ELECTRA User Guide

Patrizio Graziosi [1,2] and Neophytos Neophytou[2]

CNR – ISMN, via Gobetti 101, 40129, Bologna

School of Engineering, University of Warwick, CV4 7AL, Coventry

1. Introduction

The *ELECTRA* simulator solves the Boltzmann transport equation (BTE) for charge transport using the wave-vector and energy dependent momentum relaxation time approximation. The simulator computes the charge transport coefficients. It considers charge carrier scattering with phonons, ionized dopants, and alloy scattering [1, 2].

ELECTRA takes as input the electronic bandstructure and certain scattering parameters as detailed below, and returns the charge transport coefficients electrical conductivity (σ), Seebeck coefficient (S), thermoelectric power factor (PF = σS^2), and electronic part of the thermal conductivity (κ_e) as function of Fermi level (E_F) position and temperature (T). These are computed using the transport distribution function (TDF) within the Linearized Boltzmann Transport equation. The overall energy dependent relaxation time and mean-free-path (MFP) are computed as well, also for each scattering mechanism separately. All these quantities are computed as rank-2 tensors.

The purpose of this manual is to introduce the physics behind the functionality of the code, the several different input options that the user has, and provide a few operational examples to help the user become more familiar with the code and use it effectively.

NOTES

1. At present, the code is tested and validated only for 3D bandstructures and this is the only possible option, while 2D and 1D cases are under development and testing.

2. The code is written in MATLAB® and makes use of the Parallel Computing Toolbox™. A version ported in C language exists, but it is still under test.

Terms of Use

ELECTRA is an open-source code, released under the GNU General Public License, https://www.gnu.org/licenses/gpl-3.0.en.html, and developed under the Marie Curie project *GENESIS* - Generic semiclassical transport simulator for new generation thermoelectric materials, project ID 788465, funded by the European Commission under the MSCA actions.

We shall appreciate if any work done using the *ELECTRA* code will cite reference **[1]** and, if bipolar calculations are performed, also **[2]**.

2. Physical concepts

The *ELECTRA* code is based on the semiclassical description of charge transport where the charge carrier dynamics are described by the linearized BTE. The carrier interaction with the scatterers (phonons, ionized impurities, alloy defects) is quantum mechanically described by Fermi's Golden Rule. [1 - 4]. Here we present the theory that simulator is built upon [5]. The charge transport coefficients, electrical conductivity, σ, and the Seebeck coefficient, S, are rank-2 tensors in the Cartesian components ij and have the form:

$$\sigma_{ij(E_{\mathrm{F}},T)} = q_0^2 \int_E \; \Xi_{ij}(E) \left(-\frac{\partial f_0}{\partial E}\right) dE, \tag{1a}$$

$$S_{ij(E_{\mathrm{F}},T)} = \frac{q_0 k_B}{\sigma_{ij}} \int_E \; \Xi_{ij}(E) \left(-\frac{\partial f_0}{\partial E}\right) \frac{E-E_F}{k_{\mathrm{B}}T} dE, \tag{1b}$$

$$\kappa_{e_{ij}} = \frac{1}{T} \int_E \; \Xi_{ij}(E) \left(-\frac{\partial f_0}{\partial E}\right) (E - E_{\mathrm{F}})^2 \, dE - \; \sigma S^2 T \tag{1c}$$

where $\Xi_{ij}(E)$ is the Transport Distribution Function (TDF) defined below in equation (2), E_{F}, T, q_0, k_{B}, and f_0, are the Fermi level, the absolute temperature, the electronic charge, the Boltzmann constant, and the equilibrium Fermi distribution, respectively. In Eq. (1c) the $\sigma S^2 T$ term is important only in bipolar calculations. [6] The simulator performs a trapezoidal integration over the energy. A step of 2 to 3 meV is generally sufficient, but the user can choose a different value. The most critical aspect is the step in the *k*-space mesh, where a step of at most ~ 0.02 in units of π/a is recommended for the material's bandstructure under consideration (see validation section for more details). In this _v0 version of the code, band interpolation is not implemented for the bandstructure, however a linear interpolation is performed inside each mesh element to construct the constant energy surfaces.

The TDF is central to the computation, and is computed as a surface integral over the constant energy surfaces, $\mathfrak{L}_E^n$, for each band, and then summed over the bands, as [1, 2, 5]

$$\Xi_{ij(E,E_{\mathrm{F}},T)} = \frac{s}{(2\pi)^3} \sum_{\boldsymbol{k},n}^{\mathfrak{L}_E^n} \left| v_{i(\boldsymbol{k},n,E)} v_{j(\boldsymbol{k},n,E)} \right| \tau_{i(\boldsymbol{k},n,E,E_{\mathrm{F}},T)} \frac{dA_{\boldsymbol{k}_{\mathfrak{L}_E^n}}}{|\vec{v}_{(\boldsymbol{k},n,E)}|} \tag{2}$$

where $\boldsymbol{k}_{\mathfrak{L}_E^n}$ represents a state on the surface $\mathfrak{L}_E^n$ and $dA_{\boldsymbol{k}_{\mathfrak{L}_E^n}}$ is its corresponding surface area element, computed as explained below in the "Extraction of the constant energy surfaces" section. [1]. $v_{i(\boldsymbol{k},n,E)}$ is the i-component of the band velocity of the transport state, computed as the gradient of the bandstructure, calculated using the central finite difference method. $\tau_{i(\boldsymbol{k},n,E)}$ is its momentum relaxation time (combining the relaxation times of each scattering mechanism using Matthiessen's rule), $\frac{dA_{(\boldsymbol{k},n,E)}}{|\vec{v}_{(\boldsymbol{k},n,E)}|}$ its density-of-states (DOS), [1] and s the spin degeneracy, for which $s = 2$ is used when the two spin sub-bands are degenerate, i.e. when the material is not ferromagnetic. In order to perform the surface integrals in Eq. (2), the bandstructure, generally computed on a defined mesh in the reciprocal space as an $E(\boldsymbol{k})$, is transformed in a $\boldsymbol{k}(E)$, describing all the k-points at a given energy in an underlying uniform energy grid. This describes the constant energy surfaces, (for example the well-known ellipsoids in common semiconductors), and every summation becomes a surface integral with $\mathfrak{L}_E^n$ being the integration domain. The points on these surface are the transport states, uniquely defined by the ($\boldsymbol{k}$,n,E) triad.

Extraction of the constant energy surfaces

The code offers two possibilities to construct the constant energy surfaces. The first is based on the Delaunay Triangulation (DT) performed locally on the k-space mesh elements which are crossed by the surface of the constant energy of interest. The second is based on a scan of the nearest neighbour points on the k-mesh. Although the latter is an approximation because it detects only the points along the edges of the k-mesh 3D elements, it is around 15 to 30 times faster, but without noticeable penalty in the results compared to the DT method. For isotropic bandstructures the discrepancies are at the level of the numerical noise, while for anisotropic

bands, in terms of the density of states (DOS), the differences increase from zero at the band edge to ~ 1 % around 0.2 eV from the band edge and to ~ 4 % around 0.5 eV from the band edge.

1. Local Delaunay Triangulation (DT)

In the case of DT, the reciprocal unit cell is scanned to detect the mesh elements which are crossed by a certain constant energy surface, for each energy (and each band). The code checks the energy of all the vertex pairs (12 pairs in 3D) for the condition if at least one of these pairs consists of a vertex with energy above the value under consideration, and one below. All the relevant elements are then triangulated using the 'Delaunay' scheme implemented in Matlab®. The $E(\mathbf{k})$ is approximated to be linear between these points, and thus a $\mathbf{k}_i$-point of energy E_i between the vertexes v1 and v2 is selected on each edge of the element of the triangulation as [7]:

$$\mathbf{k}_i = \mathbf{k}_{v1} + (\mathbf{k}_{v2} - \mathbf{k}_{v1}) \frac{E_i - E_{v1}}{E_{v2} - E_{v1}} \tag{3.}$$

Each tetrahedron will have three or four points (depending on whether the vertexes above/below the energy under consideration are one or two as shown in **Figure XX** below). These define a surface computed with Heron's formula: $A = \sqrt{s_p(s_p - l_1)(s_p - l_2)(s_p - l_3)}$, where A is the triangle area, s_p is the semi-perimeter and l_i are the triangle sides. When the tetrahedron has four points, the surface element is twice the average of the areas of the four defined triangles. Such surface is the $\mathrm{d}A_{\mathbf{k}}$ described in Eq. (2) and is used in the integration when computing the density of states. When we evaluate the anisotropic scattering rates, the code needs the coordinates of the points associated to a given surface element, because the scattering rate depends on the wave-vector difference between the final and initial states. The code finds the barycentre of the surface element and this is the $\mathbf{k}$-point associated to the given surface element as shown by the yellow cross in **Figure 1**.

Three points crossing

(a)
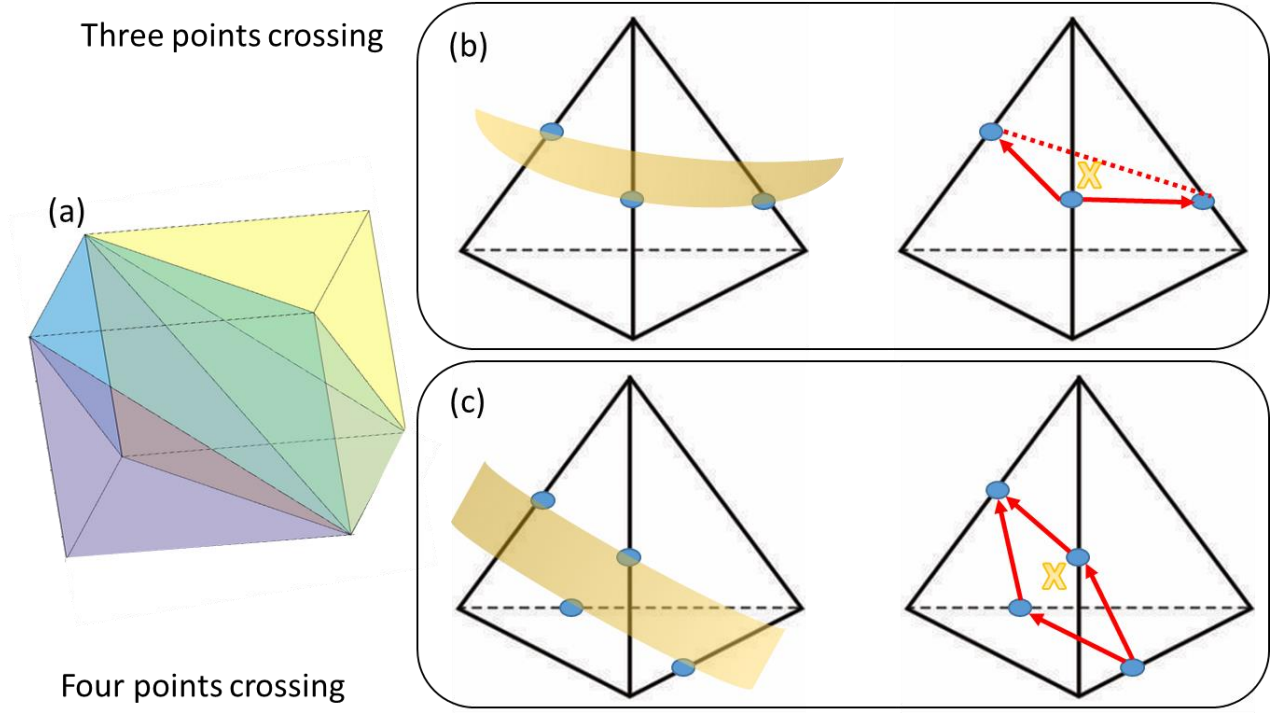
(b)

(c)

Four points crossing

Figure 1: (a) Delaunay Triangulation performed using Matlab for a 3D mesh element. (b) and (c) show the cases where the constant energy surface of interest crosses a tetrahedron in three points (b) or four points (c). A part of the energy surface is depicted in yellow and the crossing points in blue. The surface element will be the area of the triangle in (b) or the parallelogram in (c) defined by the red arrows. Its associated element coordinates are represented by the coordinates of the barycenter shown by the yellow cross.

