程序说明文件

12021108 梁星宇
12021111 林星竹

1 原理

1.1 微分方程的有限差分形式解决无内热元的稳态热传导问题

对于一个无内热源的稳态热传导问题，拉普拉斯方程可以表示为：

$$\nabla^2 T = 0$$

在二维的情况下，使用中心差分法对此方程进行离散化，我们得到：

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0$$

在本题中，$\Delta x = \Delta y$，方程可以简化为：

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} = 4T_{i,j}$$

因此每个节点的新温度

$$T_{i,j}^{new} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4}$$

故在迭代过程中，每个内部节点的新温度是上、下、左、右邻居温度的平均值，使用这个公式来更新每个内部节点的温度，直到整个网格的温度分布达到稳定状态，满足收敛条件。

1.2 程序原理

本程序基于 C++，利用简单的循环来完成迭代的过程。定义一个5×5的二维数组 `grids[5][5]`用于储存节点的温度值，如下：

```
double grids[5][5] = {
{150, 180, 200, 180, 150},
{180,   0,   0,   0, 180},
{200,   0,   0,   0, 200},
{180,   0,   0,   0, 180},
{150, 180, 200, 180, 150}
};
```

对于内部节点，初始化温度为 0，使用循环依次更新节点的温度，取周围四个温度的平均值，依据高斯-赛德尔迭代，每次计算后都立即更新节点的温度，参与下一个节点的温度计算。循环过程如下：

```
for (int iter = 0; iter < MAX_ITERATIONS && !converged; ++iter) {
    converged = true;
    // 使用高斯-塞德尔迭代法更新内部节点温度
    for (int i = 1; i < 5 - 1; ++i) {
        for (int j = 1; j < 5 - 1; ++j) {
            double T_old = grids[i][j];
            double T_new = (grids[i - 1][j] + grids[i + 1][j] +
                grids[i][j - 1] + grids[i][j + 1]) / 4.00;

            grids[i][j] = T_new;

            // 检查是否达到收敛条件
            if (fabs(T_old - T_new) > CONVERGENCE_CRITERION) {
                converged = false;
            }
        }
    }

    // 在每次迭代后打印温度网格
    cout << "第" << iter + 1 << "次的迭代结果为: " << endl;
    printTemperature(grids);
}
```

当所有节点的温度满足收敛的条件时，停止迭代，输出结果。

2 代码实现

2.1 常量定义

为了方便表示，定义几个常量

‘MAX_ITERATIONS’，表示最大的迭代次数，用于限制迭代过程，我们的程序中设置为 1000。实验中发现对于此题，最大的迭代次数为 55 次，此后无论如何修改精度，都会在 55 次迭代后得出准确的值。考虑到 double 型变量的精度，我们尝试了在 Python 中使用相同的逻辑进行迭代，发现次数也为 55 次，猜测该题经过 55 次迭代后可以获得精确值。

‘CONVERGENCE_CRITERION’，表示收敛标准，判断迭代是否结束。衡量$|T_{old} - T_{new}|$是否满足精度要求。实验中发现精度设置为10^{-14}左右经过 55 次迭代即可得出精确值，继续增加精度迭代次数不再增加。

2.2 函数解释

为了方便输出，在 `ostream & operator << (ostream& cout, double grids[5][5])` 中重载了“<<”运算符，打印最后的二维数组。

在 `int main()`实现了循环。

2.3 完整代码

```cpp
/*
 * File: transportprinciple_project.cpp
 * Author: 梁星宇
 * Student Number: 12021108
 * Email: cqqjlxy@gmail.com
 * Description: 使用高斯-赛德尔迭代法计算习题6-8
 * Created on: Dec 29, 2023
 * Last modified: Jan 1, 2024
 */

#include <cmath>
#include <iostream>
using namespace std;

// 常量：用来规定最大的迭代次数
const int MAX_ITERATIONS = 100;

// 精度要求 (°C)
const double CONVERGENCE_CRITERION = 1e-14;

// 重载一下<<，方便输出每个节点的温度迭代结果

ostream & operator << (ostream& cout, double grids[5][5]) {
    for (int i = 0; i < 5; ++i) {
        for (int j = 0; j < 5; ++j) {
            cout << grids[i][j] << "\t";
        }
        cout << endl;
    }
    return cout;
}

int main() {
    // 初始化网格，节点猜测最初的温度为0°C
    /*
    [150][180][200][180][150]
    [180][(1)][(2)][(3)][180]
    [200][(4)][(5)][(6)][200]
    [180][(7)][(8)][(9)][180]
    [150][180][200][180][150]
    */
```

```
    double grids[5][5] = {
            {150, 180, 200, 180, 150},
            {180,   0,   0,   0, 180},
            {200,   0,   0,   0, 200},
            {180,   0,   0,   0, 180},
            {150, 180, 200, 180, 150}
    };

    bool converged = false;

    for (int iter = 0; iter < MAX_ITERATIONS && !converged; ++iter) {
        converged = true;

        // 使用高斯-塞德尔迭代法更新内部节点温度
        for (int i = 1; i < 5 - 1; ++i) {
            for (int j = 1; j < 5 - 1; ++j) {
                double T_old = grids[i][j];
                double T_new = (grids[i - 1][j] + grids[i + 1][j] +
                    grids[i][j - 1] + grids[i][j + 1]) / 4.00;

                grids[i][j] = T_new;

                // 检查是否达到收敛条件
                if (fabs(T_old - T_new) > CONVERGENCE_CRITERION) {
                    converged = false;
                }
            }
        }

        // 在每次迭代后打印温度网格
        cout << "第" << iter + 1 << "次的迭代结果为: " << endl << grids << endl;
    }

    // 打印最终温度网格
    cout << "迭代后的结果为: " << endl << grids << endl;

    return 0;
}
```

