WanT User's Guide

Version 2.0.0

AN INTEGRATED APPROACH TO

AB INITIO ELECTRONIC TRANSPORT FROM MAXIMALLY-LOCALIZED WANNIER FUNCTIONS

December 13, 2005

(`http://www.wannier-transport.org`)

This *User's Guide* describes how to run and use the various features of the integrated WanT approach. This guide includes a description of the capabilities of the program, how to use these capabilities, the necessary input files and formats, and how to run the program on both serial and parallel machines.

WanT Version 2.0.0

CREDITS. The development and maintenance of the WanT code is promoted by the National Research Center on nanoStructures and bioSystems at Surfaces (S3) of the Italian INFM-CNR (http://www.s3.infm.it) and the Physics Department North Carolina State University (NCSU) (http://ermes.physics.ncsu.edu) under the coordination of Arrigo Calzolari, Andrea Ferretti and Marco Buongiorno Nardelli.

The present release of the WanT package has been realized by Andrea Ferretti (S3), Arrigo Calzolari (S3) and Marco Buongiorno Nardelli (NCSU).
A list of contributors includes Carlo Cavazzoni (CINECA), Benedetta Bonferroni (S3) and Nicola Marzari (MIT).

The routines for the calculation of the maximally-localized Wannier functions were originally written by Nicola Marzari and David Vanderbilt (©1997); Ivo Souza, Nicola Marzari and David Vanderbilt (©2002); Arrigo Calzolari, Nicola Marzari, and Marco Buongiorno Nardelli (©2003).

The routines for the calculation of the quantum conductance were originally written by Marco Buongiorno Nardelli (©1998); Arrigo Calzolari, Nicola Marzari, and Marco Buongiorno Nardelli (©2003).

GENERAL DESCRIPTION. WanT is an open-source, GNU General Public License suite of codes that provides an integrated approach for the study of coherent electronic transport in nanostructures. The core methodology combines state-of-the-art Density Functional Theory (DFT), plane-waves, pseudopotential calculations with a Green's functions method based on the Landauer formalism to describe quantum conductance. The essential connection between the two, and a crucial step in the calculation, is the use of the maximally-localized Wannier function representation to introduce naturally the ground-state electronic structure into the lattice Green's function approach at the basis of the evaluation of the quantum conductance. Moreover, the knowledge of Wannier functions allows for a direct link between the electronic transport properties of the device with the nature of the chemical bonds, providing insight into the mechanisms that govern electron flow at the nanoscale.

The WanT package operates, in principles, as a simple post-processing of any standard electronic structure code. In its present version 2.0.0 the user will find a wrapper to run WanT from the results of a self-consistent calculation done using the PWscf package (`http://www.pwscf.org`).

WanT capabilities include: - Quantum conductance spectrum for a bulk (infinite, periodic) system and for a lead-conductor-lead geometry - Density of states spectrum projected on the conductor region - Centers and spreads of the maximally-localized Wannier functions of the system.

TERMS OF USE. Although users are not under any obligation in the spirit of the GNU General Public Licence, the developers of WanT would appreciate the acknowledgment of the effort to produce such codes in the form of the following reference:

In the text: "The results of this work have been obtained using the WanT package.[ref]"

In references: "[ref] WanT code by A. Calzolari, A. Ferretti, C. Cavazzoni, N. Marzari and M. Buongiorno Nardelli, (www.wannier-transport.org). See also: A. Calzolari, N. Marzari, I. Souza and M. Buongiorno Nardelli, Phys. Rev. B 69, 035108 (2004)."

DISCLAIMER. While the developers of WanT make every effort to deliver a high quality scientific software, we do not guarantee that our codes are free from defects. Our software is provided "as is". Users are solely responsible for determining the appropriateness of using this package and assume all risks associated with the use of it, including but not limited to the risks of program errors, damage to or loss of data, programs or equipment, and unavailability or interruption of operations. Due to the limited human resources involved in the development of this software package, no support will be given to individual users for either installation or execution of the codes. Finally, in the spirit of every open source project, any contribution from external users is welcome, encouraged and, if appropriate, will be included in future releases.

LICENCE. All the material included in this distribution is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. These programs are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Contents

1 Theoretical Background

WanT is an open-source, GNU General Public License suite of codes that provides an integrated approach for the study of coherent electronic transport in low-dimensional, extended nanostructures. The core methodology combines state-of-the-art Density Functional Theory (DFT), plane-waves, pseudopotential calculations with a Green's functions method based on the Landauer formalism to describe quantum conductance. The essential connection between the two, and a crucial step in the calculation, is the use of the maximally-localized Wannier function representation to introduce naturally the ground-state electronic structure into the lattice Green's function approach at the basis of the evaluation of the quantum conductance. Moreover, the knowledge of Wannier functions allows for a direct link between the electronic transport properties of the device with the nature of the chemical bonds, providing insight into the mechanisms that govern electron flow at the nanoscale.

WanT scheme was originally described in Ref. [1]:
A. Calzolari, N. Marzari, I. Souza, and M. Buongiorno Nardelli,
Phys. Rev. B **69**, 035108 (2004).

In the following we review the theoretical background that holds the WanT method.

1.1 Quantum transport

Calculations of the quantum conductance are based on a recently developed efficient method for evaluating quantum transport in extended systems [2, 3, 4]. This method is applicable to any Hamiltonian that can be expanded within a localized-orbital basis and can be used as a general theoretical scheme for the computation and analysis of the electrical properties of nanostructures.

1.1.1 Electron transmission and Green's functions

Let us consider a system composed of a conductor, C, connected to two semi-infinite leads, R and L, as in Fig. 1. A fundamental result in the theory of electronic transport is that the zero-temperature conductance through a region of non-interacting electrons (the C region in Fig. 1) is related to the scattering properties of the region itself via the Landauer formula [5]:

$$\mathcal{C} = \frac{2e^2}{h}\mathcal{T}(E_f), \tag{1}$$

where $\mathcal{T}$ is the transmission function, $\mathcal{C}$ is the conductance and E_f the Fermi energy. The former represents the probability that an electron injected at one end of the conductor will transmit to the other end. In principle, we can compute the transmission function for a coherent conductor[1] starting from the knowledge of the scattering matrix, S. The latter is the mathematical quantity that describes the response at one lead due an excitation at another. In principle, the scattering matrix can be uniquely computed from the solution of the Schroedinger equation and would suffice to describe the transport processes we are interested in this work. However, it is a general result of conductance theory that the elements of the S-matrix can be expressed in terms of the Green's function of the conductor [6, 7, 8] which, in practice, can be sometimes simpler to compute.

[1] A conductor is said to be coherent if it can be characterized by a transmission matrix that relates each of the outgoing wave amplitudes to the incoming wave amplitudes at a given energy.

Let us consider a physical system represented by an Hamiltonian H. The Green's functions of the system can be defined as:

$$(\omega \pm i\eta - H)\, G(\omega) = I \tag{2}$$

where I is ht eidentity operator and $i\eta > 0$ is an infinitesimal imaginary part added to the energy to incorporate the boundary conditions into the equation. The solution with + sign is the retarded Green's function G^r, while the solution with $-$ sign is called advanced Green's function G^a. The transmission function can then be expressed in terms of the Green's functions of the conductor and the couplings of the conductor to the leads in a simple manner using the Fisher and Lee formula [7]:

$$\mathcal{T}(\omega) = \mathrm{Tr}\left[\Gamma_L G_C^r \Gamma_R G_C^a\right]. \tag{3}$$

Here $G_C^{\{r,a\}}$ are the retarded and advanced Green's functions of the conductor, and $\Gamma_{\{L,R\}}$ are functions that describe the coupling of the conductor to the leads.

In the following we are going to restrict the discussion to discrete systems that we can describe by ordinary matrix algebra. More precisely, we are going to work with matrices representing a physical system in the basis of localized electronic orbitals centered on the atoms constituting the system. It includes in particular the tight-binding model. For a discrete media, the Green's function defined in Eq. (2) is then the inverse of the $(\omega - H)$ matrix. To simplify the notation, we drop the exponentis $\{a, r\}$ referring to advanced and retarded functions and include the $\pm i\eta$ factor in ω. For an open system, consisting of a conductor and two semi-infinite leads (see Fig. 1), the above Green's function can be partitioned into sub-matrices that correspond to the individual subsystems:

$$\begin{pmatrix} g_L & g_{LC} & g_{LCR} \\ g_{CL} & G_C & g_{CR} \\ g_{LRC} & g_{RC} & g_R \end{pmatrix} = \begin{pmatrix} \omega - h_L & -h_{LC} & 0 \\ -h_{LC}^\dagger & \omega - H_C & -h_{CR} \\ 0 & -h_{CR}^\dagger & \omega - h_R \end{pmatrix}^{-1}, \tag{4}$$

where the matrix $(\omega - H_C)$ represents the finite "isolated" conductor (with no coupling elements to the leads), $(\omega - h_{\{R,L\}})$ represent the semi-infinite leads, and h_{CR} and h_{LC} are the coupling matrices between the conductor and the leads. As a convention, we use lower case letters for (semi-)infinite matrices and upper case for finite dimension matrices. In Eq. (4) we have made the assumption that there is no direct interaction between the left and right leads. From this equation it is straightforward to obtain an explicit expression for G_C[6]:

$$G_C(\omega) = (\omega - H_C - \Sigma_L(\omega) - \Sigma_R(\omega))^{-1} \tag{5}$$

where the finite dimension matrices

$$\Sigma_L(\omega) = h_{LC}^\dagger (\omega - h_L)^{-1} h_{LC}, \quad \Sigma_R(\omega) = h_{RC} (\omega - h_R)^{-1} h_{RC}^\dagger \tag{6}$$

are defined as the self-energies due to the semi-infinite leads. These terms can be viewed as effective Hamiltonians that arise from the coupling of the conductor with the leads. The coupling functions $\Gamma_{\{L,R\}}(\omega)$ can then be obtained as [6]:

$$\Gamma_{\{L,R\}} = \mathrm{i}\left[\Sigma_{\{L,R\}}^r(\omega) - \Sigma_{\{L,R\}}^a(\omega)\right], \tag{7}$$

where the advanced self-energy $\Sigma_{\{L,R\}}^a$ is the Hermitian conjugate of the retarded self-energy $\Sigma_{\{L,R\}}^r$. The core of the problem lies in the calculation of the self-energies of the semi-infinite leads.

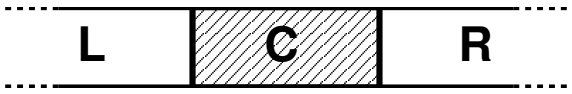

Figure 1: A conductor described by the Hamiltonian H_C, connected to two semi-infinite leads L and R, through the coupling matrices h_{LC} and h_{CR}.

It is well known that any solid (or surface) can be viewed as an infinite (semi-infinite in the case of surfaces) stack of principal layers with nearest-neighbor interactions [9, 10]. This corresponds to transforming the original system into a linear chain of principal layers. For a lead-conductor-lead system, the conductor can be considered as one principal layer sandwiched between two semi-infinite stacks of principal layers. The next sections are devoted to the computation of the self-energies using the principal layers approach.

