On Approximating Solutions of Two-Dimensional Boundary Value Problems by Using Finite Difference Method on Rectangular Uniform Mesh

Nguyen Quan Ba Hong*
Doan Tran Nguyen Tung†
Nguyen An Thinh‡
Students at Faculty of Math and Computer Science,
Ho Chi Minh University of Science, Vietnam
email. nguyenquanbahong@gmail.com
email. dtrngtung@live.com
email. anthinh297@gmail.com
blog. http://hongnguyenquanba.wordpress.com §

Sunday 25th December, 2016

Abstract

This assignment aims at solving two-dimensional boundary value problems with Dirichlet boundary conditions numerically by using finite difference methods on both rectangular uniform and square uniform meshes.

We also implement boundary value problems with Dirichlet-Neumann boundary conditions and pure Neumann boundary conditions by Matlab scripts at the end of this context.

*Student ID: 1411103
†Student ID: 1411352
‡Student ID: 1411389

Contents

List of Figures

1 Theoretical Problems

Problem 1.1 (Poisson Problem). *Let $\Omega = (0,1) \times (0,1) \subset \mathbb{R}^2$ be the domain and $f \in L^2(\Omega)$ be a given function.*

$$\Delta u(x,y) = f(x,y) \text{ in } \Omega \tag{1.1}$$

subject to a Dirichlet boundary condition

$$u(x,y) = 0, \text{ on } \partial\Omega \tag{1.2}$$

1. *How to discretize the previous system of equations by finite difference method with following mesh.*

$$\{x_i\}_{\in[0,N_x+1]} \text{ with } x_i = ih,\ h = \frac{1}{N_x+1}$$
$$\{y_j\}_{j\in[0,N_y+1]} \text{ with } y_j = jk,\ k = \frac{1}{N_y+1}.$$

   Noting that there exist positive constants α_0, β_0 such that

$$\alpha_0 \leq \frac{h}{k} \leq \beta_0 \tag{1.3}$$

2. *Demonstrate the convergence of the discrete solution to exact solution.*

Remark 1.2. We note that the sign of the left-hand side of (1.1) is not important since we can take $-f(x)$ in the right-hand side instead. Thus, we can assume there is no minus in both sides and focus to solve (1.1) numerically.

Remark 1.3 (Notation problems). We should explain slightly about our notations in *Problem 1.* There are some reasons to do so.

1. We denote mesh width in x- and y-direction by h and k instead of Δx and Δy since the notation Δ will make us confuse with Laplacian which is used many times later. Moreover, this also simplify (1.1) instead of writing

$$\nabla^2 u(x,y) = f(x,y) \tag{1.4}$$

   and Δ^2 instead of ∇^4.

2. We only consider Dirichlet boundary condition (1.2) in Problem 1. But this can be extended for Dirichlet, Dirichlet-Neumann and pure Neumann boundary conditions by modifying all the terms involving the boundary $\partial\Omega$ in the MATLAB code easily as you will see at the very end of this context.

   Hence, we just need focus on Dirichlet boundary condition. Both Dirichlet-Neumann boundary condition and pure Neumann will be demonstrated in MATLAB implementation.

3. We index our mesh from 0 to $N_x + 1$ or $N_y + 1$ instead of N_x or N_y so that we can eliminate u with indices 0 or $N_x + 1, N_y + 1$ and focus to approximate u with indices $1, 2, \ldots, N_x$ and $1, 2, \ldots, N_y$. This is more natural and compact than $1, 2, \ldots, N_x - 1$ and $1, 2, \ldots, N_y - 1$.

4. Finally, why do we need (1.3)? The reason is that (1.3) forces our mesh to be reasonable enough to implement. It would be very terrible if (1.3) collapses. In that situation, you can not both estimate theoretically and implement code practically, as we will see in the MATLAB implementations.

With a given rectangular uniform mesh, i.e., it is uniform respect to each of x- and y-directions but not completely uniform, we attempt to compute a grid function consisting of values

$$u_{ij}, \quad i = 0, 1, \ldots, N_x + 1, \quad j = 0, 1, \ldots, N_y + 1 \tag{1.5}$$

where u_{ij} is an approximation to the solution $u(x_i, y_j)$. We also define the mesh width of this rectangular uniform mesh as maximum of mesh widths in two directions.

$$H = \max\{h, k\} \tag{1.6}$$

This mesh width H is used to estimate error terms and prove the stability and convergence of our finite difference methods later.

Remark 1.4. We can freely choose H as an arbitrary norm in Euclidean space $\mathbb{R}^2$. The purpose of H is only

$$(h, k) \to (0, 0) \Leftrightarrow H \to 0 \tag{1.7}$$

So you can choose either (1.6) or

$$H = \|(h, k)\|_1 = h + k \tag{1.8}$$

$$H = \|(h, k)\|_2 = \sqrt{h^2 + k^2} \tag{1.9}$$

$$H = \|(h, k)\|_p = (h^p + k^p)^{\frac{1}{p}}, p > 1 \tag{1.10}$$

$$H = \|(h, k)\|_\infty = \max\{h, k\} \tag{1.11}$$

Return to our problem, from the boundary conditions (1.2) we know that

$$u_{ij} = 0, \text{ if } i \in \{0, N_x + 1\} \text{ or } j \in \{0, N_y + 1\} \tag{1.12}$$

and so we have $N_x \times N_y$ unknowns values

$$u_{ij}, \quad 1, 2, \ldots, N_x, \quad j = 1, 2, \ldots, N_y \tag{1.13}$$

to compute.

2 Rectangular Uniform Mesh

This section focus on solving (1.1) numerically on generally rectangular uniform mesh which also includes square uniform one. A separated section about solving (1.1) numerically on square uniform mesh is discussed after this section.

2.1 The 5-Point Stencil for The Laplacian

We are using a uniform Cartesian grid consisting of grid points $\{x_i\}_{\in[0,N_x+1]}$ with $x_i = ih$ where $h = \dfrac{1}{N_x+1}$ and $\{y_j\}_{j\in[0,N_y+1]}$ with $y_j = jk$ where $k = \dfrac{1}{N_y+1}$.

A section of such a grid is shown in Figure 1.

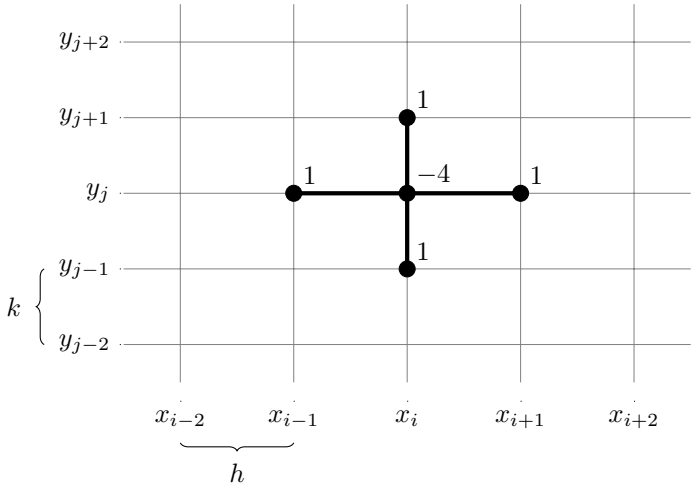

Figure 1: The 5-point stencil for the Laplacian about the point (i, j) is indicated.

Let u_{ij} represent an approximation to $u(x_i, y_j)$. To discretize (1.1) we have from (1.1)

$$u_{xx}(x_i, y_j) + u_{yy}(x_i, y_j) = f(x_i, y_j) \tag{2.1}$$

for $i = 1, 2, \ldots, N_x$, $j = 1, 2, \ldots, N_y$ and replace the x- and y-derivatives with centered finite differences, which gives

$$\frac{1}{h^2}(u_{i-1,j} - 2u_{ij} + u_{i+1,j}) + \frac{1}{k^2}(u_{i,j-1} - 2u_{ij} + u_{i,j+1}) = f_{ij} \tag{2.2}$$

or equivalently,

$$\frac{1}{h^2}u_{i-1,j} + \frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j-1} + \frac{1}{k^2}u_{i,j+1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.3}$$

We glance at (2.3) at the boundary $\partial\Omega$ in the following cases respect to i and j. A little combinatorial counting arises right here. There are three possible choices for i, in which the first and second choices indicate that (2.3) involves the x-boundary.

$$i = 1 \tag{2.4}$$
$$i = N_x \tag{2.5}$$
$$i \notin \{1, N_x\} \tag{2.6}$$

Similarly, there are also three possible choices for j, in which the first and second choices indicate that (2.3) involves the y-boundary.

$$j = 1 \tag{2.7}$$
$$j = N_y \tag{2.8}$$
$$j \notin \{1, N_y\} \tag{2.9}$$

Hence, there are $3 \times 3 = 9$ possible choices in total, which are exactly following cases.

1. Case $i = 1, j = 1$. (2.3) becomes

$$\frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j+1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.10}$$

2. Case $i = N_x, j = 1$. (2.3) becomes

$$\frac{1}{h^2}u_{i-1,j} + \frac{1}{k^2}u_{i,j+1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.11}$$

3. Case $i \notin \{1, N_x\}, j = 1$. (2.3) becomes

$$\frac{1}{h^2}u_{i-1,j} + \frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j+1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.12}$$

4. Case $i = 1, j = N_y$. (2.3) becomes

$$\frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j-1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.13}$$

5. Case $i = N_x, j = N_y$. (2.3) becomes

$$\frac{1}{h^2}u_{i-1,j} + \frac{1}{k^2}u_{i,j-1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.14}$$

6. Case $i \notin \{1, N_x\}, j = N_y$. (2.3) becomes

$$\frac{1}{h^2}u_{i-1,j} + \frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j-1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.15}$$

7. Case $i = 1, j \notin \{1, N_y\}$. (2.3) becomes

$$\frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j-1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.16}$$

8. Case $i = N_x, j \notin \{1, N_y\}$. (2.3) becomes

$$\frac{1}{h^2}u_{i-1,j} + \frac{1}{k^2}u_{i,j-1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} = f_{ij} \tag{2.17}$$

9. Case $i \notin \{1, N_x\}, j \notin \{1, N_y\}$. (2.3) does not change.

This finite difference scheme can be represented by the 5-point stencil shown in Figure 1. We have both an unknown u_{ij} and an equation of the form (2.3) at each of N_xN_y grid points for $i = 1, 2, \ldots, N_x$ and $j = 1, 2, \ldots, N_y$ where $h = \dfrac{1}{N_x + 1}$ and $k = \dfrac{1}{N_y + 1}$.

We thus have a linear system of N_xN_y unknowns. The difference equations at points near the boundary will of course involve the known boundary values, just as in the one-dimensional case, which can be moved to the right-hand side.

2.2 Ordering The Unknowns and Equations

Problem 2.1. *Specify some orders of the unknowns and equations and find the matrix equation in those orders.*

If we collect all these equations together into a matrix equation, we will have $N_xN_y \times N_xN_y$ matrix that is very sparse, i.e., most of the elements are zero. Since each equation involves at most five unknowns (fewer near the boundary), each row of the matrix has at most five nonzeros and at least $N_xN_y - 5$ elements that are zero.

Recall from one-dimensional problem, see [1], that the structure of the matrix depends on the order we choose to enumerate the unknowns. Unfortunately, in two dimensional spaces the structure of the matrix is not as compact as in one dimension, no matter how we order the unknowns, and the nonzeros cannot be as nicely clustered near the main diagonal.

One obvious choice is the *natural rowwise ordering,* where we take the unknowns along the bottom row, $u_{11}, u_{21}, \ldots, u_{N_x1}$, followed by the unknowns in the second row, $u_{12}, u_{22}, \ldots, u_{N_x2}$, and so on, until reaching the top row $u_{1N_y}, u_{2N_y}, \ldots, u_{N_xN_y}$ as in Figure 2.

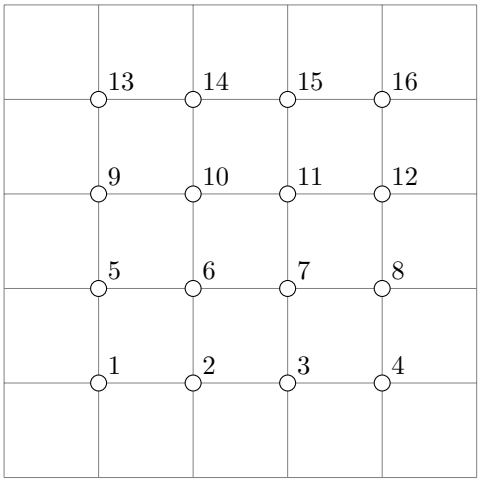

Figure 2: The natural rowwise order of unknowns and equations on a 4×4 grid.

Problem 2.2. *Find the explicit formula of the matrix equation under the natural rowwise ordering.*

The vector of unknowns is partitioned as

$$u = \begin{bmatrix} u^{[1]} \\ u^{[2]} \\ \vdots \\ u^{[N_y]} \end{bmatrix} \tag{2.18}$$

where

$$u^{[j]} = \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{N_x,j} \end{bmatrix}, \quad j = 1, 2, \ldots, N_y \tag{2.19}$$

This gives a matrix equation where A has the form

$$A^{h,k} = \begin{bmatrix} T^{h,k} & \frac{1}{k^2} I_{N_x} & & & \\ \frac{1}{k^2} I_{N_x} & T^{h,k} & \frac{1}{k^2} I_{N_x} & & \\ & \frac{1}{k^2} I_{N_x} & T^{h,k} & \frac{1}{k^2} I_{N_x} & \\ & & \ddots & \ddots & \ddots \\ & & & \frac{1}{k^2} I_{N_x} & T^{h,k} \end{bmatrix} \tag{2.20}$$

which is an $N_y \times N_y$ *block tridiagonal matrix* in which each block $T^{h,k}$ or I_{N_x} is itself an $N_x \times N_x$ matrix,

$$T^{h,k} = \begin{bmatrix} -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \frac{1}{h^2} & & \\ \frac{1}{h^2} & -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \frac{1}{h^2} & \\ & \ddots & \ddots & \ddots \\ & & \frac{1}{h^2} & -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \end{bmatrix} \tag{2.21}$$

and I_{N_x} is the $N_x \times N_x$ identity matrix. While this has a nice structure, the 1 values in the I_{N_x} matrices are separated from the diagonal by $N_x - 1$ zeros, since these coefficients correspond to grid points lying above or below the central point in the stencil and hence are in the next or previous row of unknowns.

Another possibility, which has some advantages in the context of certain iterative methods, is to use the *red-black ordering* (or checkerboard ordering) in Figure 3.

