Ψ_k Scientific Highlight Of The Month

No. 161 **October 2026**

ASE2SPRKKR: a unified Python framework integrating the Spin-Polarized Relativistic Korringa-Kohn-Rostoker method into the Atomic Simulation Environment

Ridha Eddhib[1], Matyáš Novák[1,2], Hubert Ebert[3], Aki Pulkkinen[1], and Ján Minár[1,*]

[1] *New Technologies-Research Centre, University of West Bohemia in Pilsen, 30100 Pilsen, Czech Republic*

[2] *FZU – Institute of Physics, Czech Academy of Sciences, v.v.i, Prague, Czech Republic*

[3] *Department Chemie, Ludwig-Maximilians-Universität München, Butenandtstr. 5-11, 81377 München, Germany*

Abstract

The Spin-Polarized Relativistic Korringa-Kohn-Rostoker (SPR-KKR) package is an all-electron *ab initio* multiple-scattering code that provides unique capabilities for treating chemical disorder, finite-temperature magnetism, relativistic effects, and spectroscopic properties of various types of solids through its fundamental formulation in terms of the single-particle Green's function rather than eigenstates. We present ASE2SPRKKR, a comprehensive Python interface that integrates SPR-KKR into the Atomic Simulation Environment (ASE), making SPR-KKR more accessible and providing a streamlined, uniform workflow. Our implementation extends ASE's `Atoms` object to handle fractional site occupations for coherent-potential-approximation calculations while maintaining full compatibility with ASE's extensive ecosystem of structure builders, optimizers, and analysis tools. Automated input generation with validation, comprehensive output parsing, and direct MPI support enable seamless integration into high-throughput and multi-method workflows. We demonstrate the interface through representative applications: semi-infinite surface calculations reproducing Rashba-split Au(111) surface states; one-step photoemission modeling capturing matrix-element effects; exchange-parameter extraction for atomistic spin dynamics; automated empty-sphere generation for open structures; and X-ray absorption spectroscopy including magnetic circular dichroism. Beyond these demonstrations, ASE2SPRKKR is designed with transferability as a first-class concern. By grounding its architecture in FAIR principles of Findability, Accessibility, Interoperability, and Reusability, it establishes a replicable blueprint for bringing other specialized Green's function and first-principles codes into the collaborative, reproducible workflows that modern materials discovery requires.

*jminar@ntc.zcu.cz

1 Introduction

Computational materials science relies predominantly on density functional theory (DFT) [1, 2], which maps the many-body electronic problem onto a non-interacting Kohn-Sham formulation. As computational power continues to grow, a key challenge for computational physics is to exploit high-performance computing resources while providing flexible interfaces for setting up and managing sophisticated calculations across multiple codes.

The Atomic Simulation Environment (ASE) [3, 4] has emerged as the leading Python framework for atomistic simulations, currently interfacing with over 30 electronic structure codes. Most ASE-supported codes employ eigenvalue-based methods (plane-wave [5–7], localized-orbital [8–10], or augmented-wave approaches [11–13]), solving for single-particle eigenfunctions $\psi_n(\mathbf{r})$ and eigenvalues ε_n to construct electron densities and total energies. While remarkably successful for ordered crystalline systems, these variational methods face fundamental challenges when treating broken translational symmetry, substitutional disorder [14–16], or computing response functions and spectroscopic observables [17–19].

An alternative formulation based on the single-particle Green's function $G(\mathbf{r}, \mathbf{r}'; E)$ [20, 21] addresses these limitations. The Korringa-Kohn-Rostoker (KKR) method [22, 23], formulated through multiple scattering theory [19, 24], constructs the Green's function as the central computed quantity rather than deriving it from eigenstates. While mathematically equivalent to eigenvalue methods for perfect crystals, this reformulation provides decisive computational and conceptual advantages for several problem classes that remain challenging for conventional approaches [25].

For substitutional disorder in alloys A_xB_{1-x}, the coherent potential approximation (CPA) [14, 15, 26] replaces the explicit configurational average over random atomic distributions [21, 25] with a self-consistently determined effective medium. Supercell and special quasirandom structure (SQS) approaches [16, 27] face fundamental scaling limitations: representing a minority species at concentration x requires $\mathcal{O}(x^{-1})$ atoms, and since DFT cost scales cubically with system size, the total cost grows as $\mathcal{O}(x^{-3})$ [28, 29], a $\sim 10^5$-fold penalty at $x = 0.01$ relative to the equiatomic case. SQS cells carry an additional burden of reproducing multi-body correlation functions, becoming intractable for multicomponent or dilute systems [30]. The single-site CPA [19] avoids this scaling altogether, with a computational cost that is independent of x. As a single-site mean-field theory, however, it neglects short-range order; extensions to the non-local CPA [31, 32] systematically recover short-range order effects beyond the single-site approximation.

The disordered local moment (DLM) approach [33, 34] maps finite-temperature magnetic fluctuations onto orientational disorder, which is treatable within the CPA framework, enabling *ab initio* predictions of paramagnetic properties and Curie temperatures. Combined with temperature-dependent lattice vibrations treated through the alloy-analogy model [35], this provides comprehensive access to realistic finite-temperature electronic structure.

In spectroscopy, the Green's function formalism [19, 36, 37] provides a natural connection to experimental observables. Within the sudden approximation, the photocurrent measured in photoemission spectroscopy is related to the spectral function [38–40]. The one-step photoemission model [41–44] treats excitation, photoelectron propagation, and detection as a single coherent

quantum process, capturing matrix-element effects [45, 46] and photoelectron diffraction essential for the quantitative interpretation of angle-resolved photoemission (ARPES) experiments [40, 47]. X-ray absorption and magnetic circular dichroism calculations [48–51] benefit from the all-electron treatment of core states combined with a fully relativistic formalism [52, 53] incorporating spin-orbit coupling at all orders.

Transport properties emerge naturally through the Kubo-Bastin formalism [54–56], expressing conductivity tensors in terms of the Green's function and the corresponding matrix elements without requiring explicit eigenstate construction. This enables efficient calculations of the electrical conductivity [57], thermoelectric coefficients, and the anomalous and spin Hall effects [58–61] in disordered systems where Bloch-state methods struggle, while the non-equilibrium Green's function formalism gives access to Landauer-Büttiker quantum transport across interfaces [62–64].

Multi-scale magnetic modeling benefits from direct parameter extraction capabilities. The Lichtenstein formula [65–67] expresses the Heisenberg exchange parameters J_{ij} as energy integrals of Green's functions, avoiding computationally expensive total-energy differences. Gilbert damping calculations [68–71] within the linear-response formalism yield dissipation parameters for magnetization dynamics. These *ab initio* parameters feed directly into atomistic spin dynamics and Monte Carlo simulations [72–75], bridging electronic structure to mesoscale magnetic phenomena.

The Green's function serves as the central object in diagrammatic many-body perturbation theory, facilitating systematic extensions beyond standard DFT. The fully self-consistent KKR+DMFT framework [76–79] combines the KKR-GF method with dynamical mean-field theory (DMFT), introducing frequency-dependent self-energies obtained by local quantum impurity solvers. The charge self-consistency between the KKR host and the DMFT solver ensures proper feedback between local correlations and itinerant band states, enabling *ab initio* predictions of spectroscopic properties in strongly correlated systems such as transition-metal oxides and heavy-fermion compounds [80, 81], where standard DFT fails.

The spin-polarized relativistic KKR (SPR-KKR) code [82, 83] implements these methodologies within a mature, extensively validated framework. Comprehensive reviews [19, 24, 84] detail its theoretical foundations and applications. Despite its proven capabilities and impact, SPR-KKR has remained outside the ASE ecosystem, while other KKR implementations have already been integrated into Python-based workflow frameworks [85]. This limits its accessibility for researchers who would benefit from automated workflows, integration with complementary codes, and modern Python-based analysis tools.

We address these limitations through ASE2SPRKKR, which integrates SPR-KKR within ASE's infrastructure while preserving its specialized capabilities. The overall architecture of ASE2SPRKKR is illustrated in Fig. 1. ASE2SPRKKR handles KKR-specific requirements: flexible occupation specifications for CPA disorder; cached space-group symmetry for irreducible-wedge calculations; Region objects for tight-binding embedding and layered geometries. Implementation via `SPRKKRAtoms` objects extends the standard ASE `Atoms` class without breaking its base functionality, so that promoted objects remain usable by any ASE function. Expert users retain direct access to potential and task files, while simplified interfaces support common workflows. Input

validation prevents setup errors; output parsing extracts densities of states, spectral functions, spectroscopic intensities, and magnetic properties into standard Python structures.

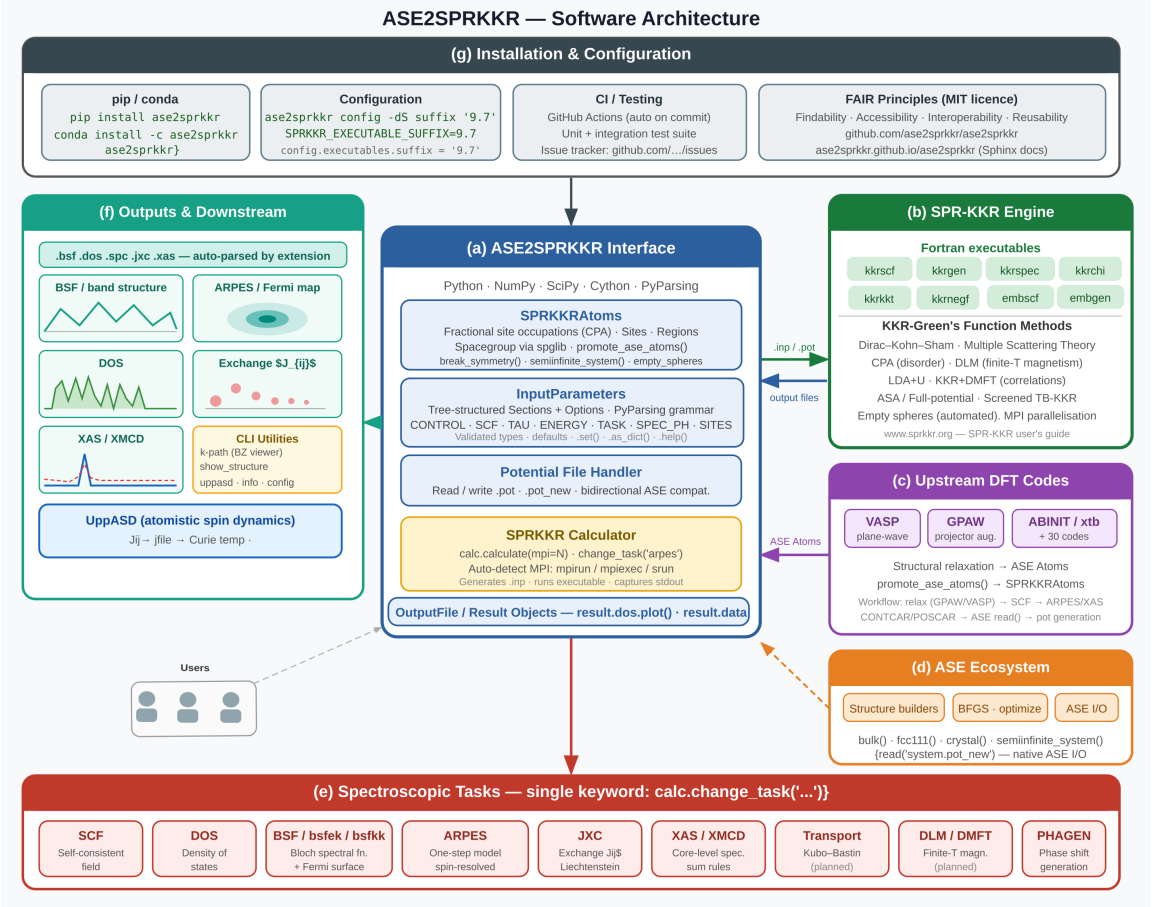


Figure 1: Software architecture of ASE2SPRKKR. (a) Core interface: `SPRKKRAtoms` extends ASE `Atoms` with fractional site occupations (CPA), spglib symmetry, and `Region` support; `InputParameters` provides a validated, tree-structured parameter hierarchy (PyParsing grammar); the `Potential File Handler` manages bidirectional `.pot`/`.pot_new` I/O; the `SPRKKR Calculator` drives input generation, MPI execution, and task switching via `change_task()`; structured `Result` objects expose arrays and built-in `plot()` methods. **(b)** SPR-KKR Fortran backend [19] (`kkrscf`/`kkrgen`/`kkrspec`/`kkrchi`/`kkrkkt`/`kkrnegf`/`embscf`/`embgen`): CPA [14], DLM [33], LDA+U [86], KKR+DMFT [79], ASA/full-potential, screened TB-KKR [87], and layer doubling [88]. **(c)** Upstream DFT codes (VASP, GPAW, ABINIT, 30+): relaxed `Atoms` structures are promoted to `SPRKKRAtoms` via `promote_ase_atoms()` without manual file conversion. **(d)** ASE ecosystem [3]: structure builders, BFGS optimizer, and native `.pot_new` I/O. **(e)** Available tasks via `calc.change_task()`: SCF, DOS, BSF/Fermi surface, ARPES (one-step model [41, 42]), exchange J_{ij} (JXC [65]), XAS/XMCD with sum rules [89, 90], and PHAGEN; DLM/DMFT and transport (Kubo-Bastin [55, 56]) workflows are planned. **(f)** Outputs and downstream: parsed result objects, CLI utilities (`k-path`, `plot`), and one-command J_{ij} export to UppASD [74] for Curie temperatures. **(g)** Distribution (pip/conda), executable configuration, GitHub Actions CI, and FAIR principles compliance (MIT license, Sphinx docs, public issue tracker).

This integration enables hybrid workflows combining complementary methods: structural relaxations with other codes followed by spectroscopic calculations with SPR-KKR's one-step photoemission model; high-throughput screening of ordered structures followed by systematic disorder and temperature exploration via CPA and DLM; exchange parameter extraction followed by spin dynamics parameterization. ASE2SPRKKR substantially lowers barriers for exploiting the unique capabilities developed over decades, particularly for problems involving disorder, magnetism, relativistic effects, and spectroscopy, where the KKR-GF framework excels. Ongoing

development aims to further streamline workflows, incorporate additional SPR-KKR features, and enhance the accessibility and versatility of this powerful framework for the broader scientific community.

This paper is organized as follows. Section 2 outlines the software architecture, core components, and integration patterns. Section 3 demonstrates the capabilities of our approach through representative applications. Section 4 summarizes the current implementation and the development roadmap. All calculations employed SPR-KKR version 9.7. The Appendix provides the theoretical foundations and SPR-KKR implementation details necessary for understanding the interface design.

2 ASE2SPRKKR Software Architecture

Python [91] has become one of the most widely used programming languages in the scientific community, primarily due to its extensive ecosystem of libraries for numerical and scientific computation. Among these, `NumPy` [92] provides efficient numerical array manipulation, `SciPy` [93] implements a wide variety of scientific algorithms including optimization, integration, and interpolation, and `Cython` [94] enables high-performance interoperability between Python and compiled languages such as C or Fortran. Given this rich ecosystem and its flexibility, Python was a natural choice for developing the ASE2SPRKKR interface.

To facilitate interoperability with other atomistic simulation codes and minimize implementation effort, ASE2SPRKKR is built on the Atomic Simulation Environment (ASE) [3]. ASE provides a standardized and extensible framework for describing, manipulating, and visualizing atomic structures directly within Python. It offers a unified interface for setting up systems, running simulations, and analyzing results using a wide range of back-end calculators—from `ABINIT` to `VASP` and `xtb`. ASE2SPRKKR serves as a bridge that allows users to perform spin-polarized and relativistic electronic structure calculations using the SPR-KKR package directly within ASE. The complete ASE documentation is available online at `https://wiki.fysik.dtu.dk/ase/`.

