第二次大作业

利用Laplace方程生成网格

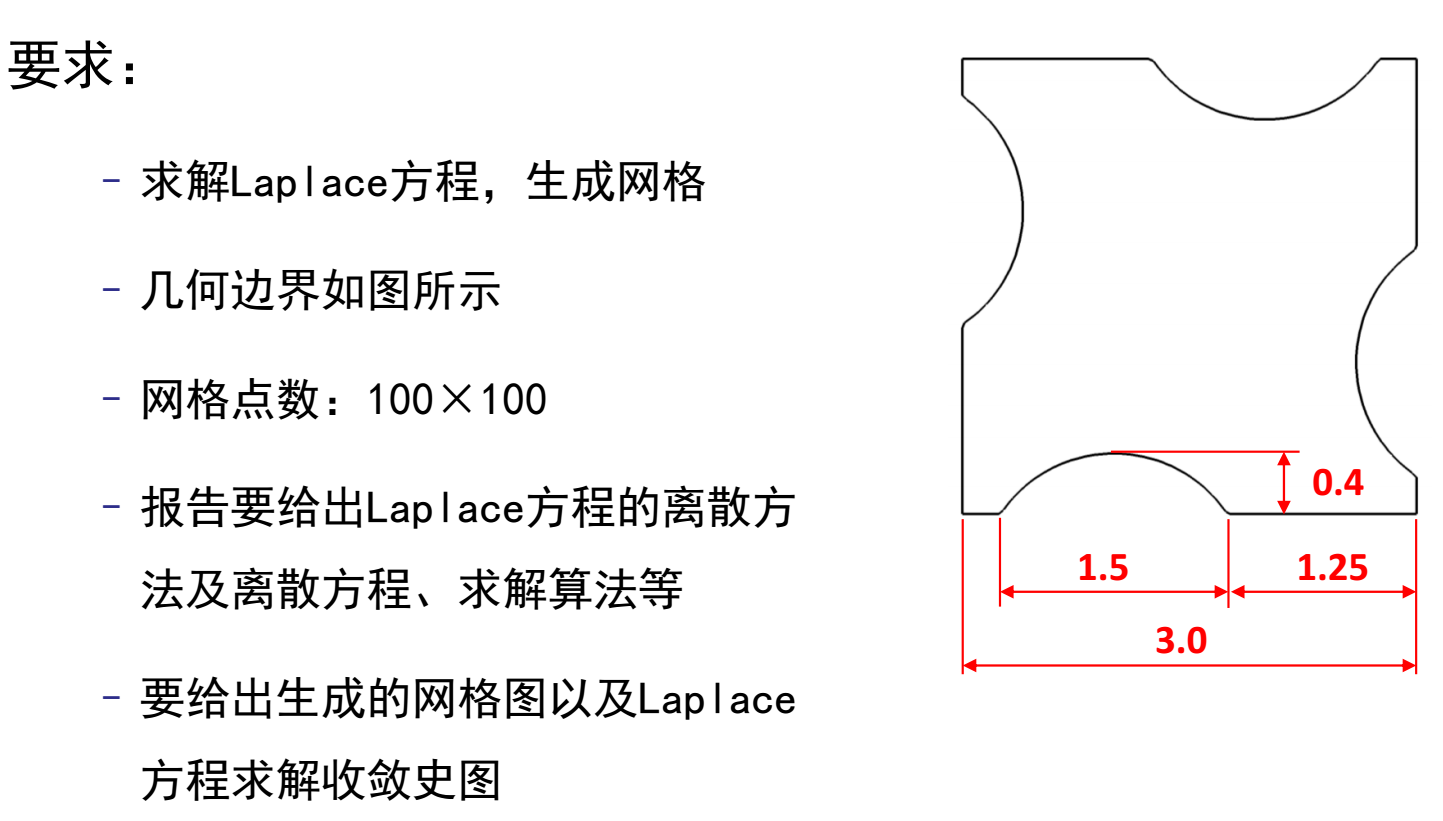


姓名：李春蕾

学号：SY1804410

指导教师：宁方飞

## 要求



## 算法

首先给定边界网格坐标，之后给出内点网格坐标的初始估计值，对于本模型可知需进行分段说明计算。首先对于下边界有

 *（2.1）*

对于上边界有

 *（2.2）*

对于左侧边界有

 *（2.3）*

对于右侧边界有

 *（2.4）*

对于内点，直接利用均布的分布方法，即将内点设置为在*x,y*坐标上均匀分布。此时可得到有

 *（2.5）*

初始网格点设置完成后，利用*Laplace*方程作为网格光顺的出发方程，由于*Laplace*方程是椭圆型方程，之间的变换是光滑的且一一对应的。*Laplace*方程为

 *（2.6）*

进行微分方程变换得到

 *（2.7）*

其中

 *（2.8）*

若物理平面边界点确定如下：

 *（2.9）*

则进行方程离散可得到

 *（2.10）*

其中系数的离散结果为



 *（2.11）*



经过转化可得到

 *（2.12）*

其中



 *（2.13）*

当物理平面求解域边界的网格点坐标已经确定时，式*（2.12）*是封闭的，利用迭代法可以解出。利用已确定的网格内点，依次迭代求解得出下一步的网格点坐标。

 *（2.14）*

其中称为的第*k*次迭代值。当（为给定满足残差的值）时，我们认为迭代收敛，由此得到了贴体网格。

## 编程

### 3.1 需求分析

#### 需求A

根据算法，最终画出网格图，并画出残差收敛图。

#### 需求B

将源代码存放到多个源文件中，并使用.H文件作为通用的接口。每个源文件实现一个功能模块。每个功能模块主体为一个函数。采用函数式编程。

#### 需求C

在main函数中控制整体循环。各个功能模块主体函数调用需要的参数应该尽量少，以保证操作的简易性。

#### 需求D

需要限定收敛条件，满足条件则判断为收敛。并且应该限定最大循环次数，超出循环次数仍未收敛则认为未收敛。收敛后应该立即停止循环，并输出残差图和最终网格图。

#### 需求E

流场节点数组应该为一个结构体变量。结构体包含x与y坐标。该数组应该能够由所有函数（不同源文件中的）共享。因此设定为全局结构体数组。在头文件中利用extern关键字声明，以添加到每个源文件。在main.cpp文件中定义。

#### 需求F

几何参数应该设定为常量，并且也应该由所有函数共享。因此定义在头文件中。数组大小和空间步长也应该设定为可共享的常量。

#### 需求G

横向与竖向的节点数应该没有关联性。即可以采取不同的数目。空间步长也应该由此自动计算出来。横向与竖向的节点数应该可以改变。即不限定为100\*100。

#### 需求H

在圆弧段，应该保证空间步长仍不变，即按弧长划分节点。

#### 需求I

应当保证程序的易读性，在不同的小功能模块间用注释划分，并注明所实现的功能。在一个大功能模块（即一个.cpp文件）中，最开始处要说明所实现的功能，并给出编译顺序。编写Makefile文件保证编译顺序的正确性。

