Document

Qiujiang Jin

Content

1 Introduction

This is the modeling document for the software that solves the steady-state heat equation in one- and two-dimensions. This document highlights the governing equations, nomenclature, boundary conditions, numerical approximations, algorithms to implement the solver, required memory, verification methodology, input/runtime options, build procedures and example results.

2 Preliminary

2.1 Equations

The steady-state heat equation with a constant coefficient in one dimensions is given by:

$$-k\frac{d^2T(x)}{dx^2} = q(x) \quad x \in (0,1)$$

Where k is the given constant and it means thermal conductivity, T is the function we want to solve and it means material temperature, and q is the given function and it means the heat source term. There is only one equation. The steady-state heat equation with a constant coefficient in two dimensions is given by:

$$-k\nabla^2 T(x,y) = q(x,y) \quad (x,y) \in \Omega = (0,1) \times (0,1)$$

which is equivalent to

$$-k(\frac{\partial^2 T(x,y)}{\partial x^2} + \frac{\partial^2 T(x,y)}{\partial y^2}) = q(x,y)$$

2.2 Boundary Conditions

I choose to use the Dirichlet boundary conditions. So for the one dimensional case it is:

$$T(0) = \alpha \quad T(1) = \beta$$

Where α and β are 2 given constants. For the second dimensional case it is:

$$T(x,y)|_{\partial\Omega} = f(x,y)$$

Where $f(x,y)$ is the given function defined on $\partial\Omega$

2.3 Other Assumptions

We know the values of k and function $q(x,y)$. We also know the values of α, β and the boundary function $f(x,y)$. The domain for the one dimensional case is $(0,1)$ and the mesh sizes are all equal to h. The domain for the two dimensional case is $\Omega = (0,1) \times (0,1)$ and the mesh sizes are all equal to h for both x-axis and y-axis. My scheme is node-based and I assume a square domain for the 2D case. For the 4th order scheme, I also assume to know the boundary condition of the points outside of the bound so that I can establish the linear system for the points near the boundary.

3 Numerical Methods

3.1 Finite Difference Methods

Using the Taylor expansion we can derive the finite-difference approximations for the second derivative in the heat equation. For the one dimensional case, we have 2nd-order finite-difference approximations:

$$\frac{d^2T(x)}{dx^2} = \frac{T(x+h) + T(x-h) - 2T(x)}{h^2} - \frac{2h^2}{4!}\frac{d^4T(x)}{dx^4} + O(h^4)$$

Denote $x_i = ih$ and $T_i = T(x_i)$, the discrete approximations of the heat equation using these formulations is:

$$\frac{d^2T(x_i)}{dx^2} = \frac{T_{i+1} + T_{i-1} - 2T_i}{h^2} - \frac{2h^2}{4!}\frac{d^4T(x_i)}{dx^4} + O(h^4)$$

So we have:

$$-k\frac{T_{i+1} + T_{i-1} - 2T_i}{h^2} = q(x_i)$$

we have 4th-order finite-difference approximations:

$$\begin{aligned}\frac{d^2T(x)}{dx^2} &= \frac{-T(x+2h) + 16T(x+h) - 30T(x) + 16T(x-h) - T(x-2h)}{12h^2} \\ &\quad + \frac{8h^4}{6!}\frac{d^6T(x)}{dx^6} + O(h^6)\end{aligned}$$

The discrete approximations of the heat equation using these formulations is:

$$\begin{aligned}\frac{d^2T(x_i)}{dx^2} &= \frac{-T_{i+2} + 16T_{i+1} - 30T_i + 16T_{i-1} - T_{i-2}}{12h^2} \\ &\quad + \frac{8h^4}{6!}\frac{d^6T(x_i)}{dx^6} + O(h^6)\end{aligned}$$

So we have:

$$-k\frac{-T_{i+2} + 16T_{i+1} - 30T_i + 16T_{i-1} - T_{i-2}}{12h^2} = q(x_i)$$

And similarly, we can derive the 2nd-order finite-difference approximations for the second dimensional case:

$$\begin{aligned}\nabla^2T(x,y) &= \frac{T(x+h,y) + T(x-h,y) - 2T(x,y)}{h^2} + \frac{T(x,y+h) + T(x,y-h) - 2T(x,y)}{h^2} \\ &\quad - \frac{2h^2}{4!}\left(\frac{\partial^4T(x,y)}{\partial x^4} + \frac{\partial^4T(x,y)}{\partial y^4}\right) + O(h^4)\end{aligned}$$

Denote $x_i = ih$, $y_j = jh$ and $T_{i,j} = T(x_i, y_j)$, the discrete approximations of the heat equation using these formulations is:

$$\begin{aligned}\nabla^2T(x_i,y_j) &= \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{h^2} + \frac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{h^2} \\ &\quad - \frac{2h^2}{4!}\left(\frac{\partial^4T(x_i,y_j)}{\partial x^4} + \frac{\partial^4T(x_i,y_j)}{\partial y^4}\right) + O(h^4)\end{aligned}$$

