NUMERICAL SCHEME AND HIERARCHICALLY SOLVER FOR ANISOTROPIC POISSON EQUATION WITH FIRST BOUNDARY CONDITION ON SQUARE DOMAIN

YIPING LU

luyiping9712@pku.edu.cn
School of Mathematic Science, Peking University.
No.5 Yiheyuan Road Haidian District,Beijing,P.R.China 100871.

This lecture note is the report of the project of **Numerical Algebra(2017-2018Autumn)** instructed by Prof. Jun Hu at Peiking Univ.

1. Numercial Schemes For Heat Equation With First Boundary Condition

In this report we consider the following anisotropic problem:

$$\Delta_\epsilon u = u_{xx} + \epsilon u_{yy} = f, u|_{\partial D\times(0,1]} = 0$$

Here $D = (0,1)\times(0,1)$ is a square domain on the plane.

If we suggest the solution is $u(x,y) = \sin(\pi x)\sin(\pi y)$, so that$f(x,y) = -(1+\epsilon)\pi^2\sin(\pi x)\sin(\pi y)$

1.1. Numerical Methods.

In this report we utilize the five point finite difference scheme, that is to say Δ_ϵ is approximated by

$$\frac{U_{j+1,k} - 2U_{j,k} + U_{j-1,k}}{h_x^2} + \epsilon\frac{U_{j,k+1} - 2U_{j,k} + U_{j,k-1}}{h_y^2}$$

The local truncation error can be easily proved that has the formula $Tu(x,y) = -\frac{1}{12}(u_{xxxx}(x,y)h_x^2 + \epsilon u_{yyyy}(x,y)h_y^2) + O(h_x^4 + h_y^4)$

Consider the maximal principle which can be formatted as following:

Theorem(Maximal Principle) For the finite difference scheme L_h defined as

$$L_h U_j = \sum_{i\in J\setminus\{j\}} c_{i,j}U_i - c_j U_j$$

satisfies

- $J_D \neq \emptyset$ and $J = J_\Omega \cup J_D$ is connected
- for every $j \in J_\Omega$ we have $c_j, c_{i,j} > 0$ and $c_j \geq \sum_{i\in D_{L_h}(j)} c_{i,j}$

Then

$$\max_{i\in J_\Omega} U_i \le \max\{\max_{i\in J_D} U_i, 0\}$$

It's easy to find that the numerical scheme satisfies the maximal principle, that is to say our scheme is stable under norm $||\cdot||_\infty$.

2. Hierarchically Numerical Linear Algebra Method For The Problem

This section is based on Volker John's Lecture nots on multigrid and Jinchao Xu's review paper 'Iterative methods: by space decomposition and subspace correction'.

2.1. **Detialed Investigation of Classical Iterative Schemes.**

2.1.1. *General Aspects of Classical Iterative Scheme.* To solve the linear system $Au = f$ we can give a general approach: let $A = M - N$, the iterative scheme can be formula as

$$\begin{aligned} u^* &= M^{-1}Nu + M^{-1}f := Su + M^{-1}f \\ u^{(m+1)} &= \omega u^* + (1-\omega)u^{(m)} \end{aligned}$$

such that $u^{(m+1)} = (\omega S + (1-\omega)I)u^{(m)} + \omega M^{-1}f$ and we have residual equation $Se^{(m)} = e^{(m+1)}$

This scheme can conclude Damped Jacobi method and the SOR method.

2.1.2. *Converge Analysis.* First apply Discrete Fourier Method to analysis the scheme. For a give function b on $[0,1]$, we can expanded in the form

$$b(x) = \sum_{k=1}^{\infty} b_k \sin(k\pi x)$$

Here we can analyze $u^{(0)} = (u_1^{(0)}, \cdots, u_{N-1}^{(0)})^T, u_j^{(0)} = \sin(\frac{jk\pi}{N})(j,k = 1, \cdots, N-1)$ This discrete Fourier modes are also the **eigenvectors** of the matrix A.

