SparseMatrixSolutor v3.3 Numerical Report

Anqi Lv*
PHYS 2200011316

2025.10.22

I Numerical Method

i Process

In this part I will explain how this program works.

The program has PoissonEQ class. This class has grid and boundary condition information inside.

The program firstly sets a Rectangle Reign with rectangle grid. Then it will scan each node to get whether it is inside the effective area. Node inside the area will be marked ”active” while others ”inactive”.

Solution on inactive node is set to 0 now(of course you can set it to any number). Then active nodes will construct the PDEMatrix. Each node has an equation with it. I called this equation as the ”MainEQ” of this node.

Active node will be classified to inner node and boundary node. Inner node’s MainEQ is five-point difference scheme. Boundary node’s MainEQ is just its boundary condition(Because nodes near boundary will also be considered as boundary. This step will make some truncation error. Use polynomial fitting can reduce it. I’m considering to update this method in the next version).

Later when the LHS of MainEQ is constructed, RHS is then constructed. BC has been saved in PoissonEQ class. User only need to provide source function of Poisson equation.

Equation is then solved by Gauss-Sider Iteration. Output is saved as .bin. You can use VisualizeMain.exe to visualize it.

*email:2200011316@stu.pku.edu.cn tel:18315391730

ii Grid

I choose rectangle grid to discretize Poisson Eqation with x-axis interval h_x, y-axis interval h_y.

iii Boundary Condition

I calculate the distance between the node and boundary sigment. If the distance is shorter than toleration, it will be judged as Boundary Point.

For Dirichlet BC node, the MainEQ is:

$$U(i_x, i_y) = g(x(i_x), y(i_y))$$

For Neumann BC and Mixed BC node, the MainEQ is:

$$\alpha U(i_x, i_y) - \beta(n_x \cdot \frac{U_{x>} - U_{x<}}{h_x} + n_y \cdot \frac{U_{y>} - U_{y<}}{h_y}) = g(x(i_x), y(i_y))$$

II Algorithm realization

To get the area I want to grid. I need to define full area and full grid.

Definition 1: Point set $A = \{(i * h_x, j * h_y) | i, j \in \mathbb{Z}, m_{low} \leq i \leq m_{high}, n_{low} \leq j \leq n_{high}\}$ is defined as a *Rectangle Grid* on $\mathbb{R}^2$. A connected subset of $\bar{A} = \{(x, y) | x, y \in \mathbb{R}, m_{low}hx \leq i \leq m_{high}hx, n_{low}hy \leq j \leq n_{high}hy\}$ is called *Fullly Covered* by rectangle grid A.

I want to solve a PDE on an area. It's obvious that I need to set a rectangle area to cover it and make the rectangle grid to fully cover it. And to reduce the difficulty of solving large matrix. I need to define connectivity.

Definition 2: A is a rectangle grid fully cover B. Two point $U_1, U_2 \in A \cap B$ is *Grid Connected*, if there exists a sequence $a_N \subseteq A$, that satisfy $||a_{n+1} - a_n|| = h$ and $a_1 = U_1$, $a_N = U_2$. $A \cap B$ is *Grid Connected*, if any two points in it is grid connected.

Then classify the nodes on $A \cap B$.

Definition 3: On $A \cap B$, point U is called V's *Neighbour* if $||a_{n+1} - a_n|| = h$. It's obviously that a point's neighbours is no more than 4. A point with 4 neighbours is called *Inner Node*, and *Boundary Node* otherwise.

To solve PDE eqation, I need N eqations, where N is the number of nodes. I want to define a bijection between nodes and equations. If this bijection is well defined, I can solve the PDE by solving the system. Namely, I assume this bijection exist and note it as σ, and note $\sigma(U)$ as U's *Main Eqation.*I want to find how to define it.

For Eqation:

$$u_{xx} + u_{yy} = f(x, y)$$

For inner nodes, it is easy to define:

$$\sigma(U_{i,j}) = \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h_x^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h_y^2} - f(x_i, y_j)$$

I want to know what main equation on boundary node is. It's hard to find it on complex Boundary. I can do it on polygonal boundary. I get $A \cap B$ using ray casting algorithm:

Theorem 1: For polygon area, a point is inside the polygon if and only if a ray cast by it intersects the boundary 2n+1 times($n \in \mathbb{N}$).

Define distance between point (x, y) and boundary line $Sigment((x_1, y_1), (x_2, y_2))$.And a obvious theorem will help me put the boundary point on a boundary line:

Definition 4: *Distance* between point (x_0, y_0) and boundary line $Sigment((x_1, y_1), (x_2, y_2))$ is $inf_{(x,y) \in Sigment}(x - x_1)^2 + (y - y_1)^2$

Theorem 2: For every boundary node, there exits a boundary line that the distance between node and boundary line is less than h.

Theorem 2 says that, if I broaden the boundary by h, each boundary point will be coverd by a boundary line.

So now which boundary does a point have is certain.But how to choose boundary condition value is still uncertain.

