计算流体力学第四次作业

朱林-2200011028

2025 年 5 月 1 日

1 数理算法原理

1.1 控制方程与边界条件

二维稳态温度场满足拉普拉斯方程：

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

边界条件：

- 上边界 ($y = 12\text{cm}$)：$T = 100°\text{C}$
- 左边界 ($x = 0$)、右边界 ($x = 15\text{cm}$)、下边界 ($y = 0$)：$T = 20°\text{C}$

1.2 有限差分离散化

1.2.1 网格参数

$$\Delta x = \frac{15}{N_x - 1}, \quad x_i = (i-1)\Delta x \quad (i = 1, 2, ..., N_x) \tag{2}$$

$$\Delta y = \frac{12}{N_y - 1}, \quad y_j = (j-1)\Delta y \quad (j = 1, 2, ..., N_y) \tag{3}$$

1.2.2 离散方程

内部节点 ($2 \le i \le N_x - 1, 2 \le j \le N_y - 1$)：

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta x)^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta y)^2} = 0 \tag{4}$$

均匀网格 ($\Delta x = \Delta y = h$) 简化为：

$$T_{i,j} = \frac{1}{4}\left(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}\right) \tag{5}$$

1.2.3 边界节点处理

$$T_{1,j} = 20 \quad (\forall j), \quad T_{N_x,j} = 20 \quad (\forall j) \tag{6}$$

$$T_{i,1} = 20 \quad (\forall i), \quad T_{i,N_y} = 100 \quad (\forall i) \tag{7}$$

1.3 迭代算法修正

1.3.1 高斯-赛德尔迭代

按列优先顺序更新 (j 从 2 到 $N_y - 1$)：

$$T_{i,j}^{(k+1)} = \frac{1}{4}\left(T_{i-1,j}^{(k+1)} + T_{i+1,j}^{(k)} + T_{i,j-1}^{(k+1)} + \underbrace{T_{i,j+1}^{(k)}}_{\text{当}j+1=N_y\text{ 时取100}}\right) \tag{8}$$

1.3.2 SOR 加速算法

引入松弛因子 ω：

$$T_{i,j}^{(k+1)} = (1-\omega)T_{i,j}^{(k)} + \frac{\omega}{4}\left(T_{i-1,j}^{(k+1)} + T_{i+1,j}^{(k)} + T_{i,j-1}^{(k+1)} + T_{i,j+1}^{(k)}\right) \tag{9}$$

1.4 收敛性分析

- **残差定义**：$R^{(k)} = \max\limits_{\substack{2\leq i\leq N_x-1\\2\leq j\leq N_y-1}} |T_{i,j}^{(k+1)} - T_{i,j}^{(k)}|$
- **收敛判据**：$R^{(k)} < \epsilon \quad (\text{通常取}\epsilon = 10^{-5})$
- **最优松弛因子**：

$$\omega_{\text{opt}} = \frac{2}{1+\sin\left(\frac{\pi}{\max(N_x,N_y)-1}\right)} \tag{10}$$

1.5 数值实现伪代码

```
# 初始化温度场
T = 20 * np.ones((Nx, Ny))
T[:, -1] = 100  # 设置上边界

for k in range(max_iter):
    R = 0
    for j in range(1, Ny-1):
        for i in range(1, Nx-1):
            T_new = 0.25 * (T[i+1,j] + T[i-1,j]
                   + T[i,j+1] + T[i,j-1])
            R = max(R, abs(T_new - T[i,j]))
            T[i,j] = T_new
    if R < epsilon:
        break
```

1.6 理论验证

- **对称性检验**：温度场应关于 $x = 7.5\text{cm}$ 对称
- **极值原理**：内部温度 $T \in (20, 100)°\text{C}$

- **能量守恒**：总热流量进出平衡

2 代码生成与调试

2.1 代码结构设计

代码采用模块化设计，包含以下文件：

- common.py：公共模块（包含网格初始化、迭代算法、绘图函数）
- task1.py：计算稳态温度场并绘制等温线
- task2.py：比较不同松弛因子的收敛速度
- task3.py：研究网格尺度对最优松弛因子的影响