So we have:

$$-k(\frac{T_{i+1,j}+T_{i-1,j}-2T_{i,j}}{h^2}+\frac{T_{i,j+1}+T_{i,j-1}-2T_{i,j}}{h^2})=q(x_i,y_j)$$

we have 4th-order finite-difference approximations:

$$\begin{aligned}\nabla^2 T(x,y) &= \frac{-T(x+2h,y)+16T(x+h,y)-30T(x,y)+16T(x-h,y)-T(x-2h,y)}{12h^2}\\ &+\frac{-T(x,y+2h)+16T(x,y+h)-30T(x,y)+16T(x,y-h)-T(x,y-2h)}{12h^2}\\ &+\frac{8h^4}{6!}(\frac{\partial^6 T(x,y)}{\partial x^6}+\frac{\partial^6 T(x,y)}{\partial y^6})+O(h^6)\end{aligned}$$

The discrete approximations of the heat equation using these formulations is:

$$\begin{aligned}\nabla^2 T(x_i,y_j) &= \frac{-T_{i+2,j}+16T_{i+1,j}-30T_{i,j}+16T_{i-1,j}-T_{i-2,j}}{12h^2}\\ &+\frac{-T_{i,j+2}+16T_{i,j+1}-30T_{i,j}+16T_{i,j-1}-T_{i,j-2}}{12h^2}\\ &+\frac{8h^4}{6!}(\frac{\partial^6 T(x_i,y_j)}{\partial x^6}+\frac{\partial^6 T(x_i,y_j)}{\partial y^6})+O(h^6)\end{aligned}$$

So we have:

$$\begin{aligned}&-k(\frac{-T_{i+2,j}+16T_{i+1,j}-30T_{i,j}+16T_{i-1,j}-T_{i-2,j}}{12h^2}\\ &+\frac{-T_{i,j+2}+16T_{i,j+1}-30T_{i,j}+16T_{i,j-1}-T_{i,j-2}}{12h^2})=q(x_i,y_j)\end{aligned}$$

3.2 Figures of Discretized Meshes

Here is the representative figures of 2D discretized meshes with domain $\Omega = (0,1)\times(0,1)$ and mesh size $h=0.1$ for both axis. My scheme is node-based. The domain is meshed in squares.

Here is the representative figures of 1D discretized meshes with domain $(0,1)$ and mesh size $h=0.1$

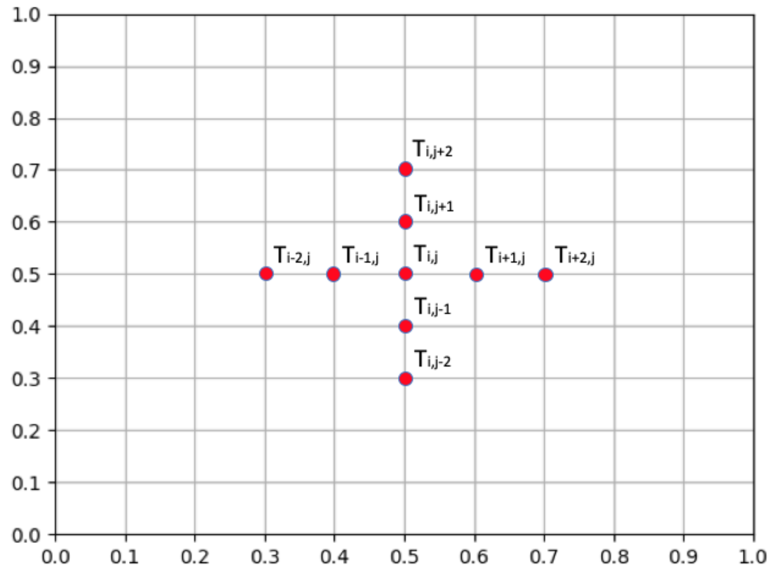


Figure 1: figures of 2D discretized meshes

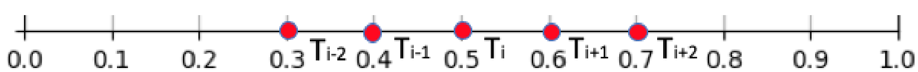


Figure 2: figures of 1D discretized meshes

3.3 Linear Systems

First consider the 2nd-order finite-difference schemes For the one dimensional case, consider $N = \frac{1}{h}$. Then $x_0 = 0$ and $x_N = 1$. We want to solve T_i for $1 \leq i \leq N-1$. And $T_0 = T(x_0) = T(0) = \alpha$ and $T_N = T(x_N) = T(1) = \beta$. So the linear system is:

$$
\begin{cases}
i = 1 & -k\dfrac{T_0 + T_2 - 2T_1}{h^2} = q(x_1) \\
i = 2 & -k\dfrac{T_1 + T_3 - 2T_2}{h^2} = q(x_2) \\
i = 3 & -k\dfrac{T_2 + T_4 - 2T_3}{h^2} = q(x_3) \\
\ldots\ldots & \\
i = N-1 & -k\dfrac{T_{N-2} + T_N - 2T_{N-1}}{h^2} = q(x_{N-1})
\end{cases}
$$

