# 流体力学中的现代数值方法I大作业

姓名：张睿谦 学号：117010910026

## 一、问题描述

## 

1

3

20

7

如图所示，计算理想不可压缩无旋流动，

1. 采用势函数或者流函数方程；
2. 写出完整计算步骤（方程建立、网格分布、单元分析、速度压力计算等）；
3. 采用多套不同密度的网格；
4. 给出每套网格计算的流线分布、速度矢量分布、压力云图等；
5. 列表比较不同密度网格计算的若干位置的速度和压力等的差别。

## 二、三角形单元有限元方程推导

1.泛函极值问题

如下图所示，将边界编号，

5

3

4

5

1

2

4

势函数方程：

边界条件：

圆柱壁面：1,2,3，

上下壁面：4, 

远场边界：5，

弱解积分表达式为

所以有极值问题，找，使得，

取得最小值。

2.确定试探函数



在三角形单元中，引入面积坐标，，

对三角形单元进行局部编码，



其中

，为三角形面积

所以试探函数为

3.求解系数矩阵

由于将各个单元内的试探函数带入表达式，并对求导，令其结果等于零可得，

，

其中

将各个单元系数矩阵进行叠加，得到整体系数矩阵，这里忽略了边界条件对方程的影响，在最后利用对角线项扩大法对系数矩阵进行修正，使得其满足边界条件。

4.求解线性方程组

利用Gauss–Seidel迭代求解线性方程组，可利用

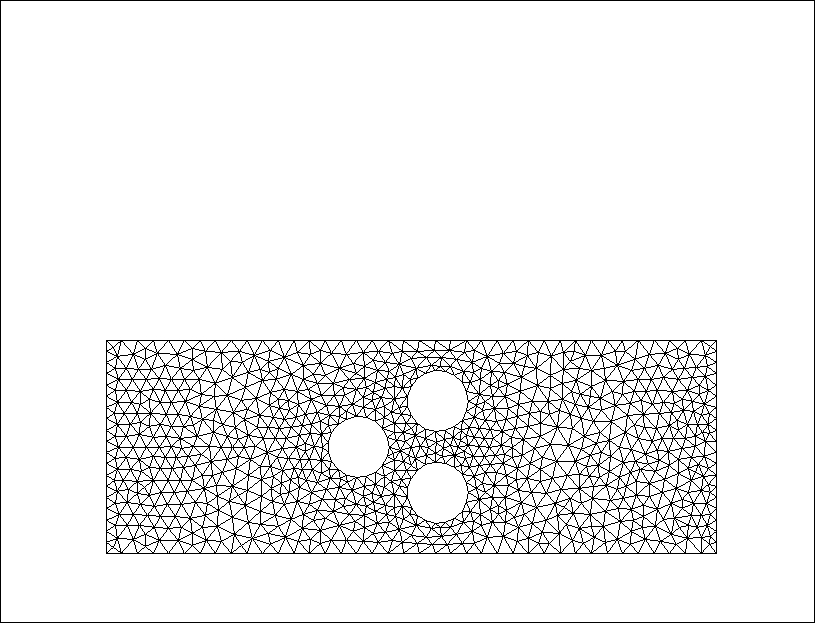
，，

解得速度与压力，在本程序中未使用此方式，而是采用tecplot后处理软件对速度、压力进行求解。

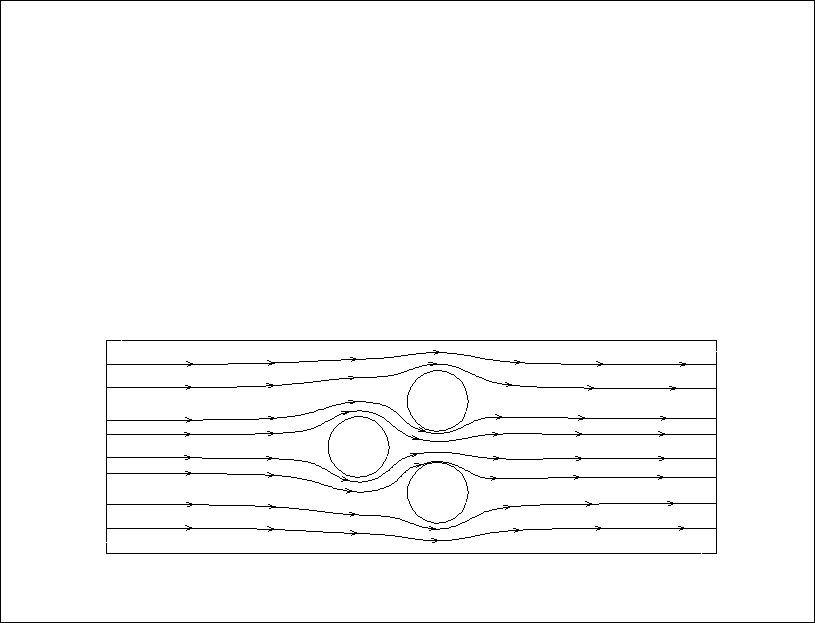
## 三、计算结果

本例中使用非结构的三角形单元，由Gmsh软件自动生成。

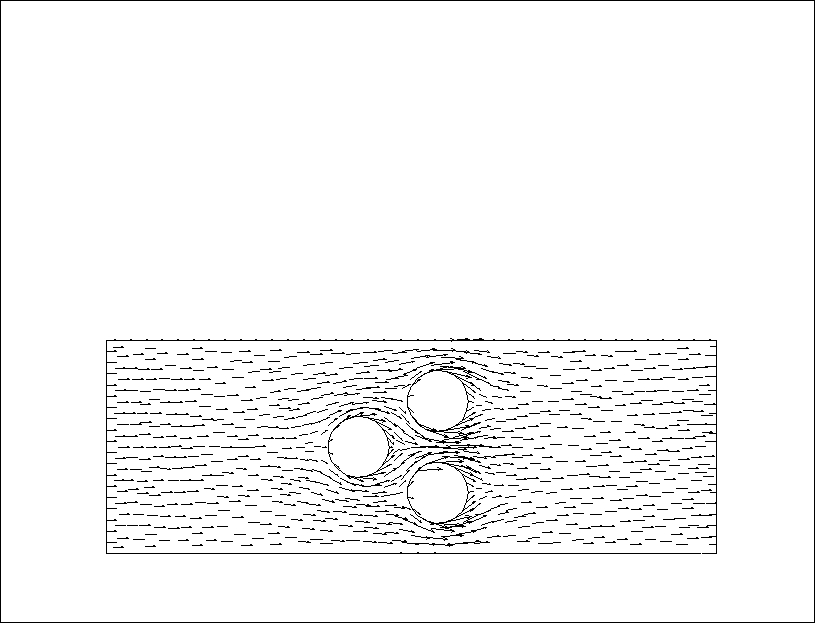
1. 网格1：1，2，3，5边界每条曲线分布20个节点，4边界分布40个节点，总共976个节点，1765个单元。
2. 网格



