RiflesSO: underlying physics, functionality, and usage of the package

Oleg G. Kharlanov[1, *]

[1]*Faculty of Physics, Lomonosov Moscow State University, 1/2 Leninskie Gory, Moscow 119991, Russia*

(Dated: 01 April 2024)

This manuscript describes a python package RiflesSO providing an object-oriented framework for *ab initio* evaluation of reflection amplitudes/probabilities in crystalline materials (potentially including spin-orbit coupling). After presenting the theoretical perspective, we list the strategies/approximations implemented or to be implemented in RiflesSO in the future, as well as show how to invoke the package to perform the corresponding calculations. Finally, a brief overview of the package design from the developer's standpoint is given.

I. THE PHYSICAL SIDE: WHAT IS REFLECTION?

A. Introduction: from light to quasiparticles

We are quite used to reflection of light, i.e., electromagnetic waves, off mirrors and smooth dielectric surfaces. In geophysics, seismic (i.e., acoustic) waves can undergo reflection at the interfaces, e.g., between the Earth's core and mantle. For a smooth enough interface (and for the wavelengths much shorter than its curvature radius), such reflection is specular: an incident wave with a wave vector/momentum[1] $\mathbf{k}$ gets converted into a finite number of reflected waves with the momenta $\mathbf{k}_r$, $r = 1, \ldots, N_{\text{refl}}$, rather than info a diffuse cloud of different momentum states. However, both electromagnetic and acoustic waves are characterized by another quantum number, the polarization λ; in turn, both the incident and reflected waves should also be described by the polarizations λ and λ_r, respectively. Of course, in a number of situations, reflection is insensitive to polarization, but in general, conversion of different polarizations can occur during reflection, the famous examples being the Brewster's angle of incidence in optics and P-to-S seismic wave conversion in acoustics. Apart from specularity, important properties of reflection are conservation of the wave frequency and considerable coherence of the incident and reflected waves. As for the law of reflection, it does not hold already for a (longitudinal) acoustic P-wave reflected into a (transversal) S-wave in the above example.

In condensed matter, periodicity of crystalline electronic materials typically allows for a classification of the mean-field one-particle electron states (or quasiparticle states) known as the Bloch states:

$$\psi_{\mathbf{k}n}(\mathbf{x}) = u_{\mathbf{k}n}(\mathbf{x})e^{i\mathbf{k}\cdot\mathbf{x}}, \tag{1}$$

$$\mathbf{k} = \sum_i k_i \mathbf{e}'_i, \quad 0 \le k_i < 1, \quad \mathbf{e}'_i \cdot \mathbf{e}_j = 2\pi\delta_{ij}, \tag{2}$$

where $\mathbf{e}_j$ and $\mathbf{e}'_i$ are the direct and reciprocal lattice vectors, respectively, and the k_i components are known as the fractional coordinates of $\mathbf{k}$. A periodic Bloch function $u_{\mathbf{k}n}(\mathbf{x}) = u_{\mathbf{k}n}(\mathbf{x} + \mathbf{e}_i)$ is defined for each momentum $\mathbf{k} \in \text{BZ}$ in the Brillouin zone (defined here as the torus corresponding to $k_i \in [0,\ 1)$) and for a number of Bloch bands n. Note that the spin quantum number is included in n and can be inseparable from the orbital degrees of freedom in systems with spin-orbit coupling; accordingly, the Bloch functions take on values in $\mathbb{C}^2$. Now, loosely speaking, (specular) reflection off a crystalline surface should transform an incident wave with momentum $\mathbf{k}$ and band ("polarization") n into an (at least partially) coherent mixture of reflected Bloch waves:

$$|\psi_{\mathbf{k}n}\rangle \rightsquigarrow \underset{r=1,\ldots,N_{\text{refl}}}{\texttt{mix}} A_r |\psi_{\mathbf{k}_r n_r}\rangle. \tag{3}$$

There are two reasons for a using the above unrigorous notations: first, for a number of reasons, reflected waves undergo rapid decoherence, so a linear superposition (a pure state) would be a limitation; second, the $\rightsquigarrow$ arrow indicates that, even in the pure-state case, the total wave function includes evanescent-state contributions on top of the incident and reflected waves (in fact, the latter property is also typical of optical and acoustic waves). The

---

*Electronic address: kharlanov@physics.msu.ru

[1] For brevity, in what follows, the (crystal) wave vectors $\mathbf{k}$ will also be referred to as the (crystal) momenta, though, technically, the two differ by a factor of $\hbar$.

amplitudes A_r of the individual reflected states are called the reflection coefficients, whereas their moduli squared are related to the reflectances, or reflection probabilities, i.e., define the average part of the total influx of the particles converted into the rth reflection channel. The `RiflesSO` package is aimed at determining these quantities for a crystalline material with a known electronic structure: band energies $\varepsilon_n(\mathbf{k})$, band velocities $\mathbf{v}_n(\mathbf{k}) = \partial\varepsilon_n(\mathbf{k})/\hbar\partial\mathbf{k}$, Bloch functions $u_{\mathbf{k}}(\mathbf{x})$, etc. In other words, the package is aimed at an analogue of the optical Fresnel equations for *ab initio* condensed-matter problems.

There are a number of features, however, that make the reflection problem in condensed matter much less straightforward and easy-to-formulate than its optical and acoustic counterparts:

1. **Nontrivial properties of the boundary.** The wavelengths corresponding to the wavevectors $\mathbf{k} \in \mathrm{BZ}$ are very short, reaching the atomic scale, while even a perfect boundary can have various atomic-scale shapes. Moreover, (periodically continued) addition or deletion of a single atom at the boundary of a crystalline half-space is able to considerably alter the reflectances. As a result, the boundary should either be defined "on average", i.e., randomly, or via some boundary potential.

2. **Boundary potential.** Unlike the case of a glass lens, whose refraction index is constant up to the very glass-air interface, a finite crystalline sample typically has quite a nontrivial potential energy profile within several Angstroms from the boundary, which affects, e.g., the electron work function, and, notably, the reflectances. A precise shape of this potential cannot be retrieved from bulk electronic structure calculations (more resource-demanding slab calculations can be used, though). In some devices, however, there is a confining electrostatic potential that is generated by the leads and confines the electrons to a certain region of the device. The corresponding potential profiles may be much smoother than the intrinsic one due to an abrupt sample edge, thus simplifying the setup and the very calculation of the reflectances.

