Scientific Computing WS 2018/2019

Lecture 12

Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

Recap

For more discussion of mesh generation, see J.R. Shewchuk: Lecture Notes on Delaunay Mesh Generation
http://web.mit.edu/ehliu/Public/ProjectX/Summer2005/delnotes.pdf

Meshes

- Regard boundary value problems for PDEs in a finite domain $\Omega \subset \mathbb{R}^d$
- Assume the domain is polygonal, its boundary $\partial\Omega$ is the union of a finite number of subsets of hyperplanes in $\mathbb{R}^n$ (line segments for $d = 2$, planar polygons for $d = 3$)
- A mesh (grid) is a subdivision Ω into a finite number of elementary closed (polygonal) subsets $T_1 \ldots T_M$.
- Mostly, the elementary shapes are triangles or quadrilaterals ($d = 2$) or tetrahedra or cuboids ($d = 3$)
- During this course: focus on $d = 2$, triangles
- Synonymous: mesh = grid = triangulation

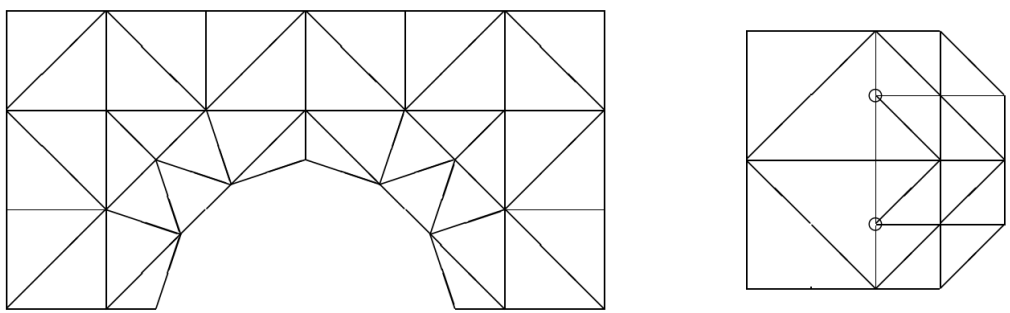
(FEM)-Admissible meshes

Definition: A grid is FEM-admissible if

(i) $\bar{\Omega} = \cup_{m=1}^{M} T_m$

(ii) If $T_m \cap T_n$ consists of exactly one point, then this point is a common vertex of T_m and T_n.

(iii) If for $m \neq n$, $T_m \cap T_n$ consists of more than one point, then $T_m \cap T_n$ is a common edge (or a common facet for $d = 3$) of T_m and T_n.



Source: Braess, FEM

Left: admissible mesh. Right: mesh with hanging nodes

Acute + weakly acute triangulations

Definition A triangulation of a domain Ω is

- acute, if all interior angles of all triangles are less than $\frac{\pi}{2}$,
- weakly acute, if all interior angles of all triangles are less than or equal to $\frac{\pi}{2}$.

Triangulation methods

- Geometrically most flexible
- Starting point for more general methods of subdivision into quadrilaterals
- Problem seems to be simple only at the first glance ...
- Here, we will discuss Delaunay triangulations, which have a number of interesting properties when it comes to PDE discretizations
  - J.R. Shewchuk: Lecture Notes on Delaunay Mesh Generation `http://web.mit.edu/ehliu/Public/ProjectX/Summer2005/delnotes.pdf`

Voronoi diagrams

After G. F. Voronoi, 1868-1908

Definition Let $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$. The set of points $H_{\mathbf{pq}} = \left\{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x} - \mathbf{p}|| \leq ||\mathbf{x} - \mathbf{q}||\right\}$ is the *half space* of points $\mathbf{x}$ closer to $\mathbf{p}$ than to $\mathbf{q}$.