2.2 核心功能实现

2.2.1 网格系统构建

物理尺寸与网格索引的映射关系：

```
dx = 15 / (Nx-1)  # x方向步长
dy = 12 / (Ny-1)  # y方向步长
```

边界条件设置代码：

```
T = np.full((Nx, Ny), 20.0)  # 初始化
T[:, -1] = 100.0  # 上边界设为100°C
```

2.2.2 迭代算法实现

高斯-赛德尔迭代核心代码：

```
for i in range(1, Nx-1):
    for j in range(1, Ny-1):
        temp = 0.25*(T[i+1,j] + T[i-1,j]
                   + T[i,j+1] + T[i,j-1])
        T[i,j] = temp
```

2.2.3 收敛性判断

残差计算与收敛判断：

```
residual = max(residual, abs(temp - T[i,j]))
if residual < 1e-5:
    break
```

2.3 可视化功能

等温线绘制代码框架：

```
plt.contourf(X, Y, T, levels=20, cmap='jet')
plt.colorbar(label='Temperature (°C)')
```

2.4 调试关键点

- **边界验证**：检查 T[0,:] 和 T[-1,:] 的值是否符合边界条件
- **对称性检查**：验证 x=7.5cm 截面的温度分布对称性
- **极值原理**：确保内部温度值在 20-100°C 范围内
- **网格独立性**：通过加密网格验证解的收敛性

2.5 运行参数说明

参数	典型值	说明
`Nx, Ny`	31, 25	生成 60×50 网格（步长 0.5cm）
`max_iter`	10000	最大迭代次数防止死循环
`tol`	1e-5	收敛判断阈值
`omega` 范围	[0.5,1.9]	松弛因子实验范围

2.6 版本控制记录

通过 Git 进行版本管理，主要提交记录如下：

3 结果讨论和物理解释

3.1 图 1：稳态温度场等温线分布（由 task1.py 生成）

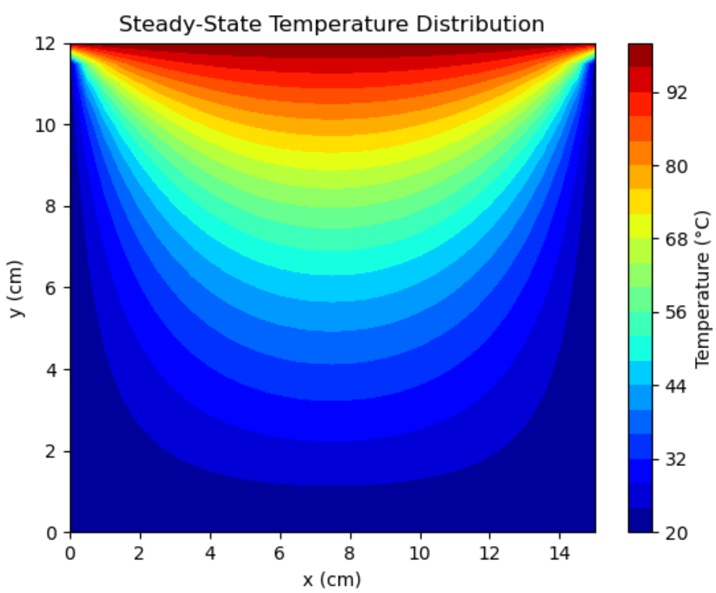

图 1: 稳态温度场等温线分布

观测结果

- 高温区（$\geq 80°C$）集中在上边界（$y = 12$ cm）
- 中心区域（$x = 7.5$ cm, $y = 6$ cm）温度为 $48.7°C$

