Computational Materials Science 143 (2018) 462–472

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

PAOFLOW: A utility to construct and operate on *ab initio* Hamiltonians from the projections of electronic wavefunctions on atomic orbital bases, including characterization of topological materials

Marco Buongiorno Nardelli [a,e,*], Frank T. Cerasoli [a], Marcio Costa [f], Stefano Curtarolo [b,e,h], Riccardo De Gennaro [g], Marco Fornari [c,d,e], Laalitha Liyanage [a], Andrew R. Supka [c,d], Haihang Wang [a]

[a] Department of Physics and Department of Chemistry, University of North Texas, Denton, TX 76203, USA
[b] Materials Science, Electrical Engineering, Physics and Chemistry, Duke University, Durham, NC 27708, USA
[c] Department of Physics, Central Michigan University, Mount Pleasant, MI 48859, USA
[d] Science of Advanced Materials Program, Central Michigan University, Mount Pleasant, MI 48859, USA
[e] Center for Materials Genomics, Duke University, Durham, NC 27708, USA
[f] Brazilian Nanotechnology National Laboratory (LNNano), CNPEM, 13083-970 Campinas, Brazil
[g] Dipartimento di Fisica, Università di Roma Tor Vergata, Via della Ricerca Scientifica 1, 00133 Roma, Italy
[h] Fritz-Haber-Institut der Max-Planck-Gesellschaft, 14195 Berlin-Dahlem, Germany

ARTICLE INFO

Article history:
Received 18 October 2017
Received in revised form 12 November 2017
Accepted 18 November 2017
Available online 12 December 2017

Keywords:
High-throughput calculations
Computer simulations
Topological materials
Electronic structure

ABSTRACT

PAOFLOW is a utility for the analysis and characterization of materials properties from the output of electronic structure calculations. By exploiting an efficient procedure to project the full plane-wave solution on a reduced space of atomic orbitals, PAOFLOW facilitates the calculation of a plethora of quantities such as diffusive, anomalous and spin Hall conductivities, magnetic and spin circular dichroism, and Z_2 topological invariants and more. The computational cost associated with post-processing first principles calculations is negligible. This code, written entirely in Python under GPL 3.0 or later, opens the way to the high-throughput computational characterization of materials at an unprecedented scale.

1. Introduction

PAOFLOW is a software tool to efficiently post-process standard first principles electronic structure plane-wave pseudopotential calculations in order to promptly compute from interpolated band structures and density of states several quantities that provide insight on transport, optical, magnetic and topological properties such as anomalous and spin Hall conductivity (AHC and SHC, respectively), magnetic circular dichroism, spin circular dichroism, and topological invariants. The methodology is based on the projection on pseudo-atomic orbitals (PAO) [1–3] and is the latest addition to the AFLOW software infrastructure [4,5]. Additional features of PAOFLOW include the calculation of selected integrated quantities using adaptive smearing, the ability to add spin orbit coupling using parametrized methods, and the calculation of surface projected band structures.

PAOFLOW is massively parallel by design (both CPU and GPU) and provides the user with the ability to determine measurable quantities with first principles accuracy and with the speed and robustness required by high-throughput materials characterization. The current implementation (using Quantum ESPRESSO, QE) [6,7] does not require any additional input with respect to a standard electronic structure calculations and, seamlessly, provides a real space tight-binding (TB) representation of the Hamiltonian matrix in a self-contained XML format. The sparse PAO matrix can be easily Fourier transformed and interpolated to determine the full energy dispersion and to compute additional properties associated with derivatives of the energy bands, such as matrix elements of the momentum operator or electron velocities, with the desired level of resolution. PAOFLOW is publicly available under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version. PAOFLOW is integrated in the AFLOWπ high-throughput framework [8] and it is distributed at http://www.aflow.org/src/aflowpi and http://www.aflow.org/src/paoflow.

* Corresponding author at: Department of Physics and Department of Chemistry, University of North Texas, Denton, TX 76203, USA.
E-mail address: mbn@unt.edu (M. Buongiorno Nardelli).

https://doi.org/10.1016/j.commatsci.2017.11.034
0927-0256/

2. Software design

PAOFLOW is written in Python 2.7 (using the Python standard libraries `NumPy` and `SciPy`). The systematic use of regular expression (`re` module) and XML parsing (`xml.etree.cElementTree` module) makes the software expandable to a variety of electronic structure engines with minimal effort. Parallelization on CPUs uses the openMPI protocol through the `mpi4py` module, while GPU parallelization is based on the CUDA kernel through the `pycuda`[1] and `skcuda` [9] modules.

Currently, PAOFLOW requires few basic calculations with the QE package: a first one to generate converged electronic density and Kohn–Sham (KS) potential on an appropriate Monkhorst and Pack (MP) **k**-point mesh (`pw.x`), a second non self consistent calculation (`pw.x`) to evaluate eigenvalues and eigenfunctions for a MP mesh centered at Γ (**k** = (0,0,0), `nosym` and `noinv` = `.true.`) and a third post-processing run using `projwfc.x` to obtain the projection of the eigenfunctions on the pseudo atomic basis functions. No additional calculations with QE are required.

Starting with highly interpolated first principles electronic properties (Fig. 1), PAOFLOW computes band derivatives and Berry's curvature (Fig. 2). These ingredients are then used to determine efficiently the AHC (Fig. 3), magnetic circular dichroism spectra (Fig. 4), SHC (Fig. 5), and spin circular dichroism (Fig. 6).

The PAOFLOW package is distributed with several examples (in the main directory of the distribution, see Section 6) describing the computable physical quantities and can be easily installed on any hardware.

3. Description of the code

– Modules: `build_Pn.py`, `build_Hks.py`
Accurate PAO Hamiltonian matrices can be built from the direct projection of the KS Bloch states $|\psi_{n\mathbf{k}}\rangle$ onto a chosen basis set of fixed localized functions, as it was discussed extensively in Ref. [1–3]. The Hamiltonian for a specific material, $\widehat{H}(\mathbf{r}_\alpha)$, is computed in real space using atomic orbitals or pseudo atomic orbitals from the pseudopotential of any given element. The key in this procedure is in the mapping of the *ab initio* electronic structure (solved on a well converged and large plane waves basis set) into a model that precisely reproduces a selected number of bands of interest. The crucial quantities that measure the accuracy of the basis set are the projectabilities $p_{n\mathbf{k}} = \langle\psi_{n\mathbf{k}}|\widehat{P}|\psi_{n\mathbf{k}}\rangle \geqslant 0$ ($\widehat{P}$ is the operator that projects onto the space of the PAO basis set, as defined in Ref. [2]) which indicate the representability of a Bloch state $|\psi_{n\mathbf{k}}\rangle$ on the chosen PAO set. Maximum projectability, $p_{n\mathbf{k}} = 1$, indicates that the particular Bloch state can be perfectly represented in the chosen PAO set; contrarily, $p_{n\mathbf{k}} \approx 0$ indicates that the PAO set is insufficient and should be augmented. Once the Bloch states with good projectabilities have been identified, the PAO Hamiltonian is constructed either as

$$\widehat{H}(\mathbf{k}) = AEA^\dagger + \kappa\left(I - AA^\dagger\right) \tag{1}$$

following Ref. [1] or

$$\widehat{H}(\mathbf{k}) = AEA^\dagger + \kappa\left(I - A\left(A^\dagger A\right)^{-1}A^\dagger\right) \tag{2}$$

as in Ref. [2], where the case can be chosen in the input of PAOFLOW (see Listing 2 below). Here E is the diagonal matrix of KS eigenenergies and A is the matrix of coefficients obtained from projecting the Bloch wavefunctions onto the PAO set. Since the filtering procedure introduces a null space, the parameter κ is used to shift all the unphysical solutions outside a given energy range of interest. The procedure in Eq. (2) is recommended for most cases. This procedure provides an accurate real space representation of the *ab initio* Hamiltonian $\widehat{H}(\mathbf{R})$ as a TB matrix of small dimension written in XML format, a crucial advantage for the accurate calculation of any physical properties that requires the precise integration in the reciprocal space.