It's easy that I can broaden this boundary line by directly projecting the point to the boundary line to get value, and if I have some estimation on it, the result can be more accrate. In the following test, I directly use the true value. Shows below:

$$U_{i,j} = f_{estimate}(x_i, y_j) \ \ (x_i, y_j) \in \bar{\Omega}_{broadenDirichletBC}$$

$$\alpha U_{i,j} + \beta \frac{\partial U_{i,j}}{\partial n} = g_{estimate}(x_i, y_j) \ \ (x_i, y_j) \in \bar{\Omega}_{broadenMixedBC}$$

Now you can see it's very easy to find Dirichlet BC.

$$U_{(i,j)} = f(x_i, y_j)$$

But for Neumann and Mixed BC, thing will be a little more complex. So I may firstly show how these problem to be solved.

Define nomal direction derivative by:

$$\alpha U(i_x, i_y) - \beta(n_x \cdot \frac{U_{x>} - U_{x<}}{h_x} + n_y \cdot \frac{U_{y>} - U_{y<}}{h_y}) = g(x(i_x), y(i_y))$$

The formula means I need at least two neighbours, in x and y direction respectively.

Some angle point will produce nodes with only one neighbour. I have no ideas but only let it to equal to its neighbour:

$$U_{(i,j)} = U_{onlyNeighbour}$$

There are two cases in two neighbours case:

1.Two neighbours in the different direction. It is easy to define a derivative. But in case of zero diagonal element(for example normal direction is 45°). I only choose those with two neighbours on the inner normal direction to produce derivative,which is the most common case.

2.Otherwise of case 1 and two neighbours in the opposite direction. The latter one sometimes happen in the "neck" structure which can be solved by reduce hx and hy. But for program running stably, I make it the average of two neighbours.

$$\begin{cases} \sigma(U_{(i,j)}) = \alpha U_{(i,j)} - \beta(n_x \frac{U_{(i,j)} - U_{(i\pm1,j)}}{h_x} + n_y \frac{U_{(i,j)} - U_{(i,j\pm1)}}{h_y}) - g(x_i, y_j) \quad (inner\ normal) & (1) \\ U_{(x,y)} = \frac{U_{neighbour1} + U_{neighbour2}}{2} \qquad (two\ neighbours\ otherwise) & (2) \end{cases}$$

Three neighbours is the same as two neighbour case.

$$\begin{cases} \sigma(U_{(i,j)}) = \alpha U_{(i,j)} - \beta(n_x \frac{U_{(i,j)} - U_{(i\pm1,j)}}{h_x} + n_y \frac{U_{(i,j)} - U_{(i,j\pm1)}}{h_y}) - g(x_i, y_j) \quad (inner\ normal) & (3) \\ U_{(x,y)} = \frac{U_{neighbour1} + U_{neighbour2} + U_{neighbour3}}{3} \qquad (three\ neighbours\ otherwise) & (4) \end{cases}$$

I encounter a problem in solving Neumann BC, that Boundary conflict with each other. It's easy to see in equation perspect:

Definition 5: If $\frac{\partial\sigma(U_(i,j))}{\partial(U_(i',j'))} \neq 0$ (i,j) and (i',j') is called coupled. If there isn't a connected path(a sequence that coupled one by one) between two subset of $A \cap B$, the two sub set is *decoupled*. If a subset's coupled node set is itself, it is *Independent*

Some Independent part will show up in some boundary part(may cause singularity), and some bad poins will happen in the case that a Neumann BC line is connected to Dirichlet BC line(Imagine that a Neumann BC node with nonzero derivative. Its neighbour is a Dirichlet BC node. And these to points is an independent part, get a very high value). So I add an average operation to automatically cope with these bad nodes.

If a Neumann or Mixed BC node's neighbours are all Boundary nodes its main eqation will be the average of all its neighbours.

$$U_{(i,j)} = average(neighbours)$$

So far, I have got all main equations.

III Numerical Test

I try function:

$$f(x, y) = sin(\pi x)cos(2\pi y)$$

as a test function.

Use grid hx=0.02, hy=0.02.

Try Dirichlet BC and the solution is right:

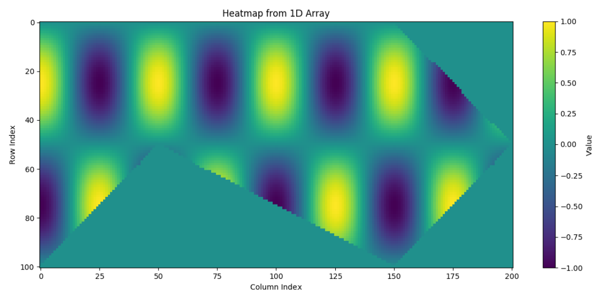
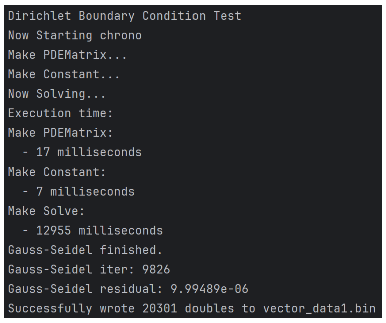


