Computational Algebraic Topology: Lecture 4

Alberto Paoluzzi

March 24, 2017

Simplicial complexes & Delaunay triangulations

Delaunay triangulations

Delaunay triangulation

In mathematics and computational geometry, a Delaunay triangulation for a set P of points in $\mathbb{E}^2$ is a triangulation $\mathcal{T}(P)$ such that no point in P is inside the circumcircle of any triangle in $\mathcal{T}(P)$

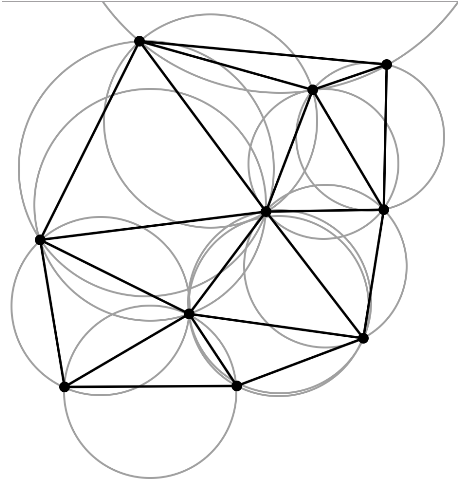
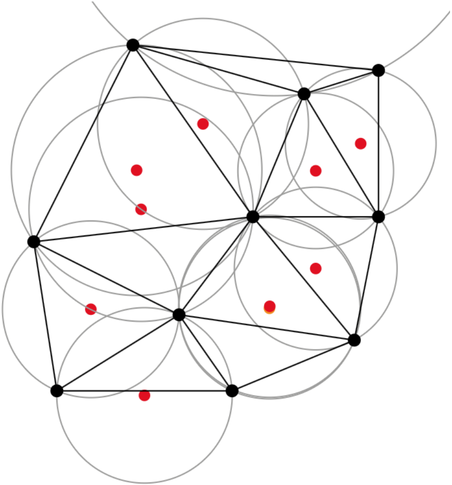


Figure 1: Delaunay triangulation

Dual vertices

The Delaunay triangulation of a discrete point set P in general position corresponds to the dual graph of the Voronoi diagram for P



Voronoi complex

- a Voronoi complex is a partitioning of a plane into a cellular complex based on distance to points in a discrete set P
- The regions are called Voronoi cells. The Voronoi diagram of a set of points is dual to its Delaunay triangulation.
- The 2-cells are convex

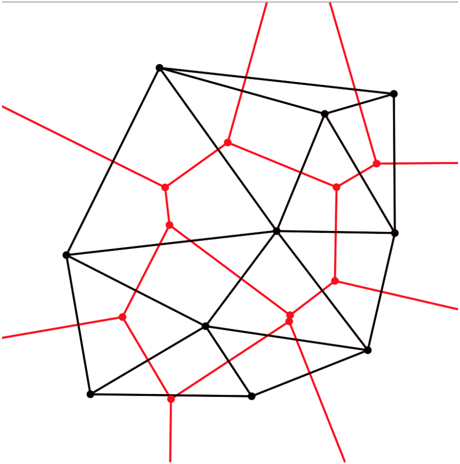


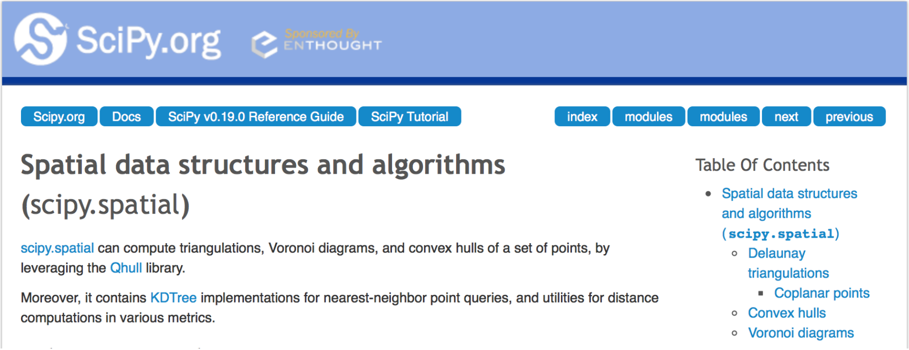
Figure 3: Delaunay triangulation

Properties of Delaunay triangulations (from Wikipedia)

- Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid sliver triangles.
- For a set of points on the same line there is no Delaunay triangulation (the notion of triangulation is degenerate for this case).
- For four or more points on the same circle (e.g., the vertices of a rectangle) the Delaunay triangulation is not unique:
  - each of the two possible triangulations that split the quadrangle into two triangles satisfies the "Delaunay condition", i.e., the requirement that the circumcircles of all triangles have empty interiors.
- By considering circumscribed spheres, the notion of Delaunay triangulation extends to three and higher dimensions.