– Module: `add_ext_field.py`
Following the TB formalism (see, for instance, Ref. [15]), the user can add an arbitrary external electric field as a (time-dependent) scalar potential acting on the diagonal elements of

[1] https://mathema.tician.de/software/pycuda/.

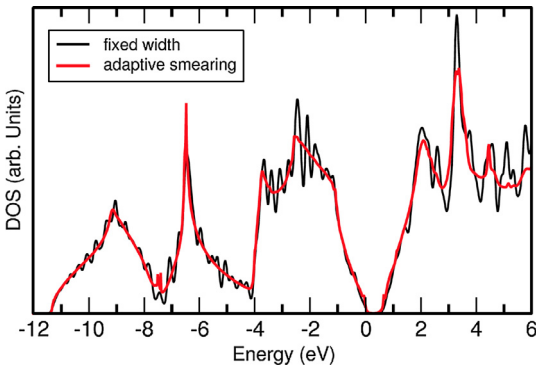

Fig. 1. Comparison between fixed width and adaptive smearing for the density of state of FCC Si. The calculation is done on a 18 × 18 × 18 MP grid and with W = 0.1 eV in the fixed width smearing algorithm. Data adapted from `example 01`.

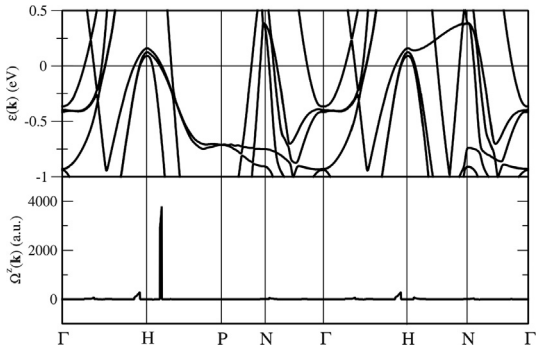

Fig. 2. Top panel: band structure of Fe including spin-orbit interaction. Bottom panel: berry curvature. Data adapted from `example 04` and evaluated along the AFLOW standard path for the BCC lattice [10].

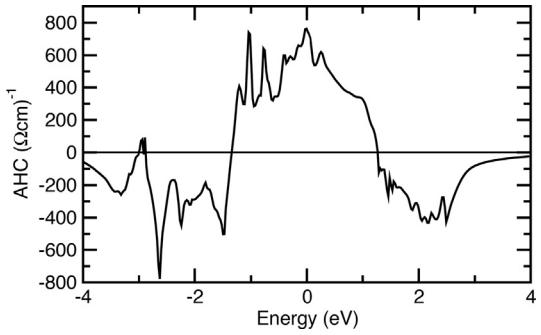

Fig. 3. Energy resolved anomalous Hall conductivity in Fe. Data adapted from `example 04` using a 42 × 42 × 42 MP grid and adaptive broadening yielding a converged value for the AHC at E_{Fermi} of 751 $(\Omega\text{cm})^{-1}$ in excellent agreement with the results from Ref. [11].

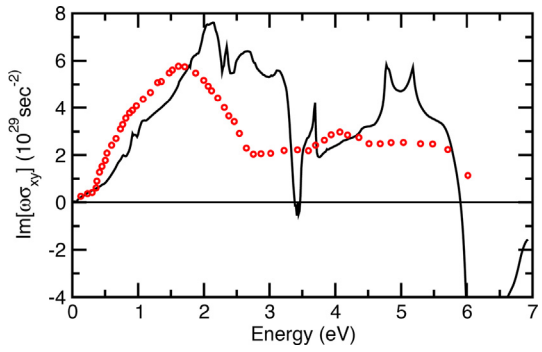

Fig. 4. Magnetic circular dichroism spectrum of Fe. Red circles are experimental data from Ref. [12]. Data adapted from `example 04` using a $42 \times 42 \times 42$ MP grid and adaptive broadening. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

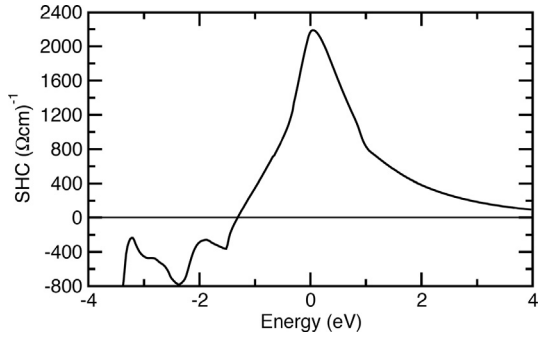

Fig. 5. Energy resolved spin Hall conductivity in Pt. Data adapted from `example 05` using a $42 \times 42 \times 42$ MP grid and adaptive broadening yielding a converged value for the SHC at E_{Fermi} of 2170 $(\Omega\text{cm})^{-1}$ in excellent agreement with previous experimental [13] and theoretical [14] results.

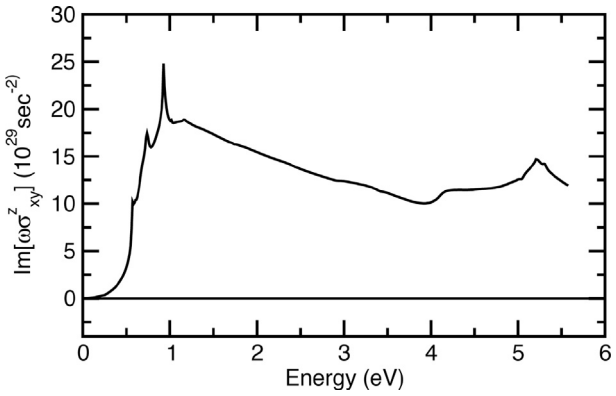

Fig. 6. Spin circular dichroism spectrum of Pt. Data adapted from `example 04` using a $42 \times 42 \times 42$ MP grid and adaptive broadening.

the PAO Hamiltonian: $\epsilon_{\alpha,\mathbf{R}} = \epsilon^0_{\alpha,\mathbf{R}} - e\Phi(\mathbf{R}, T)$. From here is straightforward to introduce an *ad hoc* Hubbard U correction in the Hamiltonian including defining a different U value for any orbital manifold on any element and obtain a fast evaluation of the effect of, for instance, self-consistently computed corrections on the band structure, *i.e.* the Agapito-Curtarolo-Buongiorno-Nardelli approach (ACBN0) [16,17]. This feature is documented in Section 4.3.

– Module: `do_spin_orbit.py`
The spin-orbit coupling (SOC) is essential to characterize the electronic structure and associated properties in topological phases (*e.g.* topological insulators, Weyl semimetals) and materials with magneto-crystalline anisotropy. In addition, the SOC modify the band structure and, in turn, affects transport and optical properties. From a fully relativistic QE calculation PAOFLOW can compute a TB Hamiltonian, $\widehat{H}(\mathbf{R})$, which includes all spin–orbit interactions. QE SOC calculations are, however, computational demanding and require relativistic pseudopotentials. PAOFLOW provides an alternative approach to include the effects of the SOC *via* an effective approximation proposed by Abate and Asdente [18]. The SOC contribution to the Hamiltonians is usually written as:

$$H_{SOC} = \lambda \mathbf{L} \cdot \mathbf{S}, \tag{3}$$

where λ is the SOC strength, which is orbital dependent, **L** and **S** are the orbital and spin angular momentum operators. The λ parameter can be added to non-SOC calculations semi-empirically and adjusted to reproduce the SOC splitting at any given band of the element's stable phase. To ensure accuracy of the results, it is important to choose bands with reasonably high projectabilities.