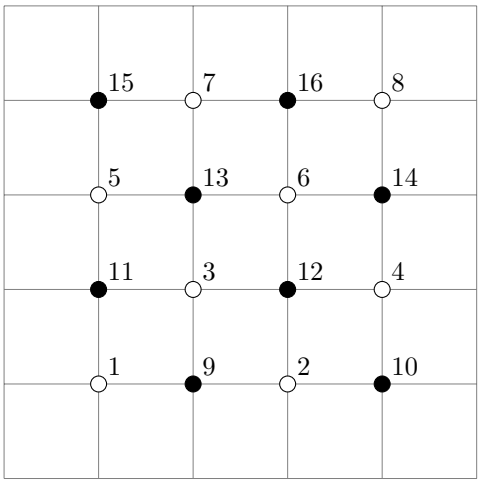

Figure 3: The red-black order of unknowns and equations on a 4×4 grid.

This is the two-dimensional analogue of the odd-even ordering in one dimension, see [1]. This ordering is significant because all four neighbors of a red grid point are black points, and vice versa, and it leads to a matrix equation with the structure

$$\begin{bmatrix} D & H \\ H^T & D \end{bmatrix} \begin{bmatrix} u_{red} \\ u_{black} \end{bmatrix} = \begin{bmatrix} f_{red} \\ -f_{black} \end{bmatrix} \tag{2.22}$$

where $D = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) I_{\frac{N_x N_y}{2}}$ is a diagonal matrix of dimension $\frac{N_x N_y}{2}$ (of course, we must assume that there is at least one even number in two numbers N_x and N_y) and H is a banded matrix of the same dimension with four nonzero diagonals.

Problem 2.3. *Find an explicit formula of the matrix $A^{h,k}$ under the red-black ordering.*

When direct methods such as Gaussian elimination are used to solve the system, one typically wants to order the equations and unknowns so as to reduce the amount of fill-in during the elimination procedure as much as possible. This is done automatically if the backslash operators in MATLAB is used to solve the system, provided it is set up using sparse storage which will be discussed later.

2.3 The Main Diagonal Block Matrices under Natural Rowwise Ordering

See Appendices, [3]. We now attempt to compute the determinant of (2.20) explicitly. To this end, we first find some necessary information about the main diagonal block $T^{h,k}$ of (2.20).

Remark 2.4. Before taking a lot of complicated computations, we must emphasize that some computations here, such as spectral radius of the main diagonal

block matrices, etc., are not necessary for proving stability of our finite difference method in 2-norm.

Why do we still compute those? We, the authors of this project, want to do all of things in our abilities. We really think these results may be some very important ingredients in future purposes, such as proving stability in other norms, generalizing some fabulous results, supporting readers' works and even discovering finite difference methods of higher order.

Perhaps you have a lot of questions after some computations. Please wait, we will indicate the purposes of those in the next section. So let's join the "crazy computation club" right now.

2.3.1 Determinant of The Main Diagonal Block Matrices

Problem 2.5. *Use* (5.5) *to compute the determinant of the main diagonal block matrices of A, i.e.,* $\det T^{h,k}$ *and then deduce the invertibility of* $T^{h,k}$.

Solution 1. Using the formula for determinnant of ordinary tridiagonal matrices in Section 5.1, (5.4)-(5.5) becomes

$$\det T^{h,k} = \left[\begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & -\frac{1}{h^4} \\ 1 & 0 \end{bmatrix}^{N_x-1} \begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & 0 \\ 1 & 0 \end{bmatrix} \right]_{11} \tag{2.23}$$

To compute this, we need to compute explicitly the term containing exponent in right-hand side.

To this end, we diagonalize the matrix

$$\begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & -\frac{1}{h^4} \\ 1 & 0 \end{bmatrix} \tag{2.24}$$

and obtain

$$\begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & -\frac{1}{h^4} \\ 1 & 0 \end{bmatrix} = VDV^{-1} \tag{2.25}$$

with

$$V = \begin{bmatrix} \lambda_1 & \lambda_2 \\ 1 & 1 \end{bmatrix} \tag{2.26}$$

$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{2.27}$$

$$V^{-1} = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} 1 & -\lambda_2 \\ -1 & \lambda_1 \end{bmatrix} \tag{2.28}$$

where

$$\lambda_1 = -\frac{1}{h^2} - \frac{1}{k^2} - \frac{1}{k}\sqrt{\frac{2}{h^2}+\frac{1}{k^2}} \tag{2.29}$$

$$\lambda_2 = -\frac{1}{h^2} - \frac{1}{k^2} + \frac{1}{k}\sqrt{\frac{2}{h^2} + \frac{1}{k^2}} \tag{2.30}$$

then

$$\begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & -\frac{1}{h^4} \\ 1 & 0 \end{bmatrix}^{N_x-1} \begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & 0 \\ 1 & 0 \end{bmatrix} \tag{2.31}$$

$$= VD^{N_x-1}V^{-1} \begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & 0 \\ 1 & 0 \end{bmatrix} \tag{2.32}$$

$$= \frac{1}{\lambda_1-\lambda_2} \begin{bmatrix} \lambda_1 & \lambda_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1^{N_x-1} & 0 \\ 0 & \lambda_2^{N_x-1} \end{bmatrix} \begin{bmatrix} 1 & -\lambda_2 \\ -1 & \lambda_1 \end{bmatrix} \begin{bmatrix} \lambda_1+\lambda_2 & 0 \\ 1 & 0 \end{bmatrix} \tag{2.33}$$

$$= \frac{1}{\lambda_1-\lambda_2} \begin{bmatrix} \lambda_1^{N_x+1}-\lambda_2^{N_x+1} & 0 \\ \lambda_1^{N_x}-\lambda_2^{N_x} & 0 \end{bmatrix} \tag{2.34}$$

Hence, (2.23) becomes

$$\det T^{h,k} = \frac{\lambda_1^{N_x+1}-\lambda_2^{N_x+1}}{\lambda_1-\lambda_2} \tag{2.35}$$

It is obvious to deduce from (2.35) that $T^{h,k}$ is nonsingular. □

Solution 2. See [6]-p.15.

We need the following lemma.

Lemma 2.6. *Let A_n be a tridiagonal matrix takes the form*

$$A_n = \begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & & & & \\ & & c_{n-2} & a_{n-1} & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix} \tag{2.36}$$

The determinant of this matrix can be computed by using the following recurrent relation. Let

$$\Delta_0 = 1 \tag{2.37}$$

$$\Delta_n = \det A_n, \forall n \in Z_+ \tag{2.38}$$

and the recurrence relation is

$$\Delta_k = a_k\Delta_{k-1} - b_{k-1}c_{k-1}\Delta_{k-2}, \forall k \geq 2 \tag{2.39}$$

Proof of Lemma 2.6. Expanding A_n with respect to the last row yields (2.39). *square*

Return to our problem, we define

$$\Delta_{N_x} = \det T^{h,k}, \forall N_x \in \mathbb{N} \tag{2.40}$$

Then (2.39) becomes

$$\Delta_0 = 1 \tag{2.41}$$

$$\Delta_1 = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \tag{2.42}$$

$$\Delta_k = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)\Delta_{k-1} - \frac{1}{h^4}\Delta_{k-2}, \forall k \geq 2 \tag{2.43}$$

We see that (2.43) is a second-order difference equation. We now solve (2.43).

The characteristic polynomial of (2.39) is

$$\lambda^2 + 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)\lambda + \frac{1}{h^4} = 0 \tag{2.44}$$

This polynomial equation of degree 2 has two roots

$$\lambda_1 = -\frac{1}{h^2} - \frac{1}{k^2} - \frac{1}{k}\sqrt{\frac{2}{h^2} + \frac{1}{k^2}} \tag{2.45}$$

$$\lambda_2 = -\frac{1}{h^2} - \frac{1}{k^2} + \frac{1}{k}\sqrt{\frac{2}{h^2} + \frac{1}{k^2}} \tag{2.46}$$

Thus, the general root of (2.39) has the form

$$\Delta_k = C_1\lambda_1^k + C_2\lambda_2^k \tag{2.47}$$

$$= C_1\left(-\frac{1}{h^2} - \frac{1}{k^2} - \frac{1}{k}\sqrt{\frac{2}{h^2} + \frac{1}{k^2}}\right)^k \tag{2.48}$$

$$+ C_2\left(-\frac{1}{h^2} - \frac{1}{k^2} + \frac{1}{k}\sqrt{\frac{2}{h^2} + \frac{1}{k^2}}\right)^k, \forall k \in \mathbb{N} \tag{2.49}$$

Substituting $k = 0$ and $k = 1$ in (2.47)-(2.49) yields

$$C_1 + C_2 = 1 \tag{2.50}$$

$$C_1\lambda_1 + C_2\lambda_2 = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \tag{2.51}$$

Solving this linear system of equations out yields

$$C_1 = \frac{h\sqrt{h^2 + 2k^2} + h^2 + k^2}{2h\sqrt{h^2 + 2k^2}} \tag{2.52}$$

$$= -\frac{hk^2}{2\sqrt{h^2 + 2k^2}}\lambda_1 \tag{2.53}$$

$$C_2 = \frac{h\sqrt{h^2 + 2k^2} - h^2 - k^2}{2h\sqrt{h^2 + 2k^2}} \tag{2.54}$$

$$= \frac{hk^2}{2\sqrt{h^2 + 2k^2}}\lambda_2 \tag{2.55}$$

Hence, (2.47)-(2.49) becomes

$$\Delta_k = -\frac{hk^2}{2\sqrt{h^2 + 2k^2}}\lambda_1^{k+1} + \frac{hk^2}{2\sqrt{h^2 + 2k^2}}\lambda_2^{k+1} \tag{2.56}$$

$$= \frac{hk^2}{2\sqrt{h^2+2k^2}} \left(\lambda_2^{k+1} - \lambda_1^{k+1}\right) \tag{2.57}$$

$$= \frac{hk^2}{2\sqrt{h^2+2k^2}} \begin{pmatrix} \left(-\frac{1}{h^2} - \frac{1}{k^2} + \frac{1}{k}\sqrt{\frac{2}{h^2}+\frac{1}{k^2}}\right)^{k+1} \\ -\left(-\frac{1}{h^2} - \frac{1}{k^2} - \frac{1}{k}\sqrt{\frac{2}{h^2}+\frac{1}{k^2}}\right)^{k+1} \end{pmatrix} \tag{2.58}$$

In particular, we obtain

$$\det T^{h,k} = \Delta_{N_x} \tag{2.59}$$

$$= \frac{hk^2}{2\sqrt{h^2+2k^2}} \begin{pmatrix} \left(-\frac{1}{h^2} - \frac{1}{k^2} + \frac{1}{k}\sqrt{\frac{2}{h^2}+\frac{1}{k^2}}\right)^{N_x+1} \\ -\left(-\frac{1}{h^2} - \frac{1}{k^2} - \frac{1}{k}\sqrt{\frac{2}{h^2}+\frac{1}{k^2}}\right)^{N_x+1} \end{pmatrix} \tag{2.60}$$

It is obvious to deduce that $\det T^{h,k} \neq 0$ from (2.59)-(2.60), i.e., $T^{h,k}$ is nonsingular. □

SOLUTION 3. We can compute $\det T^{h,k}$ by computing and multiplying all eigenvalues of $T^{h,k}$ as (2.71). □

2.3.2 Eigenvalue Decomposition of The Main Diagonal Block Matrices

Problem 2.7. *Find an eigenvalue decomposition of the main diagonal block matrices $T^{h,k}$.*

Problem 2.8. *Since $T^{h,k}$ is normal, using Theorem 5.32 we deduce that there exists a unitary matrix Q such that*

$$T^{h,k} = Q diag\left\{\lambda_i\left(T^{h,k}\right)\right\} Q^* \tag{2.61}$$

Find such a Q.

We recall that

$$T^{h,k} = \begin{bmatrix} -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & \frac{1}{h^2} & & \\ \frac{1}{h^2} & -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) & \frac{1}{h^2} & \\ & \ddots & \ddots & \ddots \\ & & \frac{1}{h^2} & -2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) \end{bmatrix} \tag{2.62}$$

Using Definition 5.21, we see that $T^{h,k}$ is a strictly diagonally dominant matrix.

$$|a_{ii}| = 2\left(\frac{1}{h^2}+\frac{1}{k^2}\right) \tag{2.63}$$

$$> \frac{2}{h^2} \tag{2.64}$$

$$= \sum_{j\neq i} |a_{ij}|, \quad i = 1, 2, \ldots, N_x \tag{2.65}$$

Hence, using Levy-Desplanques Theorem 5.22, we deduce that $T^{h,k}$ is nonsingular.

