Final Project 2 - Solving Poisson Equation

Weicheng Ye Zeyang Ye Tianshu Bao Yijiao Cao

Department of Applied Mathematics and Statistics

Stony Brook University

Contents

1 Introduction **3**

2 A is NOT Positive Definite Matrix **4**

3 LU Decomposition with Pivoting **7**

4 Cauchy Error and Order of Convergence **7**

5 Block Tridiagonal Form and Double-Sweep Algorithm **8**

6 CPU Time Analysis **11**

7 Creative Techniques **11**

1 Introduction

Our problem is to solve 2-dimensional Poisson equation using finite difference method. The problem is

$$\Delta u = f(x,y)$$

where $f(x,y) = \cos\pi\sqrt{x^2+y^2}$ if $\sqrt{x^2+y^2} \leq \frac{1}{2}$ and $f(x,y) = 0$ o.w.

The domain is

$$\Omega = \{(x,y) \in (-1,1) \times (-1,1)\}$$

with the boundary condition

$$\mu|_{\partial}\Omega = 0$$

Since

$$\frac{\partial^2 u_{i,j}}{\partial x^2} = \frac{\frac{u_{i+1,j}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i-1,j}}{h}}{h} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

and

$$\frac{\partial^2 u_{i,j}}{\partial y^2} = \frac{\frac{u_{i,j+1}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i,j-1}}{h}}{h} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}$$

Once we substitute such relations of equations into the Poisson equation, we get

$$\Delta u_{i,j} = \frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1}, u_{i,j-1}) - 4u_{i,j} = f_{i,j}$$

Here we define n as the number of meshes. According to this relation, if we convert such problem into matrix, Ax = b, with different computational mesh of the matrix A $\in R^{n^2\times n^2}$, then the matrix has the form

$$\begin{pmatrix} B & I & & & & & \\ I & B & I & & & & \\ & I & B & I & & & \\ & & & & \ddots & & \\ & & & & I & B & I \\ & & & & & I & B \end{pmatrix}$$

where B $\in R^{n\times n}$ is

$$\begin{pmatrix} -4 & 1 & & & & & \\ 1 & -4 & 1 & & & & \\ & 1 & -4 & 1 & & & \\ & & & \ddots & & & \\ & & & & 1 & -4 & 1 \\ & & & & & 1 & -4 \end{pmatrix}$$

and here $I \in R^{n\times n}$ is identity matrix.

In addition, vector x has the form

$$\begin{pmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{nn} \end{pmatrix}$$

and vector b has the form

$$\begin{pmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{1n} \\ \vdots \\ f_{nn} \end{pmatrix}$$

since all the boundary points are 0.

In Figure 1, we use MATLAB to plot the 80×80 mesh grid for the solution of Poisson equation. In Figure 2, we use 160×160 mesh grid to plot the inhomogeneous part in the Poisson equation, $f(x, y)$. In Figure 3 and 4, we plot the 20×20, 40×40, 80×80, 160×160 mesh grid for the solution of Poisson equation.

2 A is NOT Positive Definite Matrix

For checking if the matrix is positive definite, we wrote the Cholesky decomposition (inner-product form) in C since the Cholesky's Algorithm will set error flag if matrix

