利用 HYPRE 并行求解三维热方程
并行计算第四次上机作业

郑灵超

2017 年 1 月 1 日

目录

1 问题介绍 **2**

2 算法介绍 **2**

3 程序分析 **2**

3.1 程序说明 2

3.2 程序评价 3

4 数值结果 **4**

4.1 误差分析 4

5 上机报告总结 **4**

1 问题介绍

本次要求解的方程为三维的热方程问题，其定义域为八面体

$$|x| + |y| + |z| \leq 1.$$

方程形式和边界条件分别为

$$\begin{cases} u_t - \Delta u = f, \\ u|_{t=0} = u_0, \\ u|_{|x|+|y|+z=1} = g_{up}, \\ \dfrac{\partial u}{\partial \boldsymbol{n}}|_{|x|+|y|-z=1} = g_{down}. \end{cases} \tag{1}$$

2 算法介绍

此次我们采用的算法是隐式差分格式，将求解区域按正方形网格剖分，即网格尺度 $h = \frac{1}{N}$, 则网格节点为 $x = ih, y = jh, z = kh$ 其迭代格式为

$$\begin{aligned} &\frac{u^{(t+\Delta t)}(ih, jh, kh) - u^{(t)}(ih, jh, kh)}{\Delta t} + \frac{1}{h^2}[u^{(t+\Delta t)}(ih + h, jh, kh) + u^{(t+\Delta t)}(ih - h, jh, kh) \\ &+ u^{(t+\Delta t)}(ih, jh + h, kh) + u^{(t+\Delta t)}(ih, jh - h, kh) + u^{(t+\Delta t)}(ih, jh, kh + h) \\ &+ u^{(t+\Delta t)}(ih, jh, kh - h) - 6u^{(t+\Delta t)}(ih, jh, kh)] = f(ih, jh, kh, t + \Delta t) \end{aligned} \tag{2}$$

将其写成方程组的形式：

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3}$$

其中矩阵 $\mathbf{A}$ 是一个主对角线为 $1 + 6\frac{\Delta t}{h^2}$，同一行还有 6 个 $-\frac{\Delta t}{h^2}$ 的七对角矩阵，向量 $\boldsymbol{b}$ 为 $f\Delta t + u^{(t)}$。

这部分数值格式的精度为 $O(\Delta t + h^2)$，隐式格式对 CFL 条件数 $\frac{\Delta t}{h^2}$ 没有要求，因此这部分数值格式的精度为 $O(\Delta t + h^2)$。

在上边界，即狄利克雷边界上，我们直接进行赋值；在下边界，即诺依曼边界上，我们根据 $t + \Delta t$ 时刻该点的法向导数，采用内部点和当前点的差来逼近这个法向导数，得到一个关于网格密度 h 为一阶精度的数值格式，因此我么最终采用的数值格式的精度为 $O(\Delta t + h)$。

3 程序分析

3.1 程序说明

我们定义了一个类 Heat, 用于求解这一类型的热方程，用户可以通过在 main.cpp 文件中修改方程的初边值条件。网格密度 N 和计算终止时间 t_{end} 通过命令行参数读入。具

体程序的结构和声明可以参见 doc 目录下的 refman.pdf 和 html 目录下的 index.html 网页。以下我们简要说明一下并行实现的算法流程:

1. 读入信息，如网格密度，计算终止时间，所用进程数目，CFL 条件数，和设置的初边值条件。

2. 由 0 号进程进行一些预处理工作：将三维的点用一个长度为 M 的 std::vector 表示，并给出将点转化为 vector 中下标的函数。并利用这个函数计算出每个编号对应的点的坐标和邻居的编号，并将这些信息发送给所需要的进程。

3. 每个进程各自的初始化工作，包括计算每个进程所包含的点的起始编号和终止编号。我们这里将所有网格均等分给每个进程，第 i 号进程的需要处理的网格编号为 $\frac{iM}{size}$ 到 $\frac{(i+1)M}{size}$。此外，每个进程需要计算自己进行迭代计算时需要相邻进程提供的数据。

4. 每个进程计算各自的矩阵 $\mathbf{A}$，并保存为 `HYPRE_IJMatrix` 格式。此外，我们可以在此时初始化求解器，从而节省整体运行时间。

5. 每个进程分别计算自己所管辖区域的 $t = 0$ 的初值情况。

6. 进行一步迭代计算，包括从前一步的计算结果中获取向量 $\boldsymbol{u}^{(t)}$ ，计算得到向量 $\boldsymbol{b}$，并利用 HYPRE 的求解方法来求解向量 $\boldsymbol{x}$ 。

7. 迭代时间达到终止时间，将解返回给用户，并进行释放内存，清除数据等操作。

3.2 程序评价

这次我们采用的程序的优点有:

- 将整个数据拉成一个一维连续的 vector，并等分给各个进程，负载较为均衡。
- 用类进行封装，用户只需要通过主函数进行修改。
- 由于每次迭代的矩阵 $\mathbf{A}$ 并没有变化，我们只对它进行了一次计算，减少了不少计算量。同时，在迭代开始之前就对求解器进行了定义，也可以节省运行时间。

此外，我认为此次写的程序还有如下不足:

- 一些基本信息的初始化工作仍然由 0 号进程完成，这儿我认为还有改进空间。
- 对于边界，我们尚未给出一个精细的处理方法。目前求解方法的精度为 $O(\Delta t + h)$，并不能令人满意。

4 数值结果

4.1 误差分析

我们选取了一个有精确解的方程进行计算，其精确解为

$$u(x,y,z,t)=\sin((x^2+y^2+z^2)t) \tag{4}$$

采取之前所介绍的方法进行计算，并将解与真实解做了误差比较，我们得到如下的结果:

网格密度 N	L2 误差	阶数	CPU 时间 (s)	CPU 时间的阶数
10	3.25e-3	/	0.03	/
20	1.41e-3	1.2	0.23	2.94
40	6.57e-4	1.1	5.44	4.56
80	4.00e-4	0.7	181.64	5.06

此处我们采用的 $\Delta t=2h^2$, 计算终止时间为 0.1，进程数为 4.

受到诺依曼边界条件的影响，我们的数值精度只有 1 阶。

5 上机报告总结

此次上机作业我们学习了 HYPRE 软件，并利用它重新求解了第二次作业中的热方程。

此外，我们熟悉 HYPRE 中行压缩的稀疏矩阵的存储和赋值方式，了解了一些常用的解线性方程组的求解器，对 HYPRE 这类数学软件的安装和使用有了一定认识。