基于欧拉方法的一维热传导数值求解问题

任立德
南方科技大学

摘要

本项目完成了对一维热传导方程的数值求解。在时间上采用了显式欧拉方法和隐式欧拉方法，在空间上采用了有限差分方法。通过调用 PETSC 完成对向量和矩阵对运算，通过调用 MPI 实现并行计算，通过 HDF5 文件保存数据。通过调整网格点大小和时间步大小对两种方法对稳定性和误差进行了分析。通过使用“-r”参数控制程序是否重启。通过使用“-np”调整 CPU 核数对代码并行效率进行分析。

1. 理论推导

已知一维热传导方程如下所示。

$$\rho c\frac{\partial u}{\partial t}-\kappa\frac{\partial^2 u}{\partial x^2}=f \tag{1}$$

根据有限差分法和显式欧拉方法可以得到如下公式。

$$\rho c\frac{u_j^{n+1}-u_j^n}{\Delta t}-\kappa\frac{u_{j-1}^n-2u_j^n+u_{j+1}^n}{\Delta x^2}=f \tag{2}$$

经过推导得到第 n+1 个时间步，第 j 个网格点的 u_j^{n+1} 计算公式。

$$u_i^{n+1}=\frac{\kappa\Delta t}{\rho c\Delta x^2}u_{j-1}^n+\left(1-\frac{2\kappa\Delta t}{\rho c\Delta x^2}\right)u_j^n+\frac{\kappa\Delta t}{\rho c\Delta x^2}u_{j+1}^n+\frac{f\Delta t}{\rho c} \tag{3}$$

定义 $\alpha=\frac{\kappa\Delta t}{\rho c\Delta x^2}$，则问题可以转换成三对角矩阵的求解。

$$\begin{bmatrix}1-2\alpha & \alpha & & \\ \alpha & 1-2\alpha & \alpha & \\ & \ddots & \ddots & \\ & & \alpha & 1-2\alpha\end{bmatrix}\times\begin{bmatrix}u_0^n\\u_1^n\\\vdots\\u_j^n\end{bmatrix}+\begin{bmatrix}f_0\\f_1\\\vdots\\f_j\end{bmatrix}=\begin{bmatrix}u_0^{n+1}\\u_1^{n+1}\\\vdots\\u_j^{n+1}\end{bmatrix} \tag{4}$$

同理根据有限差分法和隐式欧拉方法可以得到推导公式。

$$-\frac{\kappa\Delta t}{\rho c\Delta x^2}u_{j-1}^{n+1}+\left(1+\frac{2\kappa\Delta t}{\rho c\Delta x^2}\right)u_j^{n+1}-\frac{\kappa\Delta t}{\rho c\Delta x^2}u_{j+1}^{n+1}=u_j^n+\frac{f\Delta t}{\rho c} \tag{5}$$

隐式欧拉方法同样可以转换成三对角矩阵进行求解。

$$\begin{bmatrix}1+2\alpha & -\alpha & & \\ -\alpha & 1+2\alpha & -\alpha & \\ & \ddots & \ddots & \\ & & -\alpha & 1+2\alpha\end{bmatrix}\times\begin{bmatrix}u_0^{n+1}\\u_1^{n+1}\\\vdots\\u_j^{n+1}\end{bmatrix}=\begin{bmatrix}u_0^n\\u_1^n\\\vdots\\u_j^n\end{bmatrix}+\begin{bmatrix}f_0\\f_1\\\vdots\\f_j\end{bmatrix} \tag{6}$$

为了验证计算过程是否正确，还需要精确解作为参考。当 $\frac{\partial u}{\partial t}=0$ 时，系统温度不随时间变化，可以看作稳定状态。即有如下公式。

$$-\kappa\frac{\partial^2 u}{\partial x^2}=\sin(l\pi x) \tag{7}$$

根据边界条件求得微分方程的解析解为如下公式。

$$u(x)=\frac{1}{\kappa l^2\pi^2}\sin(l\pi x) \tag{8}$$

2. 代码实现

项目将代码分成多个模块，每个模块解决特定问题。显式方法和隐式方法除了矩阵运算模块内部实现有所差别，其他模块基本相同。

在 explicit.c 中，模块一定义了与模型有关的多个常量参数，包括方程中的 Rho、C、K 等物理量以及 GRID、TSTEP 等控制网格点和迭代步长的参数。模块二定义了所需变量，包括计算时所需的向量和矩阵以及并行所需的 rank 变量和文件变量等。模块三对参数进行初始化，包括初始化 petsc 和 mpi，以及读取“-r”参数。模块四计算并打印模型运算的重