- For $1 \le k < N/2$ are called the low frequency or smooth modes.
- For $N/2 \le k \le N-1$ are called high frequency or oscillating modes.

There are several important observation

- On a fixed grid, there is a good damping of the high frequency errors whereas there is almost no damping of the low frequency errors.
- For a fixed wave number, the error is reduced on a coarser grid better than on a finer grid.
- The logarithm of the error decays linearly.

damped Jacobi methods

For damped Jacobi methods the iteration matrix is $S_{jac,\omega} = I - \omega D^{-1}A = I - \frac{\omega h}{2}A$, it has eigenvalue

$$\lambda_k(S_{jac,\omega}) = 1 - \frac{\omega h}{2}\lambda_k(A) = 1 - 2\omega \sin^2(\frac{k\pi h}{2})$$

It is easy to see that **damped Jacobi method converges fastest for** $\omega = 1$ which only need to solve a min-max problem and the error has the form $e^{(n)} = \sum_{k=1}^{N-1} c_k \lambda_k(S_{jac,\omega}) w_k$

SOR methods

The first thing we need to calculate is the eigenvalue of S_{GS}:

$$\lambda_k(S_{GS}) = \cos^2(\frac{k\pi}{N})$$

Proof. Inserting the decomposition of S_{GS} gives

$$-(D+L)^{-1}Uw_k = \lambda_k(S_{GS})w_k \Leftrightarrow \lambda_k(S_{GS})(D+L)w_k = -Uw_k$$

Considering the model problem and inserting the representation of the k-th eigenvector

$$\lambda_k(S_{GS})\left[2\lambda_k(S_{GS})^{1/2}\sin(\frac{jk\pi}{N}) - \sin(\frac{(j-1)k\pi}{N})\right] = (\lambda_k(S_{GS}))^{(j+1)/2}\sin(\frac{(j+1)k\pi}{N})$$

if we let

$$\lambda_k(S_{GS}) = \cos^2(\frac{k\pi}{N})$$

It becomes a well-known relation

$$2\sin(\frac{\alpha+\beta}{2})\cos(\frac{\alpha-\beta}{2}) = \sin\alpha + \sin\beta$$

with $\alpha = (j+1)k\pi/N, \beta = (j-1)k\pi/N$ □

With the calculated eigenvalue the converge analysis is so naive that anyone could do it.

2.2. **Multigrid Methods.** Above all, I want to say the multigrid methods may not be the fastest method in solving the implicit scheme of $u_t = \Delta u$ but is suitable for Laplace equation $\Delta u = f$

2.2.1. *Grid Transfer.* We give some properties of the restriction and interpolation operator. I_{2h}^h, I_h^{2h}

- I_{2h}^h is full rank and the trivial kernel
- $I_h^{2h} = 2(I_h^{2h})^T$
- Dual Operator:$\left\langle I_{2h}^h v^{2h}, r^h\right\rangle_{V^h,(V^h)^*} = \left\langle v^2 h, I_h^{2h} r^h\right\rangle_{V^{2h},(V^{2h})^*}$
- For the 1D model problem we have $A^{2h} = I_h^{2h} A^h I_{2h}^h$(Galerkin Projection)

2.2.2. *Two Level Methods.* The algorithm is give as

- $A^{2h}u^{2h} = f^{2h}$ compute an approximation v^{2h} and compute the residual $r^{2h} = f^{2h} - A^{2h}v^{2h}$
- Solve the coarse grid equation $A^h e^h = I_{2h}^h(r^{2h})$
- $v^h = v^h + I_h^{2h}(e^h)$

