py3nj Documentation

Release 0.1.2

xr-scipy Developers

2021-06-19

EXAMPLES

py3nj is a small library to calculate Wigner symbols, such as wigner's 3j, 6j, 9j symbols, as well as Clebsch Gordan coefficients.

py3nj mostly wraps the original Fortran implementation in slatec, but it is designed to highly compatible to numpy's nd-array, i.e. the automatic vectorization is supported.

CHAPTER

ONE

INSTALLING

py3nj is available on pypi. To install

`` `bash pip install py3nj ` ``

You may need fortran compiler installed in your environment.

CHAPTER

TWO

DOCUMENTATION

Examples

- *Examples*

2.1 Examples

2.1.1 Basic interfaces

Most basic interface are `wigner3j()`, `wigner6j()`, `wigner9j()`, `clebsch_gordan()`.

For example, if you want to compute

$$\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}.$$

then pass the doubled value, [0, 1, 1, 0, 1, -1] to `wigner3j()`.

```
In [1]: py3nj.wigner3j(0, 1, 1,
   ...:                0, 1, -1)
   ...:
Out[1]: 0.7071067811865476
```

The arguments should be integer or array of integers.

All the functions of py3nj accept array-like as arguments,

```
In [2]: py3nj.wigner3j([0, 1], [1, 2], [1, 1],
   ...:                [0, -1], [1, 2], [-1, -1])
   ...:
Out[2]: array([ 0.70710678, -0.57735027])
```

where the output has the same size of the input. `np.ndarray` with more than 1 dimension can be also used.

This vectorization not only reduce the python overhead, but also reusing the result with the same argument. Therefore, if you need to compute these coefficients for many cases, it is recommended to consider how your calculation can be vectorized.

2.1.2 Advanced interfaces

py3nj wraps slatec fortran implementation. The similar interfaces to the original slatec functions, `wigner.drc3jj()` and `drc6j()` are also supported.

This function computes all the possible values of J_1 and their corresponding 3j symbol with given J_2, J_3, M_2, M_3 values,

```
In [3]: two_l1, three_j = py3nj.wigner.drc3jj(1, 1, 1, 1)

In [4]: two_l1
Out[4]: array([0, 1, 2])

In [5]: three_j
Out[5]: array([ 0.        ,  0.        , -0.57735027])
```

This function can be also vectorized,

```
In [6]: two_l1, three_j = py3nj.wigner.drc3jj([1, 0], [1, 2], [1, 0], [1, 2])

In [7]: two_l1
Out[7]: array([0, 1, 2])

In [8]: three_j
Out[8]:
array([[ 0.        ,  0.        , -0.57735027],
       [ 0.        ,  0.        ,  0.57735027]])
```

Note that even in this advanced interfaces, the vectorized version will be much faster than that sequencial calculation if you need many calcluations.

Help & reference

- *What's New*
- *API reference*

2.2 What's New

2.2.1 v0.1 (29 May 2018)

Initial release.

2.3 API reference

This page provides an auto-generated summary of py3nj's API. For more details and examples, refer to the relevant chapters in the main part of the documentation.

2.3.1 Top-level functions

wigner3j(two_l1, two_l2, two_l3, two_m1, …)	Calculate wigner 3j symbol (L1 L2 L3) (-M2-M3 M2 M3)
wigner6j(two_l1, two_l2, two_l3, two_l4, …)	Calculate wigner 6j symbol (L1 L2 L3) (L4 L5 L6)
wigner9j(two_l1, two_l2, two_l3, two_l4, …)	Calculate wigner 9j symbol (L1 L2 L3) (L4 L5 L6) (L7 L8 L9)
clebsch_gordan(two_j1, two_j2, two_j3, …)	Calulate Clebsch-Gordan coefficient <j1 m1, j2 m2 | j3 m3>

py3nj.wigner3j

`py3nj.wigner3j`(*two_l1*, *two_l2*, *two_l3*, *two_m1*, *two_m2*, *two_m3*, *ignore_invalid=False*)

Calculate wigner 3j symbol (L1 L2 L3) (-M2-M3 M2 M3)

Parameters

two_l1: array of integers

two_l2: array of integers

two_l3: array of integers

two_m1: array of integers

two_m2: array of integers

two_m3: array of integers Since L1, …, M3 should be integers or half integers, two_l1 (which means 2 x L1) should be all integers.

ignore_invalid: boolean If True, returns 0 even for invalid arguments. Otherwise, raise a ValueError.

