CODE: Mesh Offset

Mesh normal offset with Iso-surface from FEM solution

Author: Abhijith B N

abhijithbn@gmail.com

Table of Contents

1
SECTION

Introduction

This code is created as an attempt to solve a diffrential equation over a triangular mesh surface to obtain iso-surface such that a parallel mesh can be created through offset.

1.1 Downloading Instruction

The source code can be downloaded from github repository: https://github.com/abhnar/codemesh

1.2 Compilation Instructions

Windows

Visual Studio Build Tools

```
\> cl /EHsc /I include src\*.cpp /std:c++17 /Fe:cod.exe /O2 /openmp
```

SYS2 MINGW64

```
$ g++ -I include src/*.cpp -fopenmp -O2
```

Linux (Tested under windows WSL

```
$ g++ -I include src/*.cpp -fopenmp -O2
```

1.2.1 Usage

```
$ cod meshfile.stl d [clean]
```

d : distance to offset [clean] : cleans invalid triangle after offset.
*Caution!!! Not optimised yet. Does not scale for large mesh.

Output

output.stl - Mesh that is offset by distance d

1.2.2 Yet to Implement

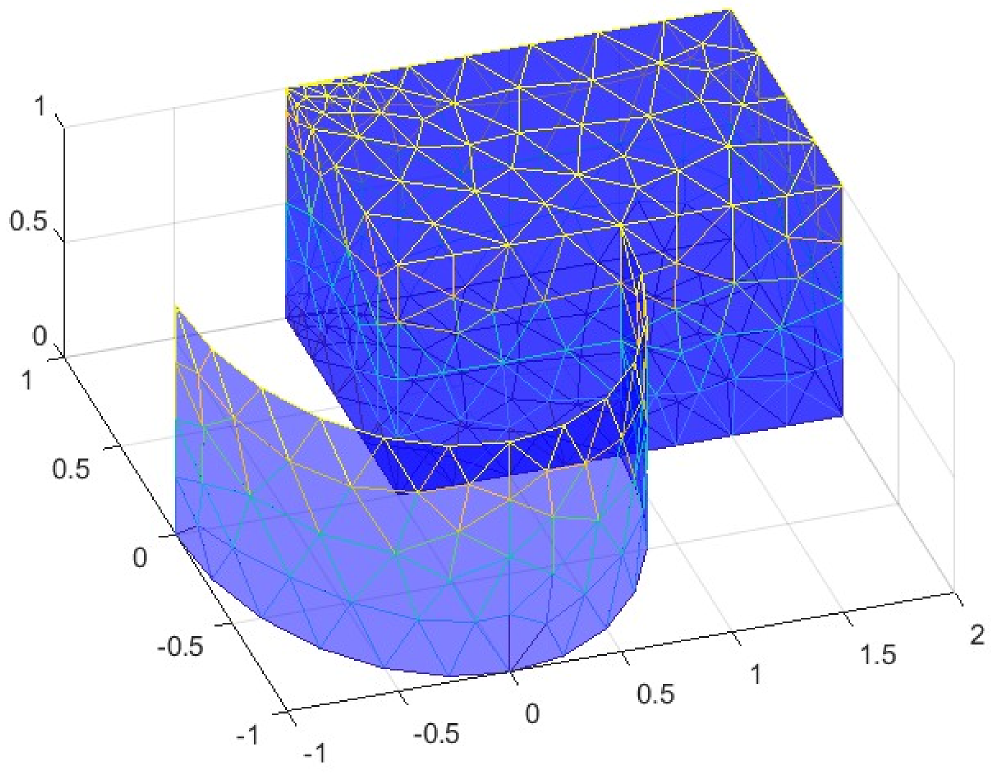
Constrained Delauny Triangulation, to fill the hole created by deleted triangles.