3 结果讨论

程序编译运行后输出如下：

第 1 次的迭代结果为：
150 180 200 180 150
180 90 72.5 108.125 180
200 72.5 36.25 86.0938 200
180 108.125 86.0938 133.047 180
150 180 200 180 150
第 2 次的迭代结果为：
150 180 200 180 150
180 126.25 117.656 140.938 180
200 117.656 101.875 143.965 200
180 140.938 143.965 161.982 180
150 180 200 180 150
第 3 次的迭代结果为：
150 180 200 180 150
180 148.828 147.91 162.969 180
200 147.91 145.938 167.722 200
180 162.969 167.722 173.861 180
150 180 200 180 150
第 4 次的迭代结果为：
150 180 200 180 150
180 163.955 168.215 173.984 180
200 168.215 167.969 178.954 200
180 173.984 178.954 179.477 180
150 180 200 180 150
第 5 次的迭代结果为：
150 180 200 180 150
180 174.108 179.015 179.492 180
200 179.015 178.984 184.488 200
180 179.492 184.488 182.244 180
150 180 200 180 150
第 6 次的迭代结果为：
150 180 200 180 150
180 179.508 184.496 182.246 180
200 184.496 184.492 187.246 200
180 182.246 187.246 183.623 180
150 180 200 180 150
第 7 次的迭代结果为：
150 180 200 180 150
180 182.248 187.247 183.623 180
200 187.247 187.246 188.623 200
180 183.623 188.623 184.311 180
150 180 200 180 150
第 8 次的迭代结果为：
150 180 200 180 150
180 183.623 188.623 184.312 180
200 188.623 188.623 189.312 200
180 184.312 189.312 184.656 180
150 180 200 180 150
第 9 次的迭代结果为：
150 180 200 180 150
180 184.312 189.312 184.656 180
200 189.312 189.312 189.656 200
180 184.656 189.656 184.828 180
150 180 200 180 150
第 10 次的迭代结果为：
150 180 200 180 150
180 184.656 189.656 184.828 180
200 189.656 189.656 189.828 200
180 184.828 189.828 184.914 180
150 180 200 180 150
第 11 次的迭代结果为：
150 180 200 180 150
180 184.828 189.828 184.914 180
200 189.828 189.828 189.914 200
180 184.914 189.914 184.957 180
150 180 200 180 150
第 12 次的迭代结果为：
150 180 200 180 150
180 184.914 189.914 184.957 180
200 189.914 189.914 189.957 200
180 184.957 189.957 184.978 180
150 180 200 180 150
第 13 次的迭代结果为：
150 180 200 180 150
180 184.957 189.957 184.978 180
200 189.957 189.957 189.978 200
180 184.978 189.978 184.989 180
150 180 200 180 150

第 14 次的迭代结果为:
150 180 200 180 150
180 184.978 189.978 184.989 180
200 189.978 189.978 189.989 200
180 184.989 189.989 184.995 180
150 180 200 180 150
第 15 次的迭代结果为:
150 180 200 180 150
180 184.989 189.989 184.995 180
200 189.989 189.989 189.995 200
180 184.995 189.995 184.997 180
150 180 200 180 150
第 16 次的迭代结果为:
150 180 200 180 150
180 184.995 189.995 184.997 180
200 189.995 189.995 189.997 200
180 184.997 189.997 184.999 180
150 180 200 180 150
第 17 次的迭代结果为:
150 180 200 180 150
180 184.997 189.997 184.999 180
200 189.997 189.997 189.999 200
180 184.999 189.999 184.999 180
150 180 200 180 150
第 18 次的迭代结果为:
150 180 200 180 150
180 184.999 189.999 184.999 180
200 189.999 189.999 189.999 200
180 184.999 189.999 185 180
150 180 200 180 150
第 19 次的迭代结果为:
150 180 200 180 150
180 184.999 189.999 185 180
200 189.999 189.999 190 200
180 185 190 185 180
150 180 200 180 150
……
第 54 次的迭代结果为:
150 180 200 180 150
180 185 190 185 180
200 190 190 190 200
180 185 190 185 180
150 180 200 180 150
第 55 次的迭代结果为:
150 180 200 180 150
180 185 190 185 180
200 190 190 190 200
180 185 190 185 180
150 180 200 180 150
迭代后的结果为:
150 180 200 180 150
180 185 190 185 180
200 190 190 190 200
180 185 190 185 180
150 180 200 180 150

可以看出，经过 55 次迭代后温度等于直接计算的温度，此后再降低精度迭代次数不变。表明代码逻辑正确，可以解决该问题。