1.1.2 Transmission through a bulk system.

Within the principal layer approach, the matrix elements of Eq. (2) between layer orbitals yield a series of matrix equations for the Green's functions:

$$\begin{aligned}(\omega - H_{00})G_{00} &= I + H_{01}G_{10} \\ (\omega - H_{00})G_{10} &= H_{01}^{\dagger}G_{00} + H_{01}G_{20} \\ &\cdots \\ (\omega - H_{00})G_{n0} &= H_{01}^{\dagger}G_{n-1,0} + H_{01}G_{n+1,0}\end{aligned} \tag{8}$$

where the finite dimension matrices H_{nm} and G_{nm} are formed by the matrix elements of the Hamiltonian and Green's function between the layer orbitals. We assume that in a bulk system $H_{00} = H_{11} = \ldots$ and $H_{01} = H_{12} = \ldots$. Following Refs. [11, 12], this chain can be transformed in order to express the Green's function of an individual layer in terms of the Green's function of the preceding (or following) one. This is done via the introduction of the transfer matrices T and $\overline{T}$, defined such that $G_{10} = TG_{00}$ and $G_{00} = \overline{T}G_{10}$. Using these definitions, we can write the bulk Green's function as [13]:

$$G(\omega) = (\omega - H_{00} - H_{01}T - H_{01}^{\dagger}\overline{T})^{-1}. \tag{9}$$

The transfer matrix can be easily computed from the Hamiltonian matrix elements via an iterative procedure, as outlined in [11, 12]. In particular T and $\overline{T}$ can be written as:

$$\begin{aligned}T &= t_0 + \tilde{t}_0 t_1 + \tilde{t}_0\tilde{t}_1 t_2 + \ldots + \tilde{t}_0\tilde{t}_1\tilde{t}_2\cdots t_n\,, \\ \overline{T} &= \tilde{t}_0 + t_0\tilde{t}_1 + t_0 t_1\tilde{t}_2 + \ldots + t_0 t_1 t_2\cdots\tilde{t}_n\,,\end{aligned} \tag{10}$$

where t_i and $\tilde{t}_i$ are defined via the recursion formulas:

$$\begin{aligned}t_i &= (I - t_{i-1}\tilde{t}_{i-1} - \tilde{t}_{i-1}t_{i-1})^{-1}t_{i-1}^2, \\ \tilde{t}_i &= (I - t_{i-1}\tilde{t}_{i-1} - \tilde{t}_{i-1}t_{i-1})^{-1}\tilde{t}_{i-1}^2,\end{aligned} \tag{11}$$

and

$$\begin{aligned}t_0 &= (\omega - H_{00})^{-1}H_{01}^{\dagger}, \\ \tilde{t}_0 &= (\omega - H_{00})^{-1}H_{01}.\end{aligned} \tag{12}$$

The process is repeated until $t_n, \tilde{t}_n \leq \delta$ with δ arbitrarily small. Usually, no more than 5 or 6 iterations are required to converge the above sum.

If we compare Eq. (9) with Eq. (5), in the hypothesis of leads and conductors being of the same material (bulk conductivity), we can identify one principal layer of the bulk system with the conductor C, so that $H_{00} \equiv H_C$. In particular, by comparing with Eq.(5), we obtain the expression of the self-energies of the conductor-leads system:

$$\Sigma_L = H_{01}^{\dagger}\overline{T}, \qquad \Sigma_R = H_{01}T. \tag{13}$$

The coupling functions are then obtained [2] from the sole knowledge of the transfer matrices and the coupling Hamiltonian matrix elements: $\Gamma_L = -\mathrm{Im}(H_{01}^{\dagger}\overline{T})$ and $\Gamma_R = -\mathrm{Im}(H_{01}\overline{T})$.

1.1.3 Transmission through a left lead-conductor-right lead (LCR) system.

The procedure outlined above can also be applied to the case of electron transmission through one or more interfaces, between different media. For the calculation of conductances in realistic experimental geometry, the method can be expanded to the general configuration of a Left-lead-Conductor-Right-lead (LCR) systems — as displayed in Fig .1. To study this case we make use of the Surface Green's Function Matching (SGFM) theory, pioneered by [14, 13].

We have to compute the Green's function G_I, where the subscript I refers to the interface region composed of two principal layers — one in each media — (L, C, R in our case). Using the SGFM method, G_I is calculated from the bulk Green's function of the isolated systems, and the coupling between the two principal layers at the two sides of the interface. Via the calculation of the transmitted and reflected amplitudes of an elementary excitation that propagates from one medium to another, it can be shown that the interface Green's function obeys the following secular equation [13]:

$$\begin{aligned} G_{LCR} &= \begin{pmatrix} G_L & G_{LC} & G_{LR} \\ G_{CL} & G_C & G_{CR} \\ G_{RL} & G_{RC} & G_R \end{pmatrix} \\ &= \begin{pmatrix} \omega - H_{00}^L - H_{01\,\dagger}^L\overline{T} & -H_{LC} & 0 \\ -H_{CL} & \omega - H_C & -H_{CR} \\ 0 & -H_{RC} & \omega - H_{00}^R - H_{01}^R T \end{pmatrix}^{-1}. \end{aligned} \tag{14}$$

where $H_{nm}^{\{L,R\}}$ are the block matrices of the Hamiltonian between the layer orbitals in the left and right leads respectively, and $T_{\{L,R\}}$ and $\overline{T}_{\{L,R\}}$ are the appropriate transfer matrices. The latter are easily computed from the Hamiltonian matrix elements via the iterative procedure already described in the bulk case (Sec.1.1.2). Correspondingly, H_{LC} and H_{CR} are the coupling matrices between the conductor and the leads principal layers in contact with the conductor. It is straightforward to obtain in the form of Eq.(5), $G_C = (\omega - H_C - \Sigma_L - \Sigma_R)^{-1}$, where Σ_L and Σ_R are the self-energy terms due to the semi-infinite leads, and identify [2]:

$$\begin{aligned} \Sigma_L(\omega) &= H_{LC}^{\dagger}\left(\omega - H_{00}^L - H_{01}^{L\,\dagger}\overline{T}_L\right)^{-1} H_{LC}, \\ \Sigma_R(\omega) &= H_{CR}\left(\omega - H_{00}^R - H_{01}^R\, T_R\right)^{-1} H_{CR}^{\dagger}. \end{aligned} \tag{15}$$

The transmission function in the LCR geometry can then be derived from Eq.(3) and (7). The knowledge of the conductor's Green's function G_C gives also direct information on the electronic

spectrum of the system via the spectral density of states:

$$N(\omega) = -\frac{1}{\pi}\mathrm{Im}\,\mathrm{Tr}\,G_C(\omega). \tag{16}$$

We have assumed a truly one-dimensional chain of principal layers, which is physical only for systems like nanotubes or quantum wires that have a definite quasi-one-dimensional character. The extension to a truly three-dimensional case is straightforward using Bloch functions in the directions perpendicular to the transport axis. The introduction of the principal layer concept implies that along the direction of the layer expansion the system is described by an infinite set of $k_\perp$ while $k_\parallel$ are still good quantum numbers for the problem. The above procedure effectively reduces the three-dimensional system to a set of non-interacting linear-chains, one for each $k_\parallel$ [9, 10]. We can then use the usual k-point summation techniques to evaluate the quantum conductance:

$$T(\omega) = \sum_{k_\parallel} w_{k_\parallel} T_{k_\parallel}(\omega) \tag{17}$$

where $w_{k_\parallel}$ are the relative weights of the different $k_\parallel$ in the irreducible wedge of the surface Brillouin zone [15].

1.2 Maximally localized Wannier functions

1.2.1 Definition of the problem

Bloch orbitals cannot be used directly to evaluate electronic transport with the method outlined in Sec. 1.1. As we have pointed out, quantum conductance is computed starting from the knowledge of the lattice Green's function, whose calculation relies on a localized orbital representation. Bloch orbitals, that are intrinsically delocalized, have to be transformed into *localized* functions in order to construct the sparse, short-ranged matrix elements of the Hamiltonian. The core of our proposed methodology is to use m*aximally-localized Wannier functions* (WFs) for the system considered. These are the most natural choice for a set of localized orbitals that still span the same Hilbert space of the Hamiltonian eigenfunctions: they allow to bridge plane-wave electronic structure and lattice Green's function calculations in a coherent fashion.

A Wannier function $w_{n\mathbf{R}}(\mathbf{r})$, labeled by the Bravais lattice vector $\mathbf{R}$, is usually defined via a unitary transformation of the Bloch functions $\psi_{n\mathbf{k}}(\mathbf{r})$ of the nth band:

$$w_{n\mathbf{R}}(\mathbf{r}) = \frac{V}{(2\pi)^3}\int_{BZ} \psi_{n\mathbf{k}}(\mathbf{r}) e^{-i\mathbf{k}\cdot\mathbf{R}} d^3k, \tag{18}$$

where V is the volume of the unit cell and the integration is performed over the entire Brillouin Zone. It is easy to show that the WFs defined as above form an orthonormal basis set, and that any two of them, for a given index n and different $\mathbf{R}$ and $\mathbf{R}'$, are just translational images of each other. Note that, as WFs are (continuous) linear combinations of Bloch functions with different energies, they do not represent stationary states, but still span the original Hilbert space.

The *ab-initio* eigenstates are well-defined, modulus an arbitrary $\mathbf{k}$-dependent phase factor; thus, the definition above does not lead to a unique set of Wannier functions [16, 17], since the electronic structure problem is invariant for the transformation $\psi_{n\mathbf{k}} \rightarrow e^{\phi_n(\mathbf{k})}\psi_{n\mathbf{k}}$. Besides this freedom in the choice of phases $\phi_n(\mathbf{k})$ for the Bloch functions, there is a more comprehensive gauge freedom stemming from the fact that the many-body wavefunction is actually a Slater determinant: a unitary transformation between orbitals will not change the manifold, and will not change the total

energy and the charge density of the system. In all generality, starting with a set of $\mathcal{N}$ Bloch functions with periodic parts $u_{n\mathbf{k}}$, we can constructs infinite sets of $\mathcal{N}$ WFs displaying different spatial characteristics:

$$w_{n\mathbf{R}}(\mathbf{r}) = \frac{V}{(2\pi)^3}\int_{BZ}\left[\sum_m U^{(\mathbf{k})}_{mn}\psi_{m\mathbf{k}}(\mathbf{r})\right]e^{-i\mathbf{k}\cdot\mathbf{R}}d^3k. \tag{19}$$

The unitary matrices $U^{(\mathbf{k})}$ include also the gauge freedom on phase factors afore mentioned [18].

The present WanT method is based on a localization algorithm that allows to transform a set of Bloch functions – calculated by means of *ab initio* approaches – into a unique set of Maximally localized Wannier functions, as proposed by Marzari and Vanderbilt in 1997 [18]. The formulation of this minimum-spread criterion extends the concept of *localized molecular orbitals*, proposed by Boys [19] for molecules, to the solid-state case. However, its generality allows to deal with both "extended" (periodic and disordered) systems as well as with "isolated" clusters and molecules, in the limit of large supercells.

