12132386 Chen Yujie

Project Report

one-dimensional heat conduction

Algorithm framework

In this project, the governing equation of one-dimensional heat conduction problem, the boundary conditions and the initial value can be expressed as

$$\rho c\frac{\partial u}{\partial t} - \kappa\frac{\partial^2 u}{\partial x^2} = f \quad on\ \Omega \times (0,T) \tag{1}$$

$$u = 0 \quad on\ \Gamma \times (0,T) \tag{2}$$

$$u|_{t=0} = u_0 = e^x \quad in\ \Omega. \tag{3}$$

Here, Ω :=(0,1) is the 1D domain. The boundary domain is $\Gamma = \{0,1\}$. ρ, c, κ are the density, heat capacity and heat conductivity respectively. $f = sin(\pi x)$ is the heat supply per unit volume.

Both explicit and implicit method is applied to solve this problem. The derivation of the two schemes are provided as follows.

Explicit Euler method

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

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

$$\frac{\partial u}{\partial t} \sim \frac{U_i^{n+1} - U_i^n}{\Delta t}$$

$$\frac{\partial^2 u}{\partial x^2} \sim \frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2}$$

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \frac{\kappa}{\rho c}\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} + \frac{f}{\rho c}$$

$$U_i^{n+1} - U_i^n = \frac{\kappa\Delta t}{\rho c\Delta x^2}(U_{i-1}^n - 2U_i^n + U_{i+1}^n) + \frac{f\Delta t}{\rho c}$$

$$U_i^{n+1} = \frac{\kappa\Delta t}{\rho c\Delta x^2}U_{i-1}^n + (1 - 2\frac{\kappa\Delta t}{\rho c\Delta x^2})U_i^n + \frac{\kappa\Delta t}{\rho c\Delta x^2}U_{i+1}^n + \frac{f\Delta t}{\rho c}$$

$$\boldsymbol{A} = \boldsymbol{I} + \frac{\kappa\Delta t}{\rho c\Delta x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}$$

$$\boldsymbol{U}^{n+1} = \boldsymbol{A}\boldsymbol{U}^n$$

Implicit Euler method

$$\rho c\frac{\partial u}{\partial t} - \kappa\frac{\partial^2 u}{\partial x^2} = f$$
$$\frac{\partial u}{\partial t} = \frac{\kappa}{\rho c}\frac{\partial^2 u}{\partial x^2} + \frac{f}{\rho c}$$
$$\frac{\partial u}{\partial t} \sim \frac{U_i^{n+1} - U_i^n}{\Delta t}$$
$$\frac{\partial^2 u}{\partial x^2} \sim \frac{U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}}{\Delta x^2}$$
$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \frac{\kappa}{\rho c}\frac{U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}}{\Delta x^2} + \frac{f}{\rho c}$$
$$U_i^{n+1} - U_i^n = \frac{\kappa\Delta t}{\rho c\Delta x^2}\left(U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}\right) + \frac{f\Delta t}{\rho c}$$
$$-\frac{\kappa\Delta t}{\rho c\Delta x^2}U_{i+1}^{n+1} + \left(1 + 2\frac{\kappa\Delta t}{\rho c\Delta x^2}\right)U_i^{n+1} - \frac{\kappa\Delta t}{\rho c\Delta x^2}U_{i-1}^{n+1} = U_i^n + \frac{f\Delta t}{\rho c}$$
$$\boldsymbol{A} = \boldsymbol{I} + \frac{\kappa\Delta t}{\rho c\Delta x^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$
$$\boldsymbol{A}\boldsymbol{U}^{n+1} = \boldsymbol{U}^n$$

The convergence condition for explicit scheme is

$$\frac{\kappa\Delta t}{\rho c\Delta x^2} < \frac{1}{2},$$

and the implicit scheme always converges.

The steady state is characterized by $\partial u/\partial t = 0$. Thus, $u = \frac{sin(\pi x)}{\pi^2}$ at steady state.

