MAE5032 · High Performace Computing

Final-Project

Fudong Deng / 12132389

June 9 2022

1 公式推导以及稳定性分析

一维传热方程如下:

$$\rho c\frac{\partial u}{\partial t}-\kappa\frac{\partial^2 u}{\partial x^2}=\sin(l\pi x)$$

边界条件为:

$$u(0,t)=u(1,t)=0$$
$$u|_{t=0}=e^x$$

则可将传热方程改写为: $\frac{\partial u}{\partial t}-\frac{\kappa}{\rho c}\frac{\partial^2 u}{\partial x^2}=\frac{1}{\rho c}\sin(l\pi x)$

1.1 显示格式 (Adams-Bashforth)

差分格式为:

$$\begin{cases}\dfrac{\partial u}{\partial t}=\dfrac{u_j^{n+1}-u_j^n}{\Delta t}\\ \dfrac{\partial^2 u}{\partial x^2}=\dfrac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta x^2}\end{cases}$$

$$\begin{aligned}
\therefore\ \frac{u_j^{n+1}-u_j^n}{\Delta t}&=\frac{\kappa}{\rho c}\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta x^2}+\frac{1}{\rho c}\sin(l\pi j\Delta x)\\
\Longrightarrow\ u_j^{n+1}&=u_j^n+\frac{\kappa\Delta t}{\rho c\Delta x^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)+\frac{\Delta t}{\rho c}\sin(l\pi j\Delta x)\\
\Longrightarrow\ u_j^{n+1}&=\frac{\kappa\Delta t}{\rho c\Delta x^2}u_{j+1}^n+(1-\frac{2\kappa\Delta t}{\rho c\Delta x^2})u_j^n+\frac{\kappa\Delta t}{\rho c\Delta x^2}u_{j-1}^n+\frac{\Delta t}{\rho c}\sin(l\pi j\Delta x)\\
make\frac{\kappa}{\rho c}&=\alpha,\frac{\alpha\Delta t}{\Delta x^2}=\beta=\frac{\kappa\Delta t}{\rho c\Delta x^2}\\
\Longrightarrow\ u_j^{n+1}&=\beta u_{j+1}^n+(1-2\beta)u_j^n+\beta u_{j-1}^n+\frac{\Delta t}{\rho c}\sin(l\pi j\Delta x)
\end{aligned}$$

$$\begin{pmatrix}u_1^{n+1}-\alpha\Delta t\sin(l\pi\Delta x)\\ u_2^{n+1}-\alpha\Delta t\sin(l\pi 2\Delta x)\\ \vdots\\ u_{n-2}^{n+1}-\alpha\Delta t\sin(l\pi(n-2)\Delta x)\\ u_{n-1}^{n+1}-\alpha\Delta t\sin(l\pi(n-1)\Delta x)\end{pmatrix}=\begin{pmatrix}1-2\beta&\beta&0&0&\cdots&0&0&0\\ \beta&1-2\beta&\beta&0&\cdots&0&0&0\\ 0&\beta&1-2\beta&\beta&\cdots&0&0&0\\ \vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\ 0&0&0&0&\cdots&0&\beta&1-2\beta\end{pmatrix}*\begin{pmatrix}u_1^n\\ u_2^n\\ \vdots\\ u_{n-2}^n\\ u_{n-1}^n\end{pmatrix}$$

1.1.1 稳定性分析

由 von-Neumann 稳定性分析可得:

$$
\begin{aligned}
&\delta u_j^{n+1} = \beta \delta u_{j-1}^n + (1-2\beta)\delta u_j^n + \beta \delta u_{j+1}^n \\
&\because \delta u_j^n \backsim e^{\sigma n \Delta t} \cdot e^{i(k \cdot j \Delta x)} \\
&\therefore e^{\sigma \Delta t} = \beta e^{ik\Delta x} + (1-2\beta) + \beta e^{-ik\Delta x} \\
&\qquad\;\; = (1-2\beta) + \beta (e^{ik\Delta x} + e^{-ik\Delta x})
\end{aligned}
$$