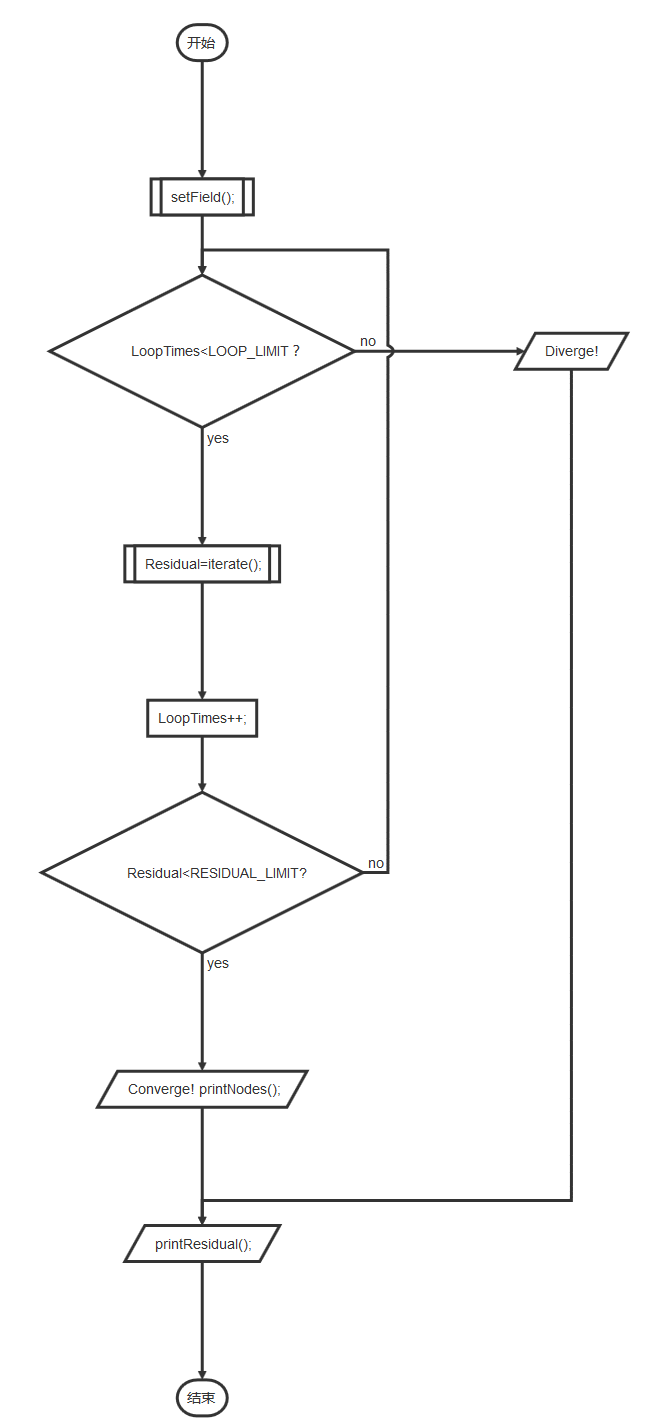
### 3.2 方案设计

#### 模块关系思维导图

本程序采用模块化，函数式编程。共分为四个.cpp文件和一个.H文件。.H文件包含函数和全局变量的声明，以及常量的定义。每个.cpp文件实现一个大功能模块，每个大功能模块下又利用注释划分小功能模块。Main函数为程序入口，并控制整体循环。各模块的关系如该思维导图所示。

#### main函数标准流程图

main()函数的流程图如下所示。



### 3.3 测试与调试

运行与测试环境为Linux(Ubantu16.04)虚拟机，虚拟机软件为VMware。编译器为gcc，调试器为gdb。文本编辑器为Visual Stdio Code。主机处理器为6\*2核，i7-5820K, 主频3.3GHz，虚拟机实际使用4\*2核。16G内存。

编写了Makefile文件来进行编译。编写了Allrun.sh文件来更改参数重复运行程序九次。 编写了seize.sh文件来抓取log输出运算时间，最终残差，循环次数等 。

残差收敛标准设定为1e-5， 最大循环次数设定为1.5e4

进行了9组测试，Mx\*My分别为：

100\*100, 100\*200, 100\*300

200\*100, 200\*200, 200\*300

300\*100, 300\*200, 300\*300

运算时间、最终残差与循环次数分别为

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.9906e-06

Loop times = 1670

Mx= 100 My= 100

Wall time = 4.58935 s

CPU time= 4.58936e+06 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99089e-06

Loop times = 3397

Mx= 100 My= 200

Wall time = 15.3069 s

CPU time= 1.5307e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99809e-06

Loop times = 5446

Mx= 100 My= 300

Wall time = 28.9463 s

CPU time= 2.89463e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99823e-06

Loop times = 3377

Mx= 200 My= 100

Wall time = 16.6037 s

CPU time= 1.66037e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99801e-06

Loop times = 4413

Mx= 200 My= 200

Wall time = 40.0099 s

CPU time= 4.00099e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99962e-06

Loop times = 6165

Mx= 200 My= 300

Wall time = 67.443 s

CPU time= 6.7443e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99927e-06

Loop times = 5425

Mx= 300 My= 100

Wall time = 37.2125 s

CPU time= 3.72125e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99712e-06

Loop times = 6130

Mx= 300 My= 200

Wall time = 71.9792 s

CPU time= 7.19792e+07 s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Final residual =9.99864e-06

Loop times = 7115

Mx= 300 My= 300

Wall time = 82.6659 s

CPU time= 8.26659e+07 s

## 结果

网格写入finalMesh.dat，残差图写入residual.plt。可视化工具为tecplot2015R1。在finalMesh.dat文件开头加入tecplot格式要求的文件头。如下所示（其中Mx与My分别是横向与纵向网格数，根据具体需要而改变

Title="sample mesh"

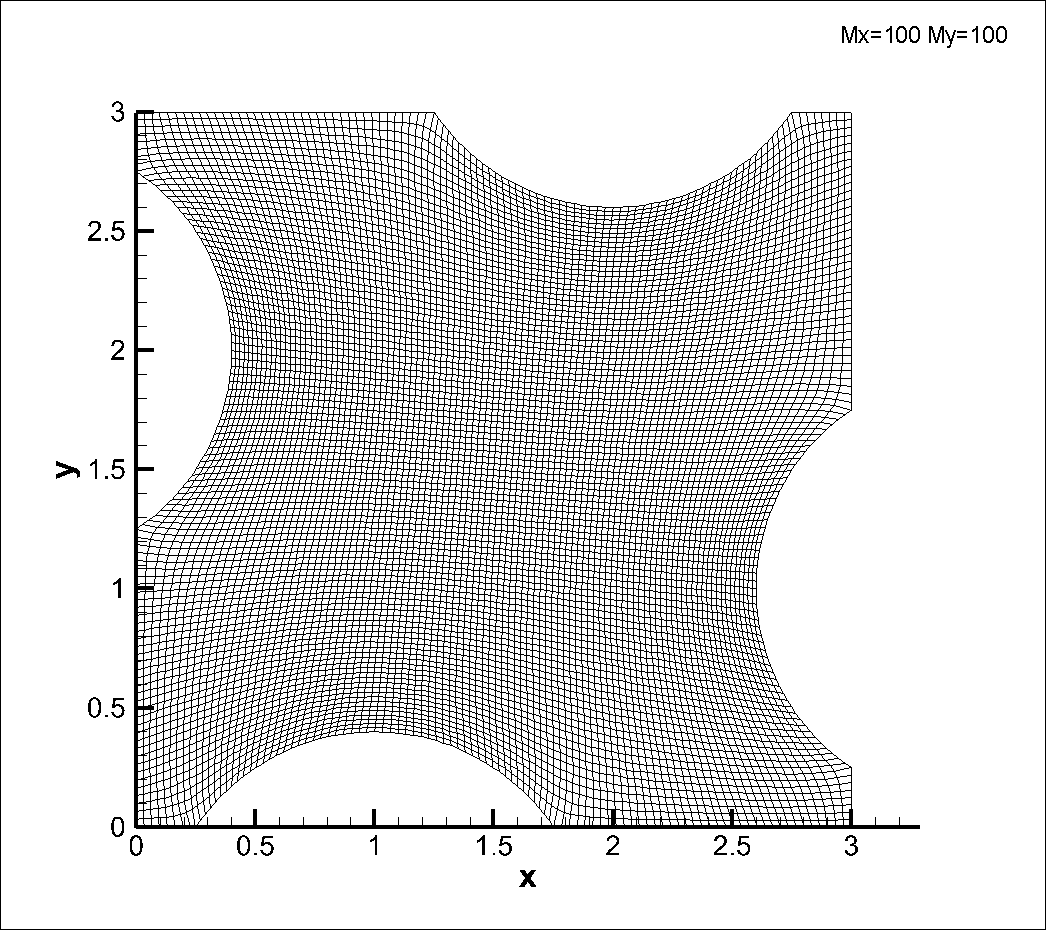
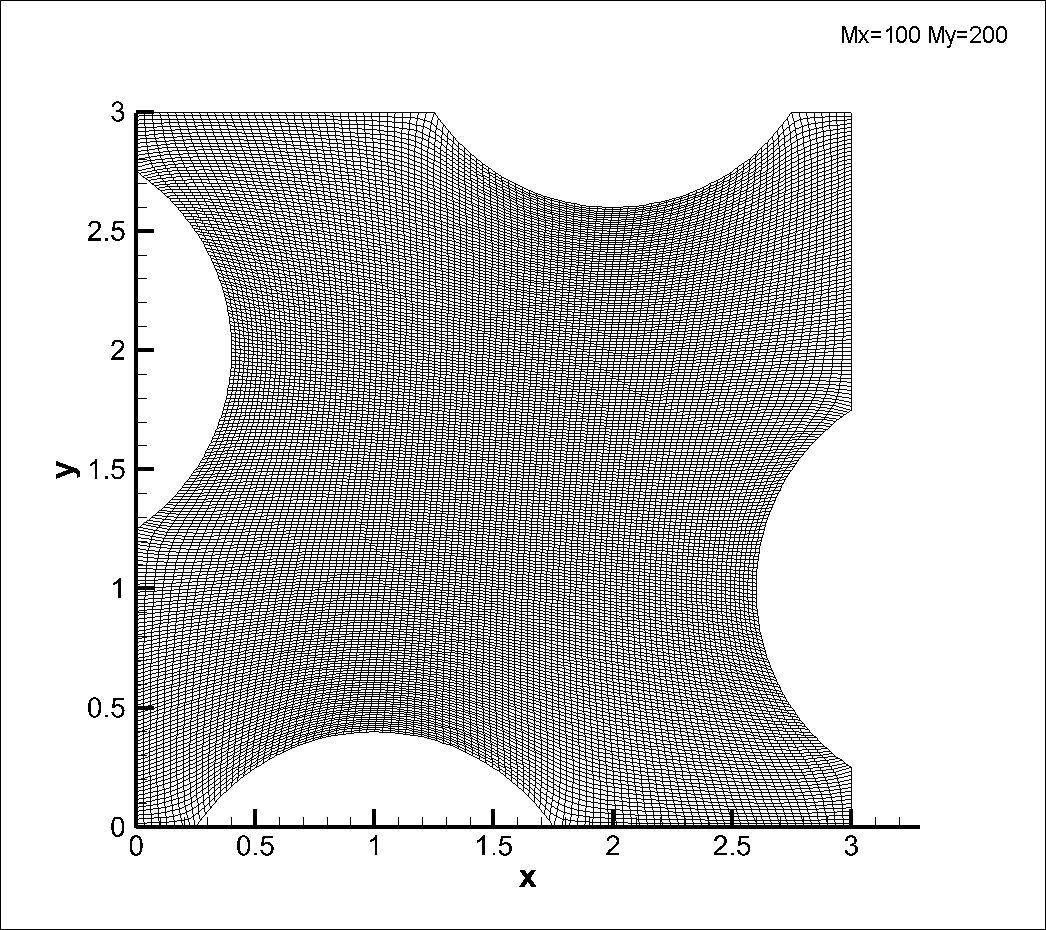
Variables="x","y"