2. Fast nearest neighbor scan method (fast k-space scan)

In this method, the code scans all the k-points and checks for their nearest neighbours along the edges of the k-mesh. This is depicted in **Figure 2(a)**, where the k-point of interest is shown in orange and its nearest neighbours in green. If the k-point of interest and a neighbour have energies one above and one below the energy value of interest, the code selects a new k-point along the edge of the mesh assuming a linear $E(k)$ between the two points, as above in Eq. (3).

This new point becomes the k-point at the energy of interest for the scattering rate evaluation – the purple cross in **Figure 2(b)**. In this way the code does resolve a constant energy surface element, and the constant energy surface is never constructed, but acquires a collection of points on the energy surface of interest. Then, the code assigns an effective dA_k surface element area value to each point to allow us to extract the density of states associated with that k-point. For this, after grouping the dispersion points on the constant energy surfaces, the surface in the neighbourhood of each k-point is explored to detect its neighbours on the surface. This is done in a radius of $\sqrt{2}dk$ where dk is the distance between adjacent points in the initial mesh, the one used in the DFT bands to calculate the bands. In the case where the k-mesh discretization is different in the different coordinates, dk is the average of the different dk. Then, the code calculates the average distance between the given point and its detected neighbours, $< \Delta k >$. The surface element associated to the k-point is approximated by a circle of radius half the average distance to the neighbouring points, i.e. $dA_k = \pi \left(\frac{<\Delta k>}{2} \right)^2$. These are represented in purple in **Figure 2(b)**, where the neighbouring points on the surfaces are represented by black crosses. Excellent agreement is achieved between this method and the DT method, as well as excellent match to analytical parabolic and non-parabolic results (see validation section and reference [1]).
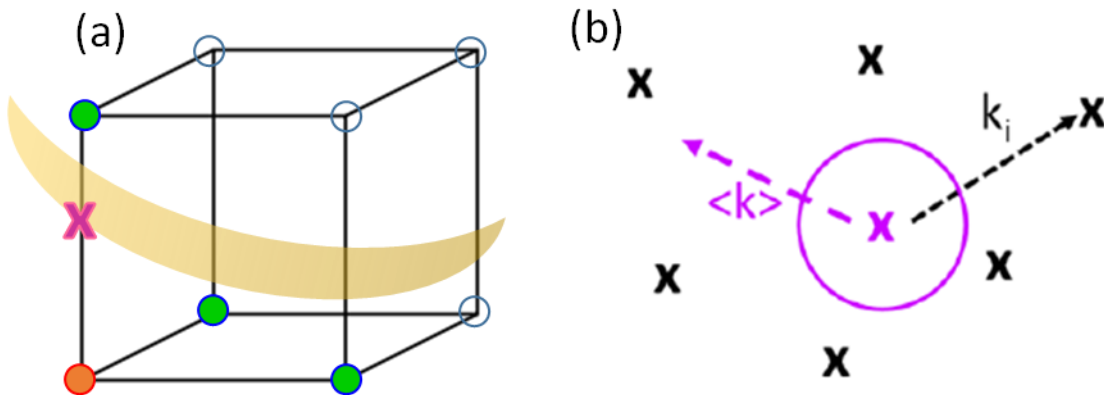
Figure 2: (a) A k-point in a 3D k-mesh element (orange) with some of its nearest neighbours (in green). When a constant energy surface (yellow) crosses the edge between the point under consideration and its neighbor, the crossing point (purple) is selected. (b) The conceptual schematic for the formation of the dA_k, where the specific k-point under consideration is in purple, its neighbours in back, k_i is the distance between the point in purple and the ith-neighbour and $<k>$ is the average distance. The surface element is represented by the purple circle.

In both the schemes, the velocity, and hence the DOS, is computed using the approach by Lehmann and Taut. **[8]**

Scattering rate calculation

For each transport state and each scattering mechanism m_s, the corresponding relaxation time $\tau_{i(\boldsymbol{k},n,E)}{}^{(m_s)}$ is defined as:

$$\frac{1}{\tau_{i(\boldsymbol{k},n,E)}{}^{(m_s)}} = \frac{1}{(2\pi)^3}\sum_{\boldsymbol{k}'} |S_{\boldsymbol{k},\boldsymbol{k}'}{}^{(m_s)}| \left(1 - \frac{v_{i(\boldsymbol{k}')}}{v_{i(\boldsymbol{k},n,E)}}\right) \tag{4}$$

where the sum runs over all the allowed final states $\boldsymbol{k}'$. This formalism assumes that the scattering does not change the spin of the carrier, however, if it does, then the rate needs to be doubled (not included in this version of the code). $|S_{\boldsymbol{k},\boldsymbol{k}'}|$ is the transition rate between the initial $\boldsymbol{k}$ and final $\boldsymbol{k}'$ states, computed as detailed by Eq. 5 below. The $\left(1 - \frac{v_{ij(\boldsymbol{k}')}}{v_{ij(\boldsymbol{k},n,E)}}\right)$ term is an approximation for the momentum relaxation time, [**9, 10**] which is the type of relaxation time that matters when computing the transport coefficients. [**4**] The simulator computes the scattering rates using Fermi's Golden Rule. [**3,4**] Equations 5 below report are the actual equations used by the code:

$$|S_{\boldsymbol{k},\boldsymbol{k}'}{}^{(ADP)}| = \frac{\pi}{\hbar} D_{ADP}^2 \frac{k_B T}{\rho v_S^2} g_{\boldsymbol{k}'} \tag{5a}$$

$$\left|S_{\boldsymbol{k},\boldsymbol{k'}}{}^{(ODP)}\right| = \frac{\pi D_{ODP}^2}{\rho\omega}\left(N_\omega + \frac{1}{2} \mp \frac{1}{2}\right)g_{\boldsymbol{k'}} \tag{5b}$$

$$\left|S_{\boldsymbol{k},\boldsymbol{k'}}{}^{(IVS)}\right| = \frac{\pi D_{IVS}^2}{\rho\omega}\left(N_\omega + \frac{1}{2} \mp \frac{1}{2}\right)g_{\boldsymbol{k'}} \tag{5c}$$

$$\left|S_{\boldsymbol{k},\boldsymbol{k'}}{}^{(POP)}\right| = \frac{\pi e^2 \omega}{|\boldsymbol{k}-\boldsymbol{k'}|^2 \varepsilon_0}\left(\frac{1}{k_\infty} - \frac{1}{k_s}\right)\left(N_{\omega,BE} + \frac{1}{2} \mp \frac{1}{2}\right)g_{\boldsymbol{k'}} \tag{5d}$$

$$\left|S_{\boldsymbol{k},\boldsymbol{k'}}{}^{(IIS)}\right| = \frac{2\pi}{\hbar}\frac{Z^2 e^4}{k_s^2 \varepsilon_0^2}\frac{N_{imp}}{\left(|\boldsymbol{k}-\boldsymbol{k'}|^2 + \frac{1}{L_D^2}\right)^2}g_{\boldsymbol{k'}} \tag{5e}$$

$$\left|S_{\boldsymbol{k},\boldsymbol{k'}}{}^{(Alloy)}\right| = \frac{2\pi}{\hbar}\Omega_c x(1-x)\Delta E_G^2 g_{\boldsymbol{k'}} \tag{5f}$$

Above, ADP stands for 'Acoustic Deformation Potential' and represents the elastic scattering between charge carriers and acoustic phonons. ODP stands for 'Optical Deformation Potential' and describes the charge carrier inelastic scattering with non-polar optical phonons. IVS stands for Inter-Valley Scattering and is specific for the inelastic intervalley scattering. POP stands for 'Polar Optical Phonon' and describes the inelastic scattering of charge carriers with polar phonons. IIS stands for 'Ionized Impurity Scattering' and describes the elastic scattering rate due to the ionized dopants. "Alloy" represents the alloy scattering due to intrinsic disorder in alloys or solid solutions. $\boldsymbol{k}$ and $\boldsymbol{k'}$ are the wave vectors of the initial and final states. The variables that appear in Eqs. 5 are as follows: D_{ADP}, D_{ODP}, D_{IVS} are the deformation potentials for the ADP, ODP, and IVS mechanisms. ρ is the mass density, v_s the sound velocity, ω the dominant frequency of optical phonons, considered as constant over the whole reciprocal unit cell, which has been validated to be a satisfactory approximation, [11] and N_ω is the phonon Bose-Einstein statistical distribution. e is the electronic charge, ε_0 the vacuum dielectric constant, k_s and k_∞ the static and high frequency relative permittivities, Z the electric charge of the ionized impurity, and N_{imp} is the density of the ionized impurities. $g_{\boldsymbol{k'}} = \frac{dA_{(\boldsymbol{k'},n,E)}}{|\vec{v}_{(\boldsymbol{k'},n,E)}|}$ is the DOS of the final state. The final state's energy is the same as that of the initial state under elastic scattering, while for inelastic processes it is increased/decreased by $\hbar\omega$ for

absorption/emission processes, respectively, represented by "–" and "+"in Eqs. 5b-5d. $L_D =$ $\sqrt{\frac{k_s \varepsilon_0}{e}\left(\frac{\partial n}{\partial E_F}\right)^{-1}}$ is the screening length with E_F being the Fermi level and n the carrier density. [1] Ω_c is the volume of the primitive cell, x the fraction of one of the alloy elements, and ΔE_G the difference between the energy gap of the two constituent materials that form the alloy. [3, 9] Rigorous studies show that the ΔE_G can be substituted by an effective scattering potential in order to better fit experimental values. [9] However, the code uses the ΔE_G term for general use for unknown, not yet synthetized or characterized, materials.

The *ELECTRA* simulator is targeted towards the predictive modelling of thermoelectric properties of materials and it is validated to room and higher temperatures. It considers that all dopants are ionized, and does not capture the carrier freeze out region. Note that the equipartition theorem at the basis of Eq. 5a does not hold at low temperatures. Screening in the electron-phonon scattering processes can be implemented for the POP mechanism only, where the screening factor $\frac{|k-k'|^2}{\left(|k-k'|^2+\frac{1}{L_D^2}\right)^2}$ can be added. The screening in the carrier-phonon scattering event has a variable effect on the transport properties according to the bandstructures features but , it increases the computational complexity (as the scattering rates become Fermi level dependent), so the user can choose whether considering this part or not.