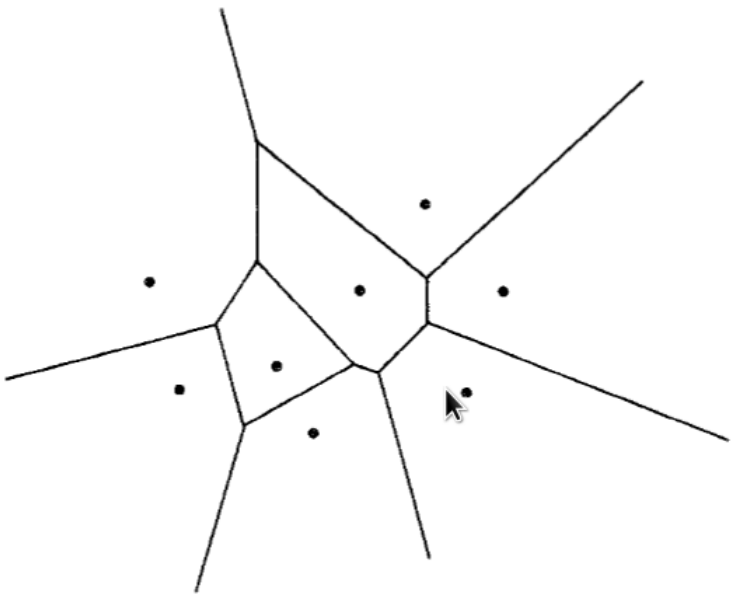
Definition Given a finite set of points $S \subset \mathbb{R}^d$, the *Voronoi region (Voronoi cell)* of a point $\mathbf{p} \in S$ is the set of points $\mathbf{x}$ closer to $\mathbf{p}$ than to any other point $\mathbf{q} \in S$:

$$V_{\mathbf{p}} = \left\{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x} - \mathbf{p}|| \leq ||\mathbf{x} - \mathbf{q}|| \, \forall \mathbf{q} \in S\right\}$$

The *Voronoi diagram* of S is the collection of the Voronoi regions of the points of S.

Voronoi diagrams II

- The Voronoi diagram subdivides the whole space into "nearest neigbor" regions
- Being intersections of half planes, the Voronoi regions are convex sets



Voronoi diagram of 8 points in the plane

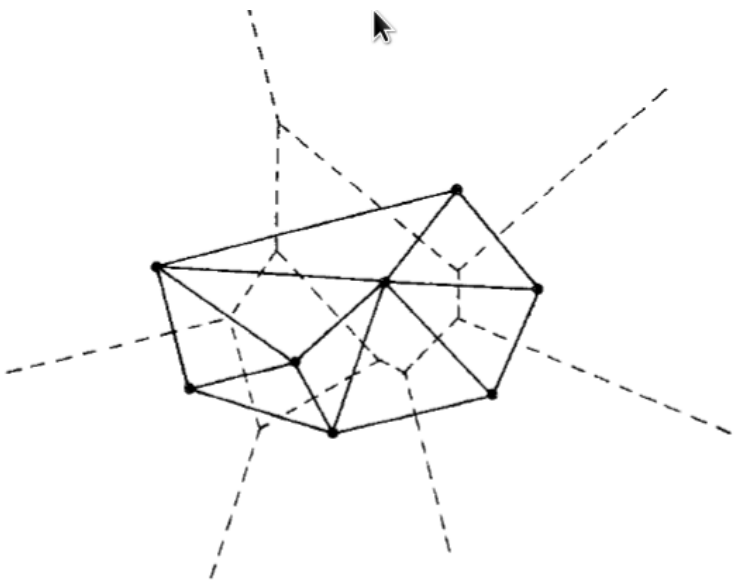
(H. Si)

Interactive example: http://homepages.loria.fr/BLevy/GEOGRAM/geogram_demo_Delaunay2d.html

Delaunay triangulation

After B.N. Delaunay (Delone), 1890-1980

- Assume that the points of S are in *general position*, i.e. no $d+2$ points of S are on one sphere (in 2D: no 4 points on one circle)
- Connect each pair of points whose Voronoi regions share a common edge with a line
- $\Rightarrow$ *Delaunay triangulation* of the convex hull of S



Delaunay triangulation of the convex hull of 8 points in the plane

(H. Si)

Delaunay triangulation II

- The circumsphere (circumcircle in 2D) of a d-dimensional simplex is the unique sphere containing all vertices of the simplex
- The circumball (circumdisc in 2D) of a simplex is the unique (open) ball which has the circumsphere of the simplex as boundary

Definition A triangulation of the convex hull of a point set S has the *Delaunay property* if each simplex (triangle) of the triangulation is Delaunay, i.e. its circumsphere (circumcircle) is empty wrt. S, i.e. it does not contain any points of S.