1. 流线图



1. 速度矢量图



1. 速度云图

图片包含 屏幕截图

已生成极高可信度的说明图片包含 屏幕截图

已生成极高可信度的说明

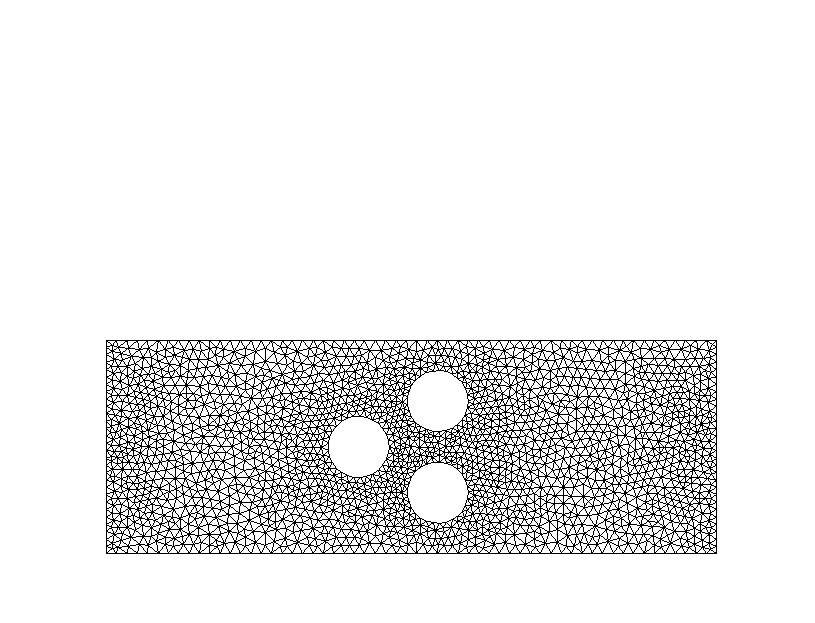
1. 压力云图

图片包含 屏幕截图

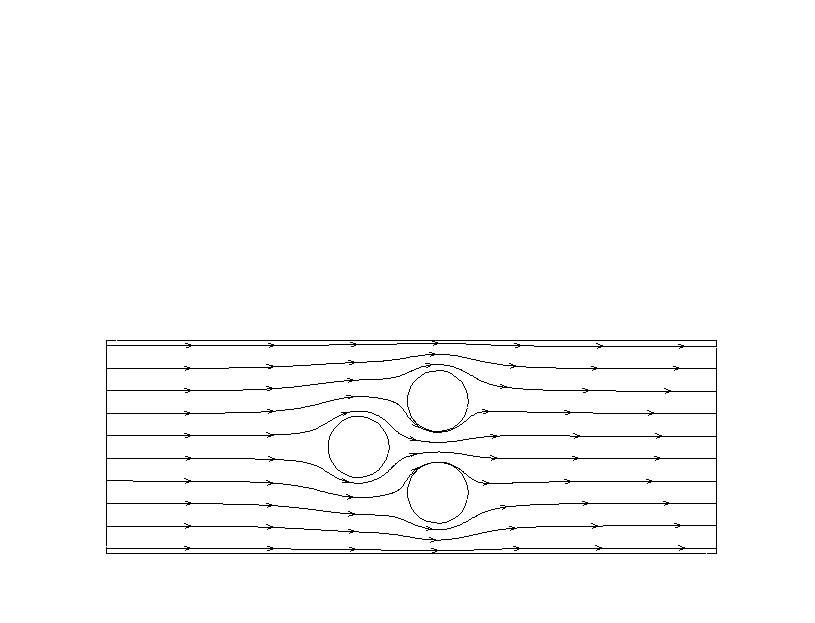
已生成极高可信度的说明图片包含 屏幕截图

已生成极高可信度的说明

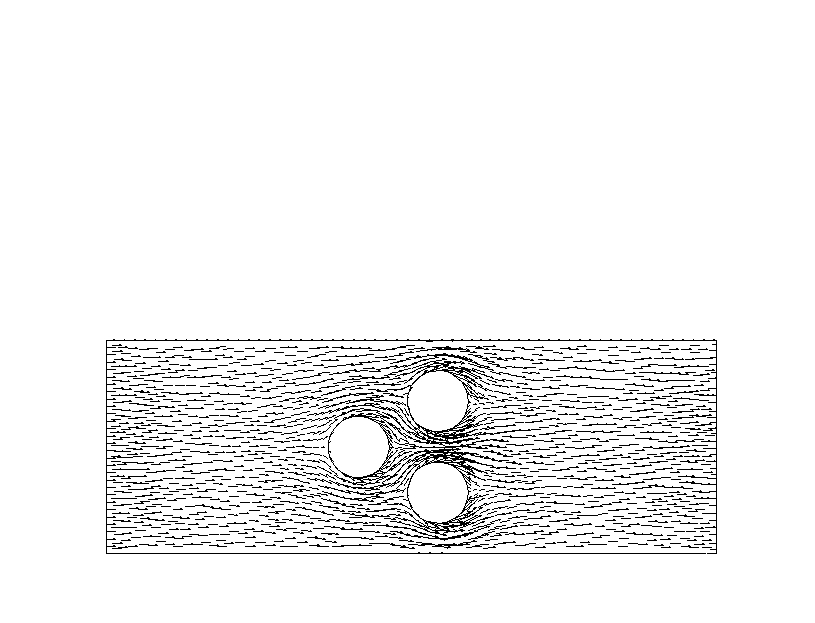
1. 网格2：1，2，3，5边界每条曲线分布30个节点，4边界分布60个节点，总共2148个节点，4037个单元。
2. 网格