ASE2SPRKKR does *not* reimplement SPR-KKR functionality. It generates input files, manages execution, and parses outputs, maintaining complete compatibility with native SPR-KKR. Users can mix ASE2SPRKKR workflows with manual input file editing, which is useful for experimental features or troubleshooting.

2.1 Installation and Configuration

Installation of ASE2SPRKKR follows standard Python package management practices. The recommended approach uses pip (Package Installer for Python), which is the standard way to install Python packages:

```
pip install ase2sprkkr
```

Using a Python virtual environment is highly recommended but not required. Alternatively, the Anaconda distribution system [95] can be used:

```
conda install -c ase2sprkkr ase2sprkkr
```

ASE2SPRKKR is under active development. The stable release version can be supplemented with the beta version, which often includes new features and bug fixes:

```
pip install --pre ase2sprkkr
```

For cutting-edge features, the development branch of the Git repository can be accessed directly:

```
 pip install git+https://github.com/ase2sprkkr/ase2sprkkr.git@develop
```

To run SPR-KKR calculations with ASE2SPRKKR, the SPR-KKR executables must be compiled and registered in the ASE2SPRKKR configuration. The configuration file path can be displayed using:

```
ase2sprkkr config -p
```

The executable suffix is set directly via:

```
ase2sprkkr config -dS executables.suffix '9.7'
```

For workflows requiring multiple SPR-KKR versions, the suffix can be modified programmatically in calculation scripts:

```
from ase2sprkkr.configuration import config
config.executables.suffix = '9.7'
```

or through environment variables:

```
export SPRKKR_EXECUTABLE_SUFFIX=9.7
```

SPR-KKR executables should be in the `PATH` environment variable. If not, the executable directory can be specified in the `executables.dir` configuration variable. Additional configuration options are accessible via:

```
ase2sprkkr config -h
```

2.2 Extended Atomic Structure: SPRKKRAtoms

ASE [3] is a powerful and versatile software package that supports a wide range of codes, from *ab initio* calculations to molecular dynamics. However, support for certain properties specifically required by SPR-KKR is either limited, not very user-friendly, or entirely missing. For example, occupation is represented only as an array of occupation dictionaries associated with specific site types. Consequently, if one wants to change or break the symmetry of a structure, the entire

array has to be rebuilt. While this is certainly possible, it sacrifices the elegance and simplicity of a scripting interface that is accessible even to users with limited programming experience. To address this and allow typical users to easily run SPR-KKR calculations, the functionality of ASE `Atoms` objects has been extended through `SPRKKRAtoms`.

To allow a smooth transition from ASE to ASE2SPRKKR and back, the method `promote_ase_atoms` has been introduced. This method adds a parent class (in the object-oriented programming sense) to the existing ASE `Atoms` object. The promotion does not alter the original object in any other way: it can still be used in the ASE way as before and merely gains additional functionality. The promotion also happens automatically when an `Atoms` object is passed to ASE2SPRKKR functions, maintaining full backward compatibility. As an example, consider the Ni_2FeGa Heusler alloy with partial Fe/Co substitution, whose structure is shown in Fig. 2:

```
from ase2sprkkr import SPRKKRAtoms
from ase.visualize import view
import numpy as np

# Create ordered Heusler structure
# 1. Define primitive FCC cell vectors
primitive_cell = np.array([
    [0.0, 0.5, 0.5],
    [0.5, 0.0, 0.5],
    [0.5, 0.5, 0.0]
]) * 5.748

# 2. Define the scaled positions
scaled_positions = [
    [0.25, 0.25, 0.25], # Ni (1)
    [0.75, 0.75, 0.75], # Ni (2)
    [0.5,  0.5,  0.5],   # Fe
    [0.0,  0.0,  0.0],   # Ga
]

# 3. Build the Atoms object directly
Ni2FeGa = SPRKKRAtoms(
    symbols=['Ni', 'Ni', 'Fe', 'Ga'],
    scaled_positions=scaled_positions,
    cell=primitive_cell,
    pbc=True
)

# 4. Define Fe/Co disorder
Ni2FeGa.sites[2].site_type.occupation.set({'Fe':0.75, 'Co':0.25})
```

```
# 5. View and save the structure
view(Ni2FeGa,viewer='avogadro',repeat=[2,2,2])
Ni2FeGa.potential.save_to_file("Ni2FeGa.pot")
```

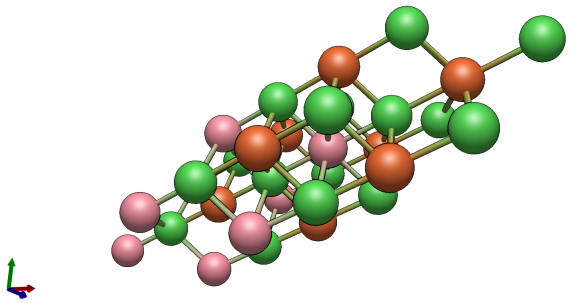


Figure 2: Full Heusler alloy Ni_2FeGa structure (space group $Fm\bar{3}m$) with partial Fe/Co substitutional disorder. The $L2_1$ structure contains four interpenetrating fcc sublattices. The Fe site (index 2 in the code above) exhibits compositional disorder with 75% Fe / 25% Co, representing a chemically complex magnetic Heusler compound suitable for CPA treatment.

Sites The `sites` property, which is in fact stored as an ASE array to retain smooth manipulation with `SPRKKRAtoms` objects via summing and slicing, provides a user-friendly way to specify atomic sites, their occupations, and to associate additional SPR-KKR-specific properties, such as radial potentials or charges.

Each `Site` holds the magnetization of that site and references a `SiteType` object, which is shared by all symmetry-equivalent sites.

`SiteType` object carries:

- **Occupation dictionary**: maps species to fractional occupations, summing to unity
- **Radial potential and charge**: result of a self-consistent calculation (once performed)
- **Radial mesh**: for the potential and charge

Moreover, `Site` and `SiteType` allow user-friendly manipulation of sites, such as breaking symmetry or changing occupations.

Some properties of a `site` or `site_type` duplicate the original ASE properties (e.g., `symbols` array, occupation array). When a `site` property is edited, the change is mirrored to the corresponding ASE property to maintain backward compatibility. However, the reverse is not true due to the inherent limitations of ASE. Therefore, modifications of ASE properties, such as the `symbols` of `Atoms` objects, are recommended only before promotion.

When saving the potential, e.g. at the beginning of the `calculate` or `save_input` methods of `calculator`, the values of `SPRKKRAtoms` are written to the `potential` using the `set_from_atoms` methods of sections. In contrast, reading potentials, or manually updating atoms via the

`update_atoms` method of a `Potential`, is done using the `update_atoms` methods.* In this way, the original structure of `potential` remains accessible, allowing low-level manipulations with `potential` data if needed, while enabling user-friendly editing of the common properties of the calculated structure via the `SPRKKRAtoms` object and its `sites` property.

Regions SPR-KKR supports tight-binding mode calculations, in which two 3D bulk materials are connected by a transition region that is periodic in two dimensions. For this purpose, the ASE `Atoms` object is extended with `Region` objects (accessible via `[add_|set_|remove_]regions` methods).

Each `Region` has its own cell, periodicity, and symmetry information, as well as a mask specifying which sites belong to the region.

Spacegroup information SPR-KKR requires information about the symmetry of atoms and regions. Spglib [96] automatically determines space group symmetry (space group 225, $Fm\bar{3}m$ for the Heusler structure above), linking symmetry-equivalent sites. Since computing this information can be computationally expensive, the results are stored in the `spacegroup_info` property of the `SPRKKRAtoms` object. This information is used during initialization of the `sites` array and when building the potential just before a calculation.

2.3 ASE2SPRKKR Data Objects

Many of the ASE2SPRKKR objects, for example `InputParameters`, `Potential`, and all `OutputFile` objects, share the same implementation: a tree-like data structure. Each item of the data-tree has an associated *definition class*, an approach similar to the way in which, e.g., the well-known NOMAD [97] organizes its data on the Python level.†

Definition classes describe both the structure of data and the way in which the data is stored in a file. Each section and option value of these objects has a `definition` property. The definition of a section states which are the members of the section, and the rules for how to read/write them: i.e., how they are separated (whether by newlines, spaces, tabs, etc.), whether they have a fixed order or not, and so on. The definition of a value specifies the data type of the value, whether it is required, and the default value for the option. The definitions also contain documentation for the data.

Data types are described by descendants of the `GrammarTypes` object; these objects describe the values and their format in the resulting file. They can be nested: one option can hold more than one value, either resulting in a NumPy array [92] or in a tuple of different types, or even 2D tables in the data (see the example in Figs. 3 and 4).

Saving the data into a file is straightforward: the data tree is traversed, and each definition takes care of saving itself and its child objects. Parsing is more involved: for this purpose, the

*In the current state of development, not all sections of `potential` are parsed; some are just stored as raw data and retained in the `potential` object.

†However, the implementation differs, since NOMAD uses metaclasses [97] for this purpose (which has its pros and cons), while ASE2SPRKKR describes the structure of the data using regular Python classes.

definition generates a grammar using PyParsing [98]. Using the grammar, the file is parsed, resulting in a dictionary that is used to set up the resulting object.

Using this mechanism, defining new sections within a `Potential`, adding new values to `InputParameters`, or introducing a new task type becomes very easy. Fig. 3b shows an example of how a Potential section can be defined to describe the unit cell of a material being calculated. As demonstrated in Fig. 4a, defining even a relatively complex nested structure is straightforward. Example data that such a section may contain are shown in Fig. 3a in their raw text form, and Fig. 4b presents their Python representation.

(a) An example of the LATTICE section of an SPR-KKR potential file

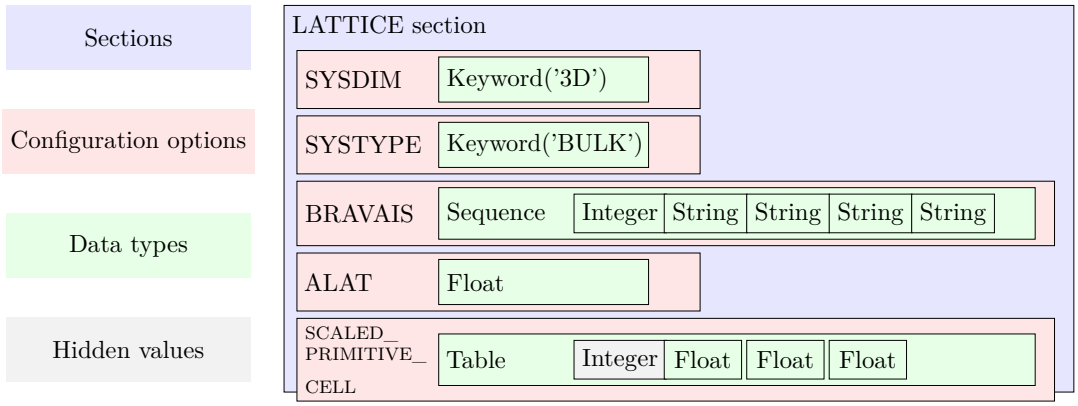
```
LATTICE
SYSDIM       3D
SYSTYPE      BULK
BRAVAIS           4        orthorombic primitive      mmm    D_2h
ALAT          7.1941868392
A(1)          0.0000000000   -0.5000000000   -0.5000000000
A(2)         -1.0000000000    0.0000000000    0.0000000000
A(3)          0.0000000000    1.0000000000   -1.0000000000
```

(b) The ASE2SPRKKR definition of the LATTICE section

```
PotSectionDefinition(
    name='LATTICE',
    members = [
      V('SYSDIM', DefKeyword('3D')),
      V('SYSTYPE', DefKeyword('BULK')),
      V('BRAVAIS', Sequence(int, *([str]*4), allowed_values =
      ↪  cell_symmetries)),
      V('ALAT', float),
      V('SCALED_PRIMITIVE_CELL', Table([float]*3,
                numbering=Integer(prefix='A(', postfix=')',
                ↪  format='<10'),
                length=3
        ),
    ],
    has_hidden_members=True,
    result_class = LatticeSection)
```

Figure 3: Mapping an SPR-KKR potential file to its ASE2SPRKKR definition. (a) The raw `LATTICE` section as written in an SPR-KKR potential file. (b) The corresponding `PotSectionDefinition` in ASE2SPRKKR, which declares each member's name, expected data type, and validation rules.

(a) The visualization of the definition of the LATTICE section (Fig. 3b)



(b) The resulting data from the example section above (Fig. 3a)

```
LatticeSection<{
    'SYSDIM': '3D',
    'SYSTYPE': 'BULK',
    'BRAVAIS': [4, 'orthorombic', 'primitive', 'mmm', 'D_2h'],
    'ALAT': 7.1941868392,
    'SCALED_PRIMITIVE_CELL': array([[ 0. , -0.5, -0.5],
                    [-1. ,  0. ,  0. ],
                    [ 0. ,  1. , -1. ]]),
}>
```

Figure 4: From definition to parsed object: how ASE2SPRKKR handles the LATTICE section. (a) A schematic of the definition in Fig. 3b, showing the nested hierarchy of section, configuration options, data types, and hidden values. (b) The resulting LatticeSection object obtained by parsing the data in Fig. 3a, with each entry converted to its native Python type.

2.4 Calculator Interface and Execution Model

Following ASE's calculator paradigm, computation proceeds in three steps: instantiation, parameter configuration, and execution (performed via the `calculate` method).

```
from ase2sprkkr import SPRKKR

calculator = SPRKKR(atoms=Ni2FeGa)
calculator.input_parameters.SCF.NITER = 300
calculator.input_parameters.TAU.NKTAB = 150
calculator.calculate()
```

The `calculate()` method:

1. Generates the potential file from the `atoms` structure.
2. Writes the task-specific input file from `input_parameters`.

3. Invokes the SPR-KKR executable, using MPI if available.
4. Monitors the execution and captures the output.
5. Parses the results into task-dependent result objects.
6. Returns the result object with the computed properties.

MPI auto-detection looks for the common MPI launchers [99] `mpirun` and `mpiexec`, and for `srun` of the Slurm workload manager [100, 101].