1.2.2 Localization procedure

For our purposes, we need to transform the Bloch eigenstates in WFs with the narrowest spatial distribution. Following the procedure proposed by Marzari and Vanderbilt [18], we search the particular unitary matrix $U^{(\mathbf{k})}_{mn}$ that transform the Bloch eigenstates in the WFs with the narrowest spatial distribution.

A measure of the spatial delocalization of WFs is given by a *Spread Operator* Ω, defined as the sum of the second moments of all the Wannier functions in a reference cell:

$$\Omega = \sum_n [\langle r^2\rangle_n - \langle \mathbf{r}\rangle_n^2], \tag{20}$$

where the sum is over a selected group of bands, and

$$\begin{aligned} \langle \mathbf{r}\rangle_n &= \langle \mathbf{0}n|\mathbf{r}|\mathbf{0}n\rangle, \\ \langle r^2\rangle_n &= \langle \mathbf{0}n|r^2|\mathbf{0}n\rangle. \end{aligned} \tag{21}$$

The value of the spread Ω depends on the choice of unitary matrices $U^{(\mathbf{k})}$; thus it is possible to evolve any arbitrary set of $U^{(\mathbf{k})}$ until we reach the stationarity condition:

$$\frac{\delta\Omega_{\mathbf{k}}}{\delta U^{(\mathbf{k})}} = 0 \tag{22}$$

At the minimum, we obtain the matrices $U^{(\mathbf{k}),ML}$ that transform the first-principles $\psi^{FP}_{n\mathbf{k}}(\mathbf{r})$ into the *maximally-localized WFs*, according to Eq. (19). If we restrict to the case of $\mathbf{k}$-point mesh calculations, we can use finite differences in reciprocal space to evaluate the derivatives of Eq. (22). For this purpose we rewrite the expectation values $\langle \mathbf{r}\rangle$ and $\langle r^2\rangle$ as proposed by Blount [20]:

$$\begin{aligned} \langle \mathbf{0}n|\mathbf{r}|\mathbf{0}n\rangle &= i\frac{1}{N}\sum_{\mathbf{k}} e^{+i\mathbf{k}\cdot\mathbf{R}}\langle u_{\mathbf{k}n}|\nabla_{\mathbf{k}}|u_{\mathbf{k}n},\rangle \\ \langle \mathbf{0}n|r^2|\mathbf{0}n\rangle &= \frac{1}{N}\sum_{\mathbf{k}} e^{+i\mathbf{k}\cdot\mathbf{R}}\langle u_{\mathbf{k}n}|\nabla^2_{\mathbf{k}}|u_{\mathbf{k}n}\rangle, \end{aligned} \tag{23}$$

where $|u_{n\mathbf{k}}\rangle = e^{-i\mathbf{k}\cdot\mathbf{r}}|\psi_{n\mathbf{k}}\rangle$ is the periodic part of the Bloch function. Making the assumption that the BZ has been discretized into a uniform $\mathbf{k}$-point mesh, and letting $\mathbf{b}$ being the vectors that connect a mesh point to its near neighbors, we can define the overlap matrix between Bloch orbitals as:

$$M_{mn}^{(\mathbf{k},\mathbf{b})} = \langle u_{m\mathbf{k}}|u_{n\mathbf{k}+\mathbf{b}}\rangle = \langle \psi_{m\mathbf{k}}|e^{-i\mathbf{k}\mathbf{b}}|\psi_{n\mathbf{k}+\mathbf{b}}\rangle. \tag{24}$$

Using the expression of the gradient in terms of finite differences and substituting $M_{mn}^{(\mathbf{k},\mathbf{b})}$ in Eq. (23) we obtain the expressions for $\langle\mathbf{r}\rangle$ and $\langle r^2\rangle$ to be used in the localization procedure:

$$\begin{aligned} \langle\mathbf{r}\rangle_n &= -\frac{1}{N}\sum_{\mathbf{k},\mathbf{b}} w_b\,\mathbf{b}\,\mathrm{Im\,Ln}M_{nn}^{(\mathbf{k},\mathbf{b})} \\ \langle r^2\rangle_n &= \frac{1}{N}\sum_{\mathbf{k},\mathbf{b}} w_b\left[\left(1-|M_{nn}^{(\mathbf{k},\mathbf{b}}|^2\right)+\left(\mathrm{Im\,Ln}M_{nn}^{(\mathbf{k},\mathbf{b})}\right)^2\right] \end{aligned} \tag{25}$$

Here, w_b are the weights of the $\mathbf{b}$-vectors, and must satisfy the completeness condition $\sum_{\mathbf{b}} w_b b_\alpha b_\beta = \delta_{\alpha\beta}$. Substituting the above expression into Eq. (20), we obtain the expression for the spread operator as a function of the overlap matrix $M_{mn}^{(\mathbf{k},\mathbf{b})}$.

In order to calculate the gradient in Eq. (22), we consider the first order change in Ω arising from an infinitesimal transformation $U_{mn}^{(\mathbf{k})} = \delta_{mn} + dW_{mn}^{(\mathbf{k})}$, where dW is an infinitesimal antiunitary matrix ($dW^\dagger = -dW$). The gauge transformation rotates the wave functions according to Eq. (19) into $|u_{\mathbf{k}n}\rangle \to |u_{\mathbf{k}n}\rangle + \sum_m dW_{mn}^{(\mathbf{k})}|u_{\mathbf{k}m}\rangle$. Following the elegant description of Ref. [18], we obtain the final expression for the gradient of the spread functional:

$$G^{(\mathbf{k})} = \frac{\delta\Omega}{dW^{(\mathbf{k})}} = 4\sum_{\mathbf{b}} w_b\left(\frac{R^{\mathbf{k},\mathbf{b})} - R^{(\mathbf{k},\mathbf{b})\,\dagger}}{2} - \frac{T^{(\mathbf{k},\mathbf{b})} + T^{(\mathbf{k},\mathbf{b})\,\dagger}}{2i}\right) \tag{26}$$

where

$$R_{mn}^{(\mathbf{k},\mathbf{b})} = M_{mn}^{(\mathbf{k},\mathbf{b})}M_{nn}^{(\mathbf{k},\mathbf{b})*}; \qquad T_{mn}^{(\mathbf{k},\mathbf{b})} = \frac{M_{mn}^{(\mathbf{k},\mathbf{b})}}{M_{nn}^{(\mathbf{k},\mathbf{b})}}\left[\mathrm{Im\,Ln}M_{nn}^{(\mathbf{k},\mathbf{b})} + \mathbf{b}\cdot\langle\mathbf{r}\rangle_n\right]. \tag{27}$$

Note that the entire expression $G^{(\mathbf{k})}$ is a function of the overlap matrices $M^{(\mathbf{k},\mathbf{b})}$. The minimization of the spread functional Ω is obtained via steepest descent or conjugate gradient schemes. The procedure does not require the updating of wavefunctions, but only of the overlap and unitary matrixes. This is the most demanding task (scaling as N^3) for each iteration in Wannier localization.

Wannier functions obtained with the above procedure should be almost real, except for an overall phase factor. This conjecture can also be used as a check of the convergence of the localization procedure. It is important to notice that whenever a Born-von Karman discretization of the Brillouin Zone is introduced, even the above-mentioned WFs are not truly localized, but will be periodic in real-space, with a *superperiodicity* determined by the BZ discretization. The truly isolated limit is recovered only in the case of continuous BZ integrations. This is easily seen remembering that $\psi_{n\mathbf{k}}(\mathbf{r}) = u_{n\mathbf{k}}(\mathbf{r})e^{i\mathbf{k}\cdot\mathbf{r}}$, and $u_{n\mathbf{k}}(\mathbf{r})$ has the periodicity of the direct lattice; thus the phase factors $e^{i\mathbf{k}\cdot\mathbf{r}}$ determine the *superperiodicity* of the $\psi_{n\mathbf{k}}$ themselves.

In the standard language of electronic-structure calculations, if the $\psi_{n\mathbf{k}}$ have $\mathbf{k}$'s that are restricted to a uniform Monkhorst-Pack mesh, they will all be periodic with a wavelength inversely proportional to the spacing of the mesh; this periodicity is consequently inherited by the WFs. For $\mathcal{N}$ $\mathbf{k}$-points along a direction of the BZ, the WFs will repeat along the corresponding direction

every $\mathcal{N}$ cells. A dense mesh of **k**-points guarantees that the adjacent replicas of a WF are sufficiently far and do not interact. However, even the case of Γ-sampling is encompassed by the above formulation. In this case the neighboring **k**-points for Γ are given by the homologous Γ-points of the neighboring cells. In this case the algebra becomes simpler and an equivalent real-space formulation is preferred [21, 22].

The method described above works properly in the case of *isolated groups* of bands. A Bloch band is called *isolated* if it does not become degenerate with any other band anywhere in the BZ. Conversely, a group of bands is said to form a *composite group* if bands are inter-connected by degeneracy, but are *isolated* from all the other bands [18]. On the other hand to study quantum conductance in extended systems we often need to compute WFs for a subset of energy bands that are entangled or mixed with other bands. Most often we are interested in the states that lie in the vicinity of the Fermi level of a conductor in a restricted energy range. Since the unitary transformations $U^{(\mathbf{k})}$ mix energy bands at each **k**-point, any arbitrary choice of states inside a prescribed window will affect the localization properties of WFs unless energy gaps effectively separate the manifold of interest from higher and lower bands. This problem has been solved by Souza, Marzari, and Vanderbilt [23], introducing an additional disentanglement procedure that automatically extracts the best possible manifold of a given dimension from the states falling in a predefined energy window. This is the generalization to *entangled* or metallic cases of the maximally-localized WF formulation. The procedure relies on minimizing the subspace dispersion across the Brillouin Zone, and effectively extracts the bands of interest from the overall band structure.

In practice, first we select a desired number of bands in an energy window; then we determine the optimally-connected subspace that can be extracted from that band structure; and finally proceed with a standard localization procedure inside the selected subspace. The resulting orbitals have the same good localization properties, and allow to apply our formalism to arbitrary systems, independently of the insulating or metallic nature of the band manifold. It should be stressed that the WFs obtained in the later case are not the WFs of the occupied subspace (that would exhibit poor localization properties), but are those of a well connected, continuous subspace that in general will contain both occupied and unoccupied Bloch functions.

1.2.3 Conditioned localization and penalty functionals

The above described formalism adopts a localization criterion which is somehow *global*: the functional we want to minimize is the total spread, while we are often interested in the localization of each Wannier function in the selected set. In this scenario it may happen that, in order to gain localization in the total spread one or more WFs may move their centers out of the system (*e.g.* in the vacuum) and therefore make the remaining functions lleave to better localize. In this cases the final WF set is useless from our point of view.

