frequency and going up to the maximum frequency **ggwin%omega** which is given in Rydberg. However, it is more convenient to use grids which are denser close in the low frequency region. We achieve this with the choice **ggwin%grid_freq=5** in this case the final grid will contain all the **ggwin%n+1** points of the equally spaced grid up to the frequency **ggwin%omega ggwin%omega** plus other grid points: for the first **ggwin%second_grid_i**+1 first grid points of the equally spaced grid we add a finer equally spaced mesh in such a way that we have other **ggwin%second_grid_n** points for for each half-grid step of the loose grid around the chosen **ggwin%second_grid_i**+1 points. See Figure 1.

Typical Parameters for grids can be found in the example and tutorial files.

The GW calculation is composed by a series of steps: 1 to 7. With the **ggwin%starting_point** we give the first step and with the option **ggwin%ending_point** we give the last step to be performed. It should be noted that if the calculation is restarted, in general, the number of MPI tasks can be changed. The only exception is the calculation of the irreducible polarizability matrices in point 3, where a file named *restart_polaw* is created, if it exists and if it does not begin with '-1', the same number of MPI tasks must be used if the calculation of the polarizability matrices is restarted. As, once the polarizability matrices are finally obtained, this file is set to '-1' , it must be deleted only when we want to discard a partially completed calculation.

The code is parallelized on the total number of grid points on the imaginary frequency positive half-axis.

The GW calculation is performed through the following series of steps:

1. initialize the Green's function
2. calculate and write on disk, partial sums to be used in point 3
3. calculate the irreducible polarizability matrices

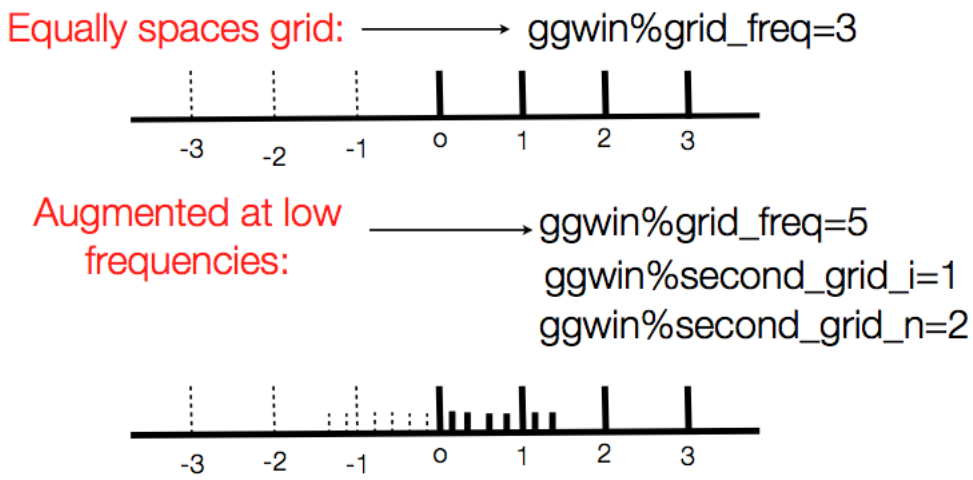

Figure 1: Grid points on the imaginary frequency axis in the gww.x code for the equally spaced grid (3) and the augmented grid (5). In the calculations only the grid points on the positive half-axis are considered. The points on the negative axis are reported (dashed lines) only for reference.

4. from the irreducible polarizability calculates the screened interaction
5. from the screened interaction calculate the reducible polarizability
6. Calculate the expectation values of the self-energy operator on the imaginary frequency axis
7. Fit the the expectation values of the self-energy operator with a multipole formula and calculate quasi-particle energy levels

5.1 Input parameters

Required parameters

ggwin%prefix (character*) Title of the calculation, must be the same as that used in *pw4gww.x* .