We also notice that $T^{h,k}$ is a tridiagonal Toeplitz matrix, see Subsection 5.3, i.e., (5.67) with

$$\delta = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \tag{2.66}$$

$$\sigma = \frac{1}{h^2} \tag{2.67}$$

$$\tau = \frac{1}{h^2} \tag{2.68}$$

Using (5.71), we deduce that the eigenvalues of

$$T^{h,k} = \left(N_x; \frac{1}{h^2}, -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right), \frac{1}{h^2}\right) \tag{2.69}$$

are given by

$$\lambda_i\left(T^{h,k}\right) = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + \frac{2}{h^2}\cos\frac{i\pi}{N_x+1}, i = 1, 2, \ldots, N_x \tag{2.70}$$

As a consequences, we have

$$\det T^{h,k} = \prod_{i=1}^{N_x}\left(-2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + \frac{2}{h^2}\cos\frac{i\pi}{N_x+1}\right) \tag{2.71}$$

We now need to estimate the magnitudes of eigenvalues of $T^{h,k}$.

$$\left|\lambda_i\left(T^{h,k}\right)\right| = \left|-2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + \frac{2}{h^2}\cos\frac{i\pi}{N_x+1}\right| \tag{2.72}$$

$$= \frac{2}{h^2}\left(1 - \cos\frac{i\pi}{N_x+1}\right) + \frac{2}{k^2} \tag{2.73}$$

$$\geq \frac{2}{k^2}, i = 1, 2, \ldots, N_x \tag{2.74}$$

Since $\sigma\tau = \frac{1}{h^4} \neq 0$, (5.73) yields that $T^{h,k}$ has N_x simple eigenvalues, which lie on the closed line segment.

$$S_{\lambda(T)} = \left\{-2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + te^{i\arg\frac{1}{h^2}} : t \in R, |t| \leq \frac{2}{h^2}\cos\frac{\pi}{N_x+1}\right\} \tag{2.75}$$

$$= \left\{-2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + t : t \in R, |t| \leq \frac{2}{h^2}\cos\frac{\pi}{N_x+1}\right\} \tag{2.76}$$

$$= \left[-2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) - \frac{2}{h^2}\cos\frac{\pi}{N_x+1}; -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + \frac{2}{h^2}\cos\frac{\pi}{N_x+1}\right] \tag{2.77}$$

Moreover, the spectral radius of $T^{h,k}$ is given by

$$\rho\left(T^{h,k}\right) = \max\left\{\frac{2}{h^2}\left(1 - \cos\frac{\pi}{N_x+1}\right) + \frac{2}{k^2}, \frac{2}{h^2}\left(1 - \cos\frac{N_x\pi}{N_x+1}\right) + \frac{2}{k^2}\right\} \tag{2.78}$$

$$= \frac{2}{h^2}\left(1 - \cos\frac{N_x\pi}{N_x+1}\right) + \frac{2}{k^2} \tag{2.79}$$

Using (5.76), we also have that the components of the right eigenvectors $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,N_x}]^T$ associated with the eigenvalue $\lambda_I\left(T^{h,k}\right)$ are given by

$$x_{i,j} = \sin\frac{ij\pi}{N_x+1}, \quad i = 1:N_x, j = 1:N_x \tag{2.80}$$

We now have

$$T^{h,k} = X diag_{N_x}\left\{\lambda_i\left(T^{h,k}\right)\right\} X^{-1} \tag{2.81}$$

where X is the matrix whose columns are the eigenvectors of A, i.e.,

$$X = \left[\begin{array}{c|c|c|c} & & & \\ x_1 & x_2 & \cdots & x_{N_x} \\ & & & \end{array}\right] \tag{2.82}$$

Hence,

$$X_{ij} = x_{j,i} \tag{2.83}$$

$$= \sin\frac{ij\pi}{N_x+1}, \quad i = 1:N_x, j = 1:N_x \tag{2.84}$$

i.e.,

$$X = \left\{\sin\frac{ij\pi}{N_x+1}\right\}_{i,j=1}^{N_x} \tag{2.85}$$

We now compute $X^T X$.

$$\left[X^T X\right]_{ij} = \sum_{k=1}^{N_x} X_{ik}^T X_{kj} \tag{2.86}$$

$$= \sum_{k=1}^{N_x} \sin\frac{ik\pi}{N_x+1}\sin\frac{kj\pi}{N_x+1} \tag{2.87}$$

$$= \frac{1}{2}\sum_{k=1}^{N_x}\left(\cos\frac{(i-j)k\pi}{N_x+1} - \cos\frac{(i+j)k\pi}{N_x+1}\right) \tag{2.88}$$

$$= \frac{1}{2}\sum_{k=1}^{N_x}\left(\cos\frac{(i-j)k\pi}{N_x+1} - \cos\frac{(i+j)k\pi}{N_x+1}\right) \tag{2.89}$$

$$= \frac{1}{2}\sum_{k=1}^{N_x}\cos\frac{(i-j)k\pi}{N_x+1} - \frac{1}{2}\sum_{k=1}^{N_x}\cos\frac{(i+j)k\pi}{N_x+1} \tag{2.90}$$

Next, we will compute each element of $X^T X$ by using the following famous identities.

Theorem 2.9 (Lagrange's trigonometric identities, [10]). *We have*

$$\sum_{n=1}^{N}\sin(n\theta) = \frac{1}{2}\cot\frac{\theta}{2} - \frac{\cos\left(\left(N+\frac{1}{2}\right)\theta\right)}{2\sin\frac{\theta}{2}} \tag{2.91}$$

$$\sum_{n=1}^{N} \cos(n\theta) = -\frac{1}{2} + \frac{\sin\left(\left(N+\frac{1}{2}\right)\theta\right)}{2\sin\frac{\theta}{2}} \tag{2.92}$$

We now consider two cases with respect to indices i and j.

1. Case $i = j$. (2.86)-(2.90) becomes

$$\left[X^T X\right]_{ii} = \frac{1}{2}\sum_{k=1}^{N_x} \cos 0 - \frac{1}{2}\sum_{k=1}^{N_x} \cos\frac{2ik\pi}{N_x+1} \tag{2.93}$$

$$= \frac{N_x}{2} - \frac{1}{2}\left(-\frac{1}{2} + \frac{\sin\left(\left(N_x+\frac{1}{2}\right)\frac{2i\pi}{N_x+1}\right)}{2\sin\left(\frac{i\pi}{N_x+1}\right)}\right) \tag{2.94}$$

$$= \frac{N_x}{2} - \frac{1}{2}\left(-\frac{1}{2} + \frac{\sin\left(2i\pi - \frac{i\pi}{N_x+1}\right)}{2\sin\left(\frac{i\pi}{N_x+1}\right)}\right) \tag{2.95}$$

$$= \frac{N_x}{2} - \frac{1}{2}\left(-\frac{1}{2} - \frac{\sin\left(\frac{i\pi}{N_x+1}\right)}{2\sin\left(\frac{i\pi}{N_x+1}\right)}\right) \tag{2.96}$$

$$= \frac{N_x+1}{2}, \quad i = 1, 2, \ldots, N_x \tag{2.97}$$

2. Case $i \neq j$. (2.86)-(2.90)

$$\left[X^T X\right]_{ij} = \frac{1}{2}\left(-\frac{1}{2} + \frac{\sin\left(\left(N_x+\frac{1}{2}\right)\frac{(i-j)\pi}{N_x+1}\right)}{2\sin\left(\frac{(i-j)\pi}{2(N_x+1)}\right)}\right) \tag{2.98}$$

$$-\frac{1}{2}\left(-\frac{1}{2} + \frac{\sin\left(\left(N_x+\frac{1}{2}\right)\frac{(i+j)\pi}{N_x+1}\right)}{2\sin\left(\frac{(i+j)\pi}{2(N_x+1)}\right)}\right) \tag{2.99}$$

$$= \frac{\sin\left((i-j)\pi - \frac{1}{2}\frac{(i-j)\pi}{N_x+1}\right)}{4\sin\left(\frac{(i-j)\pi}{2(N_x+1)}\right)} \tag{2.100}$$

$$- \frac{\sin\left((i+j)\pi - \frac{1}{2}\frac{(i+j)\pi}{N_x+1}\right)}{4\sin\left(\frac{(i+j)\pi}{2(N_x+1)}\right)} \tag{2.101}$$

$$= \frac{\sin\left(\frac{1}{2}\frac{(i-j)\,\pi}{N_x+1}\right)}{4\sin\left(\frac{(i-j)\,\pi}{2\,(N_x+1)}\right)} - \frac{\sin\left(\frac{1}{2}\frac{(i+j)\,\pi}{N_x+1}\right)}{4\sin\left(\frac{(i+j)\,\pi}{2\,(N_x+1)}\right)} \tag{2.102}$$

$$= \frac{1}{4} - \frac{1}{4} \tag{2.103}$$

$$= 0, \quad \forall i \neq j \tag{2.104}$$

Hence,

$$X^T X = \frac{N_x+1}{2} I_{N_x} \tag{2.105}$$

Similarly, we also have

$$XX^T = \frac{N_x+1}{2} I_{N_x} \tag{2.106}$$

Therefore, $Q = \sqrt{\frac{2}{N_x+1}} X$ is orthogonal (real unitary). It is fascinating to notice that this Q is the unitary matrix which is mentioned in Theorem 5.32, i.e., we now have

$$T^{h,k} = Q diag_{N_x}\left\{\lambda_i\left(T^{h,k}\right)\right\} Q^T \tag{2.107}$$

where Q is defined by

$$Q = \left\{\sqrt{\frac{2}{N_x+1}} \sin\frac{ij\pi}{N_x+1}\right\}_{i,j=1}^{N_x} \tag{2.108}$$

By (2.108), we see that Q is symmetric. Hence, (2.107) becomes

$$T^{h,k} = Q diag_{N_x}\left\{\lambda_i\left(T^{h,k}\right)\right\} Q \tag{2.109}$$

We now have enough information about $T^{h,k}$ to make the next big move in our journey.

2.4 The Block Matrix under Natural Rowwise Ordering

We are now investigating to discover some interesting properties of $A^{h,k}$ under natural rowwise ordering. Readers can also investigate to discover this under the red-black ordering.

2.4.1 Determinant of The Block Matrix

Problem 2.10. *Compute the determinant of the block matrix $A^{h,k}$.*

SOLUTION. Taking product of (2.167)-(2.169) yields the result. □

Problem 2.11. *Use Section 5.1.2, compute the determinant of the block matrix $A^{h,k}$.*

Corollary 2.12. *The matrix $A^{h,k}$ is invertible.*

SOLUTION. The determinant of $A^{h,k}$ is exactly the product of its N_xN_y eigenvalues (2.167)-(2.169). Since none of these eigenvalues is zero, we deduce that $A^{h,k}$ is invertible. □

2.4.2 Inverse Matrix of The Block Matrix

Using Section 5.2.2 for

$$T = A^{h,k} \tag{2.110}$$
$$A = \frac{1}{k^2} I_{N_x} \tag{2.111}$$
$$B = T^{h,k} \tag{2.112}$$
$$C = \frac{1}{k^2} I_{N_x} \tag{2.113}$$
$$m = N_x \tag{2.114}$$
$$n = N_y \tag{2.115}$$
$$\epsilon = 1 \tag{2.116}$$

(5.67) becomes

$$\left(\frac{1}{k^2} I_{N_x}\right)^{-1} T^{h,k} = k^2 T^{h,k} \tag{2.117}$$
$$= k^2 Q diag_{N_x}\left\{\lambda_i\left(T^{h,k}\right)\right\} Q^T \tag{2.118}$$
$$= Q diag_{N_x}\left\{k^2\lambda_i\left(T^{h,k}\right)\right\} Q^T \tag{2.119}$$

Hence,

$$\bar{\lambda}_i = k^2\lambda_i\left(T^{h,k}\right) \tag{2.120}$$

We have

$$\bar{\lambda}_i^2 = \left(k^2\lambda_i\left(T^{h,k}\right)\right)^2 \tag{2.121}$$
$$= \left(\frac{2k^2}{h^2}\left(1 - \cos\frac{i\pi}{N_x+1}\right) + 2\right)^2 \tag{2.122}$$
$$> 4 \tag{2.123}$$

Thus, we now use hyperbolic sine and hyperbolic cosine in Theorem 5.11.

And (5.68) becomes

$$\cosh\theta_i = \frac{\bar{\lambda}_i}{2} \tag{2.124}$$
$$= \frac{k^2}{2}\lambda_i\left(T^{h,k}\right) \tag{2.125}$$
$$= -\frac{k^2}{h^2}\left(1 - \cos\frac{i\pi}{N+1}\right) - 1 \tag{2.126}$$

Equations (5.60)-(5.66) becomes

$$\left(A^{-1}\right)^{m,n} \tag{2.127}$$

$$= (-1)^{m+n} Q \tag{2.128}$$

$$\times diag_{N_x} \left\{ \frac{\sinh\left(\max\left\{m,n\right\}\theta_k\right)\sinh\left(\left(N_y+1-\min\left\{m,n\right\}\right)\theta_k\right)}{\sinh\theta_k \sinh\left(\left(N_y+1\right)\theta_k\right)} \right\} \tag{2.129}$$

$$\times Q^T k^2 I_{N_x} \tag{2.130}$$

$$= (-1)^{m+n} k^2 Q \tag{2.131}$$

$$\times diag_{N_x} \left\{ \frac{\sinh\left(\max\left\{m,n\right\}\theta_k\right)\sinh\left(\left(N_y+1-\min\left\{m,n\right\}\right)\theta_k\right)}{\sinh\theta_k \sinh\left(\left(N_y+1\right)\theta_k\right)} \right\} Q^T \tag{2.132}$$

and each element of the block $\left(A^{-1}\right)^{m,n}$ is

$$\left(A^{-1}\right)^{m,n}_{p,q} \tag{2.133}$$

$$= (-1)^{m+n} k^2 \tag{2.134}$$

$$\times \sum_{j=1}^{N_x} \left[Q diag_{N_x} \left\{ \times \frac{\sinh\left(\max\left\{m,n\right\}\theta_k\right)}{\sinh\theta_k \sinh\left(\left(N_y+1\right)\theta_k\right)} \sinh\left(\left(N_y+1-\min\left\{m,n\right\}\right)\theta_k\right) \right\} \right]_{pj} Q^T_{jq} \tag{2.135}$$

$$= (-1)^{m+n} k^2 \tag{2.136}$$

$$\times \sum_{j=1}^{N_x} \left(\sum_{l=1}^{N_x} Q_{pl} \left[diag_{N_x} \left\{ \times \frac{\sinh\left(\max\left\{m,n\right\}\theta_k\right)}{\sinh\theta_k \sinh\left(\left(N_y+1\right)\theta_k\right)} \sinh\left(\left(N_y+1-\min\left\{m,n\right\}\right)\theta_k\right) \right\} \right]_{lj} \right) Q_{qj} \tag{2.137}$$

$$= (-1)^{m+n} k^2 \tag{2.138}$$

$$\times \sum_{j=1}^{N_x} Q_{pj} \frac{\sinh\left(\max\left\{m,n\right\}\theta_j\right)\sinh\left(\left(N_y+1-\min\left\{m,n\right\}\right)\theta_j\right)}{\sinh\theta_j \sinh\left(\left(N_y+1\right)\theta_j\right)} Q_{qj} \tag{2.139}$$

$$= (-1)^{m+n} k^2 \frac{N_x+1}{2} \tag{2.140}$$

$$\times \sum_{j=1}^{N_x} \left(\begin{array}{l} \sin\dfrac{pj\pi}{N_x+1} \sin\dfrac{jq\pi}{N_x+1} \sinh\left(\max\left\{m,n\right\}\theta_j\right) \\ \times \dfrac{\sinh\left(\left(N_y+1-\min\left\{m,n\right\}\right)\theta_j\right)}{\sinh\theta_j \sinh\left(\left(N_y+1\right)\theta_j\right)} \end{array} \right) \tag{2.141}$$

$$= (-1)^{m+n} k^2 \frac{N_x+1}{2} \tag{2.142}$$

$$\times \sum_{j=1}^{N_x} \left(\begin{array}{l} \dfrac{e^{i\frac{pj\pi}{N_x+1}} - e^{-i\frac{pj\pi}{N_x+1}}}{2i} \dfrac{e^{i\frac{jq\pi}{N_x+1}} - e^{-i\frac{jq\pi}{N_x+1}}}{2i} \\ \times \dfrac{\frac{e^{\max\{m,n\}\theta_j} - e^{-\max\{m,n\}\theta_j}}{2}}{\frac{e^{\theta_j} - e^{-\theta_j}}{2} \frac{e^{(N_y+1)\theta_j} - e^{-(N_y+1)\theta_j}}{2}} \\ \times \dfrac{e^{(N_y+1-\min\{m,n\})\theta_j} - e^{-(N_y+1-\min\{m,n\})\theta_j}}{2} \end{array} \right) \tag{2.143}$$

This is the best formula we obtained. Because limited time, we can not continue to transform this complicated formula into easier one. We will revisit this formula and represent it as possible as we can at our project

Project FDM: Undergraduate Finite Difference Method

which will be uploaded in our blog soon. You should give it a try to continue our work.

