Wannier90/Tight-Binding library

Motivation

- Powerful packages available for Wannier Hamiltonians (WANNIER90, wannier-tools), analysis beyond predefined functions is tedious, e.g:
    - Self-defined k-paths
    - Access to H(k)
    - Access to $|\Psi\rangle$
    - Post-processing of observables
    - No object-oriented work-flow
- General applicability to all kind of tight-binding Hamiltonians

Classes

hamiltonian

$H(k)$
$\nabla H(k)$
$H(R)$-spinless
$H(k)$-spinless
$\nabla H(k)$-spinless
$H(k)$ to w2dyn format

$H(R)$
Bravais vectors
Spin
R, R_{cart}
Basis $(s, p, d, \ldots)$

k_space

Cartesian ⇄Reduced
k-distance
Path/Grid/Monkhorst
Stereographic proj.

Bravais vectors
k_space_red
k_space_car

operator

L, S, J, Ω_n, Band inversion…
Self-defined operators
postprocessing (J^2,Band-/k-integratrion)

$\hat{O}$
val (Array for storing $\langle \hat{O} \rangle$)
val_k_int
val_b_int

observables

---

initialize_ops()
calculate_ops()
post_ops()
write_ops()

---

op_types
op_types_k

Installation

Initializing a Wannier Hamiltonian

hamiltonian(HR_FILE,BRA_VEC=None,SPIN=False,BASIS=None,EF=None,N_ELEC
None)

Required Arguments		
HR_FILE	String	Filename/path

Optional Arguments		
BRA_VEC	np-array	Bravais vectors
SPIN	Boolean	Spin
BASIS	np-array	L-quantum numbers
EF	Float	Fermi energy
N_ELEC	Integer	Number of electrons

Initializing a Wannier Hamiltonian Example

hamiltonian(HR_FILE,BRA_VEC=None,SPIN=False,BASIS=None,EF=None, N_ELEC=None)

```
#### Setting up Hamiltonian

inputfile  = "../../test_ham/In_SiC_soc.dat"
bra_vec  = np.array([
   [ 3.0730000,    0.0000000,    0.0000000],
   [-1.5365000,    2.6612960,    0.0000000],
   [ 0.0000000,    0.0000000,   20.0000000]
   ])

spin       = True
basis      = np.array([0,1]) # {Ta1:d,Ta2:d,As1:p,As2:p}
n_elec    = 4
ef          = 3.223

Ham = hamiltonian(inputfile,bra_vec,spin,basis,ef,N_ELEC=n_elec)
```

Initializing a k-space

k_space(K_TYPE,K_BASIS,VECS,BRA_VEC=None,N_POINTS=None,
RADIUS=None)

Required Arguments		
K_TYPE	String	Specifies k-space
BASIS	String	“red/car”
VECS	np-array	Vectors spanning k-space

Optional Arguments		
BRA_VEC	np-array	Bravais vectors
N_POINTS	Integer	Number of k-points
RADIUS	Float	Radius for Pontriyagin integration

k_space(K_TYPE,K_BASIS,VECS,BRA_VEC=None,N_POINTS=None,
RADIUS=None)

```
#### Setting up k-space

ktype      = "path"
kbasis     = "red"
vecs       = np.array([
                [ 1/3, 1/3,0],
                [ 1/2,   0,0],
                [   0,   0,0]])
bra_vec  = np.array([
   [ 3.0730000,   0.0000000,   0.0000000],
   [-1.5365000,   2.6612960,   0.0000000],
   [ 0.0000000,   0.0000000,  20.0000000]
   ])
npoints   = 100

K_space = k_space(ktype,kbasis,vecs,bra_vec,npoints)
```

Initializing an Observables calculation

observables(HAMILTONIAN,K_SPACE,OP_TYPES=[],OP_TYPES_K=[],PREFIX=“")

Required Arguments		
HAMILTONIAN	hamiltonian	Ham. object
K_SPACE	k_space	k_space object

Optional Arguments		
OP_TYPES	Array	Contains k-independent operators
OP_TYPES_K	Array	Contains k-dependent operators

Initializing and running observables calculation
Example

observables(HAMILTONIAN,K_SPACE,OP_TYPES=[],OP_TYPES_K=[],PREFIX=“”)

```
#### Defining operators

op_types  =["S","L","J"]
op_types_k=["BC","BC_mag","Orb_SOC_inv"]

#### Running calculation

# Initializing observables
Observables = observables(Ham,K_space,op_types,op_types_k)

# Calculating observables
Observables.calculate_ops()
```