– Module: `do_bands_calc.py`, `do_topology_calc.py`, `do_fermisurf.py`, `do_spin_texture.py`, `do_dos_calc.py`, `do_pdos_calc.py`
The calculation of topological invariants and related measurable quantities is important to analyze and validate predictions of topological effects in electronic structure theory. PAOFLOW facilitates the automatic computation of standard descriptors of the topology of the band structure such as Fermi surfaces, spin texture, band velocities ($\partial\epsilon/\partial k_i, i = x, y, z$), Berry curvature, spin Berry curvature and the Z_2 invariant for topological insulators. In particular, we follow Ref. [19] and calculate the four Z_2 invariants $\nu_0; (\nu_1\nu_2\nu_3)$ defined by

$$(-1)^{\nu_0} = \prod_{n_j=0,1} \delta_{n_1 n_2 n_3}, \tag{4}$$
$$(-1)^{\nu_{i=1,2,3}} = \prod_{n_{j\neq i}=0,1;n_1=1} \delta_{n_1 n_2 n_3},$$

where

$$\delta_i = \frac{\sqrt{\det[w(\Gamma_i)]}}{\mathrm{pf}[w(\Gamma_i)]} = \pm 1. \tag{5}$$

Here $\mathrm{pf}[w(\Gamma_i)]$ indicates the Pfaffian, $w_{ij}(\mathbf{k}) = \langle\psi_i(-\mathbf{k})|\Theta|\psi_j(\mathbf{k})\rangle$ with Θ the time-reversal operator, and Γ_i are the 8 distinct time reversal invariant momenta expressed in terms of primitive lattice vectors as $\Gamma_{i=(n_1n_2n_3)} = (n_1\mathbf{b}_1 + n_2\mathbf{b}_2 + n_3\mathbf{b}_3)/2$. [19] Of course, all the properties defined in the first Brillouin zone (BZ) including the band structure, are calculated along the standard AFLOW path (see for instance Fig. 2) [10] using Fourier interpolation scheme.

– Module: `PAOFLOW.py`, `smearing.py`, `do_gradient.py`, `do_momenta.py`
Band structure interpolation on arbitrary MP **k**-meshes as well as adaptive smearing for the integration in the BZ are at the very core of the ability of PAOFLOW to provide high-precision electronic structure data. Exploiting the analogy between the PAO and the Wannier functions representation of *ab initio* Hamiltonians, PAOFLOW implements a procedure developed in Ref. [20]. In practice, we estimate the broadening widths using the band derivatives that are readily available from the knowledge of the momentum operator and can be used to estimate the level spacing. Indeed, the TB Hamiltonian can be Fourier transformed from real space representation to the **k**-space and interpolated with arbitrary precision using an efficient procedure based on a zero-padding algorithm that operates

globally and fast Fourier transform (FFT) routines (see Fig. 7 for an illustration of the method).
The same accuracy defined by the projectabilities is conserved in this process. The expectation values of the momentum operator, which is the main quantity in the definition of the adaptive smearing integration scheme, is given by

$$\mathbf{p}_{nm}(\mathbf{k}) = \langle\psi_n(\mathbf{k})|\hat{p}|\psi_m(\mathbf{k})\rangle = \\ = \langle u_n(\mathbf{k})|\frac{m_0}{\hbar}\vec{\nabla}_{\mathbf{k}}\hat{H}(\mathbf{k})|u_m(\mathbf{k})\rangle \quad (6)$$

with

$$\vec{\nabla}_{\mathbf{k}}\hat{H}(\mathbf{k}) = \sum_{\alpha} i\mathbf{R}\exp(i\mathbf{k}\cdot\mathbf{R})\hat{H}(\mathbf{R}). \quad (7)$$

$\hat{H}(\mathbf{R})$ being the real space PAO matrix and $|\psi_n(\mathbf{k})\rangle = \exp(-i\mathbf{k}\cdot\mathbf{r})|u_n(\mathbf{k})\rangle$ the Bloch's functions [21].
Following Ref. [20] we define:

$$W_{n,\mathbf{k}} = a\left|\frac{\partial\epsilon_{n,\mathbf{k}}}{\partial k}\right| \quad (8)$$

for single band integrals and

$$W_{n,m,\mathbf{k}} = a\left|\frac{\partial\epsilon_{n,\mathbf{k}}}{\partial k} - \frac{\partial\epsilon_{m,\mathbf{k}}}{\partial k}\right| \quad (9)$$

for double band integrals. The factor a is of the order of one.[20] See Fig. 1 for a comparison between fixed *vs.* adaptive smearing in the calculation of the density of states of silicon.

- Module: `do_Boltz_tensors.py`

The electronic transport coefficients with the constant relaxation time Boltzmann theory are computed as in Ref. [21]. The electrical conductivity tensor σ_{ij} can be expressed as an integral over the BZ:

$$\sigma_{ij} = \frac{e^2}{4\pi^3}\int_{BZ}\tau\sum_n v_n^i(\mathbf{k})v_n^j(\mathbf{k})\left(-\frac{\partial f_0}{\partial\epsilon}\right)d\mathbf{k}, \quad (10)$$

where τ is the constant relaxation time, $v_n^i(\mathbf{k})$ is the i-th component of the electron velocity ($\mathbf{v}_n$) corresponding to the n-th band for each $\mathbf{k}$-point in the BZ, f_0 is the equilibrium distribution function, and ϵ is the electron energy.
Generalizing Eq. (10) it is also possible to define analogue expressions for the Seebeck coefficient, S, and the electronic contribution to thermal conductivity, κ_{el}. Following the notation of Ref. [22], we introduce the generating tensors $\mathcal{L}_\alpha$ ($\alpha = 0, 1, 2$):

$$\mathcal{L}_\alpha = \frac{1}{4\pi^3}\int\tau\sum_n \mathbf{v}_n(\mathbf{k})\mathbf{v}_n(\mathbf{k})\left(-\frac{\partial f_0}{\partial\epsilon}\right)[\epsilon_n - \mu]^\alpha d\mathbf{k}, \quad (11)$$

where $\mathbf{v}_n(\mathbf{k})\mathbf{v}_n(\mathbf{k})$ indicates the dyadic product and μ is the chemical potential. The coefficients σ, S and κ_{el} can be expressed as follows:

$$\sigma = e^2\mathcal{L}_0 \quad (12)$$
$$S = -\frac{1}{Te}[\mathcal{L}_0]^{-1}\cdot\mathcal{L}_1$$
$$\kappa_{el} = \frac{1}{T}\left(\mathcal{L}_2 - \mathcal{L}_1\cdot[\mathcal{L}_0]^{-1}\cdot\mathcal{L}_1\right),$$

where T is the temperature.

- Module: `do_epsilon.py`

In the limit of long wavelength (*i.e.* negligible momentum transfer) the optical properties of the material depends only on the frequency of the electro-magnetic field. The dielectric tensor can then be expressed in terms of the dielectric susceptibility $\chi_{ij}(\omega)$:

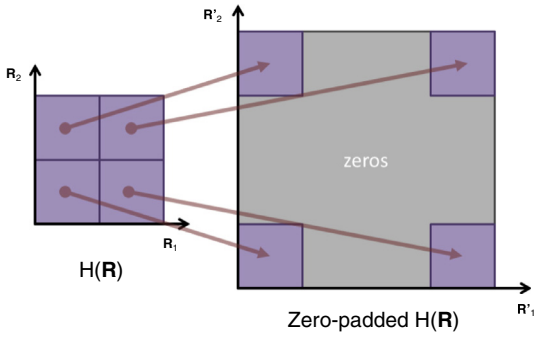

Fig. 7. Schematic illustration of the zero-padding algorithm for fast FFT interpolation. Picture adapted from http://tonyfast.com/nsf-goali/.