Calculation of the momentum exchange vector: The POP and IIS mechanisms are anisotropic as they depend on the distance in the k-space between the initial and final states $|k - k'|^2$. To compute this momentum exchange vector, the simulator takes into consideration every final state's k-point position on all neighbouring reciprocal unit cells (as shown in **Figure 3**), and then the minimum distance from the initial point is used. For this, as shown in the 2D schematic of **Figure 3**, the final point (red bullet), is shifted by the reciprocal lattice vectors in all possible directions, creating the green bullets **Figure 3**. Then the code considers the closest

distant to the initial point (blue bullet). This is necessary, because for example two k-points that are located at opposite edges of the Brillouin zone (BZ) are actually very near if the equivalent points in the nearest neighbour BZs are considered. **Figure 3** shows a 2D schematic of this, where the cell with bold edges represents the cell used in the simulations and the other cells are the equivalent ones. The blue point is the initial point, the red point is the nominal final point, the green points are all the equivalent final points in other reciprocal unit cells. The physical scattering vector is the green one, and not the red one.
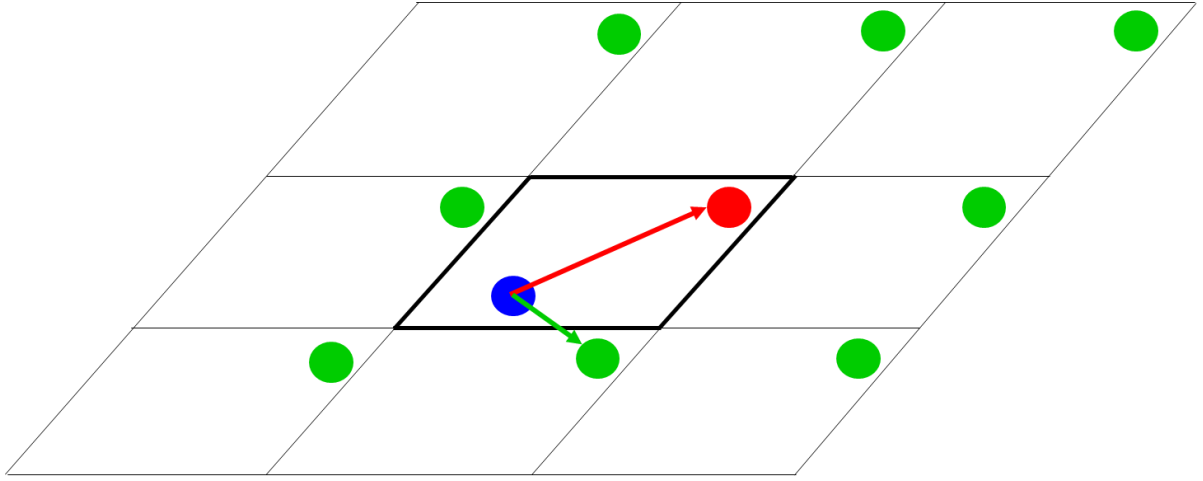
Figure 3: 2D schematic of the reciprocal unit cell used in the calculation, with bold edges, and its equivalent cells around it. The initial k-point is in blue and the final k-point in red. The equivalent final k-points are shown in green, obtained by translating the red point by one reciprocal lattice vector in all possible directions. In the anisotropic POP and IIS scattering mechanisms, all the equivalent k-points are explored, and the final k-point considered is the one closest to the initial.

Bipolar transport: When bipolar transport is considered, the TDF is constructed as in Eq. (2) for the whole energy range of interest, accounting for both the conduction and valence bands. Then the transport coefficients are computed by the integrals in Eq. (1) over the whole

energy range. With regards to screening in the case of bipolar transport, since the screening arising from majority and minority carriers is additive, [**3, 12**] the simulator computes the screening lengths for both the majority and minority carriers ($L_{D,maj}$ and $L_{D,min}$) which are then summed up as: $L_D^2 = L_{D,maj}^2 + L_{D,min}^2$. Note that consideration of unipolar transport for each band and then a merge of the results from the two bands by utilizing the usual relations is not accurate, and we recommend the combined full energy and band type calculation. [**6**]

The bipolar transport calculation can be done in two ways. (i) by defining one type of majority carriers; this is like a unipolar calculation but considering that there are also the minority carriers. (ii) by considering both type of doping, i.e. electrons as majority carriers and then holes as majority carriers, in the same calculation.

Constant relaxation time approximation (CRTA): The simulator can consider the constant relaxation time approximation (CRTA) and the constant mean-free-path approximation (CMFP). The user can choose the CRTA as an input option and set the value of the relaxation time, in fs units. When the CRTA option is selected, the CMFP calculations are also automatically performed with a constant mean free path value of 5 nm, commonly employed in the CMFP literature. [**13**]

Additional computed quantities

The code also computes an overall energy-dependent relaxation time as follows, where i is a cartesian coordinate,

$$\tau_{i(E,E_F,T)} = \frac{\sum_{k,n}^{\aleph_E^n} \frac{1}{\tau_{i(k,n,E,E_F,T)}} DOS_{(k,n,E)}}{\sum_{k,n}^{\aleph_E^n} DOS_{(k,n,E)}} \tag{6}$$

and an overall energy dependent mean-free-path as

$$\lambda_{i(E,E_{\mathrm{F}},T)} = \frac{\sum_{k,n}^{\varepsilon_E^n}\left|v_{i(k,n,E)}\tau_{i(k,n,E,E_{\mathrm{F}},T)}\right|DOS_{(k,n,E)}}{\sum_{k,n}^{\varepsilon_E^n}DOS_{(k,n,E)}}.\tag{7}$$

The TDF, Eq (2), overall relaxation time, Eq. (6), and mean-free-path, Eq. (7), are computed

in addition for each individual band and each separate phonon/alloy scattering mechanism, at

all the entered temperatures and, for the case of POP when the charge screening is selected,

also each E_{F}.

Implemented scattering types:

As general indications, the scattering mechanisms operate as follows with **Figure 4**

indicating intra- and inter-band transitions.

- `ADP`, Acoustic Deformation Potential, elastic scattering with acoustic phonons. It is

    treated as *intra*-band (JUST SAY HERE AND BELOW – process (1) in **Figure 4**).

- `ADP_IVS`, ADP as above, but both *intra-* and *inter-* band scattering transitions are

    considered (processes (1) and (3) in **Figure 4**). ADP and ADP_IVS are mutually

    exclusive, thus, if both are selected, the code will choose ADP.

- the `scattering vector restriction` option restricts the ADP and ADP_IVS

    scattering to only the points whose distance is lower than $1/q_c$ of the reciprocal unit cell

    size, where the cutoff threshold q_c can be specified by the user as `'scattering`

    `vector cutoff'` in the input file. For instance, if the user enters '10' as for this

    value, only the points that are closer than one tenth of the reciprocal unit cell size will

    be considered for ADP – in this case, the equivalent points in neighbouring cells are

    considered, as in the case for IIS and POP. This option considers the fact that acoustic

    phonons have generally low momentum and cannot easily allow electronic transitions

    that require large momentum in *k*-space.

- `ODP,` non-polar Optical Deformation Potential, describes the inelastic electron scattering with non-polar optical phonons. It is considered as *both intra-* and *inter-* band, when the `multivalley` flag is `'no'` (processes (2) and (4) in **Figure 4**), and *intra-* band only, when the `multivalley` flag is set to `'yes'` (process (2) in **Figure 4**). In the "Basic" GUI operation the `multivalley` flag is automatically set to 'no'.

- `POP`, Polar Optical Phonon: It describes the long range, short wave vector, interaction with longitudinal polar optical phonons using the Frölich approach. It is considered as *intra-* and *inter-* band (processes (2) and (4) in **Figure 4**). See reference [**14**] for more details.

- `IIS`, Ionized Impurity Scattering: Describes the scattering with the ionized impurities mediated by the Coulomb potential. Although usually considered as *intra-* band and in the 'Basic' mode it is treated as *intra-*band, process (1) in **Figure 4**, in the 'Advanced' mode the user can choose to have it as *inter-*band as well, process (3) in **Figure 4**.

- `Alloy`: This term considers the carrier scattering related to the intrinsic disorder that appears in alloys or solid solutions, [**3**] and is set to allow both *intra-* and *inter-* band transitions, processes (1) and (3) in **Figure 4**.

- `IVS`: This stands for InterValley Scattering, which describes the *inter-*band inelastic scattering, process (4) in **Figure 4**. It allows several different processes with different deformation potentials, but only the "Advanced" GUI mode allows it. In the "Basic" mode, this is lamped in the ODP, but in that case both *inter-* and *intra-*band transitions are allowed.

- `overlap_integralS.analytical:` Computes the overlap integrals between wavefunctions using the analytical equations of reference [**15**]. When selected, the code will compute the overlap integral analytically and will add this term to the scattering rate. [**3, 15**] Otherwise, the overlap integrals are considered to be one.
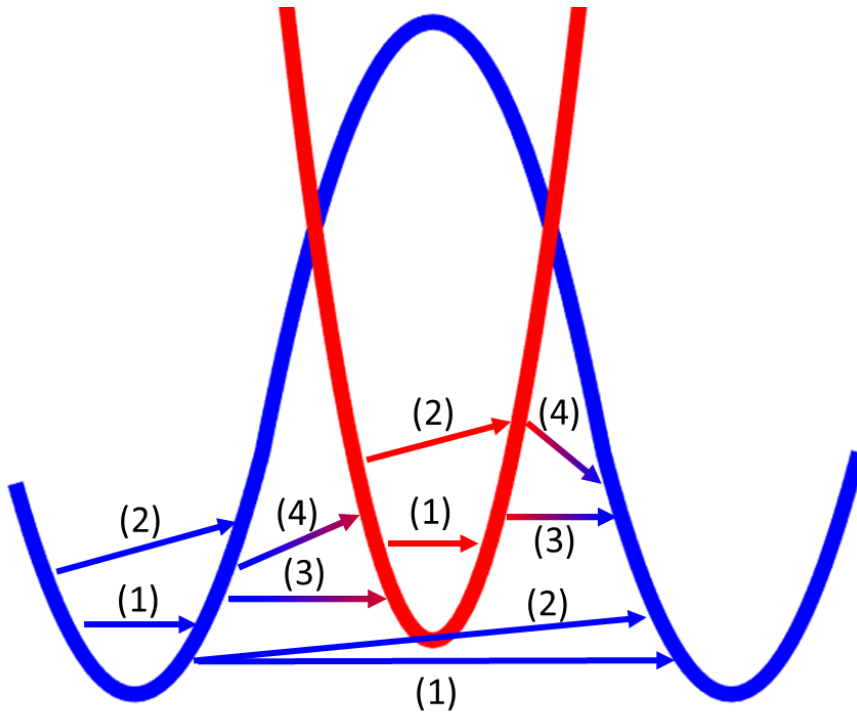
Figure 4: Schematic of two bands (one with many valleys, in blue, and one single valley, in red) with the four type of allowed transitions: (1) elastic intra-band; (2) inelastic intra-band, (3) elastic inter-band, (4) inelastic inter-band.

In **Figure 4** we pictorially show the different scattering transitions as:

    Type (1): ADP, IIS, Alloy

    Type (2): ODP

    Type (3): ADP, IIS, Alloy – the first two only if selected by the user

    Type (4): ODP (only if '`multivalley`' is not selected), IVS

3. Installation and execution

The *ELECTRA* simulator is written in MATLAB® and is preferable that the user has MATLAB® installed on their own machine, while MATLAB® version R2019a or higher are recommended. An amount of memory around few Gb per CPU. The code can be executed in parallel on multiple CPUs on a single node and scales well when executed on at least 32 CPUs. The code diagram is shown in the schematic of **Figure 5a** and examples of how to launch the code on HPC facilities are given in Section 8. To execute the code, the user prepares on their own machine the input data and code instructions, then moves them in an HPC cluster where the code can be executed using a dedicated script (e.g. a bash script is given in Section 8). After the simulation is finished, the saved data are imported in the user's machine for data analysis and post-processing.

