Travaux Pratiques : Manipulation de Maillages en C++

1 Programmes fournis

Le code fourni comprend une classe `Mesh` qui représente un maillage 2D composé de nœuds et de triangles. Le maillage est lu à partir d'un fichier au format Gmsh (.msh). Le programme principal lit un fichier de maillage nommé `square.msh` et génère un fichier lisible par Gnuplot pour visualiser le maillage.

Pour compiler et lancer le programme, utilisez les commandes suivantes dans le terminal :

```
mkdir build
cd build
cmake ..
make
./go
gnuplot gnu.plt
```

Pour générer un maillage plus fin, modifier le paramètre `raf` du fichier `square.geo` et relancer Gmsh. Puis taper (dans le bon dossier) :

```
gmsh square.geo -format msh2 -2 -o square.msh
```

2 Questions

2.1 Écrire une fonction naïve pour trouver le triangle contenant un point

Écrivez une fonction qui, étant donné un point, retrouve le numéro d'un triangle qui le contient. Commencez par une fonction naïve qui parcourt

tous les triangles du maillage.

2.2 Écrire une fonction efficace utilisant une hashmap

Écrivez une fonction plus efficace qui utilise une hashmap pour accélérer la recherche du triangle contenant un point. Vous pouvez découper l'espace en un quadrillage dont le pas sera choisi judicieusement. Pour chaque carré, il faut construire la liste des triangles qui l'intersectent. Vérifiez l'accélération du nouvel algorithme par rapport à la version naïve en fonction de la finesse du maillage.

2.3 Écrire une fonction naïve pour trouver les triangles voisins

Écrivez une fonction qui construit la liste des triangles voisins d'un triangle donné par les arêtes. Commencez par une fonction naïve qui parcourt tous les triangles du maillage.

2.4 Écrire une fonction efficace utilisant une hashmap

Écrivez une fonction plus efficace qui utilise une hashmap pour accélérer la recherche des triangles voisins. Vérifiez l'accélération du nouvel algorithme par rapport à la version naïve en fonction de la finesse du maillage.

3 Annexes

3.1 Code fourni

3.1.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)

project(go)

set(CMAKE_CXX_STANDARD 11)

include_directories(src)

add_executable(go src/msh.h src/msh.cpp src/main.cpp)
```

3.1.2 src/msh.h

```
#ifndef MESH_H
#define MESH_H
#include <iostream>
#include <fstream>
#include <vector>
#include <array>

using namespace std;

// classe maillage 2d avec liste de noeuds et liste de triangles
class Mesh
{
private:
    vector<array<double, 2>> node;
    vector<array<int, 3>> triangle;

public:
    Mesh(string gmsh2_filename);
    friend ostream &operator<<(ostream &os, const Mesh &m);
    // affichage dans un fichier lisible par gnuplot
    void writeGnuplot(const string &filename) const;
};

#endif
```

3.1.3 src/msh.cpp

```
#include "msh.h"
#include <cassert>

// classe maillage 2d avec liste de noeuds et liste de triangles
Mesh::Mesh(string gmsh2_filename)
{
    ifstream file = ifstream(gmsh2_filename);
    if (!file.is_open()) {
        throw runtime_error("file not found");
    }
    string line;
    while (getline(file, line))
    {
        if (line == "$Nodes")
        {
            int n;
            file >> n;
            node.resize(n);
            for (int i = 0; i < n; i++)
            {
                int id;
                double z;
                file >> id;
                file >> node[id - 1][0] >> node[id - 1][1] >> z;
            }
        }
        if (line == "$Elements")
        {
            int n;
```

```
30              file >> n;
31              for (int i = 0; i < n; i++)
32              {
33                  int id, type, ntags;
34                  file >> id >> type >> ntags;
35                  for (int j = 0; j < ntags; j++)
36                  {
37                      int tag;
38                      file >> tag;
39                  }
40                  if (type == 2)
41                  {
42                      array<int, 3> t;
43                      file >> t[0] >> t[1] >> t[2];
44                      t[0]--;
45                      t[1]--;
46                      t[2]--;
47                      triangle.push_back(t);
48                  } // sinon sauter la ligne (on ne connait pas sa taille)
49                  else
50                  {
51                      string line;
52                      getline(file, line);
53                  }
54              }
55          }
56      }
57  };
58  // affichage dans un fichier lisible par gnuplot
59  void Mesh::writeGnuplot(const string &filename) const
60  {
61      ofstream file(filename);
62      // file << "set terminal postscript eps enhanced color" << endl;
63      // file << "set output '" << filename << ".eps'" << endl;
64      //  file << "set xrange [-1:1]" << endl;
65      //  file << "set yrange [-1:1]" << endl;
66      //  freeze aspect ratio
67      file << "set size ratio -1" << endl;
68      file << "plot '-' with linespoints" << endl;
69      for (int i = 0; i < triangle.size(); i++)
70      {
71          file << node[triangle[i][0]][0] << " " << node[triangle[i][0]][1] <<
                endl;
72          file << node[triangle[i][1]][0] << " " << node[triangle[i][1]][1] <<
                endl;
73          file << node[triangle[i][2]][0] << " " << node[triangle[i][2]][1] <<
                endl;
74          file << node[triangle[i][0]][0] << " " << node[triangle[i][0]][1] <<
                endl;
75          file << endl;
76      }
77      file << "e" << endl;
78      file << "pause -1";
79      file.close();
80  }
81
82  ostream &operator<<(ostream &os, const Mesh &m)
83  {
84      os << "Mesh with " << m.node.size() << " nodes and " << m.triangle.size
            () << " triangles" << endl;
85      for (int i = 0; i < m.node.size(); i++)
86      {
```

```
87          os << "Node " << i << ": " << m.node[i][0] << " " << m.node[i][1] <<
                endl;
88      }
89      for (int i = 0; i < m.triangle.size(); i++)
90      {
91          os << "Triangle " << i << ": " << m.triangle[i][0] << " " << m.
                triangle[i][1] << " " << m.triangle[i][2] << endl;
92      }
93      return os;
94  }
```

3.1.4 src/main.cpp

```
1  #include <msh.h>
2
3
4  int main()
5  {
6      Mesh m("../square.msh");
7      //cout << m << endl;
8      m.writeGnuplot("gnu.plt");
9      return 0;
10 }
```

3.1.5 square.geo

```
1  // maillage avec gmsh square.geo -2 -format msh2
2  raf= 0.1;
3  L=1;
4  Point(1)= {0,0,0,raf};
5  Point(2)= {L,0,0,raf};
6  Point(3)= {L,L,0,raf};
7  Point(4)= {0,L,0,raf};
8
9  Line(1)= {1,2};
10 Line(2)= {2,3};
11 Line(3)= {3,4};
12 Line(4)= {4,1};
13
14 Line Loop(1)= {1,2,3,4};
15 Plane Surface(1)= {1};
16
17 Mesh.ElementOrder=1 ;
18 Mesh.SecondOrderIncomplete=1;
```