Explicit control supports custom configurations:

```
calculator.calculate(mpi=16)  # 16 processes
calculator.calculate(mpi=['/opt/mpi/bin/mpirun','-n','16'])
```

2.5 Input Parameter Management

The `InputParameters` object provides hierarchical access to task-specific parameters organized by sections (CONTROL, SCF, TAU, ENERGY, TASK, etc.). Each parameter includes type validation, range constraints, and documentation:

```
# Task selection modifies available parameters
calculator.change_task('arpes')

# Batch parameter setting with validation
calculator.input_parameters.set({
    'ENERGY.EMINEV': -4.0,
    'ENERGY.EMAXEV': 2.0,
    'ENERGY.NE': 300,
    'SPEC_PH.EPHOT': 50.0,
    'TASK.MILLER_HKL': [1,1,1],
})
```

Parameter values can be extracted for further processing:

```
calculator.input_parameters.as_dict()
```

Documentation of available parameters is provided in the SPR-KKR manual [83]. Brief parameter descriptions are accessible from the command line:

```
ase2sprkkr info --task                # show available tasks
ase2sprkkr info --task=scf            # show SCF task sections
ase2sprkkr info --task=scf.control    # show CONTROL section parameters
```

or in interactive Python:

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(task='scf')
calculator.input_parameters.CONTROL.help()
```

Input parameters can be read from files to reproduce or modify previous computations:

```
from ase2sprkkr import InputParameters
calculator.input_parameters = \
    InputParameters.from_file('previous.inp')
```

Input parameter properties are validated to prevent setting parameters to invalid values. However, ASE2SPRKKR may not yet support certain parameters, or validation may be overly strict. In these cases, restrictions can be overridden:

```
ip = calculator.input_parameters
# Add unsupported parameter
ip.CONTROL.add('MY_NEW_PARAMETER', its_value)
# Set parameter to unsupported value
ip.SCF.NITER.set_dangerous('infinity')
```

These should be used sparingly, as they bypass safety checks.

2.6 Potential File Handling

Potential files encode the system structure, including the lattice, atomic positions, site types, and their occupations. After the user runs a self-consistent calculation (SCF task), a new `original_file.pot_new` file is generated, which also contains the results of the calculation, such as potentials, charge densities, and magnetic moments. In some advanced workflows, such a "converged" potential can also be created manually.

Non-SCF tasks (DOS, ARPES, BSF, etc.) require such converged potentials:

```
calc = SPRKKR()
result = calc.calculate(
    potential='converged.pot_new',
    input_parameters='dos'
)
```

Potential files can be read using either the ASE2SPRKKR API:

```
from ase2sprkkr import Potential
pot = Potential.from_file('system.pot_new')
atoms = pot.atoms
print(atoms.symbols)
```

or using ASE's native `read` method:

```
from ase.io import read
atoms = read('system.pot_new')
magnetic_moments = [site.moments for site in atoms.sites]
```

This enables workflows where structural optimizations with other codes precede spectroscopic calculations with SPR-KKR.

2.7 Output Processing and Visualization

Task-dependent result objects provide property access with built-in visualization:

```
# DOS calculation
result = calculator.calculate(input_parameters='dos')
result.dos.plot()                # matplotlib visualization
e_fermi = result.dos.EFERMI()    # Fermi energy
energies = result.dos.ENERGY()   # Energy grid as NumPy array
```

Independent file parsing supports post-processing:

```
from ase2sprkkr import OutputFile
bsf = OutputFile.from_file('calc.bsf')
bsf.plot(IQ=[1, 2])  # plot selected atomic layers (IQ)
bsf.data  # access raw data arrays
```

File type inference from extensions (.bsf, .dos, .spc, .jxc, .rat) or content analysis automatically selects the appropriate parser.

2.8 Workflow Integration Patterns

ASE2SPRKKR is designed to interoperate with the broader ASE ecosystem and external codes, enabling several practical integration patterns for multi-step computational studies.

Sequential multi-method calculations: A common workflow combines a plane-wave DFT code for structural relaxation with SPR-KKR for spectroscopic post-processing. Since ASE2SPRKKR operates on standard ASE `Atoms` objects, the relaxed geometry produced by any ASE-compatible calculator can be passed directly to SPR-KKR without manual file conversion. In the example below, GPAW is used to obtain the equilibrium geometry, after which the optimized structure is forwarded to SPR-KKR for an ARPES calculation:

```
from ase.spacegroup import crystal
from ase.optimize import BFGS
from gpaw import GPAW, PW, FermiDirac
from ase2sprkkr import SPRKKR

atoms = crystal(
```

```
    symbols=['Mo', 'Se'],
    basis=[(1/3, 2/3, 1/4),
           (1/3, 2/3, 0.621)],
    spacegroup=194,
    cellpar=[3.288, 3.288, 12.900, 90, 90, 120]
    )

atoms.calc = GPAW(
    mode=PW(650),
    xc='PBE',
    kpts={'size': (8, 8, 4), 'gamma': True},
    occupations=FermiDirac(0.05),
    txt='mose2_bulk_relax.txt'
)
# Relax internal coordinates only:
opt = BFGS(atoms, trajectory='mose2.traj', logfile='mose2.log')
opt.run(fmax=0.02)
atoms.write('mose2_relaxed.cif')
# Switch to SPR-KKR calculator; empty spheres fill the open
# van der Waals gap
calc = SPRKKR(atoms=atoms, empty_spheres=True)
print("Running SPR-KKR SCF...")
scf_result = calc.calculate()

# Reuse the same calculator and switch task - this preserves
# the converged potential from the SCF step
print("Running ARPES...")
calc.change_task('arpes')
arpes_result = calc.calculate()
print("ARPES finished.")
```

High-throughput composition scanning: The `SPRKKRAtoms` site-occupation model makes ASE2SPRKKR particularly well suited for systematic alloy studies. By iterating over a composition parameter and updating the fractional occupancies of a given site, one can efficiently scan the full concentration range of a substitutional alloy series within a single script. The CPA treatment is handled internally by SPR-KKR, so no supercells are required. The following example scans the Fe_xCo_{1-x} composition in a Heusler structure across 21 uniformly spaced concentrations. The structure is read from the potential file created in the example in Section 2.2:

```
import numpy as np
from ase.io import read
from ase2sprkkr import SPRKKR
```

```
Ni2FeGa = read("Ni2FeGa.pot")

results = []
for x in np.linspace(0, 1, 21):
    atoms = Ni2FeGa.copy()
    atoms.sites[2].site_type.occupation.set({ 'Fe': x, 'Co': 1-x })
    calc = SPRKKR(atoms=atoms)
    result = calc.calculate()
    results.append( (x, result.energies.total()) )
```

Multi-scale magnetic modeling: ASE2SPRKKR provides an interface to atomistic spin-dynamics simulations by extracting Heisenberg exchange coupling constants J_{ij} from a converged KKR potential, exporting them in the input format expected by the UppASD code [74], and optionally running UppASD. This two-step procedure, first-principles exchange parameters fed into a classical spin Hamiltonian, allows finite-temperature magnetic properties such as the Curie temperature and magnon spectra to be obtained within a single automated workflow. For already computed exchange coupling constants, the `ase2sprkkr uppasd` utility creates the `posfile`, `momfile`, and `jfile` input files required by UppASD [74].

2.9 Command-Line Utilities

The `ase2sprkkr` tool supplements the Python API:

```
ase2sprkkr example            # locate and list example scripts
ase2sprkkr shell --example 8 --jupyter # run example 8
ase2sprkkr config -e          # edit configuration
ase2sprkkr plot file.bsf      # visualize results
ase2sprkkr k-path sys.pot     # interactive k-path selection
ase2sprkkr uppasd Ni2FeGa_JXC.jxc -u  # convert J_ij for UppASD
ase2sprkkr info --task=scf    # parameter documentation
```

The `k-path` tool launches an interactive Brillouin zone viewer. Clicking high-symmetry points generates SPR-KKR-compatible path specifications that automatically handle surface reciprocal lattice vectors.

3 Applications

This section demonstrates ASE2SPRKKR through representative applications that highlight different aspects of the interface. Each example is chosen to illustrate a distinct workflow pattern: task switching to spectroscopic mode (Sec. 3.1), semi-infinite surface construction (Sec. 3.2), multi-scale parameter extraction with third-party export (Sec. 3.3), automated basis completion for open structures (Sec. 3.4), and element-selective spectroscopy (Sec. 3.5). In each case, the focus is on how ASE2SPRKKR expresses the calculation as a compact Python script, replacing the manual file preparation that SPR-KKR conventionally requires.

3.1 One-Step ARPES for Au(111)

This example demonstrates task switching from a ground-state calculation to a photoemission task on the same potential, and illustrates how geometry-specific parameters such as surface orientation, photon energy, and detector configuration are expressed as plain Python dictionaries. The `arpes` task keyword invokes the full one-step photoemission implementation [41, 42]; no modification to the calculator object or potential is required.

3.1.1 Bulk SCF Calculation

```
from ase.build import bulk
from ase2sprkkr import SPRKKR

Au = bulk('Au', 'fcc', a=4.08)
opts = {
    'CONTROL.KRMT': 4,
    'TAU.NKTAB3D': 1000,
    'SCF.VXC': 'VWN',
    'SCF.FULLPOT': False,     # ASA mode
    'SCF.NITER': 2000,
    'SCF.MIX': 0.1,
    'SCF.TOL': 1E-5,
}
calc = SPRKKR(atoms=Au, options=opts)
calc.calculate()
```

A DOS calculation on the converged potential serves as an intermediate check before launching the more expensive ARPES task. The resulting density of states is shown in Fig. 5.

```
DOS_input = {
    'ENERGY.ImE': 0.001,        # Im(E) in Ry
    'ENERGY.EMINEV': -12.0,
    'ENERGY.EMAXEV': 7.0,
    'ENERGY.NE': 1000,
    'TAU.NKTAB': 2500,
}
result = calc.calculate(
    input_parameters='dos',
    options=DOS_input,
)
result.dos.plot()
```

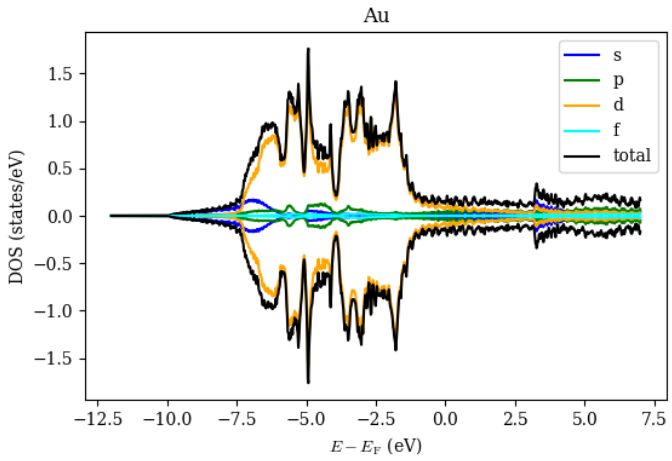


Figure 5: Bulk Au DOS obtained via the `dos` task on the self-consistent potential. The result object returned by `calculate()` is passed to a built-in `plot()` method, requiring no external post-processing.

3.1.2 ARPES Intensity Calculation

Switching to the ARPES task reuses the same potential;[‡] photon geometry, surface orientation, and detector configuration are specified entirely within the options dictionary:

```
ARPES_input = {
    'TAU.NKTAB': 500,
    'ENERGY.EMINEV': -4.0,
    'ENERGY.EMAXEV': 2.0,
    'ENERGY.EWORK_EV': 5.0,         # Work function (eV)
    'ENERGY.NE': 300,
    'ENERGY.IMV_INI_EV': 0.02,      # Initial state broadening
    'ENERGY.IMV_FIN_EV': 2.0,       # Final state broadening
    'TASK.MILLER_HKL': [1, 1, 1],   # Surface orientation
    'SPEC_PH.THETA': 45,             # Photon incident angle (deg)
    'SPEC_PH.PHI': 90,               # Photon azimuth (deg)
    'SPEC_PH.EPHOT': 50,             # Photon energy (eV)
    'SPEC_STR.N_LAYDBL': [12, 12], # Layer doubling
    'SPEC_STR.N_LAYER': 30,          # Total number of layers
    'SPEC_EL.THETA': [-20, 20],     # Detector angle range (deg)
    'SPEC_EL.NT': 200,               # Angular points
    'SPEC_EL.SPOL': 4,               # Spin-resolved output
}
result = calc.calculate(
    input_parameters='arpes',
    options=ARPES_input
)
```

[‡]If the same calculator is used, the already converged potential is used automatically. If a new calculator is created, the potential file name `Au.pot_new` must be passed to the `calculate` method via the `potential` argument.

```
result.arpes.plot()
```

The computed ARPES map, obtained by the `plot` function, is shown in Fig. 6.

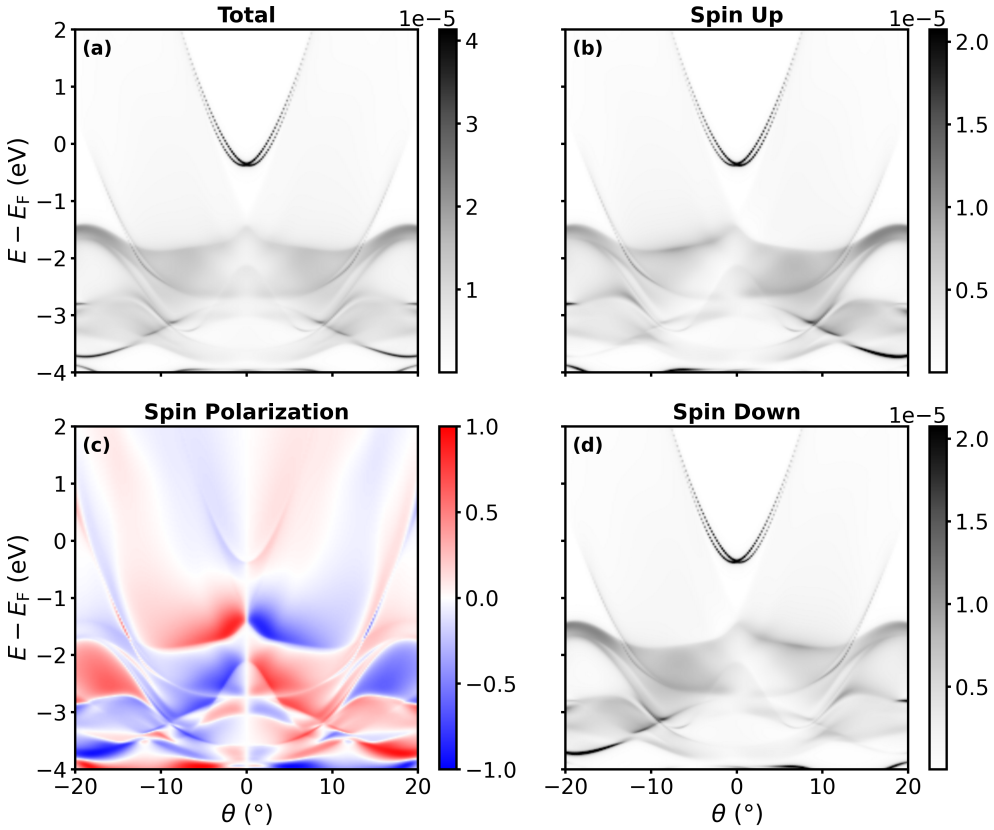


Figure 6: One-step ARPES $I(\theta, E)$ at $h\nu = 50$ eV around the $\bar{\Gamma}$ point for Au(111) returned by the `arpes` task: (a) total intensity, (b) spin-up, (c) spin polarization, and (d) spin-down. The spin polarization is defined as $P_{\mathrm{spin}}(\theta) = [I_\uparrow(\theta) - I_\downarrow(\theta)]/[I_\uparrow(\theta) + I_\downarrow(\theta)]$. For the surface state it reflects the Rashba spin-momentum locking, whereas the polarization of the spin-degenerate bulk bands is photoemission-induced, arising from spin-orbit coupling in the transition matrix elements, and depends on the light polarization and the experimental geometry. All four panels are produced in a single calculation and accessed through the structured result object.

A constant-energy Fermi map is obtained by changing only the energy window, the k-grid, and the spin-quantization axis, while reusing all structural and photon parameters through dictionary unpacking:

```
ARPES_input_fermi = {
    **ARPES_input,   #copy values from the previous options
    'ENERGY.EMINEV': 0.0,
    'ENERGY.EMAXEV': 0.0,
    'ENERGY.NE': 1,
    'SPEC_EL.KA':[-0.5,-0.5], # Corner of k-grid
    'SPEC_EL.K1':[1,0], # First vector of k-grid
    'SPEC_EL.K2':[0,1], # Second vector of k-grid
    'SPEC_EL.NK1':120, # Number k-points along K1
    'SPEC_EL.NK2':120, # Number k-points along K2
```