3. **Crystal momentum $\neq$ momentum.** A Bloch-band representation of the electronic states in crystalline materials is a handy tool, but everything comes at a price: a plane-wave-looking exponential $e^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}$ in Eq. (1) does not contain a true momentum, but a quasi-momentum (or crystal momentum) $\mathbf{k}$, which results in certain nonlocalities in the description. Indeed, unlike the momentum, which is an eigenvalue of a local operator $-\mathrm{i}\nabla$, the $\mathbf{k}$ is defined via a nonlocal shift operator, $\hat{T}_{\mathbf{e}_i}\psi_{\mathbf{k}n} = e^{\mathrm{i}\mathbf{k}\cdot\mathbf{e}_i}\psi_{\mathbf{k}n}$; as a result, even for an isolated band, the Schroedinger equation retains its 2nd-order PDE form only for a quadratic band dispersion. Such a circumstance limits straightforward integration as an approach to finding the electron wave functions in the boundary potential.

4. **Reflected state may not lie on the k grid**. Even for a parallelepiped-shaped sample along the crystallographic directions, it is only in a high-symmetry material that the reflection changes signs of the momentum components, e.g., $k_{1,2} \to k_{1,2},\ k_3 \to -k_3$. For such a material, one may find a symmetric $\mathbf{k}$ grid, such that every incident state on the grid is reflected to a set of the states also belonging to the grid. In general (or for a curved sample boundary), the reflected states do not belong to the chosen finite $\mathbf{k}$ grid, making one project them back onto it following certain physical criteria.

Last but not least, as long as electrons in crystalline materials form a many-body system, they are naturally described by the density matrix, so, eventually, reflection off a boundary should define an admissible quantum operation (a.k.a. quantum dynamical map) taking incident Bloch states to the reflected ones. Let us talk about these in more detail and then discuss the approximations one can come up with to evaluate such quantities.

B. Reflection as an operator/operation on one-particle wave functions/density matrices

Let us assume that a crystalline system with a boundary is described in a local way in terms of a one-particle density matrix/operator $\rho^{(1)}_{\mathbf{k}n,\mathbf{k}'n'}(\mathbf{x};t)$, $\mathbf{x} \in \mathcal{D}$. Such a matrix describes the quantum states of the electrons near point $\mathbf{x}$, serving as a coefficient function for an expectation value of an arbitrary one-particle observable $\hat{A} = \sum\limits_{\mathbf{k}n,\mathbf{k}'n'} A_{\mathbf{k}n,\mathbf{k}'n'}\hat{c}^{\dagger}_{\mathbf{k}n}\hat{c}_{\mathbf{k}'n'}$:

$$\langle A(\mathbf{x})\rangle = \operatorname{tr}\hat{\rho}^{(1)}(\mathbf{x})\hat{A} = \sum_{\mathbf{k}n,\mathbf{k}'n'} \rho^{(1)}_{\mathbf{k}n,\mathbf{k}'n'}(\mathbf{x})A_{\mathbf{k}'n',\mathbf{k}n}, \tag{4}$$

$$\hat{\rho}^{(1)}(\mathbf{x}) = \sum_{\mathbf{k}n,\mathbf{k}'n'} \rho^{(1)}_{\mathbf{k}n,\mathbf{k}'n'}(\mathbf{x})\hat{c}^{\dagger}_{\mathbf{k}n}\hat{c}_{\mathbf{k}'n'}, \tag{5}$$

where $\hat{c}/\hat{c}^\dagger$ are the electron creation/annihilation operators. As a positive semidefinite operator, $\hat{\rho}^{(1)}$ can be traced down to a set of effective pure states $|\phi^{(\nu)}\rangle = \sum_{\mathbf{k}n} \phi^{(\nu)}_{\mathbf{k}n}|\psi_{\mathbf{k}n}\rangle$ occupied by n_ν electrons (in what follows, the real-space coordinate $\mathbf{x}$ is omitted):

$$\hat{\rho}^{(1)} = \sum_\nu n_\nu |\phi^{(\nu)}\rangle\langle\phi^{(\nu)}|. \tag{6}$$

Neglecting quasiparticle interaction during reflection and assuming that $\hat{\rho}^{(1)}$ describes electrons that are incident (i.e., moving toward the external normal $\mathbf{n}$ to the device), we can evaluate the reflection of a mixed state $\hat{\rho}^{(1)}$ by reflecting the pure states $|\phi^{(\nu)}\rangle\langle\phi^{(\nu)}|$ comprising it. Because of that, we will omit the ν index in what follows, working with a single pure incident state.

In the pure state $|\phi\rangle = \sum_{\mathbf{k}n} \phi_{\mathbf{k}n}|\psi_{\mathbf{k}n}\rangle$, one can identify the incident $|\phi^{\text{in}}\rangle$ and the reflected $|\phi^{\text{refl}}\rangle$ parts: namely, $|\phi^{\text{in}}\rangle$ includes those $\phi_{\mathbf{k}n}|\psi_{\mathbf{k}n}\rangle$ contributions, for which $\mathbf{v}_n(\mathbf{k})\cdot\mathbf{n} > 0$, while the reflected components with $\mathbf{v}_n(\mathbf{k})\cdot\mathbf{n} < 0$ form $|\phi^{\text{refl}}\rangle$ (hereinafter, $\mathbf{n}$ denotes an *outer* unit normal vector to the boundary). Clearly, reflection should connect these two parts with each other, thereby placing a boundary condition on the density matrix.

