ASE2SPRKKR

Python interface to SPR-KKR electronic structure code

Matyáš Novák
novakmat@fzu.cz

Content

- Existing tools
  - SPR-KKR (+ xband)
  - ASE - Atomic simulation enviroment
- ASE2SPRKKR
  - How to create the structures
  - How to specify the calculation's parameters
  - How to run the calculations
  - How to install
  - A few about the background

SPR-KKR — band structure program package

Advantages and capabilities

- Arbitrary ordered/disordered 3D periodic system
- Surfaces or slab approximation
- Spin-polarised and non-collinear-spin configurations
- SCF-potential, dispersion realtion, Bloch spectral function, density of states
- Spin- and orbital moment
- Response functions: spin and orbital susceptibility, Knight-shift, field-induced MCXD, residual resistivity of Alloys
- Spectoscopic properties incl. magnetic dichroism

SPR-KKR - basic properties

Architecture

- Fortran language
- Parallelized by MPI
- Current? version 9.0

Executables: kkrscf, kkrgen, kkrchi, kkrspec, ebscf, embgen

Problem definition

- Input and output files in text format
    - *input (parameters) file* – parameters of the calculation
    - *potential file* – a definition of the structure
- xband – legacy Tcl GUI to SPR-KKR

Input (parameters) file

. . . define the type of the calculation and its parameters.

- non-whitespace on the first line ⇒ new section
- options of various types (integer, floating point, array of numbers, string)
- CONTROL.POTFIL: filename of the *potential file*

```
CONTROL
    DATASET=Fe
    POTFIL=Fe.pot
    PRINT=0

STRCONST
    ETA=0.35 RMAX=2.9 GMAX=3.3

TAU
    BZINT= POINTS NKTAB=250

ENERGY
    NE=30 EMIN=-0.2

SCF
    NITER=200 MIX=0.20 SCFVXC=VWN
    TOL=0.00001 ISTBRY=1

TASK SCF
```

Potential file

. . . defines the structure, lattice and (on output, for subsequent calculations) the computed potential.

- name-value or/and table like structure of sections
- the structure varies by sections
- sections are stars-delimited

```
*************************************************************
HEADER       SPR-KKR potential file, created at 2023-01-27 00:
*************************************************************
TITLE        Created by ASE-SPR-KKR wrapper
SYSTEM       System: Li
PACKAGE      SPR-KKR
FORMAT        7 (21.05.2007)
*************************************************************
GLOBAL SYSTEM PARAMETER
NQ           1
NT           1
NM           1
IREL         3
*************************************************************
SCF-INFO
INFO         NONE
SCF-MIX      0.2
SCF-TOL      1e-05
. ..
VMTZ         0.7
*************************************************************
LATTICE
SYSDIM       3D
SYSTYPE      BULK
BRAVAIS       14 cubic body-centered m3m O_h
ALAT         6.59514417917088
A(1)                   -0.5              0.5              0.5
A(2)                    0.5             -0.5              0.5
A(3)                    0.5              0.5             -0.5
*************************************************************
SITES
```

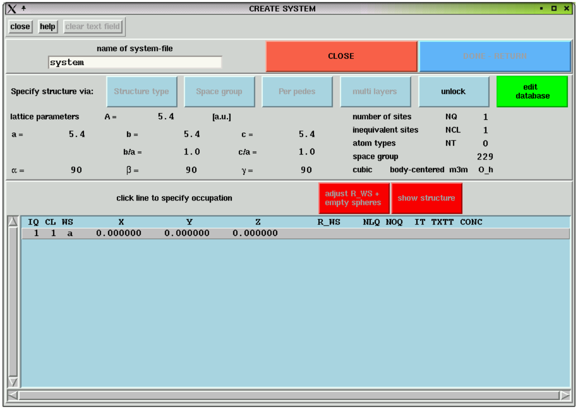
xband

GUI for generating and running SPR-KKR (and others).