Iteration Matrix. Let S_{sm} be the iteration matrix of the smoother. The calculation can be seen at **Multilevel Method** in the next section. The iteration matrix can be given as

$$S_{2lev} = (I - I_h^{2h}(A^h)^{-1}I_{2h}^h A^h)S_{sm}$$

Converge Analysis. Here we give some definition used in the converge analysis

- **Smoothing property**

$$|||A^h S_{sm}||| \leq Ch^{-\alpha}$$

- **Approxiamation property**

$$|||(A^{2h})^{-1} - I_h^{2h}(A^h)^{-2}I_{2h}^h||| \leq C_a h^{\alpha}$$

Following the above two property we can give a converge speed to the algorithm as $|||S_{2lev}||| \leq CC_a$.For every specific algorithm used in the multi-grid framework, you should analyze the above two properties.

2.2.3. *Multigrid.* The W-multigrid can be analyzed by the processing before maybe it can be the appendix of my report for Numerical Algebra. The V-multigrid can be analyzed as a multi-level subspace correction algorithm.

2.3. **Subspace Correction.** Solving the linear equation can be seen as the following three steps:

- $r^{old} = f - Au^{old}$
- $\hat{e} = Br^{old}$ with $B \approx A^{-1}$
- $u^{new} = u^{old} + \hat{e}$

The choice of B is the core of this type of alogrithm. The point is to choice B by solving appropriate subspace problems. The subspaces are provided by a decomposition of V: $V = \sum_{i=1}^{J} V_i$. Here V_i are the subspaces of V. Assume $A_i : V_i \to V_i$ is restriction operator of A on V_i, then we can let $B = R_i \approx A_i^{-1}$ **Multigrid and domain decomposition methods can be viewed under this perspective.** In this section we only consider the linear iteration methods like $u^{k+1} = u^k + B(f - Au^k)$.

Here B is a approximate of the inverse matrix A^{-1} and the sufficient condition for the convergence of the scheme is

$$||I - BA||_A < 1$$

which can be seen in the appendix of the ppt for Iteration Methods.

2.3.1. *Subspace correction and subspace equations.* For subspace decomposition$V = \sum_{i=1}^{J} V_i$, for each i we define $Q_i, P_i : V \to V_i$ and $A_i : V_i \to V_i$ by

$$(Q_i u, v_i) = (u, v_i), (P_i u, v_i)_A = (u, v_i)_A, (A_i u_i, v_i) = (A u_i, v_i)$$

Here P_i, Q_i are both orthogonal projections and A_i is the restriction of A on V_i and is SPD. It follows the definition that $A_i u_i = f_i$ with $u_i = P_i u, f_i = Q_i f$ At the same time we use R_i to represent an approximate inverse of A_i in certain sense. Thus an approximate solution is given by $\hat{u}_i = R_i f_i$

Basic Idea: Consider the residual equation $Ae = r^{old}$ Instead of $u = u^{old} + e$ we solve the restricted equation to each subspace $A_i e_i = Q_i r^{old}$, while using the subspace solver R_i described earlier equally the process can be written as $\hat{e}_i = R_i Q_i r^{old}$

PSC:Parallel Subspace Correction. Similar to Jacobi Methods.(When $V = \sum span(\{e_i\})$, PSC becomes Jacobi)

An update of the approximation of u is obtained by

$$u^{new} = u^{old} + \sum_{i=1}^{J} \hat{e}_i$$

which can be equally written as

$$u^{new} = u^{old} + B(f - Au^{old})$$

where $B = \sum_{i=1}^{J} R_i Q_i$

Lemma. The operator B is SPD.