2.5 Accuracy and Stability

The discretization of the two-dimensional Poisson problem can be analyzed using exactly the same approach as we used for the one-dimensional boundary value problem.

The local truncation error τ_{ij} at the (i,j) grid point is defined in the obvious way,

$$\tau_{ij} = \frac{1}{h^2}u\left(x_{i-1},y_j\right) + \frac{1}{h^2}u\left(x_{i+1},y_j\right) + \frac{1}{k^2}u\left(x_i,y_{j-1}\right) \tag{2.144}$$

$$+ \frac{1}{k^2}u\left(x_i,y_{j+1}\right) - 2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u\left(x_i,y_j\right) - f\left(x_i,y_j\right) \tag{2.145}$$

and by splitting this into the second order difference in the x- and y-directions by Taylor series expansion

$$u\left(x_{i-1},y_j\right) + u\left(x_{i+1},y_j\right) - 2u\left(x_i,y_j\right) \tag{2.146}$$

$$= h^2 u_{xx}\left(x_i,y_j\right) + \frac{1}{12}h^4 u_{xxxx}\left(x_i,y_j\right) + O\left(h^6\right) \tag{2.147}$$

$$u\left(x_i,y_{j-1}\right) + u\left(x_i,y_{j+1}\right) - 2u\left(x_i,y_j\right) \tag{2.148}$$

$$= k^2 u_{yy}\left(x_i,y_j\right) + \frac{1}{12}k^4 u_{yyyy}\left(x_i,y_j\right) + O\left(k^6\right) \tag{2.149}$$

for $i = 1,2,\ldots,N_x$ and $j = 1,2,\ldots,N_y$. Hence,

$$\tau_{ij} = u_{xx}\left(x_i,y_j\right) + \frac{1}{12}h^2 u_{xxxx}\left(x_i,y_j\right) + O\left(h^4\right) \tag{2.150}$$

$$+ u_{yy}\left(x_i,y_j\right) + \frac{1}{12}k^2 u_{yyyy}\left(x_i,y_j\right) + O\left(k^4\right) - f\left(x_i,y_j\right) \tag{2.151}$$

$$= \left(u_{xx}\left(x_i,y_j\right) + u_{yy}\left(x_i,y_j\right) - f\left(x_i,y_j\right)\right) \tag{2.152}$$

$$+ \frac{1}{12}\left(h^2 u_{xxxx}\left(x_i,y_j\right) + k^2 u_{yyyy}\left(x_i,y_j\right)\right) + O\left(h^4\right) + O\left(k^4\right) \tag{2.153}$$

$$= \frac{1}{12}\left(h^2 u_{xxxx}\left(x_i,y_j\right) + k^2 u_{yyyy}\left(x_i,y_j\right)\right) + O\left(h^4\right) + O\left(k^4\right) \tag{2.154}$$

$$= \frac{1}{12}\left(h^2 u_{xxxx}\left(x_i,y_j\right) + k^2 u_{yyyy}\left(x_i,y_j\right)\right) + O\left(H^4\right) \tag{2.155}$$

For this linear system of equations the global error

$$E_{ij} = u_{ij} - u\left(x_i,y_j\right) \tag{2.156}$$

then solves the linear system

$$A^{h,k}E^{h,k} = -\tau^{h,k} \tag{2.157}$$

just as in one dimension, where $A^{h,k}$ is now the discretization matrix with x-mesh spacing h and y-mesh spacing k, e.g., the matrix (2.20) if the rowwise ordering is used. The method will be globally second order accurate in some norm provided that it is stable, i.e., that $\left\|\left(A^{h,k}\right)^{-1}\right\|$ is uniformly bounded as $H \to 0$.

2.5.1 Stability in 2-Norm

Problem 2.13. *Prove that the finite difference method* (2.3) *is stable in 2-norm.*

SOLUTION. We now prepare to prove the stability of our finite difference method in 2 norm. To this end, we firstly compute explicitly eigenvalues and eigenvectors of A. See [13]. We recall that

$$T^{h,k} = Q\Lambda Q^T \tag{2.158}$$

where

$$\Lambda = diag_{N_x}\left\{\lambda_j\left(T^{h,k}\right)\right\} \tag{2.159}$$

be such a diagonalization for $T^{h,k}$. Then

$$A = \left(I_{N_y} \otimes Q\right)\widetilde{A}\left(I_{N_y} \otimes Q^T\right) \tag{2.160}$$

where $\widetilde{A}$ is the block-Toeplitz matrix obtained by replacing each T in A by Λ. Obviously, $\widetilde{A}$ is permutation-similar to

$$T_1 \oplus T_2 \oplus \cdots \oplus T_{N_y} \tag{2.161}$$

where each diagonal sub-block T_i is the $N_x \times N_x$ symmetric Toeplitz matrix

$$T_i = \begin{bmatrix} \lambda_i & \frac{1}{k^2} & & & \\ \frac{1}{k^2} & \lambda_i & \frac{1}{k^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{k^2} & \lambda_i & \frac{1}{k^2} \\ & & & \frac{1}{k^2} & \lambda_i \end{bmatrix} \tag{2.162}$$

and $\lambda_1, \ldots, \lambda_{N_x}$ are the eigenvalues of $T^{h,k}$. In fact, if $K^{(N_y,N_x)}$ denotes the commutation matrix that maps $vec\,(A)$ to $vec\left(A^T\right)$ for a generic A, then

$$A = \left(I_{N_y} \otimes Q\right)K^{(N_y,N_x)}\left(T_1 \oplus T_2 \oplus \cdots \oplus T^{h,k}\right)K^{(N_x,N_y)}\left(I_{N_y} \otimes Q^T\right) \tag{2.163}$$

Now, for each i, using the formula for symmetric Toeplitz matrix, we see that the eigenvalue matrix $\Lambda_i = diag\left(\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{iN_y}\right)$ of T_i is given by the formula

$$\lambda_{ij} = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + \frac{2}{h^2}\cos\left(\frac{i\pi}{N_x + 1}\right) + \frac{2}{k^2}\cos\left(\frac{j\pi}{N_y + 1}\right) \tag{2.164}$$

for $j = 1, 2, \ldots, N_y$. Let $T_i = Q_i \Lambda_i Q_i^T$ be an orthogonal diagonalization. It follows that

$$P = \left(I_{N_y} \otimes Q\right) K^{(N_y, N_x)} \left(Q_1 \oplus Q_2 \oplus \cdots \oplus Q_{N_y}\right) \tag{2.165}$$

and orthogonal diagonalization of A is given by

$$A = P \left(\Lambda_1 \oplus \Lambda_2 \oplus \cdots \oplus \Lambda_{N_x}\right) P^T \tag{2.166}$$

That is, the eigenvalues of A are those λ_{ij}'s for $i = 1, 2, \ldots, N_x$ and $j = 1, 2, \ldots, N_y$ and their corresponding eigenvectors are the columns of P.

Thus, we have the precise formulas of eigenvalues of A. Now we can explicitly compute the spectral radius of the matrix. The eigenvalues are strictly negative

$$\lambda_{ij} = -2\left(\frac{1}{h^2} + \frac{1}{k^2}\right) + \frac{2}{h^2}\cos\left(\frac{i\pi}{N_x + 1}\right) + \frac{2}{k^2}\cos\left(\frac{j\pi}{N_y + 1}\right) \tag{2.167}$$

$$= \frac{2}{h^2}\left(\cos\left(\frac{i\pi}{N_x + 1}\right) - 1\right) + \frac{2}{k^2}\left(\cos\left(\frac{j\pi}{N_y + 1}\right) - 1\right) \tag{2.168}$$

$$< 0, \quad i = 1, 2, \ldots, N_x, \quad j = 1, 2, \ldots, N_y \tag{2.169}$$

i.e., A is negative definite, and the closest to the origin is

$$\lambda_{11} = \frac{2}{h^2}\left(\cos\left(\frac{\pi}{N_x + 1}\right) - 1\right) + \frac{2}{k^2}\left(\cos\left(\frac{\pi}{N_y + 1}\right) - 1\right) \tag{2.170}$$

$$= \frac{2}{h^2}\left(\cos\left(\pi h\right) - 1\right) + \frac{2}{k^2}\left(\cos\left(\pi k\right) - 1\right) \tag{2.171}$$

$$= \frac{2}{h^2}\left(-\frac{\pi^2 h^2}{2} + O\left(h^4\right)\right) + \frac{2}{k^2}\left(-\frac{\pi^2 k^2}{2} + O\left(k^4\right)\right) \tag{2.172}$$

$$= -2\pi^2 + O\left(h^2\right) + O\left(k^2\right) \tag{2.173}$$

$$= -2\pi^2 + O\left(H^2\right) \tag{2.174}$$

Noticing that A is symmetric, the spectral radius of $\left(A^{h,k}\right)^{-1}$, which is also the 2-norm, is thus

$$\left\|\left(A^{h,k}\right)^{-1}\right\|_2 = \rho\left(\left(A^{h,k}\right)^{-1}\right) \tag{2.175}$$

$$= \max_{1 \le i \le N_x, 1 \le j \le N_y} \frac{1}{|\lambda_{i,j}|} \tag{2.176}$$

$$= \frac{1}{\min_{1 \le i \le N_x, 1 \le j \le N_y} |\lambda_{i,j}|} \tag{2.177}$$

$$= \frac{1}{|\lambda_{11}|} \tag{2.178}$$

$$= \frac{1}{2\pi^2 + O\left(H^2\right)} \tag{2.179}$$

$$\approx \frac{1}{2\pi^2} \tag{2.180}$$

Hence, the method is stable in the 2-norm. □

We also compute the *condition number* of the matrix $A^{h,k}$, since it turns out that this is a critical quantity in determining how rapidly certain iterative methods converge. Recall that the 2-norm condition number is defined by

$$\kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 \tag{2.181}$$

Problem 2.14. *Compute the 2-norm condition number of the matrix $A^{h,k}$.*

SOLUTION. We have just seen that

$$\left\|\left(A^{h,k}\right)^{-1}\right\|_2 \approx \frac{1}{2\pi^2} \tag{2.182}$$

for small h, and the norm of A is given by its spectral radius. The largest eigenvalue of A (in magnitude) is

$$\begin{aligned}
\lambda_{N_x N_y} &= \frac{2}{h^2}\left(\cos\left(\frac{N_x\pi}{N_x+1}\right)-1\right)+\frac{2}{k^2}\left(\cos\left(\frac{N_y\pi}{N_y+1}\right)-1\right) && (2.183)\\
&= \frac{2}{h^2}\left(\cos\left(N_x\pi h\right)-1\right)+\frac{2}{k^2}\left(\cos\left(N_y\pi h\right)-1\right) && (2.184)\\
&= \frac{2}{h^2}\left(-\frac{N_x^2\pi^2h^2}{2}+O\left(h^4\right)\right)+\frac{2}{k^2}\left(-\frac{N_y^2\pi^2k^2}{2}+O\left(k^4\right)\right) && (2.185)\\
&= -N_x^2\pi^2-N_y^2\pi^2+O\left(h^2\right)+O\left(k^2\right) && (2.186)\\
&= -\left(N_x^2+N_y^2\right)\pi^2+O\left(H^2\right) && (2.187)\\
&= -\left(\left(\frac{1}{h}-1\right)^2+\left(\frac{1}{k}-1\right)^2\right)\pi^2+O\left(H^2\right) && (2.188)
\end{aligned}$$

Hence,

$$\begin{aligned}
\kappa_2\left(A^{h,k}\right) &= \left\|A^{h,k}\right\|_2\left\|\left(A^{h,k}\right)^{-1}\right\|_2 && (2.189)\\
&\approx \frac{1}{2\pi^2}\left(\left(\frac{1}{h}-1\right)^2+\left(\frac{1}{k}-1\right)^2\right)\pi^2 && (2.190)\\
&= \frac{1}{2}\left(\frac{1}{h^2}-\frac{2}{h}+1+\frac{1}{k^2}-\frac{2}{k}+1\right) && (2.191)\\
&\leq \frac{1}{2}\left(\frac{1}{h^2}-\frac{2}{h^2}+\frac{1}{h^2}+\frac{1}{k^2}-\frac{2}{k^2}+\frac{1}{k^2}\right) && (2.192)\\
&= O\left(\frac{1}{h^2}\right)+O\left(\frac{1}{k^2}\right) && (2.193)\\
&= O\left(\frac{1}{H^2}\right) \text{ as } H\to 0 && (2.194)
\end{aligned}$$

Done. □

The fact that the matrix becomes very ill-conditioned as we refine the grid is responsible for the slow-down of iterative methods.

2.5.2 Stability in Other Norms

The reader should try to use (2.133)-(2.143) to prove the stability of our finite difference method in other norm.

Problem 2.15 (Stability). *Prove the finite difference method* (2.3) *is stable in*

1. *1-norm* $\|\cdot\|_1$.
2. *Infinity norm* $\|\cdot\|_\infty$.
3. *Frobenius norm* $\|\cdot\|_F$.

See [2] for references.

2.6 The 9-Point Laplacian

Above we used the 5-point Laplacian, which we will denote by

$$\widehat{\nabla}_5^2 u_{ij} := \frac{1}{h^2}u_{i-1,j} + \frac{1}{h^2}u_{i+1,j} + \frac{1}{k^2}u_{i,j-1} + \frac{1}{k^2}u_{i,j+1} - 2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{ij} \tag{2.195}$$

Another possible approximation is the 9-point Laplacian. We now establish an explicit formula for the 9-point Laplacian.

Problem 2.16. *Establish an explicit formula for the 9-point Laplacian.*

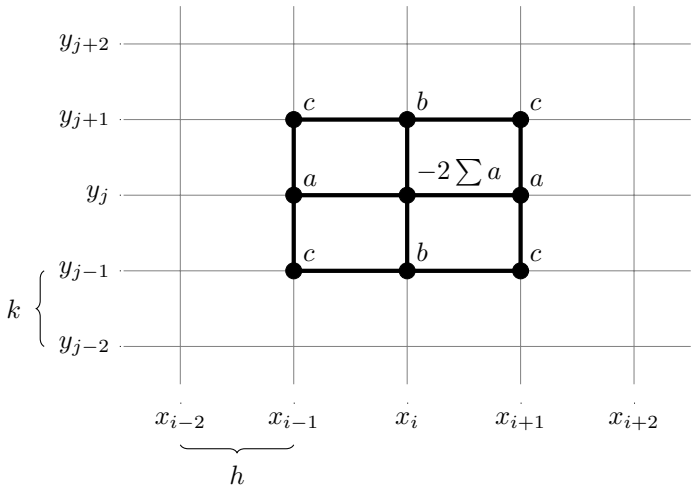

Figure 4: The red-black order of unknowns and equations on a 4×4 grid.