Feature rich

. . . but . . .

user friendly?



name of system-file: system

Specify structure via:

lattice parameters	A =	5.4	[a.u.]		number of sites	NQ	1	
a =	5.4	b =	5.4	c = 5.4	inequivalent sites	NCL	1	
		b/a =	1.0	c/a = 1.0	atom types	NT	0	
					space group		229	
α =	90	β =	90	γ = 90	cubic	body-centered	m3m	O_h

click line to specify occupation

```
IQ  CL  WS        X          Y          Z             R_WS    NLQ  NOQ   IT  TXTT  CONC
 1   1   a    0.000000   0.000000   0.000000
```

ASE - atomic simulation environment

- Python framework
- Iterface to many electronic structure calculating packages, e.g.
  - Castep
  - Fleur
  - Quantum Expresso
  - Vasp
  - ...
  - **SPR-KKR** :-)

- Easy structure definition
- The full strength and elegance of the python language
- One common input format for many programs
- Databases of the common structures

ASE - basics

The Atoms object

. . . defines

- structure of the material
- lattice
- symmetry
- occupation

Common for the all underlying packages (calculators)

ASE - basics

The Atoms object

. . . defines

- structure of the material
- lattice
- symmetry
- occupation

Common for the all underlying packages (calculators)

Calculators objects

. . . provide an interface to the given program.

- set the parameters of the calculation
- call the proper executable/routine
- read the results of the calculation

Each package (Vasp, Fleur, . . .) has its own calculator

ASE - defining the structure

(taken from ASE documentation)

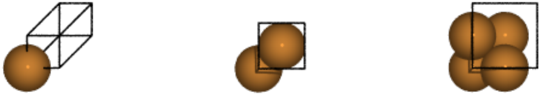
```
from ase.build import bulk
cu_atoms = bulk('Cu', 'fcc', a=3.6)
cu_orthorhombic = bulk('Cu', 'fcc', a=3.6, orthorhombic=True)
cu_cubic = bulk('Cu', 'fcc', a=3.6, cubic=True)
```

```
a = 4.0
Pt3Rh = Atoms('Pt3Rh', cell=[a, a, a],  pbc=True,
        scaled_positions=[(0, 0, 0), (0.5, 0.5, 0),
                          (0.5, 0, 0.5), (0, 0.5, 0.5)])
s3 = surface(Pt3Rh, (2, 1, 1), 9)
s3.center(vacuum=10, axis=2)
```

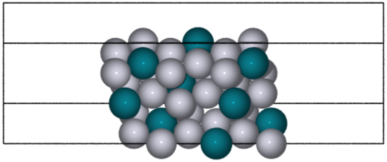
ASE - defining the structure

(taken from ASE documentation)

```
from ase.build import bulk
cu_atoms = bulk('Cu', 'fcc', a=3.6)
cu_orthorhombic = bulk('Cu', 'fcc', a=3.6, orthorhombic=True)
cu_cubic = bulk('Cu', 'fcc', a=3.6, cubic=True)
```



```
a = 4.0
Pt3Rh = Atoms('Pt3Rh', cell=[a, a, a],  pbc=True,
        scaled_positions=[(0, 0, 0), (0.5, 0.5, 0),
                          (0.5, 0, 0.5), (0, 0.5, 0.5)])
s3 = surface(Pt3Rh, (2, 1, 1), 9)
s3.center(vacuum=10, axis=2)
```



ASE - calculate

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(atoms=atoms)
calculator.calculate()
```

ASE - calculate

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(atoms=atoms)
calculator.calculate()
```

Hey, where are the parameters??

ASE - calculate

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(atoms=atoms)
calculator.calculate()
```

Hey, where are the parameters??

```
calculator = SPRKKR(atoms=atoms, input_parameters='SCF')
```

ASE - calculate

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(atoms=atoms)
calculator.calculate()
```

Hey, where are the parameters??

```
calculator = SPRKKR(atoms=atoms, input_parameters='SCF')
```

and/or

```
calculator.input_parameters = 'PHAGEN'
calculator.input_parameters.TAU.NKTAB = 1e-5
```

ASE - calculate

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(atoms=atoms)
calculator.calculate()
```

Hey, where are the parameters??