Figure 1: Dirichlet BC Solution

It just takes a little time:



```
Dirichlet Boundary Condition Test
Now Starting chrono
Make PDEMatrix...
Make Constant...
Now Solving...
Execution time:
Make PDEMatrix:
  - 17 milliseconds
Make Constant:
  - 7 milliseconds
Make Solve:
  - 12955 milliseconds
Gauss-Seidel finished.
Gauss-Seidel iter: 9826
Gauss-Seidel residual: 9.99489e-06
Successfully wrote 20301 doubles to vector_data1.bin
```

Figure 2: Dirichlet BC Print

Change h, and will get how errors change:

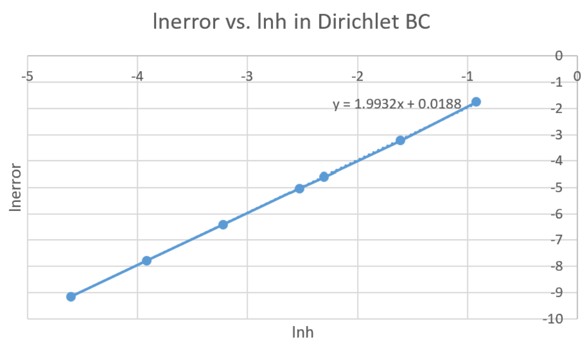


Figure 3: Dirichlet error test

Error is power 2 related to h.
Try Neumann BC with h=0.025 and the solution is right:

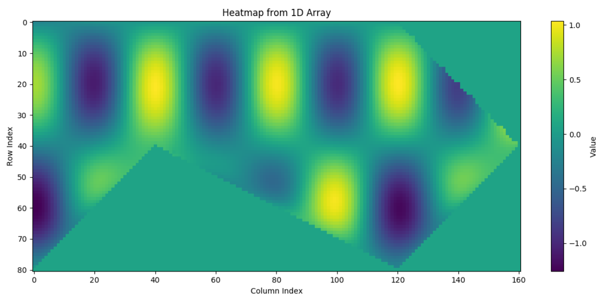
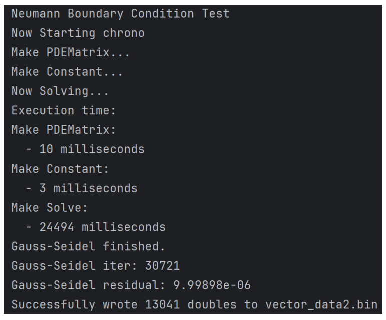


Figure 4: Neumann BC Solution

It just takes a little time:



```
Neumann Boundary Condition Test
Now Starting chrono
Make PDEMatrix...
Make Constant...
Now Solving...
Execution time:
Make PDEMatrix:
  - 10 milliseconds
Make Constant:
  - 3 milliseconds
Make Solve:
  - 24494 milliseconds
Gauss-Seidel finished.
Gauss-Seidel iter: 30721
Gauss-Seidel residual: 9.99898e-06
Successfully wrote 13041 doubles to vector_data2.bin
```

Figure 5: Neumann BC Test

It is worth noting that eqation with all Neumann BC has one degree of freedom that if u(x,y) is the solution, so do u(x,y)+c,where c is a constant independent of x,y.So I normalize the solution by:

$$< u(x,y) >_{x,y} = 0$$

To test the error change h:

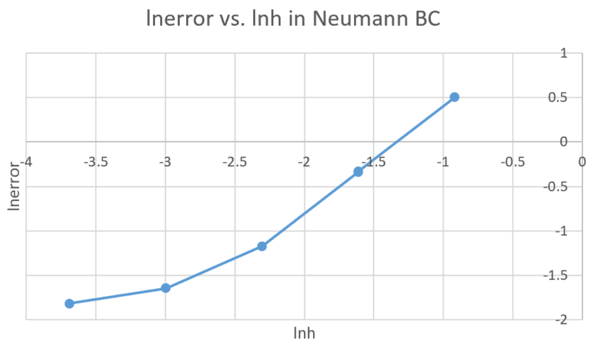


Figure 6: Neumann error test

Error is about 1 power related to h when h is big enough. It converges slower and slower when H goes smaller.

The reason of 1 power may goes as boundary error. Plot the heat map of $u(x_i, y_j) - U_{i,j}$:

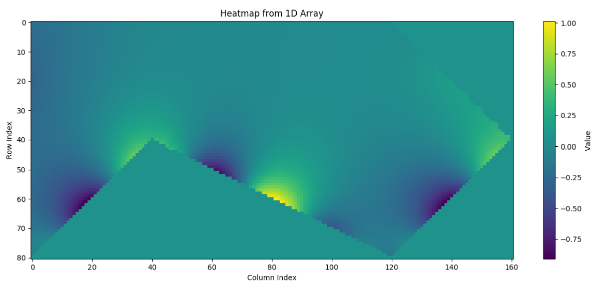


Figure 7: Neumann residual

You can see all error comes from boundary, which makes it 1 power lower than Dirichlet BC Error.