```
    'SPEC_EL.PSPIN':[0,0,1], # Spin-quantization axis
}
result = calc.calculate(
    potential='Au.pot_new',
    input_parameters='arpes',
    options=ARPES_input_fermi
)
result.arpes.plot()
```

The corresponding constant-energy map is shown in Fig. 7.

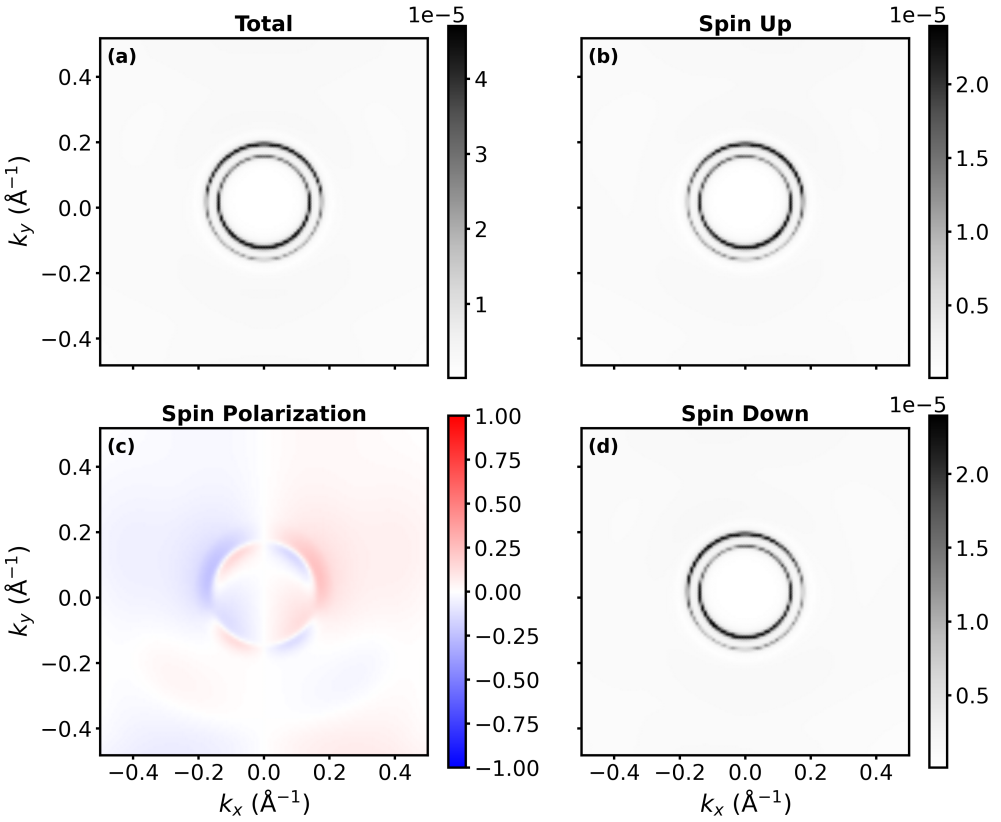


Figure 7: One-step ARPES Fermi map $I(k_x, k_y, E_F)$ at hν = 50 eV around the $\bar{\Gamma}$ point for Au(111), returned by the `arpes` task: (a) total intensity, (b) spin-up, (c) spin polarization, and (d) spin-down. The Rashba-split surface state appears as two concentric rings, and the spin-momentum locking is visible as a spin-resolved intensity asymmetry, $P_{\mathrm{spin}}(k) = [I_\uparrow(k) - I_\downarrow(k)]/[I_\uparrow(k) + I_\downarrow(k)]$, projected on the spin-quantization axis set by `SPEC_EL.PSPIN`, which reverses sign between the inner and outer ring. All four panels are produced in a single calculation and accessed through the structured result object.

3.2 Au(111) Surface: Bloch Spectral Function

This example demonstrates ASE2SPRKKR's semi-infinite surface construction and the two-stage SCF workflow. Starting from a standard ASE slab, `semiinfinite_system` builds the interaction-zone geometry shown in Fig. 8; the interface then writes the potential and input files of each stage, so that the user only passes the converged potential of one stage to the next.

3.2.1 System Construction and Convergence

```
from ase.build import fcc111
from ase2sprkkr.sprkkr.build import semiinfinite_system
from ase2sprkkr import SPRKKR
from ase.visualize import view

# Construct (111) slab: 3 atomic layers
atoms_bulk = fcc111('Au', size=(1, 1, 3), periodic=True)

# Create semi-infinite geometry along axis=2 (z); vacuum side
# (empty spheres) is used because atoms2=None
Au111 = semiinfinite_system(
    atoms=atoms_bulk,
    repeat=[3, 3],
    atoms2=None,
    axis=2
)
view(Au111, viewer='avogadro')
```

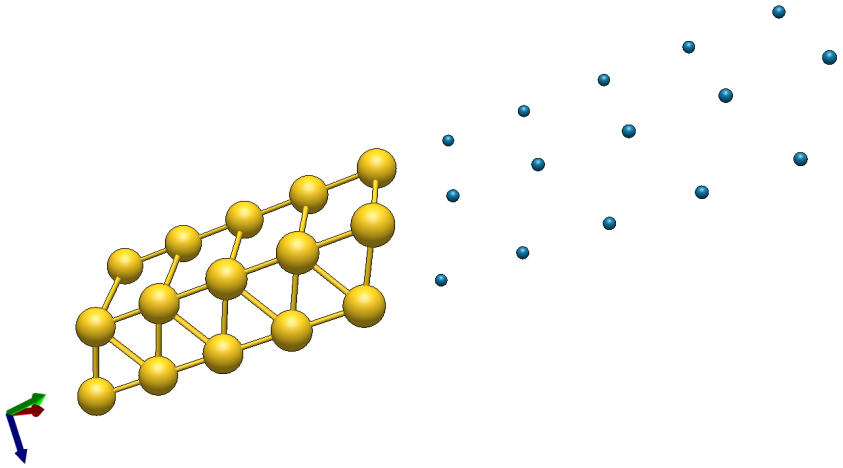


Figure 8: Semi-infinite Au(111) interaction zone constructed by `semiinfinite_system`, where the large golden spheres are the Au while the small spheres are the empty (vacuum) sites on the surface side.

The semi-infinite construction requires two SCF calculations: one for the bulk ("left") region, followed by one for the interaction zone embedded between the bulk and the vacuum. First, the bulk region is converged:

```
from ase2sprkkr import SPRKKR
opts_bulk = {
    'CONTROL.KRMT': 2,        # Muffin-tin radius scheme
```

```
    'SITES.NL': 4,              # Angular momentum lmax=3
    'TAU.KKRMODE': 'TB',        # Tight-binding (screened)
    'TAU.NKTAB3D': 1000,        # 3D k-mesh (~10x10x10)
    'TAU.NKTAB2D': 100,         # 2D k-mesh (~10x10)
    'TAU.CLURAD': 2.7,          # Cluster radius (units of ALAT)
    'SCF.VXC': 'VWN',           # Vosko-Wilk-Nusair XC
    'SCF.NITER': 500,           # Number of SCF iterations
    'SCF.MIX': 0.1,             # Broyden mixing parameter
    'SCF.TOL': 1E-5,            # Potential convergence criterion
}
calc = SPRKKR(atoms=Au111, options=opts_bulk)
calc.calculate()
```

The second stage copies the bulk options with `dict.copy()` and overrides only the parameters that differ for the surface zone, because the options are plain Python dictionaries, incremental changes do not need to duplicate the full input:

```
opts_surf = opts_bulk.copy()
opts_surf.update({
    'TAU.NKTAB2D': 71,
    'SCF.NITER': 2000,
    'SCF.MIX': 0.01,       # Smaller mixing for stability
})
calc = SPRKKR(
    potential='Au12X12.pot_new',
    options=opts_surf
)
calc.calculate()
```

3.2.2 Bloch Spectral Function Calculation

Bloch spectral function (BSF) calculations require specifying a k-path along which the function is evaluated. The built-in command-line tool can assist with its generation. The example uses the converged potential obtained in the previous example (Section 3.2.1); the second SCF run writes it to `Au12X12.pot_new_new`.

```
ase2sprkkr k-path Au12X12.pot_new_new
```

Executing this command opens an interactive pop-up window of the corresponding Brillouin zone (Fig. 9), where the desired path for the BSF task can be selected.

Running a spectral-function calculation requires only modifying `input_parameters`; all structural information and the potential are inherited from the previously converged calculation.

```
from ase2sprkkr import SPRKKR
```

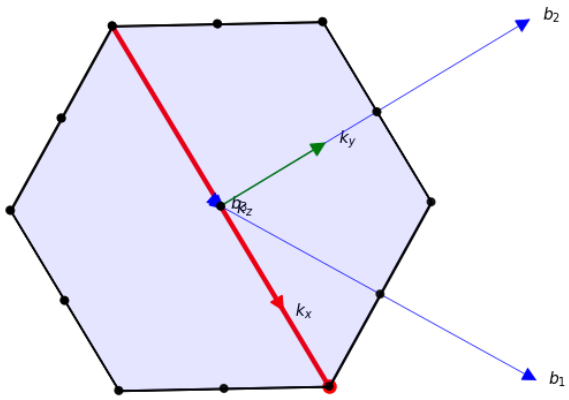


Figure 9: Interactive Brillouin zone tool launched by `ase2sprkkr k-path`. The hexagonal zone shows high-symmetry points $\bar{\Gamma}$, $\bar{M}$, and $\bar{K}$; clicking defines the path and the tool writes the corresponding SPR-KKR coordinate blocks directly into the input dictionary.

```
BSF_input = {
    'TAU.NKTAB3D': 1000,
    'TAU.NKTAB2D': 71,
    'TAU.CLURAD': 2.7,
    'ENERGY.ImE': 0.002,          # Im(E) in Ry (about 27 meV)
    'ENERGY.EMINEV': -4.0,
    'ENERGY.EMAXEV': 2.0,
    'ENERGY.NE': 400,
    # parameters generated by ase2sprkkr k-path
    'TASK.NK': 300,           # nb. of k-points along the path
    'TASK.KA': [[-0.66667, 0.0, 0.0], [0.0, 0.0, 0.0]], # start points of the
    ↪  path segments
    'TASK.KE': [[0.0, 0.0, 0.0],  [0.66667, 0.0, 0.0]], # end points of the
    ↪  path segments
}
calc = SPRKKR()
result = calc.calculate(
    potential='Au12X12.pot_new_new',
    input_parameters='bsfek',
    options=BSF_input
)
result.bsf.plot()
```

The spin-resolved BSF obtained from this `bsfek` task is shown in Fig. 10.

Layer-resolved data can also be accessed by parsing the output file directly, again without manual file handling:

```
from ase2sprkkr import OutputFile
bsf = OutputFile.from_file('Au12X12_BLOCHSF_spol.bsf')
```

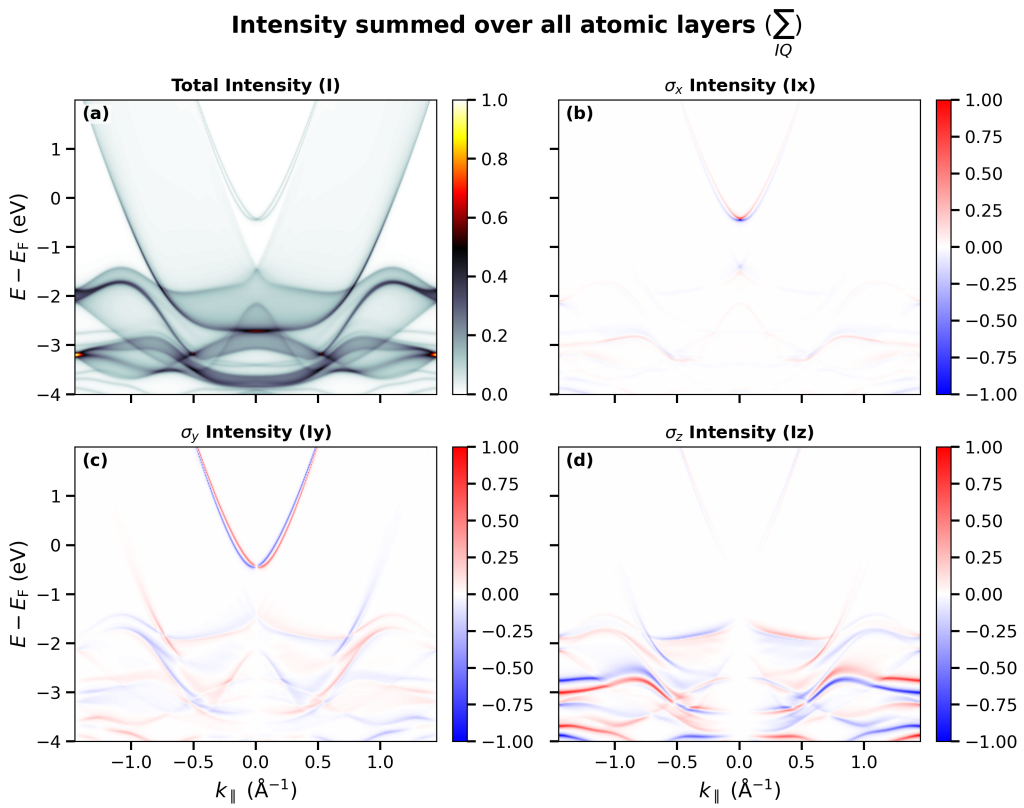


Figure 10: Spin-resolved Bloch spectral function $A(k_\parallel, E)$ for Au(111) centered at $\bar{\Gamma}$ point for Au(111), with intensity summed over all atomic layers ($\sum_{IQ}$), produced by the `bsfek` task. (a) Total intensity I, showing the bulk continuum and the Rashba-split surface state crossing E_F near $\bar{\Gamma}$. (b-d) The three Cartesian spin components σ_x, σ_y, σ_z (I_x, I_y, I_z), where red and blue denote opposite spin polarization.

```
# Select surface (IQ=11,12,13) and bulk-like (IQ=1,2,3) layers
bsf.plot(IQ=[1, 2, 3, 11, 12, 13])
```

The layer-resolved BSF obtained from this call is shown in Fig. 11. The same calculator instance switches to a Fermi-surface task by changing the `input_parameters` keyword, the energy window, and the k-grid:

```
BSF_input_fermi = {
    'TAU.NKTAB2D': 200,
    'ENERGY.EMINEV': 0.0,
    'ENERGY.EMAXEV': 0.0,
    'ENERGY.NE': 1,
    'TASK.KA': [-0.66667, -0.66667, 0.0], # corner of the k-grid
    'TASK.K1': [ 0.66667, -0.66667, 0.0], # end point of 1st edge
    'TASK.K2': [-0.66667,  0.66667, 0.0], # end point of 2nd edge
    'TASK.NK1': 200,
    'TASK.NK2': 200,
}
result = calc.calculate(
    potential='Au12X12.pot_new_new',
```

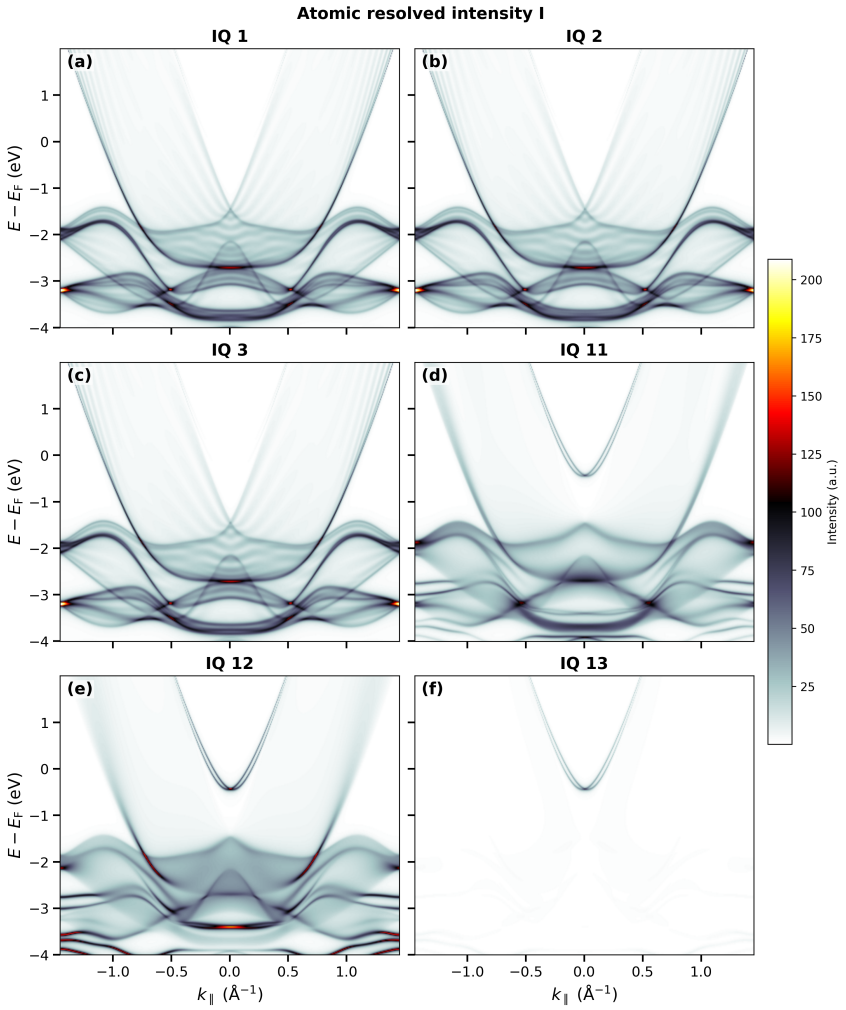


Figure 11: Layer-resolved spectral function of Au(111), with each panel showing the Bloch spectral function projected onto a single atomic layer (IQ). Deep bulk layers (IQ 1–3, panels a–c) are essentially indistinguishable, reproducing a uniform *sp*-band continuum and confirming convergence of the bulk electronic structure. The subsurface region (IQ 11, panel d) shows the emergence of layer-localized states as bulk weight is lost. The topmost layers (IQ 12–13, panels e–f) are dominated by the surface, with the Shockley surface state near $\bar{\Gamma}$ carrying most of its intensity in the outermost layer. Energy is referenced to the Fermi level E_F; intensity (color bar, right) is in arbitrary units.