由 $|e^{\sigma \Delta t}| < 1 \implies |(1-2\beta)+\beta\cos(k\Delta x))| < 1 \implies |1-4\beta| < 1 \implies 0 \leq \beta \leq 0.5$

1.2 **隐式格式** (Crank-Nicolson)

差分格式为:

$$
\begin{cases}
\dfrac{\partial u}{\partial t} = \dfrac{u_j^{n+1} - u_j^n}{\Delta t} \\
\dfrac{\partial^2 u}{\partial x^2} = \dfrac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2}
\end{cases}
$$

$$
\begin{aligned}
\therefore \frac{u_j^{n+1} - u_j^n}{\Delta t} &= \frac{\kappa}{\rho c} \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} + \frac{1}{\rho c}\sin(l\pi j \Delta x) \\
make \frac{\kappa}{\rho c} &= \alpha, \frac{\alpha \Delta t}{\Delta x^2} = \beta = \frac{\kappa \Delta t}{\rho c \Delta x^2}
\end{aligned}
$$

$$
\implies -\beta u_{j-1}^{n+1} + (1+2\beta) u_j^{n+1} - \beta u_{j+1}^{n+1} = u_j^n + \frac{\Delta t}{\rho c}\sin(l\pi j \Delta x)
$$

解的格式为:

$$
\begin{pmatrix}
1+2\beta & -\beta & 0 & \cdots & 0 & 0 & 0 \\
-\beta & 1+2\beta & -\beta & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & -\beta & 1+2\beta & -\beta \\
0 & 0 & 0 & \cdots & 0 & -\beta & 1+2\beta
\end{pmatrix}
\begin{pmatrix}
u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{n-2}^{n+1} \\ u_{n-1}^{n+1}
\end{pmatrix}
=
\begin{pmatrix}
u_1^n + \frac{\Delta t}{\rho c}\sin(l\pi \Delta x) \\
u_2^n + \frac{\Delta t}{\rho c}\sin(l\pi 2 \Delta x) \\
\vdots \\
u_{n-2}^n + \frac{\Delta t}{\rho c}\sin(l\pi (n-2) \Delta x) \\
u_{n-1}^n + \frac{\Delta t}{\rho c}\sin(l\pi (n-1) \Delta x)
\end{pmatrix}
$$

1.2.1 **稳定性分析**

由 von-Neumann 稳定性分析可得:

$$
\begin{aligned}
-\beta \delta u_{j-1}^{n+1} + (1+2\beta)\delta u_j^{n+1} - \beta \delta u_{j+1}^{n+1} &= \delta u_j^n \\
\because \delta u_j^n &\backsim e^{\sigma n \Delta t} \cdot e^{i(k \cdot j \Delta x)} \\
\therefore -\beta e^{\sigma \Delta t} \cdot e^{-ik\Delta x} + (1+2\beta) e^{\sigma \Delta t} - \beta e^{\sigma \Delta t} \cdot e^{ik\Delta x} &= 1 \\
e^{\sigma n \Delta t}(\beta e^{ik\Delta t} + \beta e^{-ik\Delta t} - (1+2\beta)) &= -1 \\
\implies e^{\sigma n \Delta t} &= \frac{-1}{\beta e^{ik\Delta t} + \beta e^{-ik\Delta t} - (1+2\beta)}
\end{aligned}
$$

由 $|e^{\sigma \Delta t}| < 1 \implies |\frac{1}{1+2\beta(1-\cos(k\Delta x))}| < 1 \implies \beta$ 可以为任何值

2 结果显示

2.1 结果验证

一维传热方程解析解为：$\frac{\sin(l\pi x)}{\pi^2} - \frac{\sin(l\pi)}{\pi^2}$, 在 $l = 1$ 的情况下，解析解为：$\frac{\sin(\pi x)}{\pi^2}$。

显式格式情况下，设置网格数量 $n = 100, CFL = 0.2$，隐式格式为设置网格数 $n = 100, CFL = 1$ 的情况下，解析解以及显式格式与隐式格式的解如下图所示：