Matrix Form $A_1 t_1 = b_1$ with $A_1 \in R^{N-1 \times N-1}$ and $t_1, b_1 \in R^{N-1}$

$$A_1 = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & ... & & \\ & & & -1 & 2 \end{bmatrix} \quad t_1 = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ ... \\ T_{N-1} \end{bmatrix} \quad b_1 = \begin{bmatrix} \frac{h^2}{k} q(x_1) + \alpha \\ \frac{h^2}{k} q(x_2) \\ \frac{h^2}{k} q(x_3) \\ ... \\ \frac{h^2}{k} q(x_{N-1}) + \beta \end{bmatrix}$$

the number of non-zero entries on an interior row of the matrix A_1 is 3

For the second dimensional case, consider $N = \frac{1}{h}$. Then $x_0 = y_0 = 0$ and $x_N = y_N = 1$. We want to solve $T_{i,j}$ for $1 \le i, j \le N-1$. And $T_{0,j} = f(0, y_j)$, $T_{N,j} = f(1, y_j)$ for $0 \le j \le N$ and $T_{i,0} = f(x_i, 0)$, $T_{i,N} = f(x_i, 1)$ for $0 \le i \le N$. So the linear system is:

$$-k \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j}}{h^2} = q(x_i, y_j) \quad 1 \le i, j \le N-1 \quad (1)$$

Matrix Form $A_2 t_2 = b_2$ with $A_1 \in R^{(N-1)^2 \times (N-1)^2}$ and $t_1, b_1 \in R^{(N-1)^2}$

$$A_2 = \begin{bmatrix} B_2 & -I & & & \\ -I & B_2 & -I & & \\ & -I & B_2 & -I & \\ & & ... & & \\ & & & -I & B_2 \end{bmatrix} \quad B_2 = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & ... & & \\ & & & -1 & 4 \end{bmatrix}$$

And $I \in R^{N-1 \times N-1}$ is the identity matrix.

$$t_2 = [T_{1,1}, ..., T_{N-1,1}, T_{1,2}, ..., T_{N-1,2}, ..., T_{1,N-1}, ..., T_{N-1,N-1}]^T$$

$$b_2 = [\frac{h^2}{k} q_{x_1,y_1} + f(0,y_1) + f(x_1,0), \quad ..., \quad \frac{h^2}{k} q_{x_i,y_1} + f(x_i,0), \quad ..., \quad \frac{h^2}{k} q_{x_{N-1},y_1} + f(1,y_1) + f(x_{N-1},0),$$

$$..., \quad \frac{h^2}{k} q_{x_i,y_j}, \quad ...,$$

$$\frac{h^2}{k} q_{x_1,y_{N-1}} + f(0,y_{N-1}) + f(x_1,1), \quad ..., \quad \frac{h^2}{k} q_{x_i,y_{N-1}} + f(x_i,1), \quad ..., \quad \frac{h^2}{k} q_{x_{N-1},y_{N-1}} + f(1,y_{N-1}) + f(x_{N-1},1)]^T$$

the number of non-zero entries on an interior row of the matrix A_2 is 5

Then consider the 4th-order finite-difference schemes For the one dimensional case, The linear system is:

$$-k \frac{-T_{i+2} + 16T_{i+1} - 30T_i + 16T_{i-1} - T_{i-2}}{12h^2} = q(x_i) \quad 1 \le i \le N-1$$

Notice that for $i = 1$ and $i = N-1$ we need information of $T_{-1} = \alpha^*$ and $T_{N+1} = \beta^*$. Matrix Form $A_3 t_3 = b_3$ with $A_3 \in R^{N-1 \times N-1}$ and $t_3, b_3 \in R^{N-1}$

$$A_3 = \begin{bmatrix} 30 & -16 & 1 & & & \\ -16 & 30 & -16 & 1 & & \\ 1 & -16 & 30 & -16 & 1 & \\ & & ... & ... & ... & \\ & & 1 & -16 & 30 & -16 \\ & & & 1 & -16 & 30 \end{bmatrix} \quad t_3 = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ ... \\ T_{N-2} \\ T_{N-1} \end{bmatrix} \quad b_3 = \begin{bmatrix} \frac{12h^2}{k} q(x_1) + 16\alpha - \alpha^* \\ \frac{12h^2}{k} q(x_2) - \alpha \\ \frac{12h^2}{k} q(x_3) \\ ... \\ \frac{12h^2}{k} q(x_{N-2}) - \beta \\ \frac{12h^2}{k} q(x_{N-1}) + 16\beta - \beta^* \end{bmatrix}$$

