Finite Element Method Solver Programming(STE 6219)

Jiaxin Lin(140740)

February 26, 2016

Abstract

This is the report about the FEM Solver Programming. Using GMlib and class TriangleFasets(TriangleSystem) from UiT-Campus Narvik.The report about the finite element method theory,both of regular point set and a random generated point set,compute the matrix and demo of project.

1 Introduction

GMlib Library

GMlib is free software for use with geometric modelling with parametric in C++.In the project, Visualization of the drum is using GMlib.The libary was make and update by UIT-Campus Narvik.

Class TriangleFasets

Table 1: main class

TSTriangle	The triangle class defined by 3 edges,use getVertices to get triangle each vertex and the return value's type is Array
TSVertex	The vertex class storing 3D position and a normal,each nodes in the drum combine different vertices of triangles.
TSEdge	The edge class defined by 2 vertices.use getFirstVertex and getLastVertex compare the vertex to get same edges of two triangles and then define the other two nodes.

In the TriangleSystem have lots of different functions to get something to use in project. In this report not decrible in detail. Detailed instructions can be see in the referrence.Just look the inheritance diagram follow the figure.

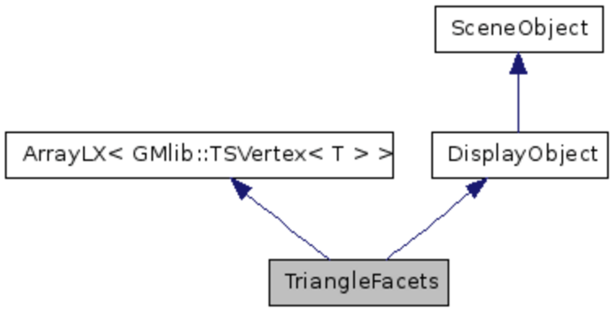

Figure 1: Inheritance diagram for TriangleFacets[1]

2 Theoretical Description

The finite element method (FEM) is a numerical technique for finding approximate solutions to boundary value problems for partial differential equations.

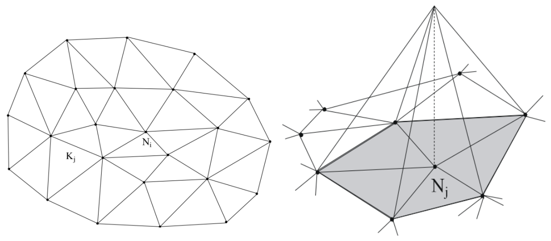

Figure 2: Finite triangulation of the body[2]

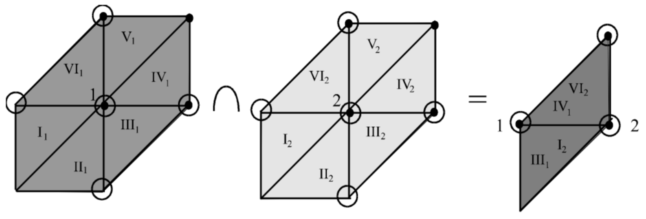

Figure 3: The two rst gures shows the support of the basis function corresponding to node 1 and node 2 respectively. The last figure shows the intersection between these supports[2]

Some formular use in the dimension 2.Observe that a $(v_i\ ,\ v_j\ ) = 0$ unless when Ni and Nj are nodes of the same triangle.

$$a\left(v_i, v_j\right) = \int_\Omega \lambda \operatorname{grad} v_i \cdot \operatorname{grad} v_j \; dx =$$

$$= \int_\Omega \lambda \left[\frac{\partial v_i}{\partial x_1}, \frac{\partial v_i}{\partial x_2}\right] \cdot \left[\frac{\partial v_j}{\partial x_1}, \frac{\partial v_j}{\partial x_2}\right] \; dx =$$

$$= \int_\Omega \lambda \left(\frac{\partial v_i}{\partial x_1}\frac{\partial v_j}{\partial x_1} + \frac{\partial v_i}{\partial x_2}\frac{\partial v_j}{\partial x_2}\right) \; dx$$

3 Random and Regular

- Generate random points in drum. Need two parameter,other can be define by the two paremeter.(radius and number of triangles) In order to avoid small triangles,have to check two inner points distance longer than s. ($s = \sqrt{\frac{4\pi r^2}{\sqrt{3}t}}$,t=number of triangles)

- Generate regular points in drum. Need three parameter begin radius,number of rings,begin points. From the first ring to increase,add one more ring in that ring's points multiply 2.