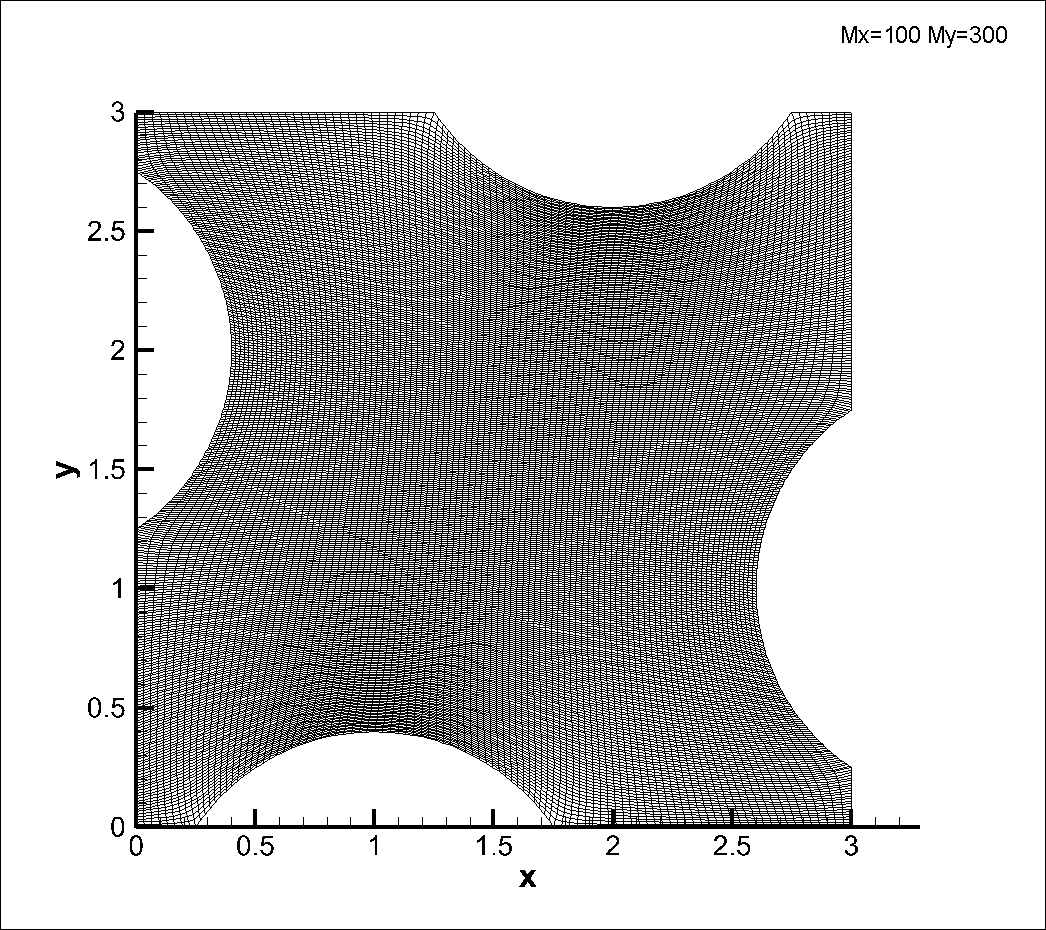
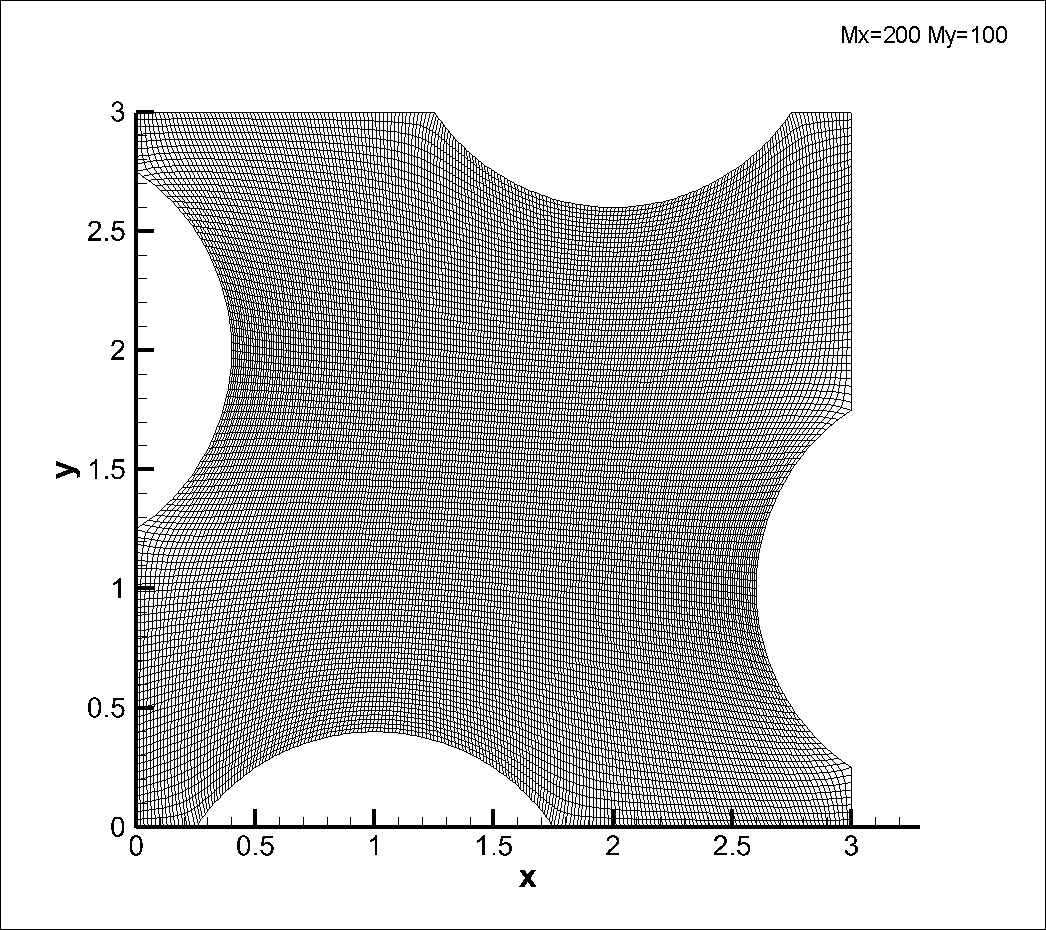
Zone i=Mx+1, j=My+1,f=point