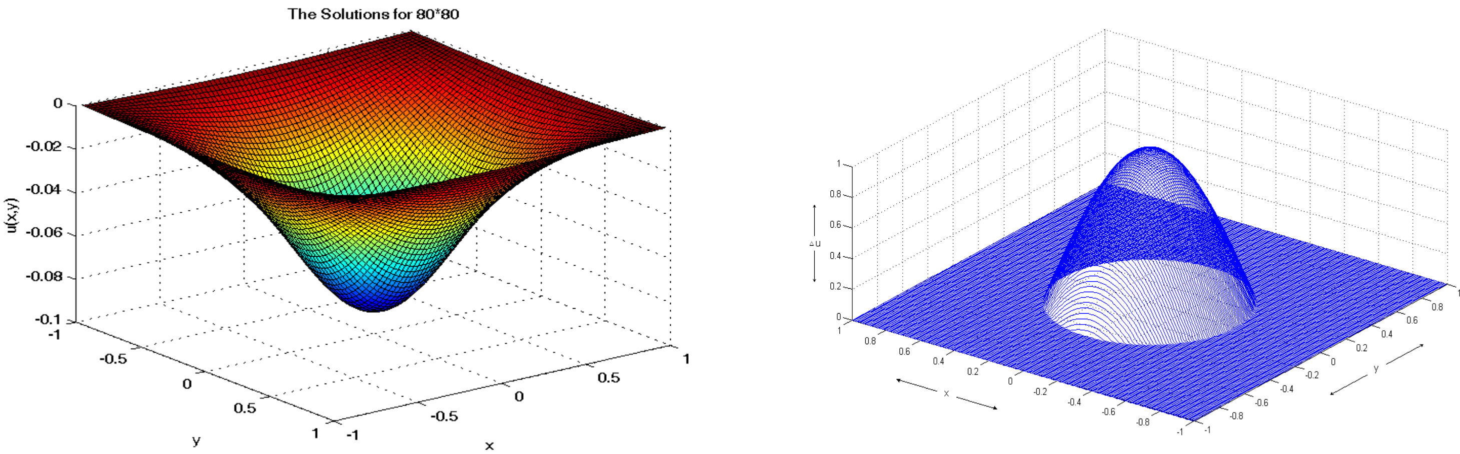

Figure 1: (Left) 80×80 mesh grid for Poisson equation solution. (Right) 160×160.mesh grid for Poisson equation inhomogeneous part f(x,y).

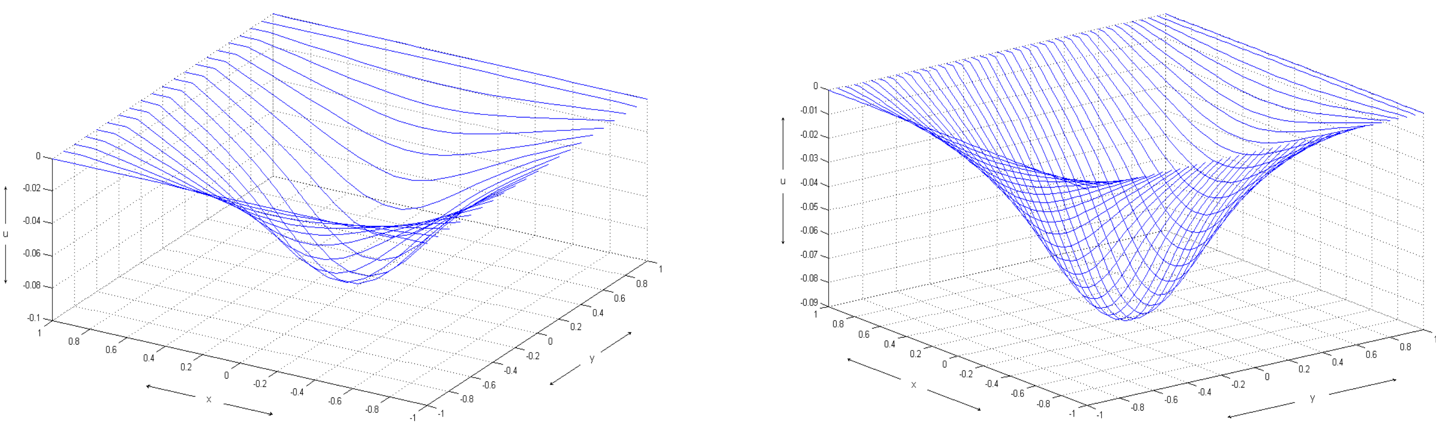

Figure 2: (Left) 20×20.mesh grid for Poisson equation solution. (Right) 40×40.mesh grid for Poisson equation solution.