ggwin%max_i (integer) total number of KS bands

ggwin%l_truncated_coulomb (logical) must be equal to the one chosen for *pw4gww.x*

ggwin%omega (double) extension of frequency grid on imaginary frequency axis (Ry)

ggwin%n (integer) number of grid steps (see previous section)

ggwin%grid_freq (integer) type of frequency grid (see previous section)

ggwin%omega_fit (double) extension of frequency grid on imaginary frequency axis (Ry) for evaluating the self-energy

ggwin%n_grid_fit (integer) number of equally spaced grid-steps for the evaluation of the self-energy

ggwin%n_fit (integer) number of grid points (starting from the origin) to be used for fitting the multipole expansion

if ggwin%grid_freq==5 we must give also

ggwin%second_grid_i (integer) number of grid steps close to the origin which requires interpolation (see previous section)

ggwin%second_grid_n (integer) number of interpolation points (see previous section)

ggwin%i_min (integer) KS energies will be calculated from state i_min to state i_max (only for steps 6 and 7)

ggwin%i_max (integer) KS energies will be calculated from state i_min to state i_max (only for steps 6 and 7)

Optional parameters

ggwin%n_multipoles (integer) number of poles for multipole expansion (default: 2)

ggwin%fit_thres (double) threshold for convergence of minpack non-linear fit (default: 1d-5)

ggwin%offset_fit (integer) it specifies the number of point on the fit grid which will be skipped during the fit (starting from the origin). (default: 1)

ggwin%n_set_pola (integer) Number of valence states to be treated together when calculating the irreducible polarizability in point 3. It affects only the performance of the code not its results. Larger n_set_pola faster the calculation but larger the memory consumption (see next Section).(default: 4)

ggwin%l_verbose (logical) enables more detailed output file, useful for debugging (default: .false.)

ggwin%l_big_system (logical) calculates the expectation values of the self-energy state by state using local s vectors, safer for large systems as it is easier to check the convergence. It requires the same option in the *pw4gww.x* code. (default: .false.)

ggwin%l_list (logical) similar to l_big_system but calculates only the states specified by a list (default: .false.)

Optional parameters enabling features still under development or testing

ggwin%whole_s (logical) calculates also off_diagonal elements of the self-energy operator (default: .false.)

ggwin%l_frac_occ (logical) if true enables the use of fractional occupancies (default:.false.)

ggwin%l_semicore (logical) enables the scheme for approximate account of semicore states (default: .false.)

ggwin%n_real_axis (integer) for calculations with the contour deformation scheme , the number of points on the real energy axis at which the self-energy will be calculated (default: 0)

ggwin%real_energy_min (double) defines the range on the real energy axis for contour deformation calculations, in Ry, (default: -1.d0)

ggwin%real_energy_max (double) defines the range on the real energy axis for contour deformation calculations, in Ry, (default: 1.d0)

5.2 Parallelization and memory usage

The *gww*.x code is parallelized on the grid spanning the imaginary energy half-axis. Therefore it can take advantage of MPI parallelization till a number of MPI tasks equal to the maximum number of grid points on the imaginary energy positive axis (usually from 128 to ca. 256-300). When dealing with machines with a larger number of computing cores it is a good choice to use a mixed OPENMP/MPI parallelization. This can be achieved simply by enabling OPENMP parallelism when compiling. See QE manuals.

The *gww.x* code needs a quite large scratch disk space. Note that each MPI task writes files only in the default starting directory.Nevertheless it is still possible to use local scratch disks (where each MPI task reads/writes on its own scratch disk unreachable by the others tasks) if: we don't change the number of MPI tasks during restarts and we distribute by hand (i.e. just copying the files) the files *q_lanczos** which are created at point 2 to every scratch disk.

The point 3 where the irreducible polarizability matrices are calculates is usually the most time and memory consuming. In order to lower the memory usage the code keeps in memory **ggwin%n_set_pola** occupied KS states at each time. There is a loop on occupied KS states. For best performance we have to select the higher value of **ggwin%n_set_pola** compatible with the memory available to each MPI task. In order to determine the memory requirement for each MPI task for a chosen **ggwin%n_set_pola** , you can use tje *memory.f90* program in the *extra* directory. It will ask: the dimension of the polarizability basis **numpw** , the number of Lanczos chain steps **nsteps_lanczos_pola**,

the number of MPI tasks, and the number of *global t* vectors. This number is reported in the output file of the *pw4gww.x* code, see Sec. 4.2.