4 Compute Matrix

Outside the diagonal

First compute the stiff matrix.Find three vector to use in formular.So Need to return three values.

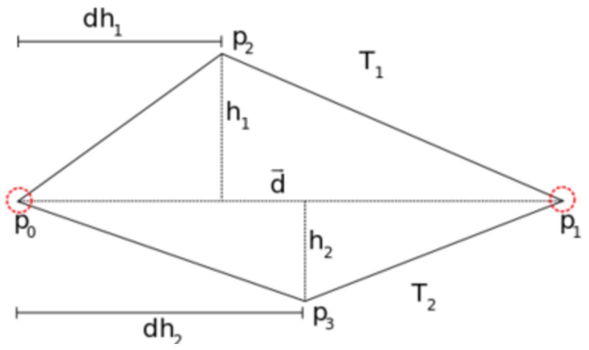

Figure 4: Computing the stiffness matrix[2]

- Formula for three vector
  $\overrightarrow{d} = p_1 - p_0$

  $\overrightarrow{a_1} = p_2 - p_0$

  $\overrightarrow{a_2} = p_3 - p_0$

- Computing Stiffness matrix elements outside the diagonal.Use the three vector to combine the below formula .

```
if(sameEdge!=nullptr){
    auto getVector=findVector(sameEdge);
    double dd=1/(getVector[0]*getVector[0]);
    //first trangle
    double area1=std::abs(getVector[0]^getVector[1]);
    double dh1=dd*(getVector[1]*getVector[0]);
    double h1=dd*area1*area1;
    //sencond triangle
    double area2=std::abs(getVector[2]^getVector[0]);
    double dh2=dd*(getVector[2]*getVector[0]);
    double h2=(dd*area2*area2);
    //computing Amatrix[i][j]

_Amatrix[i][j]=_Amatrix[j][i]=(((dh1*(1-dh1)/h1)-dd)*
((area1)/2)+((dh2*(1-dh2)/h2)-dd)*((area2)/2));

                    }
```

Diagonal

Computing stiffness matrix diagonal elements

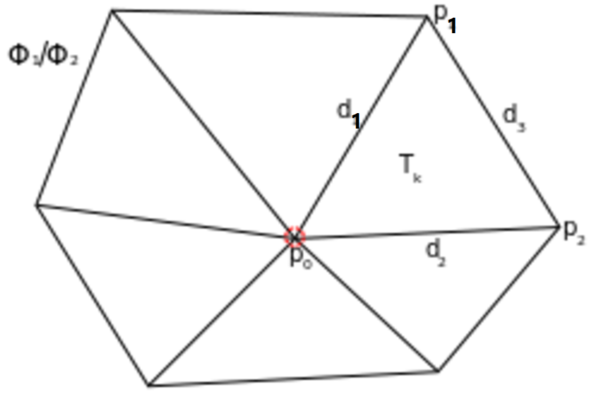

Figure 5: Computing the stiffness matrix diagonal elements[2]

- The values on the diagonal of the stiffness matrix is where the element is computed against itself.
  $\overrightarrow{d_1} = p_2 - p_0$
  $\overrightarrow{d_2} = p_1 - p_0$
  $\overrightarrow{d_3} = p_2 - p_1$

```
 _Amatrix[i][i]+=((Dvector[2]*Dvector[2])/
(2*std::abs(Dvector[0]^Dvector[1])));
```

Vector B

- Computing Vector B

```
auto Bvector=findVectorD(_nodes[i],t[j]);
_b[i]+=((std::abs(Bvector[0]^Bvector[1]))/6);
```

5 Conclusion

In conclusion,this project is based on the lectures in the course: partial differential equations and FEM. Througe this project understand the fem better.Practise better than theory.Moreover,with use the GMlib libary and see the code form the teacher.Students will make progress in C++ and math.

References

[1] Dag Lukkassen, *A short introduction to Sobolev-spaces and applications for engineering students*, November, 2004.

[2] `http://episteme.hin.no/dox/GMlib2/classGMlib_1_1TriangleFacets.html`