Scipy.spatial package in Python

Spatial data structures and algorithms (scipy.spatial)



SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | SciPy v0.19.0 Reference Guide | SciPy Tutorial | index | modules | modules | next | previous

Spatial data structures and algorithms (scipy.spatial)

scipy.spatial can compute triangulations, Voronoi diagrams, and convex hulls of a set of points, by leveraging the Qhull library.

Moreover, it contains KDTree implementations for nearest-neighbor point queries, and utilities for distance computations in various metrics.

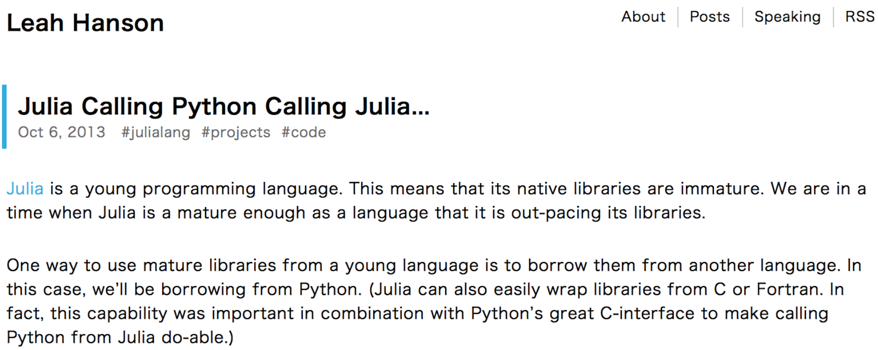
Table Of Contents

Assignment !!

: Follow step by step the tutorial . . .

Julia interface to Python

Julia Calling Python Calling Julia...



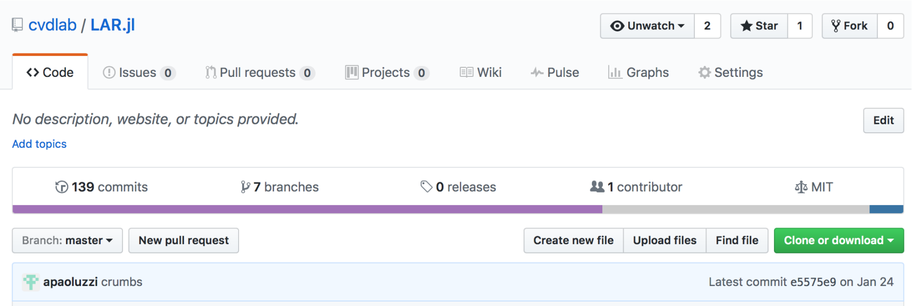
Using Python Libraries from Julia

```
using PyCall
@pyimport pylab

x = linspace(0,2*pi,1000);
y = sin(3*x + 4*cos(2*x));

pylab.plot(x, y; color="red", linewidth=2.0, linestyle="--")
pylab.show()
```