We therefore introduce the idea of *conditioned* localization adding some further *penalty* functional to the total spread. These penalty functionals may be use to drive the Wannier localization towards the achievement of some required feature. It maybe interesting *e.g.* to fix the problem of WFs moving in the vacuum region to add a *spring* potential driving the functions to have their centers as close as possible to the positions we like. The form of the penalty functional is therefore:

$$\Omega_P = A \sum_n w_n \left[\langle \mathbf{r} \rangle_n - \mathbf{r}_{n0} \right]^2, \tag{28}$$

where A is the functional amplitude, w_n a weight, and $\mathbf{r}_{n0}$ is the chosen target position for the $n-$th WF. After some tedious algebra it is possible to write and explicit form for the derivative

of this functional wrt $U^{(\mathbf{k})}$. This is then used to compute the penalty contribution to the gradient during the minimization of the functional. Setting to zero some weights w_n we are also able to selectively switch off the conditioning on some WFs. Obviously many different kinds of such penalty functionals may be written. We note that this procedure does not alter physical quantities such as the polarization, which is infact independent of the particular unitary rotations chosen for the localization. The conditioned minimization fo Eq. (28) is implemented in WanT; see Sec. 4 for more details about how to setup the input file.

1.2.4 Real space hamiltonians

In order to calculate the conductance according to the prescriptions outlined in Sec. 1.1, we need as an input the matrix elements of the Hamiltonian calculated on a localized basis: in our case, it is the minimal basis of the maximally-localized WFs. Assuming a BZ sampling fine enough to eliminate the interaction with the WF periodic images, we can simply compute the WF Hamiltonians $H_{ij}(\mathbf{R}) = \langle w_{i\mathbf{0}}|H|w_{j\mathbf{R}}\rangle$, from the unitary rotations $U^{(\mathbf{k})}$ obtained in the localization procedure.

In the Bloch representation we have by definition $H_{mn}(\mathbf{k}) = \epsilon_{m\mathbf{k}}\delta_{m,n}$. Moving to the Wannier basis we have:

$$H^{(rot)}(\mathbf{k}) = U^{(\mathbf{k})\,\dagger}\, H(\mathbf{k})\, U^{(\mathbf{k})}. \tag{29}$$

Next we Fourier transform $H^{(rot)}(\mathbf{k})$ into the corresponding set of Bravais lattice vectors $\{\mathbf{R}\}$:

$$H_{ij}(\mathbf{R}) = \frac{1}{N_{kp}} \sum_{\mathbf{k}} e^{-i\mathbf{k}\cdot\mathbf{R}}\, H^{(rot)}_{ij}(\mathbf{k}) \tag{30}$$

2 Installation procedure

NOTES: (i) The present version of the code adopts the installation procedure of the PWscf package (for more details see also `http://www.pwscf.org`). (ii) This installation procedure is still experimental, and only a limited number of architectures are currently supported. Details are also reported in the `$TOPDIR/docs/README.install` file, where `$TOPDIR` is the top directory of the WanT source tree.

Installation is a two-step procedure:

1. `cd` to the top directory of the WanT tree, and issue this command at the shell prompt:
   `./configure [⟨options⟩]`
2. Now run:
   `make ⟨target⟩`

where ⟨`target`⟩ is one (or more) of the following: `wannier, transport, libwant, libiotk, all, clean, wash`. Running `make` without arguments prints a short manual. Cross-compilation is not currently supported.

2.1 Step one: configuring

`configure` is a GNU-style configuration script, automatically generated by GNU Autoconf. (If you want to play with it, its source file is `$TOPDIR/conf/configure.ac`; you may also want to edit `$TOPDIR/conf/make.sys.in`) It generates the following files:

`$TOPDIR/make.sys`	compilation settings and flags
`$TOPDIR/*/make.depend`	dependencies, in each source dir

Files `make.depend` are actually generated by the `makedeps.sh` shell script. If you modify the program sources, you might have to rerun it. Note that you must run it from the directory it is in.

To force using a particular compiler, or compilation flags, or libraries, you may set the appropriate environment variables when running the configuration script. For example:

```
./configure CC=gcc CFLAGS=-O3 LIBS="-llapack -lblas -lfftw"
```

Some of those environment variables are:

`TOPDIR`	: top directory of the WanT tree (defaults to 'pwd')
`F90, F77, CC`	: Fortran 90, Fortran 77, and C compilers
`CPP`	: source file preprocessor (defaults to "$CC -E")
`LD`	: linker (defaults to $F90)
`CFLAGS, FFLAGS,`	
`F90FLAGS, CPPFLAGS, LDFLAGS`	: compilation flags
`LIBDIRS`	: extra directories to search for libraries (see below)

You should always be able to compile the WanT suite of programs without having to edit any of the generated files. If you ever have to, that should be considered a bug in the configuration script and you are encouraged to submit a bug report.

IMPORTANT: WanT can take advantage of several optimized numerical libraries:
– ESSL on AIX systems (shipped by IBM)
– MKL together with Intel compilers (shipped by Intel, free for non-commercial use)
– ATLAS (freely downloadable from `http://math-atlas.sourceforge.net`)
– FFTW (freely downloadable from `http://www.fftw.org`)

The configuration script attempts to find those libraries, but may fail if they have been installed in non-standard locations. You should look at the LIBS environment variable (either in the output of the configuration script, or in the generated `make.sys`) to check whether all available libraries were found. If any libraries weren't found, you can rerun the configuration script and pass it a list of directories to search, by setting the environment variable LIBDIRS; directories in the list must be separated by spaces. For example:

```
./configure LIBDIRS="/opt/intel/mkl/mkl61/lib/32 /usr/local/lib/fftw-2.1.5"
```

If this still fails, you may set the environment variable LIBS manually and retry. For example:

```
./configure LIBS="-L/cineca/prod/intel/lib -lfftw -llapack -lblas"
```

Beware that in this case, you must specify **all** the libraries that you want to link to. The configuration script will blindly accept the specified value, and will **not** search for any extra library.

If you want to use the FFTW library, the `fftw.h` include file is also required. If the configuration script wasn't able to find it, you can specify the correct directory in the INCLUDEFFTW environment variable. For example:

```
./configure INCLUDEFFTW="/cineca/lib/fftw-2.1.3/fftw"
```

2.2 Step two: compiling

Here is a list of available compilation targets:

`make wannier`	compile	`disentangle.x`	(step 1)
		`wannier.x`	(step 2)
		`bands.x`	(post proc)
		`plot.x`	(post proc)
		`blc2wan.x`	(post proc)
`make transport`	compile	`conductor.x`	(step 3)
`make all`		`make wannier + transport`	
`make libwant`	compile	WanT basic libs	
`make libiotk`	compile	Input-Output toolkit lib (iotk)	
`make clean`	remove	Object files, libs and executables	
`make clean_test`	remove	test output files	
`make wash`	remove	Configuration files too	

IMPORTANT: If you change any compilation or precompilation option after a previous (successful or failed) compilation, you must run `make clean` before recompiling, unless you know exactly which routines are affected by the changed options and how to force their recompilation.

2.3 List of directories

Within the top directory of the WanT tree there are the following directories:

conf	configuration files
docs	documentation files and manuals
iotk	source files for the iotk library (input-output toolkit, by G. Bussi)
wannier	source files for the Wannier functions suite /par
transport	source files and modules for transport code
bin	links to all the executables
include	include files *.h
libs	source files for basic common libraries
tests	tutorial examples for the use of WanT suite
utility	general utility and tools.

3 How to run WANT : a step by step description

NOTE: At present the WANT code is implemented to work as post processing of DFT calculations done using the PWSCF package (`http://www.pwscf.org`); we will refer to that code in the following.

Following the theoretical description of Section 1, the evaluation of transport properties requires three separate steps:

1. Calculation of DFT electronic structure
2. Calculation of maximally-localized Wannier functions
3. Calculation of quantum conductance

IMPORTANT: For a correct results, the following steps **MUST** be done in the reported order.

3.1 Preliminary Steps: DFT Calculations

i) Self-consistent calculation using the PWSCF code.
For the description of the input and for further details see the PWSCF manual.

ii) Bandstucture calculation.
Starting from the self-consistent charge calculated in point (1), we calculate the Bloch functions for a **REGULAR k-point grid in the COMPLETE Brillouin Zone**; Gamma point must be included. Reduction of **k**-points due to time-reversal symmetry is not (yet) allowed. The complete list of **k**-points should be specified in the `K_POINTS` card. The simple program `kgrid.f90` in `$TOPDIR/utility` can be used to generate non-symmetrized Monkhorst-Pack grids.

iii) From PWSCF to the Wannier code.
Use the post processing `pw_export.x` (distributed in the PWSCF package [2]), to extract the input data necessary for the following Wannier calculations from PWSCF output datafile. Data will be stored in the newly-created directory `$prefix.export/`.

NOTE: Steeps (i-iii) should be be run, using the parallel version of the code, paying attention to use the same number of processors. From this point to the end, instead, the code is scalar.

3.2 Calculation of maximally-localized Wannier functions

Following the list of input parameters as in Section 4 (also reported in the `README.input` file in `$TOPDIR/docs/`) and the examples in directory `$TOPDIR/tests/`, create your own input file, that will be used for the following two steps (a-b).

a) `disentangle.x`: Starting from the data stored at level (iii), the code selects the working energy window. From there it will extract a selected number (N) of WFs to setup the Hilbert subspace for Wannier localization. For each **k**-point, the energy window **must** contain a

[2] The export utility is already distributed in the Qantum-Espresso v3.0, but a patch to include it also in versions v2.1.x is available at `http://www.wannier-transport.org` or `http://www.pwscf.org`. When using v.3.0.0 do not compile with the `-D__NEWPUNCH` pre-processor flag.

number of bands not lower than N. It is possible to use an inner window (inside the working one) to treat frozen-states, for details see Ref. [23]: these frozen Bloch-states will be kept as they are in the Wannier subspace. Trial Wannier centers are not mandatory in this step (depending on the `subspace_init` flag). The code produces a standard output with the main results, and two internal files: `$prefix_$postfix.ovp` and `$prefix_$postfix.space` . The former keeps trace of the computed overlap and projection integrals (to be re-used in further or restarted calculations) while the latter describe the wannier optimal subspace.

b) `wannier.x`: the code reads the same input of `disentangle.x` and performs the localization procedure leading to the maximally localized WFs. The optimal unitary matrices $U(\mathbf{k})$ governing the transformation between Bloch and Wannier states are obtained. In this step different trial centers can be used, but it is a standard procedure to keep those used in the `disentangle.x` run. `wannier.x` produces a standard output with describing the iterative path to the optimally localized WFs. The code also writes two internal data-files: `$prefix_$postfix.wan` containing the $U(\mathbf{k})$ and `$prefix_$postfix.ham` with the hamiltonian matrix elements on the Wannier basis.

Further physical information may be obtained as external post-processing of the Wfs calculation. They are not necessary for the transport calculation, but they may be useful for a better understanding of the intrinsic electronic properties of the system. These codes require a separate input, to be generated following Sec. 4 or the file `$TOPDIR/docs/README.input` .