Example

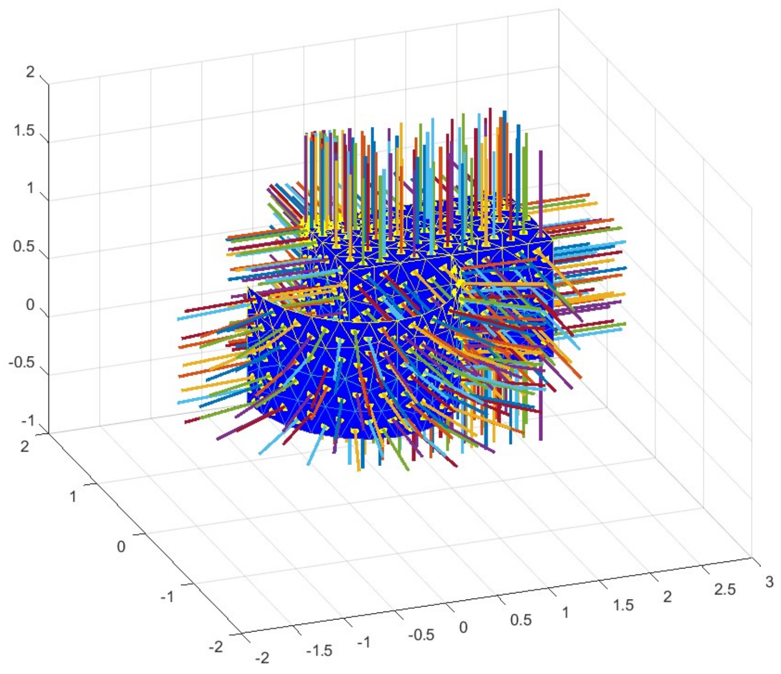
```
$ cod mesh\curve.stl 0.5
```

```
$ cod mesh\curve.stl 0.5 clean
```