The point 3 can also be restarted. For this reason the code creates a 2 line text file named *restart_polaw* which specifies which part of the polarizability matrices (files *polaw**) have been already computed. If this file does not exist or it starts with "-1" the program starts from the beginning. When the point 3 is completed it sets this file to "-1" so we have not to delete it for future calculations. Note that restating in point 3 requires not to change the number of MPI tasks. Note that if we want to abort a calculation in point 3 we have to delete the *restart_polaw* file.

5.3 Output file

We describe the output file, which is directed on standard output, according to the points as specified by the parameter **ggwin%starting_point** and **ggwin%ending_point**.

5.3.1 Point 1

- It reports some input parameters
- **freq** points of the energy grid
- **weight** the corresponding weight for integration

5.3.2 Point 2

- **Routine calculate_compact_pola_lanczos** here calculate the *q_lanczos** files

5.3.3 Point 3

- **RESTART FROM POINT 3** you can restart from point 3
- **Lanczos dimensions Nt Ns** we have **Nt** global t vectors and **Ns** lanczos steps
- **do_polarization_lanczos1 I** the codes is calculating the **I**-th irreducible polarizability matrix
- **do_polarization_lanczos iv I** now the ggwin%n_set_pola valence states starting from **I** are processed

5.3.4 Point 4

- **Call go_dressed_w** now calculates the screened Coulomb interaction (see theory notes)
- **Read polaw Np** the dimension of the polarizability basis is **Np**

- **call calculate_w I** the codes is calculating the **I**-th screened Coulomb interaction matrix

5.3.5 Point 5

- **Transform W to Pgreek** now calculates the irreducible polarizability matrices

5.3.6 Point 6

- **RESTART FROM POINT 6** you can restart from point 6
- **Lanczos dimensions Ns Nss** it reports the number of s vectors **Ns** and the number of Lanczos steps for the self-energy **Nss**
- **Fourier transform Pgreek** transform the reducible polarizability to imaginary time
- **Loop on KS: I IS** now calculates the **I** KS states of spin **IS**

5.3.7 Point 7

- **Call fit_multipole** for each KS states between **ggwin%i_min** and **ggwin%i_max** now fits the expectation values of the self-energy on the imaginary half-axis with a multipole expansion, first uses a random search
- **Calling minpack** now uses the minpack library for performing the non-linear fit
- **Chi0 initial: X** initial error
- **Minpack fit chi0 X** final error
- **FIT state : I IS** now reports the fitted parameters for KS state **I** of spin **IS**
- **FIT a_0: X**
- **FIT a: J** for J=1,number of poles a and b parameters
- **FIT b: J**
- **QUASI-PARTICLES ENERGIES IN Ev, Spin: IS** quasi-particle energies for spin **IS**
- **State: I DFT : X1 GW-PERT : X2 GW : X3 HF-pert : X4** for the KS state **I**: **X1** is the DFT energy, **X3** is the GW energy, **X4** is the perturbative Hartee-Fock energy (i.e. with DFT orbitals) and **X2** is reported just for comparison with GW codes which solve perturbatively the final self-consistent one variable equation.All the energies are in eV.

- **IMAGINARY ENERGIES IN Ev:** in the following it reports the calculated imaginary parts of the GW quasi-particle energies for each KS state

5.4 Output files

The *gww.x* code will produce at point 7 some additional output files: the file *bands.dat* and for each KS statesXXX in thebetween **ggwin%i_min** and **ggwin%i_max** a couple of files *re_on_imXXX* and *im_on_imXXX*. The *band.x* file is a list with a row for all the KS states containing:

- **I X1 X2 X3 X4** for the KS state **I**: **X1** is the DFT energy, **X3** is the GW energy, **X4** is the perturbative Hartee-Fock energy (i.e. with DFT orbitals) and **X2** is reported just for comparison with GW codes which solve perturbatively the final self-consistent one variable equation.All the energies are in eV.

