1 Introduction

This is a schematic description of the C++ code performing mesh reduction for a non-planar spatial regression analysis involving a penalty term.
The document is structured in the following way. In Section 2 we illustrate the contents of each folder of the Git repository `mesh-simplification`. Then in section **??** we present the interface of the main classes and functions defined in the code. A brief description for each code entity is given. To help the reader go through the rather complicated structure of the library, hyperlinks among different sections will be provided.

2 Repo structure

The code has been organized in the following folders:

`/bin` folder storing the binaries, i.e. `cpp` files with `main`;

`/doc` heterogeneous documentation regarding the code;

`/mesh` set of test grids, useful for numerical simulations;

`bustPredicated` suite of geometric standard tests for a grid; they are said *robust* because they are designed to properly work even with numerically dangerous objects, e.g. badly-conditioned matrices;

`/src` the header and source files constituing the library;

`/unitTests` suite of short tests aiming at validating the code.

Within the folder `/src`, one may find the following subfolders:

`/core` definition of base geometric entities, e.g. point;

`/file` to manage I/O; note that post-processing is carried out in ParaView;

`/intersec` check if two geometric elements, e.g. two triangles, intersect;

`/utility` various geometric tests, e.g. check if a point falls within a polygon;

`/geometry` definition of the data structured to store a grid; the main files it contains are:

- `connect2D.hpp`: it implements the connections between elements; remember that an element is characterized by:
  1. ID,
  2. point(s),
  3. color, representing the material;
- `mesh2D.hpp`: data structure to store a two-dimensional grid;
- `tricky2D.hpp`: performs different operations on the grid *without* altering the node-element connections;

`/doctor` perform the following operations on the grid:

- smoothing,
- edge swapping,
- edge splitting,
- edge collapsing.

Note that this operations modify the node-element connections;

`/meshOperation` compute the cost functional for each edge, whose making the `doctor` aware of which edges have to contract; the main file is `simplification2d`.