

---

# **wannier90-utils**

***Release v0.1.0***

**Jamal I. Mustafa**

**Mar 30, 2021**



# CONTENTS

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Documentation</b>	<b>7</b>
3.1	Manipulating WIN files . . . . .	7
3.2	Parsing WOUT files . . . . .	8
3.3	Parsing NNKP files . . . . .	8
3.4	Fundamental I/O routines . . . . .	8
3.5	Wannier function centers and spreads . . . . .	9
3.6	Processing <code>postw90</code> output . . . . .	10
3.7	Examples . . . . .	11
<b>4</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



This package provides a library of functions for reading/writing and manipulating the data associated with the [wannier90](#) code [1].



## FEATURES

- Routines for reading/writing/manipulating a variety of files
  - manipulating WIN files (see [here](#))
  - parsing WOUT files (see [here](#))
  - the nnkp file (see [here](#))
  - the eigenvalues, overlap matrices, and projection matrices (see [here](#))
  - output of postw90 program, such as bandstructures (see [here](#))
- Utilities for computing the centers and spreads of Wannier functions (see [here](#))





## INSTALLATION

To install the latest version of the wannier90-utils package, simply clone the repository and install using `pip`.

```
git clone https://github.com/jimustafa/wannier90-utils.git
cd wannier90-utils && pip install .
```



## 3.1 Manipulating WIN files

Wannier90 I/O routines pertaining to WIN files

```
w90utils.io.win.print_atoms (atoms, units='crystal', file=<_io.TextIOWrapper name='<stdout>'
                             mode='w' encoding='UTF-8'>)
```

```
w90utils.io.win.print_kgrid (kgrid, file=<_io.TextIOWrapper name='<stdout>' mode='w'
                             encoding='UTF-8'>)
```

```
w90utils.io.win.print_kpoints (kpoints, mp_grid=None, file=<_io.TextIOWrapper
                               name='<stdout>' mode='w' encoding='UTF-8'>)
```

```
w90utils.io.win.print_unit_cell (dlv, units='bohr', file=<_io.TextIOWrapper name='<stdout>'
                                 mode='w' encoding='UTF-8'>)
```

```
w90utils.io.win.read_atoms (fname, units='crystal')
```

```
w90utils.io.win.read_dlv (fname, units='bohr')
```

Read direct lattice vectors from WIN file.

### Parameters

- **fname** (*str*) – Wannier90 WIN file
- **units** (*str*, {'bohr', 'angstrom'}) – units of returned lattice vectors

**Returns** *dlv* (*ndarray*, *shape* (3, 3)) – direct lattice vectors

```
w90utils.io.win.read_kgrid (fname)
```

```
w90utils.io.win.read_kpoints (fname)
```

```
w90utils.io.win.read_proj_line (line, dlv, basis, spinors)
```

```
w90utils.io.win.read_projections (fname)
```

```
w90utils.io.win.remove_comments (s)
```

## 3.2 Parsing WOUT files

Wannier90 I/O routines pertaining to WOUT files

`w90utils.io.wout.read_centers_xyz(fname)`

`w90utils.io.wout.read_conv(fname)`

`w90utils.io.wout.read_sprd(fname)`

## 3.3 Parsing NNKP files

Wannier90 I/O routines pertaining to NNKP files

`w90utils.io.nnkp.read_bvectors(fname, units='angstrom')`

`w90utils.io.nnkp.read_dlv(fname, units='bohr')`

`w90utils.io.nnkp.read_excluded_bands(fname)`

`w90utils.io.nnkp.read_kpoints(fname, units='crystal')`

`w90utils.io.nnkp.read_nnkpts(fname)`

`w90utils.io.nnkp.read_projections(fname)`

`w90utils.io.nnkp.read_rlv(fname, units='bohr')`

## 3.4 Fundamental I/O routines

<code>w90utils.io.read_eig(fname)</code>	Read EIG file.
<code>w90utils.io.write_eig(fname, eig)</code>	Write $E_{nk}$ to EIG file.
<code>w90utils.io.read_hamiltonian(fname)</code>	Read EIG file and return k-dependent Hamiltonian matrix.
<code>w90utils.io.read_amn(fname)</code>	Read AMN file.
<code>w90utils.io.write_amn(fname, amn[, header])</code>	Write $A_{mn}^{(k)}$ to AMN file.
<code>w90utils.io.read_mmn(fname)</code>	Read MMN file
<code>w90utils.io.write_mmn(fname, mmn, kpb_kidx, ...)</code>	Write $M_{mn}^{(k,b)}$ to MMN file

`w90utils.io.read_eig(fname)`

Read EIG file.