The *re_on_imXXX* and *im_on_imXXX* files contain the expectation values of the correlation part of the self-energy operator for the XXX KS-state. The *re_on_imXXX* reports the real part and the *im_on_imXXX* the imaginary one. The format is:

- **E X1 X2 X3** where **E** is the frequency either on the imaginary or the real axis, **X1** is the fitted self-energy on imaginary frequency (note that only the positive half-axis has been fitted), **X2** is the calculated self-energy on imaginary frequency, **X3** is the fitted self-energy on real frequency. All quantities are in Ry.

6 head

The *head.x* program calculates the long range terms of the symmetric dielectric matrix which are required when calculations for extended systems are performed (using the option **l_truncated_coulomb=.false.**). The code uses a Lanczos chain algorithm for avoiding sums over empty KS orbitals and requires in input the same grid on imaginary frequency which will be then used in the *gww.x* code. The code MUST follow a *pw.x* ground-state DFT calculation performed with a k-points mesh. The K_POINTS part of the input must be specified. The code is very similar to the *ph.x* one and also the input file is similar.

6.1 Input file

The input file has the form

&inputph
trans=.false.
l_head=.true.
tr2_ph=1.d-4,

niter_ph=1
outdir='./'
.....
/
0.0 0.0 0.0
containing the following parameters:

prefix=(character*) Title of the calculation, must be the same as that used in *pw.x* .

omega_gauss= same as **omega** in *gww.x*

n_gauss same as **n** in *gww.x*

grid_type= same as **grid_type** in *gww.x*

second_grid_i= same as **second_grid_i** in *gww.x*

second_grid_n=same as **second_grid_n** in *gww.x*

nsteps_lanczos=(integer) number of Lanczos steps (suggested value: 30)

6.2 Output file

First the common output of Quantum-Espresso post-processing programs appears (data of the system, cutoffs, grids).

Freq I X for every **I** step on the imaginary frequency positive half-axis : **X** the frequency

Dielectric constant in cartesian axis reports all the DFT-RPA dielectric tensors for all the frequencies starting from 0. Ry; note that local-fields effects are not included.

7 gww_fit.x

The final analytic continuation of the expectation values of the self-energy, which is achieved through fitting with a multipole expansion, must be checked in order to determine the accuracy of our calculation. Thus, it is important to investigate how the GW results change upon changes in the parameters which control the fits.

The *gww_fit.x* permits to do this as a post-processing. It requires the *re_on_im_XXX* and *im_on_im_XXX* files relative to the states XXX for which the fit is desired. It requires also the file *bands.dat* which is also produced by the *gww.x* program and contains the data regarding the energy levels. Note that the first two lines of this files MUST be removed and replaced with a single line where it is written the number of valence states. All these files are simple text files, no other file is required. As the computational load is low, the *gww_fit.x* program can be used easily on an ordinary laptop.

The input file of *gww_fit.x* is the same of *gww.x*. However, only the following parameters regarding the fit with a multipole expansion are meaningful:

ggwin%i_min (integer) KS energies will be re-calculated from state i_min to state i_max

ggwin%i_max (integer) KS energies will be re-calculated from state i_min to state i_max

ggwin%n_multipoles (integer) number of poles for multipole expansion (default: 2)

ggwin%fit_thres (double) threshold for convergence of minpack non-linear fit (default: 1d-5)

ggwin%offset_fit (integer) it specifies the number of point on the fit grid which will be skipped (starting from the origin). (default: 1)

ggwin%n_fit (integer) number of grid points (starting from the origin) to be used for fitting the multipole expansion

We suggest first to find the most stable parameters for example investigating how the HOMO and LUMO energies change. Then, the best strategy, is to find the parameters which give results (e.g. for the HOMO and the LUMO) closer to those from a calculation with the contour deformation scheme.