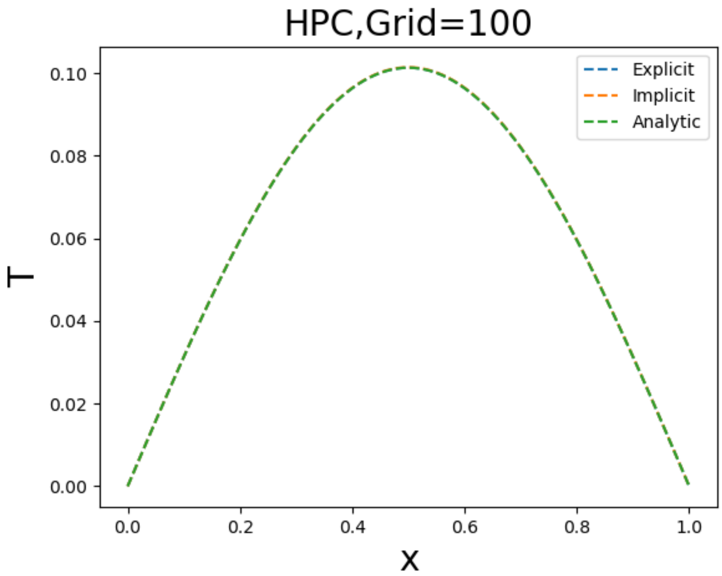


Figure 1: 结果验证

可以得到，显式解与隐式解与解析解基本重合，验证了显式格式以及隐式格式的准确性。但是由于显式格式稳定性条件的影响，当显式格式网格数增大时，时间步长需要随之减小从而使得计算量大大增加。而隐式格式的稳定性比较好，所以增加网格数量时，时间不长不用减少而计算量比显式要小，所用时间也比显式格式的要少。

2.2 并行性展示

显式格式下，展示程序的并行强可扩展性，在网格数较少的时候，CPU 核数增加，导致程序运行时间也增加，预估为网格数较少，各个 CPU 之间通信时间为主要部分，网格数 $n = 5000, CFL = 0.5$，得到在不同核数量下运行时间如图所示：

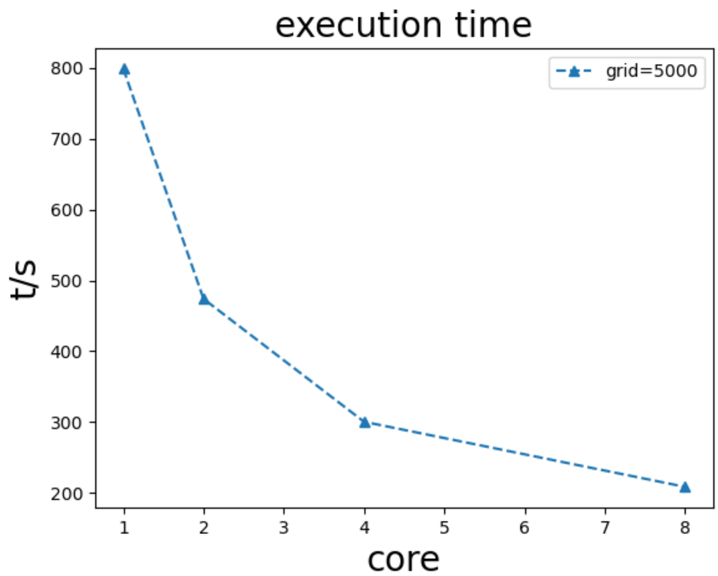


Figure 2: 不同核数相同网格的运行时间

显式格式下，展示程序的并行弱可扩展性，不同核数但是 $grid/core = 1000$，以及 $dt = 1e-8$, 得到在不同核数量下运行时间如图所示:

