tess Documentation

Release 0.1.3

Wendell Smith

Aug 03, 2018

Contents

CHAPTER 1

A 3D cell-based Voronoi library based on voro++

This library includes Python bindings, using Cython.

Code available on Github.

Documentation available at Read the Docs.

CHAPTER 2

Description

Tess is a library to calculate Voronoi (and Laguerre) tessellations in 3D and analyze their structure. The tessellation is calculated as a list of `Cell` objects, each of which can give information on its volume, centroid, number of faces, surface area, etc. The library is made with packings of spherical particles in mind, possibly with variable sizes.

2.1 voro++

The Tess library is a set of Python bindings to the Voro++ library. Voro++ provides all the algorithms, and Tess provides an easy to use interface to the voro++ library for Python, using Cython to do so.

Original work on voro++ by Chris H. Rycroft (UC Berkeley / Lawrence Berkeley Laboratory).

CHAPTER 3

Quick Start

3.1 Installation

To install, use `pip` (or `easy_install`):

```
pip install --user tess
```

Or to install from Github:

```
pip install --user git+git://github.com/wackywendell/tess@master
```

3.2 Usage

The first step is to create a *Container*:

```
>>> from tess import Container
>>> cntr = Container([[1,1,1], [2,2,2]], limits=(3,3,3), periodic=False)
```

A container is a list of *Cell* objects, representing Voronoi cells:

```
>>> [round(v.volume(), 3) for v in cntr]
[13.5, 13.5]
```

Cell objects have many methods. Here are a few:

```
>>> [v.pos for v in cntr]
[(1.0, 1.0, 1.0), (2.0, 2.0, 2.0)]

>>> [v.centroid() for v in cntr]
[(1.09375, 1.09375, 1.09375), (1.90625, 1.90625, 1.90625)]
```

(continues on next page)

(continued from previous page)

```
>>> [v.neighbors() for v in cntr]
[[-5, -2, -3, -1, -4, 1, -6], [0, -3, -6, -4, -5, -2, -1]]

>>> [v.face_areas() for v in cntr]
[[7.875, 1.125, 7.875, 7.875, 1.125, 11.691342951089922, 1.125],
 [11.691342951089922, 1.125, 7.875, 7.875, 1.125, 7.875, 1.125]]

>>> [v.normals() for v in cntr]
[[(0.0, 0.0, -1.0),
  (1.0, 0.0, 0.0),
  (0.0, -1.0, 0.0),
  (-1.0, 0.0, 0.0),
  (0.0, 1.0, 0.0),
  (0.5773502691896257, 0.5773502691896257, 0.5773502691896257),
  (0.0, 0.0, 1.0)],
 [(-0.5773502691896257, -0.5773502691896257, -0.5773502691896257),
  (-0.0, -1.0, -0.0),
  (0.0, 0.0, 1.0),
  (0.0, 1.0, -0.0),
  (0.0, 0.0, -1.0),
  (1.0, 0.0, -0.0),
  (-1.0, -0.0, -0.0)]]
```

See the Reference for more methods, or just use a Python interpreter or IPython notebook to find them on your own!

CHAPTER 4

Voro++ Copyright And Acknowledgments

4.1 Copyright Notice

4.2 Acknowledgments

This work (voro++) was supported by the Director, Office of Science, Computational and Technology Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

CHAPTER 5

Full Contents

5.1 Reference

This is a library to calculate Voronoi cells and access their information.

5.1.1 Basic Process

- **Create a *Container* object, using information about your system.**
  - a *Container* is a *list* of *Cell* objects
- Access the *Cell* methods to get information about them

Example

```
>>> from tess import Container
>>> c = Container([[1,1,1], [2,2,2]], limits=(3,3,3), periodic=False)
>>> [round(v.volume(), 3) for v in c]
[13.5, 13.5]
```

class tess.**Container**(*points*, *limits=1.0*, *periodic=False*, *radii=None*, *blocks=None*)

A container (*list*) of Voronoi cells.

This is the main entry point into the *tess* module. After creation, this will be a *list* of *Cell* objects.

The *Container* must be rectilinear, and can have solid boundary conditions, periodic boundary conditions, or a mix of the two.

```
>>> from tess import Container
>>> c = Container([[1,1,1], [2,2,2]], limits=(3,3,3), periodic=False)
>>> [round(v.volume(), 3) for v in c]
[13.5, 13.5]
```

Parameters

- **points** (iterable of iterable of *float*) – The coordinates of the points, size Nx3.
- **limits** (*float*, 3-tuple of float, or two 3-tuples of float) – The box limits. If given a float *L*, then the box limits are [0, 0, 0] to [L, L, L]. If given a 3-tuple *(Lx, Ly, Lz)*, limits are [0, 0, 0] to [Lx, Ly, Lz]. If given two 3-tuples *(x0, y0, z0), (x1, y1, z1)*, limits are [x0, y0, z0] to [x1, y1, z1].
- **periodic** (*bool* or 3-tuple of *bool*, optional) – Periodicity of the x, y, and z walls
- **radii** (iterable of *float*, optional) – for unequally sized particles, for generating a Laguerre transformation.

