Introduction to mesh generation (in Matlab)

By Allan P. Engsig-Karup

Overview

- Introduction to mesh generation
- Introduction to DistMesh for Matlab

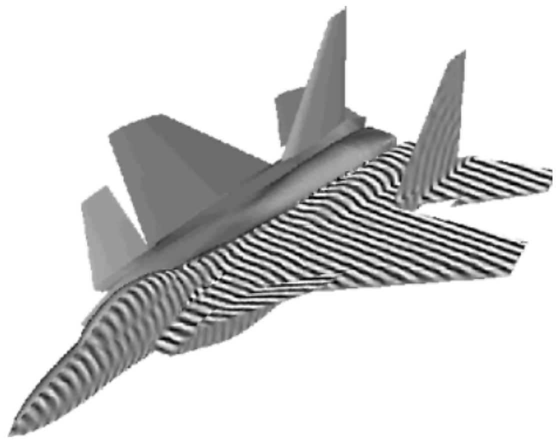
- Goal: Introduce you to DistMesh for use with DG-FEM based models.

Why do we need a mesh?

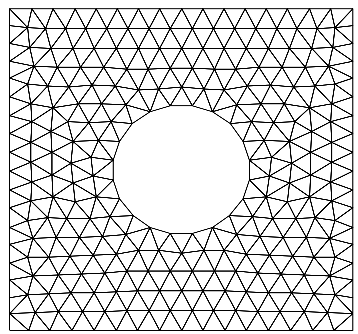
- We need to represent the (usually finite) physical domain in some way discretely for numerical computations.
- In sub domain methods, e.g. Finite volume or FEM methods, it is possible to *independently* consider the *problem solution procedure* and *mesh generation* as two distinct problems.
- This is very convenient if we want to solve more than one problem governed by the same PDEs!

What defines a mesh?

- Here we define a *mesh* as a discrete representation Ω_h of some spatial *domain* or *topology* Ω.
- A mesh can be sub divided into *K* smaller *non-overlapping* sub domains Ω_h^k such that $\Omega_h = \bigcup_{k=1}^{K} \Omega_h^k$.
- Mesh *generation* can be a demanding and non-trivial task. E.g. for complex geometries or objects.
- Unstructured triangular meshes have good support for representing *complex domains* (or geometries) and *mesh adaption* (coarsening/refinement).



F-15. From: www.USEMe.org



Example. Triangulation.

What defines a mesh?

- A mesh can be completely defined in terms of (unique) *vertices* and a *mesh element table* (*triangulation*).
- For the purpose of specifying appropriate boundary conditions we may for convenience use a *boundary type table*.
- Simple meshes can be created *manually* by hand. However, *automatic mesh generation* is generally faster and more efficient, although may require some user input for handling complex meshes.

- Note: Mesh data can conveniently be stored for reuse several times.

Mesh generators available?

- Lots of standard mesh generators available! These generators can be used to solve a given mesh generation problem. (but may require a translation script)

- An example of a free software distribution for generating *unstructured and triangular meshes* is *DistMesh (*Matlab).

Introduction to DistMesh for Matlab

- Persson, P.-O. and Strang, G. 2004 *A simple mesh generator in Matlab.* SIAM Review.
  Download scripts at:
  http://www-math.mit.edu/~persson/mesh/index.html

- A simple algorithm that combines a physical principle of *force equilibrium* in a truss structure with a mathematical representation of the geometry using *signed distance functions*.

Introduction to DistMesh for Matlab

- Algorithm (Conceptual);

  Step 1. Define a domain using signed distance functions.
  Step 2. Distribute a set of nodes interior to the domain.
  Step 3. Move interior nodes to obtain force equilibirum.
  Step 4. Apply terminate criterion when all nodes are fixed in space.