- The Delaunay triangulation of a point set S, where all points are in general position is unique
- Otherwise there is an ambiguity - if e.g. 4 points are one circle, there are two ways to connect them resulting in Delaunay triangles

Edge flips and locally Delaunay edges (2D only)

- For any two triangles **abc** and **adb** sharing a common edge **ab**, there is the *edge flip* operation which reconnects the points in such a way that two new triangles emerge: **adc** and **cdb**.
- An edge of a triangulation is locally Delaunay if it either belongs to exactly one triangle, or if it belongs to two triangles, and their respective circumdisks do not contain the points opposite wrt. the edge
- If an edge is locally Delaunay and belongs to two triangles, the sum of the angles opposite to this edge is less or equal to π.
- If all edges of a triangulation of the convex hull of S are locally Delaunay, then the triangulation is the Delaunay triangulation
- If an edge is not locally Delaunay and belongs to two triangles, the edge emerging from the corresponding edge flip will be locally Delaunay

Edge flip algorithm (Lawson)

Input: A stack L of edges of a given triangulation of S;
while $L \neq \emptyset$ **do**
    pop an edge **ab** from L;
    if **ab** *is not locally Delaunay* **then**
        flip **ab** to **cd**;
        push edges **ac**, **cb**, **db**, **da** onto L;
    end
end

- This algorithm is known to terminate. After termination, all edges will be locally Delaunay, so the output is the Delaunay triangulation of S.
- Among all triangulations of a finite point set S, the Delaunay triangulation maximises the minimum angle
- All triangulations of S are connected via a flip graph

Radomized incremental flip algorithm (2D only)

- Create Delaunay triangulation of point set S by inserting points one after another, and creating the Delaunay triangulation of the emerging subset of S using the flip algorithm
- Estimated complexity: $O(n \log n)$
- In 3D, there is no simple flip algorithm, generalizations are active research subject

Triangulations of finite domains

- So far, we discussed triangulations of point sets, but in practice, we need triangulations of domains
- Create Delaunay triangulation of point set, "Intersect" with domain

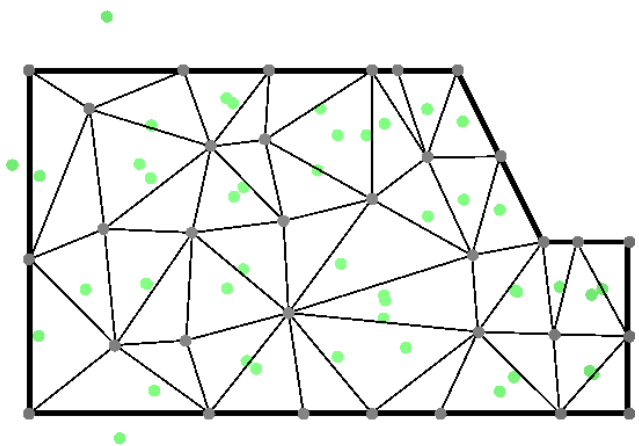
Boundary conforming Delaunay triangulations

Definition: An admissible triangulation of a polygonal Domain $\Omega \subset \mathbb{R}^d$ has the boundary conforming Delaunay property if