Returns A *list* of `Cell` objects

Return type *Container*

Notes

Voronoi Tesselation

A point $\vec{x}$ is part of a Voronoi cell i with nucleus $\vec{r}_i$ iff

$$|\vec{x} - \vec{r}_i|^2 < |\vec{x} - \vec{r}_j|^2 \, \forall j \neq i$$

Laguerre Tesselation, also known as *Radical Voronoi Tesselation*

A point $\vec{x}$ is part of a Laguerre cell i with nucleus $\vec{r}_i$ and radius R_i iff

$$|\vec{x} - \vec{r}_i|^2 - R_i^2 < |\vec{x} - \vec{r}_j|^2 - R_j^2 \forall j \neq i$$

get_widths()
Get the size of the box.

order(*l=6*, *local=False*, *weighted=True*)
Returns crystalline order parameter Q_l (such as Q_6).

Requires numpy and scipy.

Parameters

- **l** (*int, optional*) – Defines which Q_l you want (6 is standard, for detecting hexagonal lattices)
- **local** (*bool, optional*) – Calculate Local Q_6 (true) or Global Q_6
- **weighted** (*bool, optional*) – Whether or not to weight by area the faces of each polygonal side

Notes

For `local=False`, this calculates

$$Q_l = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} \left| \sum_{i=1}^{N_b} w_i Y_{lm}\left(\theta_i, \phi_i\right) \right|^2}$$

where:

N_b is the number of bonds

θ_i and ϕ_i are the angles of each bond i, in spherical coordinates

$Y_{lm}\left(\theta_i, \phi_i\right)$ is the spherical harmonic function

w_i is the weighting factor, either proportional to the area (for *weighted*) or all equal ($\frac{1}{N_b}$)

For `local=True`, this calculates

$$Q_{l,\text{local}} = \sum_{j=1}^{N} \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} \left| \sum_{i=1}^{n_b^j} w_i Y_{lm}\left(\theta_i, \phi_i\right) \right|^2}$$

where variables are as above, and each *cell* is weighted equally but each *bond* for each cell is weighted: $\sum_{i=1}^{n_b^j} w_i = 1$

Returns

Return type float

`tess.`**`cart_to_spher`**(*xyz*)

Converts 3D cartesian coordinates to the angular portion of spherical coordinates, (theta, phi).

Requires numpy.

Parameters **`xyz`** (*`array-like, Nx3`*) – Column 0: the "elevation" angle, 0 to π

Column 1: the "azimuthal" angle, 0 to 2π

Returns

Return type array, Nx2

`tess.`**`orderQ`**(*l*, *xyz*, *weights=1*)

Returns Q_l, for a given l (int) and a set of Cartesian coordinates xyz.

Requires numpy and scipy.

For global Q_6, use $l = 6$, and pass xyz of all the bonds.

For local Q_6, use $l = 6$, and the bonds have to be averaged slightly differently.

Parameters

- **`l`** (*`int`*) – The order of Q_l
- **`xyz`** (*`array-like Nx3`*) – The bond vectors $\vec{r}_j - \vec{r}_i$
- **`weights`** (*`array-like, optional`*) – How to weight the bonds; weighting by Voronoi face area is common.

Notes

This calculates

$$Q_l = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} \left| \sum_{i=1}^{N_b} w_i Y_{lm}\left(\theta_i, \phi_i\right) \right|^2}$$

where:

N_b is the number of bonds

θ_i and ϕ_i are the angles of each bond i, in spherical coordinates

$Y_{lm}(\theta_i, \phi_i)$ is the spherical harmonic function

w_i are the *weights*, defaulting to uniform: $(\frac{1}{N_b})$

class tess.**Cell**

A basic voronoi cell, usually created by *Container*.

A Voronoi cell has polygonal *faces*, connected by *edges* and *vertices*.

The various methods of a *Cell* allow access to the geometry and neighbor information.

__repr__

__str__

centroid()

face_areas()

A list of the areas of each face.

Returns

Return type A list of floats. Each inner list corresponds to a face.

face_freq_table()

face_perimeters()

face_vertices()

A list of the indices of the vertices of each face.

Returns

- *A list of lists of ints. Each inner list corresponds to a face, and each index corresponds*
- to a vertex from *vertices()*.

id

The id of the cell, which should generally correspond to its index in the Container.

max_radius_squared()

Maximum distance from pos() to outer edge of the cell (I think, see voro++ documentation.)

neighbors()

Return a list of the *neighbors* of the current *Cell*.

This is a list of indices, which correspond to the input points. The exception to this is the walls: walls are numbered -1 to -6, so an index less than 0 in the list of *neighbors()* indicates that a *Cell* is neighbors with a wall.

normals()

A list of the areas of each face.

Returns

Return type A list of 3-tuples of floats. Each tuple corresponds to a face.

number_of_edges()

number_of_faces()

pos

The position of the initial point around which this cell was created.

radius
The radius of the particle around which this cell was created.

Defaults to 0.

surface_area()

total_edge_distance()

vertex_orders()

vertices()
A list of all the locations of the vertices of each face.

Returns

Return type A list of 3-tuples of floats. Each tuple corresponds to a single vertex.

volume()
Cell volume

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index