$$\epsilon_{ij}(\omega) = 1 + 4\pi\chi_{ij}(\omega). \quad (13)$$

The imaginary part of $\chi(\omega)$ in the single particle approximation can be written as:[15]

$$\mathrm{Im}\chi_{ij}(\omega) = \frac{e^2\pi}{\omega^2\hbar m_0^2\Omega}\sum_{n,m,\mathbf{k}}[f_n(\mathbf{k}) - f_m(\mathbf{k})] \\ \times p_{nm}^i(\mathbf{k})p_{mn}^j(\mathbf{k})\delta(\omega - \omega_{mn}(\mathbf{k})) \quad (14)$$

where m_0 is the bare electron mass, Ω the unit cell volume, m and n the band indices, $f_\ell(\mathbf{k})$ the Fermi–Dirac distribution evaluated on the band with index ℓ at energy $E_\ell(\mathbf{k})$, $p_{nm}^i(\mathbf{k})$ are the matrix elements of the momentum operator calculated over the states (both occupied and empty) with indices m and n and $\hbar\omega_{mn} = E_m(\mathbf{k}) - E_n(\mathbf{k})$ is the energy of the optical transition. The real part of the dielectric susceptibility can then be expressed using the Kramers-Kronig transformation of the imaginary part

$$Re\chi(\omega) = \frac{2}{\pi}\int_0^\infty z\frac{\mathrm{Im}\chi(z)}{z^2 - \omega^2}dz. \quad (15)$$

Alternatively, for semiconductors and insulators, one can evaluate directly the real part of the susceptibility as in Eq. 23 of Ref. [15]. See Ref. [21] for a comprehensive discussion of the evaluation of the calculation of the dielectric function in the PAO scheme.

- Module: `do_Berry_curvature.py`, `do_spin_Berry_curvature.py`, `do_Berry_conductivity.py`, `do_spin_Hall_conductivity.py`

PAOFLOW automatizes the calculation of anomalous charge and spin transport quantitites. We refer to the excellent review by Gradhand et al.[23] for a in depth discussion of these topic. The basic quantity computed by PAOFLOW is the Berry curvature (see Fig. 2):

$$\Omega_n^z(\mathbf{k}) = -\sum_{m\neq n}\frac{2\mathrm{Im}\langle\psi_{n,\mathbf{k}}|v_x|\psi_{m,\mathbf{k}}\rangle\langle\psi_{m,\mathbf{k}}|v_y|\psi_{n,\mathbf{k}}\rangle}{(\omega_m - \omega_n)^2} \quad (16)$$

which provides, in turn, the anomalous Hall conductivity (AHC) (Fig. 3). Here v_x and v_y are obtained from the diagonal elements of the momentum operator defined in Eq. (6).

$$\sigma_{xy} = -\frac{e^2}{\hbar}\int_{BZ}\frac{d^3k}{(2\pi)^3}\Omega^z(\mathbf{k}) \quad (17)$$

where

$$\Omega^z(\mathbf{k}) = \sum_n f_n\Omega_n^z(\mathbf{k}) \quad (18)$$

is the sum of the Berry curvature over the occupied bands (f_n is the Fermi–Dirac distribution).
In the case of variable fields, the Hall conductivity and magnetic circular dichroism (Fig. 4) can also be computed using

$$\sigma_{xy}(\omega) = \frac{e^2}{\hbar}\int_{BZ}\frac{d^3k}{(2\pi)^3}\sum_{m\neq n}[f_n(\mathbf{k}) - f_m(\mathbf{k})]\cdot \frac{2\mathrm{Im}\langle\psi_{n,\mathbf{k}}|v_x|\psi_{m,\mathbf{k}}\rangle\langle\psi_{m,\mathbf{k}}|v_y|\psi_{n,\mathbf{k}}\rangle}{(\omega_m-\omega_n)^2-(\omega+iW_{n,m})^2} \quad (19)$$

where $W_{n,m}$ is the adaptive smearing parameter (Eq. (9)). Starting with relativistic (SOC) band structure PAOFLOW can also compute the spin Berry curvature:

$$\Omega_n^z(\mathbf{k}) = -\sum_{m\neq n}\frac{2\mathrm{Im}\langle\psi_{n,\mathbf{k}}|j_x^z|\psi_{m,\mathbf{k}}\rangle\langle\psi_{m,\mathbf{k}}|v_y|\psi_{n,\mathbf{k}}\rangle}{(\omega_m-\omega_n)^2} \quad (20)$$

where $j_x^z = \{s_z, \mathbf{v}\}$ is the spin current operator with spin $s_z = \frac{\hbar}{2}(\beta, \Sigma_z : 4\times 4$ Dirac matrices) [14] and the corresponding SHC (Fig. 5, Eqs. (17) and (18)). Similarly to Eq. (19), when fields are frequency dependent, PAOFLOW facilitates the calculation of the spin Hall conductivity (SHC) and magnetic circular dichroism (Fig. 6):

$$\sigma_{xy}^z(\omega) = \frac{e^2}{\hbar}\int_{BZ}\frac{d^3k}{(2\pi)^3}\sum_{m\neq n}[f_n(\mathbf{k}) - f_m(\mathbf{k})]\cdot \frac{2\mathrm{Im}\langle\psi_{n,\mathbf{k}}|j_x^z|\psi_{m,\mathbf{k}}\rangle\langle\psi_{m,\mathbf{k}}|v_y|\psi_{n,\mathbf{k}}\rangle}{(\omega_m-\omega_n)^2-(\omega+iW_{n,m})^2} \quad (21)$$

where $W_{n,m}$ is the adaptive broadening parameter defined in Eq. (9).

4. Installation and input file structure

PAOFLOW does not need any specific setup provided that the required Python 2.7 modules are installed on the system. PAOFLOW is executed from a simple (but expandable) `main.py` module that calls `paoflow(` *'input path'*, *'input file'*`)` where the arguments specify the desired path and the XML input file name to be read at runtime (see the file structure in the `examples` directory). PAOFLOW reads also two files from the execution of Quantum ESPRESSO: `data-file.xml` generated by the main run with `pw.x`, and `atom-proj.xml` generated by the post-processing of `projwfc.x` (see also the discussion in Section 2).

4.1. Input file format

In the following we discuss the individual input parameters in the `inputfile.xml` file. To preserve readability, in this description, the 'type' and 'size' XML tags have been neglected.

4.2. System variables

- `fpath`
  - *Description.* Directory created by `projwfc.x` where the calculation (`data-file.xml`) and atomic projection (`atomic-proj.xml`) data are saved.
  - *Type.* String. Default `'dir.save'`.
  - *Example.* See Listing 2.
- `restart`
  - *Description.* Write data to disk at selected checkpoints to skip section of calculations in restart. Data are written in the uncompressed `.npz` format of `NumPy` to ensure maximum data transferability across architectures.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `verbose`
  - *Description.* Writes run information on standard output.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `non-ortho`
  - *Description.* Read overlaps to construct a non orthogonal PAO Hamiltonian. Necessary to perform ACBN0 calculations (see Section 7). After the calculation of bands and band topology the basis is orthogonalized and the calculation proceeds in the new basis.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `write2file`
  - *Description.* Write necessary data to perform ACBN0 calculations (see Section 7).
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `write_binary`
  - *Description.* Write necessary data in binary format to perform ACBN0 calculations in AFLOWπ. [8]
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `writedata`
  - *Description.* Write 3-dim Berry curvature and spin Berry curvature for plotting (format suitble for the `Mayavi` 3D scientific data visualization and plotting package.[2]
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.

Listing 1: `inputfile.xml` - file format

```
<?xml version="1.0"?>
<root>
    <fpath>./dir.save/</fpath>
    ...
    <out_vals>Hksp</out_vals>
    ...
</root>
```

Listing 2: `inputfile.xml` - system section

```
<fpath>./silicon.save/</fpath>
<restart>F</restart>
<verbose>F</verbose>
<non_ortho>F</non_ortho>
<write2file>F</write2file>
<shift_type>1</shift_type>
<shift>auto</shift>
<pthr>0.95</pthr>
<npool>1</npool>
<do_comparison>F</do_comparison>
<naw>
<a>0 0</a>
</naw>
<sh>
<a>0 1 2 0 1 2</a>
</sh>
<nl>
<a>2 1 1 1 1 1</a>
</nl>
```

[2] http://docs.enthought.com/mayavi/mayavi/.