8 Important notes on convergence

The choice of appropriate input parameters for the *pw4gww.x* code can be quite a difficult task. Indeed, the degree of accuracy can be determined only *a posteriori*. However, for the same class of systems the most relevant parameters can be scaled with respect to the dimensions of the structural models. Input parameters must be given for the selection of: the polarizability basis, for the calculation of the polarizability matrices, and for the calculation of the self-energy. We see now in details these three parts:

8.1 Polarizability basis:

the *optimal* polarizability basis (**pmat_type=4**) is controlled by the cutoff E^* (**pmat_cutoff**) and by the choice of the total basis dimension (**numw_prod**). So that the corresponding threshold q^* can only be determined after the calculation viewing the output file. **numw_prod** scales linearly with respect to the dimensions of the system (e.g. total number of atoms, or total number of electrons). As the *optimal* polarizability basis is obtained in two steps and in the first a temporary, larger, basis is derived, we have to check that the dimension of such a basis is large enough. We have seen that a factor ~ 4 guarantees convergence. The dimensions of the temporary polarizability basis is determined by the threshold **s_pmat** which is almost system independent.

8.2 Polarizability: t vectors

The construction of the basis (t vectors) which is used to calculate the actual polarizability matrices usually is not critical. The parameter **n_pola_lanczos**

sets the number of *local* t vectors. As these are obtained from the overlaps of localized (Wannier's) functions, such parameter is essentially system independent. Then the final *global* t vectors, are obtained from all the different local t basis using the threshold **s_pola_lanczos**. We have verified that a total number of *global* t vectors of $\sim$ 4 times the dimension of the polarizability basis guarantees convergence. **s_pola_lanczos** scales linearly with the dimension of the system. Then for all the *global* t vectors Lanczos chains are calculated. The length of such chains is given by **nsteps_lanczos_pola**. A value of 20 always gave converged results.

8.3 Self-energy: s vectors

Although analogous to the case of polarizability t vectors, the construction of the basis used for calculating the actual self-energy expectation values which we refer to as s vectors is more critical. Also in this case we (usually) proceed in two steps: first we build a *local* basis of s vectors relative to each single KS state and then we build a *global* basis of s vectors. The dimension of the *local* basis is determined by the **n_self_lanczos** parameter. In contrast with the polarizability case, **n_self_lanczos** is system dependent as it is relative to a KS state and not to a Wannier's function. We observed empirically that a number in the range 1/4-1/2 the dimension of the polarizability basis guarantees convergence. The **s_self_energy** parameter determines the dimension, and accuracy, of the global s vectors basis. Empirically, we found that a basis of global s vectors with a dimension of $\sim$ 4 the number of polarizability basis vectors gives converged results. Then for all the *global* s vectors Lanczos chains are calculated. The length of such chains is given by **nsteps_lanczos_self**. In contrast to the polarizability case more Lanczos steps are required in particular for large systems **nsteps_lanczos_self** ranges for $\sim$ 20 in small systems to $\sim$ 200 in larger ones.

As the assessment of the degree of accuracy can be cumbersome for large systems, we have implemented an option **l_big_system** to be selected both in *pw4gww.x* and in *gww.x* which permits to calculate the self-energy expectation values state by state so that the global s vectors basis coincides with the local one. In this case the **s_self_lanczos** parameter is not used. As this strategy requires more computational time we suggest to use it to evaluate the energies of the most relevant states (e.g. the closest ones to the Fermi energy). Then the ordinary strategy can be used for the other states, verifying that the same energies are calculated for the states previously addressed with the **l_big_system** strategy.

References

[1] M.S. Hybertsen and S.G. Louie, Phys. Rev. B 34, 5390 (1986).

[2] P.Umari, G. Stenuit, and S. Baroni, Phys. Rev. B 79, 201104(R) (2009).

[3] P.Umari, G. Stenuit, and S. Baroni, Phys. Rev. B 81, 115104 (2010).

[4] P. Umari and S. Fabris, J. Chem. Phys. 136, 174310 (2012).