Proof.

$$(Bv, v) = \sum_{i=1}^{J} (R_i Q_i v, Q_i v) \geq 0$$

And the symmetry of B follows from the symmetry of R_i □

As a simple corollary, B can be used as a preconditioner liker CG methods. .(When $V = \sum span(\{e_i\})$, B becomes the simplest preconditioner $diag(a_{11}^{-1}, \cdots, a_{nn}^{-1})$

SSC:Successive Subspace Correction. Similar to Gauss-Seidel Methods.(When $V = \sum span(\{e_i\})$, SSC becomes G-S) This method is used as

$$\begin{aligned} v^1 &= v^0 + R_1 Q_1 (f - Av^0) \\ v^2 &= v^1 + R_2 Q_2 (f - Av^1) \\ &\cdots \end{aligned}$$

Formerly the algorithm can be written as $u^{(k+i)/J} = u^{(k+i-1)/J} + R_i Q_i (f - Au^{(k+i-1)/J})$

Let $T_i = R_i Q_i A$ Then we have

$$u - u^{(k+i)/J} = (I - T_i)(u - u^{(k+i-1)/J})$$

A successive application of this identity yields

$$u - u^{k+1} = E_J(u - u^k)$$

where $E_J = (I - T_J)(I - T_{J-1}) \cdots (I - T_1)$

Like SOR method we can also have an algorithm as$u^{(k+i)/J} = u^{(k+i-1)/J} + \omega R_i Q_i (f - Au^{(k+i-1)/J})$

Multilevel Methods. Multilevel algorithms are based on a nested sequence of subspaces

$$M_1 \subset M_2 \subset \cdots \subset M_J = V$$

Algorithm.

- Correction:$v^1 = \hat{B}_{k-1}\hat{Q}_{k-1}g$
- Smoothing:$\hat{B}_k g = v^1 + \hat{R}_k(g - \hat{A}_k v^1)$

Next we want to show that **the multilevel method is equivalent to the SSC algorithm.**

Suppose $M_k = \sum_{i=1}^k V_i$ It is easy to show that the two algorithm is equivalent.

2.3.2. *Converge Theory.*

- For PSC we need to estimate the condition number of $T = BA = \sum_{i=1}^J T_i$.
- For SSC we need to estimate $||E_J||_A < 1$

We define two parameters K_0, K_1 at the beginning of the section.

- For any $v = \sum_{i=1}^J v_i \in V$ we have $\sum_{i=1}^J (R_i^{-1} v_i, v_i) \leq K_0 (Av, v)$
- For any u_i, v_i we have

$$\sum_{\{1,2,\cdots,J\}^2} (T_i u_i, T_j v_j) \leq K_1 (\sum_{i=1}^J (T_i v_i, v_i)_A)^{1/2} (\sum_{j=1}^J (T_j v_j, v_j)_A)^{1/2}$$

PSC.

Theorem. Assume that B is the SSC preconditioner then

$$\kappa(BA) \leq K_0 K_1$$

Proof. Follow directly from the definition of K_1 that

$$||Tv||_A^2 = \sum_{i,j=1}^J (T_i v, T_j v)_A \leq K_1 (Tv, v)_A \leq K_1 ||Tv||_A ||v||_A$$

which implies $\lambda_{\max}(BA) \leq K_1$

At the same time

$$\begin{aligned}(v,v)_A = \sum_{i=1}^{J}(v_i, P_i v)_A &\le \sum_{i=1}^{J}\left(R_i^{-1}v_i, v_i\right)^{1/2}\left(R_i A_i P_i v_i, v\right)_A^{1/2} \\ &\le \left(\sum_{i=1}^{J}\left(R_i^{-1}v_i, v_i\right)\right)^{1/2}\left(\sum_{i=1}^{J}\left(R_i A_i P_i v_i, v\right)_A\right)^{1/2} \\ &\le \sqrt{K_0}\,||v||_A\,(Tv,v)_A^{1/2}\end{aligned}$$

which implies $\lambda_{\min}(BA) \ge K_0$ and $\kappa(BA) \le K_0 K_1$

□

SSC.