The user needs to compose the 'input' file and launch the 'run' file. These are located in the working directory, where the simulation results will be saved. The working directory has two sub-directories. One is named 'Code', and contains all the simulator files, except the run. The other directory is named 'Data' and contains the bandstructure and scattering parameters. A screen shot of the working directory is shown in **Figure 5b**. The Input script contains the input information, alternatively, the GUI will compose and Input file which contain the same information. The 'run' file launches the simulation. When the directory is on a supercomputing cluster, it will also contain the script that launches the 'run' file in the HPC node. Depending on the system where *ELECTRA* is run, the user will launch the 'run_Linux' or the 'run_Windows' file. For Mac, the Linux option must be used.
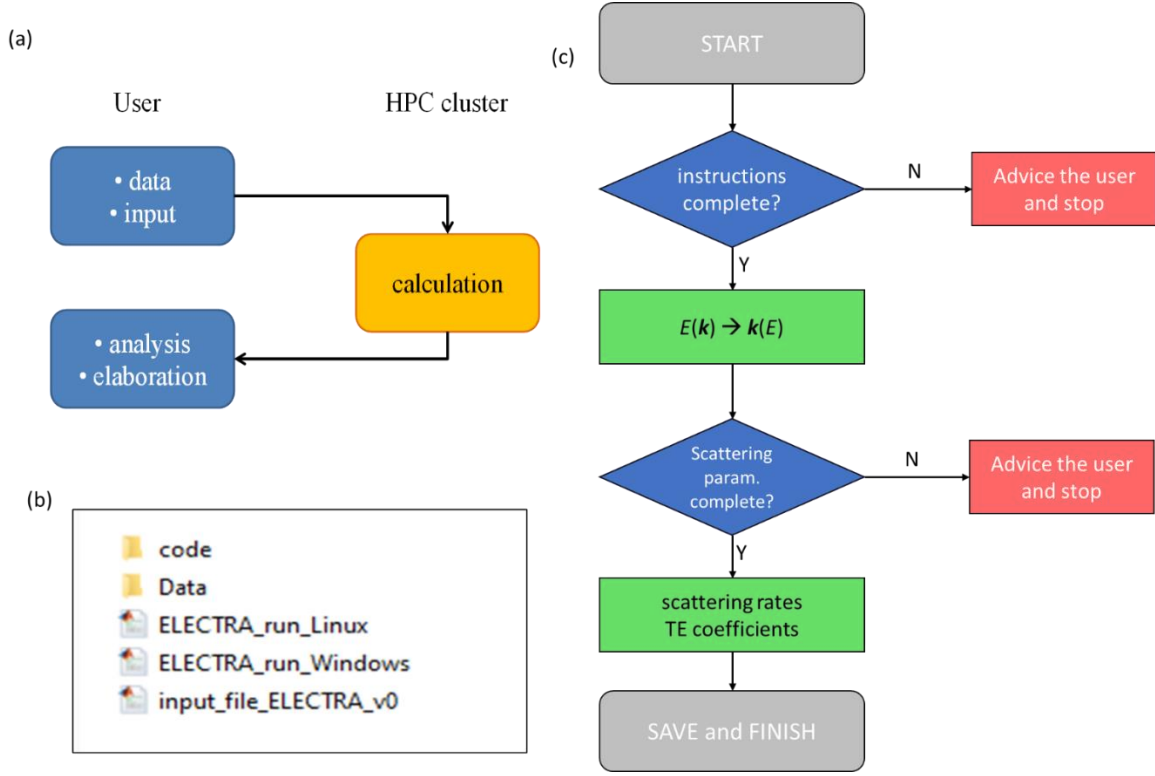
Figure 5: (a) Schematic of the code execution on an HPC cluster; (b) Screen shot of the code working folder; (c) Flow chart of the code.

The user needs to enter three files:

i. The transport calculation specifics, contained in the `Inputs.mat` file composed with the input GUI, or contained in the `input_file_ELECTRA_v0.m` file which is written by the user as described in Section 4. <u>Important</u>: when the `input_file_ELECTRA_v0.m` is present, the code reads it so, when the user uses the GUI, this file must be removed.

ii. The electronic structure, usually computed by DFT, contained in the `Ek_'material_name'.mat` file. A routine for the extraction of this from the.bxsf file are available from one of the authors, PG. This enables compatibility with several DFT codes.

iii. The scattering parameters, contained in the

`scattering_parameterS.'material_name'_3D.mat` file composed

with the scattering GUI, or in the

`scattering_parameterS.'material_name'_3D.m` file described in

Section 6.

The code flow chart is conceptually shown in **Figure 5c**. After the start of the simulation, the code checks that all instructions are consistent, and then extracts the constant energy surfaces – $k(E)$. Then, it checks if all the necessary scattering parameters are present. The scattering rates are then calculated and afterwards the TE coefficients are computed.

4. `input` file preparation

The 'input' file contains a set of options that are flagged as 'yes' or 'no', plus word or numerical instructions: the material name, the range of Fermi levels and Temperatures, and the width of the energy step for the integration. The input file can be prepared both as a script or as a .mat file using the Graphical User Interface (GUI). The GUI allows simplicity and flexibility, and has two options, the "Basic" and the "Advanced". Both are described below. The GUI prepares automatically an 'input.mat' file which will be moved to the HPC facility. The 'input' file is not user readable, but it is straightforwardly composed based on the user's selection in the GUI.

Alternatively, the user can write an 'input' text file. It is preferable that the user modifies the default example provided in the Appendix for the case of the HfNiSn half-Heusler compound. It is essential that the file name is 'input_file_ELECTRA_v0.m'. The code searches for the input instructions in the working directory. In the case where both an 'input' script file and an 'input.mat' file exist, the code will prioritize the .m script file. Thus, when the user composes the 'input.mat' file using the GUI, the script must not be present in the working directory.

Thus, the user can choose if using the GUI or not. If the user uses the GUI, then the instructions below are followed. If the user prefers to use the script, the available options are the same as the advanced GUI, the numerical entries must be entered as vectors, the options will be defined as 'yes' or 'no'; an example is provided.

The GUI – Basic Mode

The screen shot for the Basic GUI mode is shown below, **Figure 6**. The code considers all quantities in International System units excepted the energy which is in eV.
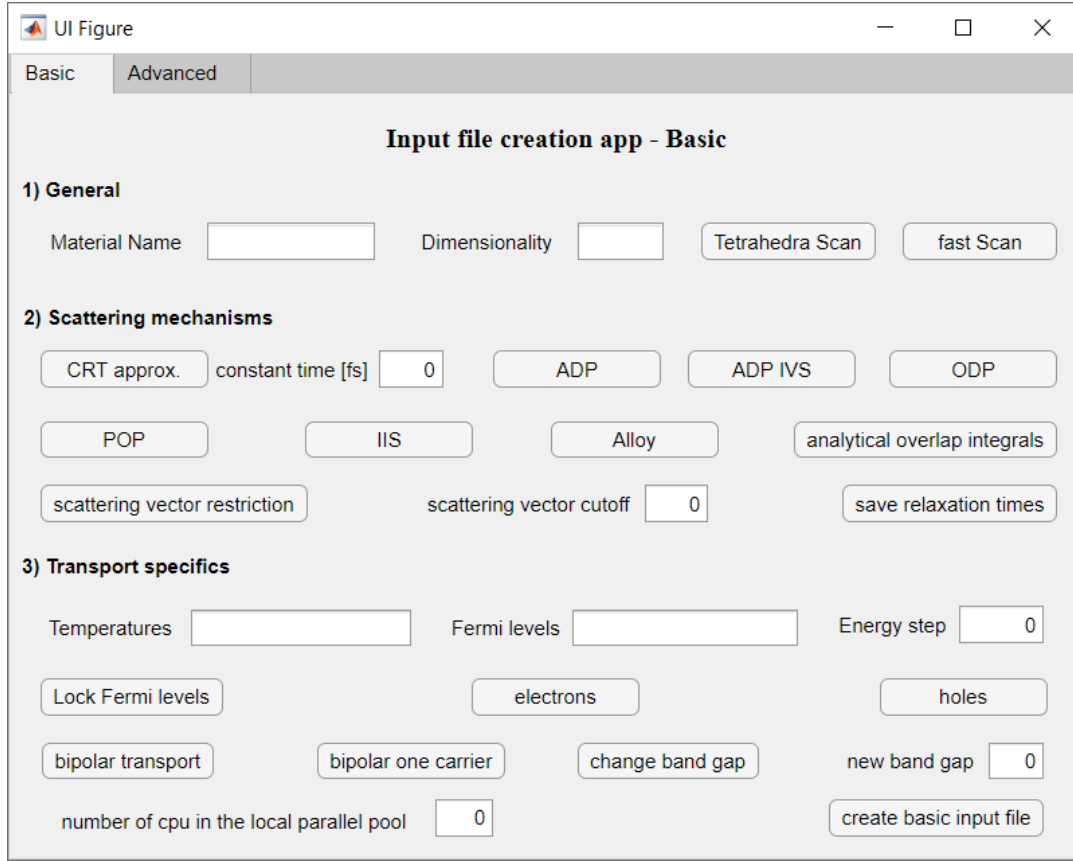
Figure 6: GUI for the Basic input mode

1) General information

These include the material name, which is used to load the E(k) file and the scattering parameters file, and the dimensionality of the system, that is specified as a number: 1, 2, or 3. *In the _v0 version, only '3' is available*. Then the user chooses the way the reciprocal space is scanned to build the constant energy surface. `Tetrahedra Scan` adopts the tetrahedron methods, using Delaunay Triangulation, while `'Fast Scan'` performs the "fast scan" described above in the paragraph "**Extraction of the constant energy surfaces**". The `'Fast Scan'` scheme is developed to speed up the whole calculation. It is around 10 to 30× faster, and it is implemented only for the 3D bandstructures.

2) Scattering mechanisms details

The user can choose the constant relaxation time (CRT) approximation and input a constant relaxation time in femtoseconds. In this case, the code will also perform a constant mean-free-path (CMFP) approximation calculation using 5 nm as CMFP value (although the user needs to have in mind that this in general is not consistent with the CRT value). Otherwise, the user can select the scattering mechanisms as explained above via selecting the corresponding button. Then the user can choose whether to save the relaxation times for all k-states or not, as these quantities are memory intensive. If this button is selected, a structure type data containing all the relaxation times (RT) for each k-state and each mechanism will be saved. In this case the code will save three files: RelaxTimeS.separate, RelaxTimeS.Matthiessen, RelaxTimeS.IIS. Details are described in Section 7. (These data can take several tens of Gbytes. However, in order to calculate the TE coefficients in Eq. 1, only the TDF is required. Once the TDF is formed, the scattering rates for each state are computed in Eq. 2, are no longer needed and are deleted unless the user chooses to store them.

3) Transport specifics

Here the user inputs the temperatures (in K) and the Fermi level positions (in eV) for which the simulation will be performed. In semiconductors, the Fermi level zero reference is automatically set at the majority carriers band edge, so negative Fermi level values place the Fermi level in the gap and positive values place it into the band. This is always the case for both band types, because when the user wants to simulate hole transport, the valence band is automatically flipped in positive energies. Thus, positive values of Fermi levels are always into the bands, regardless if the conduction or valence bands are the ones simulated.