Method Stability

In this part, Δx is fixed at 0.01. Various time steps are introduced to test the stability. For the explicit scheme, Δt is set 1e-5, 2e-5, 4e-5, 5e-5, 6e-5 respectively. The results are listed in Fig. 1. There are three criterions to finish the iteration. First, max iteration times. Second, the max error compared with theoretical solution. Third, if the difference of u between two iteration step is small enough, the iteration would break, since if Δx is not small enough, it can reach the second criterion. In the output, "value not change" means hitting the third criterion. Value at domain centre is present for comparison with theoretical data, 0.1013212.

For implicit scheme, Δt is set much larger, at 1e-3, 4e-4, 5e-5. It can be observed that implicit scheme always converges. For explicit scheme, when Δt is less than 5e-5, it would converge, otherwise diverge, which is in agreement with prediction. Larger time step results in faster convergence.

The value of u is displayed with respect to x. As shown in Fig. 3, which is in agreement with theoretical solution.

```
mae-chenyj::login03 { ~/proj/explicit_mat/stability }
-> cat 1e-5.log
dt=1e-05
Matrix size is 101 by 101
dx=0.010000
value not change
End at 123785th iter, err=1.84659e-05
max_value at centre: 0.10134
mae-chenyj::login03 { ~/proj/explicit_mat/stability }
-> cat 2e-5.log
dt=2e-05
Matrix size is 101 by 101
dx=0.010000
value not change
End at 65402th iter, err=1.33993e-05
max_value at centre: 0.101335
mae-chenyj::login03 { ~/proj/explicit_mat/stability }
-> cat 4e-5.log
dt=4e-05
Matrix size is 101 by 101
dx=0.010000
value not change
End at 34454th iter, err=1.08665e-05
max_value at centre: 0.101332
mae-chenyj::login03 { ~/proj/explicit_mat/stability }
-> cat 5e-5.log
dt=5e-05
Matrix size is 101 by 101
dx=0.010000
value not change
End at 24247th iter, err=2.13369e-05
max_value at centre: 0.101343
mae-chenyj::login03 { ~/proj/explicit_mat/stability }
-> cat 6e-5.log
dt=6e-05
Matrix size is 101 by 101
dx=0.010000
End at 200000th iter, err=inf.
max_value at centre: inf.
mae-chenyj::login03 { ~/proj/explicit_mat/stability }
-> cat 8e-5.log
dt=8e-05
Matrix size is 101 by 101
dx=0.010000
End at 200000th iter, err=inf.
max_value at centre: inf.
```

Figure 1: stability test for explicit scheme

```
mae-chenyj::login03 { ~/proj/implicit/stability }
-> cat 1e-3.log
Matrix size is 101 by 101
dx=0.010000
dt=0.001000
value not change
End at 1712th iter, err=8.436e-06
max_value=0.10133
mae-chenyj::login03 { ~/proj/implicit/stability }
-> cat 4e-4.log
Matrix size is 101 by 101
dx=0.010000
dt=0.000400
value not change
End at 4037th iter, err=8.58723e-06
max_value=0.10133
mae-chenyj::login03 { ~/proj/implicit/stability }
-> cat 5e-5.log
Matrix size is 101 by 101
dx=0.010000
dt=0.000050
value not change
End at 28026th iter, err=1.03608e-05
max_value=0.101332
```