1. 流线图



1. 速度矢量图



1. 速度云图

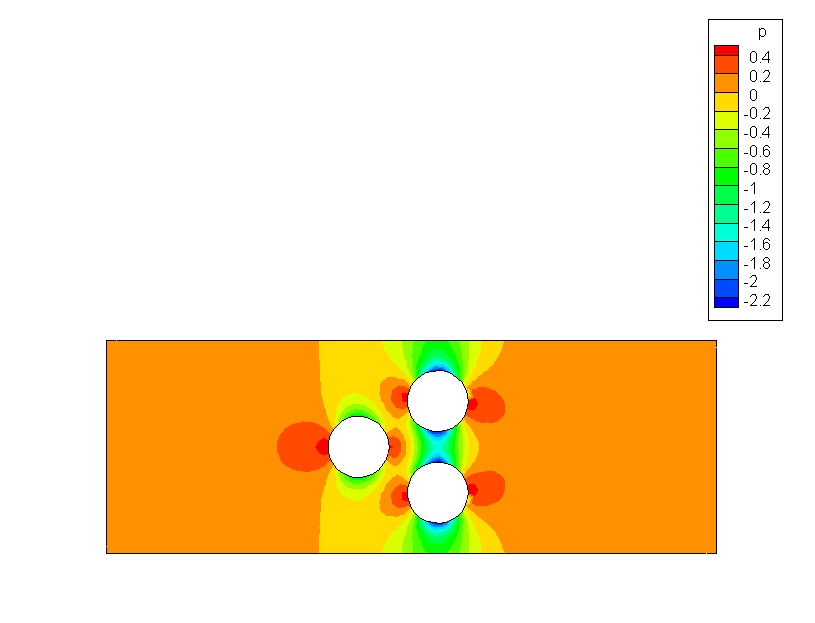
图片包含 屏幕截图

已生成极高可信度的说明图片包含 屏幕截图

已生成高可信度的说明

1. 压力云图

图片包含 屏幕截图

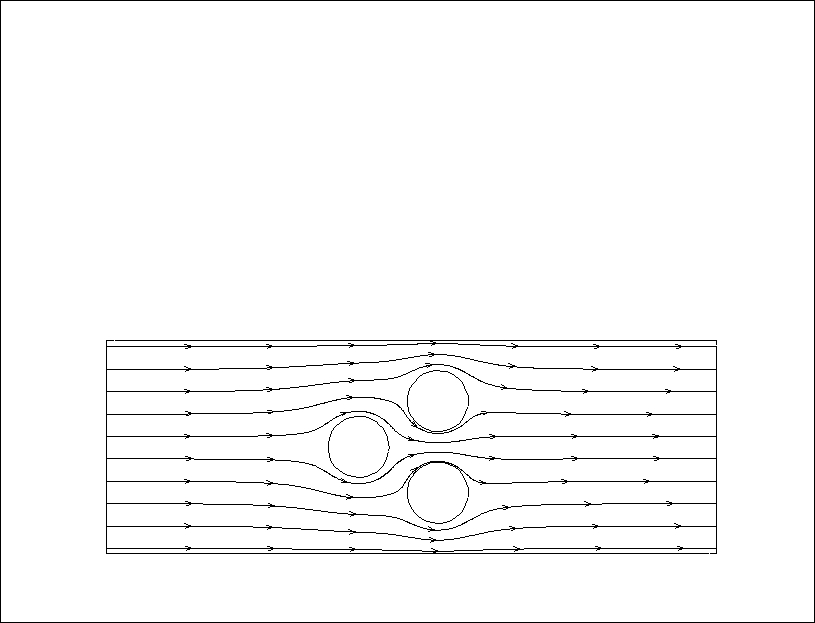
已生成极高可信度的说明

1. 网格3：1，2，3，5边界每条曲线分布40个节点，4边界分布50个节点，总共3852个节点，7355个单元。
2. 网格

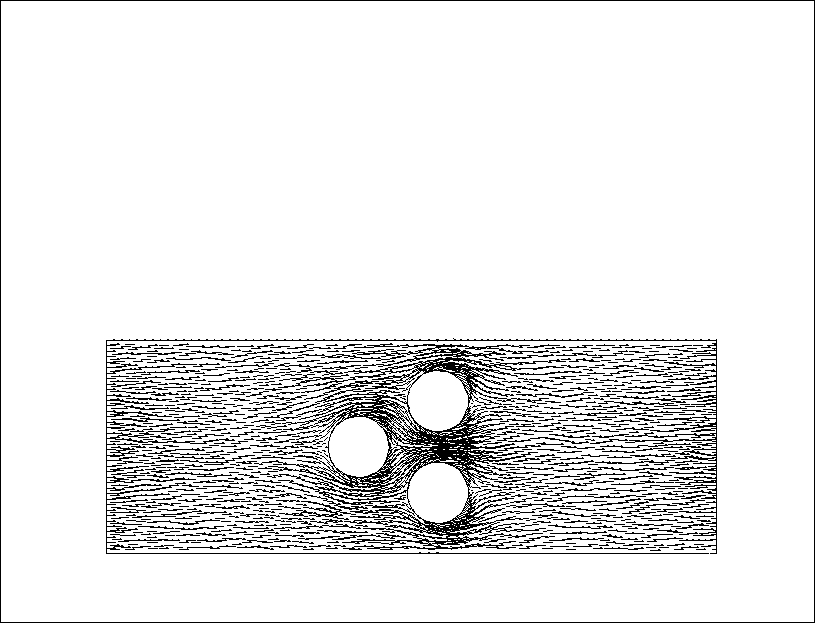
图片包含 金属器皿

已生成高可信度的说明

1. 流线图



1. 速度矢量图



1. 速度云图

图片包含 屏幕截图

已生成极高可信度的说明图片包含 屏幕截图

已生成极高可信度的说明

1. 压力云图

图片包含 屏幕截图

已生成极高可信度的说明图片包含 屏幕截图

已生成极高可信度的说明

1. 比较特定位置点的压力和速度值

1

2

3

4

5

6

7

1

2

5

6

7

如上图所示，在不同网格下，对以上7个点的压力，速度进行比较，

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 0.355 | 0.203 | 0.104 |
| V(m/s) | 0.195 | 0.000 | 0.000 |
| Velocity(m/s) | 0.405 | 0.262 | 0.167 |
| p(Pa) | 0.418 | 0.465 | 0.486 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 1.653 | 1.698 | 1.732 |
| V(m/s) | 0.028 | -0.006 | 0.011 |
| Velocity(m/s) | 1.667 | 1.706 | 1.735 |
| p(Pa) | -0.891 | -0.954 | -1.006 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 0.252 | 0.376 | 0.091 |