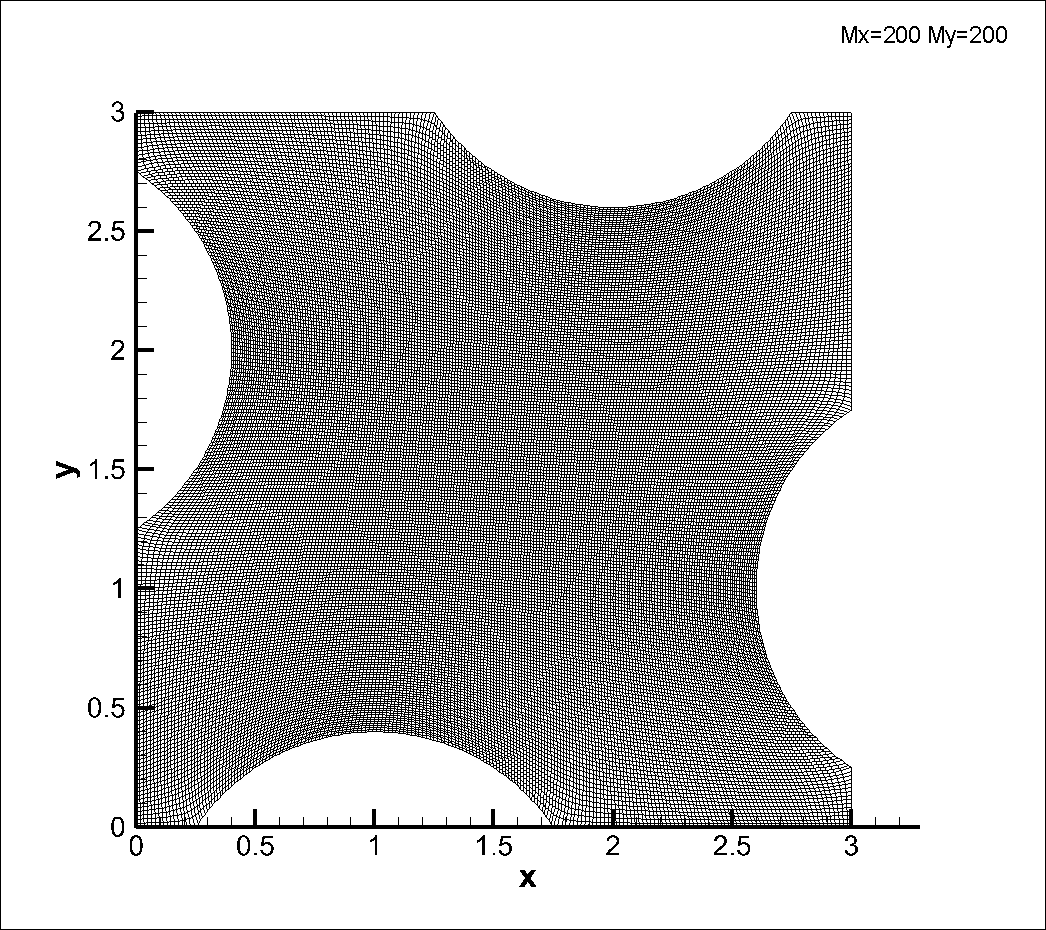
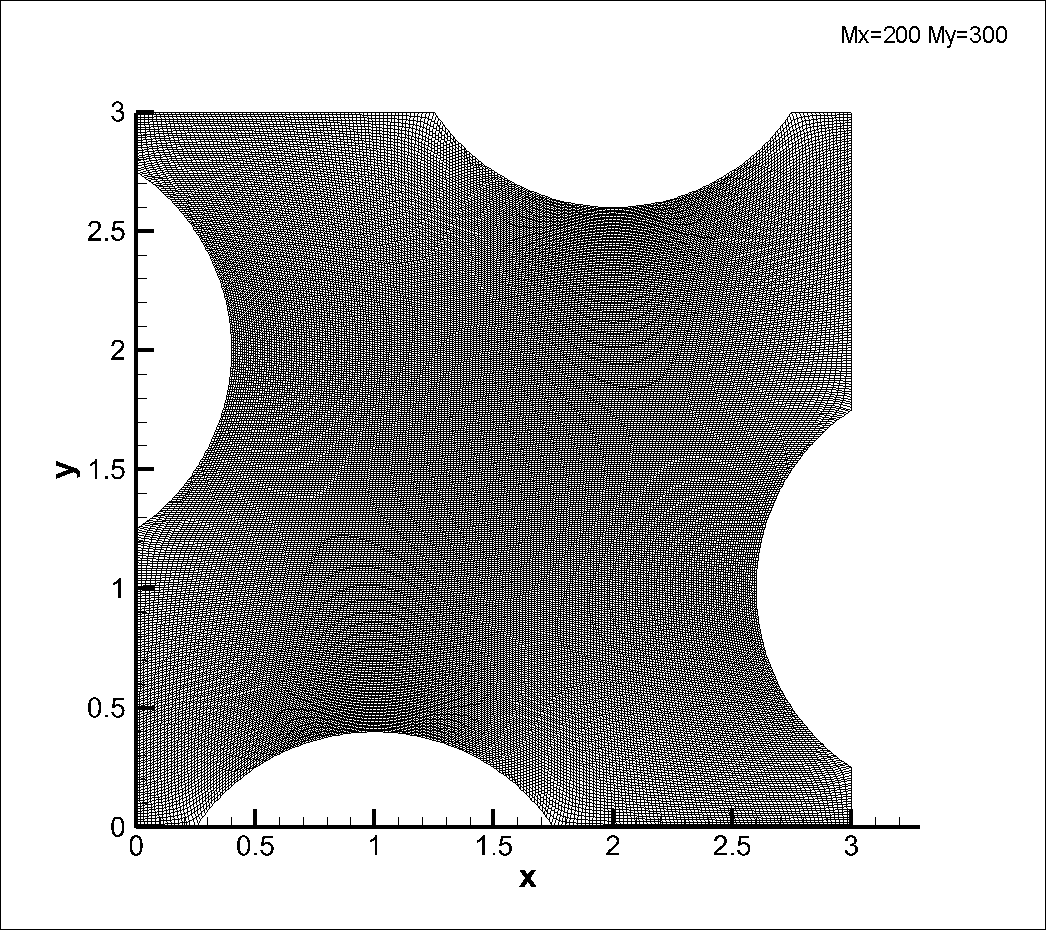
### 4.1 最终网格