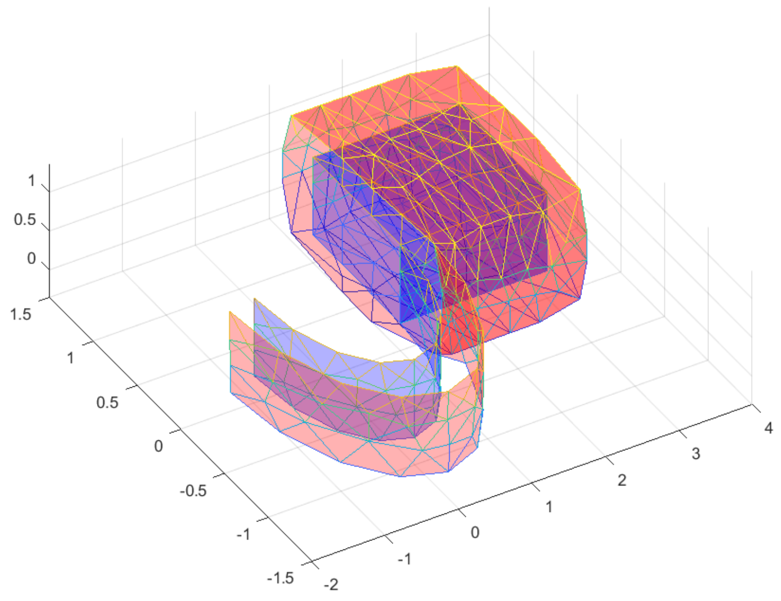
1.2.3 Original Mesh



1.2.4 Mesh Normlas



1.2.5 Mesh (blue) and Offset (red)



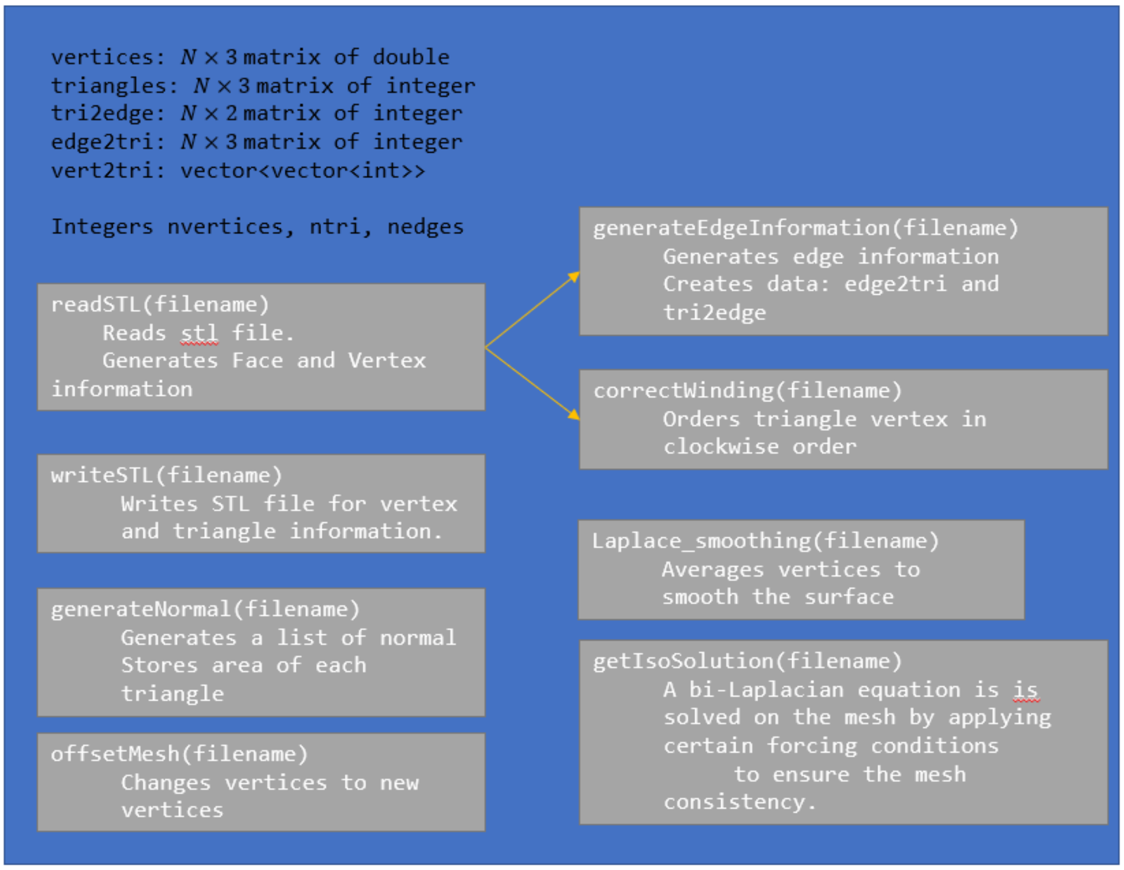
2

SECTION

Design Blocks

2.1 Data and basic functions of *Mesh* class

Class: Mesh



```
vertices: N × 3 matrix of double
triangles: N × 3 matrix of integer
tri2edge: N × 2 matrix of integer
edge2tri: N × 3 matrix of integer
vert2tri: vector<vector<int>>

Integers nvertices, ntri, nedges
```

```
readSTL(filename)
    Reads stl file.
    Generates Face and Vertex
information
```

```
writeSTL(filename)
     Writes STL file for vertex
     and triangle information.
```

```
generateNormal(filename)
     Generates a list of normal
     Stores area of each
     triangle
```

```
offsetMesh(filename)
     Changes vertices to new
     vertices
```

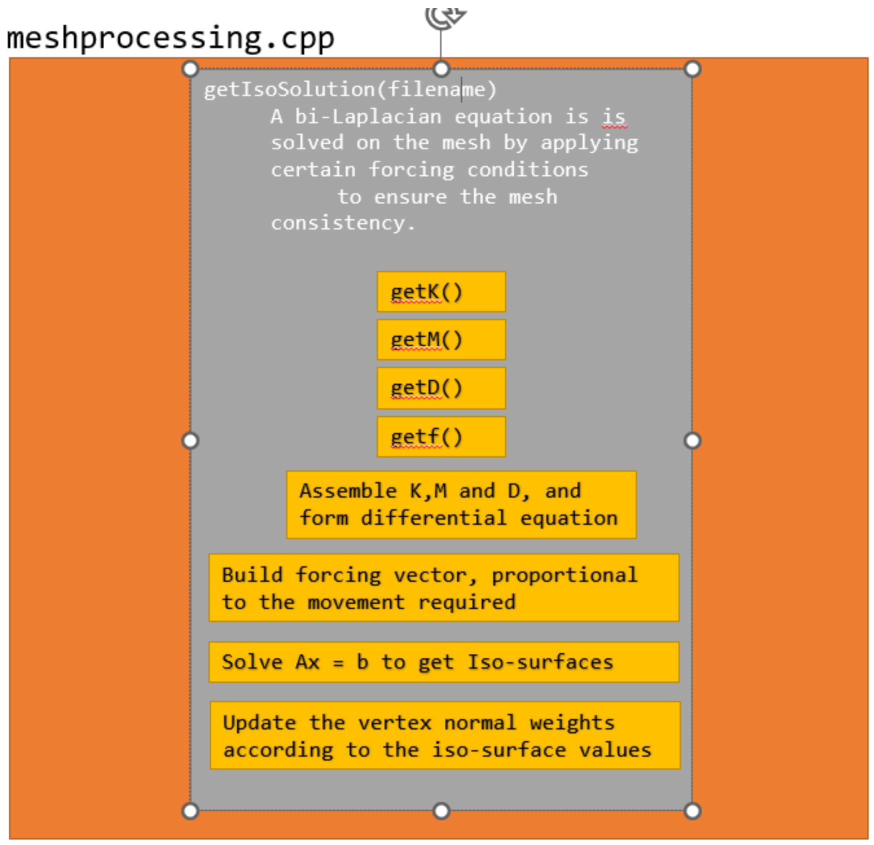
```
generateEdgeInformation(filename)
     Generates edge information
     Creates data: edge2tri and
     tri2edge
```

```
correctWinding(filename)
     Orders triangle vertex in
     clockwise order
```

```
Laplace_smoothing(filename)
     Averages vertices to
     smooth the surface
```

```
getIsoSolution(filename)
     A bi-Laplacian equation is is
     solved on the mesh by applying
     certain forcing conditions
          to ensure the mesh
     consistency.
```