- `use_cuda`
  - *Description.* Use nVIDIA CUDA libraries to perform fft on GPUs.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `shift_type`
  - *Description.* Selects the filtering and shifting scheme to construct the PAO Hamiltonians.
  - *Type.* Integer.
  - *Example.* See Listing 2.
  - *Request syntax.* if `0` uses Eq. (1), if `1` uses Eq. (2) (default), if `2` no correction.
- `shift`
  - *Description.* Define the shifting parameter (κ in Eqs. (1) and (2)).
  - *Type.* String or Float
  - *Example.* See Listing 2.
  - *Request syntax.* if `'auto'`, κ is chosen as the lowest eigenvalue of the highest band with projectability within the tolerance (default); if a number, amount of shift provided by the user.
  - *Units.* if Float, units are eV.
- `pthr`
  - *Description.* Tolerance on the projectability of bands: only bands with projectability $>$ `pthr` will be considered.
  - *Type.* Float. Default = 0.95
  - *Example.* See Listing 2.
  - *Request syntax.* For best Hamiltonian representability it must be a number between 0.9 and 1 (see Ref. [2] for an extensive discussion).
- `npool`
  - *Description.* Number of partitions of largest arrays to reduce the memory requirements and the overhead of message passing in MPI. It can be used to optimize performance for large systems.
  - *Type.* Integer. Default = 1
  - *Example.* See Listing 2.
  - *Request syntax.* Currently `npool` must be chosen so that the total number of **k**-points in the interpolated MP mesh divided by `npool` is an integer and this integer must be divisible by the number of cores.
- `do_comparison`
  - *Description.* Performs a comparison between the DFT eigenvalues from Quantum-ESPRESSO and the ones obtained from the PAO Hamiltonian. Useful when debugging or illustrative purposes. Writes a pdf file (`comparison.pdf`). Uses the `matplotlib` module.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 2.
- `out_vals`
  - *Description.* Defines the data structures that the PAOFLOW module will return after all calculations have been completed. Data structures are returned as a dictionary with keys named as the variables are named in PAOFLOW.
  - *Type.* Array of Strings. Default = [].
  - *Example.* See Listing 1.
- `naw`
  - *Description.* Dimensions of the atomic basis for each atom in the system. The order must be the same as in the output of projwfc.x.
  - *Type.* `numpy` array of Integers. Default = `np.array([0,0])`. Must be specified if external fields are required (see below in Section 4.3).
  - *Example.* See Listing 2.
- `sh` and `nl`
  - *Description.* Shell order `sh` and degeneracy `nl` of the atomic orbital basis in a spin–orbit calculation.
  - *Type.* list of Integers. Default: `sh=[0,1,2,0,1,2]`, `nl=[2,1,1,1,1,1]`. Must be specified if calculation has spin–orbit.
  - *Example.* See Listing 2.
  - *Request syntax.* `sh[:]` gives the order of shells with angular momentum l; `nl[:]` gives the multiplicity of each l shell (depending on the pseudopotential one might have duplicate shells in the list of orbitals). The default values correspond to a system with two atoms, the first with one s, one p and one d shell, the second with two s, one p and one d shell.

4.3. External fields

- `Efield`
  - *Description.* Magnitude of an external electric field added to the diagonal elements of the PAO Hamiltonian.
  - *Type.* Float. Default = `0.0`.
  - *Example.* See Listing 3.
  - *Units.* eV.
- `HubbardU`
  - *Description.* Add a Hubbard U to selected orbitals.
  - *Type.* `numpy` array of Floats. Default = `0.0`.
  - *Example.* See Listing 3.
  - *Units.* eV.
- `bval`
  - *Description.* Index of the highest occupied band for alignment of the top state to 0.
  - *Type.* Integer. Default = `0`.
  - *Example.* See Listing 3.
- `do_spin_orbit`
  - *Description.* Add the spin–orbit interaction a posteriori in the PAO Hamiltonian.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 3.
  - *Request syntax.* If `True` one must provide theta (θ), phi (ϕ), naw, orb_pseudo, lambda_p (λ_p), lambda_d (λ_d).
- `theta and phi`
  - *Description.* Rotation angle around the z-axis and y-axis in cartesian coordinates, respectively.
  - *Type.* Real
  - *Example.* See Listing 3.
  - *Units.* in degrees.

Listing 3: `inputfile.xml` **- external fields**

```
<Efield>
<a>0 0 0</a>
</Efield>
<bval>0</bval>
<do_spin_orbit>F</do_spin_orbit>
<theta>0.0</theta>
<phi>0.0</phi>
<naw>9</naw>
<orb_pseudo>spd</orb_pseudo>
<lambda_p>0.0</lambda_p>
<lambda_d>0.0</lambda_d>
```

- `orb_pseudo`
  - *Description.* Type of pseudo potential used for each atom in the system.
  - *Type.* `numpy` array of Strings. Default = `orb_pseudo.array(['spd'])`.
  - *Example.* See Listing 3.
- `lambda_p and lambda_d`
  - *Description.* Spin–orbit coupling parameter for *p* and *d* states, respectively. The SOC will be included only in the outermost *p* and (or) *d* orbitals.
  - *Type.* Real
  - *Example.* See Listing 3.
  - *Units.* in eV.

4.4. Band structure and band topology

- `onedim`
  - *Description.* Interpolate a band structure calculated along a particular symmetry line in the BZ.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 4.
  - *Request syntax.* If `True` the non self consistent calculation of `pw.x` must be compatible in the choice of the reciprocal space path.
- `do_bands`
  - *Description.* Interpolate the band structure by Fourier transforming the PAO Hamiltonian in real space.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 4.
  - *Request syntax.* If `True` the non self consistent calculation of `pw.x` must be compatible in the choice of the reciprocal space path.
- `ibrav`
  - *Description.* Bravais lattice index following the convention of Quantum-ESPRESSO. The **k**-points paths follow the AFLOW convention [10].
  - *Type.* Float. Default = `False`.
  - *Example.* See Listing 4.
- `dkres`
  - *Description.* Resolution of the **k**-grid along the high symmetry path.
  - *Type.* Float. Default = `False`.
  - *Example.* See Listing 4.
- `band_topology`
  - *Description.* Calculates the Z2 invariant and topological properties (band velocity, Berry and spin Berry curvature) along the band path in the BZ.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 4.
- `ipol, jpol, spol`
  - *Description.* Polarization indices for the components of the Berry ($\Omega_{ipol,jpol}$) and spin Berry curvature ($\Omega^{spol}_{ipol,jpol}$). Components are calculated one at a time.

Listing 4: `inputfile.xml` **- bands**

```
<onedim>F</onedim>
<do_bands>F</do_bands>
<ibrav>0</ibrav>
<nk>2000</nk>
<band_topology>F</band_topology>
<spol>0</spol>
<ipol>0</ipol>
<jpol>0</jpol>
```

  - *Type.* Integer. Default = `0`.
  - *Example.* See Listing 4.