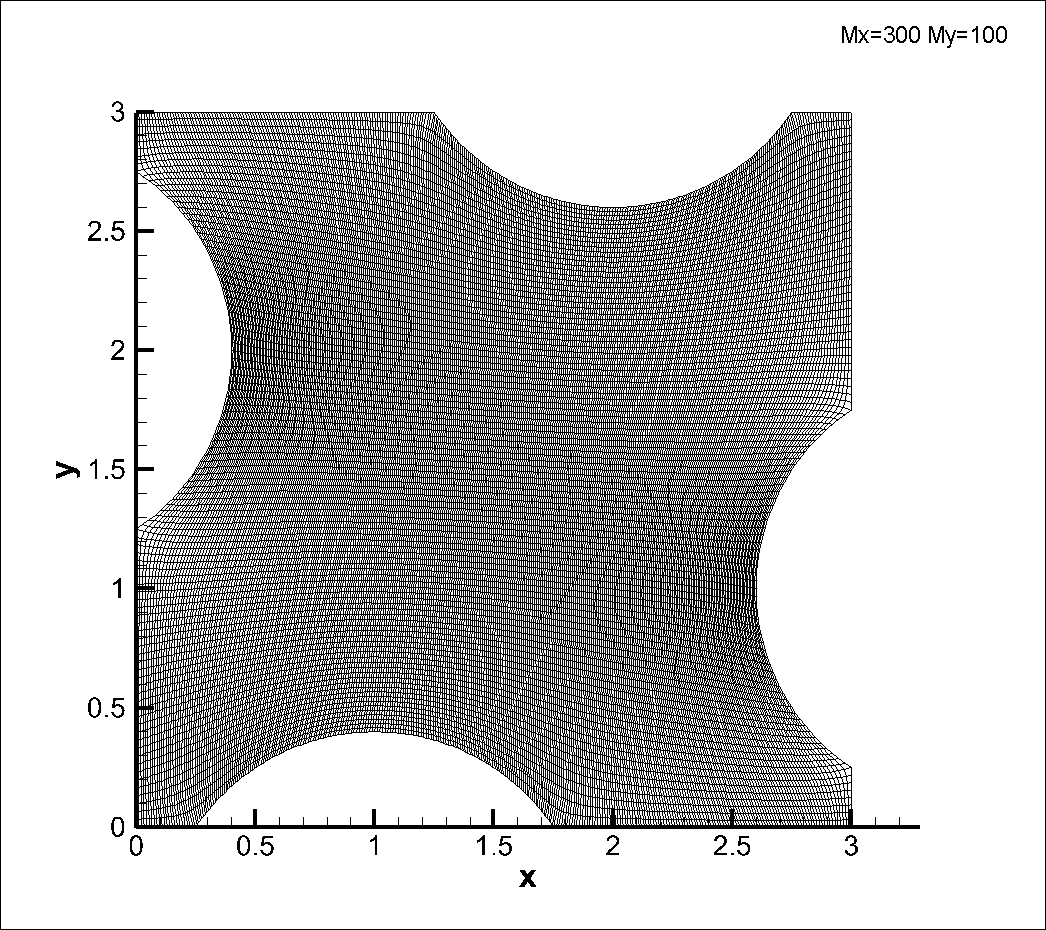
初场点阵图，Mx=100,My=200

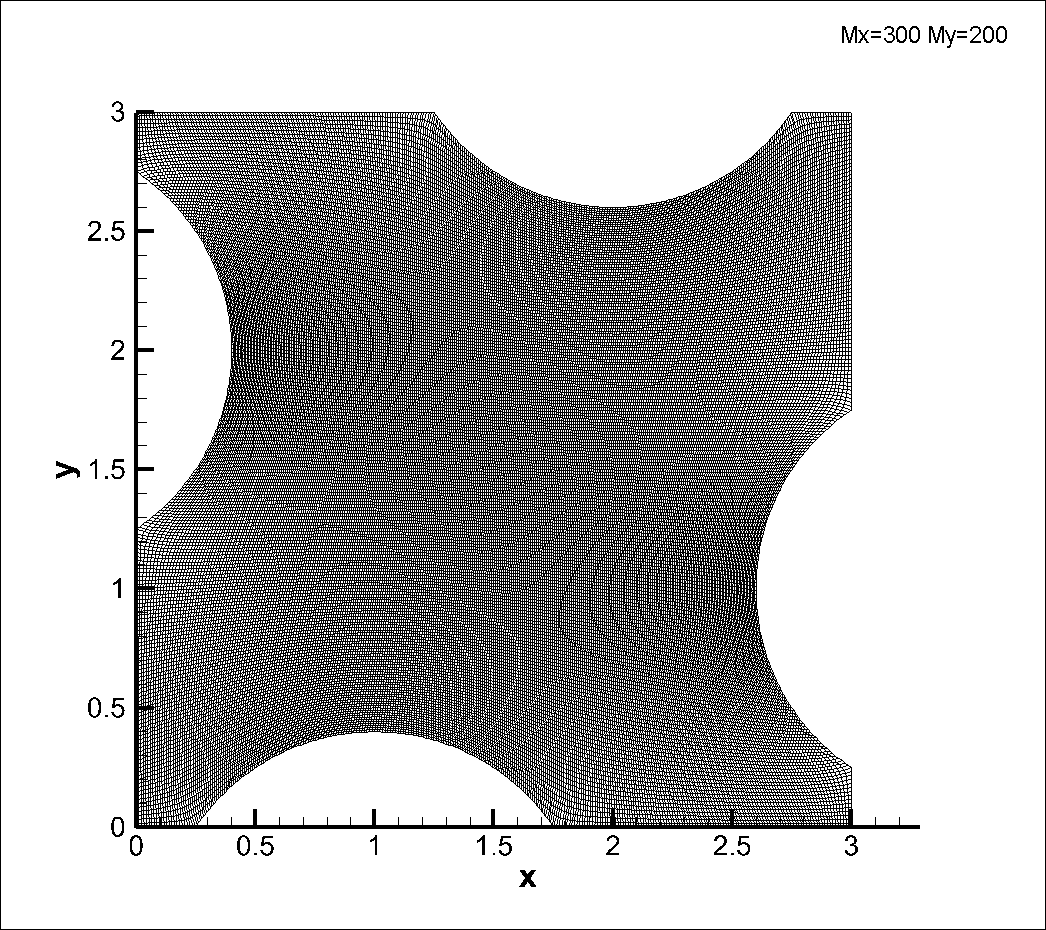
经过迭代计算最终得到的网格和残差图如下图所示

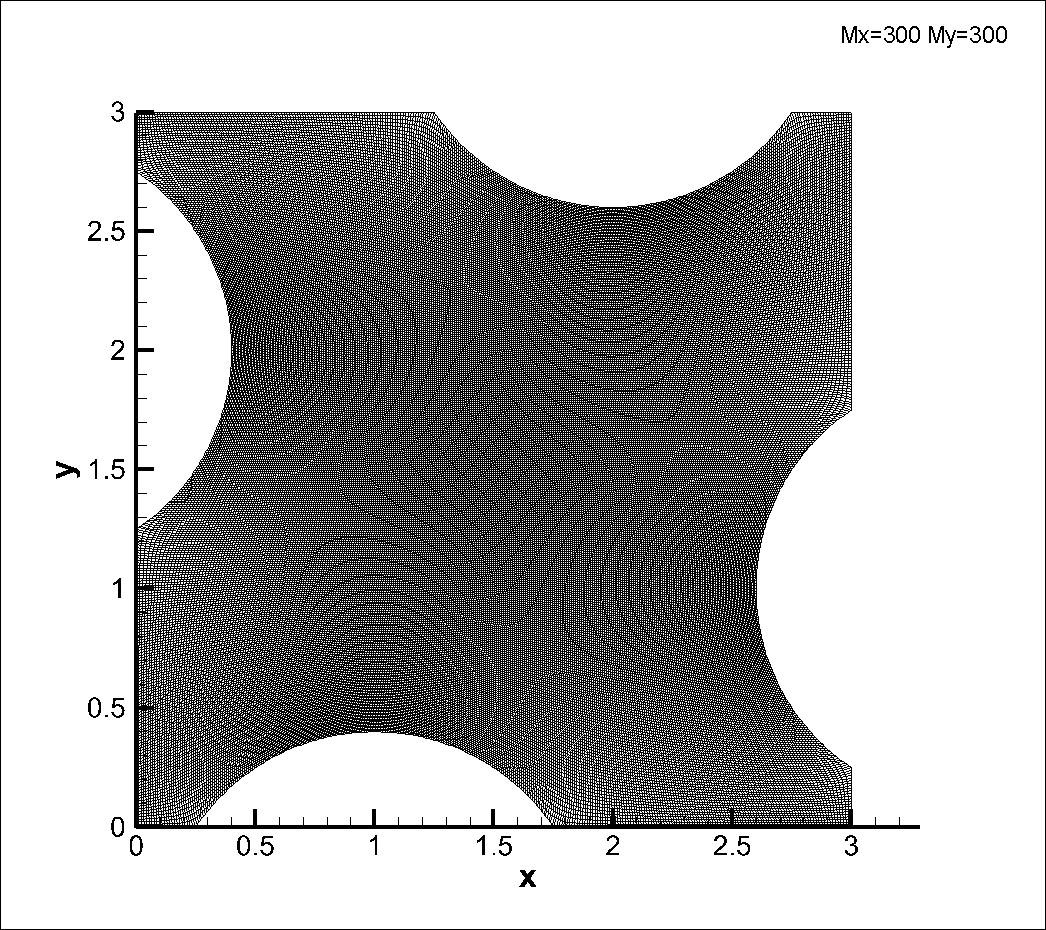




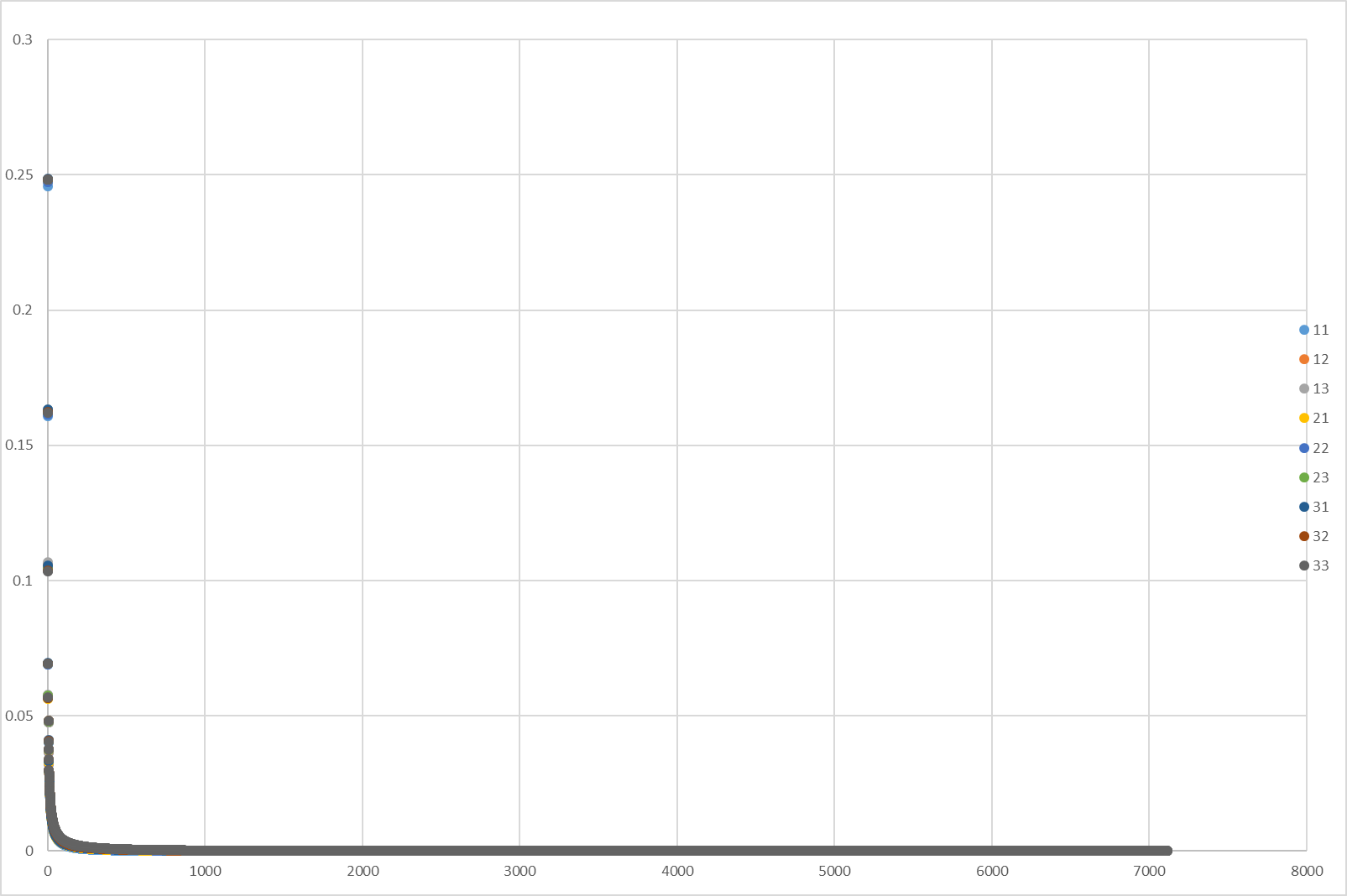








### 4.2 残差

可以大致看出，所有的残差都在最开始急剧下降然后缓慢下降。图中11,12等标签代表Mx=100 My=100; Mx=100, My=200 etc.

## 5. 源代码

### Main.H

#ifndef main\_H

#define main\_H

#include<iostream>

#include<fstream>

#include<cmath>

#include<string>

using namespace std;

void setField();

double iterate();

void printHead(ofstream &);

void printNodes(ofstream &);

//                      define the inline func                               //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

inline double max(const double a, const double b)

{

    return (a>=b)?a:b;

}

inline double sq(const double a)

{

    return pow(a, 2);

}

//                      declear the constants                                //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

    const int Mx=100, My=100;

    const double pitL=1.5, pitDepth=0.4, totalL=3.0, longL=1.25;

    const double shortL=totalL-longL-pitL;

    const double R=( sq(pitL/2)+ sq(pitDepth) ) / (2\*pitDepth);///R=0.903125

    const double arcL=2\*R\*asin( pitL/(2\*R) ), sqR=sq(R);///arcL=1.76997

    const double Oa=shortL+pitL/2, Ob=R-pitDepth;/// Oa=1, Ob=0.503125

    ///Oa & Ob stands for the x & y distance(Oa && Ob>0) of

    ///bottom Circile Center to the left bottom corner of the plane

    const double totalCurveL=arcL+longL+shortL;

    const double dKsi=totalCurveL/Mx, dEta=totalCurveL/My;///dKsi=dEta=0.0327

    const int NHShort=round(shortL/dKsi);///8

    const int NHLong =round(longL/dKsi);///38

    const int NHArc =Mx-NHShort-NHLong;///NHArc=54

    const int NVShort=round(shortL/dEta);

    const int NVLong =round(longL/dEta);

    const int NVArc =My-NVShort-NVLong;

//                      declear the globals                                  //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

struct Cnode

{

    double x,y;

};

extern struct Cnode node[Mx+1][My+1];

#endif

### Main.cpp

/\*---------------------------------------------------------------------------\*\

Description

    This File is the main file, which aims to control the loop and gives final

    results.