Fermi levels and temperatures are inserted as vectors, for instance, in the case of the Fermi levels, the user could write `-0.2,-0.1,-0.05:0.01:0.05,0.1,0.2`. This

returns the following set of Fermi levels: `-0.2, -0.1, -0.05, -0.04, -0.03, -0.02, -0.01, 0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2.`

As each Fermi level corresponds to a specific doping level. It is preferable to input the Fermi level rather than the doping, because this gives to the user a direct control over energy references. Indeed, in new materials it is not possible to know what doping levels correspond to degenerate or non-degenerate conditions.

In practice, the doping level in the material does not change with the temperature (in the extrinsic regime) due to charge neutrality. Thus, by increasing the temperature the Fermi distribution broadens, which forces the Fermi level to shift down away from the bands in order to keep the carrier density constant and equal to the doping density, as depicted in **Figure 7**. Here a parabolic band is sketched with two different impurity densities $N_i^{(1)}$ for non-degenerate doping in (a) and $N_i^{(2)}$ for degenerate doping in (b). The dashed lines represent the Fermi level positions at 300 K and 900 K. The Fermi distributions for these temperatures are depicted in (c) – blue line for 300 K and red line for 900 K. The broadening in the Fermi distribution at higher temperatures results in more states being occupied around the Fermi level E_F. Thus, in order to keep the carrier density constant, the Fermi level shifts down. This shift is more pronounced for the non-degenerate conditions.

Thus, the entered Fermi levels refer to the *lowest entered temperature*. A unique carrier density and doping impurity density are found for each Fermi level. The code assumes the semiconductor to be in the extrinsic region at that temperature, i.e. all the dopants are ionized and all the carriers come from the dopants. At the rest of the temperatures entered, the code shifts those Fermi levels for each temperature such that the impurity density (and charge density) remain constant to their value at the lowest entered temperature.
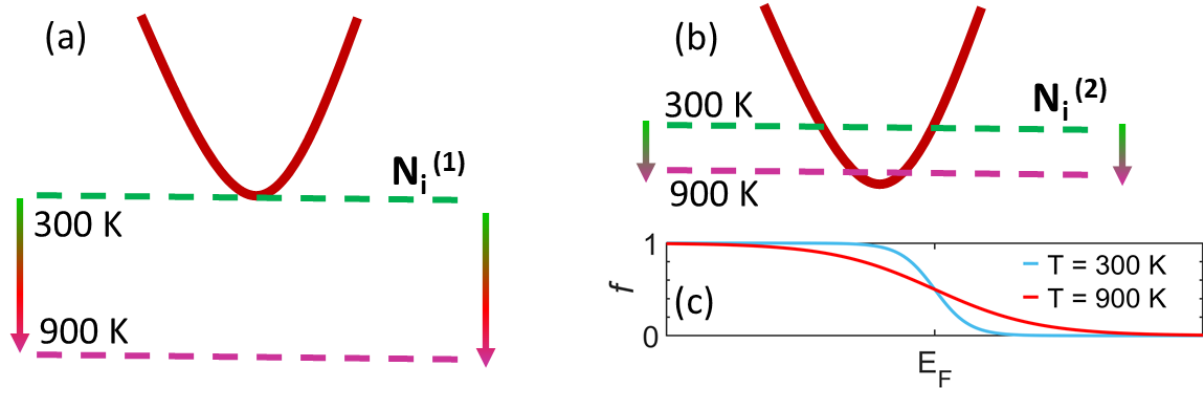
Figure 7: The shift in the Fermi level as the temperature increases from 300 K to 900 K, for: (a) non-degenerate conditions, (b) degenerate conditions. In (c), Fermi distribution is shown for the two temperatures.

However, a user might want to fix the Fermi level positions for all temperatures, being aware that these will correspond to different doping/carrier densities. In this case the user can "lock" the Fermi levels by selecting the 'Lock Fermi levels' button, so that the entered Fermi levels will be used at all temperatures without considering E_F shifts.

NOTE: In the case of scattering with phonons, the Fermi levels do not enter the evaluation of the scattering rates, so a large number of E_F positions can be used for fine resolution of the transport results without computational expense. In the case of the IIS, or of the screened POP scattering mechanisms, a different scattering rate is evaluated for each Fermi level value, so increasing the number of Fermi levels, linearly increases the computation time. Also note that in the case of including IIS, the code also returns the values for the phonon-limited transport. For this, a finer array of 200 Fermi levels starting from the minimum and ending at the maximum entered 'Fermi levels' is used.

Then, the user inputs the 'Energy step' in eV, at which the integration energy step will be performed. As the code performs the integrals following the trapezoidal rule, an energy step of 0.002 or 0.003 eV is generally sufficient. At this version of the code the energy grid is uniform.

Then, the user chooses the majority carrier type: 'electrons' or 'holes'. In the former case the conduction band will be considered, whereas in the latter case the valence band. NOTE: In the case of hole transport in the VB, the bandstructure will be flipped in energy so that the VB will be treated as a CB, with the zero set at the VB edge and the VB energy flipped to positive. A positive Fermi level will correspond to degenerate conditions for both VB and CB. The user only needs to set to zero the Fermi level from DFT output so that the code can automatically detect the CB and VB and set to zero the band edge of interest. This must be done before running the *ELECTRA* code.

If unipolar transport is considered, i.e. only electrons or only holes, the user should NOT select (leave unselected) the 'bipolar transport' button. If it is selected, both the conduction and valence bands will be considered, one as the majority carrier band and one as the minority band.

When the 'bipolar one carrier' is selected, the code will perform the bipolar calculation only for the selected majority carriers. When this button is NOT selected, the code performs a full bipolar calculation, i.e. the code will compute the transport properties for the *n*-type case, electron majority and hole minority, as well as *p*-type case, hole majority and electron minority.

In the unipolar case, the Fermi level is entered with respect to the band edge, which is set to 0 eV. In the case where the bipolar transport is chosen, still the 0 eV reference is set to the band edge of the majority band. When the user performs a full bipolar considering both hole transport and electron transport, in the input instruction only the Fermi levels referred to

the conduction band only must be entered and the code will replicate them for the valence band. The intrinsic Fermi level will be computed and added by default. When the results will be saved, the zero of the Fermi levels will be the intrinsic Fermi level and not the band edge.

In the case the user enters Fermi level values below the intrinsic level, to the *ELECTRA* code will shift the inserted Fermi level values that reside in the bandgap, in the case where any of these values are below the intrinsic level. For example, let us consider a semiconductor with a bandgap of 0.2 eV and an intrinsic Fermi level (the Fermi level for which the number of electrons is identical to the number of holes) at 0.12 eV. If the user enters a value of -0.3 eV for the Fermi level, the inserted Fermi level is shifted to 0.12 eV. In the case of many entered Fermi level values residing below the intrinsic Fermi level, the code will act as follows:

1. the lowest entered Fermi level is set at the intrinsic Fermi level,

2. the energy range between the intrinsic Fermi level and the band edge is divided in number of intervals so that the number of the Fermi levels in the bandgap is the same as entered.

3. the Fermi levels into the band are not changed.

After this, if the `'bipolar one carrier'` is not selected so that the code computes the bipolar transport coefficients for both electron majority and hole majority, *ELECTRA* code will now replicate these values for the valence band.

Finally, the parallelization details are selected. In the _v0 version, only local parallelization is supported, which means that multiple processors on the same node are used. The `'CPU number in the local parallel pool'` is the CPU number on the node on which the code will run. This depends on the HPC cluster facilities and the user will request a corresponding number of CPUs when launching the code on the cluster. The user can specify less CPUs than the total in the node, or the user can run two calculations on the same node. The user must enter the number of CPUs to the code. The user will also need to request a

consistent number of CPUs when launching the code on a supercomputing node. This is done in different ways according to the HPC facility used. In Section 8 we provide an example of the required bash script. If the user specifies a CPU number higher than the one available on the HPC node, then Matlab® will return an alert and will set this number to 12.

ELECTRA code is parallelized in the energy, each processor of the parallel computes all the quantities referred to that energy, each energy is printed in the .log file during the execution, together with a few lines telling the stage of the calculation.

The GUI – Advanced Mode

Below we show a figure of the GUI for the Advanced Mode.

Figure 8: GUI to prepare the input file in the 'Advanced mode'

We explain here only the differences between the 'Basic' and 'Advanced' modes.

1) General part

The `'spin resolved bands'` button is selected if the bands are not spin degenerate.

The `'pi/a'` button is selected if the *k*-mesh is in units of π/a, with *a* being the lattice parameter instead of international system (SI) units, that is 1/m. This can be useful if the user wants to make simulation on artificially built parametric bandstructures rather than DFT computed ones.

The `'multivalley'` option allows the user to provide different deformation potentials for different initial and final bands. See the next section for an example related to this option.

2) Details of scattering mechanisms

There are only two differences in respect to the 'Basic' mode

- The `'screening POP'` can be selected for the POP, which adds a screening term to the POP scattering rate, as in the case of IIS. This makes the computation much slower and the impact is generally very little (see validation part below).

- The `'IVS'`, Inter-Valley Scattering, describes the *inter*-band inelastic scattering. It allows several different processes with different scattering strengths (deformation potentials).

- The `'interband IIS'`, allows the IIS as *inter*-band in addition to *intra*-band.

NOTE: when the `'multivalley'` option is NOT selected, the ODP is both *intra*- and *inter*-band (as commonly considered in the valence band of silicon), while the IVS is *inter*-band only. A certain number of equivalent bands can be considered to mimic multi-valley scattering (as for example the valleys in the CB of Silicon), inserted as multiplicity *Z* in the

'scattering parameters' file. Thus, the carrier scatters into the same valley, like intra-band, considering the multiplicity entered as degeneracy in the 'scattering parameters' file. In the case of Si, this allows to simulate the *f-* and *g-* processes of electron transport in Si without the need to identify if the process should be of *f-* or *g-* type according to the location in the BZ of each pair of initial and final states. Instead, only the degeneracy needs to be inserted, in the case of Silicon, 1 for *f*-processes and 4 for *g*-processes.

For other materials, for which the valleys are different, like the Ge and GaAs conduction bands, the 'multivalley' flag must be selected. When the 'multivalley' option is selected, ODP is *intra-* band only and the IVS can be used to simulate *inter*-valley scattering between different type of bands using different deformation potential values and different phonon energies, as explained in Section 6.