```
    input_parameters='bsfkk',
    options=BSF_input_fermi
)
result.bsf.plot()
```

The resulting Fermi surface, mapped over the two-dimensional k-grid spanned from `TASK.KA` to the end points `TASK.K1` and `TASK.K2`, is shown in Fig. 12. Note that, unlike `TASK.K1/K2` here, `SPEC_EL.K1/K2` of the `arpes` task in Sec. 3.1 are edge vectors of the grid.

3.3 Exchange Coupling Parameters for Heusler Alloys

This example illustrates ASE2SPRKKR's multi-scale workflow capability: exchange parameters are extracted from the converged potential in a single additional `calculate()` call, and the

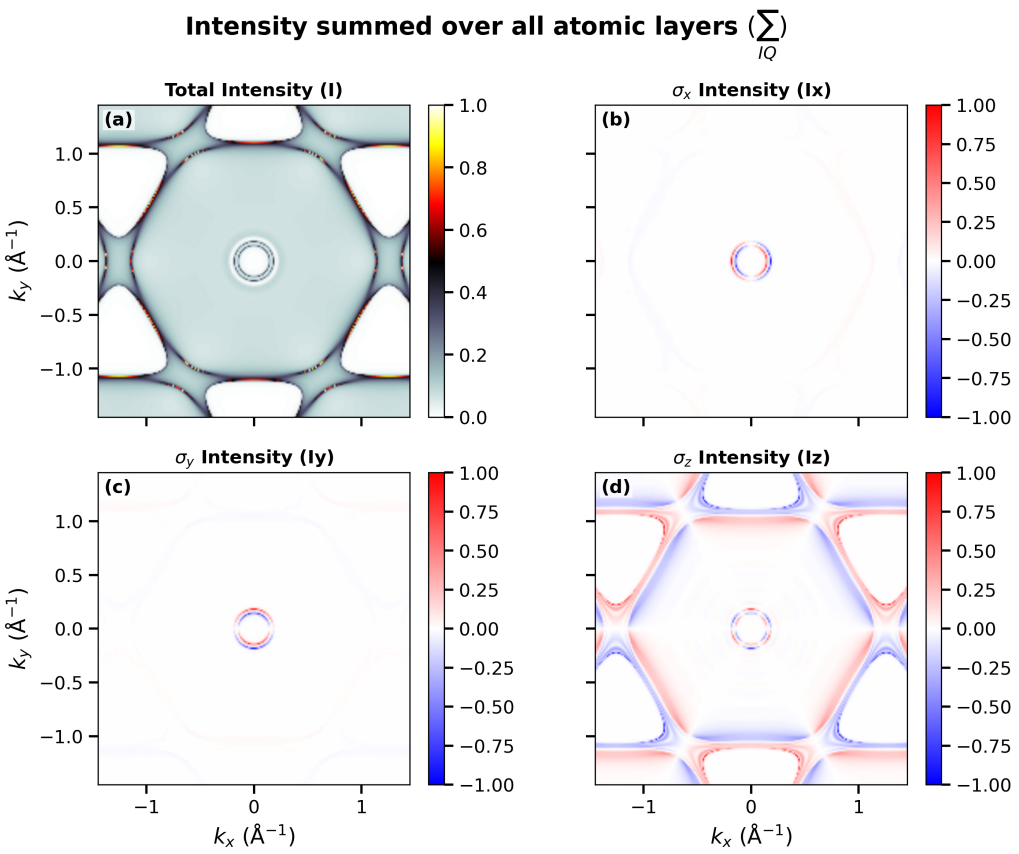


Figure 12: Spin-resolved Fermi surface $A(k_x, k_y, E_F)$ for Au(111), with intensity summed over all atomic layers ($\sum_{IQ}$), obtained from the `bsfkk` task. (a) Total intensity I, showing the bulk Fermi contours together with the concentric Rashba-split rings of the surface state at $\bar{\Gamma}$. (b-d) The three Cartesian spin components σ_x, σ_y, σ_z (I_x, I_y, I_z), where red and blue denote opposite spin polarization.

result is then converted to the input files of the UppASD atomistic spin-dynamics code, either from Python or with a built-in command-line utility, without manually editing intermediate files.

```
from ase2sprkkr import SPRKKR

calc = SPRKKR()
JXC_input = {
    'CONTROL.DATASET': 'Ni2FeGa',
    'TAU.NKTAB': 1000,
    'MODE.LLOYD': False,
    'TASK.CLURAD': 4.5,         # J_ij shell radius
    'SITES.NL': 4,
    'ENERGY.NE': 32,            # Energy integration points
}
result = calc.calculate(
    potential='Ni2FeGa.pot_new',
    input_parameters='jxc',
    options=JXC_input
)
```

```
result.jxc.plot(layout = (3,1))
print(result.mean_field_curie_temperature)
result.write_uppasd_files()
```

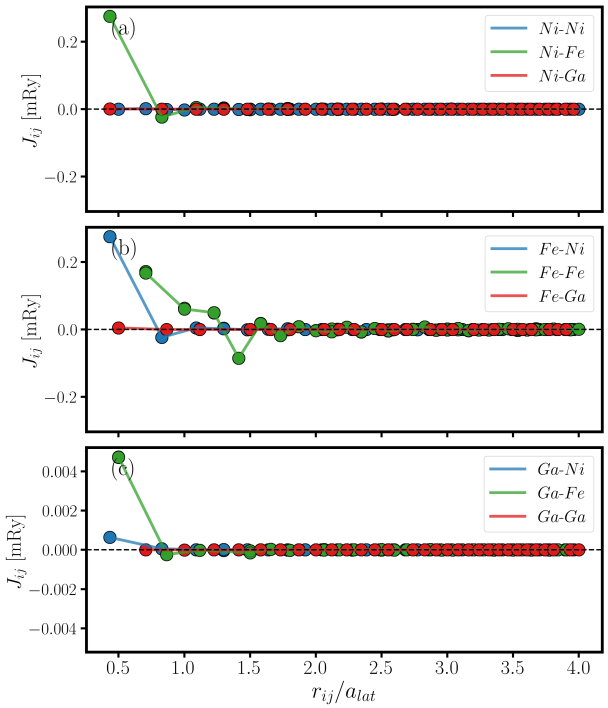


Figure 13: Exchange couplings $J_{ij}(R)$ for Ni_2FeGa produced by the `jxc` task: (a) Ni, (b) Fe, (c) Ga sublattice contributions as a function of inter-site distance. The structured result object provides per-sublattice access without post-processing of the raw output file.

The mean-field Curie temperature is computed by SPR-KKR from the exchange coupling parameters. For a single magnetic sublattice,

$$T_C^{\mathrm{MF}} = \frac{2}{3k_B} \sum_{j \neq i} J_{ij}, \tag{1}$$

consistent with the Hamiltonian $H = -\sum_{i \neq j} J_{ij}\, \hat{\mathbf{m}}_i \cdot \hat{\mathbf{m}}_j$ of the Appendix. The call to `result.jxc.plot(layout=(3,1))` produces the per-sublattice exchange-coupling plot shown in Fig. 13. The method `write_uppasd_files()` converts the result to UppASD input files (`jfile`, `momfile`, and `posfile`). The same can be achieved using the command-line utility:

```
ase2sprkkr uppasd Ni2FeGa_JXC.jxc
```

This bridges first-principles exchange parameters to atomistic spin dynamics without any manual format conversion, completing a multi-scale workflow that would otherwise require multiple specialized tools and bespoke scripts.

3.4 Empty Sphere Generation

Open covalent structures such as diamond-cubic silicon are poorly described within the Atomic Sphere Approximation (ASA) unless the large interstitial voids are explicitly represented.

Diamond-cubic Si (space group $Fd\bar{3}m$, No. 227) consists of an fcc Bravais lattice with a two-atom basis at $(0,0,0)$ and $(1/4,1/4,1/4)$, giving an atomic packing factor of only ≈ 0.34: roughly two-thirds of the primitive cell is interstitial space, making the ASA unreliable without basis completion.

ASE2SPRKKR automates this basis completion. Passing the `empty_spheres` parameter to `save_input()` or `calculate()` (or directly to the `SPRKKR` constructor) is sufficient; the interface identifies the symmetry-allowed interstitial sites, assigns sphere radii under the specified overlap constraints, and writes a complete potential file without user intervention.

```
from ase.build import bulk
from ase2sprkkr import SPRKKR
from ase.visualize import view

Si = bulk('Si', 'diamond', a=5.43)

calc = SPRKKR(atoms=Si)

calc.save_input(potential_file='Si.pot', empty_spheres={
        'max_radius': 2.0,      # sphere radius bounds (Angstrom)
        'min_radius': 0.2,
        'max_spheres': 300,
        'verbose': True })
```

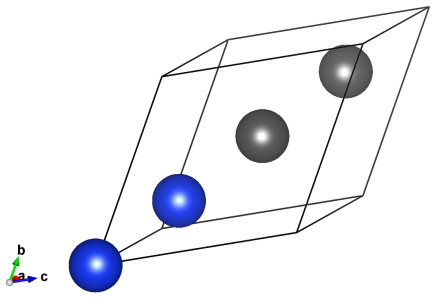


Figure 14: Diamond-cubic Si in the primitive fcc representation used by SPR-KKR. Blue spheres: Si atoms at $(0,0,0)$ and $(\frac{1}{4},\frac{1}{4},\frac{1}{4})$. Grey spheres: empty spheres placed automatically at the tetrahedral interstitial voids. The muffin-tin filling fraction increases from ≈ 0.34 (Si only) to ≈ 0.68 upon inclusion of the empty spheres.

For this setup ASE2SPRKKR augments the two-atom primitive cell with two empty spheres at the tetrahedral interstitial voids, as shown in Fig. 14, yielding a four-site basis (2 Si + 2 ES) with equal muffin-tin and Wigner-Seitz radii $R_{\mathrm{MT}} \approx 2.22$ a.u. and $R_{\mathrm{WS}} \approx 2.53$ a.u. The muffin-tin filling fraction increases from ≈ 0.34 (Si atoms alone) to ≈ 0.68 once the two empty spheres are included, halving the effective interstitial region. The Wigner-Seitz radii satisfy $\sum_i \frac{4\pi}{3} R_{\mathrm{WS},i}^3 = V_{\mathrm{cell}}$ by construction, ensuring the conservation of the volume of the unit cell. The more homogeneous real-space partitioning that results reduces the angular-momentum cutoff $\ell_{\max}$ needed for an accurate description of the interstitial charge density and mitigates the numerical artifacts that

arise in open tetrahedral networks when only atomic spheres are used. In many cases, just passing `True` to the `empty_spheres` parameter of the `calculate` or `save_input` methods of the `SPRKKR` calculator (the parameter can also be passed directly to the calculator, in which case it will be applied to all calls of the aforementioned methods) is sufficient. However, in some cases, better results can be achieved by fine-tuning the parameters, as illustrated in the example above. Manual addition of empty spheres using `ase2sprkkr.bindings.empty_spheres.add_empty_spheres` function is, of course, also possible.

3.5 X-ray Absorption and Magnetic Circular Dichroism

This example demonstrates element-selective spectroscopy: the `xas` task targets a specific atom type and core level via two keywords (`TASK.IT` and `TASK.CL`), while the interface handles energy-mesh generation, matrix-element integration, and output formatting. Post-processing via the XMCD sum rules requires only a few arithmetic operations on the structured result object.