4.5. Hamiltonian interpolation and related quantities

- `double_grid`
  - *Description.* Interpolation of the original PAO Hamiltonian on finer MP grids. Uses a "zero padding" algorithm.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 5.
  - *Notes.* When `True` the code will automatically compute the gradient of the Hamiltonian and the momentum operators.
- `nfftl, nfft2, nfft3`
  - *Description.* Dimension of the dense FFT grid.
  - *Type.* Integer. Default = `0`.
  - *Example.* See Listing 5.
  - *Request syntax.* `nfftl, nfft2, nfft3` must be even.
- `smearing`
  - *Description.* Choice of smearing protocol.
  - *Type.* String.
  - *Example.* See Listing 5.
  - *Request syntax.* Possible choices are: `'None'` simple gaussian smearing with fixed `delta`; `'gauss'` adaptive gaussian smearing; `'m-p'` adaptive Methfessel-Paxton smearing (5th order Hermite polynomials).
- `delta`
  - *Description.* Width of the simple Gaussian smearing if `smearing = None` (see above).
  - *Type.* Float. Default = 0.1
  - *Example.* See Listing 5.
  - *Units.* eV.
- `do_dos, do_pdos`
  - *Description.* Calculate the density of states/projected density of state in the interval [emin,emax].
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 5.
- `emin, emax`
  - *Description.* Energy interval for the calculation of the density of states/projected density of state
  - *Type.* Float. Default = `-10.0, 2.0`.
  - *Example.* See Listing 5.
- `do_fermisurf`
  - *Description.* Evaluate the Fermi surface for energy values between `fermi_up` and `fermi_dw`.

Listing 5: `inputfile.xml` **- Hamiltonian interpolation**

```
<double_grid>F</double_grid>
<nfft1>0</nfft1>
<nfft2>0</nfft2>
<nfft3>0</nfft3>
<smearing>gauss</smearing>
<do_dos>F</do_dos>
<do_pdos>F</do_pdos>
<emin>-10.0</emin>
<emax>2.0</emax>
<delta>0.01</delta>
<do_fermisurf>F</do_fermisurf>
<do_spintexture>F</do_spintexture>
<fermi_up>1.0</fermi_up>
<fermi_dw>-1.0</fermi_dw>
```

- *Type.* Logical. Default = `False`.
- *Example.* See Listing 5.

- `do_spintexture`
  - *Description.* Evaluate the spin texture isosurface for energy values between `fermi_up` and `fermi_dw`.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 4.
- `fermi_up, fermi_dw`
  - *Description.* Energy interval for the calculation of the Fermi surface or spin texture
  - *Type.* Float. Default = `0.1`, `−0.1`.
  - *Example.* See Listing 5.
  - *Units.* eV.

4.6. Transport and optical properties

- `d_tensor`
  - *Description.* Components of the dielectric tensor to be calculated.
  - *Type.* `numpy` array of Integers. Default = All components are calculated.
  - *Example.* See Listing 6.
- `t_tensor`
  - *Description.* Components of the Boltzmann transport tensors to be calculated.
  - *Type.* `numpy` array of Integers. Default = All components are calculated.
  - *Example.* See Listing 6.
- `a_tensor`
  - *Description.* Components of the anomalous Hall and magnetic circular dichroism tensors to be calculated.

Listing 6: `inputfile.xml` **- Transport and optical tensors**

```
<d_tensor>
    <a>0 0</a>
    <a>0 1</a>
    <a>0 2</a>
    <a>1 0</a>
    <a>1 1</a>
    <a>1 2</a>
    <a>2 0</a>
    <a>2 1</a>
    <a>2 2</a>
</d_tensor>
<temp>0.025852</temp><!--Room temperature-->
<Boltzmann>F</Boltzmann>
<epsilon>F</epsilon>
<metal>F</metal>
<kramerskronig>F</kramerskronig>
<epsmin>0.0</epsmin>
<epsmax>10.0</epsmax>
<ne>500</ne>
<critical_points>F</critical_points>
<Berry>F</Berry>
<eminAH>-1.0</eminAH>
<emaxAH>1.0</emaxAH>
<ac_cond_Berry>F</ac_cond_Berry>
<spin_Hall>F</spin_Hall>
<eminSH>-1.0</eminSH>
<emaxSH>1.0</emaxSH>
<ac_cond_spin>F</ac_cond_spin>
```

  - *Type.* `numpy` array of Integers. Default = All components are calculated.
  - *Example.* See Listing 6.
- `s_tensor`
  - *Description.* Components of the spin Hall and spin circular dichroism tensors to be calculated.
  - *Type.* `numpy` array of Integers. Default = All components are calculated.
  - *Example.* See Listing 6.
- `epsilon`
  - *Description.* Compute the real and imaginary part of the dielectric function.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 6.
- `metal`
  - *Description.* Add intraband contribution to the real and imaginary part of the dielectric function.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 6.
- `kramerskronig`
  - *Description.* Compute the real part of the dielectric function using the Kramers-Kronig relation.
  - *Type.* Logical. Default = `True`.
  - *Example.* See Listing 6.
  - *Request syntax.* If `False` the program will use Eq. (23) from Ref. [15].
- `epsmin, epsmax`
  - *Description.* Energy interval for the calculation of the dielectric function
  - *Type.* Float. Default = `0.0`, `10.0`.
  - *Example.* See Listing 6.
  - *Units.* eV.
- `ne`
  - *Description.* Number of energy points in the `[emin,emax]` interval.
  - *Type.* Integer. Default = `500`.
  - *Example.* See Listing 6.
  - *Notes.* For accurate integration of the Kramers–Kronig formula we recommend 1000 energy points/eV.
- `critical_points`
  - *Description.* Find **k**-points where a band has zero derivative.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 6.
- `Berry`
  - *Description.* Evaluate Berry curvature and anomalous Hall conductivity (AHC).
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 6.
- `eminAH, emaxAH`
  - *Description.* Energy interval for the calculation of the AHC.
  - *Type.* Float. Default = `-1.0`, `1.0`.
  - *Example.* See Listing 6.
  - *Units.* eV.
- `ac_cond_Berry`
  - *Description.* Evaluate magnetic circular dichroism spectrum in the interval `[0.0,shift]`.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 6.
- `spin_Hall`
  - *Description.* Evaluate spin Berry curvature and spin Hall conductivity (SHC).
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 5.
- `eminSH, emaxSH`
  - *Description.* Energy interval for the calculation of the SHC.
  - *Type.* Float. Default = `-1.0`, `1.0`.

- *Example.* See Listing 6.
- *Units.* eV.

- `ac_cond_spin`
  - *Description.* Evaluate spin circular dichroism spectrum in the interval `[0.0,shift]`.
  - *Type.* Logical. Default = `False`.
  - *Example.* See Listing 6.

5. Performance

Performance of PAOFLOW exploits massive parallelization over k-points throughout the code and over bands whenever possible (mainly in the Hamiltonian interpolation and the calculation of the gradients). Parallel performances have been analyzed on a Dell PowerEdge R730 server with two 2.4 GHz Intel Xeon E5-2680 v4 fourteen-core processors using `Example 01` on a $56 \times 56 \times 56$ MP grid. Results for the scaling of performance with the number of cores are presented in Fig. 8.

Similar performance tests scrutinizing FFT operations were executed on two Nvidia Tesla K80 GPUS (4,992 GPU cores/card). Again using `Example 01`, run times of select FFTs on CPUs or GPUs are compared for MP grid sizes ranging from $12 \times 12 \times 12$ to $72 \times 72 \times 72$. These results are summarized in Fig. 9.

PAOFLOW demonstrates excellent scaling properties on many-core systems and possesses massively parallel capabilities.

6. Examples, testing and continuous integration

PAOFLOW includes a full suite of examples in the `examples/` directory of the distribution. Examples can be run for testing and verifying the accuracy of the calculations upon different Python installations and computer architectures. Each example has a

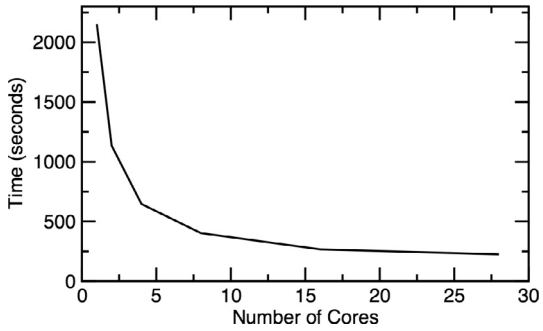

Fig. 8. Parallelized routines provide run time scaling nearly proportional to the number of cores used in a calculation, closely approaching the speed increase limit of Amdahl's Law.