For $E_i = (I - T_i)(I - T_{i-1})\cdots(I - T_1)$ and $E_0 = I$ Then

$$I - E_i = \sum_{j=1}^{i} T_j E_{j-1}$$

Lemma.

$$(2-\omega_1)\sum_{i=1}^{J}(T_i E_{i-1}v, E_{i-1}v)_A \le ||v||_A^2 - ||E_J v||_A^2$$

Proof.

$$\begin{aligned}||E_{i-1}v||_A^2 - ||E_i v||_A^2 &= ||T_i E_i v||_A^2 + 2(T_i E_{i-1}v, E_i v)_A \\ &= (T_i E_{i-1}A, T_i E_{i-1}v)_A + 2(T_i(I-T_i)E_{i-1}v, E_{i-1}v)_A \\ &= ((2I - T_i)T_i E_{i-1}v, E_{i-1}v)_A \ge (2-\omega_1)(T_i E_{i-1}v, E_{i-1}v)_A\end{aligned}$$

□

Theorem.

$$||E_J||_A^2 \le 1 - \frac{2-\omega_1}{K_0(1+K_1)^2}$$

Proof. First it is easy to show that

$$\sum_{i=1}^{J}(T_i v, v)_A \le (1+K_1)^2 \sum_{i=1}^{J}(T_i E_{i-1}v, E_{i-1}v)_A$$

At the same time we have

$$\sum_{i=1}^{J}(T_i v, E_{i-1}v)_A \le \left(\sum_{i=1}^{J}(T_i v, v)_A\right)^{1/2}\left(\sum_{i=1}^{J}(T_i E_{i-1}v, E_{i-1}v)_A\right)^{1/2}$$

and

$$\sum_{i=1}^{J}\sum_{j=1}^{i=1}(T_i v, T_j E_{j-1} v)_A \leq K_1 \left(\sum_{i=1}^{J}(T_i v, v)_A\right)^{1/2} \left(\sum_{i=1}^{J}(T_i E_{i-1} v, E_{i-1} v)_A\right)^{1/2}$$

Combining these three formulas leads to the theorem. □

At last I want to introduce a prefect websit:http://www.mgnet.org/

3. Numerical Results

In this section, we will test several algorithms

- G-S symmetric version
- Block G-S symmetric version
- Conjugate Gradient Method
- Conjugate Gradient Method Utlizing A V-cycle Multi grid method as a pre-conditioner
- Gauss Method

In order to do fair comparison, we will make the following comparison

- Conjugate Gradient Vs Pre-Condition Conjugate Gradient
- G-S Vs Block G-S

For the solution is a solution of a PDE, I will using a new method to calculate the numerical error instead of the vector $l2-$norm.

First we will give a method to do the interpolation on a square $[0,1]\times[0,1]$. Using four polynomial $xy, x(y-1), y(x-1), (x-1)(y-1)$ as bases and that is to say

$$f(x,y) = f(0,0)(x-1)(y-1) + f(1,0)x(1-y) + f(0,1)y(1-x) + f(1,1)xy$$

Using this interpolation, we get a function interpolated by our value on the grid points. Then calculating the error in $l^2([0,1]\times[0,1])$ using a Gaussian integral method. That is to say our error is defined as

$$\int_{[0,1]\times[0,1]} |\hat{f} - f_{true}|^2 dx$$

From Table1 we can see the numerical scheme is one order.

Table 1. Error For The Numerical Scheme

Grid Szie	128	256	512
Function l2 Error	7.4e-10	5.8e-11	**6.1e-12**
Vector l2 Error	0.00321316	0.00160693	0.000823503

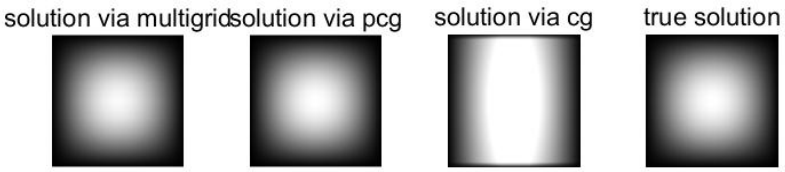


Fig. Solution of the anisotropic Poisson equation, when the diffusion coefficient is $\epsilon = 1e-2$ (grid size=512, cg method has a max iteration number of 300.)