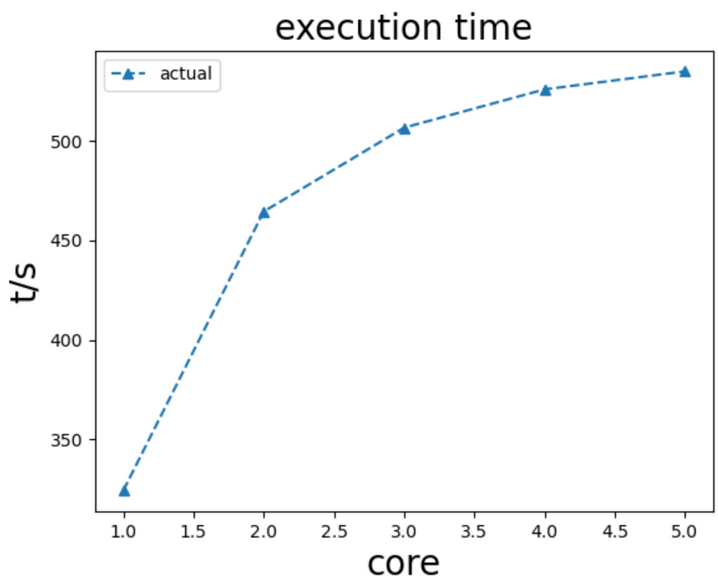


Figure 3: 不同核数不同网格的运行时间

隐式格式下，展示程序的并行强可扩展性，在网格数较少的时候，CPU 核数增加，导致程序运行时间也增加，预估为网格数较少，各个 CPU 之间通信时间为主要部分，网格数 $n = 5000, dt = 0.0001$，得到在不同核数量下运行时间如图所示：

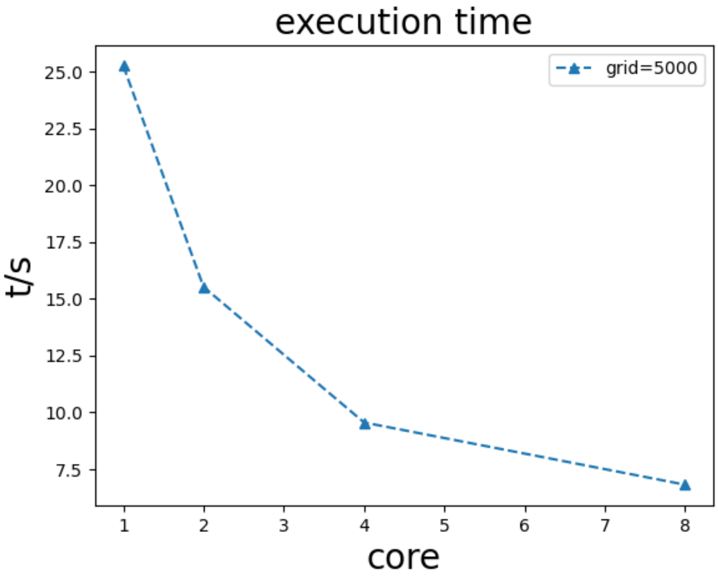


Figure 4: 不同核数相同网格的运行时间

隐式格式下，展示程序的并行弱可扩展性，不同核数但是 $grid/core = 1000$，以及 $dt = 1e-8$, 得到在不同核数量下运行时间如图所示:

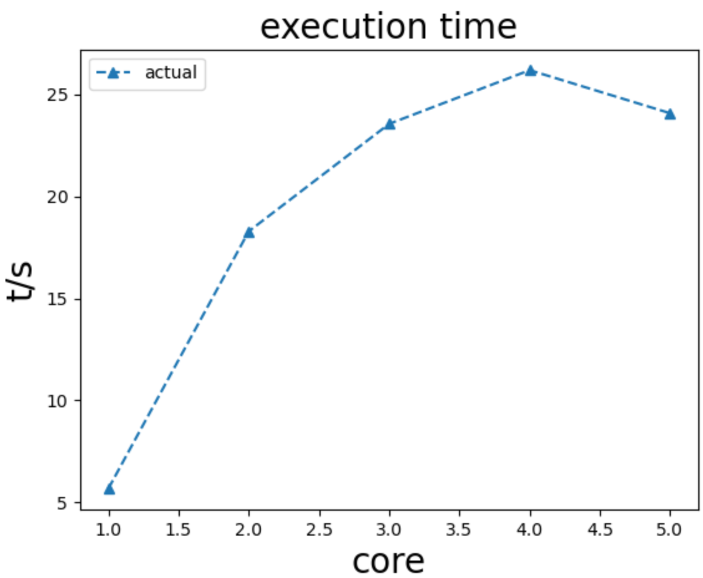
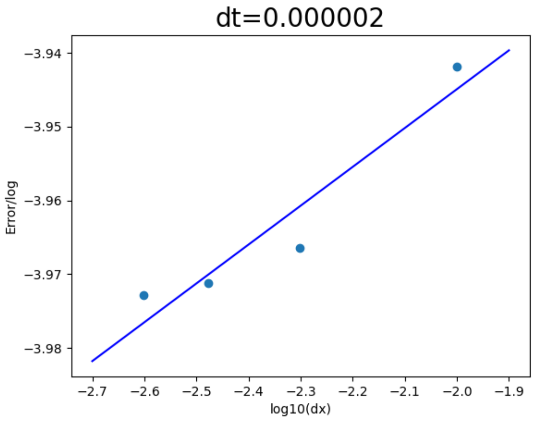