the number of non-zero entries on an interior row of the matrix A_3 is 5

For the second dimensional case, the linear system is:

$$-k \frac{-T_{i+2,j} - T_{i-2,j} - T_{i,j+2} - T_{i,j-2} + 16T_{i+1,j} + 16T_{i-1,j} + 16T_{i,j+1} + 16T_{i,j-1} - 60T_{i,j}}{12h^2} = q(x_i, y_j)$$

$$1 \leq i, j \leq N-1$$

Notice that for the point next to the boundary we need information $T_{-1,j} = f^*(x_{-1}, y_j)$ and $T_{N+1,j} = f^*(x_{N+1}, y_j)$ for $-1 \leq j \leq N+1$ and $T_{i,-1} = f^*(x_i, y_{-1})$ and $T_{i,N+1} = f^*(x_i, y_{N+1})$ for $-1 \leq i \leq N+1$. Matrix Form $A_4 t_4 = b_4$ with $A_4 \in R^{(N-1)^2 \times (N-1)^2}$ and $t_4, b_4 \in R^{(N-1)^2}$

$$A_4 = \begin{bmatrix} R & Q & P & & & \\ Q & R & Q & P & & \\ P & Q & R & Q & P & \\ & & ... & ... & ... & \\ & & P & Q & R & Q \\ & & & P & Q & R \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & ... & & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \quad Q = \begin{bmatrix} -16 & & & & & \\ & -16 & & & & \\ & & -16 & & & \\ & & ... & & & \\ & & & & -16 & \\ & & & & & -16 \end{bmatrix}$$

$$R = \begin{bmatrix} 60 & -16 & 1 & & & \\ -16 & 60 & -16 & 1 & & \\ 1 & -16 & 60 & -16 & 1 & \\ & & ... & ... & ... & \\ & & 1 & -16 & 60 & -16 \\ & & & 1 & -16 & 60 \end{bmatrix}$$

Where $I \in R^{N-1 \times N-1}$ is the indentity matrix.

$$t_4 = [T_{1,1}, ..., T_{N-1,1}, T_{1,2}, ..., T_{N-1,2}, ..., T_{1,N-1}, ..., T_{N-1,N-1}]^T$$

$$b_4 = [$$

$$\frac{12h^2}{k}q_{x_1,y_1}+16f(0,y_1)+16f(x_1,0)-f^*(x_{-1},y_1)-f^*(x_1,y_{-1}), \frac{12h^2}{k}q_{x_2,y_1}+16f(x_2,0)-f^(0,y_1)-f^*(x_2,y_{-1}),$$

$$..., \quad \frac{12h^2}{k}q_{x_i,y_1}+16f(x_i,0)-f^*(x_i,y_{-1}), \quad ...,$$

$$\frac{12h^2}{k}q_{x_{N-2},y_1}+16f(x_{N-2},0)-f(1,y_1)-f^*(x_{N-2},y_{-1}),$$

$$\frac{12h^2}{k}q_{x_{N-1},y_1}+16f(1,y_1)+16f(x_{N-1},0)-f^*(x_{N+1},y_1)-f^*(x_{N-1},y_{-1}),$$

$$\frac{12h^2}{k}q_{x_1,y_2}+16f(0,y_2)-f^*(x_{-1},y_2)-f(x_1,0), \frac{12h^2}{k}q_{x_2,y_1}-f(0,y_1)-f(x_2,0),$$

$$..., \quad \frac{12h^2}{k}q_{x_i,y_2}-f(x_i,0), \quad ...,$$

$$\frac{12h^2}{k}q_{x_{N-2},y_2}-f(1,y_2)-f(x_{N-2},0), \frac{12h^2}{k}q_{x_{N-1},y_2}+16f(1,y_2)-f^*(x_{N+1},y_2)-f(x_{N-1},0),$$

$$..., \quad \frac{12h^2}{k}q_{x_1,y_j}+16f(0,y_j)-f^*(x_{-1},y_j), \frac{12h^2}{k}q_{x_2,y_j}-f(0,y_j),$$

$$..., \quad \frac{12h^2}{k}q_{x_i,y_j}, \quad ...,$$

$$\frac{12h^2}{k}q_{x_{N-2},y_j}-f(1,y_j), \frac{12h^2}{k}q_{x_{N-1},y_j}+16f(1,y_j)-f^*(x_{N+1},y_j), \quad ...,$$

$$\frac{12h^2}{k}q_{x_1,y_{N-2}}+16f(0,y_{N-2})-f^*(x_{-1},y_{N-2})-f(x_1,1), \frac{12h^2}{k}q_{x_2,y_{N-2}}-f(0,y_{N-2})-f(x_2,1),$$