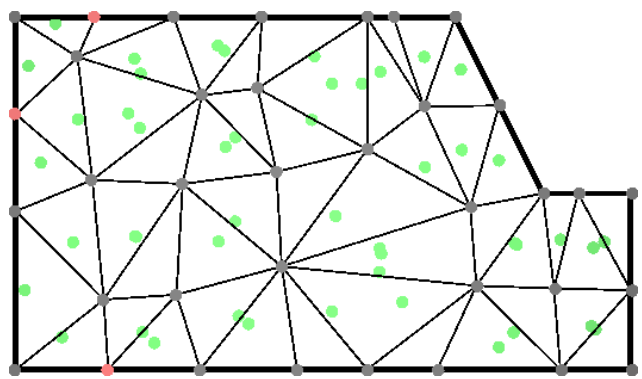
(i) All simplices are Delaunay

(ii) All boundary simplices (edges in 2D, facets in 3d) have the Gabriel property, i.e. their minimal circumdisks are empty

- Equivalent definition in 2D: sum of angles opposite to interior edges $\leq \pi$, angle opposite to boundary edge $\leq \frac{\pi}{2}$
- Creation of boundary conforming Delaunay triangulation description may involve insertion of Steiner points at the boundary



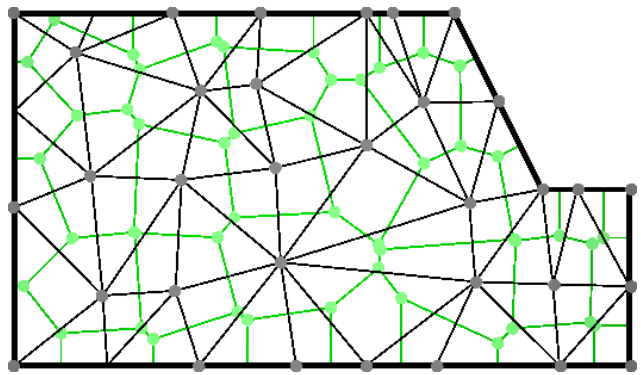
Delaunay grid of Ω



Boundary conforming Delaunay grid of Ω

Domain blendend Voronoi cells

- For Boundary conforming Delaunay triangulations, the intersection of the Voronoi diagram with the domain yields a well defined dual subdivision



Boundary conforming Delaunay triangulations II

- Weakly acute triangulations are boundary conforming Delaunay, but not vice versa!
- Working with weakly acute triangulations for general polygonal domains is unrealistic, especially in 3D
- For boundary conforming Delaunay triangulations of polygonal domains there are algoritms with mathematical termination proofs valid in many relevant cases
- Code examples:
  - 2D: Triangle by J.R.Shewchuk
    `https://www.cs.cmu.edu/~quake/triangle.html`
  - 3D: TetGen by H. Si `http://tetgen.org`
- Features:
  - polygonal geometry description
  - automatic insertion of points according to given mesh size criteria
  - accounting for interior boundaries
  - local mesh size control for a priori refinement
  - quality control
  - standalone executable & library

Further mesh generation approaches

(Most of them lose Delaunay property)

- Advancing front: create mesh of boundary, "grow" triangles from boundary to interior implemented e.g. in netgen by J. Schöberl `https://sourceforge.net/projects/netgen-mesher/`
- Quadtree/octree: place points on quadtree/octree hierachy and triangulate
- Mesh improvement: equilibrate element sizes + quality by iteratively modifying point locations
- ... active research topic with many open questions, unfortunately not exactly mainstream ...

Virtual Machine

- Install VirtualBox `https://www.virtualbox.org/` on your system (available for Linux, Windows, Mac)
- Download the virtual machine `debian-numcxx-v01.ova` from the course homepage (Attention: 2.5GB!)
- Import it into VirtualBox and start
- Log in as 'unknown' with password 'numcxx'
- Debian system, similar to that in UNIX pool
- CodeBlocks, g++, gedit, numcxx, vtk, vtkfig are installed.
- numcxx and vtkfig updates can performed from bitbucket repo, see corresponding README on the desktop
- Data transfer with your computer through shared folders or ssh login. Problems with drag&drop, unfortunately