3.1. **CG VS PCG.** PCG is method of using preconditioning in conjugate gradient methods.

Algorithm 1 (Pre-conditioned Conjugate Gradient)

Input: pre-conditioner:M, initial guess x_0

$r_0 = b - Ax_0$
while Not converged **do**
    (Pre-condition)Solve equation $Mz_k = r_k$ (If no pre-condition method $z_k = r_k$)
    $k = k + 1$
    if $k = 1$ **then**
        $p_1 = z_0$
    else
        $\beta_k = \frac{r_{k-1}^T z_{k-1}}{r_{k-2}^T z_{k-1}}$, $p_k = z_{k-1} + \beta_k p_{k-1}$
    $\alpha_k = \frac{r_{k-1}^T z_{k-1}}{p_k^T A p_k}$
    $x_k = x_{k-1} + \alpha_k p_k$, $r_k = r_{k-1} - \alpha_k A p_k$
return x_k

TABLE 2. PCG Iteration Times and CPU Time. Termination condition is $||r_k||/||r_0|| < 1e-8$.

Iteration Time	128	256	512
$\epsilon = 1e0$	4/0.15	4/0.66	4
$\epsilon = 1e-1$	6/0.22	6/0.96	6
$\epsilon = 1e-2$	16/0.58	16/1.95	16
$\epsilon = 1e-3$	39/1.07	42/4.88	43
$\epsilon = 1e-4$	55/1.95	93/14.44	110

In this section we utilize a V-cycle multi-grid method as the solve to pre-condition the problem. The converge speed is ploted as following.

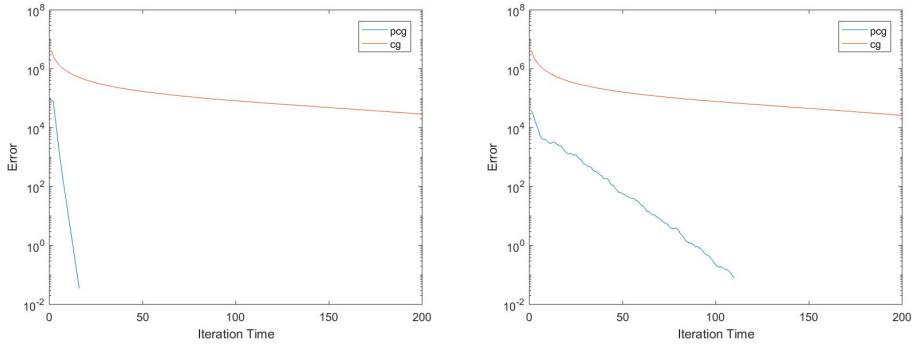


Fig. The error(vector l2-norm)-iteration time plot.(grid size:512, $\epsilon = 1e-2, 1e-4$)

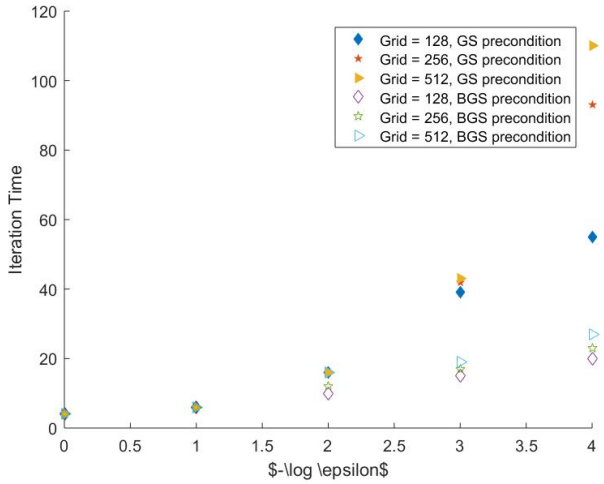
3.2. **Block G-s.** First, for the matrix can be written as