Solution. To this ends, we begin with Taylor series expansion as usual but we have to use multi-dimensional form of Taylor series expansion at this moment.

$$u(x_{i-1}, y_{j-1}) + u(x_{i+1}, y_{j+1}) - 2u(x_i, y_j) \tag{2.196}$$
$$= h^2 u_{xx}(x_i, y_j) + 2hku_{xy}(x_i, y_j) + k^2 u_{yy}(x_i, y_j) \tag{2.197}$$

$$+\frac{1}{12}h^4u_{xxxx}(x_i,y_j)+\frac{1}{3}h^3ku_{xxxy}(x_i,y_j)+\frac{1}{2}h^2k^2u_{xxyy}(x_i,y_j) \tag{2.198}$$
$$+\frac{1}{3}hk^3u_{xyyy}(x_i,y_j)+\frac{1}{12}k^4u_{yyyy}(x_i,y_j)+O\left(H^6\right) \tag{2.199}$$

and

$$u(x_{i-1},y_{j+1})+u(x_{i+1},y_{j-1})-2u(x_i,y_j) \tag{2.200}$$
$$=h^2u_{xx}(x_i,y_j)-2hku_{xy}(x_i,y_j)+k^2u_{yy}(x_i,y_j) \tag{2.201}$$
$$+\frac{1}{12}h^4u_{xxxx}(x_i,y_j)-\frac{1}{3}h^3ku_{xxxy}(x_i,y_j)+\frac{1}{2}h^2k^2u_{xxyy}(x_i,y_j) \tag{2.202}$$
$$-\frac{1}{3}hk^3u_{xyyy}(x_i,y_j)+\frac{1}{12}k^4u_{yyyy}(x_i,y_j)+O\left(H^6\right) \tag{2.203}$$

Combining (2.146)-(2.149) and (2.196)-(2.203), we consider a linear combination of the left-hand sides of these formulas with the following form

$$a(u(x_{i-1},y_j)+u(x_{i+1},y_j)-2u(x_i,y_j)) \tag{2.204}$$
$$+b(u(x_i,y_{j-1})+u(x_i,y_{j+1})-2u(x_i,y_j)) \tag{2.205}$$
$$+c(u(x_{i-1},y_{j-1})+u(x_{i+1},y_{j+1})-2u(x_i,y_j)) \tag{2.206}$$
$$+d(u(x_{i-1},y_{j+1})+u(x_{i+1},y_{j-1})-2u(x_i,y_j)) \tag{2.207}$$

A straightforward computations yields

$$a(u(x_{i-1},y_j)+u(x_{i+1},y_j)-2u(x_i,y_j)) \tag{2.208}$$
$$+b(u(x_i,y_{j-1})+u(x_i,y_{j+1})-2u(x_i,y_j)) \tag{2.209}$$
$$+c(u(x_{i-1},y_{j-1})+u(x_{i+1},y_{j+1})-2u(x_i,y_j)) \tag{2.210}$$
$$+d(u(x_{i-1},y_{j+1})+u(x_{i+1},y_{j-1})-2u(x_i,y_j)) \tag{2.211}$$
$$=a\left(h^2u_{xx}(x_i,y_j)+\frac{1}{12}h^4u_{xxxx}(x_i,y_j)+O\left(H^6\right)\right) \tag{2.212}$$
$$+b\left(k^2u_{yy}(x_i,y_j)+\frac{1}{12}k^4u_{yyyy}(x_i,y_j)+O\left(H^6\right)\right) \tag{2.213}$$
$$+c\left(\begin{array}{l}h^2u_{xx}(x_i,y_j)+2hku_{xy}(x_i,y_j)+k^2u_{yy}(x_i,y_j)\\+\frac{1}{12}h^4u_{xxxx}(x_i,y_j)+\frac{1}{3}h^3ku_{xxxy}(x_i,y_j)\\+\frac{1}{2}h^2k^2u_{xxyy}(x_i,y_j)+\frac{1}{3}hk^3u_{xyyy}(x_i,y_j)\\+\frac{1}{12}k^4u_{yyyy}(x_i,y_j)+O\left(H^6\right)\end{array}\right) \tag{2.214}$$
$$+d\left(\begin{array}{l}h^2u_{xx}(x_i,y_j)-2hku_{xy}(x_i,y_j)+k^2u_{yy}(x_i,y_j)\\+\frac{1}{12}h^4u_{xxxx}(x_i,y_j)-\frac{1}{3}h^3ku_{xxxy}(x_i,y_j)\\+\frac{1}{2}h^2k^2u_{xxyy}(x_i,y_j)-\frac{1}{3}hk^3u_{xyyy}(x_i,y_j)\\+\frac{1}{12}k^4u_{yyyy}(x_i,y_j)+O\left(H^6\right)\end{array}\right) \tag{2.215}$$
$$=(a+c+d)h^2u_{xx}(x_i,y_j)+2(c-d)hku_{xy}(x_i,y_j) \tag{2.216}$$
$$+(b+c+d)k^2u_{yy}(x_i,y_j)+\frac{1}{12}(a+c+d)h^4u_{xxxx}(x_i,y_j) \tag{2.217}$$
$$+\frac{1}{3}(c-d)h^3ku_{xxxy}(x_i,y_j)+\frac{1}{2}(c+d)h^2k^2u_{xxyy}(x_i,y_j) \tag{2.218}$$

$$+\frac{1}{3}(c-d)\,hk^3u_{xyyy}(x_i,y_j)+\frac{1}{12}(b+c+d)\,k^4u_{yyyy}(x_i,y_j) \tag{2.219}$$
$$+(a+b+c+d)\,O\left(H^6\right) \tag{2.220}$$

We want the coefficients of $u_{xy}(x_i,y_j)$, $u_{xxxy}(x_i,y_j)$ and $u_{xyyy}(x_i,y_j)$ equal to zero. This can be satisfied by taking $c=d$. Under taking $c=d$, we obtain

$$a\left(u(x_{i-1},y_j)+u(x_{i+1},y_j)-2u(x_i,y_j)\right) \tag{2.221}$$
$$+b\left(u(x_i,y_{j-1})+u(x_i,y_{j+1})-2u(x_i,y_j)\right) \tag{2.222}$$
$$+c\left(u(x_{i-1},y_{j-1})+u(x_{i+1},y_{j+1})-2u(x_i,y_j)\right) \tag{2.223}$$
$$+c\left(u(x_{i-1},y_{j+1})+u(x_{i+1},y_{j-1})-2u(x_i,y_j)\right) \tag{2.224}$$
$$=(a+2c)\,h^2u_{xx}(x_i,y_j)+(b+2c)\,k^2u_{yy}(x_i,y_j) \tag{2.225}$$
$$+\frac{1}{12}(a+2c)\,h^4u_{xxxx}(x_i,y_j)+ch^2k^2u_{xxyy}(x_i,y_j) \tag{2.226}$$
$$+\frac{1}{12}(b+2c)\,k^4u_{yyyy}(x_i,y_j)+(a+b+2c)\,O\left(H^6\right) \tag{2.227}$$

Next, we want both the coefficient of $u_{xx}(x_i,y_j)$ and the coefficients of $u_{yy}(x_i,y_j)$ equal to 1, so that

$$a\left(u(x_{i-1},y_j)+u(x_{i+1},y_j)-2u(x_i,y_j)\right) \tag{2.228}$$
$$+b\left(u(x_i,y_{j-1})+u(x_i,y_{j+1})-2u(x_i,y_j)\right) \tag{2.229}$$
$$+c\left(u(x_{i-1},y_{j-1})+u(x_{i+1},y_{j+1})-2u(x_i,y_j)\right) \tag{2.230}$$
$$+c\left(u(x_{i-1},y_{j+1})+u(x_{i+1},y_{j-1})-2u(x_i,y_j)\right) \tag{2.231}$$
$$-f(x_i,y_j) \tag{2.232}$$

works, i.e., the term $\Delta u(x_i,y_j)-f(x_i,y_j)$ in this expression will be canceled. This gives

$$(a+2c)\,h^2=(b+2c)\,k^2=1 \tag{2.233}$$

We now consider two cases with respect to h and k.

1. Case $h=k$. This gives square uniform mesh.
2. Case $h\neq k$. This give rectangular uniform mesh which is not square uniform.

2.6.1 Square Uniform Mesh

Under the assumption $h=k$, we have $H=h$. Our rectangular uniform mesh becomes square uniform mesh in this case, i.e., the most uniform one in two dimensional spaces, and (2.233) becomes

$$a+2c=b+2c=\frac{1}{h^2} \tag{2.234}$$

By (2.234), we deduce that

$$a=b=\frac{1}{h^2}-2c \tag{2.235}$$

In the terms of c, our tuple of coefficients becomes

$$(a, b, c, d) = \left(\frac{1}{h^2} - 2c, \frac{1}{h^2} - 2c, c, c\right) \tag{2.236}$$

and the luckiest thing in this case is that the coefficient of $u_{xxxx}(x_i, y_j)$ also equal to the coefficient of $u_{yyyy}(x_i, y_j)$. Under these, (2.221)-(2.233) becomes

$$\left(\frac{1}{h^2} - 2c\right)(u(x_{i-1}, y_j) + u(x_{i+1}, y_j) - 2u(x_i, y_j)) \tag{2.237}$$
$$+ \left(\frac{1}{h^2} - 2c\right)(u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 2u(x_i, y_j)) \tag{2.238}$$
$$+ c(u(x_{i-1}, y_{j-1}) + u(x_{i+1}, y_{j+1}) - 2u(x_i, y_j)) \tag{2.239}$$
$$+ c(u(x_{i-1}, y_{j+1}) + u(x_{i+1}, y_{j-1}) - 2u(x_i, y_j)) \tag{2.240}$$
$$= u_{xx}(x_i, y_j) + u_{yy}(x_i, y_j) + \frac{1}{12}h^2 u_{xxxx}(x_i, y_j) \tag{2.241}$$
$$+ ch^4 u_{xxyy}(x_i, y_j) + \frac{1}{12}h^2 u_{yyyy}(x_i, y_j) + O\left(H^6\right) \tag{2.242}$$
$$= \Delta u(x_i, y_j) \tag{2.243}$$
$$+ \frac{1}{12}h^2\left(u_{xxxx}(x_i, y_j) + 12ch^2 u_{xxyy}(x_i, y_j) + u_{yyyy}(x_i, y_j)\right) \tag{2.244}$$
$$+ \left(\frac{2}{h^2} - 2c\right) O\left(h^6\right) \tag{2.245}$$

We have just obtained an general formula for 9-point stencil Laplacian in terms of c. You can choose c as arbitrary as you want. Since (2.237)-(2.245) always give the $O\left(h^2\right)$ error, which the 5-point discretization also gives, with an arbitrarily chosen c. Is this general c 9-point not better than the 5-point discretization?

No, we still have a little hope on choosing c. What is the best choice of c. At this moment, we recall

$$\Delta^2 u \equiv \Delta(\Delta u) \tag{2.246}$$
$$= u_{xxxx} + 2u_{xxyy} + u_{yyyy} \tag{2.247}$$

This is the Laplacian of the Laplacian of u and Δ^2 is called the *biharmonic operator*.

Return to our problem, we now try to choose c so that (2.244) contains biharmonic operator. It is straightforward to choose the coefficient of $u_{xxyy}(x_i, y_j)$ as

$$12ch^2 = 2 \tag{2.248}$$

i.e.,

$$c = \frac{1}{6h^2} \tag{2.249}$$

Under this choice of c, (2.237)-(2.245) becomes

$$\frac{2}{3h^2}(u(x_{i-1}, y_j) + u(x_{i+1}, y_j) - 2u(x_i, y_j)) \tag{2.250}$$

$$+\frac{2}{3h^2}\left(u\left(x_i,y_{j-1}\right)+u\left(x_i,y_{j+1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.251}$$

$$+\frac{1}{6h^2}\left(u\left(x_{i-1},y_{j-1}\right)+u\left(x_{i+1},y_{j+1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.252}$$

$$+\frac{1}{6h^2}\left(u\left(x_{i-1},y_{j+1}\right)+u\left(x_{i+1},y_{j-1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.253}$$

$$=\Delta u\left(x_i,y_j\right)+\frac{1}{12}h^2\Delta^2 u\left(x_i,y_j\right)+O\left(h^4\right) \tag{2.254}$$

We now define the 9-point Laplacian as

$$\nabla_9^2 u_{ij}:=\frac{1}{6h^2}\left(\begin{array}{l}4u_{i-1,j}+4u_{i+1,j}+4u_{i,j-1}+4u_{i,j+1}+u_{i-1,j-1}\\+u_{i-1,j+1}+u_{i+1,j-1}+u_{i+1,j+1}-20u_{ij}\end{array}\right) \tag{2.255}$$

We will continue this in Subsection 3.5. □

2.6.2 Rectangular Uniform Mesh

Under the assumption $h\neq k$, (2.233) becomes

$$a+2c=\frac{1}{h^2} \tag{2.256}$$

$$b+2c=\frac{1}{k^2} \tag{2.257}$$

In terms of c, our tuple of coefficients becomes

$$a=\frac{1}{h^2}-2c \tag{2.258}$$

$$b=\frac{1}{k^2}-2c \tag{2.259}$$

$$d=c \tag{2.260}$$

and the luckiness in the previous square uniform mesh $h=k$ have gone away, i.e., the coefficient of $u_{xxxx}\left(x_i,y_j\right)$ is different from the coefficient of $u_{yyyy}\left(x_i,y_j\right)$. Under (2.260), (2.221)-(2.233) becomes

$$\left(\frac{1}{h^2}-2c\right)\left(u\left(x_{i-1},y_j\right)+u\left(x_{i+1},y_j\right)-2u\left(x_i,y_j\right)\right) \tag{2.261}$$

$$+\left(\frac{1}{k^2}-2c\right)\left(u\left(x_i,y_{j-1}\right)+u\left(x_i,y_{j+1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.262}$$

$$+c\left(u\left(x_{i-1},y_{j-1}\right)+u\left(x_{i+1},y_{j+1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.263}$$

$$+c\left(u\left(x_{i-1},y_{j+1}\right)+u\left(x_{i+1},y_{j-1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.264}$$

$$=u_{xx}\left(x_i,y_j\right)+u_{yy}\left(x_i,y_j\right)+\frac{1}{12}h^2u_{xxxx}\left(x_i,y_j\right) \tag{2.265}$$

$$+ch^2k^2u_{xxyy}\left(x_i,y_j\right)+\frac{1}{12}k^2u_{yyyy}\left(x_i,y_j\right) \tag{2.266}$$

$$+\left(\frac{1}{h^2}+\frac{1}{k^2}-2c\right)O\left(H^6\right) \tag{2.267}$$

$$=\Delta u\left(x_i,y_j\right)+\frac{1}{12}h^2u_{xxxx}\left(x_i,y_j\right)+ch^2k^2u_{xxyy}\left(x_i,y_j\right) \tag{2.268}$$

$$+\frac{1}{12}k^2u_{yyyy}\left(x_i,y_j\right)+\left(\frac{1}{h^2}+\frac{1}{k^2}-2c\right)O\left(H^6\right) \tag{2.269}$$

We now focus on the term

$$h^2u_{xxxx}\left(x_i,y_j\right)+12ch^2k^2u_{xxyy}\left(x_i,y_j\right)+k^2u_{yyyy}\left(x_i,y_j\right) \tag{2.270}$$

We need to transform (2.270) into an expression in terms of

$$\Delta f=\Delta^2u \tag{2.271}$$
$$=u_{xxxx}+2u_{xxyy}+u_{yyyy} \tag{2.272}$$

so that it can computed by using known function f.