要信息，包括 dx、dt 等。模块五用于创建计算所需对向量和矩阵。模块六用于初始化三对角矩阵。模块七根据“-r”参数对初始条件进行初始化操作。当“-r”不存在时，表示程序无需重启，计算时直接从 HDF5 文件中获取数据。当“-r”不存在时，表示程序需要重启，默认时间 t=0，重新计算各项数据。模块八用于迭代计算 uj 的值，从当前时间计算到初始设定的时间上限 TEND，且迭代次数为 10 的整数倍时，将数据写入 HDF5 文件中，对数据进行更新。模块九用于关闭向量和矩阵，防止内存泄漏。

通过 Makefile 对代码进行编译，使用 make explicit 或 make implicit 即可生成相应对可执行文件，使用 make clean 可以清除所有生成的中间文件和可执行文件。使用 explicit.LSF 提交 TaiYi 脚本，通过“-r”和“-np”等参数对程序进行相应控制。程序执行完成或者强制中断后，均可生成对应对 HDF5 文件，可用于分析计算结果和代码性能以及用于继续计算。

完整代码及使用说明文档的 github 仓库链接为 https://github.com/RenLide/HPC-project.git，TaiYi 服务器中项目代码文件夹位置为/work/maerenld/HPC/project。

3. 计算结果分析

3.1. 精度分析

选择精确解作为结果误差分析的基准。选择 GRID=100，TSTEP=100000 进行计算，此时 CFL=0.2。计算结果如图（1）所示。可以看出显式欧拉方法的

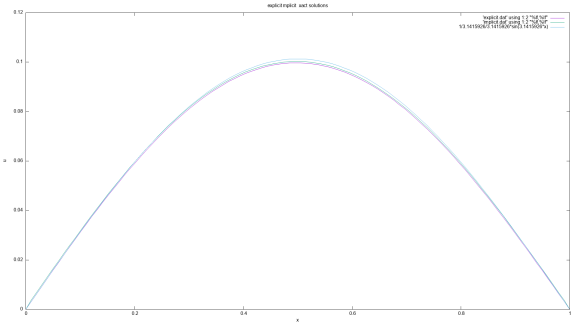

图 1. solution CFL=0.2

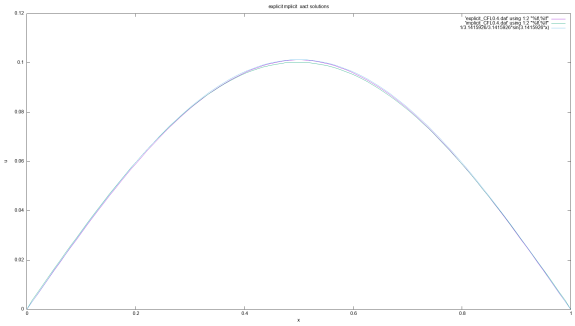

图 2. solution CFL=0.4

3.2. 稳定性分析

对于显式欧拉方法，根据公式 (3) 进行冯诺依曼稳定性分析，可以得到如下公式。

$$\delta u_j^{n+1} = \alpha * \delta u_{j-1}^n + (1-2\alpha) * \delta u_j^n + \alpha * \delta u_{j+1}^n \tag{9}$$

即有如下近似公式。

$$\delta u_j^{n+1} \sim e^{\delta n \Delta t} * e^{i(k*j\Delta x)} \tag{10}$$

$$\begin{aligned} e^{\delta \Delta t} &= \alpha * e^{ik\Delta x} + (1-2*\alpha) + \alpha * e^{-ik\Delta x} \\ &= \alpha * \left(e^{ik\Delta x} + e^{-ik\Delta x}\right) + (1-2*\alpha) \\ &= 1 - 2\alpha + 2\alpha\cos(k\Delta x) \end{aligned} \tag{11}$$

根据稳定条件可以计算出 α 的取值范围，在代码中用 CFL 表示 α。

$$\begin{aligned} |1-2\alpha+2\alpha\cos(k\Delta x)| \le 1 &\Rightarrow -1 \le 1-4\alpha \le 1 \\ &\Rightarrow 0 \le \alpha \le \frac{1}{2} \end{aligned} \tag{12}$$

在 3.1 精度分析中，取 $\alpha = 0.2$，显式与隐式欧拉方法均稳定。

3.3. 重启分析

3.4. 并行分析

3.5. 内存分析

4. 总结与展望

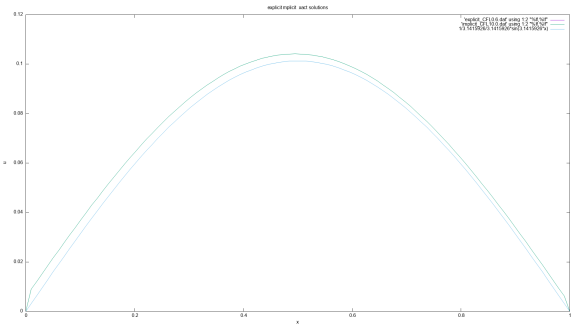

图 3. solution CFL over 0.6