
n-sphere Documentation

Release unknown

Wai-Shing Luk

Dec 24, 2020

Contents

1	Contents	3
1.1	License	3
1.2	Contributors	3
1.3	Changelog	3
1.4	n_sphere	4
2	Indices and tables	9
	Python Module Index	11
	Index	13

This is the documentation of **n-sphere**.

Note: This is the main page of your project's [Sphinx](#) documentation. It is formatted in [reStructuredText](#). Add additional pages by creating `rst`-files in `docs` and adding them to the [toctree](#) below. Use then [references](#) in order to link them from this page, e.g. [Contributors](#) and [Changelog](#).

It is also possible to refer to the documentation of other Python packages with the [Python domain syntax](#). By default you can reference the documentation of [Sphinx](#), [Python](#), [NumPy](#), [SciPy](#), [matplotlib](#), [Pandas](#), [Scikit-Learn](#). You can add more by extending the `intersphinx_mapping` in your Sphinx's `conf.py`.

The pretty useful extension [autodoc](#) is activated by default and lets you include documentation from docstrings. Docstrings can be written in [Google style](#) (recommended!), [NumPy style](#) and [classical style](#).

1.1 License

The MIT License (MIT)

Copyright (c) 2019 Wai-Shing Luk

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.2 Contributors

- Wai-Shing Luk <luk036@gmail.com>

1.3 Changelog

1.3.1 Version 0.1

- Feature A added

- FIX: nasty bug #1729 fixed
- add your changes here!

1.4 n_sphere

1.4.1 n_sphere package

Submodules

n_sphere.discrep_2 module

`n_sphere.discrep_2.discrep_2(K, X)`
dispersion measure

Parameters

- `K` (`[type]`) – [description]
- `X` (`[type]`) – [description]

Returns dispersion

Return type float

n_sphere.halton_n module

class `n_sphere.halton_n.halton(b)`
Bases: `object`

Generate base-b Halton sequence

Parameters

- `n` (`int`) – [description]
- `b` (`[int]`) – sequence base, integer exceeding 1

Returns base-b low discrepancy sequence

Return type (`[float]`)

class `n_sphere.halton_n.halton_n(n, b)`
Bases: `object`

Generate base-b Halton sequence

Parameters

- `n` (`int`) – [description]
- `b` (`[int]`) – sequence base, integer exceeding 1

Returns base-b low discrepancy sequence

Return type (`[float]`)

n_sphere.skeleton module

This is a skeleton file that can serve as a starting point for a Python console script. To run this script uncomment the following lines in the [options.entry_points] section in setup.cfg:

```
console_scripts = fibonacci = n_sphere.skeleton:run
```

Then run *python setup.py install* which will install the command *fibonacci* inside your current environment. Besides console scripts, the header (i.e. until `_logger...`) of this file can also be used as template for Python modules.

Note: This skeleton file can be safely removed if not needed!

```
n_sphere.skeleton.fib(n)
    Fibonacci example function
```

Parameters `n` (*int*) – integer

Returns `n`-th Fibonacci number

Return type *int*

```
n_sphere.skeleton.main(args)
    Main entry point allowing external calls
```

Parameters `args` (*[str]*) – command line parameter list

```
n_sphere.skeleton.parse_args(args)
    Parse command line parameters
```

Parameters `args` (*[str]*) – command line parameters as list of strings

Returns command line parameters namespace

Return type *argparse.Namespace*

```
n_sphere.skeleton.run()
    Entry point for console_scripts
```

```
n_sphere.skeleton.setup_logging(loglevel)
    Setup basic logging
```

Parameters `loglevel` (*int*) – minimum loglevel for emitting messages

n_sphere.sphere module

```
class n_sphere.sphere.circle(base=2)
    Bases: object
```

Generate Circle Halton sequence 0,...,k

Parameters `k` (*int*) – maximum sequence index, non-negative integer

Keyword Arguments `base` (*int*) – [description] (default: {2})

Returns base-b low discrepancy sequence

Return type (*[float]*)

```
class n_sphere.sphere.sphere(b)
    Bases: object
```