To this ends, we consider two changes of variables $\alpha=\alpha\left(x,y\right),\beta=\beta\left(x,y\right)$ satisfying $\alpha,\beta\in C^1\left(\mathbb{R}^2\right)$ and

$$\det\left(J_{\alpha,\beta}\left(x,y\right)\right)=\det\begin{pmatrix}\alpha_x & \alpha_y\\ \beta_x & \beta_y\end{pmatrix} \tag{2.273}$$
$$=\alpha_x\beta_y-\alpha_y\beta_x\neq 0 \tag{2.274}$$

Applying the chain rule four times gives

First Order Partial Derivatives.

$$u_x=u_\alpha\alpha_x+u_\beta\beta_x \tag{2.275}$$
$$u_y=u_\alpha\alpha_y+u_\beta\beta_y \tag{2.276}$$

Second Order Partial Derivatives.

$$u_{xx}=u_{\alpha\alpha}\alpha_x^2+u_{\beta\beta}\beta_x^2+2u_{\alpha\beta}\alpha_x\beta_x+u_\alpha\alpha_{xx}+u_\beta\beta_{xx} \tag{2.277}$$
$$u_{xy}=u_{\alpha\alpha}\alpha_x\alpha_y+u_{\beta\beta}\beta_x\beta_y+u_{\alpha\beta}\alpha_x\beta_y+u_{\beta\alpha}\alpha_y\beta_x+u_\alpha\alpha_{xy}+u_\beta\beta_{xy} \tag{2.278}$$
$$u_{yy}=u_{\alpha\alpha}\alpha_y^2+u_{\beta\beta}\beta_y^2+2u_{\alpha\beta}\alpha_y\beta_y+u_\alpha\alpha_{yy}+u_\beta\beta_{yy} \tag{2.279}$$

Third Order Partial Derivatives.

$$u_{xxx}=u_{\alpha\alpha\alpha}\alpha_x^3+3u_{\alpha\alpha\beta}\alpha_x^2\beta_x+3u_{\alpha\beta\beta}\alpha_x\beta_x^2+u_{\beta\beta\beta}\beta_x^3+3u_{\alpha\alpha}\alpha_x\alpha_{xx} \tag{2.280}$$
$$+3u_{\alpha\beta}\alpha_{xx}\beta_x+3u_{\alpha\beta}\alpha_x\beta_{xx}+3u_{\beta\beta}\beta_x\beta_{xx}+u_\alpha\alpha_{xxx}+u_\beta\beta_{xxx} \tag{2.281}$$
$$u_{xxy}=u_{\alpha\alpha\alpha}\alpha_x^2\alpha_y+u_{\alpha\alpha\beta}\alpha_x^2\beta_y+2u_{\alpha\alpha\beta}\alpha_x\alpha_y\beta_x+u_{\alpha\beta\beta}\alpha_y\beta_x^2 \tag{2.282}$$
$$+2u_{\alpha\beta\beta}\alpha_x\beta_x\beta_y+u_{\beta\beta\beta}\beta_x^2\beta_y+2u_{\alpha\alpha}\alpha_x\alpha_{xy}+u_{\alpha\alpha}\alpha_y\alpha_{xx} \tag{2.283}$$
$$+2u_{\alpha\beta}\alpha_{xy}\beta_x+2u_{\alpha\beta}\alpha_x\beta_{xy}+u_{\alpha\beta}\alpha_{xx}\beta_y+u_{\alpha\beta}\alpha_y\beta_{xx} \tag{2.284}$$
$$+2u_{\beta\beta}\beta_x\beta_{xy}+u_{\beta\beta}\beta_y\beta_{xx}+u_\alpha\alpha_{xxy}+u_\beta\beta_{xxy} \tag{2.285}$$
$$u_{xyy}=u_{\alpha\alpha\alpha}\alpha_x\alpha_y^2+u_{\alpha\alpha\beta}\beta_x\alpha_y^2+2u_{\alpha\alpha\beta}\alpha_x\alpha_y\beta_y+u_{\alpha\beta\beta}\alpha_x\beta_y^2 \tag{2.286}$$
$$+2u_{\alpha\beta\beta}\alpha_y\beta_x\beta_y+u_{\beta\beta\beta}\beta_x\beta_y^2+2u_{\alpha\alpha}\alpha_y\alpha_{xy}+u_{\alpha\alpha}\alpha_x\alpha_{yy} \tag{2.287}$$
$$+2u_{\alpha\beta}\alpha_{xy}\beta_y+2u_{\alpha\beta}\alpha_y\beta_{xy}+u_{\alpha\beta}\alpha_{yy}\beta_x+u_{\alpha\beta}\alpha_x\beta_{yy} \tag{2.288}$$
$$+2u_{\beta\beta}\beta_y\beta_{xy}+u_{\beta\beta}\beta_x\beta_{yy}+u_\alpha\alpha_{xyy}+u_\beta\beta_{xyy} \tag{2.289}$$
$$u_{yyy}=u_{\alpha\alpha\alpha}\alpha_y^3+3u_{\alpha\alpha\beta}\alpha_y^2\beta_y+3u_{\alpha\beta\beta}\alpha_y\beta_y^2+u_{\beta\beta\beta}\beta_y^3+3u_{\alpha\alpha}\alpha_y\alpha_{yy} \tag{2.290}$$
$$+3u_{\alpha\beta}\alpha_{yy}\beta_y+3u_{\alpha\beta}\alpha_y\beta_{yy}+3u_{\beta\beta}\beta_y\beta_{yy}+u_\alpha\alpha_{yyy}+u_\beta\beta_{yyy} \tag{2.291}$$

Fourth Order Partial Derivatives.

$$u_{xxxx} = u_{\alpha\alpha\alpha\alpha}\alpha_x^4 + 4u_{\alpha\alpha\alpha\beta}\alpha_x^3\beta_x + 6u_{\alpha\alpha\beta\beta}\alpha_x^2\beta_x^2 + 4u_{\alpha\beta\beta\beta}\alpha_x\beta_x^3 \tag{2.292}$$
$$+ u_{\beta\beta\beta\beta}\beta_x^4 + 6u_{\alpha\alpha\alpha}\alpha_x^2\alpha_{xx} + 12u_{\alpha\alpha\beta}\alpha_x\alpha_{xx}\beta_x + 6u_{\alpha\alpha\beta}\alpha_x^2\beta_{xx} \tag{2.293}$$
$$+ 12u_{\alpha\beta\beta}\alpha_x\beta_x\beta_{xx} + 6u_{\alpha\beta\beta}\alpha_{xx}\beta_x^2 + 6u_{\beta\beta\beta}\beta_x^2\beta_{xx} + 3u_{\alpha\alpha}\alpha_{xx}^2 \tag{2.294}$$
$$+ 4u_{\alpha\alpha}\alpha_x\alpha_{xxx} + 4u_{\alpha\beta}\alpha_{xxx}\beta_x + 6u_{\alpha\beta}\alpha_{xx}\beta_{xx} + 4u_{\alpha\beta}\alpha_x\beta_{xxx} \tag{2.295}$$
$$+ 3u_{\beta\beta}\beta_{xx}^2 + 4u_{\beta\beta}\beta_x\beta_{xxx} + u_\alpha\alpha_{xxxx} + u_\beta\beta_{xxxx} \tag{2.296}$$
$$u_{yyyy} = u_{\alpha\alpha\alpha\alpha}\alpha_y^4 + 4u_{\alpha\alpha\alpha\beta}\alpha_y^3\beta_y + 6u_{\alpha\alpha\beta\beta}\alpha_y^2\beta_y^2 + 4u_{\alpha\beta\beta\beta}\alpha_y\beta_y^3 \tag{2.297}$$
$$+ u_{\beta\beta\beta\beta}\beta_y^4 + 6u_{\alpha\alpha\alpha}\alpha_y^2\alpha_{yy} + 12u_{\alpha\alpha\beta}\alpha_y\alpha_{yy}\beta_y + 6u_{\alpha\alpha\beta}\alpha_y^2\beta_{yy} \tag{2.298}$$
$$+ 12u_{\alpha\beta\beta}\alpha_y\beta_y\beta_{yy} + 6u_{\alpha\beta\beta}\alpha_{yy}\beta_y^2 + 6u_{\beta\beta\beta}\beta_y^2\beta_{yy} + 3u_{\alpha\alpha}\alpha_{yy}^2 \tag{2.299}$$
$$+ 4u_{\alpha\alpha}\alpha_y\alpha_{yyy} + 4u_{\alpha\beta}\alpha_{yyy}\beta_y + 6u_{\alpha\beta}\alpha_{yy}\beta_{yy} + 4u_{\alpha\beta}\alpha_y\beta_{yyy} \tag{2.300}$$
$$+ 3u_{\beta\beta}\beta_{yy}^2 + 4u_{\beta\beta}\beta_y\beta_{yyy} + u_\alpha\alpha_{yyyy} + u_\beta\beta_{yyyy} \tag{2.301}$$
$$u_{xxxy} = u_{\alpha\alpha\alpha\alpha}\alpha_x^3\alpha_y + u_{\alpha\alpha\alpha\beta}\alpha_x^3\beta_y + 3u_{\alpha\alpha\alpha\beta}\alpha_x^2\alpha_y\beta_x + 3u_{\alpha\alpha\beta\beta}\alpha_x^2\beta_x\beta_y \tag{2.302}$$
$$+ 3u_{\alpha\alpha\beta\beta}\alpha_x\alpha_y\beta_x^2 + u_{\alpha\beta\beta\beta}\alpha_y\beta_x^3 + 3u_{\alpha\beta\beta\beta}\alpha_x\beta_x^2\beta_y \tag{2.303}$$
$$+ u_{\beta\beta\beta\beta}\beta_x^3\beta_y + 3u_{\alpha\alpha\alpha}\alpha_x^2\alpha_{xy} + 3u_{\alpha\alpha\alpha}\alpha_x\alpha_{xx}\alpha_y \tag{2.304}$$
$$+ 6u_{\alpha\alpha\beta}\alpha_x\alpha_{xy}\beta_x + 3u_{\alpha\alpha\beta}\alpha_x^2\beta_{xy} + 3u_{\alpha\alpha\beta}\alpha_x\alpha_{xx}\beta_y \tag{2.305}$$
$$+ 3u_{\alpha\alpha\beta}\alpha_y\alpha_{xx}\beta_x + 3u_{\alpha\alpha\beta}\alpha_x\alpha_y\beta_{xx} + 3u_{\alpha\beta\beta}\alpha_{xy}\beta_x^2 \tag{2.306}$$
$$+ 6u_{\alpha\beta\beta}\alpha_x\beta_x\beta_{xy} + 3u_{\alpha\beta\beta}\alpha_{xx}\beta_x\beta_y + 3u_{\alpha\beta\beta}\alpha_x\beta_{xx}\beta_y \tag{2.307}$$
$$+ 3u_{\alpha\beta\beta}\alpha_y\beta_x\beta_{xx} + 3u_{\beta\beta\beta}\beta_x^2\beta_{xy} + 3u_{\beta\beta\beta}\beta_x\beta_{xx}\beta_y \tag{2.308}$$
$$+ 3u_{\alpha\alpha}\alpha_{xy}\alpha_{xx} + 3u_{\alpha\alpha}\alpha_x\alpha_{xxy} + u_{\alpha\alpha}\alpha_{xxx}\alpha_y + 3u_{\alpha\beta}\alpha_{xxy}\beta_x \tag{2.309}$$
$$+ 3u_{\alpha\beta}\alpha_{xx}\beta_{xy} + 3u_{\alpha\beta}\alpha_{xy}\beta_{xx} + 3u_{\alpha\beta}\alpha_x\beta_{xxy} + u_{\alpha\beta}\alpha_{xxx}\beta_y \tag{2.310}$$
$$+ u_{\alpha\beta}\alpha_y\beta_{xxx} + 3u_{\beta\beta}\beta_{xx}\beta_{xy} + 3u_{\beta\beta}\beta_x\beta_{xxy} + u_{\beta\beta}\beta_{xxx}\beta_y \tag{2.311}$$
$$+ u_\alpha\alpha_{xxxy} + u_\beta\beta_{xxxy} \tag{2.312}$$
$$u_{xyyy} = u_{\alpha\alpha\alpha\alpha}\alpha_y^3\alpha_x + u_{\alpha\alpha\alpha\beta}\alpha_y^3\beta_x + 3u_{\alpha\alpha\alpha\beta}\alpha_y^2\alpha_x\beta_y + 3u_{\alpha\alpha\beta\beta}\alpha_y^2\beta_y\beta_x \tag{2.313}$$
$$+ 3u_{\alpha\alpha\beta\beta}\alpha_y\alpha_x\beta_y^2 + u_{\alpha\beta\beta\beta}\alpha_x\beta_y^3 + 3u_{\alpha\beta\beta\beta}\alpha_y\beta_y^2\beta_x \tag{2.314}$$
$$+ u_{\beta\beta\beta\beta}\beta_y^3\beta_x + 3u_{\alpha\alpha\alpha}\alpha_y^2\alpha_{xy} + 3u_{\alpha\alpha\alpha}\alpha_y\alpha_{yy}\alpha_x \tag{2.315}$$
$$+ 6u_{\alpha\alpha\beta}\alpha_y\alpha_{xy}\beta_y + 3u_{\alpha\alpha\beta}\alpha_y^2\beta_{xy} + 3u_{\alpha\alpha\beta}\alpha_y\alpha_{yy}\beta_x \tag{2.316}$$
$$+ 3u_{\alpha\alpha\beta}\alpha_x\alpha_{yy}\beta_y + 3u_{\alpha\alpha\beta}\alpha_y\alpha_x\beta_{yy} + 3u_{\alpha\beta\beta}\alpha_{xy}\beta_y^2 \tag{2.317}$$
$$+ 6u_{\alpha\beta\beta}\alpha_y\beta_y\beta_{xy} + 3u_{\alpha\beta\beta}\alpha_{yy}\beta_y\beta_x + 3u_{\alpha\beta\beta}\alpha_y\beta_{yy}\beta_x \tag{2.318}$$
$$+ 3u_{\alpha\beta\beta}\alpha_x\beta_y\beta_{yy} + 3u_{\beta\beta\beta}\beta_y^2\beta_{xy} + 3u_{\beta\beta\beta}\beta_y\beta_{yy}\beta_x \tag{2.319}$$
$$+ 3u_{\alpha\alpha}\alpha_{xy}\alpha_{yy} + 3u_{\alpha\alpha}\alpha_y\alpha_{xyy} + u_{\alpha\alpha}\alpha_{yyy}\alpha_x + 3u_{\alpha\beta}\alpha_{xyy}\beta_y \tag{2.320}$$
$$+ 3u_{\alpha\beta}\alpha_{yy}\beta_{xy} + 3u_{\alpha\beta}\alpha_{xy}\beta_{yy} + 3u_{\alpha\beta}\alpha_y\beta_{xyy} + u_{\alpha\beta}\alpha_{yyy}\beta_x \tag{2.321}$$
$$+ u_{\alpha\beta}\alpha_x\beta_{yyy} + 3u_{\beta\beta}\beta_{yy}\beta_{xy} + 3u_{\beta\beta}\beta_y\beta_{xyy} + u_{\beta\beta}\beta_{yyy}\beta_x \tag{2.322}$$
$$+ u_\alpha\alpha_{xyyy} + u_\beta\beta_{xyyy} \tag{2.323}$$
$$u_{xxyy} = u_{\alpha\alpha\alpha\alpha}\alpha_x^2\alpha_y^2 + 2u_{\alpha\alpha\alpha\beta}\alpha_x\alpha_y^2\beta_x + 2u_{\alpha\alpha\alpha\beta}\alpha_x^2\alpha_y\beta_y \tag{2.324}$$
$$+ u_{\alpha\alpha\beta\beta}\alpha_y^2\beta_x^2 + 4u_{\alpha\alpha\beta\beta}\alpha_x\alpha_y\beta_x\beta_y + u_{\alpha\alpha\beta\beta}\alpha_x^2\beta_y^2 \tag{2.325}$$
$$+ u_{\alpha\beta\beta\beta}\alpha_x\beta_x\beta_y^2 + 2u_{\alpha\beta\beta\beta}\alpha_y\beta_x^2\beta_y + u_{\alpha\beta\beta\beta}\alpha_x\beta_x\beta_y^2 \tag{2.326}$$
$$+ u_{\beta\beta\beta\beta}\beta_x^2\beta_y^2 + u_{\alpha\alpha\alpha}\alpha_{xx}\alpha_y^2 + 4u_{\alpha\alpha\alpha}\alpha_x\alpha_y\alpha_{xy} \tag{2.327}$$