```
calculator = SPRKKR(atoms=atoms, input_parameters='SCF')
```

and/or

```
calculator.input_parameters = 'PHAGEN'
calculator.input_parameters.TAU.NKTAB = 1e-5
```

and/or

```
calculator.calculate(input_parameters=...)
```

ASE - calculate

```
from ase2sprkkr import SPRKKR
calculator = SPRKKR(atoms=atoms)
calculator.calculate()
```

Hey, where are the parameters??

```
calculator = SPRKKR(atoms=atoms, input_parameters='SCF')
```

and/or

```
calculator.input_parameters = 'PHAGEN'
calculator.input_parameters.TAU.NKTAB = 1e-5
```

and/or

```
calculator.calculate(input_parameters=...)
```

and/or

```
calculator.calculate(options={'NKTAB':5, 'SCF.TOL':0.1,
'SITES': {NL:2}})
```

Setting InputParameters

Input parameters can be initialized by

- A task name (`SCF`, `PHAGEN`, `ARPES`, `DOS`)
  the default values will be used
- A filename
  the parameters will be readed from the a file
- `ase2sprkkr.InputParameters` object
  e.g. created by `ase2sprkkr.InputParameters.from_file()`

Setting `InputParameters`

Input parameters can be initialized by

- A task name (`SCF`, `PHAGEN`, `ARPES`, `DOS`)
  the default values will be used
- A filename
  the parameters will be readed from the a file
- `ase2sprkkr.InputParameters` object
  e.g. created by `ase2sprkkr.InputParameters.from_file()`

. . . and modified using

- a direct access
  `input_parameters.SCF.TOL=1e-5`
- a set method (a dictionary as the argument)
  `input_parameters.set({'TOL : 1e-5})`

InputParameters validation

```
>>> calc.input_parameters.SCF.TOL = 'not a float value'

ValueError: Value 'not a float_value' for paramater TOL
of type Real is not valid...
```

InputParameters validation

```
>>> calc.input_parameters.SCF.TOL = 'not a float value'

ValueError: Value 'not a float_value' for paramater TOL
of type Real is not valid...
```

From version 2.0.0

```
>>> calc.input_parameters.SCF.TOL.set_dangerous('x')
>>> calc.input_parameters.SCF.TOL()
'x'
```

Unknown input parameter

```
>>> calc.input_parameters.SCF.UNKNOWN = 1.0
...
AttributeError: There is no value with name UNKNOWN
in SECTION SCF.
Maybe, you want to add a custom value using
the add method?
```

Unknown input parameter

```
>>> calc.input_parameters.SCF.UNKNOWN = 1.0
...
AttributeError: There is no value with name UNKNOWN
in SECTION SCF.
Maybe, you want to add a custom value using
the add method?
```

So, let's do as they ask...

```
>>> calc.input_parameters.SCF.add('UNKNOWN', 1.0)
>>> calc.input_parameters.SCF.UNKNOWN()
1.0 >>> calc.input_parameters.SCF.UNKNOWN = 'x'
>>> calc.input_parameters.SCF.UNKNOWN()
'x'
>>> calc.input_parameters.SCF.UNKNOWN.remove()
```

Inspecting the input parameters

- Read the value of an option by calling it:

```
>>> atoms.input_parameters.SCF.TOL()
```

- The <TAB> key is your best friend!

```
>>> atoms.input_parameters.SCF.<TAB>
```

- `to_dict()` method and `to_string()` method.

```
>>> print(atoms.input_parameters.SCF.to_string())
```

- And of course, the help is available!