Figure 2: stability test for implicit scheme

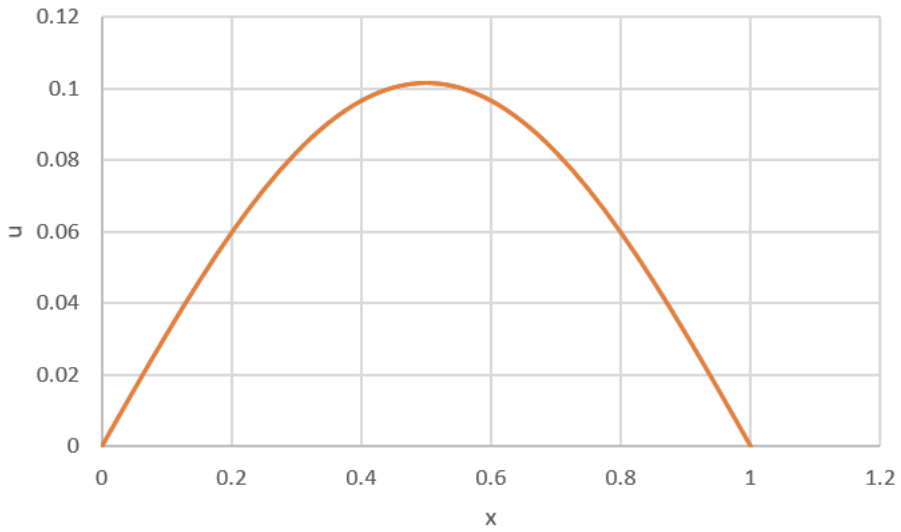

Figure 3: distribution of u

Error Analysis

Here, I focused on the value at the center of the domain, i.e. at x=0.5. In explicit scheme, the error of the data is related to the discretization of space and time:

$$e \approx C_1 \Delta x^{\alpha} + C_2 \Delta t^{\beta}. \tag{4}$$

To get the value of α , Δ t is fixed at 5e-6, and the domain is divided into 8, 16, 32, 64, 128, 256 parts respectively. The value of u changes little after 1.5 seconds, i.e., the steady state is roughly after 1.5 seconds. In this part, the iteration number is set 400000 to stop after 2 seconds. The corresponding error is defined as the difference between the numerical data and the theoretical data at x=0.5. Note that the theoretical data at x=0.5 at steady state is $1/(\pi^2)$. The results are plot is log form. As shown in Fig. 4, the slop of the line is around 2.001. Thus, the value of α is determined as 2.

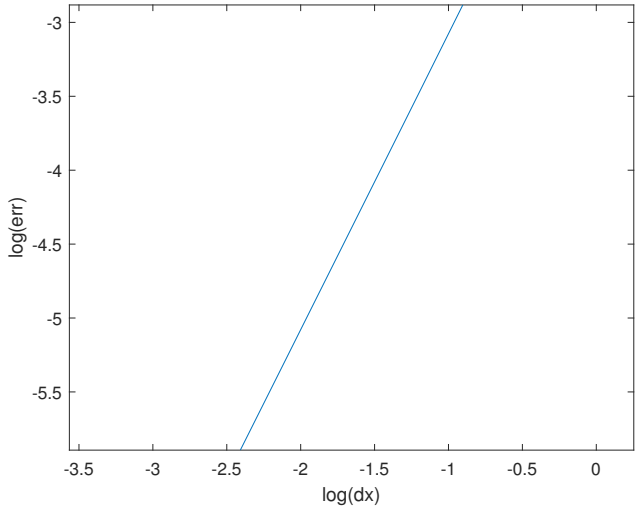

Figure 4: err versus dx

To get the value of β, Δx is fixed at 0.005, Δt varies at 1e-6, 5e-6, 2.5e-6, 1.25e-6, 6.25e-7, 3.125e-7. Each sample stop at 0.1 second. The error is defined as the difference between each sample at the sample of $\Delta t = 3.125e-7$. The results are plot in log form. As shown in Fig. 5, the slop of the line passing through the red points is 0.94. Thus, the value of β is determined as 1.