In general, additional decoherence of individual Bloch states $|\psi_{\mathbf{k}n}\rangle$ in $|\phi\rangle$ can occur during reflection: (i) different-energy components produce time-dependent interference terms, which get suppressed for sufficiently long scattering times, (ii) components with different phase velocities decohere because of wave packet separation in space. However, both decoherence mechanisms typically underlie the setup of scattering problems, in which the incident and reflected waves are defined very far from the scattering region, whereas in multiscale simulations, one may deal with the density matrix a reasonable distance, say, 10–20 Å, away from the boundary: at such a distance, evanescent-wave tails may be negligible, while wave packet separation may not have occurred yet. As a result, in such a near-boundary zone, reflected waves can be quite coherent with each other and with the incident one as well. Moreover, it is expected that the *bulk* equations of motion used for the simulations do their part and also lead to decoherence of the reflected waves, regardless of whether decoherence during reflection has been accounted for. Therefore, for simulations, it appears reasonable to apply the reflection algorithms to different Bloch waves $|\phi\rangle$ either without decoherence, or with a partial, phenomenological decoherence of the form:

$$|\phi^{\text{in}}\rangle\langle\phi^{\text{in}}| \rightsquigarrow \mathcal{R}\big[|\phi^{\text{in}}\rangle\langle\phi^{\text{in}}|\big] = \begin{cases} \hat{\mathfrak{r}}|\phi^{\text{in}}\rangle\langle\phi^{\text{in}}|\hat{\mathfrak{r}}^\dagger, & \text{no decoherence,} \\ \sum\limits_{\mathbf{k}n,\mathbf{k}'n'} \phi^{\text{in}}_{\mathbf{k}n}\phi^{\text{in}*}_{\mathbf{k}'n'} e^{-(\varepsilon_n(\mathbf{k})-\varepsilon_{n'}(\mathbf{k}'))^2/2\sigma_E^2}\hat{\mathfrak{r}}|\psi_{\mathbf{k}n}\rangle\langle\psi_{\mathbf{k}'n'}|\hat{\mathfrak{r}}^\dagger, & \text{decoherence in energy,} \\ \sum\limits_{\mathbf{k}n,\mathbf{k}'n'} \phi^{\text{in}}_{\mathbf{k}n}\phi^{\text{in}*}_{\mathbf{k}'n'} e^{-(\mathbf{v}_n(\mathbf{k})-\mathbf{v}_{n'}(\mathbf{k}'))^2/2\sigma_v^2}\hat{\mathfrak{r}}|\psi_{\mathbf{k}n}\rangle\langle\psi_{\mathbf{k}'n'}|\hat{\mathfrak{r}}^\dagger, & \text{decoherence in velocity,} \end{cases} \tag{7}$$

where $\sigma_{E,v}$ are the parameters describing the degree of decoherence. Note that the above dynamical maps are completely positive and trace-preserving (CPTP), which can be proved, e.g., by taking a derivative wrt. $1/\sigma_E^2$ or $1/\sigma_v^2$ and demonstrating that this derivative has a Lindbladian form. In `RiflesSO`, the simplest strategy of the unitary evolution is preferred, though, since it does not lead to decoherence of different spin polarization states expected to have both very close energies and band velocities.

In any case, reflection of the pure state on the l.h.s. boils down to that of a pure Bloch-wave one $|\psi_{\mathbf{k}n}\rangle$ with energy $E = \varepsilon_n(\mathbf{k})$ and band velocity $\mathbf{v}_n(\mathbf{k})$ facing outside of the device $\mathcal{D}$:

$$|\psi_{\mathbf{k}n}\rangle \rightsquigarrow \hat{\mathfrak{r}}|\psi_{\mathbf{k}n}\rangle := \sum_{\mathbf{k}'n'} \mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}|\psi_{\mathbf{k}'n'}\rangle, \tag{8}$$

where coefficients $\mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}$ form the reflection coefficient matrix. One can note the following properties of such elementary reflection processes:

1. **Energy conservation:** $\mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n} = 0$ unless $\varepsilon_{n'}(\mathbf{k}') = \varepsilon_n(\mathbf{k}) = E$. As a result, the $\hat{\mathfrak{r}}$ operator can be (and usually is) represented in a spectral form, $\hat{\mathfrak{r}} = \int \hat{\mathfrak{r}}(E)\mathrm{d}E$, with $\hat{\mathfrak{r}}(E)$ commuting with the bulk one-particle Hamiltonian $\hat{h}_0$ and making reflection in a given energy subspace.

2. **In $\leftrightarrow$ out state mapping:** $\mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n} = 0$ unless $v^{\mathbf{n}}_n(\mathbf{k}) > 0$ and $v^{\mathbf{n}}_{n'}(\mathbf{k}') < 0$ (for brevity, we denote $v^{\mathbf{n}}_{n'} := \mathbf{n}\cdot\mathbf{v}_{n'}$).

3. **Probability (flux) conservation:** $\sum\limits_{\mathbf{k}'n'} |\mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}|v^{\mathbf{n}}_{n'}(\mathbf{k}') = -v^{\mathbf{n}}_n(\mathbf{k})$,

4. **Unitarity:** a stronger version of the above property, $\sum\limits_{\mathbf{k}'n'} \mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}\mathfrak{r}_{\mathbf{k}'n',\mathbf{q}m}v^{\mathbf{n}}_{n'}(\mathbf{k}') = -v^{\mathbf{n}}_n(\mathbf{k})\delta_{\mathbf{k}\mathbf{q}}\delta_{nm}$. By including the velocities into the definition of the reflection matrix, $\tilde{\mathfrak{r}}_{\mathbf{k}'n',\mathbf{k}n} = \mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}\sqrt{\frac{-v^{\mathbf{n}}_{n'}(\mathbf{k}')}{v^{\mathbf{n}}_n(\mathbf{k})}}$, we can simply write $\tilde{\mathfrak{r}}\tilde{\mathfrak{r}}^\dagger = \mathbb{1}$.

5. **Time reversal:** if the Hamiltonian is T-invariant, i.e., commutes with $\hat{\mathcal{T}} = -\mathrm{i}\sigma_2\hat{K}$, $\hat{K}$ denoting the complex conjugation, then the action of $\hat{\mathcal{T}}$ maps incident states to reflected states and vice versa. For example, for an incident state $|\psi_{\mathbf{k}}\rangle$, $\hat{\mathcal{T}}\psi_{\mathbf{k}}(\mathbf{x}) = \sum_{n'} T^{\mathrm{in}}_{nn'}\psi_{-\mathbf{k},n'}(\mathbf{x})$. After some math, one arrives at the property $\tilde{\mathfrak{r}}(E) = -T^{\mathrm{in}}\,\tilde{\mathfrak{r}}^{\mathrm{T}}(E)\,T^{\mathrm{in}*}$; thereby, in the simplest case $T^{\mathrm{in}} = \mathbb{1}$, the reflection matrix is skew-symmetric.