$$..., \quad \frac{12h^2}{k}q_{x_i,y_{N-2}}-f(x_i,0), \quad ...,$$

$$\frac{12h^2}{k}q_{x_{N-2},y_{N-2}}-f(1,y_{N-2})-f(x_{N-2},0), \frac{12h^2}{k}q_{x_{N-1},y_{N-2}}+16f(1,y_{N-2})-f^*(x_{N+1},y_{N-2})-f(x_{N-1},1),$$

$$\frac{12h^2}{k}q_{x_1,y_{N-1}}+16f(0,y_{N-1})+16f(x_1,1)-f^*(x_{-1},y_{N-1})-f^*(x_1,y_{N+1}),$$

$$\frac{12h^2}{k}q_{x_2,y_{N-1}}+16f(x_2,1)-f(0,y_{N-1})-f^*(x_2,y_{N+1}),$$

$$..., \quad \frac{12h^2}{k}q_{x_i,y_{N-1}}+16f(x_i,1)-f^*(x_i,y_{N+1}), \quad ...,$$

$$\frac{12h^2}{k}q_{x_{N-2},y_{N-1}}+16f(x_{N-2},0)-f(1,y_{N-1})-f^*(x_{N-2},y_{N+1}),$$

$$\frac{12h^2}{k}q_{x_{N-1},y_{N-1}}+16f(1,y_{N-1})+16f(x_{N-1},1)-f^*(x_{N+1},y_{N-1})-f^*(x_{N-1},y_{N+1}),]^T$$

the number of non-zero entries on an interior row of the matrix A_4 is 9

3.4 Algorithms to Solve Linear Systems

I use 2 simple iterative methods to solve the linear system.
First consider the Jacobi method to solve $Ax = b$
Suppose $A = D + R$ where D is the diagonal component of A and R is the remainder. The Jacobi method is $x^{k+1} = D^{-1}(b - Rx^k)$, which is equivalent to:

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^k) \quad i = 1, ..., n$$

The pseudo-code is the following:

```
Input: initial guess x^0, diagonal dominant matrix A, right handed vector
  b, convergence criterion ϵ
Output: solution when convergence is reached
k = 0
while ||b − Ax^k|| ≥ ϵ do
    for i := 1 to n do
        s = 0
        for j := 1 to n do
            if j ≠ i then
                s = s + a_ij x_j^k
            end
        end
        x_i^(k+1) = (1/a_ii)(b_i − s)
    end
    k = k + 1
end
```

Algorithm 1: Jacobi Method

Then consider the Gauss Seidel method to solve $Ax = b$
Suppose $A = L + U$ where L is the diagonal and lower triangular component of A and U is the strictly upper triangular component of A. The Gauss Seidel method is $x^{k+1} = L^{-1}(b - Ux^k)$, which is equivalent to:

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k) \quad i = 1, ..., n$$

The pseudo-code is the following:

Input: initial guess x^0, diagonal dominant matrix A, right handed vector b, convergence criterion ϵ
Output: solution when convergence is reached
$k = 1$
while $||b - Ax^k|| \geq \epsilon$ **do**
  for $i := 1$ *to* n **do**
    $s = 0$
    for $j := 1$ *to* $i - 1$ **do**
      $s = s + a_{ij}x_j^k$
    end
    for $j := i + 1$ *to* n **do**
      $s = s + a_{ij}x_j^{k-1}$
    end
    $x_i^k = \frac{1}{a_{ii}}(b_i - s)$
  end
  $k = k + 1$
end

Algorithm 2: Gauss Seidel Method

3.5 Required Memory

I use sparse matrix in my numerical implementation. So just need to store the position(which is an integer) and the value(which is double-precision float number) of the non-zero elements in the matrix. So the memory of the matrix is just the number of non-zero elements times 12 Bytes(4 Bytes for integer and 8 Bytes for double-precision float number). And for the right hand sided vector each element requires 8 Bytes.

For the one dimensional problem, the 2nd-order schemes requires $(3N \times 12 + N \times 8) = 44N$ Bytes. the 4th-order schemes requires $(5N \times 12 + N \times 8) = 68N$ Bytes.

For the two dimensional problem, the 2nd-order schemes requires $(5N^2 \times 12 + N^2 \times 8) = 68N^2$ Bytes. the 4th-order schemes requires $(9N^2 \times 12 + N^2 \times 8) = 116N^2$ Bytes.