- `bands.x`: the code computes the interpolated band-structure of the system along selected direction in the Brillouin Zone. The comparison with indipendently calculated DFT eigenvalues is a nice test to check the localization of the obtained WFs. When they are well behaved (*i.e.* localized), only few $\mathbf{R}$ lattice vectors are required to described the Hamiltonian on the Wannier basis $[H_{ij}(\mathbf{R})]$ Starting with a small set of $\mathbf{k}$ in the DFT calculation we obtain the Hamiltonian on the related (small) set of lattice vectors. When WFs are well localized the Hamiltonian is fully described on this set of $\mathbf{R}$ and we can diagonalize it for an arbitrary large set of $\mathbf{k}$ points (as a post-processing of the Wannier code). This is the interpolation of the band structure using WFs. If they are not localized we are essentially throuwing away some non-negligible matrix elements in the Hamiltonian representation, and the bands are not accurate. Typical unphysical oscillations appears in these cases. The code produces three files `$prefix_$postfix_dftband.dat, $prefix_$postfix_wanband.dat` and `$prefix_$postfix_intband.dat` which cab be used for a direct visualization.

- `plot.x`: this is an utility to plot WFs in real space. The plotting region can be tuned and a genereic number of WFs can be handled in a single run. The real or immaginary parts of the WFs as well as their squared moduli are allowed fields to be plotted. The code produces plot files in various formats (txt, gaussian cube, xsf, plt) allowing the use of standard open source visualization-packages such as gOpenMol or xCrysDen. Output files are labelled as `$prefix_$postfix_WF$type_$num.$fmt`, where `$type` can be `R, I, M` according to the plotted field (real or immaginary part, squared modulus), `$num` is the index of the chosen WF and `$fmt` is the output format. If needed, an auxiliary file with the atomic structure in the `xyz` format is also produced.

- `blc2wan.x`: transforms a generic (eventually dynamic) operator given on the Bloch eigenstates basis $A_{mn}(\mathbf{k}) = \langle \psi_{m\mathbf{k}} | \, \widehat{A}(\omega) \, | \psi_{n\mathbf{k}} \rangle$ to the WFs representation. Using the unitary matrices $U(\mathbf{k})$

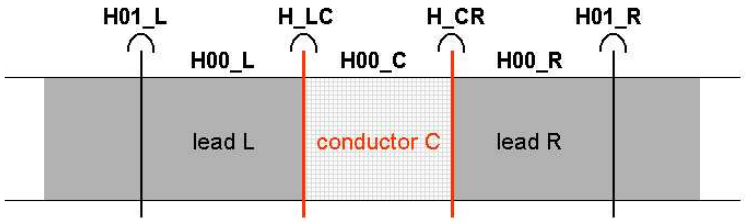

Figure 2: Schematic definitions for Hamiltonian block matrices used in transport calculations.

calculated during the Wannier localization (b) and the definition of the Wannier subspace in terms of the Bloch eigenstates (a), $\widehat{A}(\omega)$ is described in terms of the matrix elements $A_{ij}(\mathbf{R}) = \langle w_{i\mathbf{0}} | \widehat{A}(\omega) | w_{j\mathbf{R}} \rangle$.

3.3 Calculation of electronic transport

Using the hamiltonian matrices calculated in step (b) we can calculate both the *bulk* and the *two-terminal* transmittance. Input file may be generated following Sec. 4 or the file `README.input` in `$TOPDIR/docs`. The main result of the calculation is the quantum transmittance across the conductor region which is written on file `$SCRATCH/cond.dat`. Conductance is given in $2e^2/h$ units. Calculations are performed using the Fisher-Lee formula according to Sec. 1.1. As an auxiliary piece of information, the density of states (DOS) projected on the conductor is also writte in `$SCRATCH/dos.dat`. This DOS is computed as the trace of the conductor Green's function $N(E) = -(1/\pi)\,\mathrm{Im}\,\mathrm{Tr}\,G_C(E)$.

Except the special case of the bulk transmittance (see below), the systems one is interested in are generally not periodic along the transport direction. This does not in principle avoid a 2D periodicity in the orthogonal plane. The use of 2D mesh of **k**-points in this context is implemented in the present version of the code.

Before discussing in more ditail the actual calculations performed by the transport code, we precisely define the adopted convention for the names of the real-space (i.e. WF) Hamiltonian blocks. Given a conductor (C) connected to left (L) and right (R) leads, containing respectively N_C, N_L and N_R WFs, we define the following matrices (see Fig. 2):

`H00_L`	=	$N_L \times N_L$ on-site hamiltonian of the leads L	(from L-bulk calc.)
`H01_L`	=	$N_L \times N_L$ hopping hamiltonian of the leads L	(from L-bulk calc.)
`H00_R`	=	$N_R \times N_R$ on site hamiltonian of the leads R	(from R-bulk calc.)
`H01_R`	=	$N_R \times N_R$ hopping hamiltonian of the leads R	(from R-bulk calc.)
`H00_C`	=	$N_C \times N_C$ on site hamiltonian of the conductor C	(from C-supercell calc.)
`H_LC`	=	$N_L \times N_C$ coupling between lead L and conductor C	(from C-supercell calc.)
`H_CR`	=	$N_C \times N_R$ coupling between conductor C and lead R	(from C-supercell calc.)

Table 1: Main Hamiltonian blocks neede for transport calculation

In the general case we need to compute the electronic structure and WFs for three different regions L, C, R. Very often one is interested in a situation where the leads are composed of the same

material. We wil discuss in more detail this case, the being treatable in the same way except fo a more complicated geometry in the conductor region.

First we note that the conductor calculation should contain part of the leads in the simulation cell, in order to treat the interface from frist principles. The amount of lead layers to be included should be converged upto the local electronic structure of the bulk lead is reached at the edges of the supercell. This convergence can be controlled taking a look at the hamiltonian matrix elements on Wannier states located in the lead region (*e.g.* nearest neighbour interactions). Note that this is a physical condition related to the need for a matching of different calculations and not to the peculiar use of WFs as a basis: nevertheless the smaller the WFs the more indipendend on the environment are the matrix elements, which leads to a faster convergence.

As from Tab. 1, the `H00_C` term can be obtained directly from the conductor supercell calculation. The on-site block ($\mathbf{R} = \mathbf{0}$) is automatically selected. The same is true in general for the lead-conductor coupling (consider for instance `H_CR`): here the rows of the matrix are related to (all) the WFs in the conductor reference cell while the columns usually refer to the some of them in the nearest neighbor cell along transport direction (say *e.g.* the third lattice vector). The `H_CR` Hamiltonian we are interested in is therefore a $N_C \times N_R$ submatrix of the $\mathbf{R} = (0, 0, 1)$ block. In order to understand which rows and columns should enter the submatrix, we need to identify some WFs in the conductor with those obtained for bulk lead calculation. This assumption is strictly correlated with that about the local electronic structure at the edge of the conductor supercell: the more we reach the electronics of the leads, the more WFs will be similar to those of bulk leads. As before, the smaller the WFs, the more indipendent of the environment they are. In this way we can extract the lead-conductor coupling matrices directly from the supercell conductor calculation. Much care must be taken in this step in order to avoid numerical noise. See Sec. 4 for the input file details.

The missing Hamiltonians (`H00_x, H01_x`, where `x=L,R`) can be obtained from direct calculations for the bulk leads and are taken from the $\mathbf{R} = (0, 0, 0)$ and $\mathbf{R} = (0, 0, 1)$ blocks respectively (as from Tab. 1). All these Hamiltonian matrix elements are related to a zero of the energy scale set at the Fermi energy of the computed system (the top of valence band for semiconductors). It is not therefore guaranteed the zero of the enrgy to be exactly the same when moving from the conductor to the leads (which comes from different calculations.) In order to match the hamiltonian matrices at the boundary, it is necessary to check that the corresponding diagonal elements (the only affected by a shift in the energy scale) of `H00_L, H00_C and H00_R` matrices are aligned (see the end of the `wannier.x` output-file to easily find these elements). If not, a rigid shift may be applied.

Alternatively, the `H00_x` and `H01_x` matrices may be obtained from the conductor supercell calculation too. We need to identify the WFs corresponding to some principal layer of the leads and extract the corresponding rows and columns. This procedure is not affected by any energy-offset problem, but larger supercells should be used in order to obtain environment (conductor) indipendent matrices. See the tests to check the numerical performance of these schemes.

The transport code distributed within the WanTpackage is `conductor.x` which calculates both the *bulk* and the *two-terminal* transmittance for the systems. While in the former case we only need the `H00_C` and the `H_CR` matrices, the whole complexity of input data described above is managed in the latter.

4 How to setup input files

According to the methodological scheme of Sec. 3, it is necessary to use separate input files at the different step of the WanT procedure.

Input files are organized using several `NAMELIST`s, followed by other fields with more massive data `CARDS`. Namelists are begin with the flag `&NAMELIST` and end with the "/" bar. The order of variables within a namelist is arbitrary. Most variables have default values mandatory. If a variable is not explicitly defined in the input file, its default value is assumed. Other variables are mandatory and must be always supplied. In the following we report the list and the description of the details of each required input file.

4.1 Input for DFT-PW calculations

Step 1. i-ii: `pw.x`
WanT is currently interfaced with PWscf code. For the description of the input for steps 1-2 (Sec. 3) and for further details see the PWscf manual at `http://www.pwscf.org`.

Step 1. iii: `pw_export.x`
Input file layout:

```
&INPUTPP
    ...
    /
```

List of variables

`prefix` STRING
the first part of the name of all the files written by the code. When using PWscf for the DFT calculation, `prefix` should be the same.
DEFAULT = `mandatory`

`outdir` STRING
the scratch directory where the massive data-files will be written.
DEFAULT = `"./"`

`pseudo_dir` STRING
directory containing pseudopotential (PP) files.
DEFAULT = `"./"`

`psfile(i)` STRING
files containing i-th pseudopotential, where $i = 1, N_{\rm type}$. PP numbering must follow the ordering defined in the input of `pw.x` .
DEFAULT = `""`

`single_file` LOGICAL
if `.TRUE.` a single output file is produced, otherwise the output is a directory with few files.
DEFAULT = `.FALSE.`

`ascii` LOGICAL
if `.TRUE.` output files are textual, otherwise they are partly binary.
DEFAULT = `.FALSE.`

4.2 Input for Wannier function calculations

Step 2. a-b: `disentangle.x wannier.x`
Both codes for WF calculation use the same input file.