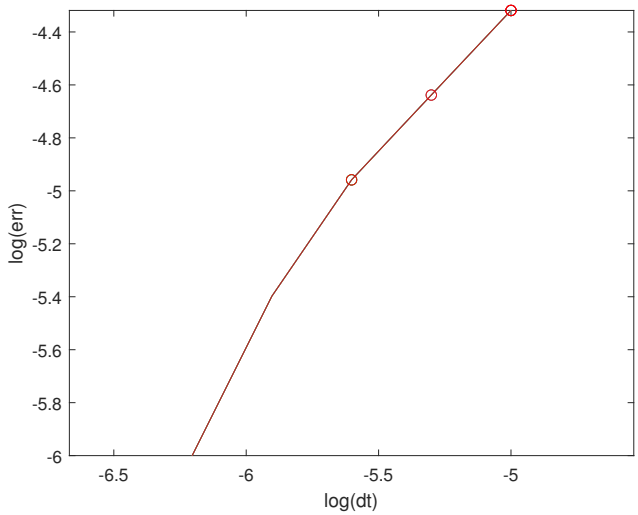

Figure 5: err versus dt

Parallelism

In this part, examples of parallel behaviours are presented from explicit scheme. To observe Amdahl's Law, the domain is divided into 40000 parts, and the processor number is set 1, 2, 4, 8 respectively. Fig. 6 shows the speed-up ratio with respect to processor number. The yellow line has a slope of 1 for reference.

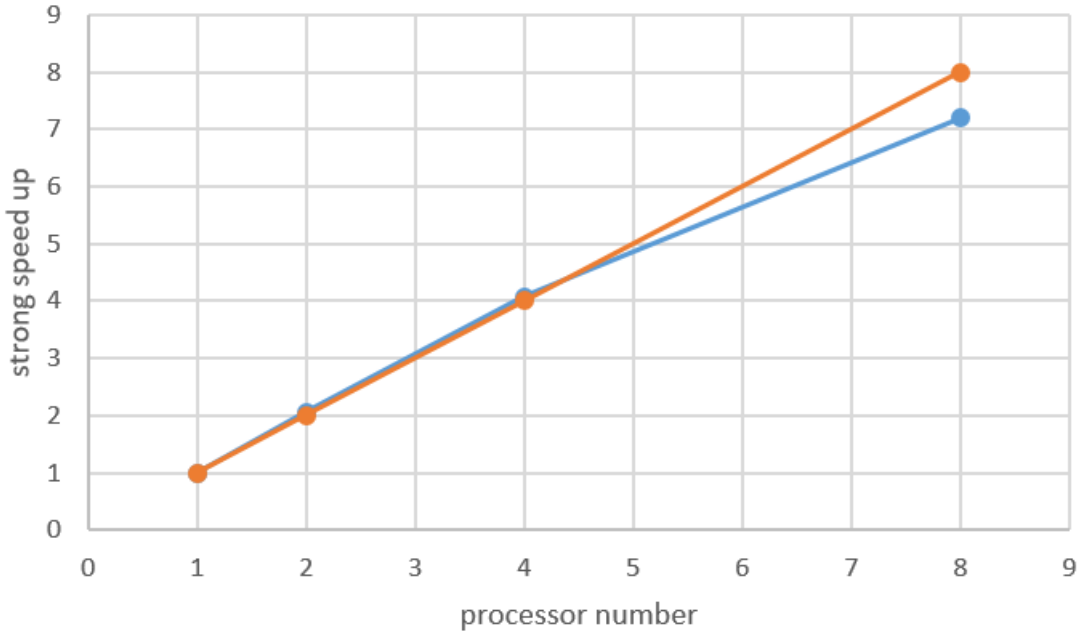

Figure 6: speed-up ratio versus processor number

To observe Gustafson's Law, division number is set 500, 1000, 1500, 2000, 2500, 3000 with processor number at 1, 4, 9, 16, 27, 36 respectively. Iteration times is 1e7. Fig. 7 illustrates the time spent with respect to processor number. Time spent increase gradually as processor number increases. While the time cost using 4 processors is significant higher abnormally, which needs further discussion.