```
from ase2sprkkr import SPRKKR

calc = SPRKKR()
XAS_input = {
    'CONTROL.DATASET': 'Ni2FeGa',
    'TAU.NKTAB': 500,
    'TASK.IT': 1,              # Target atom type: Ni
    'TASK.CL': '2p',           # Core level: 2p -> 3d
    'TASK.FRAMETET': 0,        # Polar angle
    'TASK.FRAMEPHI': 0,        # Azimuthal angle
    'ENERGY.NE': 180,
    'ENERGY.GRID': 6,
}
result = calc.calculate(
    input_parameters='xas',
    options=XAS_input,
    potential='Ni2FeGa.pot_new'
)
```

The resulting Ni $L_{2,3}$ XAS and XMCD spectra, obtained directly from the `xas` task, are shown in Fig. 15, where the interface manages the energy integration and returns both spectra in the result object.

Following the sum rules of Refs. [89, 90, 102], the orbital and spin moments are extracted from the integrated dichroic signal:

$$m_{\mathrm{orb}} = -\frac{4q}{3r}\, n_h\, \mu_B, \tag{2}$$

$$m_{\mathrm{spin}} = -\frac{6p-4q}{r}\, n_h \left(1 + \frac{7\langle T_z\rangle}{2\langle S_z\rangle}\right)^{-1} \mu_B, \tag{3}$$

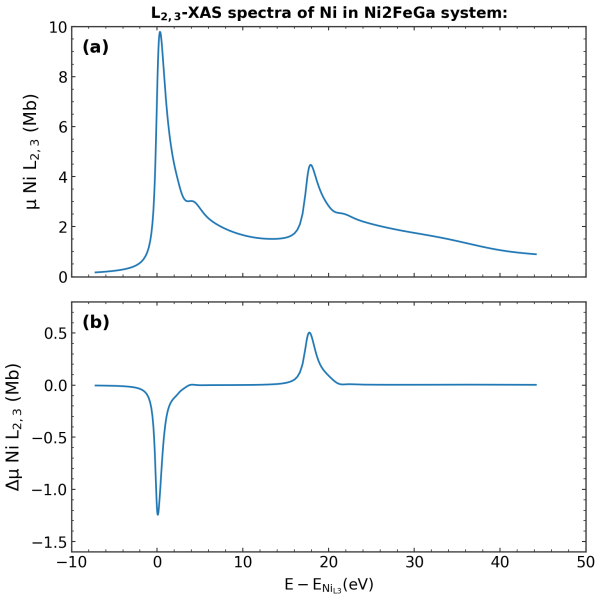


Figure 15: Ni $L_{2,3}$ XAS (a) and XMCD (b) for Ni_2FeGa obtained via the `xas` task. Targeting a specific site and edge requires only setting `TASK.IT` and `TASK.CL`; the interface manages energy integration and outputs both spectra in the result object.

where $p = \int_{L_3}(I^+ - I^-)\mathrm{d}E$, $q = \int_{L_3+L_2}(I^+ - I^-)\mathrm{d}E$, $r = \int_{L_3+L_2}(I^+ + I^-)\mathrm{d}E$, with $I^{\pm}$ the absorption spectra for the two circular polarizations and the linearly polarized spectrum replaced by $(I^+ + I^-)/2$. L_3 and L_2 denote the integration ranges, $\langle T_z \rangle$ is the expectation value of the magnetic dipole operator, $\langle S_z \rangle$ is equal to half of m_{spin} in Hartree atomic units, and $n_h = 10 - n_{3\mathrm{d}}$ is obtained directly from the self-consistent calculation.

4 Conclusion

We have presented ASE2SPRKKR, a Python interface that embeds the Spin-Polarized Relativistic Korringa-Kohn-Rostoker code within the Atomic Simulation Environment. The central design goal is to make SPR-KKR's unique capabilities accessible through the same scripting paradigm that ASE users already employ for other codes: structures are built with standard ASE tools, calculations are launched through a unified `SPRKKR` calculator, tasks are switched by changing a single keyword, and results are returned as structured objects with built-in plotting and export methods. The `SPRKKRAtoms` extension accommodates fractional site occupations for CPA calculations while retaining full ASE compatibility, and automated input validation, MPI support, and potential file management collectively eliminate the manual bookkeeping that has historically limited access to the code.

The five applications illustrate how this design reduces complex multi-step workflows to compact scripts. Semi-infinite surface geometries are built with `semiinfinite_system`, and the two-stage SCF requires only a modified options dictionary. Task switching between BSF, DOS, ARPES, JXC, and XAS calculations requires only a keyword change on the same calculator and potential. Multi-scale export to UppASD is a single method or command-line call. Automated empty-sphere placement extends the interface to open covalent structures without user intervention. In each

case the interface, not the underlying physics code, is what changes the user experience.

By grounding its architecture in the FAIR principles of Findability, Accessibility, Interoperability, and Reusability [103], ASE2SPRKKR establishes a replicable blueprint for bringing other specialized Green's function codes into reproducible, collaborative workflows. Future development will extend coverage to transport property calculations via the Kubo-Bastin formalism and provide high-level workflows for DLM and for correlated materials (LDA+U, DMFT), all within the same framework.

Acknowledgments

We thank the ASE development team for maintaining the open-source framework that underpins this work. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through e-INFRA CZ (ID: 90254).

Funding

R.E. and J.M. acknowledge funding from Horizon Europe MSCA Doctoral Network Grant No. 101073486, EUSpecLab, funded by the European Union. This work was supported by the project Quantum materials for applications in sustainable technologies (QM4ST), funded as Project No. CZ.02.01.01/00/22_008/0004572 by Programme Johannes Amos Comenius, call Excellent Research (R.E., J.M., A.P.). M.N. acknowledges funding from the Czech Science Foundation (GAČR), project No. 23-04746S until 2025, and project No. 25-16339S from 2026 onwards.

Code availability

ASE2SPRKKR is open-source software released under the MIT license. The source code is publicly available at `https://github.com/ase2sprkkr/ase2sprkkr`. The SPR-KKR multiple scattering package is freely available (no costs apply) under a specific user license, and it can be downloaded after registration at `https://sprkkr.org/software-request`. Further documentation is available at `https://ase2sprkkr.github.io/ase2sprkkr/`. We also encourage users to subscribe to our mailing list at `https://groups.google.com/g/ase2sprkkr`, where they can ask questions about the software and receive updates on new features and releases.

Data availability

All input scripts required to reproduce the five workflow demonstrations in Section 3 are included in the ASE2SPRKKR repository under `examples/`. The SPR-KKR potential files and raw output used to generate the figures are available from the corresponding author upon request.

Competing interests

The authors declare no competing interests.

Appendix. Theoretical Background: The KKR-Green's Function Method

This appendix provides essential background on the KKR-Green's function formalism implemented in SPR-KKR, focusing on aspects directly relevant to the ASE2SPRKKR interface design.

A.1 Fundamental Formulation

SPR-KKR solves the fully relativistic Dirac-Kohn-Sham equation [19, 52, 104]

$$\left[-\mathrm{i}\hbar c\,\boldsymbol{\alpha}\cdot\boldsymbol{\nabla} + \boldsymbol{\beta}mc^2 + V_{\mathrm{eff}}(\mathbf{r}) + \boldsymbol{\beta}\,\boldsymbol{\sigma}\cdot\mathbf{B}(\mathbf{r}) + \boldsymbol{\alpha}\cdot\mathbf{A}(\mathbf{r})\right]\psi(\mathbf{r}) = E\,\psi(\mathbf{r}), \tag{A.1}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are 4×4 Dirac matrices and ψ is a four-component spinor. The effective potential V_{eff} incorporates the Kohn-Sham potential and exchange-correlation contributions. The magnetic field $\boldsymbol{B}(\boldsymbol{r}) = \boldsymbol{B}_{\mathrm{ext}}(\boldsymbol{r}) + \delta E_{\mathrm{xc}}[n,\boldsymbol{m}]/\delta\boldsymbol{m}(\boldsymbol{r})$ combines any external field with the spin-dependent exchange-correlation field, while the vector potential $\boldsymbol{A}$ couples to the electronic current density $\boldsymbol{j} = -|e|c\,\boldsymbol{\alpha}$. In Eq. (A.1) the Bohr magneton and the electron charge are absorbed into $\boldsymbol{B}$ and $\boldsymbol{A}$, respectively.

The multiple scattering formulation [19, 24] partitions space into non-overlapping atomic regions. Single-site Dirac equations are solved for each atom type, yielding t-matrices $t^n_{\Lambda\Lambda'}(E)$ with combined quantum numbers $\Lambda = (\kappa,\mu)$, where κ and μ are the relativistic spin-orbit and magnetic quantum numbers, respectively. The Λ sums are truncated at a maximum angular momentum $\ell_{\max}$, which is an important convergence parameter exposed in ASE2SPRKKR as `SITES.NL` $= \ell_{\max}+1$ (see Section 3.2). The structural Green's function $G^0(E)$ describes free-electron propagation between sites and depends only on the geometry. These ingredients are combined through the scattering path operator:

$$\underline{\tau}^{nn'}(E) = \frac{1}{\Omega_{\mathrm{BZ}}}\int_{\Omega_{\mathrm{BZ}}} \mathrm{d}^3k\,\left[\underline{t}(E)^{-1} - \underline{G}^0(\boldsymbol{k},E)\right]^{-1} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n-\boldsymbol{R}_{n'})}, \tag{A.2}$$

from which the site-resolved Green's function follows:

$$\begin{aligned} G(\boldsymbol{r},\boldsymbol{r}',E) = &\sum_{\Lambda\Lambda'} Z^n_\Lambda(\boldsymbol{r},E)\,\tau^{nn'}_{\Lambda\Lambda'}(E)\,Z^{n'\times}_{\Lambda'}(\boldsymbol{r}',E) \\ &- \delta_{nn'}\sum_{\Lambda}\Big[Z^n_\Lambda(\boldsymbol{r},E)\,J^{n\times}_\Lambda(\boldsymbol{r}',E)\,\Theta(r'-r) \\ &+ J^n_\Lambda(\boldsymbol{r},E)\,Z^{n\times}_\Lambda(\boldsymbol{r}',E)\,\Theta(r-r')\Big], \end{aligned} \tag{A.3}$$

where $\boldsymbol{r}$ ($\boldsymbol{r}'$) are cell-centered coordinates in atomic cell n (n'), the superscript $\times$ denotes the left-hand-side solution, and Z^n_Λ, J^n_Λ are the regular and irregular single-site solutions.

With the Green's function available, all ground-state observables follow directly. The density of states and electron density are given by

$$n(E) = -\frac{1}{\pi}\,\mathrm{Im}\,\mathrm{Tr}\int_{\Omega_n}\mathrm{d}^3r\;G(\boldsymbol{r},\boldsymbol{r},E), \tag{A.4}$$

$$\rho(\boldsymbol{r}) = -\frac{1}{\pi}\,\mathrm{Im}\,\mathrm{Tr}\int^{E_\mathrm{F}} \mathrm{d}E\; G(\boldsymbol{r},\boldsymbol{r},E), \tag{A.5}$$

where the trace applies when G is given in matrix form. In practice, the energy integrals in Eqs. (A.5) and (A.12) are performed along a contour in the complex energy plane rather than on the real axis, exploiting the analyticity of G to achieve rapid convergence with far fewer energy points than real-axis integration would require. The contour parameters, start energy, number of points, and imaginary part, are exposed as convergence parameters in ASE2SPRKKR. Notably, ground-state quantities require only site-diagonal scattering path operators $\underline{\tau}^{nn}$.

For a $\boldsymbol{k}$-resolved representation of the electronic structure, the Bloch spectral function [21] is defined as

$$A_\mathrm{B}(\boldsymbol{k},E) = -\frac{1}{\pi N}\,\mathrm{Im}\,\mathrm{Tr}\sum_{n,n'}^{N} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot(\boldsymbol{R}_n-\boldsymbol{R}_{n'})}\int_\Omega \mathrm{d}^3r\; G(\boldsymbol{r}+\boldsymbol{R}_n,\boldsymbol{r}+\boldsymbol{R}_{n'},E). \tag{A.6}$$

For a perfectly ordered system, $A_\mathrm{B}(\boldsymbol{k},E)$ reduces to a sum of δ-functions $\delta(E-E_{\boldsymbol{k}})$, recovering the usual band structure. For disordered systems, it provides a broadened but physically meaningful spectral representation.

A.2 Coherent Potential Approximation

A key advantage of the KKR method is its natural connection to the coherent potential approximation (CPA) for substitutionally disordered alloys [14, 15, 26]. For a binary alloy with components A and B at concentrations x_A and x_B, the CPA effective medium is determined by the condition that embedding either component produces no additional scattering on average:

$$x_A\,\underline{\tau}_A^{nn} + x_B\,\underline{\tau}_B^{nn} = \underline{\tau}_\mathrm{CPA}^{nn}, \tag{A.7}$$

with the component-projected scattering path operator

$$\underline{\tau}_A^{nn} = \underline{\tau}_\mathrm{CPA}^{nn}\left[1+\left(t_A^{-1}-t_\mathrm{CPA}^{-1}\right)\tau_\mathrm{CPA}^{nn}\right]^{-1}. \tag{A.8}$$

The computational cost is independent of composition, making dilute impurities and equiatomic alloys equally tractable. SPR-KKR extends the CPA to multi-component alloys [21, 25] and non-collinear magnetic structures. In ASE2SPRKKR, disorder is specified through fractional site occupations in the `SPRKKRAtoms` structure, which automatically generates the appropriate CPA input.

The same CPA framework naturally accommodates the disordered local moment (DLM) approach [33, 34], which maps finite-temperature magnetic fluctuations onto orientational disorder. In the paramagnetic state, each site carries a local moment whose orientation is treated as a random variable; in its simplest, collinear form, "up" and "down" orientations enter the CPA with equal weights. This enables *ab initio* predictions of the paramagnetic electronic structure and of Curie temperatures without requiring supercells or empirical magnetic models.

A.3 Spectroscopic Observables

The Green's function formulation provides a unified framework for computing spectroscopic response functions, including transition matrix elements and final-state effects, which are essential for quantitative comparison with experiment.

A.3.1 Valence band photoemission

Angle-resolved photoemission spectroscopy (ARPES) [38–40] is described within the one-step model, which expresses the photocurrent as [41–44]

$$j_{km_s}^{q,\lambda}(E_\mathrm{f}) \propto \mathrm{Im} \int \mathrm{d}^3r \int \mathrm{d}^3r' \, \left[T\phi_{km_s}^{\mathrm{LEED}}(\boldsymbol{r}, E_\mathrm{f})\right]^\dagger X_{\boldsymbol{q}\lambda}(\boldsymbol{r})\, G(\boldsymbol{r}, \boldsymbol{r}', E_i)\, X_{\boldsymbol{q}\lambda}^\dagger(\boldsymbol{r}') \, \left[T\phi_{km_s}^{\mathrm{LEED}}(\boldsymbol{r}', E_\mathrm{f})\right], \tag{A.9}$$

where $G(E_i)$ represents the initial valence-band states, ϕ^{LEED} is the low-energy electron diffraction (LEED) state at $E_f = E_i + \hbar\omega$, whose time reverse $T\phi^{\mathrm{LEED}}$ is the photoemission final state, $X_{\boldsymbol{q}\lambda}$ is the electron-photon interaction operator, and T is the time-reversal operator. This formulation naturally incorporates matrix-element effects [45, 46], surface contributions, and finite-lifetime broadening through a self-energy $\Sigma(E)$, enabling quantitative modeling of spin-orbit-induced Rashba splittings at surfaces [105, 106].

A.3.2 Core level spectroscopies

X-ray absorption spectroscopy (XAS) probes element-specific electronic structure. Using the Green's function identity $-\pi^{-1}\,\mathrm{Im}\,G^+(E) = \sum_\alpha |\Psi_\alpha\rangle\langle\Psi_\alpha|\,\delta(E_\alpha - E)$, the absorption coefficient for radiation with wave vector $\boldsymbol{q}$ and polarization λ takes the form [49–51]:

$$\mu^{\boldsymbol{q}\lambda}(\omega) \propto -\sum_{i\,\mathrm{occ}} \langle\Phi_i| X_{\boldsymbol{q}\lambda}^\times \,\mathrm{Im}\, G^+(E_i + \hbar\omega)\, X_{\boldsymbol{q}\lambda} |\Phi_i\rangle \times \theta(E_i + \hbar\omega - E_\mathrm{F}), \tag{A.10}$$