```
>>> atoms.input_parameters.SCF.help()
```

- Even a more descriptive one:

```
>>> atoms.input_parameters.SCF.help(True)
```

```
Configuration section SCF

SECTION SCF contains:
---------------------
    NITER : Integer = 200                   Maximal number of iterations of the S
    MIX : Real = 0.2                        Mixing parameter
    VXC : AnyOf(VWN,MJW,VBH,PBE) = VWN

        Possible values:
          VWN       Vosko, Wilk, Nusair
          MJW       Janak, Williams, Moruzzigit g
          VBH       von Barth, Hedin
          PBE       Perdew, Burke, Ernzendorfer GGA

    ALG : AnyOf(BROYDEN2,TCHEBY) = BROYDEN2

        Possible values:
          BROYDEN2  Broyden's second method
          TCHEBY    Tchebychev

    EFGUESS : Real = 0.7
    TOL : Real = 1e-05                      Tolerance threshold for the mixing al
    ISTBRY : Integer = 1                    Start Broyden after ISTBRY iterations
    ITDEPT : Integer = 40                   Iteration depth for Broyden algorithm
    QION : Array(of Real)  (optional)       Guess for the ionic charges Qt for at
    MSPIN : Array(of Real)  (optional)      Guess for the magnetic moment u\_{spi
```

Options

Each option has

- name
- type
- default value (not necessary)
- flags (properties)

Flags can be

- **optional** – value is not needed
- **read only** – value can't be changed
- **expert** – the option is printed to the output, only if differs from the default value

The task

- The available options are determined by the **TASK**

```
>>> calculator.input_parameters.TASK.TASK()
```

- Task is determined during creating the parameters
- Task can be changed only by replacing the input parameters.

```
>>> calculator.input_parameters = 'PHAGEN'
>>> calculator.calculate(input_parameters='PHAGEN')
```

- However, you can copy the options (in version 2.0)

```
>>> options = calculator.input_parameters.to_dict()
>>> calculator.input_parameters = 'PHAGEN'
>>> calculator.input_parameters.set(
                                options,
                                unknown='ignore'
                                )
```

The called executable

- Task determines the executable to be ran. The calculator argument `executable_suffix` (the default value is the environment variable `SPRKKR_EXECUTABLE_SUFFIX`) is appended to the executable name)
- In version 2.0, you can ask for the executable:

```
>>> calculator.input_parameters.get_executable()
          [ 'kkrscf_myhostname' ]
```

- and override it:

```
>>> input_parameters.set_executable(
['rm','-rf','/']
)
```

(but, do not do it, please. . . :-))

Running the executable

The `calculate()` method

- saves the input parameters

  the `input_file` parameter controls the filename
- saves the potential file

  the `potential_file` parameter
- run the executable

  thats whay not to set it to `rm -rf` :-)
- stores the output of the called program to the given file

  if the `output_file` parameter have been specified
- parses the output of the runned process

  currently, it is implemented only for the SCF task
- returns the result object

The `print_output` parameter (accepts `True`, `False`,or the default `'info'`) controls the amount of the output

Result

Currently, for SCF task, the result has the parameters:

- **energy**
- **converged**
- **potential** the result potential.
- **calculator** the (new) calculator object, associated with the result potential.
- **iterations** array of iterations data
  - **iteration**
  - **energy**
  - **error**
  - **moment** (spin and orbital)

So, to run subsequent calculations, you can:

```
out = calc.calculate(input_parameters='SCF',options={...})
out.calculator.calculate(
                  input_parameters='PHAGEN',options={...})
```

MPI

No worry, its simple. Just pass to the `mpi` parameter of the `calculate` method:

- `True`

  if batch system is used

- an integer

  to determine the number of processes

  ```
  >>> calculator.calculate(..., mpi=4)
  ```

- `[ 'command', 'parameter', 'parameter', ...]`

  to achieve anything more special, e.g.