- Post-processing steps (Preparation);

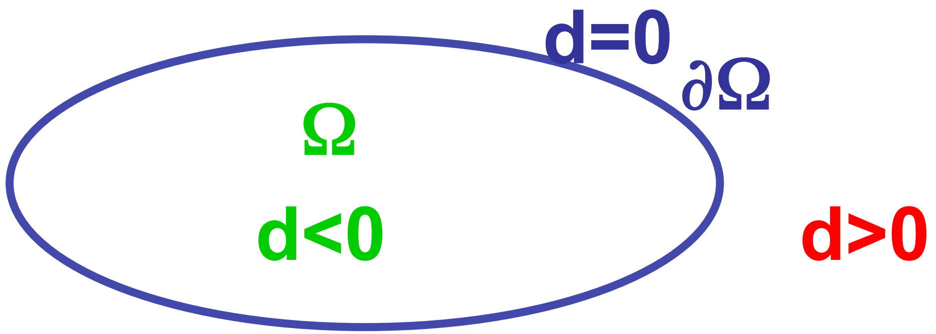
  Step 5. Validate output!
  Step 6. Reorder element vertices to be defined anti-clockwise for use with DG-FEM.
  Step 7. Setup boundary table.
  Step 8. Store mesh for reuse.

Introduction to DistMesh for Matlab

Signed distance function, d(x);

$$d(x) = \begin{cases} < 0 & , x \in \Omega \quad \text{(interior)} \\ 0 & , x \in \partial\Omega \quad \text{(boundary)} \\ > 0 & , x \notin \Omega \quad \text{(exterior)} \end{cases}$$

Define metric using an appropriate norm. E.g. The usual Euclidian metric.



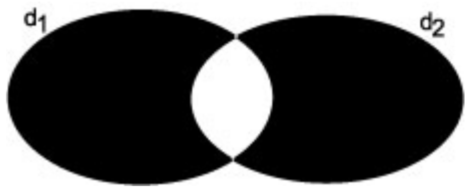
Introduction to DistMesh for Matlab

- Combine geometries defined by distance functions using the *Union*, *difference* and *intersection* operations (set theory);

Union:

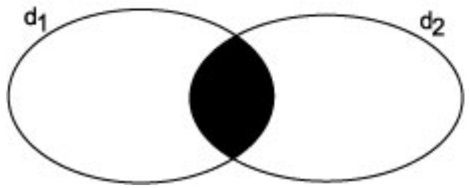
$$\min(d_1(x), d_2(x))$$

Difference:



$$\max(d_1(x), -d_2(x)) \cup$$
$$\max(-d_1(x), d_2(x))$$

Intersection;

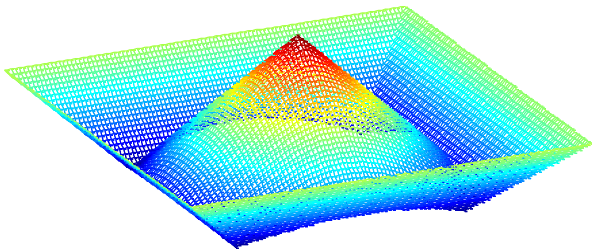


$$\max(d_1(x), d_2(x))$$

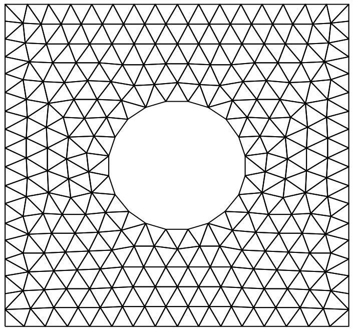
Introduction to DistMesh for Matlab

Example 1. Create a uniform mesh using DistMesh.

Square with hole



Visualized distance function



Mesh

Using DistMesh (in Matlab) in only 3 lines of code:

```
>> fd=inline('ddiff(drectangle(p,-1,1,-1,1),dcircle(p,0,0,0.4))','p');
>> pfix = [-1,-1;-1,1;1,-1;1,1];
>> [p,t] = distmesh2d(fd,@huniform,0.125,[-1,-1;1,1],pfix);
```