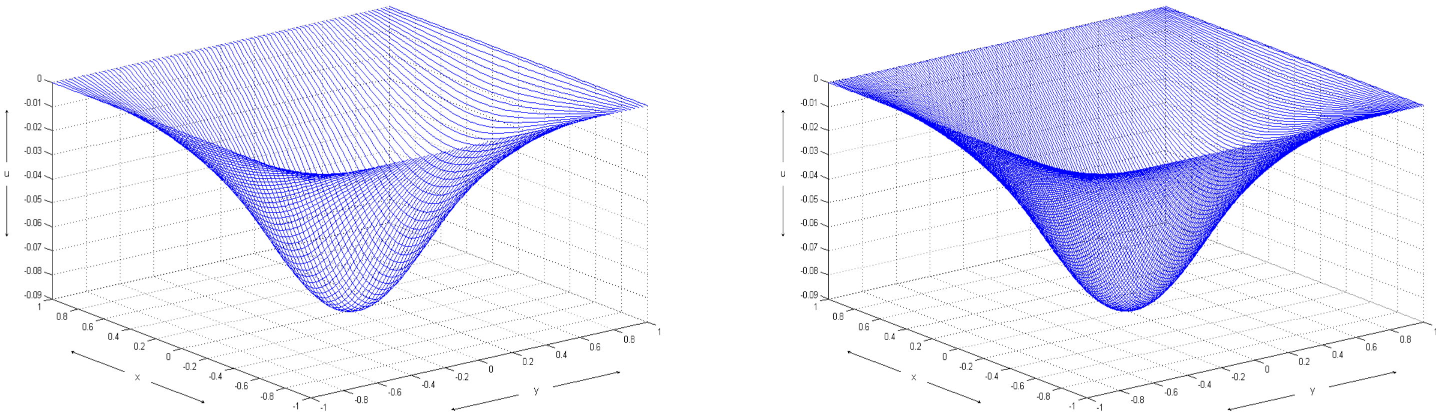

Figure 3: (Left) 80×80.mesh grid for Poisson equation solution. (Right) 160×160.mesh grid for Poisson equation solution.

A is not positive definite. Here is the pseudocode for the Algorithm:

for i = 1, ..., n^2
    for k = 1, ..., i-1 (not executed when i = 1)
        $A_{ii} = A_{ii} - A_{ki}^2$
    end
    if $A_{ii} \leq 0$, set error falg that A is NOT Positive Definite, exit
    $A_{ii} = \sqrt{A_{ii}}$ (this is r_{ii})
    for j = i+1,..., n^2 (not executed when $i = n^2$)
        for i = 1,...,i-1 (not executed when i = 1)
            $A_{ij} = A_{ij} - A_{ki}A_{kj}$
        end
        $A_{ij} = A_{ij}/A_{ii}$
    end
end

When we run such program for different mesh sizes, the Cholesky decomposition always exits the execution so that we know the matrix is NOT positive definite.

3 LU Decomposition with Pivoting

When we program the LU decomposition with pivoting, we first record the interchanging swap in each round into an array. For example, the array $P = \ldots[6][3]$ means when we do the first LU decomposition with pivoting, we interchange the matrix row 3 and row 1 since $|A_{31}|$ is the maximum in the first column. In addition, the row interchanging for the second round is row 2 and row 6 in the new matrix after the updating of the matrix.Once we have the whole array list $P = P_{n^2-1}\ldots P_2P_1$, we can do $PAx = P$b so that the LU decomposition without pivoting can be applied to $PAx = P$b directly. The LU decomposition will then follow the same procedure as shown in the textbook. Note that both L and U will have semi band n in this case. Recall that the semi-band n means that $\forall\ k>j+n$, $l_{kj} = 0$ and $\forall\ k<j-n$, $u_{kj} = 0$.