3) Transport specifics

Here the 'metal' option can be selected. This describes the case where the system is metallic rather than semiconducting, see **Figure 9** for a schematic. While in a semiconductor the charge transport can occur mainly in the CB with electrons as the main carriers (*n*-type) or in the VB with holes as the main carriers (*p*-type), in a metal the Fermi level E_F falls in the middle of a band and there is absence of bandgap. In this case the distinction between the conduction and valence band is not made, the bands are not shifted to zero and the inserted Fermi levels are directly used without shift. Thus, when inputting the Fermi levels, the user should know the precise energy range coming from the DFT bandstructure calculation. In this case, the IIS is not considered because it is assumed that the carrier density does not come from dopants, and unipolar transport is adopted.
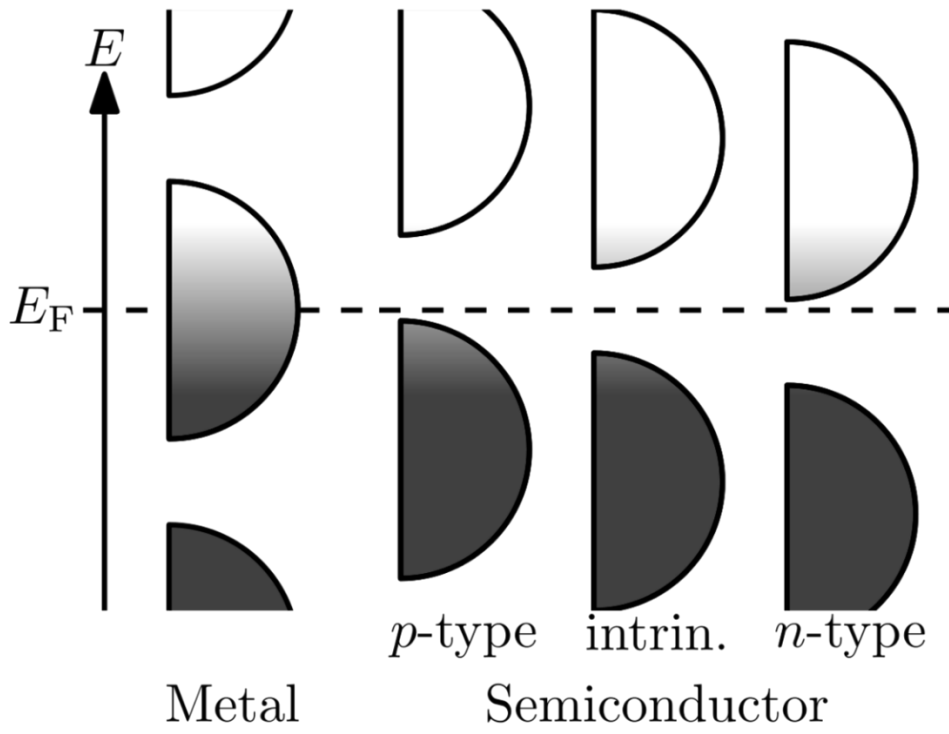
Figure 9: Schematics of the electron energy levels of a metal, left, and of a semiconductor, right, from **[16]**.

5. Input data preparation – format and structure

We detail here the input data that are required by the simulator. These are contained in a file named `Ek_'material_name'`, e.g. `Ek_TiNiSn`, which contains the bandstructure file and the coordinates of the k-points that map the reciprocal unit cell. The code requires the information from *the whole reciprocal unit cell*, or Brillouin Zone, and not the Irreducible wedge of the Brillouin Zone (IBZ). Future versions will implement also the exploitation of the symmetry to reduce the computational cost. Note that the code considers all the inputs as in the International System (SI) units except the energy, in eV.

According to the dimensionality of the bandstructure, the simulator requires as input a bandstructure named `Ek`, which is a matrix of dimensionality D+1, for instance in the case of a 3D material, `Ek` is 4D matrix. `Ek` contains the energy values for each k-point and for each band in the format $n_x \times n_y \times n_z \times n_b$ where $n_{x/y/z}$ are the number of point of the k-grid along the three dimensions, and n_b is the number of bands / eigenvalues for each k-point. Each k-point is identified by numerical indexes (i_x , i_y , i_z), where i_x ranges from 1 to n_x, i_y ranges from 1 to n_y and i_z ranges from 1 to n_z. There is a one-to-one correspondence between the numerical indexes and the coordinates in the reciprocal space, i.e. the k-points which are nearest neighbour in the matrix are also nearest neighbour in the k-space. **Figure 10** sketches this concept. It depicts the points of a $3 \times 3 \times 3$ Monkhorst-Pack grid for a Zincblende material, which lie in the $i_z = 1$ plane ($k_z = 0$ nm^{-1}). The numerical indexes of the represented points are indicated, to show that the numerical indexes reflect the physical positions in the reciprocal space, e.g. the point (i_x , i_y , i_z) and the point (i_x+1 , i_y , i_z), nearest neighbour in the matrix, are also nearest neighbour in the k-coordinates.

The cartesian coordinates of the points must be contained in three $n_x \times n_y \times n_z$ matrixes `kx_matrix`, `ky_matrix` and `kz_matrix`, for the k_x, k_y, k_z coordinates, respectively.
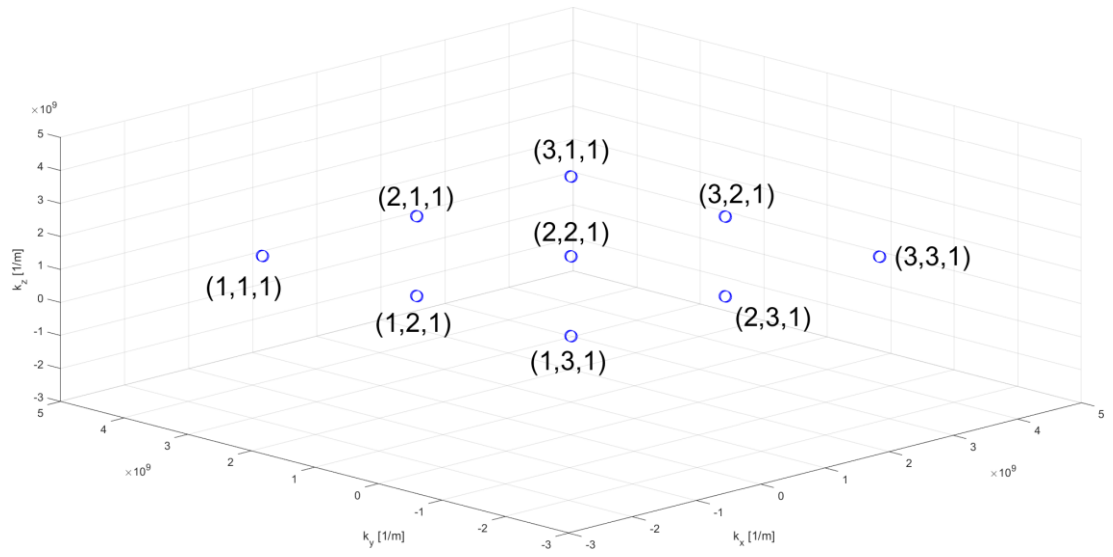
Figure 10: Representation in Cartesian axes of a $3 \times 3 \times 3$ Monkhorst-Pack grid that samples the reciprocal unit cell of a zincblende lattice.

6. Scattering parameters file preparation

As in the case of the input calculation instructions, another GUI is provided to input the scattering parameters. Note that the provided GUI for the scattering parameters is only for the case 'Basic' mode. This is because when the `multivalley` button is selected in the 'Advanced' mode, the scattering parameters to be inserted are diverse and unpredictable since their presence or absence determines also if a transition is allowed or not, as detailed below. This is much simpler to be inserted as a text file. As for the input file, the user can either input a written .m file and a .mat file created by the GUI. The code will automatically check for both and *will prioritize the .m file*.

Two examples of scattering parameters file are given, one for HfNiSn, for calculations without the `multivalley` option, and one for GaAs, where the `multivalley` option is required. In the case the user prepares the .m file, and the file must be named:

`'scattering_parameterS.',material_name,'_',dimensionality,'.m',` e.g. `scattering_parameterS.GaAS.3D.m.`

GUI – 1

We first describe the GUI for the case where the `Alloy scattering` is not enabled. The `Alloy scattring`, for the case of an alloy between material "A" and material "B", requires a large number of parameters because everything entered for material "A" has to be repeated for material "B". For this reason, the GUI has two tabs: when the Alloy scattering is not considered, the user needs only of the first tab, when the user desires to consider the Alloy scattering as well, both the tabs need to be used as explained below.

Figure 11: Tab for the GUI for the scattering parameters. When the 'Alloy scattering' is used, and the alloy is between material "A" and "B", this refers to material "A".

The material name and dimensionality operate as in the input GUI, but then a series of material parameters can be inserted, according to the scattering mechanism under consideration. All the parameter fields need to be filled as a missing parameter will result in 'error'. Equations 5 in paragraph 2 explain how the parameters are used. All the quantities need to be inserted in the indicated units. The dielectric constants are inserted as unitless numbers relative to ε_0. CB and VB refer to the Conduction Band and the Valence Band, respectively. The phonon frequency for ODP and POP in the 'Basic' mode are the same for CB and VB.

The speed of sound can be inserted in two ways:

The bulk modulus K and the shear modulus G can be inserted to compute the speed of sound, [1] *alternatively*, the speed of sound can be directly inserted; in that case K and G will be ignored. The code allows for more ODP or POP mechanisms. If the more than one of such mechanisms is to be included, the deformation potentials and phonon energies will be inserted

one by one and separated by a comma, e.g. `2, 3.5, 4` for ODP deformation potential values and `0.030, 0.032, 0.040` for the phonon energies. The high frequency dielectric constant is required by the code even if it is used only for the POP; if it is not known, the user can enter the same value as the static one. The s.d. array is an array of value which is required when the bands are spin resolved, s.d. = spin degeneracy. The array must contain two different number which identify the two spin orientation, e.g. ' 0, 1, 0, 1, 0, 1, 0, 1 ' can be an options for the case of eight bands. The order must be the same as the bands appear in the `Ek` matrix. The user can use whatever numbers, the code allows the scattering only between bands with the same number in the s.d. array because the scattering does not do spin flip. [**3**] The consideration of spin flip scattering in ferromagnetic material is under study.

GUI – 2

GUI-2 is for the case where Alloy scattering is considered between material "A" and material "B". The material parameters inserted in the first tab (see **Figure 11** above) are for material "A", while in this part of the GUI the user enters some parameters for the alloy and the material "B". This includes the fraction of material "B" and the band gap difference (see Eqs. (4)), and the unit cell volumes of the two materials, in $Å^3$.

NOTES:

- in the case of more than one ODP or POP processes, the number of processes in the materials "A" and "B" must be the same. If, as in the example above, three ODP processes are considered for material "A", also three ODP processes will be entered for material "B" in order to have consistency in the calculation of the alloy parameters.

- The code does not "mix", or "alloy", the band structures; a single band structure, specific for the alloy, needs to be used as input data. This means that in the alloy

between material "A" and material "B", the scattering parameters are scaled according to the composition, but not the bandstructure. In this way the code has been validated for the SiGe alloys with detail in the validation section.

Figure 12: Tab for the GUI for the scattering parameters when the 'Alloy scattering' is used. The alloy is formed between "A" and "B". The GUI contains some alloying information and the parameters for material "B".

The scattering_parameters.m file

Examples of the script for the scattering parameters are provided in the "Data" folder**.** An example is D_adp_e for the deformation potential, where the 'D' stands for the deformation potential, 'adp' for ADP, and 'e' for electrons (CB). The phonon frequency is called, for

instance, `hbar_w_odp_h`, where 'hbar_w' is for $\hbar\omega$, 'odp' for ODP, and 'h' for holes in the VB. According to the calculation under consideration for the carrier type, the code will search and use variables ending with '_e' or '_h'. If the code does not find them, will search for the variable without the specified ending, e.g., in the case of a calculation for electrons, if the code does not find `hbar_w_odp_e`, will search for `hbar_w_odp`, if does not find `D_odp_e`, will search for `D_odp` . If none is found, an error will be returned.