\\*---------------------------------------------------------------------------\*/

#include"main.H"

#include<cstdlib>

#include<ctime>

const int LOOP\_LIMIT=1e4;

const double RESIDUAL\_LIMIT=1e-5;

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

///definition of the global arries here

struct Cnode node[Mx+1][My+1];

int main()

{

    cout<<"\nSetting the initial field...\n";

    setField();

    cout<<"\nInitial field set over\n";

    cout<<"\nResidual limit= "<<RESIDUAL\_LIMIT<<endl;

    cout<<"Loop times limit = "<<LOOP\_LIMIT<<endl;

    cout<<"Number of horizentoal nodes(Mx+1)= "<<Mx+1<<endl;

    cout<<"Number of vertical nodes(My+1)= "<<My+1<<endl;

    cout<<"\nBegin iterate ? (y/n)\n";

    char c;

    cin>>c;

    if(c=='n')

    {

        exit(0);

    }

    ofstream foutResidual("data/residual.plt");

    ofstream foutFinalMesh("data/finalMesh.dat");

    int loopTimes=0;

    double currentResidual;

    while(loopTimes<LOOP\_LIMIT)

    {

        currentResidual=iterate();

        foutResidual<<currentResidual<<endl;

        loopTimes++;

        cout<<"loopTimes= "<<loopTimes

            <<"\tResidual= "<<currentResidual<<endl;

        if(currentResidual<RESIDUAL\_LIMIT)

        {

            cout<<"\nConverged! \nFinal residual ="<<currentResidual

            <<"\nLoop times = "<<loopTimes<<endl

            <<"Final field has been written into \"finalMesh.dat\" \n"

            <<"History of residual has been written into \"residual.plt\"\n";

            printNodes(foutFinalMesh);

            break;

        }

    }

    if(loopTimes>=LOOP\_LIMIT)

    {

        cout<<"\nFail to converge! in "<<loopTimes<<" times\n"

        <<"Final residual ="<<currentResidual<<"\n\n\n";

    }

    cout<<"\nMx= "<<Mx<<"\tMy= "<<My<<endl;

    cout<<"\nWall time = "<<(double)clock()/CLOCKS\_PER\_SEC<<" s"<<endl;

    cout<<"CPU time= "<<(double)clock()<<" s"<<endl;

return 0;

}

### setField.cpp

/\*---------------------------------------------------------------------------\*\

Description

This File is to set the initial fields, including boundaries.

It should be comipled secondly.

\\*---------------------------------------------------------------------------\*/

#include"main.H"

void setInternal();

void setBoundaries();

void setLeft(const int N1=NVLong, const int N2=NVArc);

void setRight(const int N1=NVShort, const int N2=NVArc);

void setBottom(const int N1=NHShort, const int N2=NHArc);

void setTop(const int N1=NHLong, const int N2=NHArc);

static void printBoundaries(ofstream &fout, string flag);

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

const double theta0=(M\_PI-arcL/R)/2;

double dx, dy;

const double dThetaH=(arcL/R)/NHArc;///NHArc=NVArc=54

const double dThetaV=(arcL/R)/NVArc;

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

void setField()

{

setBoundaries();

ofstream foutBoundary("data/boundary.dat");

printBoundaries(foutBoundary,"all");

setInternal();

ofstream foutInit("data/InitMesh.dat");

printNodes(foutInit);

cout<<"\nBoundary is written in \"boundary.dat\""<<endl;

cout<<"Initial mesh is written in \"initMesh.dat\""<<endl;

}

// Set the Boundaries //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

void setBoundaries()

{

setLeft();

setRight();

setBottom();

setTop();

}

void setLeft(const int N1, const int N2)

{

const int i=0, M=My;

const double Ox=-Ob, Oy=totalL-Oa, L1=longL, L2=pitL;

double d1=L1/N1, d2=arcL/N2, d3= (totalL-L1-L2)/(M-N1-N2);

int j;

for(j=0;j<N1;j++)

{

node[i][j].x=0;

node[i][j].y=d1\*j;

}

int n;

for(j=N1;j<N1+N2;j++)

{

n=j-N1;

dx=R\*sin(theta0+n\*dThetaV);

dy=R\*cos(theta0+n\*dThetaV);

dy\*=-1;///it depends on where is Circle center relative to

///the first point of arc. Simply put, if first point is

///on the left, down side of the Center , dx\*=-1,dy\*=-1;

node[i][j].x=Ox+dx;

node[i][j].y=Oy+dy;

}

for(j=N1+N2;j<=M;j++)

{

n=j-N1-N2;

node[i][j].x=0;

node[i][j].y=L1+L2+d3\*n;

}

}

void setRight(const int N1, const int N2)

{

const int i=Mx, M=My;

const double Ox=totalL+Ob, Oy=Oa, L1=shortL, L2=pitL;

int j;

for(j=0;j<N1;j++)

{

node[i][j].x=totalL;

node[i][j].y=dEta\*j;

}

int n;

for(j=N1;j<N1+N2;j++)

{

n=j-N1;

dx=R\*sin(theta0+n\*dThetaV);

dy=R\*cos(theta0+n\*dThetaV);

dx\*=-1;

dy\*=-1;

node[i][j].x=Ox+dx;

node[i][j].y=Oy+dy;

}

for(j=N1+N2;j<=M;j++)

{

n=j-N1-N2;

node[i][j].x=totalL;

node[i][j].y=L1+L2+dEta\*n;

}

}

void setBottom(const int N1, const int N2)

{

const int j=0, M=Mx;

const double Ox=Oa, Oy=-Ob, L1=shortL, L2=pitL;

double d3= (totalL-L1-L2)/(M-N1-N2);

int i;

for(i=0;i<N1;i++)

{

node[i][j].x=dKsi\*i;

node[i][j].y=0;

}

int n;

for(i=N1;i<N1+N2;i++)

{

n=i-N1;

dx=R\*cos(theta0+n\*dThetaH);///sin/Cos is opposite to Left&Right

dy=R\*sin(theta0+n\*dThetaH);

dx\*=-1;

node[i][j].x=Ox+dx;

node[i][j].y=Oy+dy;

}

for(i=N1+N2;i<=M;i++)

{

n=i-N1-N2;

node[i][j].x=L1+L2+d3\*n;

node[i][j].y=0;

}

}

void setTop(const int N1, const int N2)

{

const int j=My, M=Mx;

const double Ox=longL+pitL/2, Oy=totalL+Ob, L1=longL, L2=pitL;

double d3= (totalL-L1-L2)/(M-N1-N2);

int i;

for(i=0;i<N1;i++)

{

node[i][j].x=dKsi\*i;

node[i][j].y=totalL;

}

int n;

for(i=N1;i<N1+N2;i++)

{

n=i-N1;

dx=R\*cos(theta0+n\*dThetaH);///sin/Cos is opposite to Left&Right

dy=R\*sin(theta0+n\*dThetaH);

dx\*=-1;

dy\*=-1;

node[i][j].x=Ox+dx;

node[i][j].y=Oy+dy;

}

for(i=N1+N2;i<=M;i++)

{

n=i-N1-N2;

node[i][j].x=L1+L2+d3\*n;

node[i][j].y=totalL;

}

}

// Set the internal fields //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

void setInternal()

{

int i,j;

for(i=1;i<Mx;i++)

for(j=1;j<My;j++)

{

node[i][j].x=pitDepth+dKsi\*(totalL-2\*pitDepth)/totalCurveL\*i;

node[i][j].y=pitDepth+dEta\*(totalL-2\*pitDepth)/totalCurveL\*j;

}

}

// Print the Boundaries //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

static void printBoundaries(ofstream &fout, string flag)

{

printHead(fout);

// fout<<"flag= "<<flag<<endl;

if(!(flag.compare("left")\*flag.compare("all")))

{

// fout<<"begin print left"<<endl;

const int i=0;

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

fout<<node[i][j].x<<'\t'<<node[i][j].y<<endl;

// fout<<"\nend print left\n\n"<<endl;

}

if(!(flag.compare("right")\*flag.compare("all")))

{

// fout<<"begin print right"<<endl;

const int i=Mx;

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

fout<<node[i][j].x<<'\t'<<node[i][j].y<<endl;

// fout<<"\nend print right\n\n"<<endl;

}

if(!(flag.compare("right")\*flag.compare("all")))

{

// fout<<"begin print bottom"<<endl;

const int j=0;

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

{

fout<<node[i][j].x<<'\t'<<node[i][j].y<<endl;

}

// fout<<"\nend print bottom\n\n"<<endl;

}

if(!(flag.compare("right")\*flag.compare("all")))

{

// fout<<"begin print top"<<endl;

const int j=My;

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

{

fout<<node[i][j].x<<'\t'<<node[i][j].y<<endl;

}

// fout<<"\nend print top\n\n"<<endl;

}

}

/\*---------------------------------------------------------------------------\*\

Description

This File is to iterate the loop once. It contains the core algorithm.