| V(m/s) | 0.004 | -0.004 | 0.001 |
| Velocity(m/s) | 0.252 | 0.376 | 0.091 |
| p(Pa) | 0.468 | 0.429 | 0.495 |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 0.467 | 0.199 | 0.116 |
| V(m/s) | -0.238 | -0.275 | -0.287 |
| Velocity(m/s) | 0.589 | 0.373 | 0.314 |
| p(Pa) | 0.321 | 0.415 | 0.436 |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 2.182 | 2.312 | 2.372 |
| V(m/s) | 0.005 | 0.009 | 0.002 |
| Velocity(m/s) | 2.200 | 2.326 | 2.376 |
| p(Pa) | -1.920 | -2.206 | -2.322 |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 0.441 | 0.157 | 0.048 |
| V(m/s) | -0.129 | -0.144 | -0.154 |
| Velocity(m/s) | 0.459 | 0.213 | 0.161 |
| p(Pa) | 0.395 | 0.477 | 0.487 |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 网格1 | 网格2 | 网格3 |
| U(m/s) | 2.244 | 2.326 | 2.396 |
| V(m/s) | 0.004 | -0.008 | -0.020 |
| Velocity(m/s) | 2.254 | 2.333 | 2.400 |
| p(Pa) | -2.046 | -2.222 | -2.380 |

