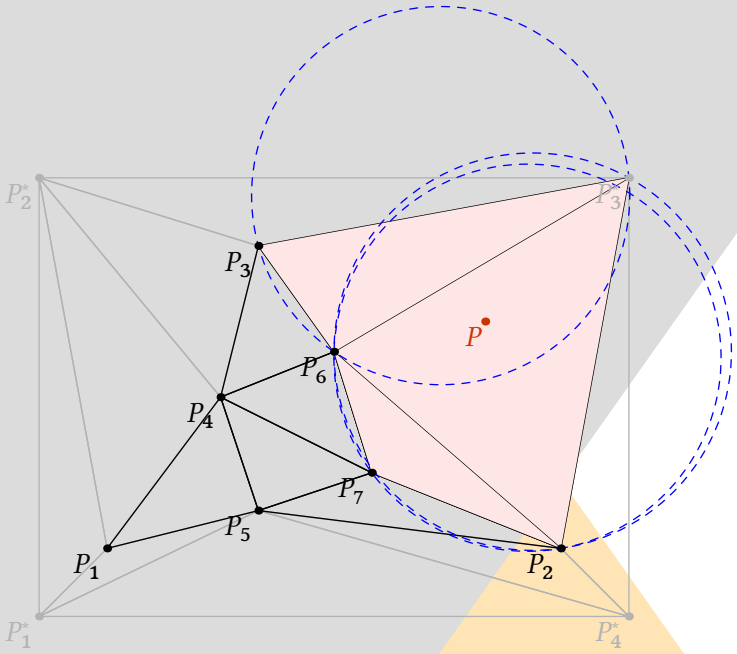
# mp-gdd

compute and draw meshes with LuaLaTeX




**Contributor**
Maxime Chupin

Version 0.1, 25 novembre 2016
http://melusine.eu.org/syracuse/G/delaunay/

# luamesh: compute and draw meshes with LuaLaTeX

Maxime Chupin <mc@melusine.eu.org>

November 25, 2016

The package luamesh allows to compute and draw 2D triangulation of Delaunay. The algorithm is written with lua, and depending of the choice of the "engine", the draw is done by MetaPost (with luamplib) or by tikz.

The Delaunay triangulation algorithm is the Bowyer and Watson algorithm. Several macros are provided to draw the global mesh, the set of points, a particular step of the algorithm.

## 1 Installation

### 1.1 With Linux

To install luamesh with TeXlive, you have to create the local texmf directory in your home.

```
user $> mkdir ~/texmf
```

Then we have to files to place in the correct directories. First, the luamesh.sty file must be in the directory:

~/texmf/tex/latex/luamesh/

and secondly, the luamesh.lua must be in the directory:

~/texmf/scripts/luamesh/

Once you have done this, luamesh can be included in your document with

```
\usepackage{luamesh}
```

### 1.2 Dependencies

This package is built upon two main packages to draw the triangulations :

1. `luamplib` to use MetaPost via the LuaTEX library `mplib`;
2. or `tikz`.

We will see how to choose between these two *drawing engines*.

Moreover, the following packages are necessary:

1. `xkeyval` to manage the optional arguments;
2. `xcolor` to use colors (needed by `luamplib`);
3. `ifthen` to help the programming with TEX.

## 2 The Basic Macros

If you want to use this package, you must compile your document with `lualatex`:

```
user $>   lualatex mylatexfile.tex
```

Let us recall that this package provides macros to draw two dimensional triangulations (or meshes).

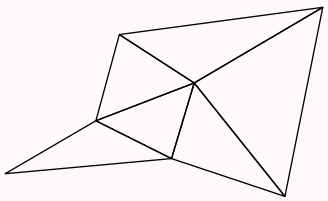
### 2.1 Draw a Complete Mesh

`\buildMeshBW[`⟨*options*⟩`]{`⟨*list of points*⟩ `or` ⟨*file name*⟩`}`

This macro produce the Delaunay triangulation (using the Bowyer and Watson algorithm) of the given ⟨*list of points*⟩. The list of points must be given in the following way :

```
(x1,y1);(x2,y2);(x3,y3);...;(xn,yn)
```

```
\buildMeshBW{(0.3,0.3);(1.5,1);(4,0);(4.5,2.5);(1.81,2.14);(2.5,0.5);(2.8,1.5)}
```



### 2.1.1 The Options

There are several options to customize the drawing.