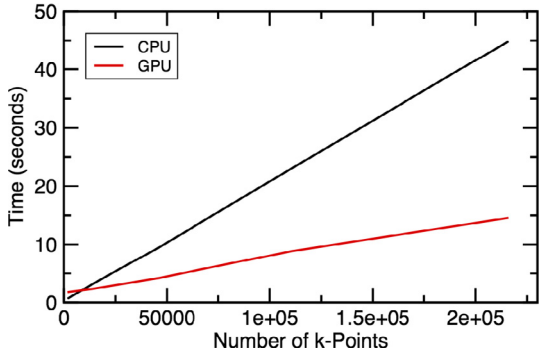

Fig. 9. Executing Fast Fourier Transforms in parallel on GPUs yields a significant increase in run time performance on calculations with a large MP mesh.

`Reference` directory where we have collected reference data for comparison. Examples can be run automatically using the module `run_examples.py`, which needs a small customization to define the directory where QE executables and PAOFLOW are installed. The module will run automatically all the DFT calculations and the PAOFLOW post processing steps. If run with no arguments it will run all examples, if an example name is given as argument, only that example will be run. `run_examples.py` will also automatically verify the accuracy of the results against the data in `Reference` within a given tolerance to avoid false positive due to hardware-specific numerical precision. Examples included in the distribution are:

- `Example 01`
  - *Description.* Silicon with an spd pseudopotential: bands, density of states, projected density of states, Boltzmann transport and dielectric function.
- `Example 02`
  - *Description.* Aluminum with an spd pseudopotential: density of states, Boltzmann transport and dielectric function (`metal = True`).
- `Example 03`
  - *Description.* Platinum in the local spin density approximation (`nspin = 2`): density of states, projected density of states, Boltzmann transport and dielectric function (`metal = True`).
- `Example 04`
  - *Description.* Iron with non-collinear magnetism and spin–orbit interaction: bands, band topology, density of states, anomalous Hall conductivity and magnetic circular dichroism.
- `Example 05`
  - *Description.* Platinum with non-collinear magnetism and spin-orbit interaction: bands, band topology, density of states, spin Hall conductivity and spin circular dichroism.
- `Example 06`
  - *Description.* AlP with *ad hoc* ACBN0 correction: bands, density of states, Boltzmann transport and dielectric function.
- `Example 07`
  - *Description.* Bismuth with the effective spin–orbit interaction approximation: bands, density of states

Further tests can be added simply by incorporating more `exampleXX` directories with the same data structure as the existing ones. No modifications are needed in the `run_examples.py` module. This versatility is essential to ensure continuous integration and early detection of problems in new modules.

7. Externals

Modules and utilities that use data generated by PAOFLOW, or provide input information for a run, have been collected in the `src/external/` directory.

7.1. ACBN0 calculations

The construction of the PAO Hamiltonian allows the direct computation of the direct and self-consistent evaluation of the on-site Coulomb U and exchange J parameters from the ACBN0 functional approach, recently introduced by some of us [16]. Thanks to the accurate PAO representation, the evaluation of the U and J for atoms in different chemical environments or close to topological defects (surfaces, interfaces, impurities, etc.) or for closed-shell atoms (like Zn) becomes trivial, thus overcoming the limitations of traditional linear response techniques with a computational cost comparable to a regular (LDA) PBE calculation. ACBN0 is integrated

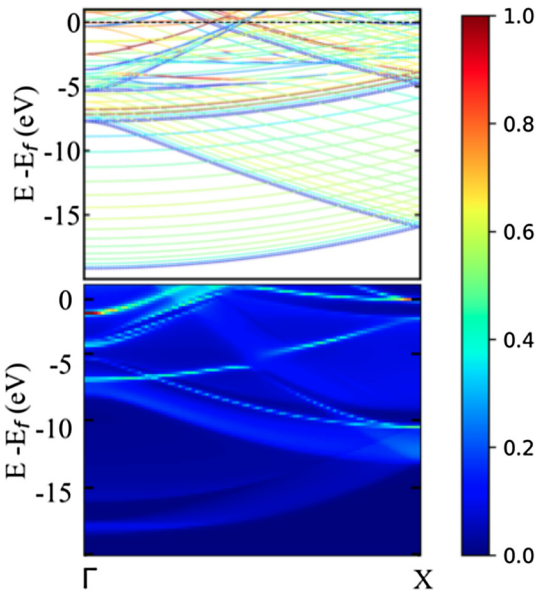

Fig. 10. Si(001) surface projected band structure. Upper panel is the 32 atom slab projected onto the 2 outmost atomic planes. Lower panel is the surface projected bulk band structure using a semi-infinity bulk. Dark red (dark blue) is surface (bulk) bands. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in the AFLOWπ framework and can be run seamlessly as part of a user-defined workflow. However, in order to provide maximum flexibility to users, we have included an external module to PAOFLOW that performs the ACBN0 calculation directly.

ACBN0 uses some modules and the `cints` library from `PyQuante` (http://pyquante.sourceforge.net/). In order to run the program properly, in `clib/` run the shell script `install.sh` (check the $`PATH` to the `include` files in the current python distribution). In order to run the main ACBN0 module `scfuj.py`, PAOFLOW must be run with the variable `write2file = True` (see Listing 1).

7.2. Ballistic transport

Calculations of the ballistic electrical conductance à *la* Landauer are naturally built on a local representation of the electronic structure like the one provided by the PAO Hamiltonians. Our procedure reduces the problem of calculating electron transport [24,25] to a computationally inexpensive post-processing maintaining the predictive power and the accuracy of first principles methods. Briefly, using the Landauer approach the conductance is determined via the transmission function that can be written as: [24,26]

$$\mathcal{T}_{el} = \mathrm{Tr}(\Gamma_L G_C^r \Gamma_R G_C^a),$$

where $G_C^{\{r,a\}}$ are the retarded and advanced Green's functions of the conductor, respectively, and $\Gamma_{\{L,R\}}$ are functions that describe the coupling of the conductor to the leads. The Green's function for the whole system can be explicitly written as: [27]

$$G_C = (\epsilon - H_C - \Sigma_L - \Sigma_R)^{-1} \tag{22}$$

where Σ_L and Σ_R are the self-energy terms due to the semi-infinite leads.

Once the self-energy functions are known, the coupling functions $\Gamma_{\{L,R\}}$ can be easily obtained as [27]

$$\Gamma_{\{L,R\}} = \mathrm{i}[\Sigma^r_{\{L,R\}} - \Sigma^a_{\{L,R\}}].$$

The expression of the self-energies can be deduced along the lines of Ref. [24] using the formalism of principal layers in the framework of the surface Green's function matching theory. We obtain:

$$\begin{aligned} \Sigma_L &= H^\dagger_{LC}(\epsilon - H^L_{00} - (H^L_{01})^\dagger \overline{T}_L)^{-1} H_{LC} \\ \Sigma_R &= H_{CR}(\epsilon - H^R_{00} - H^R_{01} T_R)^{-1} H^\dagger_{CR}, \end{aligned} \tag{23}$$

where $H^{L,R}_{nm}$ are the matrix elements of the Hamiltonian between the layer orbitals of 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 an iterative procedure [24].

A simple generalization of the procedure above allows also for the calculation of surface projected bulk band structures, the essential and definitive tool to determine the existence or not of topologically protected surface states. In Fig. 10 we compare the surface projected bulk band structure for the Si (001) surface with an actual 32 atoms slab calculation.

The calculations are managed by the `transportPAO` utility, an adaptation of the WanT code [25] originally written for a Wannier functions representation of the DFT Hamiltonians.