6. **[for boundaries allowing for a nontrivial residual translation symmetry with primitive vectors $\mathbf{e}_i^{(\partial\mathcal{D})}$] Conservation of the tangent quasimomentum:** $\mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n} = 0$ unless $(\mathbf{k}' - \mathbf{k})\cdot\mathbf{e}_i^{(\partial\mathcal{D})} \in 2\pi\mathbb{Z}$. Note that this is not a conservation of the *momentum* along the boundary (typical of optical reflection off a mirror); moreover, for irrational boundary slopes, there is no residual translational symmetry whatsoever. Despite that, the "zeroth-order diffraction maximum approximation" appears quite meaningful even in the context of condensed matter, in which $\mathbf{k}' \equiv \mathbf{k} + \kappa\mathbf{n}$ (modulo *bulk* reciprocal lattice vectors).

An important (though unpleasant) property of *ab initio* reflection is that final momenta $\mathbf{k}'$ do not have to lie on (final) momentum grid chosen to represent the initial (incident) states. Moreover, if we "snap" these to nearest grid points, we will inevitably sacrifice the energy conservation. That is, while time reversal maps "good" grids to themselves ($\mathbf{k} \to -\mathbf{k}$), energy conservation cannot be guaranteed on a finite $\mathbf{k}$ grid for a general system. As for the unitarity, a simple snap-to-grid strategy breaks it, as two *different* final momentum states may be snapped to the same grid point. Therefore, if such properties are desired, a special k-grid projection procedure is necessary, e.g., replacing every final $|\psi_{\mathbf{k}'n'}\rangle$ state by an (in)coherent superposition of k-grid states. An incoherent projection, though explicitly breaks time reversal symmetry, being an irreversible operation.

C. Boundary conditions on the density matrix

Just like a incident/reflected classification can be made in the one-particle Hilbert space $\mathcal{H}^{(1)}$, the density matrix $\hat{\rho}^{(1)} \in \mathcal{B}(\mathcal{H}^{(1)})$ can also be symbolically written as

$$\hat{\rho}^{(1)} = \begin{pmatrix} \hat{\rho}^{(1)\mathrm{in,in}} & \hat{\rho}^{(1)\mathrm{in,out}} \\ \hat{\rho}^{(1)\mathrm{out,in}} & \hat{\rho}^{(1)\mathrm{out,out}} \end{pmatrix}. \tag{9}$$

Now, given Eq. (7) for pure-state reflection, we can immediately conclude that

$$\hat{\rho}^{(1)\mathrm{out,out}} = \mathcal{R}\big[\hat{\rho}^{(1)\mathrm{in,in}}\big]. \tag{10}$$

As for the off-diagonal terms, one can either follow a coherent-reflection path, $\hat{\rho}^{(1)\mathrm{out,in}} = \hat{\rho}^{(1)\mathrm{in,out}\dagger} = \hat{\mathfrak{r}}\hat{\rho}^{(1)\mathrm{in,in}}$, or introduce another set of coherence parameters in the manner of Eq. (7). However, at least the current version of `RiflesSO` assumed total suppression of these off-diagonal terms: physically, this can is related to separation of the incident and reflected wave packets in space.

In realistic simulations of density matrix dynamics, it is often quite hard to store and process all the components of the density *matrix* (say, even for 3 "active" bands and 10^3 k points, the density matrix has about 10^7 complex entries, and operations on such matrices get drastically slow), and one may choose to store only its k-diagonal components $\rho^{(1)}_{\mathbf{k}n,\mathbf{k}m} \equiv \rho^{(1)}_{nm}(\mathbf{k})$. In such a case, since reflection (almost) always produces the states with $\mathbf{k}' \neq \mathbf{k}$, we can also ignore coherence of the incident and reflected states (this time because of the description of the density matrix in the simulation) and, even more, coherence the states with different momenta. This leads to a boundary condition of the form:

$$\rho^{(1)\mathrm{out,out}}(\mathbf{k}') \;=\; \sum_{\mathbf{k}} \mathcal{R}_{\mathbf{k}',\mathbf{k}}\big[\rho^{(1)\mathrm{in,in}}(\mathbf{k})\big], \tag{11}$$

$$\rho^{(1)\mathrm{in,out}}(\mathbf{k}) = \rho^{(1)\mathrm{out,in}}(\mathbf{k}) \;=\; 0, \tag{12}$$

where $\rho^{(1)\mathrm{in,in}}(\mathbf{k})$, $\rho^{(1)\mathrm{out,out}}(\mathbf{k})$ are Hermitian matrices describing in/out states with momentum $\mathbf{k}$, while a superoperator $\mathcal{R}_{\mathbf{k}',\mathbf{k}}$ is similar to the one defined with Eq. (7):

$$\mathcal{R}_{\mathbf{k}',\mathbf{k}}\big[\phi^{\mathrm{in}}(\mathbf{k})\phi^{\mathrm{in}\dagger}(\mathbf{k})\big] = \begin{cases} \mathfrak{r}_{\mathbf{k}',\mathbf{k}}\cdot\phi^{\mathrm{in}}(\mathbf{k})\phi^{\mathrm{in}\dagger}(\mathbf{k})\cdot\mathfrak{r}^{\dagger}_{\mathbf{k}',\mathbf{k}}, & \text{no decoherence,} \\ \sum\limits_{n,n'}\phi^{\mathrm{in}}_n(\mathbf{k})\phi^{\mathrm{in}*}_{n'}(\mathbf{k})e^{-(\varepsilon_n(\mathbf{k})-\varepsilon_{n'}(\mathbf{k}'))^2/2\sigma_E^2}\mathfrak{r}_{\mathbf{k}',\mathbf{k}}\cdot\psi_n(\mathbf{k})\psi^{\dagger}_{n'}(\mathbf{k})\cdot\mathfrak{r}^{\dagger}_{\mathbf{k}',\mathbf{k}}, & \text{decoh. in energy,} \\ \sum\limits_{n,n'}\phi^{\mathrm{in}}_n(\mathbf{k})\phi^{\mathrm{in}*}_{n'}(\mathbf{k})e^{-(\mathrm{v}_n(\mathbf{k})-\mathrm{v}_{n'}(\mathbf{k}))^2/2\sigma_v^2}\mathfrak{r}_{\mathbf{k}',\mathbf{k}}\cdot\psi_n(\mathbf{k})\psi^{\dagger}_{n'}(\mathbf{k})\cdot\mathfrak{r}^{\dagger}_{\mathbf{k}',\mathbf{k}}, & \text{decoh. in velocity.} \end{cases} \tag{13}$$