Generate Sphere Halton sequence 0,...,k

Parameters `k` (*int*) – maximum sequence index, non-negative integer

Keyword Arguments `b` (*[int]*) – sequence base, integer exceeding 1

Returns base-b low discrepancy sequence

Return type ([float])

n_sphere.sphere3 module

class n_sphere.sphere3.sphere3(*b*)

Bases: object

Generate Sphere-3 Halton sequence

Parameters *k* ([int]) – maximum sequence index, non-negative integer

Keyword Arguments *b* ([int]) – sequence base, integer exceeding 1

Returns base-b low discrepancy sequence

Return type ([float])

class n_sphere.sphere3.sphere3_hopf(*b*)

Bases: object

sphere3_hopf Halton sequence INPUTS : *k* - maximum sequence index, non-negative integer

b - sequence base, integer exceeding 1

n_sphere.sphere_n module

class n_sphere.sphere_n.cylind_n(*n*, *b*)

Bases: object

Generate using cylindrical coordinate method

Parameters

- *k* ([int]) – maximum sequence index, non-negative integer
- *n* ([int]) – [description]
- *b* ([int]) – sequence base, integer exceeding 1

Returns base-b low discrepancy sequence

Return type ([float])

n_sphere.sphere_n.int_sin_power(*n*, *x*)

Evaluate integral $\sin^n(x) dx$

Parameters

- *n* ([int]) – power
- *x* ([float]) – [description]

Returns [description]

Return type float

class n_sphere.sphere_n.sphere_n(*n*, *b*)

Bases: object

Generate Sphere-3 Halton sequence

Parameters *k* ([int]) – maximum sequence index, non-negative integer

Keyword Arguments *b* ([int]) – sequence base, integer exceeding 1

Returns base-b low discrepancy sequence

Return type ([float])

n_sphere.vdcorput module

`n_sphere.vdcorput.vdc` (*n*, *base*=2)
[summary]

Parameters *n* (*int*) – number

Keyword Arguments *base* (*int*) – [description] (default: {2})

Returns [description]

Return type *int*

class `n_sphere.vdcorput.vdcorput` (*base*=2)
Bases: `object`

`n_sphere.vdcorput.vdcorput_co` (*k*, *base*=2)
[summary]

Parameters *k* (*int*) – number of points

Keyword Arguments *base* (*int*) – [description] (default: {2})

Returns [description]

Return type *int*

Module contents

CHAPTER 2

Indices and tables

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

n

- `n_sphere`, 7
- `n_sphere.discrep_2`, 4
- `n_sphere.halton_n`, 4
- `n_sphere.skeleton`, 5
- `n_sphere.sphere`, 5
- `n_sphere.sphere3`, 6
- `n_sphere.sphere_n`, 6
- `n_sphere.vdcorput`, 7

C

`circle` (class in `n_sphere.sphere`), 5
`cylin_n` (class in `n_sphere.sphere_n`), 6

D

`discrep_2()` (in module `n_sphere.discrep_2`), 4

F

`fib()` (in module `n_sphere.skeleton`), 5

H

`halton` (class in `n_sphere.halton_n`), 4
`halton_n` (class in `n_sphere.halton_n`), 4

I

`int_sin_power()` (in module `n_sphere.sphere_n`), 6

M

`main()` (in module `n_sphere.skeleton`), 5

N

`n_sphere` (module), 7
`n_sphere.discrep_2` (module), 4
`n_sphere.halton_n` (module), 4
`n_sphere.skeleton` (module), 5
`n_sphere.sphere` (module), 5
`n_sphere.sphere3` (module), 6
`n_sphere.sphere_n` (module), 6
`n_sphere.vdcorput` (module), 7

P

`parse_args()` (in module `n_sphere.skeleton`), 5

R

`run()` (in module `n_sphere.skeleton`), 5

S

`setup_logging()` (in module `n_sphere.skeleton`), 5

`sphere` (class in `n_sphere.sphere`), 5
`sphere3` (class in `n_sphere.sphere3`), 6
`sphere3_hopf` (class in `n_sphere.sphere3`), 6
`sphere_n` (class in `n_sphere.sphere_n`), 6

V

`vdc()` (in module `n_sphere.vdcorput`), 7
`vdcorput` (class in `n_sphere.vdcorput`), 7
`vdcorput_co()` (in module `n_sphere.vdcorput`), 7