$$+ u_{\alpha\alpha\alpha}\alpha_x^2\alpha_{yy} + u_{\alpha\alpha\beta}\alpha_y^2\beta_{xx} + 4u_{\alpha\alpha\beta}\alpha_y\alpha_{xy}\beta_x \tag{2.328}$$
$$+ 2u_{\alpha\alpha\beta}\alpha_{xx}\alpha_y\beta_y + 4u_{\alpha\alpha\beta}\alpha_x\alpha_{xy}\beta_y + 4u_{\alpha\alpha\beta}\alpha_x\alpha_y\beta_{xy} \tag{2.329}$$
$$+ 2u_{\alpha\alpha\beta}\alpha_x\alpha_{yy}\beta_x + u_{\alpha\alpha\beta}\alpha_x^2\beta_{yy} + u_{\alpha\beta\beta}\alpha_{xx}\beta_y^2 + 4u_{\alpha\beta\beta}\alpha_x\beta_y\beta_{xy} \tag{2.330}$$
$$+ 4u_{\alpha\beta\beta}\alpha_{xy}\beta_x\beta_y + 2u_{\alpha\beta\beta}\alpha_y\beta_{xx}\beta_y + 4u_{\alpha\beta\beta}\alpha_y\beta_x\beta_{xy} \tag{2.331}$$
$$+ u_{\alpha\beta\beta}\alpha_{yy}\beta_x^2 + 2u_{\alpha\beta\beta}\alpha_x\beta_x\beta_{yy} + u_{\beta\beta\beta}\beta_{xx}\beta_y^2 + 4u_{\beta\beta\beta}\beta_x\beta_y\beta_{xy} \tag{2.332}$$
$$+ u_{\beta\beta\beta}\beta_x^2\beta_{yy} + 2u_{\alpha\alpha}\alpha_{xy}^2 + 2u_{\alpha\alpha}\alpha_y\alpha_{xxy} + u_{\alpha\alpha}\alpha_{xx}\alpha_{yy} \tag{2.333}$$
$$+ 2u_{\alpha\alpha}\alpha_x\alpha_{xyy} + 2u_{\alpha\beta}\alpha_{xxy}\beta_y + 4u_{\alpha\beta}\alpha_{xy}\beta_{xy} + 2u_{\alpha\beta}\alpha_y\beta_{xxy} \tag{2.334}$$
$$+ 2u_{\alpha\beta}\alpha_{xyy}\beta_x + u_{\alpha\beta}\alpha_{yy}\beta_{xx} + u_{\alpha\beta}\alpha_{xx}\beta_{yy} + 2u_{\alpha\beta}\alpha_x\beta_{xyy} \tag{2.335}$$
$$+ 2u_{\beta\beta}\beta_{xy}^2 + 2u_{\beta\beta}\beta_y\beta_{xxy} + u_{\beta\beta}\beta_{xx}\beta_{yy} + 2u_{\beta\beta}\beta_x\beta_{xyy} \tag{2.336}$$
$$+ u_\alpha\alpha_{xxyy} + u_\beta\beta_{xxyy} \tag{2.337}$$

Hence, (2.270) becomes

$$h^2u_{xxxx} + 12ch^2k^2u_{xxyy} + k^2u_{yyyy} \tag{2.338}$$
$$= h^2\begin{pmatrix} u_{\alpha\alpha\alpha\alpha}\alpha_x^4 + 4u_{\alpha\alpha\alpha\beta}\alpha_x^3\beta_x + 6u_{\alpha\alpha\beta\beta}\alpha_x^2\beta_x^2 + 4u_{\alpha\beta\beta\beta}\alpha_x\beta_x^3 \\ +u_{\beta\beta\beta\beta}\beta_x^4 + 6u_{\alpha\alpha\alpha}\alpha_x^2\alpha_{xx} + 12u_{\alpha\alpha\beta}\alpha_x\alpha_{xx}\beta_x + 6u_{\alpha\alpha\beta}\alpha_x^2\beta_{xx} \\ +12u_{\alpha\beta\beta}\alpha_x\beta_x\beta_{xx} + 6u_{\alpha\beta\beta}\alpha_{xx}\beta_x^2 + 6u_{\beta\beta\beta}\beta_x^2\beta_{xx} + 3u_{\alpha\alpha}\alpha_{xx}^2 \\ +4u_{\alpha\alpha}\alpha_x\alpha_{xxx} + 4u_{\alpha\beta}\alpha_{xxx}\beta_x + 6u_{\alpha\beta}\alpha_{xx}\beta_{xx} + 4u_{\alpha\beta}\alpha_x\beta_{xxx} \\ +3u_{\beta\beta}\beta_{xx}^2 + 4u_{\beta\beta}\beta_x\beta_{xxx} + u_\alpha\alpha_{xxxx} + u_\beta\beta_{xxxx} \end{pmatrix} \tag{2.339}$$
$$+ 12ch^2k^2\begin{pmatrix} u_{\alpha\alpha\alpha\alpha}\alpha_x^2\alpha_y^2 + 2u_{\alpha\alpha\alpha\beta}\alpha_x\alpha_y^2\beta_x + 2u_{\alpha\alpha\alpha\beta}\alpha_x^2\alpha_y\beta_y \\ +u_{\alpha\alpha\beta\beta}\alpha_y^2\beta_x^2 + 4u_{\alpha\alpha\beta\beta}\alpha_x\alpha_y\beta_x\beta_y + u_{\alpha\alpha\beta\beta}\alpha_x^2\beta_y^2 \\ +u_{\alpha\beta\beta\beta}\alpha_x\beta_x\beta_y^2 + 2u_{\alpha\beta\beta\beta}\alpha_y\beta_x^2\beta_y + u_{\alpha\beta\beta\beta}\alpha_x\beta_x\beta_y^2 \\ +u_{\beta\beta\beta\beta}\beta_x^2\beta_y^2 + u_{\alpha\alpha\alpha}\alpha_{xx}\alpha_y^2 + 4u_{\alpha\alpha\alpha}\alpha_x\alpha_y\alpha_{xy} \\ +u_{\alpha\alpha\alpha}\alpha_x^2\alpha_{yy} + u_{\alpha\alpha\beta}\alpha_y^2\beta_{xx} + 4u_{\alpha\alpha\beta}\alpha_y\alpha_{xy}\beta_x \\ +2u_{\alpha\alpha\beta}\alpha_{xx}\alpha_y\beta_y + 4u_{\alpha\alpha\beta}\alpha_x\alpha_{xy}\beta_y + 4u_{\alpha\alpha\beta}\alpha_x\alpha_y\beta_{xy} \\ +2u_{\alpha\alpha\beta}\alpha_x\alpha_{yy}\beta_x + u_{\alpha\alpha\beta}\alpha_x^2\beta_{yy} + u_{\alpha\beta\beta}\alpha_{xx}\beta_y^2 \\ +4u_{\alpha\beta\beta}\alpha_x\beta_y\beta_{xy} + 4u_{\alpha\beta\beta}\alpha_{xy}\beta_x\beta_y + 2u_{\alpha\beta\beta}\alpha_y\beta_{xx}\beta_y \\ +4u_{\alpha\beta\beta}\alpha_y\beta_x\beta_{xy} + u_{\alpha\beta\beta}\alpha_{yy}\beta_x^2 + 2u_{\alpha\beta\beta}\alpha_x\beta_x\beta_{yy} \\ +u_{\beta\beta\beta}\beta_{xx}\beta_y^2 + 4u_{\beta\beta\beta}\beta_x\beta_y\beta_{xy} + u_{\beta\beta\beta}\beta_x^2\beta_{yy} \\ +2u_{\alpha\alpha}\alpha_{xy}^2 + 2u_{\alpha\alpha}\alpha_y\alpha_{xxy} + u_{\alpha\alpha}\alpha_{xx}\alpha_{yy} + 2u_{\alpha\alpha}\alpha_x\alpha_{xyy} \\ +2u_{\alpha\beta}\alpha_{xxy}\beta_y + 4u_{\alpha\beta}\alpha_{xy}\beta_{xy} + 2u_{\alpha\beta}\alpha_y\beta_{xxy} \\ +2u_{\alpha\beta}\alpha_{xyy}\beta_x + u_{\alpha\beta}\alpha_{yy}\beta_{xx} + u_{\alpha\beta}\alpha_{xx}\beta_{yy} + 2u_{\alpha\beta}\alpha_x\beta_{xyy} \\ +2u_{\beta\beta}\beta_{xy}^2 + 2u_{\beta\beta}\beta_y\beta_{xxy} + u_{\beta\beta}\beta_{xx}\beta_{yy} + 2u_{\beta\beta}\beta_x\beta_{xyy} \\ +u_\alpha\alpha_{xxyy} + u_\beta\beta_{xxyy} \end{pmatrix} \tag{2.340}$$
$$+ k^2\begin{pmatrix} u_{\alpha\alpha\alpha\alpha}\alpha_y^4 + 4u_{\alpha\alpha\alpha\beta}\alpha_y^3\beta_y + 6u_{\alpha\alpha\beta\beta}\alpha_y^2\beta_y^2 + 4u_{\alpha\beta\beta\beta}\alpha_y\beta_y^3 \\ +u_{\beta\beta\beta\beta}\beta_y^4 + 6u_{\alpha\alpha\alpha}\alpha_y^2\alpha_{yy} + 12u_{\alpha\alpha\beta}\alpha_y\alpha_{yy}\beta_y + 6u_{\alpha\alpha\beta}\alpha_y^2\beta_{yy} \\ +12u_{\alpha\beta\beta}\alpha_y\beta_y\beta_{yy} + 6u_{\alpha\beta\beta}\alpha_{yy}\beta_y^2 + 6u_{\beta\beta\beta}\beta_y^2\beta_{yy} + 3u_{\alpha\alpha}\alpha_{yy}^2 \\ +4u_{\alpha\alpha}\alpha_y\alpha_{yyy} + 4u_{\alpha\beta}\alpha_{yyy}\beta_y + 6u_{\alpha\beta}\alpha_{yy}\beta_{yy} + 4u_{\alpha\beta}\alpha_y\beta_{yyy} \\ +3u_{\beta\beta}\beta_{yy}^2 + 4u_{\beta\beta}\beta_y\beta_{yyy} + u_\alpha\alpha_{yyyy} + u_\beta\beta_{yyyy} \end{pmatrix} \tag{2.341}$$
$$= \left(h^2\alpha_x^4 + 12ch^2k^2\alpha_x^2\alpha_y^2 + k^2\alpha_y^4\right)u_{\alpha\alpha\alpha\alpha} \tag{2.342}$$
$$+ \left(h^2\beta_x^4 + 12ch^2k^2\beta_x^2\beta_y^2 + k^2\beta_y^4\right)u_{\beta\beta\beta\beta} \tag{2.343}$$

$$+\left(\begin{array}{l}4h^2\alpha_x^3\beta_x+24ch^2k^2\alpha_x\alpha_y^2\beta_x\\+24ch^2k^2\alpha_x^2\alpha_y\beta_y+4k^2\alpha_y^3\beta_y\end{array}\right)u_{\alpha\alpha\alpha\beta} \tag{2.344}$$

$$+\left(\begin{array}{l}6h^2\alpha_x^2\beta_x^2+12ch^2k^2\alpha_y^2\beta_x^2+48ch^2k^2\alpha_x\alpha_y\beta_x\beta_y\\+12ch^2k^2\alpha_x^2\beta_y^2+6k^2\alpha_y^2\beta_y^2\end{array}\right)u_{\alpha\alpha\beta\beta} \tag{2.345}$$