Introduction to DistMesh for Matlab

DistMesh output; *(two tables)*

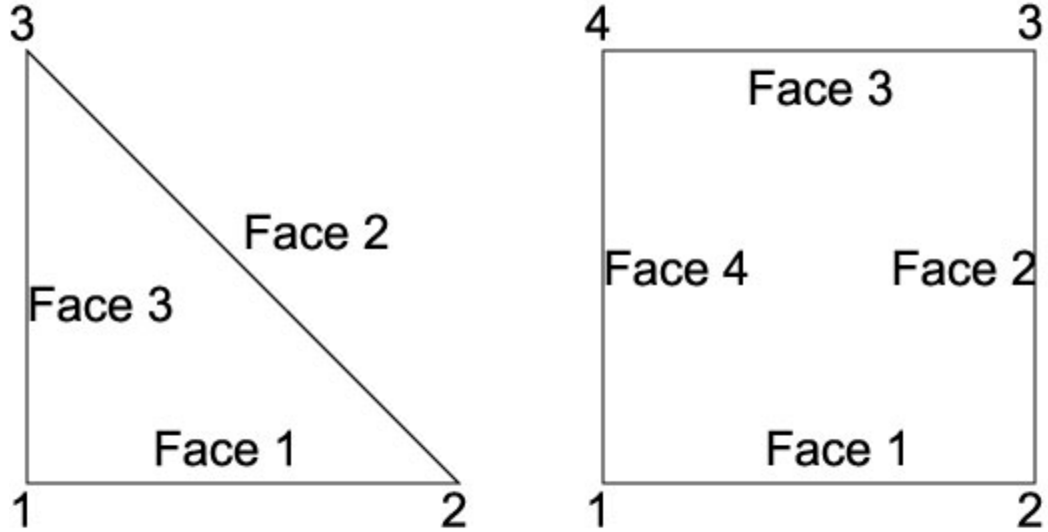
p	*Unique vertice coordinates*
t	*Element to Vertice table* *(not reordered automatically by DistMesh)*

From this we can determine, e.g.

```
>> K=size(t,1);            %Number of elements
>> Nv=size(p,1);           %Number of vertices in mesh
>> Nfaces=size(t,2);       %Number of faces/element
>> VX = p(:,1);            %Vertice x-coordinates
>> VY = p(:,2);            %Vertice y-coordinates
>> EToV = t;               %Element to Vertice table
```

Introduction to DistMesh for Matlab

DG-FEM convention for standard element definitions;

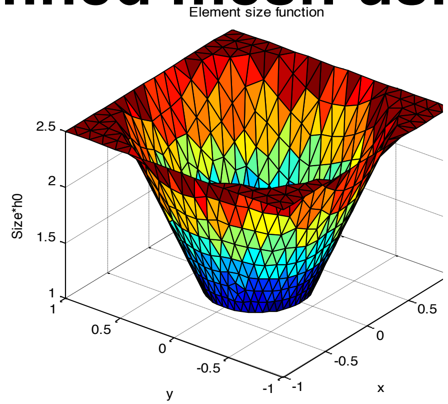


- Vertices are numbered *anti-clockwise*.
- Faces are numbered anti-clockwise with the first face beeing the one that connects the first two vertices.

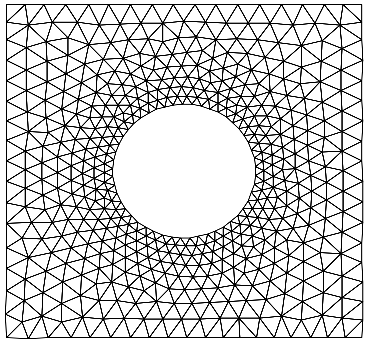
Introduction to DistMesh for Matlab

Example 2. Create a refined mesh using DistMesh.

Square with hole



Visualized element size function

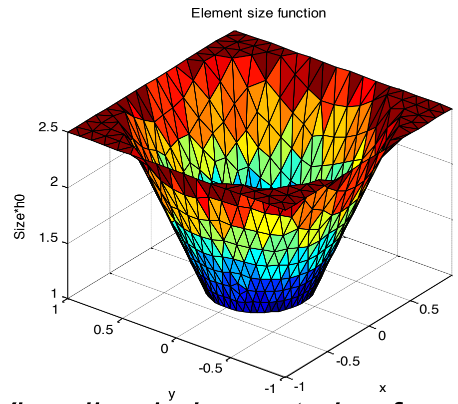


Mesh

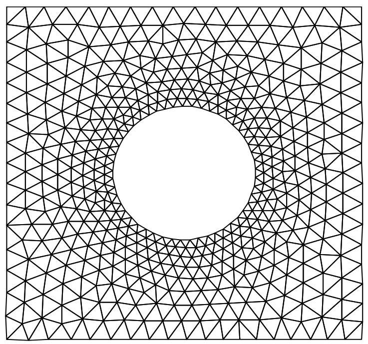
```
>> fd = inline('ddiff(drectangle(p,-1,1,-1,1),dcircle(p,0,0,0.4))','p');
>> pfix = [-1,-1;-1,1;1,-1;1,1];
>> fh = inline(['min( sqrt( p(:,1).^2 + p(:,2).^2 ) , 1 )'],'p');
>> [p,t] = distmesh2d(fd,fh,0.125/2.5,[-1,-1;1,1],pfix);
```