2.2 Mesh Processing Procedure


meshprocessing.cpp
getIsoSolution(filename)
A bi-Laplacian equation is is solved on the mesh by applying certain forcing conditions to ensure the mesh consistency.
getK()
getM()
getD()
getf()
Assemble K,M and D, and form differential equation
Build forcing vector, proportional to the movement required
Solve Ax = b to get Iso-surfaces
Update the vertex normal weights according to the iso-surface values

2.3 References

1. Localized bi-Laplacian Solver on a Triangle Mesh and Its Applications, Jarek Rossignac (Localized bi-Laplacian Solver on a Triangle Mesh and Its Applications, Jarek Rossignac)

2. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Thomas J. R. Hughes. (Book by Dover Publishers)

3. Shape Inflation With an Adapted Laplacian Operator For Hybrid Quad/Triangle Meshes, Alexander Pinzon. (https://ucsp.edu.pe/sibgrapi2013/eproceedings/technical/114677_2.pdf)

4. Curvature-based Offset Distance: Implementations and Applications, Wei Zhuoa, Jarek Rossignaca (https://faculty.cc.gatech.edu/~jarek/papers/localVolume.pdf)

5. Lagrangian And Moving Mesh Methods For The Convection Diffusion Equation, Konstantinos Chrysafinos (https://www.esaim-m2an.org/articles/m2an/pdf/2008/01/m2an0576.pdf)

6. Laplacian Mesh Processing, Olga Sorkine (https://people.eecs.berkeley.edu/~jrs/meshpapers/Sorkine.pdf)

7. MeshSDF: Differentiable Iso-Surface Extraction, Edoardo Remelli

SECTION 3

Theory and Technical details

The first approach I tried is to solve an electrostatic Laplace/Poisson equation on the surface boundary. The problem looked straight forward, but the nature of the problem makes it hard to solve on a 3D surface, because

1. FEM electrostatics for a 3D geometry is solved on a volume mesh, on the surface mesh it can be done only if it is planar. Curved mesh it is not done generally. Complicated mapping would yield infeasible solution for general complex geometries. Existence of the open surface makes it difficult to construct a 3D geometry out of the given model.

2. The surface problems can be solved using Boundary Element, but the resulting matrix is dense, thus solving it on a large mesh of this size is not possible.

3. Acharge or multiple charge on the model is not going to give the iso-surface that we want. Assuming, through this iso-surface we need to get a proper normal offset for the mesh.

Further literature survey lead to the references mentioned above, of which the bi-Lapalacian method looked promising. I decided to move forward with that paper and the related works. This basically solves a 4th order equation

$$\alpha u_{tt} - \gamma_1 \nabla^2 u + \gamma_2 \nabla^4 u = f \tag{1}$$

Quting from the paper,

> The feedback force can be computed by solving a volumetric physical model of an elastic deformable body, using the boundary element method (BEM). Their method fully considers the volume of the object while using the surface mesh only, resulting in much a smaller matrix compared to a volumetric mesh. The resulting matrix is

much more dense but it is well conditioned and can be solved efficiently. However, their approach does not scale to larger meshes.

To overcome this issue, the PDE is solved using dynamically self adjusting computational domain. I was trying to get this to to work for our application to control the mesh movement.

The field obtained as the solution of the differential equation is used to move the mesh in the normal direction. Forcing term is created, by keeping in mind that the vertex with more than three triangles connected to it may not have a unique solution for the offset position. Thus they should have less force. By dividing by square of Atilde and square of the length of the vertex normal vector ensures that the force is proportionally reduced when more number of triangles are connected to a vertex.

Value of alpha is obtained from trian and error, the value is small but plays a big role in keeping the system stable. alpha provides momentum to the system so that the solution does not overshoot.