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1r} \\ A_{21} & A_{22} & \cdots & A_{2r} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ A_{r1} & A_{r2} & \cdots & A_{rr} \end{bmatrix}$$

here A_{ij} is a $n_i \times n_j$ sub-matrix and $\sum n_i = n$. Rewrite $x = (x_1^T, x_2^T, \cdots, x_r^T)^T, b = (b_1^T, b_2^T, \cdots, b_r^T)^T$. Let $A = D_B - L_B - U_B$, here $D_B = diag(A_{11}, A_{22}, \cdots, A_{rr})$

Then the block G-S method can be consider as iteration for solving the equation in order

$$A_{ii}x^{(k+1)} = b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{r} A_{ij}x_j^{(k)}$$



Setting	Iteration Time		CPU Time	
	G-S	Block G-S	G-S	Block G-S
$\epsilon = 1$	16428	8218	1.337	3.374
$\epsilon = 1e-1$	15964	1462	1.296	0.596
$\epsilon = 1e-2$	16108	180	1.306	0.079
$\epsilon = 1e-3$	16334	34	1.296	0.017
$\epsilon = 1e-4$	16646	14	1.328	0.008
$\epsilon = 1e-5$	16692	**8**	1.318	**0.005**

FIGURE 1. Iteration Times For G-S and Block G-S When Grid size is 64 and termination condition is $||r_k||/||r_0|| < 1e-6$ and comparison of being pre-conditioner.

3.3. **Partial Gauss Method.** In this section, we utilize the Gaussian elimination method to solve the linear equation. For the matrix have a bandwidth of *grid_size*, we don't need to calculate the zero part. Moreover, the matrix is **dominant diagonally dominant**, *the partial pivoting method is no longer needed here.* In short the algorithm can be listed as

Algorithm 2 (Partial Gaussian Elimination For Poisson Equation)

```
for k = 1:n-1 do
    A(k+1:min(k+grid_size,n),k)=A(k+1:min(k+grid_size,n),k)/A(k,k);
    A(k+1:min(k+grid_size,n),k+1:n)=A(k+1:min(k+grid_size,n),k+1:n)-
A(k+1:min(k+grid_size,n),k)*A(k,k+1:n);
L=speye(n,n)+tril(A,-1);
U=triu(A);
return [L, U]
```

TABLE 3. CPU Time & Error By Partial Gauss Method

CPU Time/Error	1e0	1e-2	1e-5
16	0.13/3.617e-6	0.09/3.617e-6	0.08/3.617e-6
32	0.21/2.35e-7	0.17/2.35e-7	0.18/2.35e-7
64	10.52/1.687e-8	10.93/1.687e-8	10.58/1.687e-8

3.4. **Heat Equation Solution.** We also made a test for the heat equation. We use several numerical methods to solve the PDE. We test the **explicit scheme, implicit scheme** and the **Crank-Nicolson Scheme**. For test, we utlize the initial condition $u(x, y, 0) = \sin(\pi x)\sin(\pi y)$, this PDE can be easily solved with solution $u(x, y, t) = e^{-2\pi^2 t}\sin(\pi x)\sin(\pi y)$

3.4.1. *Explicit Scheme.* The forward explicit scheme can be simply written as

$$\frac{U_{j,k}^{m+1} - U_{j,k}^{m}}{h_t} = \frac{U_{j+1,k}^{m} - 2U_{j,k}^{m} + U_{j-1,k}^{m}}{h_x^2} + \frac{U_{j,k+1}^{m} - 2U_{j,k}^{m} + U_{j,k-1}^{m}}{h_y^2}$$