Introduction to DistMesh for Matlab

Element size function in DistMesh;



Visualized element size function



Mesh

From former example;

```
>> fh = inline(['min( sqrt( p(:,1).^2 + p(:,2).^2 ) , 1 )'],'p');
>> [p,t] = distmesh2d(fd,fh,h0,[-1,-1;1,1],pfix);
```

- Function *fh* Defines *relative* sizes of elements in final mesh. (*fh=constant* result in uniform distribution)
- *The initial characteristic* size of the elements *is h0.*
- In final distribution, the characteristic size of the *smallest* elements in the mesh will be approx. *h0*;

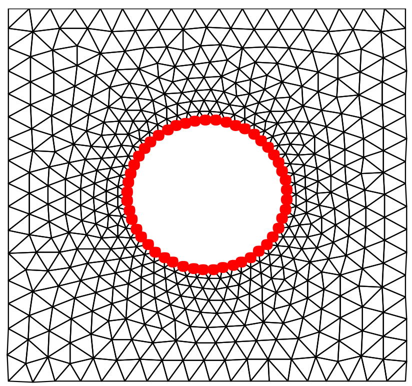
Introduction to DistMesh for Matlab

Example 3. Selecting boundary nodes.

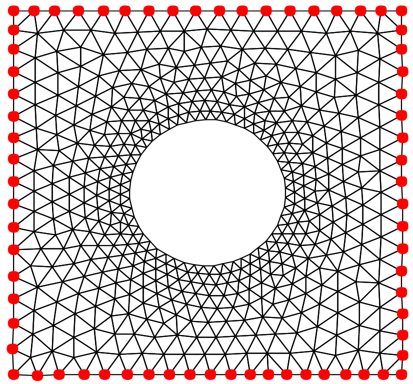
Square with hole

Nodes can be selected using distance functions;

|d| = 0   or |d| < tol



Inner boundary nodes



Outer boundary nodes

```
>> fdInner = inline('dcircle(p,0,0,0.4)','p');
>> nodesInner = find(abs(fdInner([p]))<1e-3);
>> fdOuter = inline('drectangle(p,-1,1,-1,1)','p');
>> nodesOuter = find(abs(fdOuter([p]))<1e-3);
>> nodesB  = find(abs(fd([p]))<1e-3);
```

Introduction to DistMesh for Matlab

Example 4. Updating boundary table.

Square with hole

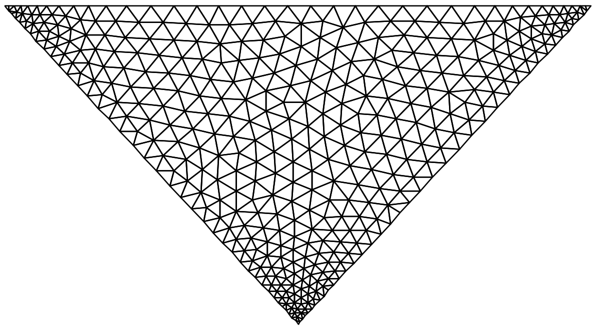
```
>> BCcode = 99;
>> BCType = zeros(size(EToV')); % empty BCType table
>> BCType = CorrectBCTable(K,EToV,BCType,nodesB,BCcode);
```

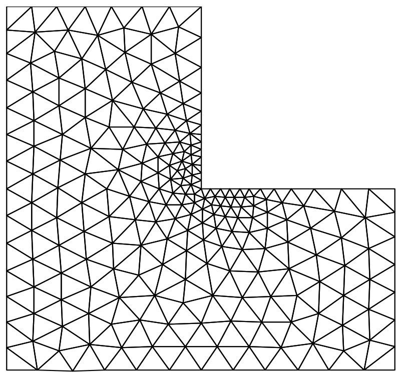
The BCType boundary table can be used to create different maps (see the script BuildBCMaps2D.m, Section 6.4 in the textbook) for imposing different types of boundary conditions.

Final remarks

These notes together with example scripts for DistMesh can be found at my webpage:

http://www.imm.dtu.dk/~apek/





Feel free to contact me at *apek@imm.dtu.dk*