根据比较我们可以发现，在不同的网格密度下，3、4、6点处速度结果相差较大，应该在圆柱的前后部分生成较密的网格以增加计算精度。

## 四、C++程序

#include<iostream>

#include<fstream>

#include<string>

#include<vector>

using namespace std;

int\* read\_mesh(double \*\*point, vector<int> &number\_boundary, vector<int> &boundary\_point1, vector<int> &boundary\_point2, vector <int> &element\_node1, vector <int> &element\_node2, vector <int> &element\_node3)//读取坐标

{

int number\_point;//节点个数

ifstream infile;

double temp;//将无用信息赋值给temp

infile.open("point.txt");//打开节点文件

infile >> number\_point;

for (int i = 0; i < 2; i++)

{

point[i] = new double[number\_point];

}

//二维数组point[0][j],[1][j]分别为第j个节点的x,y坐标

for (int j = 0; j < number\_point; j++)

{

infile >> temp >> point[0][j] >> point[1][j] >> temp;

}

infile.close();

infile.open("boundary.txt");//打开边界文件

while (!infile.eof())

{

int n;

infile >> temp >> temp >> temp >> n;

number\_boundary.push\_back(n);//边界编号，其中5为进出口边界

infile >> temp >> n;

boundary\_point1.push\_back(n - 1);//x坐标

infile >> n;

boundary\_point2.push\_back(n - 1);//y坐标

}

infile.close();

infile.open("field.txt");//打开内部场单元文件

while (!infile.eof())

{

int n;

infile >> temp >> temp >> temp >> temp >> temp >> n;

element\_node1.push\_back(n - 1);

infile >> n;

element\_node2.push\_back(n - 1);

infile >> n;

element\_node3.push\_back(n - 1);

}//三角形单元的三个节点

infile.close();

int \*p = new int[3];

p[0] = number\_point;//节点个数

p[1] = number\_boundary.size();//边界个数

p[2] = element\_node1.size();//单元个数

return p;//返回数组指针，分别为节点个数，边界个数，单元个数

}

double\* matrix\_subtract(double \*a, double \*b, int &m)

{

for (int i = 0; i < m; i++)

{

b[i] = a[i] - b[i];

}

return b;

}//矩阵相减

double Norm\_2(double \*a, int &m)

{

double n = 0;

for (int i = 0; i < m; i++)

{

n = n + a[i] \* a[i];

}

n = sqrt(n);

return n;

}//求范数

void Gauss\_Seidel(double \*\*A, double \*y, double \*r, int &m, double error)

{

double \*y\_old;

y\_old = new double[m];

double N\_2;

do

{

for (int i = 0; i < m; i++)

{

y\_old[i] = y[i];

}

for (int i = 0; i < m; i++)

{

double sum = 0;

for (int j = 0; j < m; j++)

{

if (j == i) continue;//跳过Aii

sum = sum + A[i][j] \* y[j];

}

y[i] = (r[i] - sum) / A[i][i];

}

N\_2 = Norm\_2(matrix\_subtract(y, y\_old, m), m);

cout << N\_2<<endl;

} while (N\_2> error);

delete[] y\_old;

y\_old = NULL;

}//高斯-赛德尔迭代法

int main()

{

double \*\*point = new double\*[2];//节点坐标

vector <int> number\_boundary;//节点编号，其中5为出入口边界

vector <int> boundary\_point1, boundary\_point2;//边界节点

vector <int> element\_node1, element\_node2, element\_node3;//单元内节点编号

int\* info=read\_mesh(point, number\_boundary, boundary\_point1, boundary\_point2, element\_node1, element\_node2, element\_node3);

int \*\*element\_node = new int\*[info[2]];

for (int i = 0; i < info[2]; i++)

{

element\_node[i] = new int[3];

}

for (int i = 0; i < info[2]; i++)

{

element\_node[i][0] = element\_node1[i];

element\_node[i][1] = element\_node2[i];

element\_node[i][2] = element\_node3[i];

}//第i个单元的第1,2,3个节点

double \*\*A = new double\*[info[2]];

double \*\*B = new double\*[info[2]];

double \*\*C = new double\*[info[2]];

double \*E = new double[info[2]];

for (int i = 0; i < info[2]; i++)

{

A[i] = new double[3];

B[i] = new double[3];

C[i] = new double[3];

}

for (int i = 0; i < info[2]; i++)

{

for (int j = 0; j < 3; j++)

{

A[i][j] = point[0][element\_node[i][(j + 1) % 3]] \* point[1][element\_node[i][(j + 2) % 3]] - point[1][element\_node[i][(j + 1) % 3]] \* point[0][element\_node[i][(j + 2) % 3]];

B[i][j] = point[1][element\_node[i][(j + 1) % 3]] - point[1][element\_node[i][(j + 2) % 3]];

