# SparseMatrixSolutor v2.3 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 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:

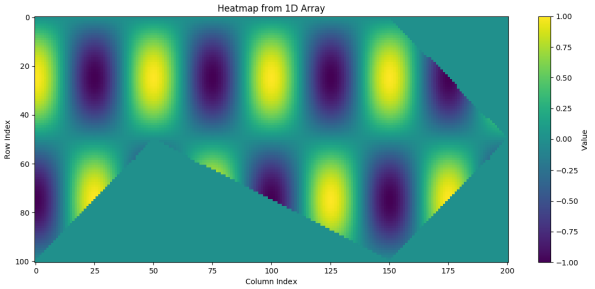


Figure 1: Dirichlet BC Solution

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

Try Dirichlet BC and the solution is right:

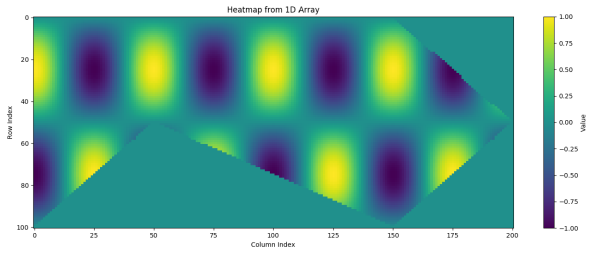


Figure 3: Neumann BC Solution

It just takes a little time:

```
Neumann Boundary Condition Test
Now Starting chrono
Make PDEMatrix...
Make Constant...
Now Solving...
Execution time:
Make PDEMatrix:
  - 18 milliseconds
Make Constant:
  - 6 milliseconds
Make Solve:
  - 23173 milliseconds
Gauss-Seidel finished.
Gauss-Seidel iter: 9826
Gauss-Seidel residual: 9.99489e-06
Successfully wrote 20301 doubles to vector_data2.bin
```

Figure 4: Neumann BC Test