3.4.2. *Implicit scheme.* Here we consider the implicit scheme has the following formula which contains the Crank-Nicolson Scheme as a special case

$$\begin{aligned}\frac{U_{j,k}^{m+1} - U_{j,k}^{m}}{h_t} &= (1-\theta)(\frac{\delta_x^2}{h_x^2} + \frac{\delta_y^2}{h_y^2})U_{j,k}^{m} + \theta(\frac{\delta_x^2}{h_x^2} + \frac{\delta_y^2}{h_y^2})U_{j,k}^{m+1} \\ &= (1-\theta)(\frac{U_{j+1,k}^{m} - 2U_{j,k}^{m} + U_{j-1,k}^{m}}{h_x^2} + \frac{U_{j,k+1}^{m} - 2U_{j,k}^{m} + U_{j,k-1}^{m}}{h_y^2}) + \\ &\theta(\frac{U_{j+1,k}^{m+1} - 2U_{j,k}^{m+1} + U_{j-1,k}^{m+1}}{h_x^2} + \frac{U_{j,k+1}^{m+1} - 2U_{j,k}^{m+1} + U_{j,k-1}^{m+1}}{h_y^2})\end{aligned}$$

For the Fourier wave $U_{j,k}^{m} = \lambda_\alpha^m e^{i(\alpha_x x_j + \alpha_y y_k)}$, we have

$$\lambda_\alpha = \frac{1 - 4(1-\theta)(\mu_x \sin^2\frac{\alpha_x h_x}{2} + \mu_y \sin^2\frac{\alpha_y h_y}{2})}{1 + 4\theta(\mu_x \sin^2\frac{\alpha_x h_x}{2} + \mu_y \sin^2\frac{\alpha_y h_y}{2})}$$

In order to obtain the stability we need

$$2(\mu_x + \mu_y)(1 - 2\theta) \le 1, 0 \le \theta \le \frac{1}{2}$$

Why Crank-Nicolson Scheme Choose $\theta = \frac{1}{2}$

The local truncation error can be proof to be $O(h_t^2 + h_x^2 + h_y^2)$ when $\theta = \frac{1}{2}$ and to be $O(h_t + h_x^2 + h_y^2)$ when θ is other value.

3.5. **Performance of different schemes.** In this section, all of the results is using **pcg** in order to get fast and robust solutions:

3.6. **Time that different NLA methods take.** In this part we test several NLA methods in the implicit scheme and plot the result in the table below

Some interesting observation.

- Sometimes the pde solver may become faster if the time step becomes smaller. It seems that the iterative number will become bigger, but at the same time the algebra equation $(I - \Delta t\Delta)u_{t+1} = u_t$ becomes easier. Here is not the case as the **Poisson Equation**. So I discover that **the Elliptic equation is a better case to exam the numerical schemes but not the evolution equations.**
- CN scheme is also faster than implicit scheme.

Test	Space step	Time step	CN Scheme	Implicit Scheme	Explicit Scheme
1	1/512	1/50	2.44e-7	6.88e-5	1.39e56
2	1/512	1/500	2.40e-11	5.79e-7	NaN
3	1/512	1/5000	**3.97e-16**	5.674e-9	1.393e56
4	1/128	1/50	2.43e-7	6.88e-5	7.80e32
5	1/128	1/500	1.55e-11	5.81e-7	7.79e32
6	1/128	1/5000	1.04e-12	5.82e-9	7.79e32
7	1/32	1/50	2.28e-7	6.90e-5	2.71e201
8	1/32	1/500	1.32e-10	6.02e-7	5.23e8
9	1/32	1/5000	2.55e-10	8.20e-9	3.62e-9

TABLE 4. Error at time $t = 0.2$

Test	Space step	Time step	Cholesky	Multigrid	Gauss-Seidel
1	1/32	1/20	0.08	0.19	0.38
2	1/64	1/20	0.31	0.37	1.48
3	1/128	1/20	5.36	**1.31**	6.45

TABLE 5. CPU time of different NLA methods while calculating the solution at $t = 0.2$