C[i][j] = point[0][element\_node[i][(j + 2) % 3]] - point[0][element\_node[i][(j + 1) % 3]];

}

E[i] = (B[i][0] \* C[i][1] - B[i][1] \* C[i][0]);

}

double \*\*K = new double\*[info[0]];

for (int i = 0; i < info[0]; i++)

{

K[i] = new double[info[0]];

}//系数矩阵

for (int i = 0; i < info[0]; i++)

{

for (int j = 0; j < info[0]; j++)

{

K[i][j] = 0;

}

}//初始为零

for (int i = 0; i < info[2]; i++)

{

for (int j = 0; j < 3; j++)

{

for (int k = 0; k < 3; k++)

{

K[element\_node[i][j]][element\_node[i][k]] = K[element\_node[i][j]][element\_node[i][k]] + (B[i][j

] \* B[i][k] + C[i][j] \* C[i][k]) / (4 \* E[i]);

}

}

}

double \*Phi = new double[info[0]];

double \*R = new double[info[0]];

for (int i = 0; i < info[0]; i++)

{

Phi[i] = 0;

R[i] = 0;

}//待求量与右边项

for (int i = 0; i < info[1]; i++)

{

if (number\_boundary[i] == 5)

{

K[boundary\_point1[i]][boundary\_point1[i]] = 1e20;

K[boundary\_point2[i]][boundary\_point2[i]] = 1e20;

R[boundary\_point1[i]] = 1e20\*point[0][boundary\_point1[i]];

R[boundary\_point2[i]] = 1e20\*point[0][boundary\_point2[i]];

}

}//处理边界条件

Gauss\_Seidel(K, Phi, R, info[0], 1e-6);//求解

//输出文件

ofstream outfile;

outfile.open("result.plt");

outfile << "VARIABLES=" << "\"X\"" << "\"Y\"" << "\"Phi\"" << endl;

outfile << "ZONE N=" << info[0] << ", E=" << info[2] << ", F=FEPOINT, ET=TRIANGLE" << endl;

for (int i = 0; i < info[0]; i++)

{

outfile << point[0][i] << " " << point[1][i] << " " << Phi[i] << endl;

}

cout << endl;

for (int i = 0; i < info[2]; i++)

{

outfile << element\_node[i][0]+1 << " " << element\_node[i][1]+1 << " " << element\_node[i][2]+1 << endl;

}

outfile.close();

//释放内存

delete[] point[0];

delete[] point[1];

delete[] point;

for (int i = 0; i < info[2]; i++)

{

delete[] element\_node[i];

element\_node[i] = NULL;

delete[] A[i];

A[i] = NULL;

delete[] B[i];

B[i] = NULL;

delete[] C[i];

C[i] = NULL;

}

delete[] element\_node;

element\_node = NULL;

delete[] A;

A = NULL;

delete[] B;

B = NULL;

delete[] C;

C = NULL;

delete[] E;

E = NULL;

for (int i = 0; i < info[0]; i++)

{

delete[] K[i];

K[i] = NULL;

}

delete[] K;

K = NULL;

delete[] Phi;

Phi = NULL;

delete[] R;

R = NULL;

return 0;