The notations here are quite as before: $\phi^{\rm in}(\mathbf{k}) = \sum_n \phi_n^{\rm in}(\mathbf{k})\psi_n(\mathbf{k})$ is a pure incident state with momentum $\mathbf{k}$ (a complex vector in a simulation), $\psi_n(\mathbf{k})$ are the eigenvectors of the matrix Hamiltonian $h^{(1)}(\mathbf{k})$ describing the incident Bloch states, and $\varepsilon_n(\mathbf{k})$ are the corresponding eigenvalues. Therefore, once the reflection coefficient matrices $\mathfrak{r}_{\mathbf{k}',\mathbf{k}} = \big(\mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}\big)$ are found, one can evaluate the $\mathcal{R}_{\mathbf{k}',\mathbf{k}}$ dynamical map and thereby completely define the boundary conditions (11), (12). Note these boundary conditions determine just one half of the whole set of independent density matrix components in terms of the other half — physically speaking, relate the reflected channels to the incident ones — which should be enough to solve the evolution equations on the density matrix.

D. Physical aspects of k-grid projection

As we mentioned above, the reflected Bloch waves $|\phi^{\rm refl}\rangle = \sum_{\mathbf{k}'n'} \mathfrak{r}_{\mathbf{k}'n',\mathbf{k}n}|\psi_{\mathbf{k}'n'}\rangle$ resulting from a single incident Bloch wave $|\phi^{\rm in}\rangle = |\psi_{\mathbf{k}n}\rangle$ generally do not lie on the finite momentum grid $\mathcal{K} \subset \mathrm{BZ}$ used in the calculation. That is, while $\mathbf{k} \in \mathcal{K}$, the reflected momenta $\mathbf{k}' \notin \mathcal{K}$ almost surely. While solvers are generally able to yield reflection coefficients for such $\mathbf{k}'$, further incorporation of such data into, e.g., simulations of density matrix dynamics needs another, projected reflection coefficient matrix $\mathfrak{r}^{\Pi}_{\mathbf{k}''n'',\mathbf{k}n}$ with $\mathbf{k}, \mathbf{k}'' \in \mathcal{K}$, which is in some sense close to $\mathfrak{r}$ and retains at least some of the symmetries of the latter one listed in Sec. I B.

There is also another aspect of projection related to spin-orbit coupling and the usage of $\mathbf{k}$-diagonal density matrices $\rho^{(1)}_{mn}(\mathbf{k})$ in the simulations. For example, the zeroth adiabatic approximation (Sec. II A) outputs a single reflected wave $|\psi_{\mathbf{k}'n'}\rangle$ per each incident one $|\psi_{\mathbf{k}n}\rangle$ (up to a certain complex phase). As a result, two incident states with the same $\mathbf{k} = \mathbf{k}_1 = \mathbf{k}_2$ and the bands $n_{1,2}$ corresponding to the two electron spin polarizations acquire very close but different momenta $\mathbf{k}'_{1,2}$ upon reflection. Even if the k-grid were dense enough to incorporate both reflected momenta, within the $\rho^{(1)}_{mn}(\mathbf{k})$ description, this would result in a nonphysical complete decoherence of the two spin projections for arbitrarily small spin-orbit couplings. One of the ways to avoid such behavior is to *intentionally* replace each of the reflected states $\psi_{\mathbf{k}'_{1,2}n'_{1,2}}$ by a "cloud" of states with momenta $\mathbf{k}' \approx \mathbf{k}'_{1,2}$, $\mathbf{k}' \in \mathcal{K}$ and "the same" (i.e., adiabatically transported) Bloch band. As a result, the two clouds overlap, therefore retaining an ability to interfere even within the chosen $\mathbf{k}$-diagonal density matrix description.

The projection issue thus boils down to a choice of the coefficients describing the "cloud":

$$\Pi : |\psi_{\mathbf{k}'n'}\rangle \mapsto \sum_{\mathbf{k}''\in\mathcal{K}} \Pi_{\mathbf{k}'',\mathbf{k}'}(n')|\psi_{\mathbf{k}''n'}\rangle. \tag{14}$$

In the case of the adiabatic solver, reasonable minimal requirements constitute conservation of the probability and the energy fluxes:

$$\sum_{\mathbf{k}''} |\Pi_{\mathbf{k}'',\mathbf{k}'}(n')|^2 v^{\mathbf{n}}_{n'}(\mathbf{k}'') = v^{\mathbf{n}}_{n'}(\mathbf{k}'), \tag{15}$$

$$\sum_{\mathbf{k}''} |\Pi_{\mathbf{k}'',\mathbf{k}'}(n')|^2 v^{\mathbf{n}}_{n'}(\mathbf{k}'')\varepsilon_{n'}(\mathbf{k}'') = v^{\mathbf{n}}_{n'}(\mathbf{k}')\varepsilon_{n'}(\mathbf{k}'), \tag{16}$$

$$\Pi_{\mathbf{k}'',\mathbf{k}'}(n') = 0 \quad \text{unless } v^{\mathbf{n}}_{n'}(\mathbf{k}'), v^{\mathbf{n}}_{n'}(\mathbf{k}'') < 0 \text{ and } \mathrm{dist}(\mathbf{k}'',\mathbf{k}') < R_\Pi, \tag{17}$$

where the distance between the exact and the projected momenta $\mathrm{dist}(\mathbf{k}'',\mathbf{k}') := \min_{\mathbf{G}} |\mathbf{k}'' - \mathbf{k}' - \mathbf{G}|$ is restricted to a rather small projection radius R_Π. The phases of the coefficients are left unfixed by the above equations, however, it is quite natural to choose $\Pi_{\mathbf{k}'',\mathbf{k}'}(n') \geq 0$, which is typical of the wave packet envelope functions. Moreover, even a small "cloud" is expected to contain at least several k-grid points, so that the two above equations are insufficient to fix all the coefficients. Therefore, a functional optimization should be added, e.g., a kind of the maximum-entropy condition:

$$-\sum_{\mathbf{k}''} p_{\mathbf{k}''} \log p_{\mathbf{k}''} \to \max, \quad \text{where} \quad p_{\mathbf{k}''} = \frac{v^{\mathbf{n}}_{n'}(\mathbf{k}'')}{v^{\mathbf{n}}_{n'}(\mathbf{k}')}|\Pi_{\mathbf{k}'',\mathbf{k}'}(n')|^2. \tag{18}$$

Together with the energy and probability conservation *equations*, this entropy condition gives a linear convex optimization problem on the set of nonnegative coefficients $p_{\mathbf{k}''}$ inside the projection radius summing up to unity, which has an explicit solution in terms of linear equations.