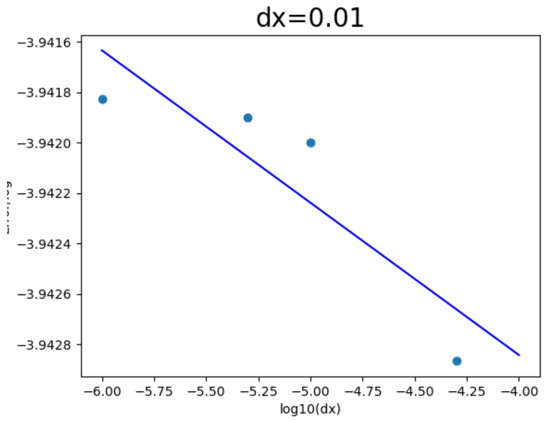
Figure 5: 不同核数不同网格的运行时间

2.3 误差分析

显式格式情况下，固定 $dt = 0.000002$, 改变网格数量 n 分别取 $100, 200, 300, 400$ 得到显式格式的解，与解析解比较最大误差值 $e := max_{1\le i\le n} \mid u_{exact,i} - u_{num,i} \mid$，对于不同的 Δx 得到的不同的误差，得到关系式中 $e \approx C_1(\Delta x)^{\alpha} + C_2(\Delta t)^{\beta}$ 中的 α，同理，固定 $dx = 0.01$，改变时间步长 dt 分别取 $0.00005, 0.00001, 0.000005, 0.000001$ 得到不同的解，拟合直线得到关系式中的 β。如下图所示：



(a) alpha

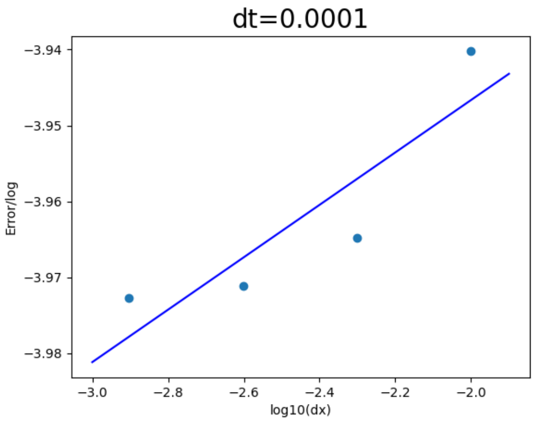


(b) beta

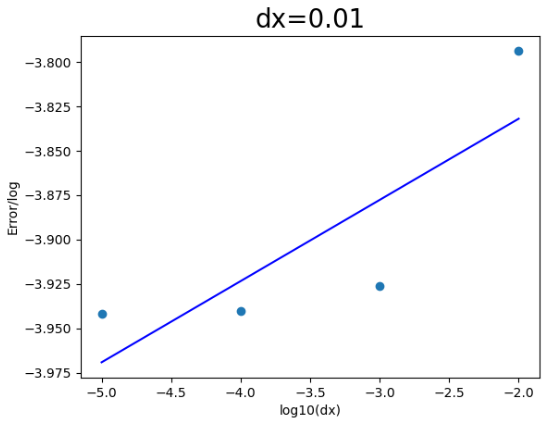
Figure 6: Manufactured solution method For Explicit