`transportPAO` is written in Fortran90 and can be easily installed by running a standard `configure` + `make` procedure. Input for `transportPAO` is generated by `restart = True`.

8. Conclusions

With PAOFLOW we provide the electronic structure and materials community with a versatile and agile tool for the *ab initio* characterization of materials properties. The modularity of the code, its speed, and its accuracy, make it an ideal platform for the development of modern materials property databases.

Acknowledgments

We would like to thank Allan H. MacDonald, Arrigo Calzolari, Priya Gopal, Marta dos Santos Guzmao and Cormac Toher for useful discussions. We are grateful to the High Performance Computing Center at the University of North Texas and the Texas Advanced Computing Center at the University of Texas, Austin. The members of the AFLOW Consortium (http://www.aflow.org) acknowledge support by DOD-ONR (N00014-15-1-2266, N00014-13-1-0635, N00014-11-1-0136, and N00014-15-1-2863). The authors also acknowledge Duke University — Center for Materials Genomics. S.C. acknowledges the Alexander von Humbolt Foundation for financial support.

References

[1] L.A. Agapito, A. Ferretti, A. Calzolari, S. Curtarolo, M. Buongiorno Nardelli, Effective and accurate representation of extended Bloch states on finite Hilbert spaces, Phys. Rev. B 88 (2013) 165127.
[2] L.A. Agapito, S. Ismail-Beigi, S. Curtarolo, M. Fornari, M. Buongiorno Nardelli, Accurate tight-binding Hamiltonian matrices from ab initio calculations: Minimal basis sets, Phys. Rev. B 93 (2016) 035104–035109.
[3] L.A. Agapito, M. Fornari, D. Ceresoli, A. Ferretti, S. Curtarolo, M. Buongiorno Nardelli, Accurate tight-binding Hamiltonians for two-dimensional and layered materials, Phys. Rev. B 93 (2016) 125137–125138.
[4] S. Curtarolo, W. Setyawan, G.L.W. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M.J. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, AFLOW: an automatic framework for high-throughput materials discovery, Comput. Mater. Sci. 58 (2012) 218–226.
[5] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R.H. Taylor, L.J. Nelson, G.L. W. Hart, S. Sanvito, M. Buongiorno Nardelli, N. Mingo, O. Levy, AFLOWLIB.ORG: a distributed materials properties repository from high-throughput ab initio calculations, Comput. Mater. Sci. 58 (2012) 227–235.

[6] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. De Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A.P. Seitsonen, A. Smogunov, P. Umari, R.M. Wentzcovitch, Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials, J. Phys.: Condens. Matter 21 (2009) 395502.

[7] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. Dal Corso, S. De Gironcoli, P. Delugas, R.A. DiStasio Jr., A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Kucukbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N.L. Nguyen, H.-V. Nguyen, A. Otero-de-la-Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A.P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, S. Baroni, Advanced capabilities for materials modelling with Quantum ESPRESSO, J. Phys.: Condens. Matter 29 (2017) 465901.

[8] A.R. Supka, T.E. Lyons, L. Liyanage, P. D'Amico, R. Al Rahal Al Orabi, S. Mahatara, P. Gopal, C. Toher, D. Ceresoli, A. Calzolari, S. Curtarolo, M. Buongiorno Nardelli, M. Fornari, AFLOWpi: a minimalist approach to high-throughput ab initio calculations including the generation of tight-binding Hamiltonians, Comput. Mater. Sci. 136 (2017) 76–84.

[9] L.E. Givon, T. Unterthiner, N.B. Erichson, D.W. Chiang, E. Larson, L. Pfister, S. Dieleman, G.R. Lee, S. van der Walt, B. Menn, T.M. Moldovan, F. Bastien, X. Shi, J. Schlüter, B. Thomas, C. Capdevila, A. Rubinsteyn, M.M. Forbes, J. Frelinger, T. Klein, B. Merry, L. Pastewka, S. Taylor, F. Wang, Y. Zhou, scikit-cuda 0.5.1: a Python interface to GPU-powered libraries (2015). https://doi.org/10.5281/zenodo.40565.

[10] W. Setyawan, S. Curtarolo, High-throughput electronic band structure calculations: challenges and tools, Comput. Mater. Sci. 49 (2010) 299–312.

[11] Y. Yao, L. Kleinman, A.H. MacDonald, J. Sinova, T. Jungwirth, D.-s. Wang, E. Wang, Q. Niu, First principles calculation of anomalous Hall conductivity in ferromagnetic bcc Fe, Phys. Rev. Lett. 92 (2004), 037204–4.

[12] G.S. Krinchik, V.A. Artem'ev, Magneto-optical properties of Ni, Co, and Fe in the ultraviolet, visible, and infrared parts of the spectrum, Sov Phys JETP 28, 1080.

[13] T. Kimura, Y. Otani, T. Sato, S. Takahashi, S. Maekawa, Room-temperature reversible spin Hall effect, Phys. Rev. Lett. 98 (2007) 156601.

[14] G.Y. Guo, S. Murakami, T.W. Chen, N. Nagaosa, Intrinsic spin Hall effect in platinum: first-principles calculations, Phys. Rev. Lett. 100 (2008) 096401–096404.

[15] M. Graf, P. Vogl, Electromagnetic fields and dielectric response in empirical tight-binding theory, Phys. Rev. B 51 (1995) 4940.

[16] L.A. Agapito, S. Curtarolo, M. Buongiorno Nardelli, Reformulation of DFT+U as a pseudo-hybrid hubbard density functional for accelerated materials discovery, Phys. Rev. X 5 (2015) 011006.

[17] P. Gopal, M. Fornari, S. Curtarolo, L.A. Agapito, L.S.I. Liyanage, M. Buongiorno Nardelli, Improved predictions of the physical properties of Zn- and Cd-based wide band-gap semiconductors: a validation of the ACBN0 functional, Phys. Rev. B 91 (2015) 245202.

[18] E. Abate, M. Asdente, Tight-binding calculation of 3*d* bands of Fe with and without spin-orbit coupling, Phys. Rev. 140 (1965) A1303–A1308.

[19] L. Fu, C. Kane, E. Mele, Topological insulators in three dimensions, Phys. Rev. Lett. 98 (2007) 106803.

[20] J.R. Yates, X. Wang, D. Vanderbilt, I. Souza, Spectral and Fermi surface properties from Wannier interpolation, Phys. Rev. B 75 (2007), 195121–11.

[21] P. D'Amico, L. Agapito, A. Catellani, A. Ruini, S. Curtarolo, M. Fornari, M. Buongiorno Nardelli, A. Calzolari, Accurate *ab initio* tight-binding Hamiltonians: Effective tools for electronic transport and optical spectroscopy from first principles, Phys. Rev. B 94 (2016) 165166.

[22] N.A. Mecholsky, L. Resca, I.L. Pegg, M. Fornari, Theory of band warping and its effects on thermoelectronic transport properties, Phys. Rev. B 89 (2014) 155131.

[23] M. Gradhand, D.V. Fedorov, F. Pientka, P. Zahn, I. Mertig, B.L. Györffy, First-principle calculations of the Berry curvature of Bloch states for charge and spin transport of electrons, J. Phys.: Condens. Matter 24 (2012) 213202–213224.

[24] M. Buongiorno Nardelli, Electronic transport in extended systems: application to carbon nanotubes, Phys. Rev. B 60 (1999) 7828–7833.

[25] A. Calzolari, N. Marzari, I. Souza, M. Buongiorno Nardelli, Ab initio transport properties of nanostructures from maximally localized Wannier functions, Phys. Rev. B 69 (2004) 035108.

[26] D.S. Fisher, P.A. Lee, Relation between conductivity and transmission matrix, Phys. Rev. B 23 (1981) 6851–6854.

[27] S. Datta, Electronic Transport in Mesoscopic Systems, Cambridge University Press, 1997.