When we execute it in program, we create the Augmented matrix $Ax = b$ and apply the pivoting for both matrix part and pivoting part directly. This is the form of augmented matrix (A $\in R^{n^2\times n^2}$):

$$\left(\begin{array}{cccc|c} A_{11}^{(1)} & A_{12}^{(1)} & A_{13}^{(1)} & \ldots & f_{11} \\ A_{21}^{(1)} & A_{22}^{(1)} & A_{23}^{(1)} & \ldots & f_{12} \\ & \ldots & & \ldots & \ldots \\ A_{n1}^{(n)} & A_{n2}^{(n)} & A_{n3}^{(n)} & \ldots & f_{nn} \end{array}\right)$$

4 Cauchy Error and Order of Convergence

First, Cauchy error has the formula

$$e_M = ||u_M - u_{2M}||_2$$

In this case, the vertex in $n\times n$ mesh will also in the $2n\times 2n$ mesh, and it satiesfies the formula (*)

$$u_{i,j}|_n = u_{2i-1,2j-1}|_{2n}$$

which means the ij term in $n\times n$ mesh's x vector is the $2i$-1,$2j$-1 term in $n^2\times n^2$ mesh's x vector.Following the same procedure, one can get the relations between $n\times n$ mesh and $3n\times 3n$, $n\times n$ and $4n\times 4n$ etc.

When we do the Cauchy error test for 20×20 mesh, the size of u_{20} is 400. According to the formula (*), we know that we can select $\frac{1}{4}\times 40^2$ terms in u_{40} to find the Cauchy error of u_{20} and u_{40}. The same formula holds for the Cauchy error computing.

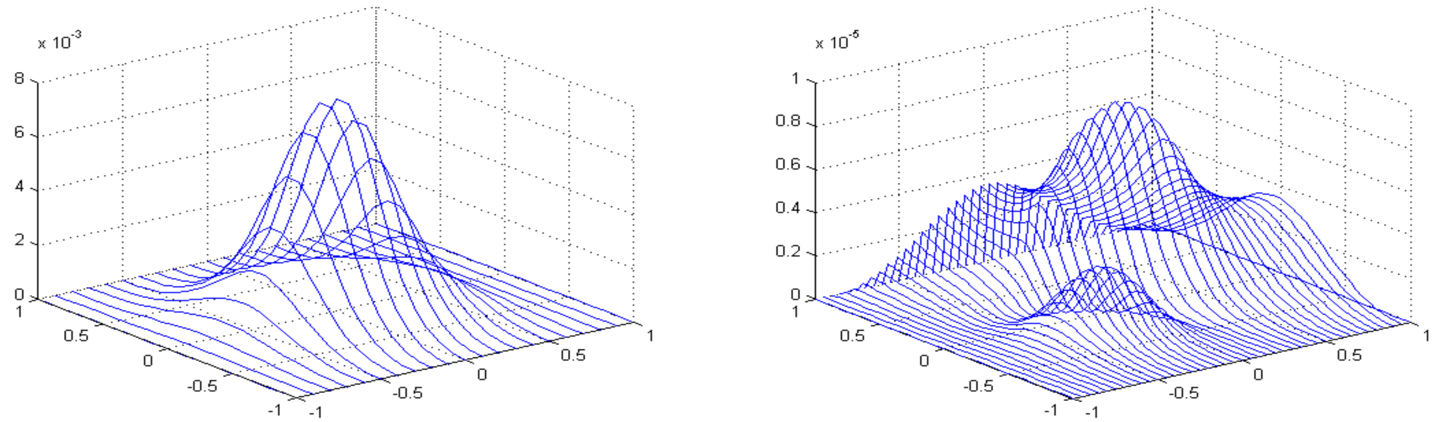

Figure 4: (Left) order of convergence for 20-40. (Right) order of convergence for 40-80.

5 Block Tridiagonal Form and Double-Sweep Algorithm

Recall that matrix A has the form

$$\begin{pmatrix} B & I & & & & \\ I & B & I & & & \\ & I & B & I & & \\ & & & \ddots & & \\ & & & I & B & I \\ & & & & I & B \end{pmatrix}$$

Therefore, A has the block-tridiagonal form.

We now derive the block-tridiagonal double-sweep algorithm. We split x into

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

where each x_i is an $n \times 1$ column vector.