with the operator $X_{\boldsymbol{q}\lambda} = -\frac{1}{c}\,\boldsymbol{j}_\mathrm{el} \cdot \boldsymbol{A}_{\boldsymbol{q}\lambda}$, where $\boldsymbol{j}_\mathrm{el} = -|e|c\,\boldsymbol{\alpha}$ already contains the electron charge. This expression covers both the XANES (0–50 eV above the edge) and EXAFS ($>$ 50 eV) regimes. For magnetically ordered systems with polarized X-rays, the formalism yields X-ray magnetic circular dichroism (XMCD) [48, 50], from which spin and orbital moments can be extracted via the XMCD sum rules [89, 90].

A.4 Exchange coupling constants

Magnetic exchange coupling constants J_{ij} between atomic sites i and j are obtained from the Lichtenstein formula [65], which maps the energy change upon relative rotation of magnetic moments onto a classical Heisenberg Hamiltonian $H = -\sum_{i\neq j} J_{ij}\,\hat{\mathbf{m}}_i \cdot \hat{\mathbf{m}}_j$, in which each pair is counted twice:

$$J_{ij} = -\frac{1}{4\pi}\,\mathrm{Im} \int^{E_\mathrm{F}} \mathrm{d}E \;\; \mathrm{Tr}\left[(t_i^{\uparrow -1} - t_i^{\downarrow -1})\,\tau_{ij}^{\uparrow}\,(t_j^{\uparrow -1} - t_j^{\downarrow -1})\,\tau_{ji}^{\downarrow}\right]. \tag{A.11}$$

The overall sign depends on the convention used for the Heisenberg Hamiltonian and on whether the scattering path operator or the T-matrix is written in terms of t or t^{-1}; with the convention above, $J_{ij} > 0$ favors ferromagnetic alignment. In the fully relativistic implementation of SPR-KKR, spin-orbit coupling is incorporated at all orders and the formula is generalized to treat non-collinear configurations and anisotropic exchange [66, 67]. These parameters serve as input for subsequent finite-temperature studies via mean-field theory, Monte Carlo simulations, or spin-dynamics approaches [72–75], enabling realistic modeling of Curie temperatures and spin-wave spectra.

A.5 Exchange-Correlation Treatment and Correlation Extensions

For systems requiring treatment of strong correlations beyond standard LSDA or GGA, SPR-KKR supports both the LDA+U and the dynamical mean-field theory (DMFT) frameworks [76–79, 86, 107]. Both approaches share a common starting point: the orbital-resolved density matrix $n^n_{\Lambda\Lambda'}$ constructed by projecting the Green's function onto a set of localized reference states $\Phi_\Lambda(\mathbf{r})$ of appropriate d- or f-character, determined at a reference energy E_Λ:

$$
\begin{aligned}
n^n_{\Lambda\Lambda'} = -\frac{1}{\pi}\,\mathrm{Im}\int^{E_F}\mathrm{d}E\,\Bigg\{ & \sum_{\Lambda''\Lambda'''}\int_{\Omega_n}\mathrm{d}^3r\;\Phi^\times_\Lambda(\mathbf{r})\,Z^n_{\Lambda''}(\mathbf{r},E)\,\tau^{nn}_{\Lambda''\Lambda'''}(E) \\
& \times\int_{\Omega_n}\mathrm{d}^3r'\;Z^{n\times}_{\Lambda'''}(\mathbf{r}',E)\,\Phi_{\Lambda'}(\mathbf{r}') \\
& -\sum_{\Lambda''}\int_{\Omega_n}\mathrm{d}^3r\int_{\Omega_n}\mathrm{d}^3r'\;\Phi^\times_\Lambda(\mathbf{r})\Big[Z^n_{\Lambda''}(\mathbf{r},E)\,J^{n\times}_{\Lambda''}(\mathbf{r}',E)\,\Theta(r'-r) \\
& +\;J^n_{\Lambda''}(\mathbf{r},E)\,Z^{n\times}_{\Lambda''}(\mathbf{r}',E)\,\Theta(r-r')\Big]\,\Phi_{\Lambda'}(\mathbf{r}')\Bigg\},
\end{aligned}
\tag{A.12}
$$

where the first term accumulates contributions from all energies up to E_F through the site-diagonal scattering path operator $\tau^{nn}_{\Lambda''\Lambda'''}(E)$ and the regular solutions Z^n_Λ, while the second, single-site term is built from the regular and irregular solutions Z^n_Λ and J^n_Λ and ensures the proper normalization within the atomic cell Ω_n [19]. Owing to the radial ordering $\Theta(r'-r)$, $\Theta(r-r')$ inherited from Eq. (A.3), the single-site term does not factorize into independent integrals over $\mathbf{r}$ and $\mathbf{r}'$, and, like the first term, it is integrated over energy.

Both LDA+U and DMFT introduce a self-energy $\Sigma_{\Lambda\Lambda'}$ acting within the correlated subspace spanned by $\{\Phi_\Lambda\}$, and modify the total energy and Green's function accordingly. They differ in whether this self-energy is static or frequency-dependent.

A.5.1 LDA+U

Transforming $n^n_{\Lambda\Lambda'}$ from the relativistic $\Lambda=(\kappa,\mu)$ representation to the $L=(l,m,\sigma)$ representation, the LDA+U total energy functional reads [86, 107]:

$$
E^{\mathrm{DFT}+U}(\rho,n) = E^{\mathrm{DFT}}(\rho) + E^U(n) - E^{\mathrm{dc}}(n),
\tag{A.13}
$$

where $E^{\mathrm{DFT}}(\rho)$ is the LSDA or GGA functional in terms of the spin densities $\rho^\uparrow$ and $\rho^\downarrow$, $E^U(n)$ is the electron-electron interaction energy of the localized electrons, and $E^{\mathrm{dc}}(n)$ is the double-counting correction subtracting the Coulomb interaction already contained in $E^{\mathrm{DFT}}(\rho)$. The double-counting term is not uniquely defined; SPR-KKR supports both the fully localized limit and the around-mean-field schemes [107]. The corresponding static self-energy in the Λ-subspace is

$$
\Sigma^U_{\Lambda\Lambda'} = \frac{\partial E^U(n)}{\partial n^n_{\Lambda'\Lambda}} - \frac{\partial E^{\mathrm{dc}}(n)}{\partial n^n_{\Lambda'\Lambda}},
\tag{A.14}
$$

which enters the Green's function as an energy-independent shift. This reduces over-delocalization and improves the description of Mott insulators such as NiO and CoO [80], but cannot capture dynamical effects such as Kondo resonances or Hubbard satellites.

A.5.2 DMFT

For systems requiring dynamical correlations, heavy-fermion compounds, Kondo systems, and materials near the Mott transition, the static self-energy $\Sigma^U_{\Lambda\Lambda'}$ is replaced by a frequency-dependent counterpart $\Sigma_{\Lambda\Lambda'}(E)$ obtained from a local quantum impurity problem [76]. The impurity is defined by the same correlated subspace $\{\Phi_\Lambda\}$ and density matrix $n^n_{\Lambda\Lambda'}$ as in LDA+U, ensuring a consistent projection throughout. The self-energy is projected back onto the KKR basis as

$$\Sigma(E) = \sum_{\Lambda\Lambda'} \Phi_\Lambda \, \Sigma_{\Lambda\Lambda'}(E) \, \Phi^\times_{\Lambda'}, \tag{A.15}$$

where $\Phi^\times_\Lambda$ is the left-hand solution consistent with the KKR biorthogonality convention of Eq. (A.3), and modifies the lattice Green's function through

$$[E - H - \Sigma(E)]\, G(E) = \mathbf{1}. \tag{A.16}$$

The frequency dependence of $\Sigma_{\Lambda\Lambda'}(E)$ encodes Kondo resonances, Hubbard satellites, and quasiparticle renormalization inaccessible to LDA+U [76, 77]. Charge self-consistency between the KKR host and the DMFT impurity solver [78, 79] ensures proper feedback between local correlations and itinerant band states, essential for quantitative spectroscopic predictions in strongly correlated systems [80, 81].

A.6 Implementation Aspects Relevant for the Interface

SPR-KKR supports two potential approximations: the atomic sphere approximation (ASA) [108], which assumes spherical potentials within touching spheres, and the full-potential mode [109], which expands $V_i(\boldsymbol{r}) = \sum_{\ell m} V_{i\ell m}(r)\, Y_{\ell m}(\hat{\boldsymbol{r}})$ at greater computational cost. ASA is efficient for close-packed structures, while the full-potential mode is necessary for open structures and directional bonding. For such open structures, empty spheres with zero nuclear charge can be inserted to improve basis completeness; ASE2SPRKKR provides automated empty-sphere generation.

Semi-infinite geometries for surface calculations [88, 110] partition the system into a periodic bulk region and an interaction zone extending to vacuum. The screened (tight-binding) KKR variant [20, 87] truncates real-space sums beyond a cluster radius `CLURAD`, which is essential for large surface unit cells and reduces the $\mathcal{O}(N^3)$ matrix inversion cost significantly.

Task-specific calculations (SCF, DOS, BSF, ARPES, J_{ij}, XAS, and others) require distinct parameter sets organized hierarchically. ASE2SPRKKR exposes this structure through validated `InputParameters` objects while providing high-level interfaces for common workflows.

References

[1] P. Hohenberg, W. Kohn, Inhomogeneous electron gas, Physical Review 136 (1964) B864–B871. doi:10.1103/PhysRev.136.B864.

[2] W. Kohn, L. J. Sham, Self-consistent equations including exchange and correlation effects, Physical Review 140 (1965) A1133–A1138. doi:10.1103/PhysRev.140.A1133.

[3] A. H. Larsen, et al., The atomic simulation environment—a Python library for working with atoms, Journal of Physics: Condensed Matter 29 (2017) 273002. doi:10.1088/1361-648X/aa680e.

[4] S. R. Bahn, K. W. Jacobsen, An object-oriented scripting interface to a legacy electronic structure code, Comput. Sci. Eng. 4 (2002) 56–66. doi:10.1109/5992.998641.

[5] G. Kresse, J. Furthmüller, Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (1996) 11169. doi:10.1103/PhysRevB.54.11169.

[6] P. Giannozzi, et al., QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials, J. Phys.: Condens. Matter 21 (2009) 395502. doi:10.1088/0953-8984/21/39/395502.

[7] X. Gonze, et al., The Abinit project: Impact, environment and recent developments, Comput. Phys. Commun. 248 (2020) 107042. doi:10.1016/j.cpc.2019.107042.

[8] J. M. Soler, et al., The SIESTA method for *ab initio* order-n materials simulation, J. Phys.: Condens. Matter 14 (2002) 2745. doi:10.1088/0953-8984/14/11/302.

[9] T. D. Kühne, et al., CP2K: An electronic structure and molecular dynamics software package, J. Chem. Phys. 152 (2020) 194103. doi:10.1063/5.0007045.

[10] V. Blum, et al., *Ab initio* molecular simulations with numeric atom-centered orbitals, Comput. Phys. Commun. 180 (2009) 2175–2196. doi:10.1016/j.cpc.2009.06.022.

[11] P. Blaha, et al., WIEN2k: An apw+lo program for calculating the properties of solids, J. Chem. Phys. 152 (2020) 074101. doi:10.1063/1.5143061.

[12] D. Wortmann, et al., FLEUR, Zenodo (May 2023). doi:10.5281/zenodo.7576163.
URL https://doi.org/10.5281/zenodo.7576163

[13] J. Enkovaara, et al., Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method, Journal of Physics: Condensed Matter 22 (2010) 253202. doi:10.1088/0953-8984/22/25/253202.

[14] P. Soven, Coherent-potential model of substitutional disordered alloys, Physical Review 156 (1967) 809–813. doi:10.1103/PhysRev.156.809.

[15] B. L. Györffy, Coherent-potential approximation for a nonoverlapping-muffin-tin-potential model of random substitutional alloys, Physical Review B 5 (1972) 2382–2384. doi:10.1103/PhysRevB.5.2382.

[16] A. Zunger, S.-H. Wei, L. G. Ferreira, J. E. Bernard, Special quasirandom structures, Phys. Rev. Lett. 65 (1990) 353. doi:10.1103/PhysRevLett.65.353.

[17] E. Runge, E. K. U. Gross, Density-functional theory for time-dependent systems, Phys. Rev. Lett. 52 (1984) 997. doi:10.1103/PhysRevLett.52.997.

[18] G. Onida, L. Reining, A. Rubio, Electronic excitations: density-functional versus many-body Green's-function approaches, Rev. Mod. Phys. 74 (2002) 601. doi:10.1103/RevModPhys.74.601.

[19] H. Ebert, D. Ködderitzsch, J. Minár, Calculating condensed matter properties using the KKR-Green's function method—recent developments and applications, Reports on Progress in Physics 74 (2011) 096501. doi:10.1088/0034-4885/74/9/096501.

[20] R. Zeller, P. H. Dederichs, B. Újfalussy, L. Szunyogh, P. Weinberger, Theory and convergence properties of the screened Korringa-Kohn-Rostoker method, Physical Review B 52 (1995) 8807–8812. doi:10.1103/PhysRevB.52.8807.

[21] J. S. Faulkner, G. M. Stocks, Calculating properties with the coherent-potential approximation, Physical Review B 21 (1980) 3222–3244. doi:10.1103/PhysRevB.21.3222.

[22] J. Korringa, On the calculation of the energy of a bloch wave in a metal, Physica 13 (1947) 392–400. doi:10.1016/0031-8914(47)90013-X.

[23] W. Kohn, N. Rostoker, Solution of the Schrödinger equation in periodic lattices with an application to metallic lithium, Physical Review 94 (1954) 1111–1120. doi:10.1103/PhysRev.94.1111.

[24] A. Gonis, W. H. Butler, Multiple Scattering in Solids, Springer, New York, 2000. doi:10.1007/978-1-4612-1290-4.

[25] I. Turek, V. Drchal, J. Kudrnovský, M. Šob, P. Weinberger, Electronic Structure of Disordered Alloys, Surfaces and Interfaces, Kluwer Academic, Boston, 1997. doi:10.1007/978-1-4615-6255-9.

[26] D. W. Taylor, Vibrational properties of imperfect crystals with large defect concentrations, Phys. Rev. 156 (1967) 1017. doi:10.1103/PhysRev.156.1017.

[27] R. Magri, S. Froyen, A. Zunger, Electronic structure and density of states of the random $Al_{0.5}Ga_{0.5}As$, $GaAs_{0.5}P_{0.5}$, and $Ga_{0.5}In_{0.5}As$ semiconductor alloys, Phys. Rev. B 44 (1991) 7947. doi:10.1103/PhysRevB.44.7947.

[28] A. V. Ruban, I. A. Abrikosov, Configurational thermodynamics of alloys from first principles: effective cluster interactions, Rep. Prog. Phys. 71 (2008) 046501. doi:10.1088/0034-4885/71/4/046501.

[29] I. A. Abrikosov, A. V. Ponomareva, P. Steneteg, S. A. Barannikova, B. Alling, Recent progress in simulations of the paramagnetic state of magnetic materials, Curr. Opin. Solid State Mater. Sci. 20 (2016) 85–106. doi:10.1016/j.cossms.2015.07.003.

[30] B. Alling, A. V. Ruban, A. Karimi, O. E. Peil, S. I. Simak, L. Vitos, I. A. Abrikosov, Mixing and decomposition thermodynamics of $Ti_{1-x}Al_xN$ from first-principles calculations, Phys. Rev. B 75 (2007) 045123. doi:10.1103/PhysRevB.75.045123.

[31] D. A. Rowlands, J. B. Staunton, B. L. Györffy, KKR nonlocal coherent-potential approximation, Physical Review B 67 (2003) 115109. doi:10.1103/PhysRevB.67.115109.

[32] M. Jarrell, H. R. Krishnamurthy, Systematic and causal corrections to the coherent potential approximation, Phys. Rev. B 63 (2001) 125102. doi:10.1103/PhysRevB.63.125102.

[33] B. L. Györffy, A. J. Pindor, J. Staunton, G. M. Stocks, H. Winter, A first-principles theory of ferromagnetic phase transitions in metals, Journal of Physics F: Metal Physics 15 (1985) 1337–1386. doi:10.1088/0305-4608/15/6/018.

[34] J. B. Staunton, R. Banerjee, M. dos Santos Dias, A. Deak, L. Szunyogh, Fluctuating local moments, itinerant electrons, and the magnetocaloric effect: Compositional hypersensitivity of FeRh, Physical Review B 89 (2014) 054427. doi:10.1103/PhysRevB.89.054427.