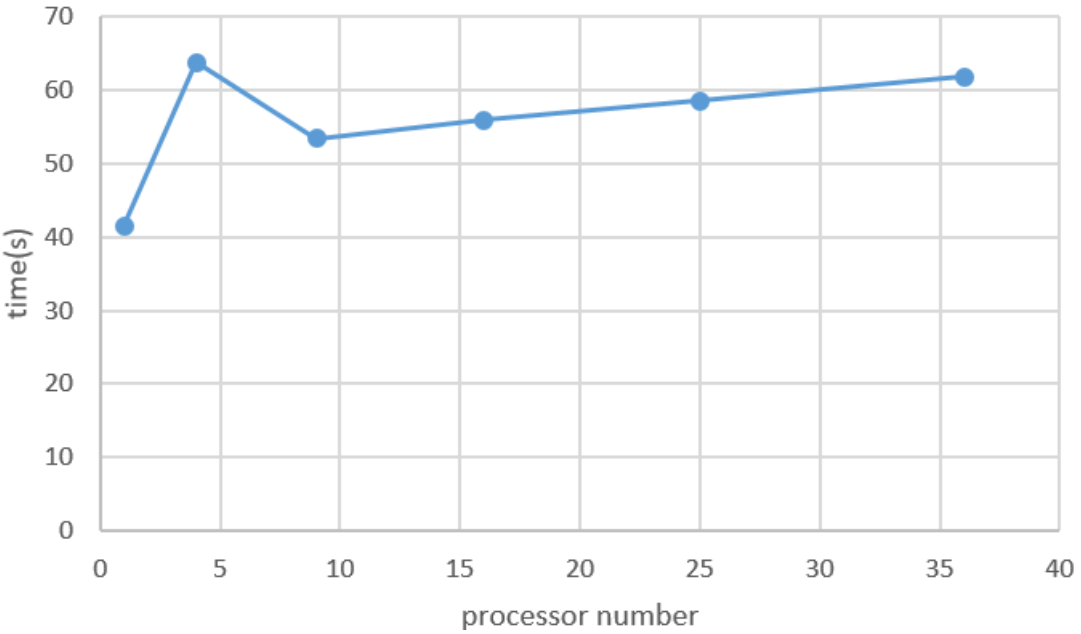

Figure 7: time spent versus processor number

Visualization

To visualize the results, output data is written in tecplot manner. Fig. 8 is from implicit scheme with matrix size of 1000 by 1000.