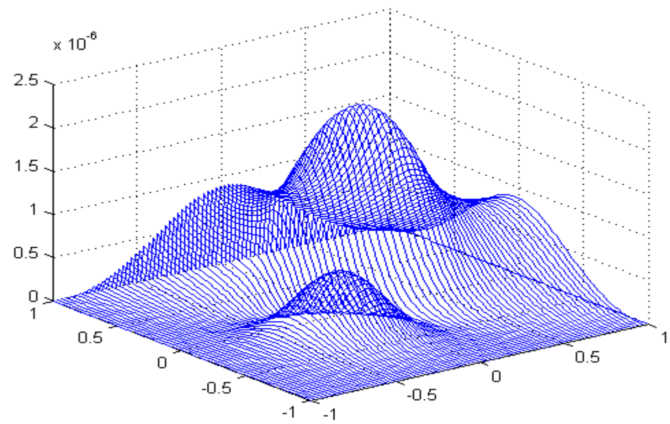

Figure 5: (Left) order of convergence for 80-160.

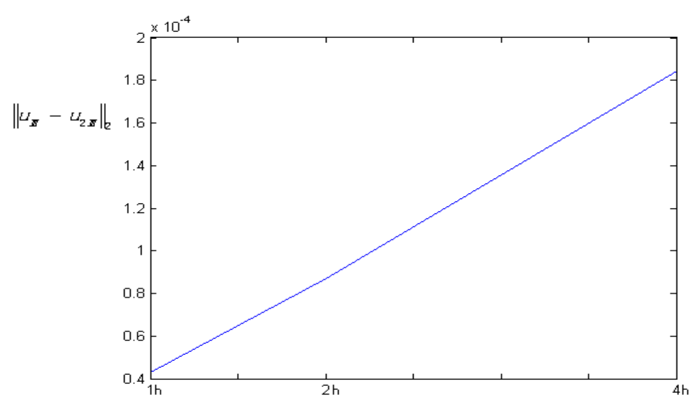

Figure 6: Order of Convergence..

For the last row of the equation, it has

$$Ix_{n-1} + Bx_n = f_n$$

and we know (define)

$$A_{n-1}x_{n-1} + C_{n-1} = x_n$$

Therefore,

$$Ix_{n-1} + B(A_{n-1}x_{n-1} + C_{n-1}) = f_n$$
$$(I + BA_{n-1})x_{n-1} + BC_{n-1} = f_n$$
$$I + BA_{n-1} = 0$$
$$A_{n-1} = -B^{-1}I = -B^{-1} \quad (1)$$
$$C_{n-1} = B^{-1}f_n \quad (2)$$

For the middle process, it has

$$Ix_{i-1} + Bx_i + Ix_{i+1} = f_i$$

$$A_{i-1}x_{i-1} + C_{i-1} = x_i$$
$$A_ix_i + C_i = x_{i+1}$$
$$A_{i-1}^{-1}(x_i - C_{i-1}) + Bx_i + A_ix_i + C_i = f_i$$
$$(A_{i-1}^{-1} + B + A_i)x_i + (C_i - C_{i-1}) = f_i$$
$$A_{i-1}^{-1} + B + A_i = 0$$
$$A_{i-1}^{-1}C_{i-1} + C_i = f_i$$

Therefore, the solution is

$$C_{i-1} = -(A_i + B)^{-1}(C_i - f_i) \quad (3)$$
$$A_{i-1} = -(A_i + B)^{-1} \quad (4)$$

We eventually work on the initial case:

$$Bx_1 + Ix_2 = f_1$$

$$A_1x_1 + C_1 = x_2$$

Therefore,

$$Bx_1 + A_1x_1 + C_1 = f_1$$
$$(B + A_1)x_1 + C_1 = f_1$$
$$x_1 = (B + A_1)^{-1}(f_1 - C_1) \quad (5)$$

Therefore, we can use (1) and (2) to find the initial A_n and C_n first, and then use induction relation to solve for all the A_i and C_i, $\forall\ i \in$ (1,2,...,n), using (3) and (4). Finally, we can use (5) to get the initial case of x_1 and then use the solved A_i, C_i to get all x_i. The vector x is thus solved.