`mode = int (default) or ext:` this option allows to use either the previously described set of point in the argument, or a file, containing, line by line (2 columns), the points. Such a file looks like :

```
x1  y1
x2  y2
x3  y3
...
xn yn
```

`bbox = none (default) or show:` this option allows to draw the added points to form a *bounding box*[1] and the corresponding triangulation. By default, these triangles are removed at the end of the algorithm.

`color =` ⟨***value***⟩ `(default: black):` The color of the drawing.

`colorBbox =` ⟨***value***⟩ `(default: black):` The color of the drawing for the elements (points and triangles) belonging to the bounding box.

`print = none (default) or points:` To label the vertices of the triangulations with an adding dot.

`meshpoint =` ⟨***value***⟩ `(default: P):` The letter(s) used to label the vertices of the triangulation. It is include in the math mode delimiters `$...$`. The bounding box points are labeled with a star exponent, and numbered from 1 to 4.

`tikz (boolean, default:false):` By default, this boolean is set to `false`, and MetaPost (with `luamplib`) is used to draw the picture. With this option, it is `tikz` the *drawing engine*.

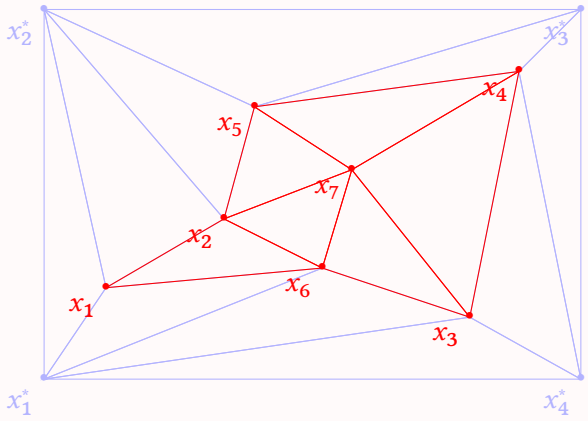
`scale =` ⟨***value***⟩ `(default: 1cm):` The scale option defines the scale at which the picture is draw (the same for the two axis). It must contain the unit of length (cm, pt, etc.).

To illustrate the options, let us show you an example. We consider a file `mesh.txt`:

```
0.3    0.3
1.5    1
4      0
4.5    2.5
1.81   2.14
2.5    0.5
2.8    1.5
```

[1] The bounding box is defined by four points place at 15% around the box defined by $(x_{\min}, y_{\min})$, $(x_{\min}, y_{\max})$, $(x_{\max}, y_{\max})$, and $(x_{\min}, y_{\max})$.

```
\buildMeshBW[%
tikz,
mode = ext,
bbox = show,
color = red,
colorBbox = blue!30,
print = points,
meshpoint = x,
scale = 1.3cm,
]{mesh.txt}
```



The drawing engine is not here very relevant. But it is useful to understand how the drawing is made. However, the engine will make sens for the so called *inc* macros (section 3), where we will be allowed to add code before and after the generated one by `luamesh`.

### 2.2 Draw the Set of Points

`\drawPointsMesh[`⟨*options*⟩`]{`⟨*list of points*⟩ `or` ⟨*file name*⟩`}`

### 2.3 Draw a Step of the Bowyer and Watson Algorithm

`\meshAddPointBW[`⟨*options*⟩`]{`⟨*list of points*⟩ `or` ⟨*file name*⟩`}{`⟨*point*⟩ `or` ⟨*number of line*⟩`}`

## 3 The *inc* Macros

## 4 Gallery of Examples