The Triangle mesh generator

- Free for non-commercial use
- By J.R.Shewchuk, Berkeley
- Distributed with numcxx

```
$ triangle --help
```

- Accompanied by `showme` program to visualize grids
- Triangle as a standalone program is controlled by certain flags, reads input from disk, writes output to disk
- Triangle as a library is controlled by the same flags, but takes input as `double*` and `int*` arrays, and creates output in the same form

Convex hulls of point sets with Triangle

```
$ triangle --help
...
-c  Creates segments on the convex hull of the triangulation.  If you
    are triangulating a vertex set, this switch causes a .poly file to
    be written, containing all edges of the convex hull.
...
-v  Outputs the Voronoi diagram associated with the triangulation.
...
-V  Verbose:  Gives detailed information about what Triangle is doing.
...
$ triangle -Vvc hello.node
$ ls hello.*
hello.node      # input
hello.1.node    # output, nodes of the Delaunay triangulation
hello.1.ele     # output, triangles of the Delaunay triangulation
hello.1.poly    # output, Edges of the convex hull of the point set
hello.1.v.node  # output, nodes of Voronoi diagram
hello.1.v.edge  # output, edges of Voronoi diagram
```

- `.node` files contain lists of points (possibly with attributes)
- `.poly` files contain lists of polygons (possibly with attributes)
- `.edge` files contain lists of edges (possibly with attributes)
- `.ele` files contain lists of triangles (possibly with attributes)

Discretization of a domain with Triangle

```
$ triangle --help
-p  Reads a Planar Straight Line Graph (.poly file), which can specify
    vertices, segments, holes, regional attributes, and regional area
    constraints.  Generates a constrained Delaunay triangulation (CDT)
    fitting the input; or, if -s, -q, -a, or -u is used, a conforming
    constrained Delaunay triangulation (CCDT).
...
-a  Imposes a maximum triangle area.  If a number follows the ‘a’, no
    triangle is generated whose area is larger than that number.
...
-q  Quality mesh generation by Delaunay refinement.  Adds vertices
    to the mesh to ensure that all angles are between 20 and 140 degrees.
    An alternative bound on the minimum angle, replacing 20 degrees, may
    be specified after the ‘q’.
...
 -D  Conforming Delaunay triangulation:  use this switch if you want to
     ensure that all the triangles in the mesh are Delaunay, and not
     merely constrained Delaunay; or if you want to ensure that all the
     Voronoi vertices lie within the triangulation.  (Some finite volume
     methods have this requirement.)
...
$ triangle -Vp hello.poly
$ ls hello.*
hello.1.node    # output, nodes of the triangulation
hello.1.ele     # output, triangles of the triangulation
hello.1.poly    # output, boundary edges
```

Triangle in numcxx

- In addition to nodes, triangles and boundary edges we need a region attribute for each triangle (for different data in different regions) and a boundary region attribute for each boundary edge (for different boundary conditions)
- Triangle handles these attributes
- `class numcxx::Geometry` $\rightarrow$ `class numcxx::SimpleGrid`
- `class numcxx::SimpleGrid` is ready for use in finite element and finite volume methods
  - points
  - cells
  - boundary faces
  - cell and boundary regions

Partial Differential Equations

Differential operators: notations

Given: domain $\Omega \subset \mathbb{R}^d$.

- Dot product: for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{d} x_i y_i$
- Bounded domain $\Omega \subset \mathbb{R}^d$, with piecewise smooth boundary
- Scalar function $u : \Omega \to \mathbb{R}$
- Vector function $\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} : \Omega \to \mathbb{R}^d$
- Write $\partial_i u = \frac{\partial u}{\partial x_i}$
- For a multiindex $\alpha = (\alpha_1 \dots \alpha_d)$, let
  - $|\alpha| = \alpha_1 + \cdots + \alpha_d$
  - $\partial^\alpha u = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}$

