Mesh generation via triangle – short tutorial

Dirk Peschka

This short tutorial works best under Linux, but as well on MacOS and Windows if compilers are installed. Execution of the following commands require a terminal/shell.

Installation

Download triangle (a zip-file) from the page

`http://www.cs.cmu.edu/~quake/triangle.html`

and put it into a new folder and extract the zip container. You should get the following files

```
A.poly  makefile  README  showme.c  triangle.c
triangle.h  triangle.zip  tricall.c
```

In order to compile these into executable files, type `make` (on Linux systems). This should create the two executable files `triangle` and `showme`. On MacOS or Windows system just compile the `triangle.c` with your favorite C compiler. Compilation of `showme` might not work, depending on the configuration of your system.

The program `triangle` is for mesh generation, whereas `showme` with plot a generated mesh.

Congratulations, installation finished.

Mesh generation

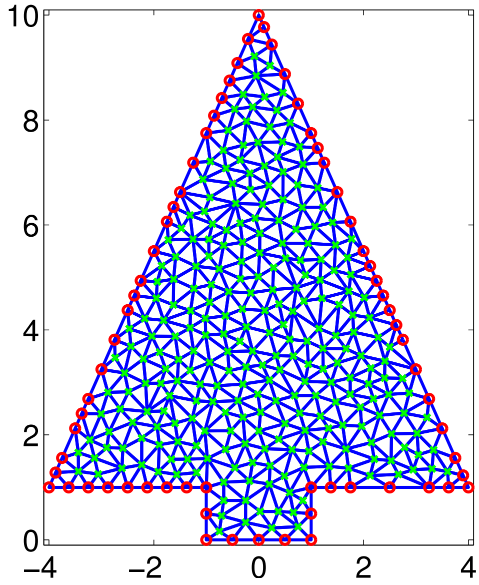
In contrast to MATLABs `delaunay` triangle is a versatile "Two-Dimensional Quality Mesh Generator and Delaunay Triangulator." It can easily handle non-convex domains, returns boundary edges and boundary markers, features subparametric elements (for experts) etc.

For mesh generation the normal procedure is that you first create a `.poly` file, from which you then generate your triangulation. If you call `triangle -h` you get an extensive list of options and a documentation. In particular input and output formats `.poly`,`.node` and `.ele` are specified. It is useful to read and understand the specification of the `.poly` input file.

Along with this documention, you should find the file `readtriamesh.m` which contains the `function [x,y,npoint,nelement,e2,idp,ide] = readtria(fname)`. The function reads a simple triangle mesh into a set of variables:

- `x,y`: array of points of size `npoint`.
- `npoint`: number of vertices in the decomposition/mesh.
- `nelement`: number of elements in the decomposition/mesh.
- `e2`: adjacency information of the mesh (integer array of size `nelement`×3 or `nelement`×6 for subparametric elements)
- `idp`: point attributes of size `npoint`
- `ide`: element attributes of size `nelement`

Example 1: X-mas tree



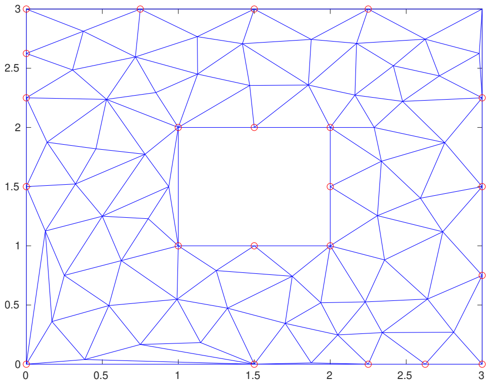
The simple Christmas tree below was generated using the command

```
./triangle -qa0.1 xmas.poly
```

and can be loaded using `fname = 'xmas.1'`. The input files looks like this

```
# simple Xmas tree, Dirk Peschka
# 7 points, 2 dimension, 0 attributes, 1 boundary marker
7 2 0 1
# tree vertices, vertex number, x position, y position, marker
1     1  0    0
2     1  1    0
3     4  1    0
4     0 10    0
5    -4  1    0
6    -1  1    0
7    -1  0    0
# 7 segments, 1 boundary marker
7 1
# segment number, node 1, node 2, segment marker
1    1 2    1
2    2 3    1
3    3 4    1
4    4 5    1
5    5 6    1
6    6 7    1
7    7 1    1
# no hole
0
```

Example 2: Box



The simple box was generated using the command

```
./triangle -a0.1 box_v1.poly
```

and can be loaded using `fname = 'box_v1.1'`. The input files looks like this

```
# A box with eight vertices in 2D, no attributes, one boundary marker.
    8 2 0 1
     # Outer box has these vertices:
     1   0 0   0
     2   0 3   0
     3   3 0   0
     4   3 3   33     # A special marker for this vertex.
     # Inner square has these vertices:
     5   1 1   0
     6   1 2   0
     7   2 1   0
     8   2 2   0
    # 8 segments with boundary markers.
    8 1
     1   1 2   0.
     2   2 4   0
     3   4 3   0
     4   3 1   0
     5   5 6   0
     6   6 8   0
     7   8 7   0
     8   7 5   0
    # One hole in the middle of the inner square.
    1
     1   1.5 1.5
```

This mesh in loaded into MATLAB and plotted using the following commands:

```
[x,y,npoint,nelement,e2,idp,ide] = readtria('box_v1.1');
triplot(e2,x,y);
hold on;
plot(x(idp==1),y(idp==1),'ro');
hold off
```

Have fun creating your own triangulations for fluid problems!