Note that the energy is not conserved as a quantum mechanical observable, as should be for the exact scattering process, while is only conserved *on average* upon projection. Also, the above projection algorithm is consistent with the coherent version of the reflection, first line in Eq. (13). That is, even when $\mathfrak{r}^{\Pi}$ is non-unitary, reflection+projection constitute a positive map taking incident states to reflected ones.

In general, one can come up with a more profound (yet a more physical), unitary projection technique, for which the resulting reflection matrix $\mathfrak{r}^{\Pi}_{\mathbf{k}''n'',\mathbf{k}n}$ is a unitary map between the subspaces of the incident and reflected states *for all momenta in* $\mathcal{K}$. Because the dimension of each of these spaces is huge, $|\mathcal{K}|\cdot N_{\text{bands}}/2$, such a unitary projection $\mathfrak{r}\mapsto\mathfrak{r}^{\Pi}$ should be quite slow. On top of the slowness of finding $\mathfrak{r}^{\Pi}$, sparseness of the reflection matrix — "one Bloch wave gets reflected to *several* waves" — may be destroyed upon unitarization, which could additionally suppress the performance. Thus, it is to be decided whether support of such a method will be added to future versions of `RiflesSO`.

II. APPROACHES/APPROXIMATIONS IMPLEMENTED IN `RIFLESSO` AND FEATURES TO BE ADDED

A. Zeroth adiabatic approximation

In this approximation, the total one-particle Hamiltonian near the boundary is

$$\hat{h}=\hat{h}^{(1)}+V(z),\quad \hat{h}^{(1)}=\sum_{\mathbf{k}n}\varepsilon_n(\mathbf{k})|\psi_{\mathbf{k}n}\rangle\langle\psi_{\mathbf{k}n}|, \tag{19}$$

where the boundary potential $V(z)$ is treated as a slowly-changing monotonic function of the outward-facing coordinate $z\equiv\mathbf{n}\cdot\mathbf{x}$. In other words, such a potential contains only small-k_z components and is unable to produce a sizeable change of the particle's momentum. Accordingly, away from band crossing, such an adiabatic evolution approximately conserves a quantity $\varepsilon_n(\mathbf{k})+V(z)=E$, where the momentum $\mathbf{k}$ can be treated quasiclassically as a function of z and interband transitions are suppressed. Now, assuming that the tangent components of the momentum, $\mathbf{k}_{\parallel}\equiv\mathbf{k}-(\mathbf{n}\cdot\mathbf{k})\mathbf{n}$, is conserved, we can trace the evolution of the wave packet along the nth band — just like that of a classical particle — starting from some "entry point" $z_*=0$ with a negligible $V(z_*)\approx 0$, until the same point is reached with an inward-facing velocity, now acting as an "exit point". Mathematically, this amounts to solving the classical Hamilton's equations of the form:

$$\dot{\mathbf{k}}=-\frac{\mathbf{n}}{\hbar}V'(z),\quad \dot{z}=\mathbf{n}\cdot\mathbf{v}_n(\mathbf{k}(z)). \tag{20}$$

The time t, however, is not important in our calculations, and one can, e.g., choose $q:=(\mathbf{k}(z)-\mathbf{k}(z_*))\cdot\mathbf{n}$ as a parameter, so that $\mathbf{k}(z)=\mathbf{k}(z_*)+q\mathbf{n}$. This lets one "scan" the section of the Brillouin zone using steps of the form

$$q\to q-\Delta q,\quad z\to V^{-1}\big(E-\varepsilon_n(\mathbf{k}(z_*)+q\mathbf{n})\big), \tag{21}$$

where $V^{-1}(\varepsilon)$ is an inverse function for the boundary potential, until one hits a point $z\le z_*$. Here, Δq is a sufficiently small positive number, which can be a constant of can be chosen adaptively.

It is also important to understand that during the above momentum scan, one has to adiabatically move along the given nth band, avoiding leaps to the other bands. For well-separated bands, this is a trivial task (e.g., sorting the bands in the ascending-energy order does it), however, for close bands it becomes more complicated. For this goal, a function is implemented in `RiflesSO` that adiabatically transports (propagates) a given band in the momentum space. In view of the fact that quasi-degenerate bands are typically associated with the opposite spin polarization, the band propagation from band n at $\mathbf{k}$ to band $\bar{n}$ at $\bar{\mathbf{k}}\approx\mathbf{k}$ is done in such way that the two Bloch states are maximally collinear as rays in the Hilbert space:

$$\bar{n}=\arg\max_{\bar{\bar{n}}}|\psi^{\dagger}_{\bar{\bar{n}}}(\bar{\mathbf{k}})\cdot\psi_n(\mathbf{k})|, \tag{22}$$

where the row-column product in brackets corresponds to a scalar product of the Bloch functions over the unit cell (u.c.) $\int_{\text{u.c.}}\psi_{\bar{\mathbf{k}}\bar{\bar{n}}}(\mathbf{x})\cdot\psi_{\mathbf{k}n}(\mathbf{x})\mathrm{d}^dx$. Such a band propagation strategy proves to be able to get through band crossing points as well, retaining a "correct" branch of the Bloch function, along which its behavior is continuous (up to the dynamical/Berry phase, of course).

The result of the zeroth-order adiabatic approximation is always a *single* reflected state $|\psi_{\mathbf{k}'n'}\rangle$ (or $|\psi_{\mathbf{k}'n}\rangle$ if the bands are ordered adiabatically). The relative phase between the two can be evaluated as $\int q\mathrm{d}z\approx\sum_{\text{steps }s}q_s(z_{s+1}-z_s)$ over the "scanned" trajectory. Retaining this phase may be important for a correct reflection of the states with an arbitrary spin polarization (i.e., of superpositions of spin-up and spin-down states).

