

## Chapter 1

# Existing (unstructured) meshes: Wrappers to third-party mesh generators

`oomph-lib` does not provide its own unstructured mesh generator but has several mesh classes that generate unstructured meshes from the output of third-party unstructured mesh generators.

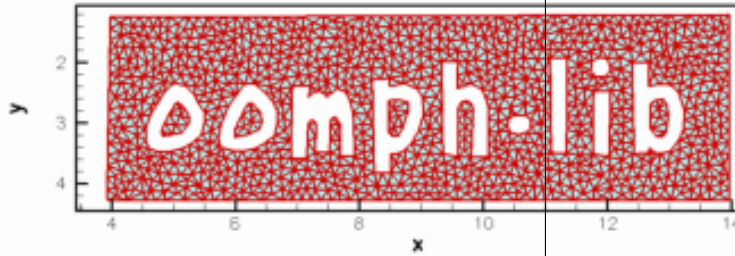
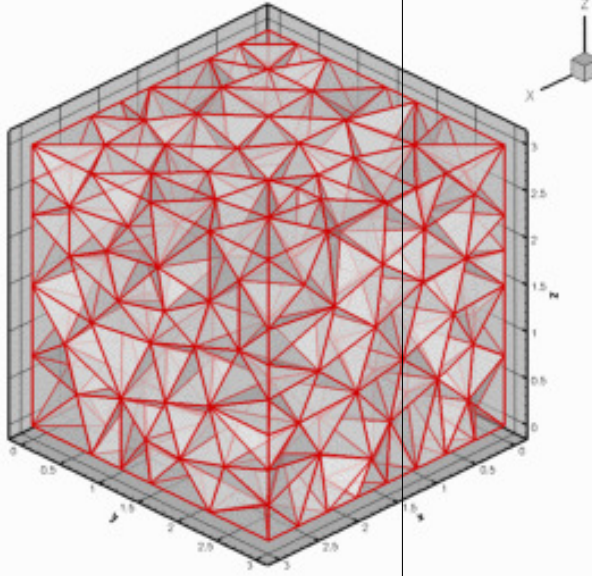
### Notes:

1. The unstructured tet and triangle meshes listed below can **not** be used with `oomph-lib`'s mesh adaptation or node-update procedures. A suitably fine mesh has to be generated offline by the third-party mesh generator. If required, node-updates (in response to changes in the domain boundaries) have to be performed manually.
2. For some element types, the mesh generation process is not particularly efficient (yet!). A suitable warning message is issued in such cases.
3. Since the third-party mesh generators tend to triangulate the domain with simplex elements, curvilinear boundaries are not resolved more accurately by using higher-order elements unless some post-processing is performed.
4. The meshes have not been tested as extensively as `oomph-lib`'s structured meshes, described [elsewhere](#).

---

### 1.1 Mesh list

---

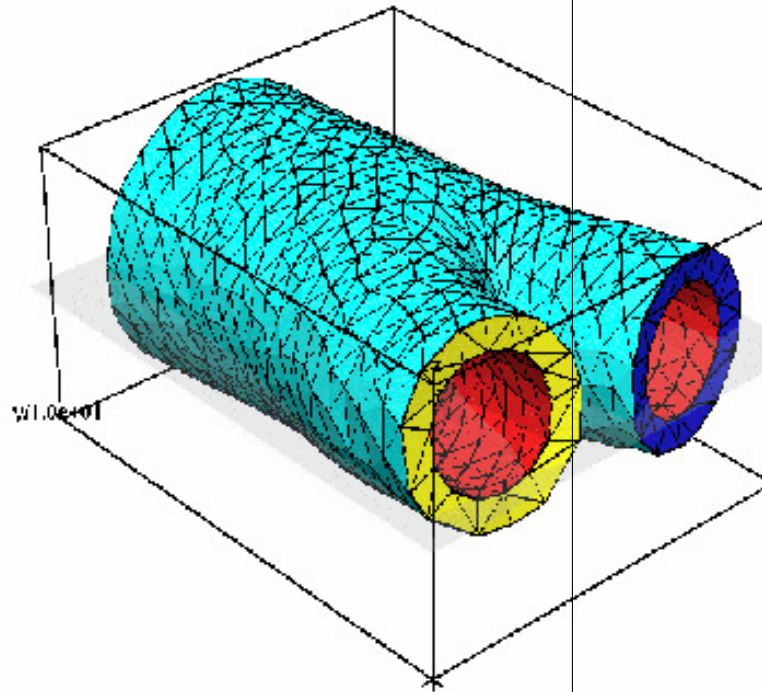
Mesh	Representative Mesh plot
<p><b>TriangleMesh&lt;ELEMENT&gt;</b></p> <ul style="list-style-type: none"> <li>This class creates <code>oomph-lib</code> meshes based on the output from <a href="#">J.R.Shewchuk's</a> Delaunay mesh generator <code>Triangle</code></li> <li>The mesh can be used with all <code>FiniteElement</code>s that are derived from the geometric finite element <code>TElement&lt;2, NNODE_1D&gt;</code>.</li> </ul> <p><b>Example driver codes:</b></p> <ul style="list-style-type: none"> <li>The use of <code>Triangle</code> and the <code>TriangleElement</code> class are explained in a <a href="#">separate tutorial</a>.</li> <li>In <a href="#">another tutorial</a> we demonstrate how the code <code>fig2poly.cc</code> may be used to generate input files for <code>Triangle</code> based on the output from the open-source drawing program <code>xfig</code>.</li> </ul>	
<p><b>TetgenMesh&lt;ELEMENT&gt;</b></p> <ul style="list-style-type: none"> <li>This class creates <code>oomph-lib</code> meshes based on the output from <a href="#">Hang Si's</a> open-source mesh generator <code>Tetgen</code>.</li> <li>The mesh can be used with all <code>FiniteElement</code>s that are derived from the geometric finite element <code>TElement&lt;3, NNODE_1D&gt;</code>.</li> </ul> <p><b>Example driver codes:</b></p> <ul style="list-style-type: none"> <li>The use of <code>Tetgen</code> and the <code>TetgenElement</code> class are explained in a <a href="#">separate tutorial</a>.</li> </ul>	

### Generating meshes from medical scans with VMTK

- We provide the option to generate tetgen-based meshes for physiological fluid-structure interaction problems, using the **Vascular Modeling Toolkit (VMTK)**.

#### Example driver codes and tutorials:

- We provide a **separate tutorial** that shows how to generate oomph-lib meshes from medical images.
- The methodology is used in the following driver codes:
  - The inflation of a blood vessel.
  - Finite Reynolds number flow through a (rigid) iliac bifurcation.
  - Finite Reynolds number flow through an elastic iliac bifurcation.



### GeompackQuadMesh<ELEMENT>

- This class creates oomph-lib meshes based on the output from Barry Joe's mesh generator **Geompack++**, available as freeware at <http://members.shaw.ca/bjoe/>.
- The mesh can be used with all **Finite Elements** that are derived from the geometric finite element **QElement<2, 2>**.

#### Example driver codes:

- The use of **Geompack++** and the **GeompackQuadMesh** class are explained in a **separate tutorial**.



---

## 1.2 PDF file

A [pdf version](#) of this document is available.