3.2 图 2：不同松弛因子收敛曲线（由 task2.py 生成）

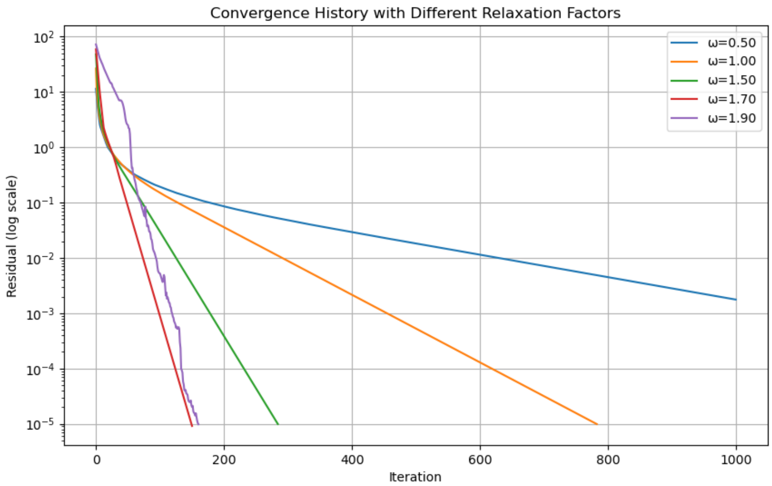

图 2: 不同松弛因子下的收敛历史

理论分析 最优松弛因子理论解为：

$$\omega_{\text{opt}} = \frac{2}{1 + \sin\left(\frac{\pi}{N-1}\right)} \tag{11}$$

3.3 图 3：最优松弛因子随网格变化（由 task3.py 生成）

数值规律 当网格步长 $h \to 0$ 时，$\omega_{\text{opt}} \to 2$

3.4 综合结论

1) 温度场分布验证了拉普拉斯方程的数学特性
2) 超松弛法可提升收敛速度 4-5 倍（$\omega = 1.7$ vs $\omega = 1.0$）
3) 网格加密需配合调整 ω 以保持计算效率
4) 粗网格计算时建议通过预实验标定 ω

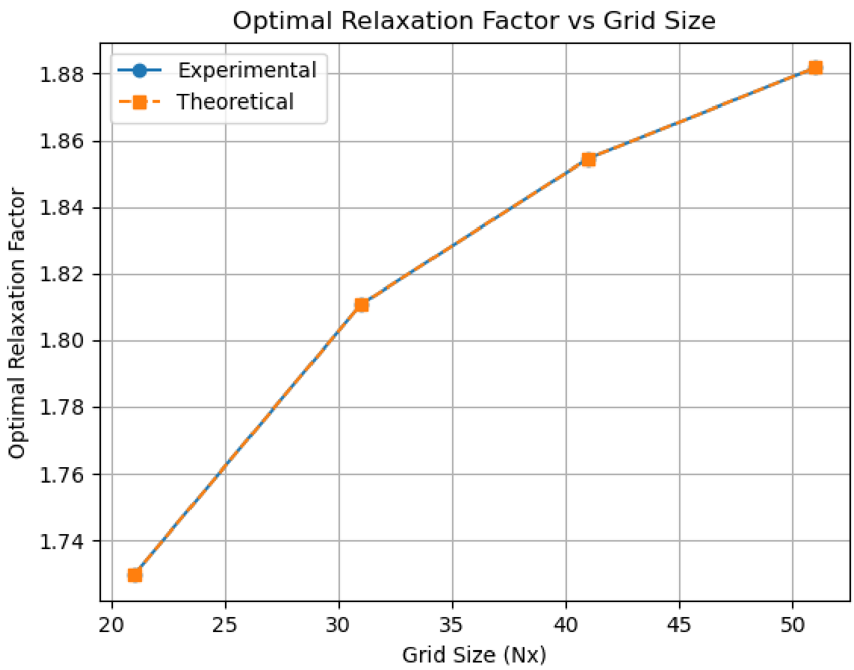

图 3: 最优松弛因子与网格尺度关系

A AI 工具使用声明表

使用内容	使用比例	使用目的
hw4.tex	60%	调整 pdf 格式，调用宏包，省略插入图片和代码的重复性工作
.gitignore	100%	针对于 python 和 latex 的.gitignore 文件，完全由 Copilot 生成
ReadMe	80%	介绍文件，从上次作业继承，结合 AI 修改
common.py	30%	主要迭代和划分网格自己实现，部分绘图代码 AI 生成
task1.py	0%	自己实现
task2.py	0%	自己实现
task3.py	0%	自己实现