**流函数反问题程序求解**

张睿谦 张浩 丁旺

1.问题描述

，

BC. 

2.离散格式

采用TS-ADI方法

第一步：



第二步：



3.计算结果

m=60,n=40时，计算结果如下图所示



4.C++源程序

#include<iostream>

#include<fstream>

#include<cmath>

#include <iomanip>

using namespace std;

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\*m + j] \* y[j];

}

y[i] = (r[i] - sum) / A[i\*m + i];

}

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

//cout << N\_2<<'\t';

} while (N\_2> error);

delete[] y\_old;

y\_old = NULL;

}//高斯-赛德尔迭代法

int main()

{

int m, n;//横纵网格数

cout << "请输入横竖坐标网格数：";

cin >> m >> n;

double delt\_Xi = 4.0 / m;

double delt\_phi = 2.0 / n;//网格的长度和宽度

double delt\_t;

cout << "请输入时间步长:";

cin >> delt\_t;

double \*Xi, \*phi;//横坐标与纵坐标

double \*y, \*Y\_old, \*A, \*r;//A为系数矩阵

int size = (m + 1)\*(n + 1);//系数矩阵大小

double y\_Xi, y\_phi, y\_Xi\_phi, y\_Xi\_Xi, y\_phi\_phi;//偏导数

Xi = new double[m+1];

phi = new double[n+1];

y = new double[size];

Y\_old = new double[size];

A = new double[size\*size];

r = new double[size];

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

{

Xi[j] = -4 + j\*delt\_Xi;

}//横坐标赋值[-4:0.1:0]

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

{

phi[i] = i\*delt\_phi;

}//纵坐标赋值[0:0.1:2]

//

//

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

{

y[i] = 0;

r[i] = 0;

}

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

{

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

{

A[i\*size + j] = 0;

}

}//矩阵赋初值

//

//

for (int j = 0.75\*m; j <= m; j++)

{

int i = 0;

y[i\*(m + 1) + j] = sqrt(1 - Xi[j] \* Xi[j]);

}//下边界

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

{

int i = n;

y[i\*(m + 1) + j] = 2;

}//上边界

for (int i = 1; i < n; i++)

{

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

{

y[i\*(m + 1) + j] = y[j]+(y[n\*(m+1)+j]-y[j])\*i/n;

}

}//插值得到场中间值

for (int j = 0; j < 0.75\*m; j++)

{

int i = 0;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1;

r[j] = 0;

}

for (int j = 0.75\*m; j <= m; j++)

{

int i = 0;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1;

r[i\*(m + 1) + j] = sqrt(1 - Xi[j] \* Xi[j]);

}//y[i,j] i=0 j [0 m] 下边界

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

{

int i = n;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1;

r[i\*(m + 1) + j] = 2;

}//y[i,j] i=n j [0 m] 上边界

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

{

int j = 0;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1;

r[i\*(m + 1) + j] = phi[i];

}//y[i,j] i [0 n] j=0 左边界

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

{

int j = m;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j - 1] = -1;

r[i\*(m + 1) + j] = 0;

}//y[i,j] i [1 n] j=m 右边界

double norm\_2;//2-范数

do

{

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

{

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

}

//将y赋值给Y\_old

for (int i = 1; i <= n-1; i++)

{

for (int j = 1; j <= m-1; j++)

{

y\_phi = 0.5\*(y[(i + 1)\*(m + 1) + j] - y[(i - 1)\*(m + 1) + j]) / delt\_phi;

y\_Xi = 0.5\*(y[i \*(m + 1) + j + 1] - y[i\*(m + 1) + j - 1]) / delt\_Xi;

y\_Xi\_phi = 0.25\*(y[(i + 1) \*(m + 1) + j + 1] - y[(i + 1) \*(m + 1) + j - 1] - y[(i - 1) \*(m + 1) + j + 1] + y[(i - 1) \*(m + 1) + j - 1]) / (delt\_phi\*delt\_Xi);

y\_phi\_phi = (y[(i + 1)\*(m + 1) + j] - 2 \* y[i \*(m + 1) + j] + y[(i - 1)\*(m + 1) + j]) / (delt\_phi\*delt\_phi);