得到的 $\alpha \approx 0.0526$, $\beta \approx -0.000604$, 所以显式格式的误差与网格大小与时间步长关系为：中 $e \approx C_1(\Delta x)^0.0526 + C_2(\Delta t)^{-0.000604}$。

隐式格式情况下，固定 $dt = 0.0001$，改变网格数量 n 分别取 $100, 200, 300, 400$ 得到隐式格式的解，与解析解比较最大误差值 $e := max_{1\le i\le n} \mid u_{exact,i} - u_{num,i} \mid$，对于不同的 Δx 得到的不同的误差，得到 α，同理，固定 $dx = 0.01$，改变时间步长 dt 分别取 $0.01, 0.001, 0.0001, 0.00001$ 得到不同的解，拟合直线得到关系式中的 β。如下图所示：



(a) alpha



(b) beta

Figure 7: Manufactured solution method For Implicit

得到的 $\alpha \approx 0.03452$，$\beta \approx 0.04571$，所以隐式格式关系为：中 $e \approx C_1(\Delta x)^0.03452 + C_2(\Delta t)^0.04571$。

2.4 内存检查 (Valgrind)

使用 valgrind 检查内存泄漏情况，结果如图：

```
==125840== For counts of detected and suppressed errors, rerun with: -v
==125840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==125968==
==125968== HEAP SUMMARY:
==125968==     in use at exit: 60,881 bytes in 1,308 blocks
==125968==   total heap usage: 3,559 allocs, 2,251 frees, 150,862 bytes allocated
==125968==
==125968== LEAK SUMMARY:
==125968==    definitely lost: 10 bytes in 1 blocks
==125968==    indirectly lost: 0 bytes in 0 blocks
==125968==      possibly lost: 0 bytes in 0 blocks
==125968==    still reachable: 60,871 bytes in 1,307 blocks
==125968==         suppressed: 0 bytes in 0 blocks
==125968== Rerun with --leak-check=full to see details of leaked memory
==125968==
==125968== For counts of detected and suppressed errors, rerun with: -v
==125968== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Figure 8: valgrind 结果

结果中的 definitely lost 是由 Petsc 库中的函数导致的，总体程序并没有错误。

2.5 隐式格式中不同 pc_type 影响

对于 $dx = 0.01$, $dt = 0.00001$ 的相同条件下，分别使用三种不同的 pc_type，pc_type 为 Jacobi 时的性能报告如下：

```
------------------------------------------ PETSc Performance Summary: ------------------------------------------

./main.out on a  named r01n30 with 1 processor, by mae-dengfd Thu Jun  9 17:23:36 2022
Using Petsc Release Version 3.16.6, Mar 30, 2022

                         Max       Max/Min     Avg       Total
Time (sec):           4.117e+01     1.000   4.117e+01
Objects:              3.000e+01     1.000   3.000e+01
Flop:                 2.749e+08     1.000   2.749e+08  2.749e+08
Flop/sec:             6.677e+06     1.000   6.677e+06  6.677e+06
MPI Messages:         0.000e+00     0.000   0.000e+00  0.000e+00
MPI Message Lengths:  0.000e+00     0.000   0.000e+00  0.000e+00
MPI Reductions:       0.000e+00     0.000

Flop counting convention: 1 flop = 1 real number operation of type (multiply/divide/add/subtract)
                            e.g., VecAXPY() for real vectors of length N --> 2N flop
                            and VecAXPY() for complex vectors of length N --> 8N flop

Summary of Stages:   ----- Time ------  ----- Flop ------  --- Messages ---  -- Message Lengths --  -- Reductions --
                        Avg     %Total     Avg     %Total    Count   %Total     Avg         %Total    Count   %Total
 0:      Main Stage: 4.1168e+01 100.0%  2.7490e+08 100.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%