\\*---------------------------------------------------------------------------\*/

#include"main.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

double iterate()

{

double alpha, beta, gamma, bw, be, bs, bn, bp, cpx, cpy;

double r, rMax=0.0;

int i,j;

for(i=1;i<Mx;i++)

{

for(j=1;j<My;j++)

{

// Calculate the coefficients //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

alpha= sq

(

( node[i][j+1].x - node[i][j-1].x ) / ( 2\*dEta )

)

+ sq

(

( node[i][j+1].y - node[i][j-1].y ) / ( 2\*dEta )

);

beta= ( node[i+1][j].x - node[i-1][j].x ) / ( 2\*dKsi )

\* ( node[i][j+1].x - node[i][j-1].x ) / ( 2\*dEta )

+

( node[i+1][j].y - node[i-1][j].y ) / ( 2\*dKsi )

\* ( node[i][j+1].y - node[i][j-1].y ) / ( 2\*dEta );

gamma= sq

(

( node[i+1][j].x - node[i-1][j].x ) / ( 2\*dKsi )

)

+ sq

(

( node[i+1][j].y - node[i-1][j].y ) / ( 2\*dKsi )

);

bw= be= alpha/(dKsi\*dKsi);

bs= bn= gamma/(dEta\*dEta);

bp=bw+be+bs+bn;

cpx= - beta

\*(

node[i+1][j+1].x

-node[i+1][j-1].x

-node[i-1][j+1].x

+node[i-1][j-1].x

)

/ (2\*dKsi\*dEta);

cpy= - beta

\*(

node[i+1][j+1].y

-node[i+1][j-1].y

-node[i-1][j+1].y

+node[i-1][j-1].y

)

/ (2\*dKsi\*dEta);

// Calculate the new nodes value //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

double x\_new=(

bw\*node[i-1][j].x

+ be\*node[i+1][j].x

+ bs\*node[i][j-1].x

+ bn\*node[i][j+1].x

+ cpx

) / bp ;

double y\_new=(

bw\*node[i-1][j].y

+ be\*node[i+1][j].y

+ bs\*node[i][j-1].y

+ bn\*node[i][j+1].y

+ cpy

) / bp ;

// calculate the residual //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

r=max

(

fabs( x\_new - node[i][j].x ) ,

fabs( y\_new - node[i][j].y )

);

if(r>rMax)

rMax=r;

// update the nodes, move to next //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

node[i][j].x=x\_new;

node[i][j].y=y\_new;

}

}

return rMax;

}

/\*---------------------------------------------------------------------------\*\

Description

This File is to define functions to print out data.

It should be compiled firstly.

\\*---------------------------------------------------------------------------\*/

#include"main.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

void printHead(ofstream &fout)

{

fout

<<"Title=\"Mesh\""<<endl

<<"Variables=\"x\",\"y\""<<endl

<<"Zone i="<<Mx+1<<", j="<<My+1<<", f=point"<<endl;

}

void printNodes(ofstream &fout)

{

printHead(fout);

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

{

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

{

fout<<node[i][j].x<<'\t'<<node[i][j].y<<endl;

// fout<<"\t\t["<<i<<"]"<<"["<<j<<"]"<<endl;

}

}

}

### Iterate.cpp

/\*---------------------------------------------------------------------------\*\

Description

This File is to iterate the loop once. It contains the core algorithm.

\\*---------------------------------------------------------------------------\*/

#include"main.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

double iterate()

{

double alpha, beta, gamma, bw, be, bs, bn, bp, cpx, cpy;

double r, rMax=0.0;

int i,j;

for(i=1;i<Mx;i++)

{

for(j=1;j<My;j++)

{

// Calculate the coefficients //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

alpha= sq

(

( node[i][j+1].x - node[i][j-1].x ) / ( 2\*dEta )

)

+ sq

(

( node[i][j+1].y - node[i][j-1].y ) / ( 2\*dEta )

);

beta= ( node[i+1][j].x - node[i-1][j].x ) / ( 2\*dKsi )

\* ( node[i][j+1].x - node[i][j-1].x ) / ( 2\*dEta )

+

( node[i+1][j].y - node[i-1][j].y ) / ( 2\*dKsi )

\* ( node[i][j+1].y - node[i][j-1].y ) / ( 2\*dEta );

gamma= sq

(

( node[i+1][j].x - node[i-1][j].x ) / ( 2\*dKsi )

)

+ sq

(

( node[i+1][j].y - node[i-1][j].y ) / ( 2\*dKsi )

);

bw= be= alpha/(dKsi\*dKsi);

bs= bn= gamma/(dEta\*dEta);

bp=bw+be+bs+bn;

cpx= - beta

\*(

node[i+1][j+1].x

-node[i+1][j-1].x

-node[i-1][j+1].x

+node[i-1][j-1].x

)

/ (2\*dKsi\*dEta);

cpy= - beta

\*(

node[i+1][j+1].y

-node[i+1][j-1].y

-node[i-1][j+1].y

+node[i-1][j-1].y

)

/ (2\*dKsi\*dEta);

// Calculate the new nodes value //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

double x\_new=(

bw\*node[i-1][j].x

+ be\*node[i+1][j].x

+ bs\*node[i][j-1].x

+ bn\*node[i][j+1].x

+ cpx

) / bp ;

double y\_new=(

bw\*node[i-1][j].y

+ be\*node[i+1][j].y

+ bs\*node[i][j-1].y

+ bn\*node[i][j+1].y

+ cpy

) / bp ;

// calculate the residual //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

r=max

(

fabs( x\_new - node[i][j].x ) ,

fabs( y\_new - node[i][j].y )

);

if(r>rMax)

rMax=r;

// update the nodes, move to next //

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

node[i][j].x=x\_new;

node[i][j].y=y\_new;

}

}

return rMax;

}

### Print.cpp

/\*---------------------------------------------------------------------------\*\

Description

This File is to define functions to print out data.

It should be compiled firstly.

\\*---------------------------------------------------------------------------\*/

#include"main.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

void printHead(ofstream &fout)

{

fout

<<"Title=\"Mesh\""<<endl

<<"Variables=\"x\",\"y\""<<endl

<<"Zone i="<<Mx+1<<", j="<<My+1<<", f=point"<<endl;

}

void printNodes(ofstream &fout)

{

printHead(fout);

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

{

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

{

fout<<node[i][j].x<<'\t'<<node[i][j].y<<endl;

// fout<<"\t\t["<<i<<"]"<<"["<<j<<"]"<<endl;

}

}

}

### 辅助脚本：Makefile

OBJS = setField.o print.o iterate.o main.o

SRCS = setField.C print.C iterate.C main.C

LIBS = -lm

EXE = out

main : $(OBJS) main.H

g++ -o ${EXE} $(OBJS) $(LIBS)

print.o : print.C main.H

g++ -g -c print.C

setField.o : setField.C main.H

g++ -g -c setField.C

iterate.o : iterate.C setField.C main.H

g++ -g -c iterate.C

main.o : $(SRCS) main.H

g++ -g -c main.C

.PHONY:clean clean\_data clean\_log clean\_all

clean :

rm -rf \*.o

clean\_data :

rm -rf data/\*

clean\_log :

rm -rf log/\*

clean\_all :

rm -rf data/\* log/\* \*.o ${EXE}

### 辅助脚本：Allrun.sh

mkdir log

for Mx in 100 200 300

do

for My in 100 200 300

do

sed -i "s/Mx=..., My=.../Mx=${Mx}, My=${My}/g" main.H

make

./out | tee log/log${Mx}\*${My}

mv data/residual.dat data/residual${Mx}\*${My}.dat

mv data/finalMesh.dat data/finalMesh${Mx}\*${My}.dat

done

done

### 辅助脚本：seize.sh

rm seize.log

for Mx in 100 200 300

do

for My in 100 200 300

do

tail -n 10 log${Mx}\*${My} >>seize.log

done

done

sed -i '/has been/'d seize.log

sed -i 's/Converged!/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/g' seize.log