//

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1 / delt\_t + 2 \* y\_phi\*y\_phi / (delt\_Xi\*delt\_Xi);

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j - 1] = -y\_phi\*y\_phi / (delt\_Xi\*delt\_Xi);

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j + 1] = -y\_phi\*y\_phi / (delt\_Xi\*delt\_Xi);

//

r[i\*(m + 1) + j] = y[i\*(m + 1) + j] / delt\_t - y\_phi\*y\_Xi\*y\_Xi\_phi;

}

}

//

Gauss\_Seidel(A, y, r, size, 1e-6);//解A\*y=r；

for (int i = 1; i <= n - 1; i++)

{

for (int j = 1; j <= m - 1; j++)

{

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 0;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j - 1] = 0;

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j + 1] = 0;

r[i\*(m + 1) + j] = 0;

}

}//清除A矩阵内容

for (int i = 1; i <= n-1; i++)

{

for (int j = 1; j <= m-1; j++)

{

y\_phi = 0.5\*(y[(i + 1)\*(m + 1) + j] - y[(i - 1)\*(m + 1) + j]) / delt\_phi;

y\_Xi = 0.5\*(y[i \*(m + 1) + j + 1] - y[i\*(m + 1) + j - 1]) / delt\_Xi;

y\_Xi\_phi = 0.25\*(y[(i + 1) \*(m + 1) + j + 1] - y[(i + 1) \*(m + 1) + j - 1] - y[(i - 1) \*(m + 1) + j + 1] + y[(i - 1) \*(m + 1) + j - 1]) / (delt\_phi\*delt\_Xi);

y\_Xi\_Xi = (y[i \*(m + 1) + j + 1] - y[i \*(m + 1) + j] + y[i \*(m + 1) + j - 1]) / (delt\_Xi\*delt\_Xi);

//

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 1 / delt\_t + 2 \* (1 + y\_Xi\*y\_Xi) / (delt\_phi\*delt\_phi);

A[(i\*(m + 1) + j)\*size + (i - 1)\*(m + 1) + j] = -(1 + y\_Xi\*y\_Xi) / (delt\_phi\*delt\_phi);

A[(i\*(m + 1) + j)\*size + (i + 1)\*(m + 1) + j] = -(1 + y\_Xi\*y\_Xi) / (delt\_phi\*delt\_phi);

//

r[i\*(m + 1) + j] = y[i\*(m + 1) + j] / delt\_t - y\_phi\*y\_Xi\*y\_Xi\_phi;

}

}

//

Gauss\_Seidel(A, y, r, size, 1e-6); //解A\*y=r;

for (int i = 1; i <= n - 1; i++)

{

for (int j = 1; j <= m - 1; j++)

{

A[(i\*(m + 1) + j)\*size + i\*(m + 1) + j] = 0;

A[(i\*(m + 1) + j)\*size + (i - 1)\*(m + 1) + j] = 0;

A[(i\*(m + 1) + j)\*size + (i + 1)\*(m + 1) + j] = 0;

r[i\*(m + 1) + j] = 0;

}

}//清除A矩阵内容

norm\_2 = Norm\_2(matrix\_subtract(y, Y\_old, size), size);

cout << norm\_2 << endl;

} while (norm\_2 > 1e-6);

ofstream Ofile("y\_result.txt");

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

{

Ofile << Xi[j] << '\t';

for (int i = 0; i <= 5; i++)

{

Ofile << setprecision(5) << y[n\*i\*(m + 1)/5 + j] << '\t';

}

cout << '\n';

}

Ofile.close();

delete[] A;

A = NULL;

delete[] Y\_old;

Y\_old = NULL;

delete[] r;

r = NULL;

delete[] y;

y = NULL;

delete[] Xi;

Xi = NULL;

delete[] phi;

phi = NULL;

return 0;

}