[35] H. Ebert, S. Mankovsky, K. Chadova, S. Polesya, J. Minár, D. Ködderitzsch, Calculating linear-response functions for finite temperatures on the basis of the alloy analogy model, Physical Review B 91 (2015) 165132. doi:10.1103/PhysRevB.91.165132.

[36] E. N. Economou, Green's Functions in Quantum Physics, Springer, Berlin, 1983. doi:10.1007/978-3-662-02369-3.

[37] A. L. Fetter, J. D. Walecka, Quantum Theory of Many-Particle Systems, McGraw-Hill, New York, 1971.

[38] S. Hüfner, Photoelectron Spectroscopy: Principles and Applications, 3rd Edition, Springer, Berlin, 2003. doi:10.1007/978-3-662-09280-4.

[39] A. Damascelli, Z. Hussain, Z.-X. Shen, Angle-resolved photoemission studies of the cuprate superconductors, Rev. Mod. Phys. 75 (2003) 473. doi:10.1103/RevModPhys.75.473.

[40] B. Q. Lv, T. Qian, H. Ding, Angle-resolved photoemission spectroscopy and its application to topological materials, Nat. Rev. Phys. 1 (2019) 609–626. doi:10.1038/s42254-019-0088-5.

[41] J. B. Pendry, The theory of photoemission, Surface Science 57 (1976) 679–705. doi:10.1016/0039-6028(76)90355-1.

[42] J. Braun, The theory of angle-resolved ultraviolet photoemission and its applications to ordered materials, Reports on Progress in Physics 59 (1996) 1267–1338. doi:10.1088/0034-4885/59/10/002.

[43] J. Braun, J. Minár, H. Ebert, M. I. Katsnelson, A. I. Lichtenstein, Spectral function of ferromagnetic $3d$ metals: A self-consistent LSDA + DMFT approach combined with the one-step model of photoemission, Phys. Rev. Lett. 97 (2006) 227601. doi:10.1103/PhysRevLett.97.227601.

[44] J. Minár, S. Mankovsky, J. Braun, H. Ebert, One-step model of photoemission at finite temperatures: Spin fluctuations of Fe(001), Phys. Rev. B 102 (2020) 035107. doi:10.1103/PhysRevB.102.035107.

[45] C. S. Fadley, Atomic-level characterization of materials with core- and valence-level photoemission: basic phenomena and future directions, Surf. Interface Anal. 40 (2008) 1579–1605. doi:10.1002/sia.2902.

[46] W. Schattke, M. A. Van Hove, Solid-State Photoemission and Related Methods, Wiley-VCH, Weinheim, 2003. doi:10.1002/9783527602506.

[47] F. Reinert, S. Hüfner, Photoemission spectroscopy—from early days to recent applications, New J. Phys. 7 (2005) 97. doi:10.1088/1367-2630/7/1/097.

[48] J. Stöhr, Exploring the microscopic origin of magnetic anisotropy with X-ray magnetic circular dichroism (XMCD) spectroscopy, J. Magn. Magn. Mater. 200 (1999) 470–497. doi:10.1016/S0304-8853(99)00407-2.

[49] A. L. Ankudinov, B. Ravel, J. J. Rehr, S. D. Conradson, Real-space multiple-scattering calculation and interpretation of x-ray-absorption near-edge structure, Physical Review B 58 (1998) 7565–7576. doi:10.1103/PhysRevB.58.7565.

[50] H. Ebert, Magneto-optical effects in transition metal systems, Reports on Progress in Physics 59 (1996) 1665–1735. doi:10.1088/0034-4885/59/12/003.

[51] E. Arola, P. Strange, B. L. Gyorffy, Relativistic theory of magnetic scattering of x rays: Application to ferromagnetic iron, Phys. Rev. B 55 (1997) 472–484. doi:10.1103/PhysRevB.55.472.

[52] M. E. Rose, Relativistic Electron Theory, Wiley, New York, 1961.

[53] P. Strange, Relativistic Quantum Mechanics, Cambridge University Press, Cambridge, 1998. doi:10.1017/CBO9780511622755.

[54] R. Kubo, Statistical-mechanical theory of irreversible processes. I. general theory and simple applications to magnetic and conduction problems, J. Phys. Soc. Jpn. 12 (1957) 570–586. doi:10.1143/JPSJ.12.570.

[55] A. Bastin, C. Lewiner, O. Betbeder-Matibet, P. Nozieres, Quantum oscillations of the hall effect of a fermion gas with random impurity scattering, J. Phys. Chem. Solids 32 (1971) 1811–1823. doi:10.1016/S0022-3697(71)80147-6.

[56] A. Crépieux, P. Bruno, Theory of the anomalous hall effect in ferromagnetic metals with strong spin-orbit coupling, Phys. Rev. B 64 (2001) 014416. doi:10.1103/PhysRevB.64.014416.

[57] W. H. Butler, Theory of electronic transport in random alloys: Korringa-Kohn-Rostoker coherent-potential approximation, Physical Review B 31 (1985) 3260–3277. doi:10.1103/PhysRevB.31.3260.

[58] P. Středa, Theory of quantised Hall conductivity in two dimensions, Journal of Physics C 15 (1982) L717–L721. doi:10.1088/0022-3719/15/22/005.

[59] S. Lowitzer, D. Ködderitzsch, H. Ebert, Coherent description of the intrinsic and extrinsic anomalous Hall effect in disordered alloys on an ab initio level, Physical Review Letters 105 (2010) 266604. doi:10.1103/PhysRevLett.105.266604.

[60] S. Lowitzer, M. Gradhand, D. Ködderitzsch, D. V. Fedorov, I. Mertig, H. Ebert, Extrinsic and intrinsic contributions to the spin Hall effect of alloys, Physical Review Letters 106 (2011) 056601. doi:10.1103/PhysRevLett.106.056601.

[61] N. Nagaosa, J. Sinova, S. Onoda, A. H. MacDonald, N. P. Ong, Anomalous Hall effect, Rev. Mod. Phys. 82 (2010) 1539. doi:10.1103/RevModPhys.82.1539.

[62] R. Landauer, Spatial variation of currents and fields due to localized scatterers in metallic conduction, IBM Journal of Research and Development 1 (1957) 223–231. doi:10.1147/rd.13.0223.

[63] M. Büttiker, Four-terminal phase-coherent conductance, Physical Review Letters 57 (1986) 1761–1764. doi:10.1103/PhysRevLett.57.1761.

[64] S. Datta, Electronic Transport in Mesoscopic Systems, Cambridge University Press, Cambridge, 1995. doi:10.1017/CBO9780511805776.

[65] A. I. Liechtenstein, M. I. Katsnelson, V. P. Antropov, V. A. Gubanov, Local spin density functional approach to the theory of exchange interactions in ferromagnetic metals and alloys, Journal of Magnetism and Magnetic Materials 67 (1987) 65–74. doi:10.1016/0304-8853(87)90721-9.

[66] L. Udvardi, L. Szunyogh, K. Palotás, P. Weinberger, First-principles relativistic study of spin waves in thin magnetic films, Phys. Rev. B 68 (2003) 104436. doi:10.1103/PhysRevB.68.104436.

[67] H. Ebert, S. Mankovsky, Anisotropic exchange coupling in diluted magnetic semiconductors: *Ab initio* spin-density functional theory, Phys. Rev. B 79 (2009) 045209. doi:10.1103/PhysRevB.79.045209.

[68] T. L. Gilbert, A phenomenological theory of damping in ferromagnetic materials, IEEE Trans. Magn. 40 (2004) 3443–3449. doi:10.1109/TMAG.2004.836740.

[69] A. Brataas, Y. Tserkovnyak, G. E. W. Bauer, Scattering theory of Gilbert damping, Phys. Rev. Lett. 101 (2008) 037207. doi:10.1103/PhysRevLett.101.037207.

[70] H. Ebert, S. Mankovsky, D. Ködderitzsch, P. J. Kelly, *Ab initio* calculation of the Gilbert damping parameter via the linear response formalism, Physical Review Letters 107 (2011) 066603. doi:10.1103/PhysRevLett.107.066603.

[71] S. Mankovsky, D. Ködderitzsch, G. Woltersdorf, H. Ebert, First-principles calculation of the Gilbert damping parameter via the linear response formalism with application to magnetic transition metals and alloys, Physical Review B 87 (2013) 014430. doi:10.1103/PhysRevB.87.014430.

[72] B. Skubic, J. Hellsvik, L. Nordström, O. Eriksson, A method for atomistic spin dynamics simulations: implementation and examples, Journal of Physics: Condensed Matter 20 (2008) 315203. doi:10.1088/0953-8984/20/31/315203.

[73] R. F. L. Evans, W. J. Fan, P. Chureemart, T. A. Ostler, M. O. A. Ellis, R. W. Chantrell, Atomistic spin model simulations of magnetic nanomaterials, J. Phys.: Condens. Matter 26 (2014) 103202. doi:10.1088/0953-8984/26/10/103202.

[74] O. Eriksson, A. Bergman, L. Bergqvist, J. Hellsvik, Atomistic Spin Dynamics: Foundations and Applications, Oxford University Press, Oxford, 2017. doi:10.1093/oso/9780198788669.001.0001.

[75] A. Bergman, A. Taroni, L. Bergqvist, J. Hellsvik, B. Hjörvarsson, O. Eriksson, Magnon softening in a ferromagnetic monolayer: A first-principles spin dynamics study, Phys. Rev. B 81 (2010) 144416. doi:10.1103/PhysRevB.81.144416.

[76] A. Georges, G. Kotliar, W. Krauth, M. J. Rozenberg, Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions, Reviews of Modern Physics 68 (1996) 13–125. doi:10.1103/RevModPhys.68.13.

[77] G. Kotliar, S. Y. Savrasov, K. Haule, V. S. Oudovenko, O. Parcollet, C. A. Marianetti, Electronic structure calculations with dynamical mean-field theory, Reviews of Modern Physics 78 (2006) 865–951. doi:10.1103/RevModPhys.78.865.

[78] J. Minár, L. Chioncel, A. Perlov, H. Ebert, M. I. Katsnelson, A. I. Lichtenstein, Multiple-scattering formalism for correlated systems: A KKR-DMFT approach, Physical Review B 72 (2005) 045125. doi:10.1103/PhysRevB.72.045125.

[79] J. Minár, Correlation effects in transition metals and their alloys studied using the fully self-consistent kkr-based lsda+ dmft scheme, Journal of Physics: Condensed Matter 23 (25) (2011) 253201. doi:10.1088/0953-8984/23/25/253201.

[80] M. Imada, A. Fujimori, Y. Tokura, Metal-insulator transitions, Rev. Mod. Phys. 70 (1998) 1039. doi:10.1103/RevModPhys.70.1039.

[81] K. Held, Electronic structure calculations using dynamical mean field theory, Adv. Phys. 56 (2007) 829–926. doi:10.1080/00018730701619647.

[82] H. Ebert, Fully relativistic band structure calculations for magnetic solids, in: H. Dreyssé (Ed.), Electronic Structure and Physical Properties of Solids, Vol. 535 of Lecture Notes in Physics, Springer, Berlin, 2000, pp. 191–246. doi:10.1007/3-540-46437-9_6.

[83] H. Ebert, J. Minar, et al., The SPR-KKR package, version 9.7 (2026).
URL https://sprkkr.org/

[84] J. Zabloudil, R. Hammerling, P. Weinberger, L. Szunyogh, Electron Scattering in Solid Matter: a theoretical and computational treatise, Springer, 2005.

[85] P. Rüßmann, F. Bertoldo, S. Blügel, The AiiDA-KKR plugin and its application to high-throughput impurity embedding into a topological insulator, npj Computational Materials 7 (2021) 13. doi:10.1038/s41524-020-00482-5.

[86] A. I. Liechtenstein, V. I. Anisimov, J. Zaanen, Density-functional theory and strong interactions: orbital ordering in Mott-Hubbard insulators, Physical Review B 52 (1995) R5467–R5470. doi:10.1103/PhysRevB.52.R5467.

[87] L. Szunyogh, B. Újfalussy, P. Weinberger, J. Kollár, Self-consistent localized KKR scheme for surfaces and interfaces, Physical Review B 49 (1994) 2721–2729. doi:10.1103/PhysRevB.49.2721.

[88] J. M. MacLaren, S. Crampin, D. D. Vvedensky, J. B. Pendry, Layer Korringa-Kohn-Rostoker technique for surface and interface electronic properties, Physical Review B 40 (1989) 12164–12175. doi:10.1103/PhysRevB.40.12164.

[89] B. T. Thole, P. Carra, F. Sette, G. van der Laan, X-ray circular dichroism as a probe of orbital magnetization, Physical Review Letters 68 (1992) 1943–1946. doi:10.1103/PhysRevLett.68.1943.

[90] P. Carra, B. T. Thole, M. Altarelli, X. Wang, X-ray circular dichroism and local magnetic fields, Physical Review Letters 70 (1993) 694–697. doi:10.1103/PhysRevLett.70.694.

[91] G. Van Rossum, F. L. Drake, The Python Language Reference, Python Software Foundation, 2003.

[92] C. R. Harris, et al., Array programming with NumPy, Nature 585 (2020) 357–362. doi:10.1038/s41586-020-2649-2.

[93] P. Virtanen, et al., SciPy 1.0: fundamental algorithms for scientific computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[94] S. Behnel, et al., Cython: The best of both worlds, Computing in Science & Engineering 13 (2) (2011) 31–39. doi:10.1109/MCSE.2010.118.

[95] Anaconda, Inc., Anaconda distribution (2025).
URL https://www.anaconda.com/products/distribution

[96] A. Togo, K. Shinohara, I. Tanaka, Spglib: a software library for crystal symmetry search, Science and Technology of Advanced Materials: Methods 4 (1) (2024) 2384822. doi:10.1080/27660400.2024.2384822.

[97] M. Scheidgen, et al., NOMAD: A distributed web-based platform for managing materials science research data, Journal of Open Source Software 8 (2023) 5388. doi:10.21105/joss.05388.

[98] P. McGuire, Getting started with PyParsing (2007).
URL https://pyparsing-docs.readthedocs.io/

[99] MPI Forum, MPI: A message passing interface standard, version 1.0 (1994).
URL https://www.mpi-forum.org

[100] A. B. Yoo, M. A. Jette, M. Grondona, SLURM: Simple Linux utility for resource management, in: Job Scheduling Strategies for Parallel Processing (JSSPP 2003), Vol. 2862 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2003, pp. 44–60. doi:10.1007/10968987_3.

[101] SchedMD, Slurm Workload Manager Documentation, Slurm Workload Manager Project (2025).
URL https://slurm.schedmd.com/documentation.html

[102] C. T. Chen, Y. U. Idzerda, H.-J. Lin, N. V. Smith, G. Meigs, E. Chaban, G. H. Ho, E. Pellegrin, F. Sette, Experimental confirmation of the X-ray magnetic circular dichroism sum rules for iron and cobalt, Physical Review Letters 75 (1995) 152–155. doi:10.1103/PhysRevLett.75.152.

[103] M. D. Wilkinson, et al., The FAIR guiding principles for scientific data management and stewardship, Scientific Data 3 (2016) 160018. doi:10.1038/sdata.2016.18.

[104] A. H. MacDonald, S. H. Vosko, A relativistic density functional formalism, Journal of Physics C: Solid State Physics 12 (1979) 2977–2990. doi:10.1088/0022-3719/12/15/007.

[105] S. LaShell, B. A. McDougall, E. Jensen, Spin splitting of an Au(111) surface state band observed with angle resolved photoelectron spectroscopy, Physical Review Letters 77 (1996) 3419–3422. doi:10.1103/PhysRevLett.77.3419.

[106] E. I. Rashba, Properties of semiconductors with an extremum loop. I. Cyclotron and combinational resonance in a magnetic field perpendicular to the plane of the loop, Soviet Physics Solid State 2 (1960) 1109–1122.

[107] V. I. Anisimov, F. Aryasetiawan, A. I. Lichtenstein, First-principles calculations of the electronic structure and spectra of strongly correlated systems: the LDA+U method, Journal of Physics: Condensed Matter 9 (1997) 767–808. doi:10.1088/0953-8984/9/4/002.

[108] O. K. Andersen, Linear methods in band theory, Physical Review B 12 (1975) 3060–3083. doi:10.1103/PhysRevB.12.3060.

[109] B. Drittler, M. Weinert, R. Zeller, P. Dederichs, Vacancy formation energies of fcc transition metals calculated by a full potential Green's function method, Solid State Communications 79 (1) (1991) 31–35. doi:10.1016/0038-1098(91)90474-A.

[110] H. L. Skriver, N. M. Rosengaard, Self-consistent green's-function technique for surfaces and interfaces, Physical Review B 43 (1991) 9538–9549. doi:10.1103/PhysRevB.43.9538.