}

## 五、附录

网格文件举例

point.txt

注：第一行为节点个数，第一列为编号，第二、三、四列为x，y，z坐标

95

1 0 0 0

2 20 0 0

3 20 7 0

4 0 7 0

5 9.268000000000001 3.5 0

6 11.866 2 0

7 11.866 5 0

8 8.891489801858734 4.28183148246803 0

9 8.045479066043685 4.474927912181823 0

10 7.367031132097581 3.933883739117556 0

11 7.367031132097583 3.06611626088244 0

12 8.045479066043688 2.525072087818176 0

13 8.891489801858734 2.718168517531971 0

14 11.48948980185873 2.78183148246803 0

15 10.64347906604368 2.974927912181823 0

16 9.965031132097579 2.433883739117556 0

17 9.965031132097581 1.56611626088244 0

18 10.64347906604369 1.025072087818176 0

19 11.48948980185873 1.218168517531971 0

20 11.48948980185873 5.78183148246803 0

21 10.64347906604368 5.974927912181823 0

22 9.965031132097579 5.433883739117556 0

23 9.965031132097581 4.56611626088244 0

24 10.64347906604369 4.025072087818176 0

25 11.48948980185873 4.218168517531971 0

26 2.857142867218561 0 0

27 5.714285737427272 0 0

28 8.571428605306497 0 0

29 11.42857146282696 0 0

30 14.28571432034742 0 0

31 17.14285716785606 0 0

32 20 1.400000004516114 0

33 20 2.800000008938199 0

34 20 4.200000008919393 0

35 20 5.600000004459696 0

36 17.14285712263523 7 0

37 14.28571424527047 7 0

38 11.42857137256467 7 0

39 8.571428520576422 7 0

40 5.714285668588161 7 0

41 2.857142826611735 7 0

42 0 5.599999998716303 0

43 0 4.199999999861216 0

44 0 2.799999997710159 0

45 0 1.39999999885508 0

46 14.35313308079221 3.500000003777131 0

47 3.421236938966502 3.499999997816183 0

48 16.89708359618837 3.500000008891035 0

49 8.764032942912952 5.495180929728397 0

50 8.76403294729883 1.504819071679103 0

51 5.49623841148998 3.499999999593039 0

52 2.342023453226764 1.973592879973981 0

53 2.342023448088763 5.02640711696318 0

54 18.13787704613577 5.032915217992071 0

55 18.1378770531494 1.967084795431792 0

56 6.529729867966005 5.611936021284485 0

57 6.529729881638894 1.388063968286891 0

58 13.03789457703695 0.9540706969732947 0

59 13.03789454934369 6.045929289690332 0

60 4.699222420340781 1.834977026602407 0

61 4.69922240150732 5.165022971709726 0

62 1.394067728789749 3.499999998785688 0

63 1.224512269173438 5.905281422993649 0

64 1.224512278422655 1.094718575422336 0

65 18.63456753909471 3.500000008034498 0

66 18.82018355525136 0.8417711999869764 0

67 18.82018354219275 6.158228805612941 0

68 12.53547904863178 3.499999999972587 0

69 13.27200134149448 2.421734244158977 0

70 13.27200133528811 4.578265759369213 0

71 10.05595941430246 3.499999999999998 0

72 9.688722667017274 6.160453450007557 0

73 9.68872269284253 0.8395465400433381 0

74 15.30379442509341 5.236730039959969 0

75 15.30379444812356 1.763269964176033 0

76 11.36028335688732 3.499999999994518 0

77 6.659619995388545 3.500000001290198 0

78 9.321027970067171 2.087796913601465 0

79 9.321027969838836 4.912203086471813 0

80 7.744335024098264 5.670543325215523 0

81 7.744335041772048 1.329456662588099 0

82 6.34128580584887 4.404975233459294 0

83 6.341285811409305 2.595024764994651 0

84 12.33356328539601 2.706741229740504 0

85 12.33356328539601 4.293258770259498 0

86 9.514036329696527 2.899940529506065 0

87 9.514036329696525 4.100059470493933 0

88 7.139944816536177 0.6793801577187475 0

89 7.139944770307213 6.320619836625002 0

90 10.08305045675492 0.4661546569653785 0

91 10.08305040655051 6.533845340547344 0

92 7.205572179210882 4.819253246251736 0

93 7.205572186592303 2.180746748914051 0

94 0.99212068921039 4.846337707464008 0

95 0.9921206920878337 2.153662290149449 0

boundary.txt

注：第四列为边界编号，第六、七列为节点编号

1 1 2 1 1 5 8

2 1 2 1 1 8 9

3 1 2 1 1 9 10

4 1 2 1 1 10 11

5 1 2 1 1 11 12

6 1 2 1 1 12 13

7 1 2 1 1 13 5

8 1 2 3 2 6 14

9 1 2 3 2 14 15

10 1 2 3 2 15 16

11 1 2 3 2 16 17

12 1 2 3 2 17 18

13 1 2 3 2 18 19