Input file layout:

```
&CONTROL
...
/

&SUBSPACE
...
/

&LOCALIZATION
...
/

WANNIER_CENTERS  ( "crystal" | "angstrom" | "bohr" )
⟨type_1⟩    ⟨specific_fmt⟩
...
⟨type_N⟩    ⟨specific_fmt⟩
```

Namelist `&CONTROL`

`prefix` STRING
the first part of the name of all the files written by the code.
DEFAULT = `mandatory`

`postfix` STRING
the tail of the names of the above mentioned files (useful *e.g.* to distinguish among different calculations having a common part).
DEFAULT = `""`

`work_dir` STRING
the scratch directory where the massive data-files will be written.
DEFAULT = `"./"`

`title` STRING
the title of the calculation.
DEFAULT = `"Wannier Transport Calculation"`

`restart_mode` STRING

("from_scratch" | "restart")
define whether to restart a previous calculation; at the moment the "restart" choice implies to give the value "from_file" to the input variables overlaps, projections, start_mode_dis and start_mode_wan (see below for thier meanings).
DEFAULT = "from_scratch"

verbosity — STRING
("low" | "medium" | "high")
the level of detail of the textual output files.
DEFAULT = "medium"

overlaps — STRING
("from_scratch" | "from_file")
determine how to get overlap integrals:
"from_scratch": overlaps are calculated from wfcs
"from_file": overlaps are read from a previous data file. In this second case the dimensions of the problem should be the same as in the original calculation.
DEFAULT = "from_scratch"

projections — STRING
("from_scratch" | "from_file")
determine how to get projections integrals: the meaning of the options is as for overlaps variable.
DEFAULT = "from_scratch"

assume_ncpp — LOGICAL
if .TRUE. avoids the reading of pseudopotential files assuming that the DFT calculation has been performed using norm-conserving pseudopotentials (no knowledge of them is required in the WANT calculation in this case).
DEFAULT = .FALSE.

unitary_thr — REAL
threshold for the check of matrix unitariery.
DEFAULT = 1.0d-6

Namelist &SUBSPACE

dimwann — INTEGER
the number of Wannier functions, *i.e.* the dimension of the Wannier subspace.
DEFAULT = mandatory

win_min — REAL
the lower limit [eV] of the energy window containing the Bloch states forming the Wannier subspace).
DEFAULT = $-\infty$

win_max — REAL
the upper limit [eV] of the energy window described above.
DEFAULT = $+\infty$

`froz_min` REAL
the lower limit [eV] of the energy window containing *frozen* Bloch states: they will be taken as they are in the Wannier subspace and do not enter the disentanglement procedure.
DEFAULT = $-\infty$

`froz_max` REAL
upper limit [eV] of the frozen window described above.
DEFAULT = $-\infty$

`alpha_dis` REAL
mixing parameter for the disentangle iterative procedure.
DEFAULT = `0.5`

`maxiter_dis` INTEGER
maximum number of iterations during the disentangle procedure.
DEFAULT = `1000`

`nprint_dis` INTEGER
every `nprint_dis` iterations in disentangle minimization write to stdout.
DEFAULT = `10`

`nsave_dis` INTEGER
every `nsave_dis` iterations save subspace data to disk.
DEFAULT = `10`

`use_blimit` LOGICAL
if `.TRUE.`, **b**-vectors are set to zero when calculating overlap augmentations. This essentially means we are doing a sort of thermodynamic limit even if this is not consistent with the actual kpt grid. The `.TRUE.` value should be considered for debug purposes.
DEFAULT = `.FALSE.`

`disentangle_thr` REAL
threshold for convergence of the iterative disentangle procedure.
DEFAULT = `1.0d-8`

`subspace_init` STRING
`( "randomized" | "lower_states" | "upper_states" | "center_projections" | "from_file" )`
Determine how the trial subspace is chosen
`"randomized"` : random starting point is chosen
`"lower_states"` : the lower `dimwann` bands from DFT calculation are used to define the subspace
`"upper_states"` : the upper `dimwann` bands from DFT calculation are used to define the subspace
`"center_projections"` : a subspace is extracted from the DFT bands by means of a projections on the given Wannier trial-centers (see the card `WANNIER_CENTERS`)
`"from_file"` : subspace initialization is read from an existing data file; this is the choice used during restart.

DEFAULT = "center_projections"

spin_component STRING
("up" | "down" | "none")
defines whether the calculation is spin polarized and if the case which spin component is to be treated.
DEFAULT = "none"

Namelist &LOCALIZATION

wannier_thr REAL
threshold for convergence of the iterative Wannier minimization.
DEFAULT = 1.0d-6

alpha0_wan REAL
mixing parameter during the steepest-descent minimization steps.
DEFAULT = 0.5

alpha1_wan REAL
mixing parameter during the conjugate-gradient minimization steps.
DEFAULT = 0.5

maxiter0_wan INTEGER
maximum number of steps for the steepest descent minimization (first part).
DEFAULT = 500

maxiter1_wan INTEGER
maximum number of steps for the conjugate-gradient minimization (second part).
DEFAULT = 500

nprint_wan INTEGER
every nprint_wan iterations in wannier minimization write to stdout.
DEFAULT = 10

nsave_wan INTEGER
every nsave_wan iterations save subspace data to disk.
DEFAULT = 10

ncg INTEGER
every ncg iterations in the second minimization part, do a steepest-descent step.
DEFAULT = 3

localization_init STRING
("no_guess" | "randomized" | "center_projections" | "from_file")
Determine how the Wannier localization is started
"no_guess" : disentangle states are used as starting point without any further localization guess
"randomized" : a random rotation is applied to the states found by the disentangle procedure

"center_projections" : a subspace is extracted from the DFT bands by means of a projections on the given Wannier trial-centers
"from_file" : subspace initialization is read from an existing data file; this is the choice used during restart.
DEFAULT = "center_projections"

ordering_mode STRING
("none" | "spatial" | "spread" | "complete")
specifies whether to order the computed Wannier functions and which ordering criterion adopt
"none": no ordering is performed
"spatial": ordering based on WF center positions
"spread": ordering based on WF increasing spreads
"complete": spatial + spread.
DEFAULT = "none"

a_condmin REAL
the amplitude of the conditioned minimization functional. If set to zero ordinary minimization is performed.
DEFAULT = 0.0

niter_condmin INTEGER
the number of steps for which minimization is conditioned.
DEFAULT = maxiter0_wan + maxiter1_wan (if a_condmin $\neq 0.0$); 0 (otherwise)

dump_condmin REAL
the dumping factor for a_condmin during the conditioned minimization. If the dumping factor is specified, after niter_condmin iterations a_condmin is dumped according to a_condmin = a_condmin * dump_condmin at each iteration.
DEFAULT = 0.0

Card WANNIER_CENTERS

WANNIER_CENTERS ("crystal" | "angstrom" | "bohr")

Aside the tag WANNIER_CENTERS, units for positions maybe specified:
"crystal" : relative coordinates on the basis of $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ direct lattice vector (default)
"bohr" : cartesian coordinates in Bohr
"angstrom" : cartesian coordinates in Angstrom

Next the card contains dimwann lines giving the trial centers for the WFs. Depending on the ⟨type⟩ flag at the beginning of the line, formats are different.

⟨type⟩ may assume the values: "atomic" , "1gauss", "2gauss"

```
if ( type == "atomic" ) → atomic   iatomic   l  m                [weight]
if ( type == "1gauss" ) → 1gauss   x  y  z   l  m         rloc   [weight]
if ( type == "2gauss" ) → 2gauss   x  y  z   xx  yy  zz   rloc   [weight]
```

`type == "1gauss"`
The trial center is given by a single gaussian set at a given position with a given angular momentum. Standard positions are usually atomic sites or bond midpoints.

`x, y, z` REAL
define the position of the trial function. Units maybe specified aside the tag `WANNIER_CENTERS`, see above for more details.

`l, m` INTEGER
are the angular momentum quantum numbers for the spherical harmonics giving the angular part of the trial center. `l` can be set equal to 0, 1, or 2, (and `m` values are then as usual) for standard spherical harmonics or `l` == -1 indicating the sp3 geometry. Here spherical harmonics are the real ones:

```
l == -1:  m = -4  →   1,1,-1    dir
          m = -3  →   1,-1, 1   dir
          m = -2  →  -1, 1,1    dir
          m = -1  →  -1,-1,-1   dir
          m = 1   →   1,1, 1    dir
          m = 2   →   1,-1,-1   dir
          m = 3   →  -1,1,-1    dir
          m = 4   →  -1,-1, 1   dir
l == 0:   m = 0   →  spherical
l == 1:   m = -1  →  x
          m = 0   →  z
          m = 1   →  y
l == 2:   m = -2  →  x^2 − y^2
          m = -1  →  xz
          m = 0   →  3z^2 − r^2
          m = 1   →  yz
          m = 2   →  xy
```

`rloc` REAL
specifies the spread of the gaussian used for the radial part of the trial WF. Units are Bohr for both `"bohr"` and `"crystal"` and Angstrom for `"angstrom"` specifier.

`weight` REAL
this value is required when conditioned minimization is performed. In case, it should be set in the interval [0, 1]. It weights the relative importance of each center in the penalty functional. Weight = 0 is used to switch off the constrain for a given center.

`type == "2gauss"`
The trial function is given as the difference between s-symmetry gaussians placed at the positions selected by the user. This is useful to mimic some antibonding state.

`x, y, z` REAL
as before for `type == "1gauss"`.

`xx, yy, zz` REAL
as before for `x,y,z` but specify the center of the second gaussian used to set up the trial center.

`rloc` REAL
as before for `type == "1gauss"`.

`weight` REAL
as before for `type == "1gauss"`.

`type == "atomic"`
Atomic (pseudo)-orbitals from pseudopotential data-files are used as trial functions. They are specified by the atomic index and the required angular momentum quantum-numbers.

`iatom` INTEGER
the index of the chosen atom (ordering is directly taken from DFT data).

`l, m` REAL
as before for `type == "1gauss"`.

`weight` REAL
as before for `type == "1gauss"`.

4.3 Input for electronic transport calculations

Step 3 : conductor.x
Both bulk and two-terminal calculations use similar input. Labels follow the scheme of Sec. 3.

Input file layout:

```
&INPUT_CONDUCTOR
...
/

<HAMILTONIAN_DATA>
   <ham_1 attr="" />
   ...
   <ham_N attr="" />
</HAMILTONIAN_DATA>
```

Namelist `&INPUT_CONDUCTOR`

`dimL` INTEGER
number of sites in the L-lead.
DEFAULT = 0

`dimR` INTEGER
number of sites in the R-lead.