Returns

threej: array The value of 3J symbol with the same shape of the arguments.

py3nj.wigner6j

`py3nj.wigner6j`(*two_l1*, *two_l2*, *two_l3*, *two_l4*, *two_l5*, *two_l6*, *ignore_invalid=False*)

Calculate wigner 6j symbol (L1 L2 L3) (L4 L5 L6)

Parameters

two_l1: array of integers

two_l2: array of integers

two_l3: array of integers

two_l4: array of integers

two_l5: array of integers

two_l6: array of integers Since L1, …, L6 should be integers or half integers, two_l1 (which means 2 x L1) should be all integers.

ignore_invalid: boolean If True, returns 0 even for invalid arguments. Otherwise, raise a ValueError.

Returns

threej: array The value of 6J symbol with the same shape of the arguments.

py3nj.wigner9j

`py3nj.wigner9j`(*two_l1*, *two_l2*, *two_l3*, *two_l4*, *two_l5*, *two_l6*, *two_l7*, *two_l8*, *two_l9*)

Calculate wigner 9j symbol (L1 L2 L3) (L4 L5 L6) (L7 L8 L9)

defined as 2x (a b c) (d e f) (g h j)

sum_x (-1) (2x+1) (f j x) (b x h) (x a d)

2x (x f b) (x h b) (x a j)

sum_x (-1) (2x+1) (c a j) (e f d) (g h d)

Parameters

two_l1: array of integers

two_l2: array of integers

two_l3: array of integers

two_l4: array of integers

two_l5: array of integers

two_l6: array of integers

two_l7: array of integers

two_l8: array of integers

two_l9: array of integers Since L1, …, L9 should be integers or half integers, two_l1 (which means 2 x L1) should be all integers.

Returns

threej: array The value of 9J symbol with the same shape of the arguments.

py3nj.clebsch_gordan

`py3nj.clebsch_gordan`(*two_j1*, *two_j2*, *two_j3*, *two_m1*, *two_m2*, *two_m3*, *ignore_invalid=False*)

Calulate Clebsch-Gordan coefficient <j1 m1, j2 m2 | j3 m3>

Parameters

two_j1: array of integers

two_j2: array of integers

two_j3: array of integers

two_m1: array of integers

two_m2: array of integers

two_m3: array of integers Since j1, …, m3 should be integers or half integers, two_j1 (which means 2 x j1) should be all integers.

force_compute: boolean If True, returns 0 even for invalid arguments. Otherwise, raise a ValueError.

Returns

clebch-gordan: array The value of Clebsch Gordan coefficients, with the same size of the arguments.

2.3.2 Wigner module

wigner.drc3jj(two_l2, two_l3, two_m2, two_m3)	Calculate Wigner's 3j symbol (L1 L2 L3) (-M2-M3 M2 M3) for all the possible L1 values.
wigner.drc6j(two_l2, two_l3, two_l4, two_l5, ...)	Calculate Wigner's 6j symbol (L1 L2 L3) (L4 L5 L6) for all the possible L1 values.

py3nj.wigner.drc3jj

py3nj.wigner.**drc3jj**(*two_l2*, *two_l3*, *two_m2*, *two_m3*, *ignore_invalid=False*)
Calculate Wigner's 3j symbol (L1 L2 L3) (-M2-M3 M2 M3) for all the possible L1 values.

Parameters

two_l2: array of integers, size (...)

two_l3: array of integers, size (...)

two_m2: array of integers, size (...)

two_m3: array of integers, size (...) Since L2, ..., M3 should be integers or half integers, two_l1 (which means 2 x L1) should be all integers.

Returns

two_l1: 1d-np.ndarray of integer, shape (n,) The possible L1 values.

threej: array, shape (..., n) The value of 3J symbol

py3nj.wigner.drc6j

py3nj.wigner.**drc6j**(*two_l2*, *two_l3*, *two_l4*, *two_l5*, *two_l6*, *ignore_invalid=False*)
Calculate Wigner's 6j symbol (L1 L2 L3) (L4 L5 L6) for all the possible L1 values.

Parameters

two_l2: array of integers, size (...)

two_l3: array of integers, size (...)

two_l4: array of integers, size (...)

two_l5: array of integers, size (...)

two_l6: array of integers, size (...) Since L2, ..., L6 should be integers or half integers, two_l2, ... (whichs 2 x L1) should be all integers.

Returns

two l1: 1d-np.ndarray of integer, shape (n,) The possible L1 values.

threej: array, shape (..., n) The value of 3J symbol

CHAPTER

THREE

LICENSE

py3nj is available under the open source Apache License.

INDEX