4 Implementation

4.1 Build Procedures

Here is how to build the code

$ autoreconf -i

$ module load hdf5

$ module load petsc/3.9-uni

$ export PKGPATH=/work/00161/karl/stampede2/public

$ ./configure --with-masa=$PKGPATH/masa-gnu7-0.50
--with-grvy=$PKGPATH/grvy-gnu7-0.34 --with-hdf5=$TACC_HDF5_DIR

Then you can *make* or *make check* and run the code: *./solver input.dat*

4.2 Input Options

Here is the various input options relevant for your code

1. dimension
   dimension is either 1 or 2
2. xmin
   minimal location on the x-axis
3. xmax
   maximal location on the x-axis
4. ymin
   minimal location on the y-axis
5. ymax
   maximal location on the y-axis
6. nx
   number of mesh points on the x-axis
7. ny
   number of mesh points on the y-axis
8. fd_method
   finite difference scheme is either 2 or 4
9. iter_method
   iteration method is either Jacobi or Gauss-Seidel
10. verify_mode
    verification mode is either 0 or 1. 0 means to use the verification and 1 means not to use the verification
11. output_mode
    output mode is either 0 or 1. 0 means standard output mode and 1 means debug output mode
12. k
    k is the thermal conductivity

13. eps
    This is the error tolerance of the iterative solver

14. max_iter
    This is the maximal number of solver iterations

15. output_file
    This is the name of the file containing numerical solutions generated by the solver

16. masa_file
    This is the name of the file containing analytical solutions generated by the masa

Notice that for convenience it is required that xmin $<$ xmax, ymin $<$ ymax, xmax $-$ xmin $=$ ymax $-$ ymin, nx $=$ ny. If you give the wrong inputs or inputs that don't obey the requirements, the program will give the error information and exit.

4.3 Verification Procedures

Make sure that verification mode is 0. Then after building the code, *make* then *make check* to run the regression test. Then give the command *./solver input.dat* to run the code in verification mode. Here is the example of the standard output of verification mode.

```
**  Finite-difference based Heat Equation Solver (steady-state)


**  Parsing runtime options from the file input.dat
**  Runtime mesh settings:
--> dimension                  = 1
--> xmin                       = 0.000000000000
--> xmax                       = 1.000000000000
--> nx                         = 20


** Runtime solver settings:
--> finite difference order    = 2nd
--> iteration method           = Gauss-Seidel
--> verification mode          = verify
--> output mode                = standard
--> thermal conductivity       = 1.000000000000
--> convergence tolerance      = 0.000000000001
--> max iterations             = 300000
--> numerical solution file    = output.dat
--> analytical solution file   = masa.dat


MASA :: Solution has 2 variables.
*-------------------------------------*
A_x is set to: 10.0
k_0 is set to: 1.0
*-------------------------------------*


** Initializing data structures...


** Building one dimensional linear system...
--> Enforcing analytic Dirichlet BCs using MASA (1D)


** Solving linear system...
--> Terminated at iteration: 731
--> The error norm: 0.000000000001
--> The residual norm: 0.000000000004


** Writing solution to output.dat


** The verification mode is launched
** Computing l2 error norm...
--> l2 norm of the error between numerical solution and the analytic solution= 0.016626160860
```

Figure 3: example of the standard output of verification mode

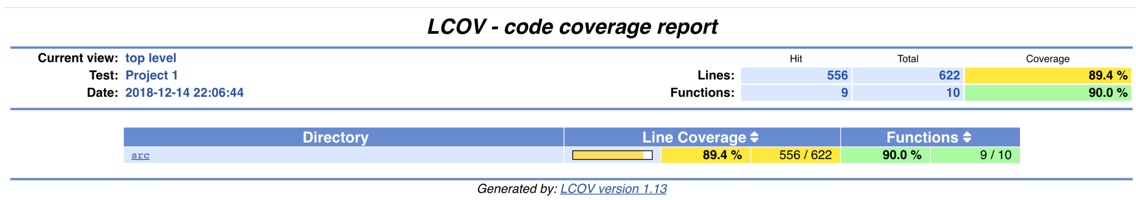
5 Inprovement

6.1 Code Coverage

Figure 4 is the result of the code coverage:

6.2 HDF5

I add HDF5 and improve the regression test.



LCOV - code coverage report

Current view:	top level		Hit	Total	Coverage
Test:	Project 1	Lines:	556	622	89.4 %
Date:	2018-12-14 22:06:44	Functions:	9	10	90.0 %

Directory	Line Coverage		Functions	
src	89.4 %	556 / 622	90.0 %	9 / 10

Generated by: LCOV version 1.13

Figure 4: Report of Code Coverage

6.3 PETSc

I add PETSc and the runtime is reduced for smaller mesh size. You can see the results in figure 12 and figure 13.

6 Results

6.1 Verification Exercise

Figure 5, 6 and 7 are the figures of the functions given by analytical and numerical solutions. You can compare them and see that the numerical solutions are very closed to the analytical solutions. Figure 8, 9 and 10 are the plottings of the resulting error norms from the 2nd and 4th order uniform refinement studies. The slope of 2nd order scheme is approximately -2 and the slope of 4th order scheme is approximately -4. They are closed the expected asymptotic convergence rates. I use Matlab to plot the figures of the results.