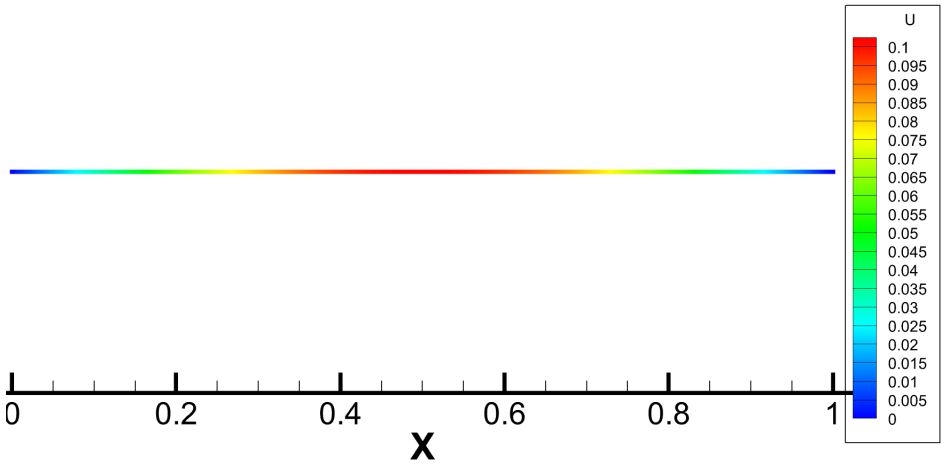

Figure 8: result shown by tecplot. X is the coordinate

Data log in HDF5

To use hdf5 to record the data, ¡petscviewerhdf5.h¿ was included. The data of u is recorded in a hdf5 file every 100 steps. Fig. 9 gives an example of the record where dx=0.025.

```
mae-chenyj::login03 { ~/proj/explicit_mat/h5_write }
-> h5dump data.h5
HDF5 "data.h5" {
GROUP "/" {
   DATASET "explicit-data" {
      DATATYPE  H5T_IEEE_F64LE
      DATASPACE  SIMPLE { ( 41 ) / ( 41 ) }
      DATA {
      (0): 0, 0.0079537, 0.0158584, 0.0236652, 0.0313262, 0.0387941,
      (6): 0.0460227, 0.0529677, 0.059586, 0.065837, 0.0716821, 0.0770852,
      (12): 0.0820131, 0.0864354, 0.0903247, 0.0936572, 0.0964122, 0.0985728,
      (18): 0.100126, 0.101061, 0.101374, 0.101061, 0.100126, 0.0985728,
      (24): 0.0964122, 0.0936572, 0.0903247, 0.0864354, 0.0820131, 0.0770852,
      (30): 0.0716821, 0.065837, 0.059586, 0.0529677, 0.0460227, 0.0387941,
      (36): 0.0313262, 0.0236652, 0.0158584, 0.0079537, 2.44929e-20
      }
      ATTRIBUTE "timestepping" {
         DATATYPE  H5T_STD_I32LE
         DATASPACE  SCALAR
         DATA {
         (0): 0
         }
      }
   }
}
}
```

Figure 9: sample of HDF5 file

Profiling analysis

The profile of the code is provided by callgrind. The two figures bellow are samples given by implicit scheme and explicit scheme. In the implicit scheme, most time is consumed in the linear solver and finding the location of the functions in the library. In the explicit scheme, matrix-vector multiplication consumed most time.