DEFAULT = 0

dimC
INTEGER
number of sites in the conductor region.
DEFAULT = mandatory

calculation_type STRING
("conductor" | "bulk") determines the kind of calculation to be performed:
"conductor": ordinary transport calculation for a leads|conductor|lead junction
"bulk": transport in a bulk system.
DEFAULT: "conductor"

transport_dir
INTEGER
transport direction according to crystal axis indexing.
DEFAULT = mandatory

conduct_formula STRING
("landauer" | "generalized") "landauer": transport is computed using the standard Landauer formula
"generalized": a generalized Landauer formula accounting for a specific correlation correction is used. This case is experiemntal, see Ref. [24, 25] for more details.
DEFAULT: "landauer"

ne
INTEGER
dimension of the energy grid for transmittance and spectral function calculation.
DEFAULT = 1000

emin
REAL
lower limit [eV] of the energy grid.
DEFAULT = -10.0

emax
REAL
upper limit [eV] of the energy grid.
DEFAULT = +10.0

delta
REAL
small imaginary part used to get off the real axis in the calculation of Green's functions.
DEFAULT = 1.0e-5

nprint
INTEGER
every nprint energy step write to stdout .
DEFAULT = 20

niterx
INTEGER
maximum number of iterations in the calculation of transfer matrices.
DEFAULT = 200

`use_overlap` LOGICAL
If `.TRUE.` reads the overlap matrices from file, otherwise basis orthonormality is assumed (which is by definition the case of Wannier functions).
DEFAULT : `.FALSE.`

`use_correlation` LOGICAL
If `.TRUE.` correlation corrections are read from file and included in the calculation. See also the `datafile_sgm` variable.
DEFAULT : `.FALSE.`

`datafile_C` STRING
Name of the file containing the Wannier Hamiltonian blocks for the conductor region.
DEFAULT: `"mandatory"`

`datafile_L` STRING
Name of the file containing the Wannier Hamiltonian blocks for the L-lead. It is nor required for `bulk` calculations.
DEFAULT: `"mandatory"` if not `calculation_type == "bulk"`

`datafile_R` STRING
as for `datafile_L` but for R-lead.
DEFAULT: `"mandatory"` if not `calculation_type == "bulk"`

`datafile_sgm` STRING
Name of the file containing the correlation self-energy. It is required only when correlation is included in the calculation.
DEFAULT: `"mandatory"` if `use_correlation == .TRUE.`

Card `<HAMILTONIAN_DATA>`

The card `<HAMILTONIAN_DATA>` is mandatory and specifies the details about hamiltonian blocks to be used in transport calculation. It includes a variable number of subtags (XML format) to be used the order shown below. The name and the number of these subcards depend on the `calculation_type` variable:

`if ( calculation_type = "bulk" )`	→	two subcards are needed `<H00_C>`, `<H_CR>`
`if ( calculation_type = "conductor" )`	→	seven subcards are needed `<H00_C>`, `<H_CR>`, `<H_LC>`, `<H00_L>`, `<H01_L>`, `<H00_R>`, `<H01_R>`

Names of the subcards refer to Fig. 2 in Sec. 1.1. Each subcard (tag) may contain a number of attribute, according to the format (XML compliant):

`<H`$_{\texttt{whatever}}$ `cols="" rows="" />`

`cols (rows)` STRING
the string describing which index should be considered to define the columns

(rows) of the specific H submatrix. The format in the string is quite standard, according *e.g.* to the default for printing programs:
"1-3,5,7-9" stands for "1,2,3,5,7,8,9", and so on. The string "ALL" is allowed as well, being equivalent to "1-N_{max}".
DEFAULT = "ALL"

4.4 Input for post-processing calculations

4.4.1 bands.x

Input file layout:

```
&INPUT
...
/

label_1     k1  k2  k3
...
label_N     k1  k2  k3
```

Namelist &INPUT

prefix STRING
the first part of the name of all the files written by the code; it should be equal to the value given in the main calculations.
DEFAULT = mandatory

postfix STRING
the tail of the names of the above mentioned files (useful *e.g.* to distinguish among different calculations having a common part); it should be equal to the value given in the main calculations.
DEFAULT = ""

work_dir STRING
the scratch directory where the massive data files will be written.
DEFAULT = "./"

verbosity STRING
("low" | "medium" | "high")
the level of detail of the textual output file.
DEFAULT = "medium"

nkpts_in INTEGER
number of edge **k**-points defining the directions on which bands will be calculated.
DEFAULT: mandatory

nkpts_max INTEGER
maximum number of interpolated **k**-points. The actucal number of points is calculated in the run and is written in the output file.
DEFAULT = 100

`spin_component` STRING
`( "up" | "down" | "none" )`
define whether the calculation is spin polarized and, if the case, which spin component is treated.
DEFAULT = `"none"`

After the `&INPUT` namelist, a description line for each of the `nspts_in` points must be provided. The format is as follows:

`label` k_1 k_2 k_3

`label` STRING
a string with the conventional name of the **k**-point.

k_1 k_2 k_3 REAL
components of the **k**-point vector in units of crystal reciprocal lattice
(*i.e.* $\mathbf{k} = k_1 * \mathbf{b}_1 + k_2 * \mathbf{b}_2 + k_3 * \mathbf{b}_3$).

4.4.2 plot.x

Input file layout:

```
&INPUT
...
/
```

Namelist `&INPUT`

`prefix` STRING
the first part of the name of all the file written by the code; it should be equal to the value given in the main calculations.
DEFAULT = `mandatory`

`postfix` STRING
the tail of the names of the above mentioned files. It should be equal to the value given in the main calculations.
DEFAULT = `""`

`work_dir` STRING
the scratch directory where the massive data files will be written.
DEFAULT = `"./"`

`wann` STRING
specifies the indeces of the Wannier functions to be plotted. It is a string of format *e.g.* `"1-3,5,7-9"` (analogous to the fmt used to specify pages to standard print utilities)
DEFAULT = `mandatory`

`datatype` STRING
`("modulus" | "real" | "imaginary")`

specifies the type of data plotted:
"modulus": plot the real space square modulus of the WFs.
"real": plot the real part (in real space) of the WFs.
"imaginary": plot the imaginary part (in real space) of the WFs this choice shouldbe intended as a check because WFs are expected to be more or less *real*.
DEFAULT = "modulus"

output_fmt — STRING
("plt" | "txt" | "cube" | "xsf")
Define the format of the output files to be plotted. "plt" (read by GOPENMOL) is binary and smaller than "cube", "xsf" (read by XCRYSDEN) and "txt". While "cube" and "xsf" deal also with non-orthorombic lattices, "txt" is suitable to be converted to further format.
DEFAULT = "plt"

r1min, r1max — REAL
the starting and ending points of the plotting cell along the first direct lattice vector $\mathbf{a}_1$. Units of $\mathbf{a}_1$.
DEFAULT = -0.5, 0.5

r2min, r2max — REAL
as before but for $\mathbf{a}_2$ direction.
DEFAULT = -0.5, 0.5

r3min, r3max — REAL
as before but for $\mathbf{a}_2$ direction.
DEFAULT = -0.5, 0.5

assume_ncpp — LOGICAL
if using norm-conserving pseudopot which are not readable by WANT set this value to .TRUE. in order to avoid their reading.
DEFAULT = .FALSE.

locate_wf — LOGICAL
if .TRUE. moves the WFs in a unit cell centered around the midpoint of the plotting cell. Useful to plot purposes.
DEFAULT = .TRUE.

spin_component — STRING
("up" | "down" | "none")
define whether the calculation is spin polarized and, if the case, which spin component is treated.
DEFAULT = "none"

4.4.3 blc2wan.x

Input file layout:

```
&INPUT
...
/
```

Namelist `&INPUT`

`prefix` STRING
the first part of the name of all the file written by the code; it should be equal to the value given in the main calculations.
DEFAULT = `mandatory`

`postfix` STRING
the tail of the names of the above mentioned files. It should be equal to the value given in the main calculations.
DEFAULT = `""`

`work_dir` STRING
the scratch directory where the massive data files will be written.
DEFAULT = `"./"`

`filein` STRING
the name of the file containing the input operator in the Bloch representation.
DEFAULT = `mandatory`

`fileout` STRING
the name of the file containing the computed operator on the Wannier basis.
DEFAULT = `mandatory`

`ascii` LOGICAL
if `.TRUE.` the output file is written in textual XML format.
DEFAULT = `.FALSE.`

5 What to do when things go wrong?

This section is not at all intended to be complete. Here we report a summary of the most frequent and well-known problems in the day-by-day practice with WanT, and some tentative suggestions to solve them. Please, report any better solution or explaination you find to the maintainer of this manual to make it more detailed.

5.1 When do things go really wrong ?

First it is necessary to understand which behaviors should be considered buggy and which may be conversely related to some failure of the implemented algorithms. This section is devoted to guide the user to understand whether the code is properly running or not.

- **Stops and crashes.**
  When the code stops, it is expected or to have reached the end of the calculation (which is signed in the output file by a summary of the timing) or to print out an error message and give a fortran stop. Any different behavior should be considered a bug and should be reported (obviously it may be related to machine dependent problems, indipendently of WanT).

- **Problems connected to memory usage.**
  Since the code is still serial, no distribution of the memory among the CPU is performed and crashes may result. In these cases, both an error message during allocations or a more drastic error (even segmentation fault) may occur. The largest amount of memory is used by `disentangle.x` (which manage wave-functions), while `wannier.x` does not require so much. Post-processing (`bands.x, blc2wan.x`) do not have special requirement too, while `plot.x` needs a sensible amount of memory, about half of `disentangle.x`.

- **Convergence in disentanglement minimization.**
  The **iterative procedure** in `disentangle.x` is quite robust. Non-monotonic behaviors of the invariant spread are unexpected and are probably related to algorith failures. Although, convergence maybe very slow, especially when some parameters are not properly set (too many empty states in the main energy window, strange numbers of requested WFs...) .

- **Convergence in Wannier localization.**
  In the case of `wannier.x`, convergence is less strightfarward. Particularly for large systems, the behavior of the total spread is typically decreasing for a number of iterations, then it suddenly junps to a higher value and after it keeps decreasing. This cycle may be repeated several times. From the expressions [18] of the spread functional, we see that many evaluation of the immaginary part of complex logarithms (*i.e.* the pases of the arguments, which iin the present case are the overlap integrals) are needed. Since logarith in the complex plane is a multivalued function, sudden junps in the total spread (towards larger values) can be related to changes in the logarith branch (*e.g.* a phase which is increasing from 0 to 2π is suddenly taken to 0 (*i.e.* to a different branch) when corssing the 2π value. The problem becomes more evident when a large number of **k**-points is used for the sampling of the Brillouin zone.

  Such behaviors should be considered ordinary: the user is suggested to increase the maximum number of iterations (50000 steps may be fully acceptable) when running `wannier.x` . Since this problem is mainly connected with large **k**-point mesh, it should luckily leave unaffected large scale calculations (where we usually have large number of bands in a very small BZ).

- **How can I understand when my Wannier functions are well behaved ?**
  Directly from the definition of *maximally localized* WFs, we basically are interested in *localized* orbitals spanning the original subspace of Bloch states. Our measure of localization (the spread functional Ω) is anyway a global property of the WF set as a whole. This means that even if we are reaching lower and lower values of the spread, the set of WFs we found is a good one if *each* single WF is localized.

  If our application of WFs is based on their localization property, then a single WF not properly localized can make the all set useless. This is exactly what happens in the current application to transport (but it is not for instance the case of the calculation of spontaneus polaritation, which is a property of the occupied manyfold).

  Once we attach the idea of *good behavior* to that of localization of each WF, it is possible to identify the following criteria:

  - Typical values for the average invariant spreads are lower than $7 - 10$ Bohr^2. Note that this value is a sort of lower bound for the final spreads.
  - the spread of each WF should be in a reasonable range ($< 15 - 20$ Bohr^2). Note that the current version of WanT fully adopts Bohr units, while older versions (*e.g.* v1.0) use $\mathring{A}^2$ for the spread in the `wannier.x` output file.
  - Since well-localized WFs are expected (by conjecture) to be almost real, their Hamiltonian matrix elements should be nearly real. See the end of `wannier.x` output file where these matrix elements are reported.
  - The spatial decay of the Hamiltonian $H(\mathbf{R})$ on the WF basis is expected to have a nearly exponential decay wrt $\mathbf{R}$ when WF are well-localized.
  - Using the `bands.x` postprocessing, it is possible to interpolate the band structure using the computed $H(\mathbf{R})$ matrices. If WF are well-behaved, bands are usually reproduced using a small number of $\mathbf{R}$ vectors (since the real space decaying of the Hamiltonian). Otherwise this last assumption is not allowed and the band structure typically shows unphysical oscillations.