B. First adiabatic, or the ‘on-shell’ approximation

This method is also dealing with the one-particle Hamiltonian of the form (19) with a slowly-varying $V(z)$, but it adds a possibility of interband transitions. To accomplish such a calculation, the path of the particle is split into a number of small segments Δz, and a separate scattering problem is then solved at each of them. In other words, adjacent segments share only the *scattering* information on the locally-defined *travelling* states, while the contributions of *evanescent* states are dropped. This is the reason to refer to the method as the ‘on-shell’ one: only states at the energy surface $\varepsilon_n(\mathbf{k}) + V(z) = E$ are used in the description.

Implementation of the method is planned to be added in later versions of RiflesSO, after the zeroth-order adiabatic approximation is thoroughly tested.

C. Transfer matrix method

This method, based on solution of a recurrence relation for the scattering wave function in the supercells stacked in the z direction, is not implemented in current version of RiflesSO, and whether this feature will be added further depends on the future pathways chosen by the project and his developer(s). Note that, though widely used, the transfer matrix method is not applicable to crystals with an arbitrary transfer integral matrix, as the recurrence relations they generate may become underdetermined or at least poorly-conditioned. When the conditioning is fine, though, the transfer matrix method in principle gives an *exact* result for the reflection coefficients, so adding it could a possible future option as a reference method to test other approximations on a subset of “good” systems.

III. MAIN USAGE MODES AND BASIC EXAMPLES

A. Keyword summary

A list of most important keywords recognized in the RiflesSO input YAML script is presented in Table 1. Amongst the possible values (3rd column), those that are planned but not implemented yet are put in brackets. A more exhaustive keyword list can be seen from an example input script file reflect_graphene.yaml:

```
# reflect_graphene.yaml: Find reflectance of graphene bands
riflesso:
    lattice:                    # Crystal lattice parameters
        lattice-vectors-units:  Angstrom       # The units in which a, b, c are measured
        a.Cartesian:            [4.651, 0, 0]
        b.Cartesian:            [-2.3255, 4.02788, 0]
        c.Cartesian:            [0, 0, 15]              # Just a dummy vector that's large enough
    electronic-structure:       # Electronic structure to be imported
        type:                   Wannierized
        format:                 jdftx
        filename.dft-output:    "../jdftx_calculations/graphene/scf_fine.out"
        filename.wannier-stem:  "../jdftx_calculations/graphene/wannier"
    solver:                     # Reflectance solver parameters
        method:                 adiabatic
        E-tolerance:            5e-4 eV
    boundary-potential:         # The confining potential shape
        shape:                  kink.tanh        # Another option is kink.piecewise_linear
        height:                 10.0 eV          # V(+inf) - V(-inf); should be large enough
        width:                  5.0 Angstrom     # A characteristic _spatial_ scale of V(z)
    projector:                  # Projection of the result onto the finite k-grid
        type:                   default
        k-grid:                 input
    task:                       # Parameters of the batch caclulation
        type:                   reflection-coeffs
        bands:                  all
        k-point-grid:           [8, 8, 1]
        boundary-normals:       # Define a set of boundary normal vectors
```

Parameter	**Belongs to group**	**Possible values**	**Description**
`riflesso`	None	⟨section⟩	The parent section with all `RiflesSO` paramaters
`solver`	`riflesso`	⟨section⟩	Parameters of the quantum-mechanical approximations used for the reflectance calculation
`method`	`solver`	`adiabatic` (`on-shell`)	The approach/approximation used
`E-tolerance`	`solver`	`<N> [eV|erg|Ha]`	Energy tolerance for retrieving the final state obeying energy conservation; `eV` is the default unit
`dq`	`solver`	`<N>`	Momentum step used when scanning the BZ (`method=adiabatic`), in lattice units
`task`	`riflesso`	⟨section⟩	What to do and where to write the results to
`type`	`task`	`reflection-coeffs` (`reflectances`),	The quantities to caclulate
`k-point-grid`	`task`	`[<Nx>, <Ny>, <Nz>]` (`[from-file "filename.ext"]`)	The **k** points comprising the finite grid $\mathcal{K}$
`boundary-normals`	`task`	⟨section⟩	Definition of the boundary normal vectors **n** for which to calculate the reflection coefficients
`type`	`boundary-normals`	`in-plane|from-file|list`	The method of import/generation of the **n** vectors
`output-filename`	`task`	`<filename.ext>`	Where to write the calculated quantities to
`output-format`	`task`	`table|binary`	Format of the output file
`output`	`riflesso`	⟨section⟩	General printing/logging parameters
`logfile`	`output`	`<logfile.ext>|console`	Where to write the log to
...			

FIG. 1: Important `RiflesSO` keywords (to be appended)

```
      type:               in-plane
      plane-normal:       [0, 0, 1]       # The axis around which to rotate \vec{n}
      degrees-range:      [0, 360, 10]    # Azimuthal angle: 0...360deg in 10deg steps
    output-filename:      "reflection_coeffs_graphene.dat"   # Reflection coeffs go here
    output-format:        table                              # Format: text
output:
    logfile:              "output.log"    # Main RiflesSO log file
```

B. How to run `RiflesSO`

To do the tasks described in the above input script, one should invoke the `RiflesSO` package:

```
python -m riflesso -i reflect_graphene.yaml
```

Other options/aliases are also available, which can be listed by calling `python -m riflesso --help`.