- The "`multivalley`" case

In this case, each valley, or band, is labelled by a letter. The user can choose the letters they wish (e.g. `X` and `W`, or `A` and `B`...) as the code automatically recognizes and searches for the deformation potentials relative to each specific transition between the different valleys. (e.g. `XW_D_adp_e` is the deformation potential `D` for the `adp` process for electrons `e` from the band `X` to the band `W`). If a process is forbidden it is sufficient not to input anything. In this case the code will recognize a process as prohibited. There is no limit in the number of bands that can be labelled, nor to the number of allowed *inter*-band processes. Thus, it is more straightforward that the user enters all required data in a text scattering_parameters .m file rather that in a GUI.

7. Computed quantities

a. <u>Main quantities</u>

The main goal of the code is to compute the transport coefficients: mobility, electrical conductivity, Seebeck coefficients, power factor. These are named, respectively, `mu`, `sigma`, `S`, and `PF`. These quantities are tensors, represented as structure type variables where eah field is the tensor components, e.g. `sigma.xx`, `sigma.yy`, `sigma.zz`, `sigma.xy`, `sigma.yx`, `sigma.xz`, `sigma.zx`, `sigma.yz`, `sigma.zy`, following the i,j convention in Eqs. (1) to (3). The same holds for `S.xx`, etc. and `PF.xx`, etc.. In the same way, the mobility `mu.xx` etc., is computed as the ratio between the electrical conductivity and carrier density. All these quantities are 2-dimensional matrixes where the first index corresponds on the different Fermi levels and the second index on the different temperatures. When the IIS is included, following the same scheme as above, the code also computes the phonon limited quantities, identified with an added '_ph': `sigma_ph.xx`, `sigma_ph.yy`, `sigma_ph.zz`, `sigma_ph.xy`... `S_ph.xx`, `S_ph.yx`... `PF_ph.xx`, `PF_ph.xy`... `mu_ph.xx`, `mu_ph.yy`.... .

In addition, the transport coefficients related to each electron-phonon scattering mechanism separately are computed. These are contained in the '_sep' labelled variables, which is for 'separate', where there is a field for each mechanism and the nine fields for the nine components of the tensors. For instance: `sigma_sep.ADP.xx`, `sigma_sep.ADP.yy`, ... `S_sep.ADP.xx`, `S_sep.ADP.yx` ... `PF_sep.ADP.xx`, `PF_sep.ADP.xy`... `mu_sep.ADP.xx`, `mu_sep.ADP.yy`....; `sigma_sep.ODP.xx`, `sigma_sep.ODP.yy`, ... `S_sep.ODP.xx`, `S_sep.ODP.yx`... `PF_sep.ODP.xx`, `PF_sep.ODP.xy`... `mu_sep.ODP.xx`, `mu_sep.ODP.yz`... ; `sigma_sep.POP.xx`, `sigma_sep.POP.xy`, ... `S_sep.POP.xx`, `S_sep.POP.yx`... `PF_sep.POP.xx`,

38

`PF_sep.POP.xy`… `mu_sep.POP.xx`, `mu_sep.POP.yy`… ; `sigma_sep.Alloy.xx` `sigma_sep.Alloy.xz` etc..

The above quantities depend on temperature, contained in the one-dimensional array called `T_array`, and Fermi levels, contained in the 2-dimensional matrix `EF_matrix`. The first index of the `EF_matrix,` as in the case of the transport coefficients, correspond to the doping level,, and the second to the temperature. This is because the code considers the Fermi level movement with the temperature, thus, for example, in a given row of the matrix there are the Fermi levels which corresponds at the same doping density, at different temperature.. [**1, 2**] For example, if the user enters: `'-0.2,-0.1,-0.05:0.01:0.05,0.1,0.2'`, 15 Fermi levels will be considered, which correspond to 15 doping levels. In order to have the same doping density at each entered temperature, the 15 entered Fermi levels will be shifted at each temperature to get the same doping density. If 6 temperature values would have been entered, the `EF_matrix` turns out to be a 15 x 6 matrix. Because the `EF_matrix` maps the impurity density and the carrier density (and this mapping is different in the unipolar and bipolar cases), other two 2-dimensional matrixes exist: `n_carrier` for the carrier density and `N_imp_matrix` for the impurity density. When the IIS mechanism is selected, the phonon limited quantities are also computed and saved in the following matrixes: `EF_matrix_ph`, `n_carrier_ph`, `N_imp_matrix_ph`.

NOTE: when the calculation is perfoermed in the full bipolar fashion, that is by computing both *p*-type and *n*-type in the same run, the `EF_matrix` is centred at the intrinsic Fermi level, except in the CRT/CMFP approximation, where it is centred in the midgap in analogy with other CRT codes.

b. <u>Secondary quantities</u>

Carrier energy dependent quantities are also computed, for each band separately and then combined. These are the Transport Distribution Function (`TDF`), the relaxation times (`tauE`), the mean-free-path (`MFP`). The former are labelled according to the *ij* convention: `TDF.xx`, `TDF.xy`, etc., following exactly the same notation concept as in the transport coefficientsThese are three dimensional matrixes where the first dimension runs along the carrier energy, the second along the Fermi level, and the third along the temperature. The components for each singular bands are labelled as `TDF_n.xx`, `TDF_n.xz`, etc. . These are four dimensional matrixes, where the dimensions run along: energy, band index, Fermi level, and temperature.

As above, when the IIS is selected, structure variables ending with _ph, are computed for the phonon limited transport: `TDF_ph.xx`, `TDF_ph_n.xx`, etc.. These are normally two-dimensional matrixes (carrier energy, temperature) or three dimensional when there data are for each band, '_n' labelled. But when the charge screening is added in the POP, because the screening depends on the doping level, these are three- and four-dimensions matrixes.

Similarly, they are computed also for each individual considered phonon mechanism, to allow comparison between contributions, with the same nomenclature: `TDF_sep.ADP.xx`, `TDF_sep.ODP.xx`, `TDF_sep_n.ADP.xx`, etc.. As for the phonon limited case, generally these are two dimensions matrixes (carrier energy, temperature) but the POP field has one more dimension, corresponding to the Fermi level/doping density, when the charge screening is added.

The relaxation times (`tauE`) and the mean-free-paths (`MFP`) are computed in a similar way, along x, or y, or z, as in equations (6) and (7). Thus, there will be: `tauE.x`, `tauE.y`, `tauE.z`,   `tauE_ph.x`,   `tauE_sep.ADP.x`,   `tauE_n.x`,   `tauE_ph_n.x`,

`tauE_sep_n.ADP.x` ... etc. , and `MFP.x`, `MFP.y`, `MFP_ph .x`, `MFP_sep.ADP.x`, `MFP_sep_n.x`, `MFP_ph_n.x`, `MFP_sep_n.ADP.x`... etc. .

Other quantities, derived from the construction of the constant energy surfaces, the $E(\mathbf{k})$ → $k(E)$ process, are also available to the user. These are the energy dependent Density of States (DOS) and the band velocity. They are one dimensional arrays of the size of the energy interval, `E_array`, and are named `DOS_tot` and `V_tot` respectively. The variables named `DOS` and `V` contain the DOS and band velocity information for each band separately.

In the case of bipolar transport, all the band dependent quantities are distinguished between "majority" and "minority". The energy interval for these quantities are the `E_majority` or `E_minority` for each band type.

All the quantities are stored in a .mat file named `'TE_properties.'`,`material_name`, `',scan type,'_',carrier,'.m'`, eventually `'bipolar'` is added in the case of bipolar transport, for example: `TE_TiNiSn_kScan_electrons.mat` or `TE_TiNiSn_kScan_electrons_bipolar.mat`. When a bipolar calculation is run considering both *n*-type and *p*-type cases in the same simulation, the `'majority'` label refers to the conduction band and the word `'full'` is added to the name: `TE_TiNiSn_kScan_full_bipolar`.

The file is saved in the working directory, where also the `'run'` file is located.

8. Execution on a supercomputing facility

In order to execute the code on a supercomputing facility, all the code files need to be placed in a directory on a node with Matlab[R] 2019a or higher installed. Also, the input files and data need to be placed in that directory, as well as the scattering parameters. Then, a script to launch the code needs to be launched, with an example given by the bash script below (however, the specific HPC cluster will have its own language and specifics):

```bash
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=24
#SBATCH --mem-per-cpu=2400mb
#SBATCH --mem=90000mb
#SBATCH --time=50:00:00
module load MATLAB/2019a
matlab -nosplash <ELECTRA_run_Linux.m> run_example.log
```

Please, consider that different HPC clusters might have different requirements to set a parallel calculation with Matlab®, for instance some clusters require to ask one cpu more than the number of cpu defined in Matlab®. These are really infrastructure-specific, any user can ask to the authors in the case it is necessary to modify ad-hoc the Parallel Computing Toolbox™ setting.

Note on the memory managing. *ELECTRA* is parallelized in the energy; when the relaxation times are saved, each processor of the parallel returns the scattering rates of all the *k*-states at that energy. For a bandstructure containing ~ 200,000 k-points and a few bands, if the "fast

scan" option is selected, 2 Gb per CPU and a total of ~ 60 to 90 Gb are largely sufficient. In the example above 24 CPUs are requested with 2.4 Gb each, but the total requested memory is 90 Gb, more than the sum of the memory in the CPUs, which is 57.6 Gb. More memory is required when the relaxation times are saved because the master CPU of the parallel calculation will collect a large amount of data. A good practice is to leave at least 20 Gb not occupied in the parallel execution, i.e. the total requested memory in the node should be at least 20 Gb larger than the sum of the memory used in the individual CPUs (32.4 Gb in the example above). If the "Tetrahedron" method is chosen in 3D band-structures, much more memory can be required, even 5 Gb per CPU and > 100 Gb in the node.

Differently, when the relaxation times are not saved, each processor of the parallel returns directly the TDF and the other energy dependent quantities, say that equations (2), (6) and (7) are computed inside the parallel loop onto each processor and the k-state dependent scattering times are not saved. This allows a much easy managing of the memory because the master processor does not collect tens of Gb anymore, everything is generally of the order of 500 Mb. Besides, the impossibility to access to the state dependent scattering rates makes difficult possible debugging.

9. Validation

The code has been validated versus parabolic and non-parabolic bands, where an analytical solution is known, and for some well known semiconductors for which the transport coefficients are well assessed. A validation for the density-of-states for both parabolic band and complex bandstructure materials can be found in reference [1]. **Figure 13** reports the TDF Ξ, equation (2), and the electrical conductivity σ, equation (1a), for an anisotropic parabolic band with $m_x = 1$, $m_y = 0.5$ and $m_z = 0.1$, in units of electron rest mass, and considering ADP scattering with deformation potential of 10 eV. The *xx* tensorial component as well as the average of the three diagonal components is reported. It can be observed that the matching is nearly perfect. Here, the analytical results are obtained following the approach detailed in reference [6].
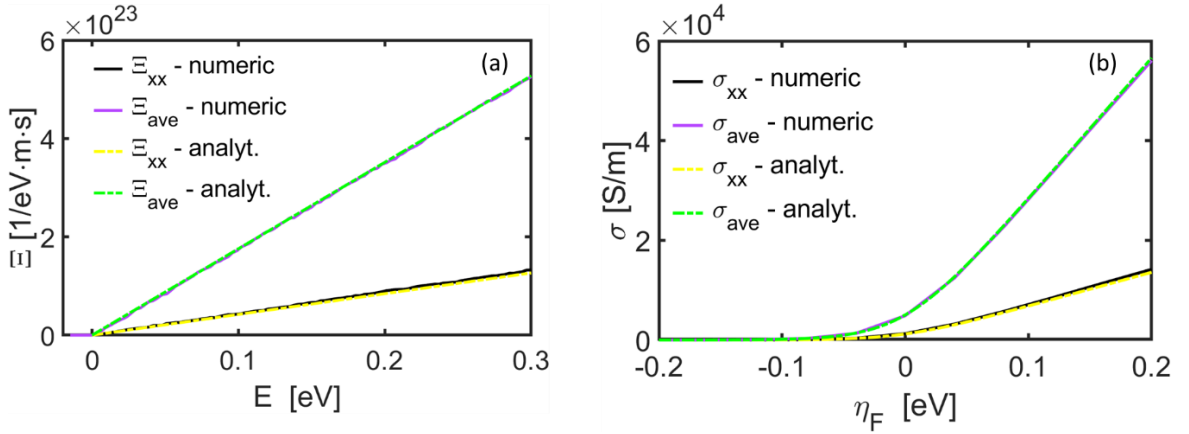

Figure 13: (a) TDF Ξ and (b) electrical conductivity σ, for an anisotropic parabolic band with $m_x = 1$, $m_y = 0.5$ and $m_z = 0.1$, in units of electron rest mass, and considering ADP scattering with deformation potential of 10 eV. The quantities labelled with "*xx*" refer to that tensorial component, the quantities labelled "ave" are the average of the three diagonal tensor components, the quantities identified as "numeric" are computed within the *ELECTRA* scheme while the ones identified as "analyt." are computed using analytical equations known to be exact for parabolic bands.