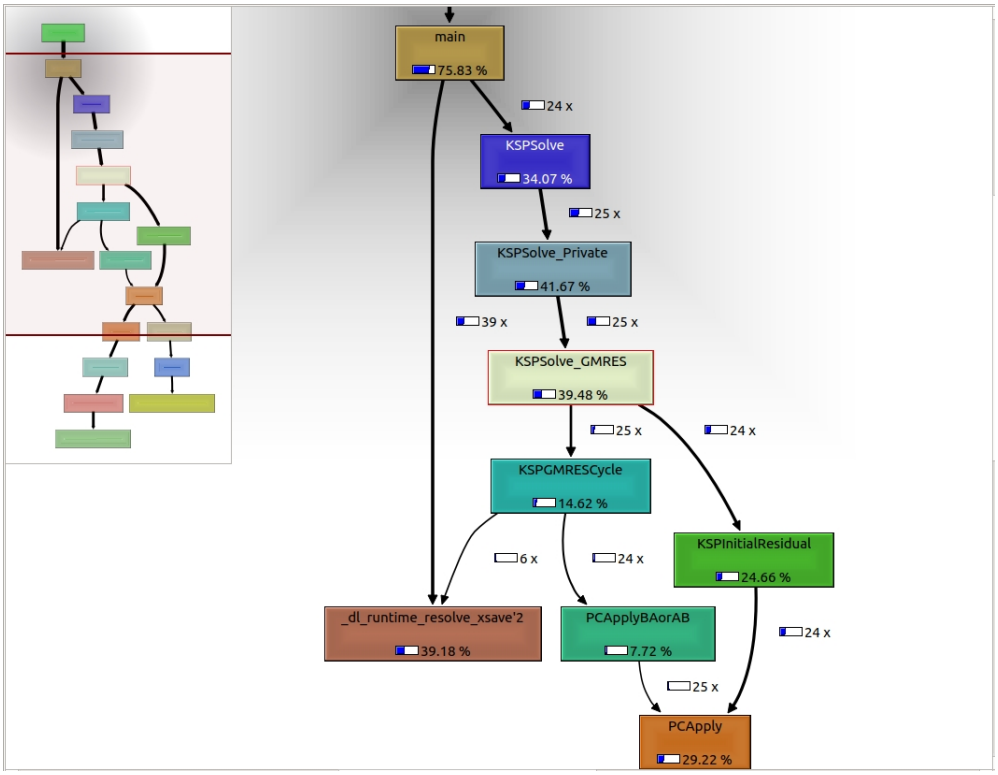

Figure 10: profile of implicit scheme

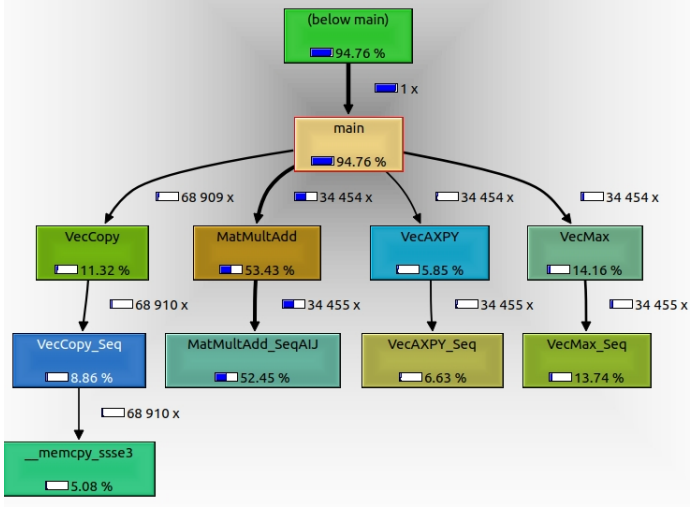

Figure 11: profile of explicit scheme

Preconditioner

The result in this part maybe irrational, since there are some bugs I can't handle in my implicit coding: when the processor number is larger than 4, the program tends to give wrong result. Various of preconditioners are tested, as shown in Fig. 12. All the jobs in this script gave out the correct result.

```
mae-chenyj::login03 { ~/PC_test }
-> cat script_pc
#!/bin/bash
#BSUB -J 1.1
#BSUB -q debug
#BSUB -n 4
#BSUB -o %J.err
#BSUB -R "span[ptile=40]"


mpirun -np 1 ./main -n 10000 -dt 0.1 -log_view > default-n2.log 2>&1
mpirun -np 1 ./main -n 10000 -dt 0.1 -log_view -ksp_type cg -pc_type jacobi >cg_jacobi-n2.log 2>&1
mpirun -np 1 ./main -n 10000 -dt 0.1 -log_view -pc_type lu -pc_factor_mat_solver_type mumps > mumps-n2.log 2>&1
mpirun -np 1 ./main -n 10000 -dt 0.1 -log_view -ksp_type cg -pc_type asm >cg_asm-n2.log 2>&1
mpirun -np 1 ./main -n 10000 -dt 0.1 -log_view -ksp_type cg -pc_type hypre >cg_hypre-n2.log 2>&1


mae-chenyj::login03 { ~/PC_test }
-> cat script_pc2
#!/bin/bash
#BSUB -J 1.1
#BSUB -q debug
#BSUB -n 4
#BSUB -o %J.err
#BSUB -R "span[ptile=40]"


mpirun -np 2 ./main -n 10000 -dt 0.1 -log_view -ksp_type gmres -pc_type jacobi >gmres_jacobi-n2.log 2>&1
mpirun -np 2 ./main -n 10000 -dt 0.1 -log_view -pc_type lu -pc_factor_mat_solver_type mumps > mumps-n2.log 2>&1
mpirun -np 2 ./main -n 10000 -dt 0.1 -log_view -ksp_type gmres -pc_type asm >gmres_asm-n2.log 2>&1
```

Figure 12: job script for different preconditioners

The hypre-type preconditioner gave the best result.