**Parameters** `fname` (*str*) – path to EIG file

**Returns** *ndarray, shape (nkpts, nbnds, nproj)*

`w90utils.io.write_eig(fname, eig)`

Write  $E_{nk}$  to EIG file.

**Parameters**

- `fname` (*str*) – path to EIG file
- `eig` (*ndarray, shape (nkpts, nbnds)*) –

`w90utils.io.read_hamiltonian(fname)`

Read EIG file and return k-dependent Hamiltonian matrix.

**Parameters** `fname` (*str*) – path to EIG file

**Returns** *ndarray, shape (nkpts, nbnds, nbnds)*

`w90utils.io.read_amn(fname)`

Read AMN file.

**Parameters** `fname` (*str*) –

**Returns** *ndarray, shape (nkpts, nbnds, nproj)*

`w90utils.io.write_amn(fname, amn, header='HEADER')`

Write  $A_{mn}^{(k)}$  to AMN file.

**Parameters**

- `fname` (*str*) –
- `amn` (*ndarray, shape (nkpts, nbnds, nproj)*) –
- `header` (*str*) –

`w90utils.io.read_mmn(fname)`

Read MMN file

**Parameters** `fname` (*str*) –

**Returns** *ndarray, shape (nkpts, nntot, nbnds, nbnds)*

`w90utils.io.write_mmn(fname, mmn, kpb_kidx, kpb_g)`

Write  $M_{mn}^{(k,b)}$  to MMN file

**Parameters**

- `fname` (*str*) –
- `mmn` (*ndarray, shape (nkpts, nntot, nbnds, nbnds)*) –

## 3.5 Wannier function centers and spreads

Functions for computing Wannier centers and components of the spread

`w90utils.sprd.omega(Mmn, bvectors, bweights)`

Compute the spread functional

**Parameters**

- `Mmn` (*ndarray, shape (nkpts, nntot, nbnds, nbnds)*) – the overlap matrix
- `bvectors` (*ndarray, shape (nkpts, nntot, 3)*) –
- `bweights` (*ndarray, shape (nntot,)*) –

`w90utils.sprd.omega_d(m, bvectors, bweights, idx=None)`

Compute the diagonal contribution to the spread functional

**Parameters**

- `m` (*ndarray, shape (nkpts, nntot, nbnds, nbnds)*) – the overlap matrix
- `bvectors` (*ndarray, shape (nkpts, nntot, 3)*) –

- **bweights**(ndarray, shape (nntot,)) –

w90utils.sprd.omega\_dod(Mmn, bvectors, bweights)

Compute the sum of the diagonal and off-diagonal contribution to the spread functional

**Parameters**

- **Mmn**(ndarray, shape (nkpts, nntot, nbnds, nbnds)) – the overlap matrix
- **bvectors**(ndarray, shape (nkpts, nntot, 3)) –
- **bweights**(ndarray, shape (nntot,)) –

w90utils.sprd.omega\_i(Mmn, bweights)

Compute the invariant contribution to the spread functional

**Parameters**

- **Mmn**(ndarray, shape (nkpts, nntot, nbnds, nbnds)) – the overlap matrix
- **bweights**(ndarray, shape (nntot,)) –

w90utils.sprd.omega\_iod(m, bweights, idx=None)

Compute the sum of the invariant and off-diagonal contribution to the spread functional

**Parameters**

- **m**(ndarray, shape (nkpts, nntot, nbnds, nbnds)) – the overlap matrix
- **bweights**(ndarray, shape (nntot,)) –

w90utils.sprd.omega\_od(Mmn, bweights)

Compute the off-diagonal contribution to the spread functional

**Parameters**

- **Mmn**(ndarray, shape (nkpts, nntot, nbnds, nbnds)) – the overlap matrix
- **bweights**(ndarray, shape (nntot,)) –

w90utils.sprd.wannier\_centers(m, bvectors, bweights)