  ```
  >>> calculator.calculate(...,
                               mpi=[ 'mpirun', '-np', 4]
                              )
  ```

The `'MPI'` suffix to the executable is appended automatically.

ASE2SPRKKR Installation

Either

- `pip install ase2sprkkr`
- `conda install ase2sprkkr`
- `pip install -pre ase2sprkkr` for the development versions
- `git clone https://github.com/ase2sprkkr/ase2sprkkr.git` ./install.sh

  for the bleeding edge version and for development

A bit of the background - enhancing the Atoms

ASE `Atoms` object (the structure) is “enhanced”, when

- it is pass to the calculator
- `SPRKKRAtoms.promote_ase_atoms` is called

Promoted `Atoms` receives `sites` property, which allows to

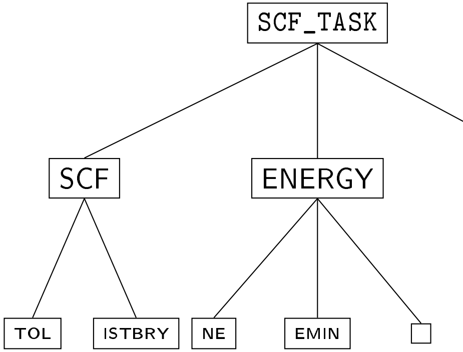
- deal with a symmetry
- specify occupation (in a better way than in ASE)

  ```
  >>> atoms.sites[3].occupation.set({'Cl': 0.5,'I': 0.5})
  ```
- specify SPRKKR radial meshes
- specify number of valence and semi-valence electrons
- ...

A bit of the background - Task definitions

- Available options are given by definitions
- Tree-like structure
- Each option has own `GrammarType`
- Grammar type defines the input & output format
- `GrammarTypes` can be combined to make lists, tables, etc.

So, if you miss an option, you can alter the definition and send me a pullrequest...



```
""" SCF task input parameters definition"""
from ...common.grammar_types  import *
from .sections import *
from ..input_parameters_definitions import \
    InputParametersDefinition as InputParameters, \
    InputValueDefinition as V

input\_parameters = InputParameters(
      'scf', [
        CONTROL('SCF').copy([
          V('KRWS', 1)
        ]),
        TAU,
        ENERGY,
        SCF,
        SITES,
        STRCONST,
        CPA,
        MODE
      ],
      info = "SCF - calculate a .... potential",
      description = "",
      executable = 'kkrscf',
      mpi = True
)
""" SCF task input parameters definition"""

from ...common.doc import process\_input\_parameters\_definition
process_input_parameters_definition(\_\_name\_\_)
```

```
SCF = Section('SCF', [
      V('NITER', 200, info='Maximal number of iteration
      V('MIX', 0.2, info='Mixing parameter'),
      V('VXC', DefKeyword({
        'VWN' : 'Vosko, Wilk, Nusair',
        'MJW' : 'Janak, Williams, Moruzzigit g',
        'VBH' : 'von Barth, Hedin',
        'PBE' : 'Perdew, Burke, Ernzendorfer GGA'
        }), info='parametrisation of the exchange-corr
      V('ALG', DefKeyword({
          'BROYDEN2': 'Broyden's second method',
          'TCHEBY': 'Tchebychev'
        }), info='Mixing algorithm'),
      V('EFGUESS', 0.7),
      V('TOL', 0.00001, info='Tolerance threshold for
      V('ISTBRY', 1, info='Start Broyden after ISTBRY
      V('ITDEPT', 40, info='Iteration depth for Broyde
      V('QION', Array(float), required=False, info='Gu
      V('MSPIN', Array(float), required=False, info='G
      V('USEVMATT', False, info='Set up the starting p
                                    'construction for the
  ])
"""The definition of the SCF section of the task input
```

Conclusion

. . . some superb superlatives about ASE2SPRKKR. . .

- ASE2SPRKKR shloud serve to you, thus. . .

Conclusion

. . . some superb superlatives about ASE2SPRKKR. . .

- ASE2SPRKKR shloud serve to you, thus. . .
- . . . if you have an recommendation, suggestion etc..., don't hesitate to tell me

Thank you for your attention.