The capability to describe the charge transport has been validated under several aspects. The CRT functionality of *ELECTRA* has been compared with the results obtained with BoltzTraP, [17] a CRT code widely used in the thermoelectric community, for the complex bandstructure material TiCoSb. The electrical conductivity σ and the power factor PF are reported in **Figure 14a** and **14b**, respectively. Here the blue solid lines are for the BoltzTraP results, the lines with circles are for the *ELECTRA* results, dark yellow for the conductivity and dark green for the PF. The agreement is quite satisfactory with a maximum difference in the PF of below 10 %. This difference is likely due to a different numeric, for instance BoltzTraP performs volume integrals with a smeared gaussian to represent the delta function whereas *ELECTRA* performs surface integrals over the constant energy surfaces defined by the delta function.[1]
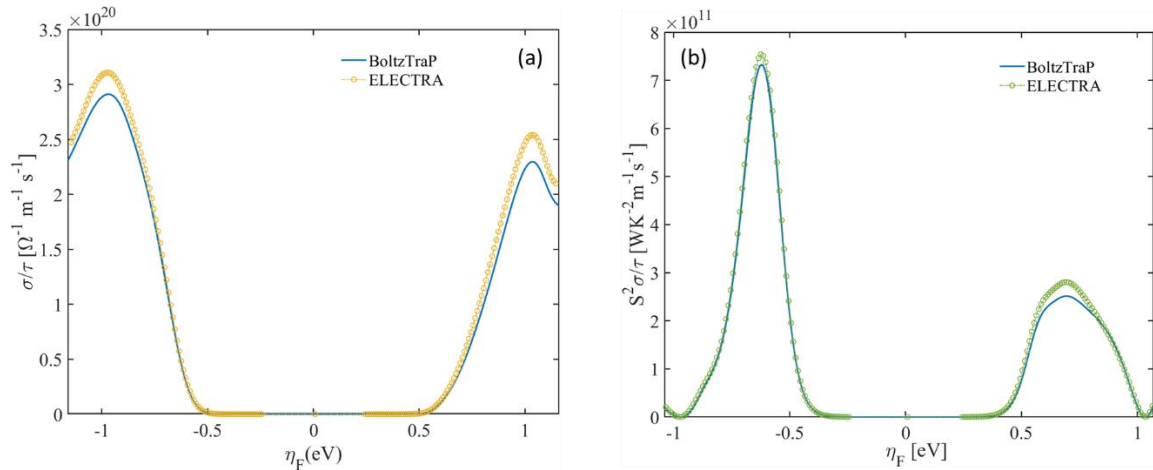

Figure 14: validation of the *ELECTRA* code (yellow and green circles) versus BoltzTraP (blue lines) for TiCoSb; (a) electrical conductivity and (b) PF in units of τ.

In order to validate the treatment of the scattering rate, we apply the *ELECTRA* scheme to the conduction bands of the following material systems: Silicon, GaAs, Ge, SiGe. The bandstructures have been built numerically by following the parameters which can be found in literature, [4, 18, 19] and compared with experimental results from the literature. [18, 20, 21, 22] The scattering treatment is basically the one described in [3, 4] with the addition that in Ge

the IIS has been considered to be inter-valley as well as. For the SiGe alloys, the definition of the scattering deformation potentials has been based on **[9]**. The results are shown in **Figure 15**, the satisfactory agreement corroborate the validity of the implemented scattering scheme. The discrepancies at the higher doping can be ascribed to the electron-electron scattering, **[4]** which is not included here. In addition, **Figure 15c** shows the effect of the spacing in the mesh, in units of π/a, being a the lattice parameter, where a spacing of 0.01 π/a corresponds to a 200×200×200 mesh in the employed bandstructures. A spacing of 0.02 π/a can be sufficient to capture general trends in the transport properties, so it looks enough when screening, or comparing, different materials. However, in order to be more accurate in specific situations, a spacing at least ~ 0.015 π/a is recommended. This is the spacing used in **Figures 15a**, **15b**, and **15d**. For the case of SiGe alloyes, **Figure 15d**, a low doping ~ 10^{14} cm^{-3} has been considered.
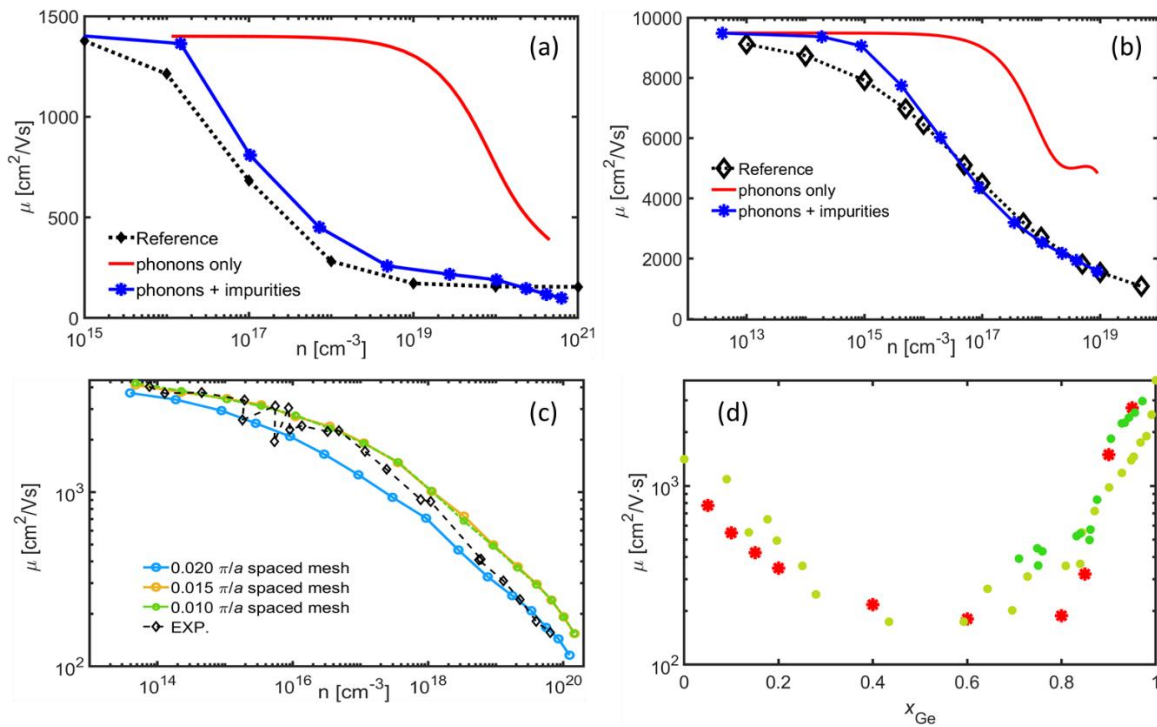

Figure 15: validation of the scattering scheme implemented in *ELECTRA* in respect of known semiconductor; electron mobility for (a) Silicon, (b) GaAs, (c) Ge, (d) SiGe alloys. In (a) and (b) the red lines refer to the phonon-limited mobility, the blue lines to the phonon + IIS

46

mobilities and the black diamonds are the experimental reference values. In (c) the colour circle lines are for the phonon + IIS mobility for k-mesh of different spacing; the black diamonds are the experimental values. In (d) the mobility versus composition for different compositions is reported, the circles are the reference experimental values while the calculated points are reported in red asterisks.

References

1 P. Graziosi, C. Kumarasinghe, N. Neophytou, J. Appl. Phys. 126, 155701 (2019)

2 P. Graziosi, C. Kumarasinghe, N. Neophytou, ACS Appl. Energy Mater. 3, 5913 (2020)

3 B. R. Nag, *Electron Transport in Compound Semiconductors* (Springer-Verlag Berlin Heidelberg, New York, 1980)

4 M. Lundstrom, *Fundamentals of Carrier Transport* (Cambridge University Press, 2000)

5 N. Neophytou, *Theory and Simulation Methods for Electronic and Phononic Transport in Thermoelectric Materials* (Springer) 2020

6 P. Graziosi, C. Kumarasinghe, N. Neophytou, . Phys. Chem. C 124, 18462 (2020)

7 N. Neophytou, H. Karamitaheri, and H. Kosina, J. Comput. Electron. 12(4), 611–622 (2013)

8 G. Lehmann and M: Taut, phys. Stat. Sol. (b) 54, 469 (1972)

9 M. V. Fischetti and S. E. Laux, J. Appl. Phys. 80, 2234 (1996)

10 N. Neophytou and H. Kosina, Phys. Rev. B **83**(24), 245305 (2011)

11 G. Samsonidze, B. Kozinsky, Adv. Energy Mater. 8, 1800246 (2018)

12 M. Combescot, R. Combescot and J. Bok, EPL 2, 31 (1986)

13 S. Wang, Z. Wang, W. Setyawan, N. Mingo, and S. Curtarolo Phys. Rev. X 1, 021012 (2011)

14 P. Yu and M. Cardona, *Fundamentals of Semiconductors* (Springer, 1999)

15 G. Ottaviani, L. Reggiani, C. Canali, F. Nava, and A. Alberigi-Quaranta Phys. Rev. B 12, 3318 (1975)

16 https://en.wikipedia.org/wiki/Semimetal#/media/File:Band_filling_diagram.svg

17 G. K. H. Madsen and D. J. Singh, Comput. Phys. Commun. **175**(1), 67–71 (2006)

18 C.Jacoboni, C.Canali, G.Ottaviani, A.Alberigi Quaranta, solid-State Electronics 20, 77, (1977)

19 W Fawcett and E G S Paige, J. Phys. C: Solid St. Phys., 4, 1801, 1971

20 V. I. Fistul, M. I. Iglitsyn, and E. M. Omelyanovskii, Sov. Phys. Solid State **4,** 784 (1962)

21 Landoldt-Bornstein, Numerical Data and Functional Relationships in Science and Technology , New Series Group III, Vol. 17a, Springer, Berlin, 1982

22 M. Sotoodeh, A. H. Khalid, and A. A. Rezazadeh, J. Appl. Phys. 87, 2890 (2000)