Using `Larlib` from Julia



Using Larlib from Julia

```
using LAR

jsonModel = """
    {"V" : [[5.0,0.0],[7.0,1.0],[9.0,0.0],[13.0,2.0],[15.0,4.0],[17.0,
    8.0],[14.0,9.0],[13.0,10.0],[11.0,11.0],[9.0,10.0],[5.0,9.0],[7.0,
    9.0],[3.0,8.0],[0.0,6.0],[2.0,3.0],[2.0,1.0],[8.0,3.0],[10.0,2.0],
    [13.0,4.0],[14.0,6.0],[13.0,7.0],[12.0,10.0],[11.0,9.0],[9.0,7.0],
    [7.0,7.0],[4.0,7.0],[2.0,6.0],[3.0,5.0],[4.0,2.0],[6.0,3.0],[11.0,
    4.0],[12.0,6.0],[12.0,7.0],[10.0,5.0],[8.0,5.0],[7.0,6.0],[5.0,5.0]],
    "FV" : [[0,1,16,28,29],[0,15,28],[1,2,17],[1,16,17,33],[2,3,17],
    [3,4,18,19],[3,17,18,30],[4,5,19],[5,6,19],[6,7,20,21,22,32],
    [6,19,20],[7,8,21],[8,9,21,22],[9,11,23,24],[9,22,23],
    [10,11,24,25],[10,12,25],[12,13,25,26],[13,14,27],[13,26,27],
    [14,15,28],[14,27,28,29,36],[16,29,34],[16,33,34],[17,30,33],
    [18,19,31],[18,30,31],[19,20,31,32],[22,23,32,33],[23,24,34,35],
    [23,33,34],[24,25,27,36],[24,35,36],[25,26,27],[29,34,35],
    [29,35,36],[30,31,32,33],[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]]}
""";
model = json2larmodel(jsonModel);
viewexploded(model.Verts,rebase(model.Lar.FV[1:end-1]))
```

Using Larlib from Julia

```
# Input of a LAR representation (2-complex in JSON format)
larDict = JSON.parse(jsonModel);
V = larDict["V"];
FV = larDict["FV"];

# extraction of facets (1-cells)
v,ev = p.larFacets((V,FV),2);

# visualization in Julia
viewexploded(v',(ev+1)')
viewLarIndices(v',(ev+1)')
```

Assignment !!

: look for `viewexploded` in `LAR.jl` and in `Larlib`

‘Facet and extrusion operations in Simple_X^n

Extraction of facets from a set of d-simplices

$$\partial\,\sigma^d = \sum_{k=0}^{d} (-1)^d \langle v_0, \ldots, v_{k-1}, v_{k+1}, \ldots, v_d \rangle$$

Implementation The `larSimplexFacets` function, for estraction of non-oriented $(d-1)$-facets of d-dimensional simplices, returns a list of d-tuples of integers, i.e. the input LAR representation of the topology of a cellular complex. The final steps are used to remove the duplicated facets, by transforming the sorted facets into a *set of strings*, so removing the duplicated elements.

⟨ Facets extraction from a set of simplices 8b ⟩ ≡

```
def larSimplexFacets(simplices):
    out = []
    d = len(simplices[0])
    for simplex in simplices:
        out += AA(sorted)([simplex[0:k]+simplex[k+1:d] for k in range(d)])
    out = set(AA(tuple)(out))
    return  sorted(out)
```

◇

Macro referenced in 10.

Extraction of facets from a set of d-simplices

```
def larSimplexFacets(simplices):
    """ Extraction of facets from a set of $d$-simplices"""
    out = []
    d = len(simplices[0])

    for simplex in simplices:
        out += AA(sorted)([simplex[0:k]+simplex[k+1:d] for k i

    out = set(AA(tuple)(out))
    return  sorted(out)
```

Assignment

In synthesis:

- Prepare a simplicial complex T using random point in Scipy
- Translate `larSimplexFacets` from Python to Julia
- Generate a Julia representation of 1-cells of T

Extrusion of a simplicial complex

```
def larExtrude1(model,pattern):
    V, FV = model
    d, m = len(FV[0]), len(pattern)
    coords = list(cumsum([0]+(AA(ABS)(pattern))))
    offset, outcells, rangelimit = len(V), [], d*m
    for cell in FV:
        tube = [v + k*offset for k in range(m+1) for v in cell]
        cellTube = [tube[k:k+d+1] for k in range(rangelimit)]
        outcells += [reshape(cellTube, newshape=(m,d,d+1)).tolist()]

    outcells = AA(CAT)(TRANS(outcells))
    cellGroups = [group for k,group in enumerate(outcells) if pattern[k]>0
    outVertices = [v+[z] for z in coords for v in V]
    outModel = outVertices, CAT(cellGroups)
    return outModel
```

Assignment

use Nuweb !!

1. Look for the meaning of `pattern` in `Larlib`
2. Translate in Julia
3. Generate

   two

   test examples

References