------------------------------------------------------------------------------------------------------------------------
```

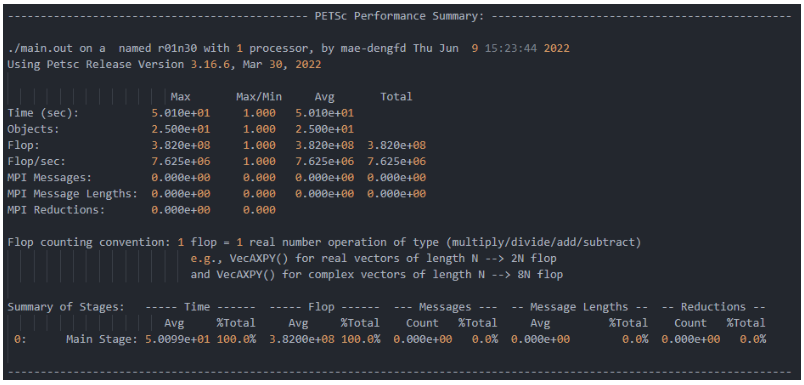
Figure 9: jacobi

pc_type 为 Additive Schwarz 时的性能报告如下：

```
------------------------------------------ PETSc Performance Summary: ------------------------------------------

./main.out on a  named r01n30 with 1 processor, by mae-dengfd Thu Jun  9 15:28:06 2022
Using Petsc Release Version 3.16.6, Mar 30, 2022

                         Max       Max/Min     Avg       Total
Time (sec):           4.742e+01     1.000   4.742e+01
Objects:              4.400e+01     1.000   4.400e+01
Flop:                 3.116e+08     1.000   3.116e+08  3.116e+08
Flop/sec:             6.571e+06     1.000   6.571e+06  6.571e+06
MPI Messages:         0.000e+00     0.000   0.000e+00  0.000e+00
MPI Message Lengths:  0.000e+00     0.000   0.000e+00  0.000e+00
MPI Reductions:       0.000e+00     0.000

Flop counting convention: 1 flop = 1 real number operation of type (multiply/divide/add/subtract)
                            e.g., VecAXPY() for real vectors of length N --> 2N flop
                            and VecAXPY() for complex vectors of length N --> 8N flop

Summary of Stages:   ----- Time ------  ----- Flop ------  --- Messages ---  -- Message Lengths --  -- Reductions --
                        Avg     %Total     Avg     %Total    Count   %Total     Avg         %Total    Count   %Total
 0:      Main Stage: 4.7423e+01 100.0%  3.1160e+08 100.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%

------------------------------------------------------------------------------------------------------------------------
```

Figure 10: Additive Schwarz

pc_type 为 LU 时的性能报告如下：



```
--------------------------------------------- PETSc Performance Summary: ----------------------------------------------

./main.out on a  named r01n30 with 1 processor, by mae-dengfd Thu Jun  9 15:23:44 2022
Using Petsc Release Version 3.16.6, Mar 30, 2022

                         Max       Max/Min     Avg       Total
Time (sec):           5.010e+01     1.000   5.010e+01
Objects:              2.500e+01     1.000   2.500e+01
Flop:                 3.820e+08     1.000   3.820e+08  3.820e+08
Flop/sec:             7.625e+06     1.000   7.625e+06  7.625e+06
MPI Messages:         0.000e+00     0.000   0.000e+00  0.000e+00
MPI Message Lengths:  0.000e+00     0.000   0.000e+00  0.000e+00
MPI Reductions:       0.000e+00     0.000

Flop counting convention: 1 flop = 1 real number operation of type (multiply/divide/add/subtract)
                            e.g., VecAXPY() for real vectors of length N --> 2N flop
                            and VecAXPY() for complex vectors of length N --> 8N flop

Summary of Stages:   ----- Time ------  ----- Flop ------  --- Messages ---  -- Message Lengths --  -- Reductions --
                        Avg     %Total     Avg     %Total    Count   %Total     Avg         %Total    Count   %Total
 0:      Main Stage: 5.0099e+01 100.0%  3.8200e+08 100.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0%

------------------------------------------------------------------------------------------------------------------------
```

Figure 11: LU

从结果报告中可以得到，对于隐式格式程序 Jacobi 运行时间最短，asm 运行时间最长。

3 报告总结

本次 Project 实现了使用显式差分格式和隐式差分格式对一维传热问题的求解，并使用 Petsc 库对代码做并行化处理，同时借助 HDF5 库实现最终数值解数据的传输和存储，通过 HDF5 文件在 Petsc 中的读写功能实现重启功能防止机器出现突然断电等突发情况。