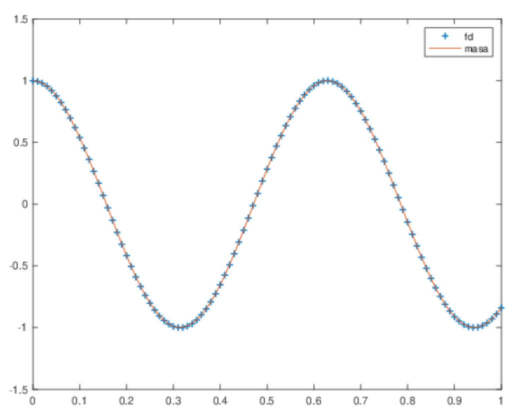


Figure 5: 1D heat equation finite difference solution compared with analytical solution given by MASA

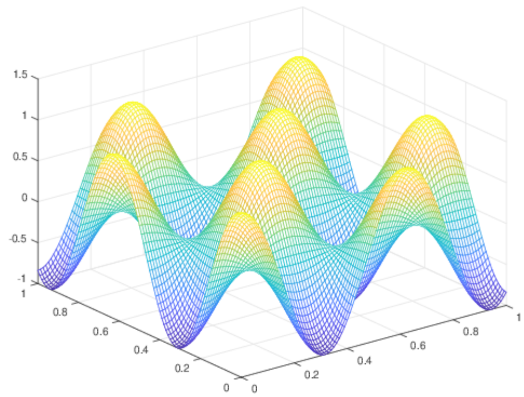


Figure 6: 2D heat equation finite difference solution

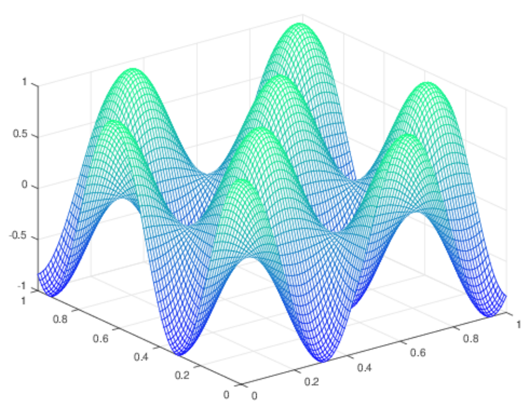


Figure 7: 2D heat equation analytical solution given by MASA

6.2 Runtime Performance

Figure 11 is the figure presenting runtime performance measurements of my application. The total runtime is split into 5 parts: input, initialize, build linear system, solve linear system and output. Figure 12 and 13 are comparison of runtime between Jacobi, Gauss-Seidel and PETSc for 1 or 2 dimension and 2nd order scheme.

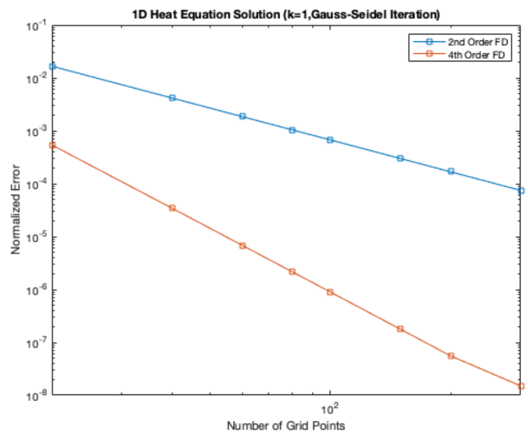


Figure 8: The slope of 2nd order FD is -2.07 and the slope of 4th order FD is -3.98

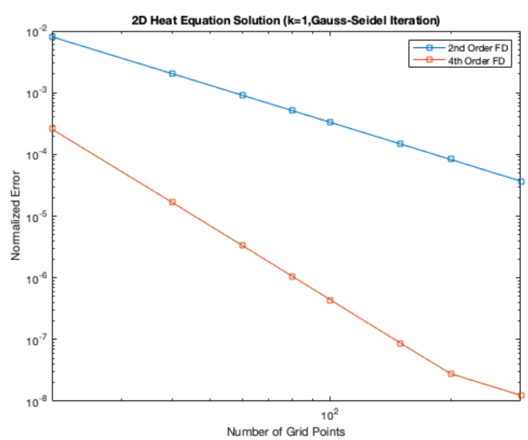


Figure 9: The slope of 2nd order FD is -2.03 and the slope of 4th order FD is -4.05