Basic Differential operators

- Gradient of scalar function $u : \Omega \to \mathbb{R}$

$$\mathrm{grad} = \nabla = \begin{pmatrix} \partial_1 \\ \vdots \\ \partial_d \end{pmatrix} : \; u \mapsto \nabla u = \begin{pmatrix} \partial_1 u \\ \vdots \\ \partial_d u \end{pmatrix}$$

- Divergence of vector function $\mathbf{v} = \Omega \to \mathbb{R}^d$

$$\mathrm{div} = \nabla \cdot : \; \mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} \mapsto \nabla \cdot \mathbf{v} = \partial_1 v_1 + \cdots + \partial_d v_d$$

- Laplace operator of scalar function $u : \Omega \to \mathbb{R}$

$$\Delta = \mathrm{div} \cdot \mathrm{grad} = \nabla \cdot \nabla : \; u \mapsto \Delta u = \partial_{11} u + \cdots + \partial_{dd} u$$

Lipschitz domains

Definition:

- Let $D \subset \mathbb{R}^n$. A function $f : D \to \mathbb{R}^m$ is called *Lipschitz continuous* if there exists $c > 0$ such that $\|f(x) - f(y)\| \leq c\|x - y\|$
- A hypersurface in $\mathbb{R}^n$ is a *graph* if for some k it can be represented as

$$x_k = f(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)$$

  defined on some domain $D \subset \mathbb{R}^{n-1}$
- A domain $\Omega \subset \mathbb{R}^n$ is a *Lipschitz domain* if for all $x \in \partial\Omega$, there exists a neigborhood of x on $\partial\Omega$ which can be represented as the graph of a Lipschitz continuous function.

Corollaries

- Boundaries of Lipschitz domains are continuous
- Boundaries of Lipschitz domains have no cusps
  (e.g. the graph of $y = \sqrt{|x|}$ has a cusp at $x = 0$)
- Polygonal domains are Lipschitz
- Standard PDE calculus happens in Lipschitz domains

Divergence theorem (Gauss' theorem)

Theorem: Let Ω be a bounded Lipschitz domain and $\mathbf{v} : \Omega \to \mathbb{R}^d$ be a continuously differentiable vector function. Let $\mathbf{n}$ be the outward normal to Ω. Then,

$$\int_\Omega \nabla \cdot \mathbf{v} \, d\mathbf{x} = \int_{\partial\Omega} \mathbf{v} \cdot \mathbf{n} \, ds$$

□

Species balance over an REV

- Let $u(\mathbf{x}, t) : \Omega \times [0, T] \to \mathbb{R}$ be the local amount of some species.
- Assume *representative elementary volume (REV)* $\omega \subset \Omega$
- Subinterval in time $(t_0, t1) \subset (0, T)$
- $-\delta \nabla u \cdot \mathbf{n}$ describes the flux of these species trough $\partial\omega$, where δ is some transfer coefficient
- Let $f(\mathbf{x}, t)$ be some local source of species. Then the flux through the boundary is balanced by the change of the amount of species in ω and the source strength:

$$0 = \int_\omega (u(\mathbf{x}, t_1) - u(\mathbf{x}, t_0))\, d\mathbf{x} - \int_{t_0}^{t_1} \int_{\partial\omega} \delta \nabla u \cdot \mathbf{n}\, ds\, dt - \int_{t_0}^{t_1} \int_\omega f(\mathbf{x}, t)\, ds$$

- Using Gauss' theorem, rewrite this as

$$0 = \int_{t_0}^{t_1} \int_\omega \partial_t u(\mathbf{x}, t)\, d\mathbf{x}\, dt - \int_{t_0}^{t_1} \int_\omega \nabla \cdot (\delta \nabla u)\, d\mathbf{x}\, dt - \int_{t_0}^{t_1} \int_\omega f(\mathbf{x}, t)\, ds$$

- True for all $\omega \subset \Omega$, $(t_0, t1) \subset (0, T) \Rightarrow$ parabolic second order PDE

$$\partial_t u(x, t) - \nabla \cdot (\delta \nabla u(x, t)) = f(x, t)$$

No lecture on Tue, Dec. 4!