- **What about the symmetry properties of the WF set ?**
  When computing WFs for the occupied manyfold of valence bands, the total charge, which is fully symmetric, becomes partitioned on that set (the charge is the sum of the WF squared moduli). WFs retain therefore a direct link to the symmetry properties of the crystal. This is no longer guaranteed when we are mixing valence and conduction states in the most general way (in the `disentangle.x` run). A total lack of symmetry in these cases is not the sign of a bug, but may be the sign of some unwanted behavior of the localization process. See Sec. 5.2 in order to try to avoid these results.

5.2 Troubleshootting (sort of)

- Energy window and `dimwin`
  If the `disentangle.x` code complains when starting the run about energy windows or the variable `dimwin`, it means that the window you chosen is probably too small. For each $\mathbf{k}-$point the window must contain a number of bands (called `dimwin`) not smaller than the number of required Wannier functions (`dimwann`).

- The code complains about **k**-points
  The **k**-point checking algorith has been found to be quite stable, and check failures are typically due to errors in calculation setup. The most common problem is related to a non-Monkhorst-Pack **k**-points mesh. The code is not yet able to take advantage of space symmetries (and neither of time reversal) and therefore DFT calculations should be done using a regular grid on the whole Brillouin Zone and not on its irreducible wedge only. Please report other failures if any.

- How can I set the starting Wannier centers ?
  The position and the type of starting centers are very important to speed up the convergence of both `disentangle.x` and `wannier.x`. They are anyway much less effective on the final results of the minimizations, which usually do not depend that much on them. Standard positions for centers are on the atomic sites (you can use both `"1gauss"` and `"atomic"` types) and on the covalent bonds (using `"1gauss"` type with s-symmetry). When using atomic positions, angular channels are usually attributed on the basis of the related atomic orbitals. When the chemical environment is more complex or you are requiring alot of WFs the choice may be more difficult but probably also less effectiveon the minimization speed. Take a look at the test suite to see how simple examples work.

- When reading overlaps or projections the code exits with an error
  The flags `overlaps="from_file"` and `projections="from_file"` (which are silently activated also in the restart procedure) make the code to read overlaps and projections from a datafile of a previous run. At the moment (but hopefully it will be fixed soon) this operation is allowed only if the dimensions of these data are the same as in the current calculation. Basically if you want to use these flags you should not change the energy window, the number of Wannier functions, and on.

- `disentangle.x` does not converge
  This is a quite starnge bahevior: if it only takes so long, please enlarge the maximun number of steps. On the contrary, if it largely oscillates leaving no hope for a convergence, please report your input file to the developers.

- `disentangle.x` converges, but average spread is large
  This usually means that some bands needed to achieve a better localization are out of the energy window which should be enlarged. This operation should be done with care since too large windows will lead to small spread on one side, but the obtained subspace will aquire components on very high energy Bloch states. Usually this must be avoided since we want to represent a physically interesting subspace. Looking at the end of the `disentangle.x` output file when setting the variable `verbosity="high"` you would be able to see all the projections of the Bloch states onto the computed subspace.

- $U^{\mathbf{k}}$ matrices are not unitary
  These failures are detected by `WARNING` keywords in the `disentangle.x` output files and leads to calculation stopping in `wannier.x` or post-processing. If you are dealing with a very large systems (hundreds of atoms) first try to increase the check threshold (`unitary_thr`). Values larger than $10^{-6}, 10^{-5}$ (or even of the same order for small systems) are usually a sign of numeric instability, probably due to porting issues. Please report any failure of this kind.

- Wannier functions does not converge
  The `wannier.x` localization algorith may take so long to reach the convergence. Many large oscillations in the total spread may also happen. If conditioned minimization are not used, be confident and let the code going on in the iterative minimization (restart or enlarge the maximum number of steps). If the minimization takes too long, consider it not converged and look at the following point.

- Wannier functions converge, but they are weird
  This is a very common case. The quality of WFs depends so much on the starting subspace provided by `disentangle.x`. Try to modify it (avoid high energy Bloch state components) properly re-setting the energy window. This is anyway a quite drastic measure. Before doing this you can try to change the starting centers (in this case you should also re-run `disentangle.x` since projections must be re-calculated).

  If you understand that WFs are not well-behaved since they got in strange positions you may wonder to use conditioned minimization and penalty functionals. See Sec. 1.2.3 for the theoretical details of the method. In the current implementation, the conditioning tries to move the centers of WFs as close as possible to the positions of the starting centers provided in the input. Their importance is therefore crucial and dab centers are easily detected since the minimization path is completely crazy. The use of this conditioning is typically useful when the simulation cell includes some vacuum regions.

- Transport calculations have no physical meaning
  Unfortunately the input file for transport calculation (particularly the card related to the real space Hamiltonians) is very complicated. When your transport calculations are completely wrong (weird oscillations or narrow peaks in the transmittance) it is usually due to some error in the input file (*e.g.* the indices of the required submatrices).

- Transport calculations have has some sense but are noisy
  Numerical inaccurancies may be present. Typical effects are boring dips in the transmittance curves. Make sure that your conductor cell is converged wrt the bulk properties of the lead. If not enlarge the cell. Are your principal layer large enough to make their interaction decaying just after nearest enighbors ? Are the energy levels of your conductor and lead calculation the same ? See Sec. 3 about calculation setup to know how to check it.

6 The test suite

WANT package is distributed with a suite of tests which are intended to check the portability of the code and to show its features. Along these lines, reference results as well as detailed explainations are supplied. A discussion on how to fruitfully use the test suite is reported in the following.

HOW TO set up the environment

In order to get started with running tests you should modify the `$TOPDIR/tests/environment.conf` file according to your system. You should provide the location of the executables for DFT calculation (at present, tests are suited for the PWSCF code) and the directory that will be used for temporary files (`scratch`). In the case of parallel machine you are also requested to specify some special commands to run the DFT codes within an MPI environment.

HOW TO run the tests

After environment setting, you can directly start running tests. The script `run.sh` in `$TOPDIR/tests/` (type `./run.sh` to get a short manual) manages all operations on tests. The script should be run with the following arguments:

```
./run.sh -r <what> [<test_list>]
```

`<test_list>` is optional and is the list of the tests we want to perform. A missing list performs all tests. Possible action to be performed (`<what>`) are:

`help`	print the manual
`info`	print detailed info about the implemented tests
`all`	perform all the calculations
`dft`	perform DFT calculations only
`want`	perform WANTcalculations only
`check`	check results with the reference outputs
`clean`	delete all output files and the temporary directory

Tests can also be performed by by cd-ing in the specific test directory and running the local script typing `./run.sh <flag>`. The list of possible `<flag>`'s and a brief explaination is printed by executing `./run.sh` (typical flags with obvious meanings are `scf, nscf, disentangle, wannier`) If you type `./run.sh all` inside a specific test directory, the test will be completely performed. Note that some tests contain more than one WANTcalculation; each of them has its specific run_$suffix.sh script.

HOW TO check the tests

In each test directory you will find a further dir named Reference containing the distributed reference output files. Each `run.sh` script accept the flag `check`: this makes the print a short version of the `diff` between the current output files and the reference ones. If you need more detail (as in case of a clear failure), you can advised to directly check your output with a simple `diff` command:

```
diff myreport.out Reference/myreport.out
```

In doing this remenber that you will usually obtain a large amount of data: since for instance the phases of the wave-functions are almost randomized, the initial steps both in disentanglement and wannier minimizations may be very different.

HOW TO understand the tests

In each test directory you will find a `README` file describing the physics of the test and the results obtained in the calculation. Some tricky and subtle problems about the input files are eventually raised and discussed.

References

[1] A. Calzolari, N. Marzari, I. Souza, and M. Buongiorno Nardelli, Phys. Rev. B **69**, 035108 (2004).

[2] M. Buongiorno Nardelli, Phys. Rev. B **60**, 7828 (1999).

[3] M. Buongiorno Nardelli and J. Bernholc, Phys. Rev. B **60**, R16338 (1999).

[4] M. Buongiorno Nardelli, J.-L. Fattebert, and J. Bernholc, Phys. Rev. B **64**, 245423 (2001).

[5] R. Landauer, Philos. Mag. **21**, 863 (1970).

[6] S. Datta, *Electronic transport in mesoscopic systems*, Cambridge University Press, 1995.

[7] D. S. Fisher and P. A. Lee, Phys. Rev. B **23**, 6851 (1981).

[8] Y. Meir and N. S. Wingreen, Phys. Rev. Lett. **68**, 2512 (1992).

[9] D. H. Lee and J. D. Joannopoulos, Phys. Rev. B **23**, 4988 (1981).

[10] D. H. Lee and J. D. Joannopoulos, Phys. Rev. B **23**, 4997 (1981).

[11] M. Lopez-Sancho, J. Lopez-Sancho, and J. Rubio, J. Phys. F: Metal Phys. **14**, 1205 (1984).

[12] M. Lopez-Sancho, J. Lopez-Sancho, and J. Rubio, J. Phys. F: Metal Phys. **15**, 851 (1985).

[13] F. Garcia-Moliner and V. Velasco, *Theory of Single and Multiple Interfaces*, World Scientific, Singapore, 1992.

[14] F. Garcia-Moliner and V. Velasco, Phys. Rep. **200**, 83 (1991).

[15] A. Baldereschi, Phys. Rev. B **7**, 5212 (1973).

[16] W. Kohn, Phys. Rev. **115**, 809 (1959).

[17] M. Geller and W. Kohn, Phys. Rev. B **48**, 1485 (1993).

[18] N. Marzari and D. Vanderbilt, Phys. Rev. B **56**, 12847 (1997).

[19] S. F. Boys, Rev. Mod. Phys. **32**, 296 (1960).

[20] E. Blount, Surf. Sci. **13**, 305 (1962).

[21] P. L. Silvestrelli, N. Marzari, D. Vanderbilt, and M. Parrinello, Solid State Commun. **107**, 7 (1998).

[22] L. Bernasconi and P. A. Madden, J. Mol. Struct. **544**, 49 (2001).

[23] I. Souza, N. Marzari, and D. Vanderbilt, Phys. Rev. B **65**, 035109 (2002).

[24] A. Ferretti, A. Calzolari, R. Di Felice, F. Manghi, M. J. Caldas, M. Buongiorno Nardelli, and E. Molinari, Phys. Rev. Lett. **94**, 116802 (2005).

[25] A. Ferretti, A. Calzolari, R. Di Felice, and F. Manghi, Phys. Rev. B **72**, 125114 (2005).