A more detailed description of how to invoke individual `RiflesSO` classes and their methods will be included in future releases of this documentation. For now, we limit ourselves to quoting that the above command-line call to `RiflesSO` has a Python equivalent:

```
from riflesso import RiflesSO
import numpy as np

# Initialize the input parameters: lattice vectors, electronic structure, etc.
engine = RiflesSO(from_yaml='reflect_graphene.yaml')
```

```
# Calculate and write reflection coefficients to the output file, as described in my.yaml
engine.run()
```

IV. TERMS OF USAGE/DEVELOPMENT IN A NUTSHELL

A. Licensing

In a nutshell, `RiflesSO` is meant to be a free software, which can be integrated into other community projects by other collaborations (i.e., be used as a *library*). Regardless of whether `RiflesSO` or its fragments are included as is or modified, a correct copyright notice should be included into the code/publications/etc., naming the `RiflesSO` copyright holder, providing a link to the GitHub repository `https://github.com/OKharl/RiflesSO`. At the same time, as `RiflesSO` is being currently developed by a single author, O.K., he retains the right to include some parts of `RiflesSO` into his other projects, potentially including proprietary ones. For example, this relates to the "*full* version" of `RiflesSO`, which may contain additional functions that are neither implemented, nor declared in the publicly available "limited version" of `RiflesSO`.

The latter circumstance excludes the use of GPL licenses, thus, the repository is made public under a BSD license. For details, please follow the link to the LICENSE file: `https://github.com/OKharl/RiflesSO/blob/main/LICENSE`. In particular, the code is presented "as is" without any guarantee of its correct behavior, performance, and correctness of the approximations used and/or implemented. Any posts/suggestions are welcome via the project's GitHub page, but please be informed that at the moment, the project is in its initial phase and the author(s) have their university duties, so that immediate feedback is not guaranteed.

B. Class summary

A graphical class diagram is to be added soon. A `doxygen`-generated class documentation is available in the docs/doxygen folder of the repository. One can preview these HTML files at htmlpreview.github.io, though, unfortunately, this external tool does not display all the information correctly so far.

C. Version history

The current version is 0.1.1 posted on 01 April 2024 (see Table 3), which is being tested, and certain features are under development. Some further features soon to be added are also given in the Table.

D. Dependencies

On top of a bunch of standard Python 3 packages `re`, `argparse`, `RiflesSO` currently uses `numpy` and `scipy` for mathematics and algebraic equation solving. A YAML-formatted configuration file support is provided via `pyyaml` package, however, if a user-formatted configuration file is parsed in a different way into a `dict` object, the latter can be fed to `RiflesSO` as well, and the use of `pyyaml` becomes unnecessary. Currently tested on Python 3.7 under Windows and Linux. Note that the source contains type annotations, which may conflict with earlier Python versions. The required versions of the Python packages are summarized in the requirements.txt file in the repository root.

On top of the above, the documentation for `RiflesSO` was created using `doxygen` tool, version 1.9.8.

Acknowledgments

The project was created by Prof. Oleg G. Kharlanov (present address: Moscow State University, `http://theorphys.phys.msu.ru/en/staff/kharlanov.html`). I would like to thank Prof. Yuan Ping (University of Wisconsin–Madison, https://directory.engr.wisc.edu/mse/Faculty/Ping_Yuan) and Prof. Ravishankar Sundararaman (Rensselaer Polytechnic Institute, `https://mse.rpi.edu/people/faculty/ravishankar-sundararaman`) for valuable advice and inspiring discussions on the way to its current design. If things (hopefully) go right, `RiflesSO` may be further merged with their joint `qimpy` project (`https://github.com/shankar1729/qimpy`) for *ab initio* quantum

Class (module)	Parent(s)	Description/encapsulation
`RiflesSO` (`riflesso_main.py`)	None	The main engine organizing the toolchain and extracting the tasks from the configuration file, handling logging, etc.
`ReflectionTask` (`riflesso_main.py`)	None	A particular reflection task (e.g., "calculate me the reflectance for electronic structure from file `abc.out` off the crystal edge with $\mathbf{n} = (0.5, 0.5, 1)$")
`ElectronicStructure` (`elecstructure.py`)	`ABC`	An abstract class for accessing Bloch states and transporting them across BZ
`WannierElectronic-Structure` (`elecstructure.py`)	`ElectronicStructure`	Wannierized electronic structure and its import
`BlochState` (`elecstructure.py`)	None	A structure describing a Bloch state with a certain energy, velocity, and wave function
`ReflectionSolver` (`reflsolver.py`)	`ABC`	An abstract class for reflectance solvers of ReflectionTasks
`Adiabatic-ReflectionSolver` (`reflsolver_adiabatic.py`)	`ReflectionSolver`	Zeroth-order adiabatic solver
`BoundaryPotential` (`boundarypot.py`)	None	Boundary potential $V(z)$, its inversion and parameters
`KinkPotential` (`boundarypot.py`)	`BoundaryPotential`	Potential with a classical tanh shape
`KgridProjector` (`projector.py`)	None	A default projector class and a base class for projectors
`Kgrid` (`projector.py`)	None	Monkhorst–Pack grid in the k-space
`CrystalLattice` (`lattice.py`)	None	Crystal lattice, direct/reciprocal and absolute/fractional coordinates
`CrystallineHalfSpace` (`lattice.py`)	`CrystalLattice`	A crystalline half-space with a normal vector $\mathbf{n}$
None (`utils.py`)	N/A	A module with various maths utilities (e.g., for retrieving the boundary lattice vectors $\mathbf{e}_i^{(\partial\mathcal{D})}$ from Diophantine equations) functions for advanced RegEx-based parsing, etc.
None (`units.py`)	N/A	List of physical constants in CGS units

FIG. 2: `RiflesSO` class summary

transport simulations of electronic/spintronic devices; whether a limited of a full version of `RiflesSO` will become part of `qimpy` depends on the affiliation issues, etc. In any case, however, certain methods implemented in `RiflesSO` can be useful in their own right beyond `qimpy` or any other external tool.

Version, date	**Description**	**Features added/modified**
0.1.0 (21 March 2024)	The first publicly available pre-release version. Includes class hierarchy for further solvers, description, and basic usage examples. Testing is still underway.	• 0th-order adiabatic reflection solver • import of Wannierized electronic bands from `qimpy` • reflection of density matrices with partial decoherence • export of the calculated reflectances into an external file • YAML configuration file support
0.1.x (planned in April–May 2024)	Update(s) with a virtually complete description and configuration, as well as with various k-grid projectors added.	• full description • k-grid projectors • documentation
0.2 (planned in May 2024)	A thoroughly version with a virtually complete description and configuration, as well as with various k-grid projectors added.	• improved interface/wrappers for being used as a library • a richer palette of k-grid projectors • first implementation of the on-shell reflectance solver

FIG. 3: `RiflesSO` version history