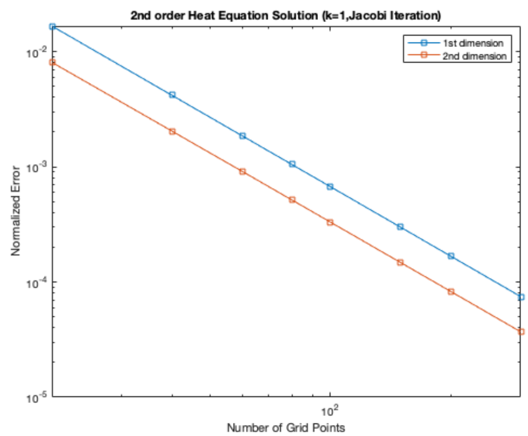


Figure 10: The slope of 1st dimension is -2.04 and the slope of 2nd dimension is -2.03

```
-----------------------------------------------------------------------------------------------
Heat Equation Solver - Performance Timings:                |      Mean      Variance       Count
--> solve_linear_system    : 2.43218e-02 secs ( 63.9544 %) | [2.43218e-02  0.00000e+00          1]
--> output                 : 7.25484e-03 secs ( 19.0767 %) | [7.25484e-03  0.00000e+00          1]
--> input                  : 3.11303e-03 secs (  8.1857 %) | [3.11303e-03  0.00000e+00          1]
--> build_linear_system_1d : 2.58207e-03 secs (  6.7896 %) | [2.58207e-03  0.00000e+00          1]
--> initialize             : 5.29289e-05 secs (  0.1392 %) | [5.29289e-05  0.00000e+00          1]
--> GRVY_Unassigned        : 7.05242e-04 secs (  1.8544 %)

       Total Measured Time = 3.80299e-02 secs (100.0000 %)
-----------------------------------------------------------------------------------------------
```

Figure 11: Runtime Performance Measurements

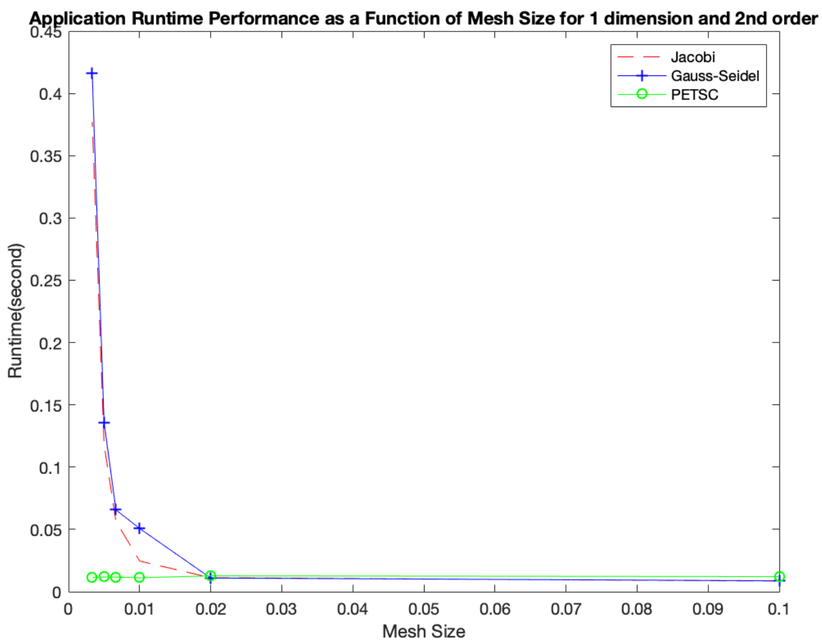


Figure 12: Comparison of Runtime Performance Measurements between different methods

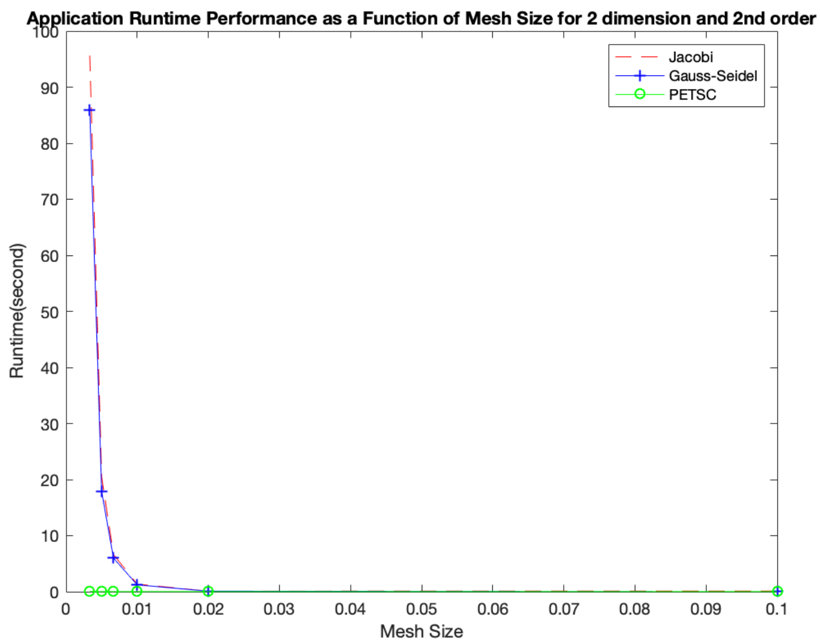


Figure 13: Comparison of Runtime Performance Measurements between different methods