## 3.6 Processing postw90 output

```
w90utils.io.postw90.print_kpoints(kpoints, header="", units='crystal',
                                  file=<_io.TextIOWrapper name='<stdout>' mode='w'
                                  encoding='UTF-8'>)
```

```
w90utils.io.postw90.read_band_velocities(fname)
```

```
w90utils.io.postw90.read_bands(fname)
```

```
w90utils.io.postw90.read_bands_kpoints(fname)
```

Read k-points from the geninterp dat file

**Parameters** **fname**(str) –

**Returns** ndarray, shape (nkpts, 3) – array of kpoints using for geninterp, in units of  $x_{212}B^{-1}$

```
w90utils.io.postw90.read_dos(fname)
```

```
w90utils.io.postw90.read_elcond(fname)
```

```
w90utils.io.postw90.read_kpoints(fname)
w90utils.io.postw90.read_vnk(fname)
w90utils.io.postw90.write_kpoints(fname, kpoints)
```

## 3.7 Examples

### 3.7.1 Read the Hamiltonian in the Wannier representation

```
from w90utils import io as w90io
HR, Rvectors, Rweights = w90io.read_hr('wannier_hr.dat')
```





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### W

w90utils.io.nnkp, [8](#)  
w90utils.io.postw90, [10](#)  
w90utils.io.win, [7](#)  
w90utils.io.wout, [8](#)  
w90utils.sprd, [9](#)



## O

omega() (in module w90utils.sprd), 9  
 omega\_d() (in module w90utils.sprd), 9  
 omega\_dod() (in module w90utils.sprd), 10  
 omega\_i() (in module w90utils.sprd), 10  
 omega\_iod() (in module w90utils.sprd), 10  
 omega\_od() (in module w90utils.sprd), 10

## P

print\_atoms() (in module w90utils.io.win), 7  
 print\_kgrid() (in module w90utils.io.win), 7  
 print\_kpoints() (in module w90utils.io.postw90), 10  
 print\_kpoints() (in module w90utils.io.win), 7  
 print\_unit\_cell() (in module w90utils.io.win), 7

## R

read\_amn() (in module w90utils.io), 9  
 read\_atoms() (in module w90utils.io.win), 7  
 read\_band\_velocities() (in module w90utils.io.postw90), 10  
 read\_bands() (in module w90utils.io.postw90), 10  
 read\_bands\_kpoints() (in module w90utils.io.postw90), 10  
 read\_bvectors() (in module w90utils.io.nnkp), 8  
 read\_centers\_xyz() (in module w90utils.io.wout), 8  
 read\_conv() (in module w90utils.io.wout), 8  
 read\_dlv() (in module w90utils.io.nnkp), 8  
 read\_dlv() (in module w90utils.io.win), 7  
 read\_dos() (in module w90utils.io.postw90), 10  
 read\_eig() (in module w90utils.io), 8  
 read\_elcond() (in module w90utils.io.postw90), 10  
 read\_excluded\_bands() (in module w90utils.io.nnkp), 8  
 read\_hamiltonian() (in module w90utils.io), 8  
 read\_kgrid() (in module w90utils.io.win), 7  
 read\_kpoints() (in module w90utils.io.nnkp), 8  
 read\_kpoints() (in module w90utils.io.postw90), 10  
 read\_kpoints() (in module w90utils.io.win), 7  
 read\_mmn() (in module w90utils.io), 9  
 read\_nnkppts() (in module w90utils.io.nnkp), 8

read\_proj\_line() (in module w90utils.io.win), 7  
 read\_projections() (in module w90utils.io.nnkp), 8  
 read\_projections() (in module w90utils.io.win), 7  
 read\_rlv() (in module w90utils.io.nnkp), 8  
 read\_sprd() (in module w90utils.io.wout), 8  
 read\_vnk() (in module w90utils.io.postw90), 11  
 remove\_comments() (in module w90utils.io.win), 7

## W

w90utils.io.nnkp (module), 8  
 w90utils.io.postw90 (module), 10  
 w90utils.io.win (module), 7  
 w90utils.io.wout (module), 8  
 w90utils.sprd (module), 9  
 wannier\_centers() (in module w90utils.sprd), 10  
 write\_amn() (in module w90utils.io), 9  
 write\_eig() (in module w90utils.io), 8  
 write\_kpoints() (in module w90utils.io.postw90), 11  
 write\_mmn() (in module w90utils.io), 9