14 1 2 3 2 19 6

15 1 2 2 3 7 20

16 1 2 2 3 20 21

17 1 2 2 3 21 22

18 1 2 2 3 22 23

19 1 2 2 3 23 24

20 1 2 2 3 24 25

21 1 2 2 3 25 7

22 1 2 5 4 1 26

23 1 2 5 4 26 27

24 1 2 5 4 27 28

25 1 2 5 4 28 29

26 1 2 5 4 29 30

27 1 2 5 4 30 31

28 1 2 5 4 31 2

29 1 2 4 5 2 32

30 1 2 4 5 32 33

31 1 2 4 5 33 34

32 1 2 4 5 34 35

33 1 2 4 5 35 3

34 1 2 5 6 3 36

35 1 2 5 6 36 37

36 1 2 5 6 37 38

37 1 2 5 6 38 39

38 1 2 5 6 39 40

39 1 2 5 6 40 41

40 1 2 5 6 41 4

41 1 2 4 7 4 42

42 1 2 4 7 42 43

43 1 2 4 7 43 44

44 1 2 4 7 44 45

45 1 2 4 7 45 1

field.txt

注：第六、七、八列为三角形节点编号

46 2 2 6 1 6 19 58

47 2 2 6 1 20 7 59

48 2 2 6 1 38 20 59

49 2 2 6 1 19 29 58

50 2 2 6 1 13 12 50

51 2 2 6 1 9 8 49

52 2 2 6 1 48 54 74

53 2 2 6 1 55 48 75

54 2 2 6 1 48 46 75

55 2 2 6 1 46 48 74

56 2 2 6 1 34 35 54

57 2 2 6 1 32 33 55

58 2 2 6 1 68 25 76

59 2 2 6 1 14 68 76

60 2 2 6 1 29 19 18

61 2 2 6 1 20 38 21

62 2 2 6 1 69 58 75

63 2 2 6 1 59 70 74

64 2 2 6 1 24 23 71

65 2 2 6 1 16 15 71

66 2 2 6 1 31 55 75

67 2 2 6 1 54 36 74

68 2 2 6 1 16 71 86

69 2 2 6 1 71 23 87

70 2 2 6 1 13 50 78

71 2 2 6 1 49 8 79

72 2 2 6 1 15 24 71

73 2 2 6 1 24 15 76

74 2 2 6 1 17 50 73

75 2 2 6 1 49 22 72

76 2 2 6 1 50 17 78

77 2 2 6 1 22 49 79

78 2 2 6 1 22 21 72

79 2 2 6 1 18 17 73

80 2 2 6 1 47 53 62

81 2 2 6 1 52 47 62

82 2 2 6 1 55 33 65

83 2 2 6 1 34 54 65

84 2 2 6 1 41 53 61

85 2 2 6 1 52 26 60

86 2 2 6 1 43 44 62

87 2 2 6 1 53 47 61

88 2 2 6 1 47 52 60

89 2 2 6 1 58 30 75

90 2 2 6 1 37 59 74

91 2 2 6 1 28 50 81

92 2 2 6 1 49 39 80

93 2 2 6 1 33 34 65

94 2 2 6 1 32 55 66

95 2 2 6 1 54 35 67

96 2 2 6 1 30 31 75

97 2 2 6 1 36 37 74

98 2 2 6 1 68 14 84

99 2 2 6 1 25 68 85

100 2 2 6 1 4 42 63

101 2 2 6 1 45 1 64

102 2 2 6 1 61 51 82

103 2 2 6 1 51 60 83

104 2 2 6 1 55 31 66

105 2 2 6 1 36 54 67

106 2 2 6 1 25 24 76

107 2 2 6 1 15 14 76

108 2 2 6 1 9 49 80

109 2 2 6 1 50 12 81

110 2 2 6 1 11 10 77

111 2 2 6 1 2 32 66

112 2 2 6 1 35 3 67

113 2 2 6 1 40 41 61

114 2 2 6 1 26 27 60

115 2 2 6 1 6 58 69

116 2 2 6 1 59 7 70

117 2 2 6 1 48 55 65

118 2 2 6 1 54 48 65

119 2 2 6 1 70 46 74

120 2 2 6 1 46 69 75

121 2 2 6 1 17 16 78

122 2 2 6 1 23 22 79

123 2 2 6 1 56 61 82

124 2 2 6 1 60 57 83

125 2 2 6 1 5 71 87

126 2 2 6 1 71 5 86

127 2 2 6 1 5 13 86

128 2 2 6 1 8 5 87

129 2 2 6 1 29 18 90

130 2 2 6 1 21 38 91

131 2 2 6 1 43 62 94

132 2 2 6 1 62 44 95

133 2 2 6 1 13 78 86

134 2 2 6 1 78 16 86

135 2 2 6 1 23 79 87

136 2 2 6 1 79 8 87

137 2 2 6 1 51 47 60

138 2 2 6 1 47 51 61

139 2 2 6 1 50 28 73

140 2 2 6 1 39 49 72

141 2 2 6 1 42 43 94

142 2 2 6 1 44 45 95

143 2 2 6 1 53 41 63

144 2 2 6 1 26 52 64

145 2 2 6 1 63 42 94

146 2 2 6 1 45 64 95

147 2 2 6 1 14 6 84

148 2 2 6 1 7 25 85

149 2 2 6 1 6 69 84

150 2 2 6 1 70 7 85

151 2 2 6 1 77 10 82

152 2 2 6 1 11 77 83

153 2 2 6 1 51 77 82

154 2 2 6 1 77 51 83

155 2 2 6 1 10 9 92

156 2 2 6 1 12 11 93

157 2 2 6 1 28 81 88

158 2 2 6 1 80 39 89

159 2 2 6 1 56 40 61

160 2 2 6 1 27 57 60

161 2 2 6 1 18 73 90

162 2 2 6 1 72 21 91

163 2 2 6 1 62 53 94

164 2 2 6 1 52 62 95

165 2 2 6 1 9 80 92

166 2 2 6 1 81 12 93

167 2 2 6 1 53 63 94

168 2 2 6 1 64 52 95

169 2 2 6 1 82 10 92

170 2 2 6 1 11 83 93

171 2 2 6 1 69 68 84

172 2 2 6 1 68 70 85

173 2 2 6 1 40 56 89

174 2 2 6 1 57 27 88

175 2 2 6 1 81 57 88

176 2 2 6 1 56 80 89

177 2 2 6 1 46 68 69

178 2 2 6 1 68 46 70

179 2 2 6 1 80 56 92

180 2 2 6 1 57 81 93

181 2 2 6 1 56 82 92

182 2 2 6 1 83 57 93

183 2 2 6 1 1 26 64

184 2 2 6 1 31 2 66

185 2 2 6 1 3 36 67

186 2 2 6 1 41 4 63

187 2 2 6 1 27 28 88

188 2 2 6 1 28 29 90

189 2 2 6 1 73 28 90

190 2 2 6 1 29 30 58

191 2 2 6 1 37 38 59

192 2 2 6 1 38 39 91

193 2 2 6 1 39 40 89

194 2 2 6 1 39 72 91