6 CPU Time Analysis

Here is the result for LU Decomposition:
Dimension: 20×20
Time: 0.10 sec

Dimension: 40×40
Time: 7.62 sec

Dimension: 80×80
Time:534.23 sec

Dimension: 160×160
Time: ...sec

Here is the result for Double-sweep Algorithm:
Dimension: 20×20
Time: 0.000938 sec

Dimension: 40×40
Time: 0.012500 sec

Dimension: 80×80
Time: 0.170938 sec

Dimension: 160×160
Time: 2.492812 sec

7 Creative Techniques

Since the matrix A has the form

$$\begin{pmatrix} B & I & & & & & \\ I & B & I & & & & \\ & I & B & I & & & \\ & & & & \ddots & & \\ & & & & I & B & I \\ & & & & & I & B \end{pmatrix}$$

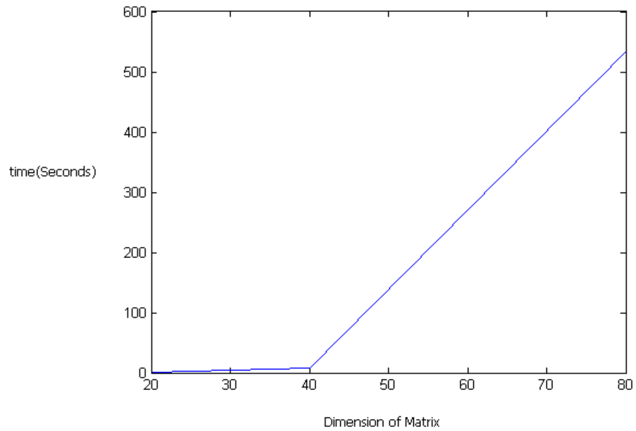

Figure 7: LU decomposition speed against the dimension

We do the block LU decomposition here for matrix A. Here we multiply the first row of block matrices by B^{-1} and then use the second row to subtract the first row. After this step, the matrix will be

$$\begin{pmatrix} B & I & & & & & \\ I-I & B-B^{-1} & I & & & & \\ & I & B & I & & & \\ & & & & \ddots & & \\ & & & & I & B & I \\ & & & & & I & B \end{pmatrix}$$

Then we multiply $(B-B^{-1})^{-1}$ for the second row and use the third line to subtract that. We get the matrix

$$\begin{pmatrix} B & I & & & & & \\ & B-B^{-1} & I & & & & \\ & & B-(B-B^{-1})^{-1} & I & & & \\ & & & & \ddots & & \\ & & & & I & B & I \\ & & & & & I & B \end{pmatrix}$$

Once we apply this procedure to the whole main diagonal we get the matrix

$$\begin{pmatrix} B & I & & & & \\ & B-B^{-1} & I & & & \\ & & B-(B-B^{-1})^{-1} & I & & \\ & & & B-(B-(B-B^{-1})^{-1})^{-1} & I & \\ & & & & \ddots & \end{pmatrix}$$

Another method is to multiply the second row by B first, and then use the new second line to subtract the first line. The matrix is

$$\begin{pmatrix} B & I & & & & \\ B-B & B^2-I & B & & & \\ & I & B & I & & \\ & & & \ddots & & \\ & & & I & B & I \\ & & & & I & B \end{pmatrix}$$

Then for the second round, we multiply (B^2 - I) to the third row and then subtract the second row.

$$\begin{pmatrix} B & I & & & & \\ & B^2-I & B & & & \\ & & B(B^2-I)-B & B^2-I & & \\ & & & \ddots & & \\ & & & I & B & I \\ & & & & I & B \end{pmatrix}$$

Once we follow this procedure

$$\begin{pmatrix} B & I & & \\ & B^2-I & B & \\ & & B(B^2-I)-B & B^2-I \\ & & & & \ddots \end{pmatrix}$$

The advatage of this method is that we don't need to compute the inverse matrix at all and can do direct matrix multiplication.