$$+\left(\begin{array}{l}4h^2\alpha_x\beta_x^3+12ch^2k^2\alpha_x\beta_x\beta_y^2+24ch^2k^2\alpha_y\beta_x^2\beta_y\\+12ch^2k^2\alpha_x\beta_x\beta_y^2+4k^2\alpha_y\beta_y^3\end{array}\right)u_{\alpha\beta\beta\beta} \tag{2.346}$$

$$+\left(\begin{array}{l}6h^2\alpha_x^2\alpha_{xx}+12ch^2k^2\alpha_{xx}\alpha_y^2+48ch^2k^2\alpha_x\alpha_y\alpha_{xy}\\+12ch^2k^2\alpha_x^2\alpha_{yy}+6k^2\alpha_y^2\alpha_{yy}\end{array}\right)u_{\alpha\alpha\alpha} \tag{2.347}$$

$$+\left(\begin{array}{l}12h^2\alpha_x\alpha_{xx}\beta_x+6h^2\alpha_x^2\beta_{xx}+12ch^2k^2\alpha_y^2\beta_{xx}\\+48ch^2k^2\alpha_y\alpha_{xy}\beta_x+24ch^2k^2\alpha_{xx}\alpha_y\beta_y\\+48ch^2k^2\alpha_x\alpha_{xy}\beta_y+48ch^2k^2\alpha_x\alpha_y\beta_{xy}\\+24ch^2k^2\alpha_x\alpha_{yy}\beta_x+12ch^2k^2\alpha_x^2\beta_{yy}\\+12k^2\alpha_y\alpha_{yy}\beta_y+6k^2\alpha_y^2\beta_{yy}\end{array}\right)u_{\alpha\alpha\beta} \tag{2.348}$$

$$+\left(\begin{array}{l}12h^2\alpha_x\beta_x\beta_{xx}+6h^2\alpha_{xx}\beta_x^2+12ch^2k^2\alpha_{xx}\beta_y^2\\+48ch^2k^2\alpha_x\beta_y\beta_{xy}+48ch^2k^2\alpha_{xy}\beta_x\beta_y\\+24ch^2k^2\alpha_y\beta_{xx}\beta_y+48ch^2k^2\alpha_y\beta_x\beta_{xy}\\+12ch^2k^2\alpha_{yy}\beta_x^2+24ch^2k^2\alpha_x\beta_x\beta_{yy}\\+12k^2\alpha_y\beta_y\beta_{yy}+6k^2\alpha_{yy}\beta_y^2\end{array}\right)u_{\alpha\beta\beta} \tag{2.349}$$

$$+\left(\begin{array}{l}6h^2\beta_x^2\beta_{xx}+12ch^2k^2\beta_{xx}\beta_y^2+48ch^2k^2\beta_x\beta_y\beta_{xy}\\+12ch^2k^2\beta_x^2\beta_{yy}+6k^2\beta_y^2\beta_{yy}\end{array}\right)u_{\beta\beta\beta} \tag{2.350}$$

$$+\left(\begin{array}{l}3h^2\alpha_{xx}^2+4h^2\alpha_x\alpha_{xxx}+24ch^2k^2\alpha_{xy}^2\\+24ch^2k^2\alpha_y\alpha_{xxy}+12ch^2k^2\alpha_{xx}\alpha_{yy}\\+24ch^2k^2\alpha_x\alpha_{xyy}+3k^2\alpha_{yy}^2+4k^2\alpha_y\alpha_{yyy}\end{array}\right)u_{\alpha\alpha} \tag{2.351}$$

$$+\left(\begin{array}{l}4h^2\alpha_{xxx}\beta_x+6h^2\alpha_{xx}\beta_{xx}+4h^2\alpha_x\beta_{xxx}+24ch^2k^2\alpha_{xxy}\beta_y\\+48ch^2k^2\alpha_{xy}\beta_{xy}+24ch^2k^2\alpha_y\beta_{xxy}+24ch^2k^2\alpha_{xyy}\beta_x\\+12ch^2k^2\alpha_{yy}\beta_{xx}+12ch^2k^2\alpha_{xx}\beta_{yy}+24ch^2k^2\alpha_x\beta_{xyy}\\+4k^2\alpha_{yyy}\beta_y+6k^2\alpha_{yy}\beta_{yy}+4k^2\alpha_y\beta_{yyy}\end{array}\right)u_{\alpha\beta} \tag{2.352}$$

$$+\left(\begin{array}{l}3h^2\beta_{xx}^2+4h^2\beta_x\beta_{xxx}+24ch^2k^2\beta_{xy}^2\\+24ch^2k^2\beta_y\beta_{xxy}+12ch^2k^2\beta_{xx}\beta_{yy}\\+24ch^2k^2\beta_x\beta_{xyy}+3k^2\beta_{yy}^2+4k^2\beta_y\beta_{yyy}\end{array}\right)u_{\beta\beta} \tag{2.353}$$

$$+\left(h^2\alpha_{xxxx}+12ch^2k^2\alpha_{xxyy}+k^2\alpha_{yyyy}\right)u_\alpha \tag{2.354}$$

$$+\left(h^2\beta_{xxxx}+12ch^2k^2\beta_{xxyy}+k^2\beta_{yyyy}\right)u_\beta \tag{2.355}$$

Remark 2.17. We just need to compute

$$u_{xx},u_{yy},u_{xxxx},u_{xxyy},u_{yyyy} \tag{2.356}$$

but we also computer other partial derivatives to make references for further purposes.

What are suitable choices of the change of variables α and β? To answer this, we go back to (1.1).

$$f=u_{xx}+u_{yy} \tag{2.357}$$

$$=\left(\alpha_x^2+\alpha_y^2\right)u_{\alpha\alpha}+\left(\beta_x^2+\beta_y^2\right)u_{\beta\beta} \tag{2.358}$$

$$+ 2\left(\alpha_x\beta_x + \alpha_y\beta_y\right) u_{\alpha\beta} + \left(\Delta\alpha\right) u_\alpha + \left(\Delta\beta\right) u_\beta \tag{2.359}$$

Hence, we want that Δu appears in (2.358) and the rest (2.359) are removed, i.e., α, β need choosing to satisfy

$$\alpha_x^2 + \alpha_y^2 = \beta_x^2 + \beta_y^2 \tag{2.360}$$
$$\alpha_x\beta_x + \alpha_y\beta_y = 0 \tag{2.361}$$
$$\Delta\alpha = 0 \tag{2.362}$$
$$\Delta\beta = 0 \tag{2.363}$$

It is obvious to see that (2.360)-(2.363) has many solutions. Indeed, a family of solutions is easily obtained by taking two non-constant harmonic functions α, β such that

$$\alpha_x = \beta_y \tag{2.364}$$
$$\alpha_y = -\beta_x \tag{2.365}$$

Remark 2.18. It is interesting to notice that (2.364)-(2.365) satisfy (2.274). Indeed, we have

$$\det\left(J_{\alpha,\beta}\left(x,y\right)\right) = \alpha_x\beta_y - \alpha_y\beta_x \tag{2.366}$$
$$= \alpha_x^2 + \alpha_y^2 > 0 \tag{2.367}$$

since α is non-constant.

For simplicity, we will take α and β as two linear functions of x and y. The choice

$$\alpha\left(x,y\right) = x + y \tag{2.368}$$
$$\beta\left(x,y\right) = -x + y \tag{2.369}$$

may be the simplest and brightest one.

Remark 2.19. Other choices of α, β will give a lot of interesting results. You should give it a try to discover deeper.

Under (2.368)-(2.369), (2.357)-(2.359) becomes

$$f = u_{xx} + u_{yy} \tag{2.370}$$
$$= 2\left(u_{\alpha\alpha} + u_{\beta\beta}\right) \tag{2.371}$$

i.e., our original Poisson problem (1.1) becomes

$$u_{\alpha\alpha} + u_{\beta\beta} = \frac{f}{2} \tag{2.372}$$

in new coordinate defined by (2.368)-(2.369) We now rewrite all partial derivatives of u computed generally above.

First Order Partial Derivatives.

$$u_x = u_\alpha - u_\beta \tag{2.373}$$

$$u_y = u_\alpha + u_\beta \tag{2.374}$$

Second Order Partial Derivatives.

$$u_{xx} = u_{\alpha\alpha} - 2u_{\alpha\beta} + u_{\beta\beta} \tag{2.375}$$
$$u_{xy} = u_{\alpha\alpha} - u_{\beta\beta} \tag{2.376}$$
$$u_{yy} = u_{\alpha\alpha} + 2u_{\alpha\beta} + u_{\beta\beta} \tag{2.377}$$

Third Order Partial Derivatives.

$$u_{xxx} = u_{\alpha\alpha\alpha} - 3u_{\alpha\alpha\beta} + 3u_{\alpha\beta\beta} - u_{\beta\beta\beta} \tag{2.378}$$
$$u_{xxy} = u_{\alpha\alpha\alpha} - u_{\alpha\alpha\beta} - u_{\alpha\beta\beta} + u_{\beta\beta\beta} \tag{2.379}$$
$$u_{xyy} = u_{\alpha\alpha\alpha} + u_{\alpha\alpha\beta} - u_{\alpha\beta\beta} - u_{\beta\beta\beta} \tag{2.380}$$
$$u_{yyy} = u_{\alpha\alpha\alpha} + 3u_{\alpha\alpha\beta} + 3u_{\alpha\beta\beta} + u_{\beta\beta\beta} \tag{2.381}$$

Fourth Order Partial Derivatives.

$$u_{xxxx} = u_{\alpha\alpha\alpha\alpha} - 4u_{\alpha\alpha\alpha\beta} + 6u_{\alpha\alpha\beta\beta} - 4u_{\alpha\beta\beta\beta} + u_{\beta\beta\beta\beta} \tag{2.382}$$
$$u_{yyyy} = u_{\alpha\alpha\alpha\alpha} + 4u_{\alpha\alpha\alpha\beta} + 6u_{\alpha\alpha\beta\beta} + 4u_{\alpha\beta\beta\beta} + u_{\beta\beta\beta\beta} \tag{2.383}$$
$$u_{xxxy} = u_{\alpha\alpha\alpha\alpha} - 2u_{\alpha\alpha\alpha\beta} + 2u_{\alpha\beta\beta\beta} - u_{\beta\beta\beta\beta} \tag{2.384}$$
$$u_{xyyy} = u_{\alpha\alpha\alpha\alpha} + 2u_{\alpha\alpha\alpha\beta} - 2u_{\alpha\beta\beta\beta} - u_{\beta\beta\beta\beta} \tag{2.385}$$
$$u_{xxyy} = u_{\alpha\alpha\alpha\alpha} - 2u_{\alpha\alpha\beta\beta} + u_{\beta\beta\beta\beta} \tag{2.386}$$

and

$$h^2 u_{xxxx} + 12ch^2k^2 u_{xxyy} + k^2 u_{yyyy} \tag{2.387}$$
$$= h^2 \left(u_{\alpha\alpha\alpha\alpha} - 4u_{\alpha\alpha\alpha\beta} + 6u_{\alpha\alpha\beta\beta} - 4u_{\alpha\beta\beta\beta} + u_{\beta\beta\beta\beta}\right) \tag{2.388}$$
$$+ 12ch^2k^2 \left(u_{\alpha\alpha\alpha\alpha} - 2u_{\alpha\alpha\beta\beta} + u_{\beta\beta\beta\beta}\right) \tag{2.389}$$
$$+ k^2 \left(u_{\alpha\alpha\alpha\alpha} + 4u_{\alpha\alpha\alpha\beta} + 6u_{\alpha\alpha\beta\beta} + 4u_{\alpha\beta\beta\beta} + u_{\beta\beta\beta\beta}\right) \tag{2.390}$$
$$= \left(h^2 + 12ch^2k^2 + k^2\right) u_{\alpha\alpha\alpha\alpha} + 4\left(k^2 - h^2\right) u_{\alpha\alpha\alpha\beta} \tag{2.391}$$
$$+ \left(6h^2 - 24ch^2k^2 + 6k^2\right) u_{\alpha\alpha\beta\beta} + 4\left(k^2 - h^2\right) u_{\alpha\beta\beta\beta} \tag{2.392}$$
$$+ \left(h^2 + 12ch^2k^2 + k^2\right) u_{\beta\beta\beta\beta} \tag{2.393}$$

We now try to compute (2.387)-(2.393) in terms of f. To this end, we entirely base our hope on (2.372). Computing some partial derivatives of f gives

$$f = 2u_{\alpha\alpha} + 2u_{\beta\beta} \tag{2.394}$$
$$f_x = 2u_{\alpha\alpha\alpha} - 2u_{\alpha\alpha\beta} + 2u_{\alpha\beta\beta} - 2u_{\beta\beta\beta} \tag{2.395}$$
$$f_y = 2u_{\alpha\alpha\alpha} + 2u_{\alpha\alpha\beta} + 2u_{\alpha\beta\beta} + 2u_{\beta\beta\beta} \tag{2.396}$$
$$f_{xx} = 2u_{\alpha\alpha\alpha\alpha} - 4u_{\alpha\alpha\alpha\beta} + 4u_{\alpha\alpha\beta\beta} - 4u_{\alpha\beta\beta\beta} + 2u_{\beta\beta\beta\beta} \tag{2.397}$$
$$f_{yy} = 2u_{\alpha\alpha\alpha\alpha} + 4u_{\alpha\alpha\alpha\beta} + 4u_{\alpha\alpha\beta\beta} + 4u_{\alpha\beta\beta\beta} + 2u_{\beta\beta\beta\beta} \tag{2.398}$$

We now consider the "computable term"

$$\xi f_{xx} + \eta f_{yy} = \xi \left(2u_{\alpha\alpha\alpha\alpha} - 4u_{\alpha\alpha\alpha\beta} + 4u_{\alpha\alpha\beta\beta} - 4u_{\alpha\beta\beta\beta} + 2u_{\beta\beta\beta\beta}\right) \tag{2.399}$$
$$+ \eta \left(2u_{\alpha\alpha\alpha\alpha} + 4u_{\alpha\alpha\alpha\beta} + 4u_{\alpha\alpha\beta\beta} + 4u_{\alpha\beta\beta\beta} + 2u_{\beta\beta\beta\beta}\right) \tag{2.400}$$
$$= 2\left(\xi + \eta\right) u_{\alpha\alpha\alpha\alpha} + 4\left(\eta - \xi\right) u_{\alpha\alpha\alpha\beta} + 4\left(\xi + \eta\right) u_{\alpha\alpha\beta\beta} \tag{2.401}$$

$$+ 4(\eta - \xi) u_{\alpha\beta\beta\beta} + 2(\xi + \eta) u_{\beta\beta\beta\beta} \tag{2.402}$$

Identifying the coefficients of (2.387)-(2.393) and (2.399)-(2.402) yields

$$h^2 + 12ch^2k^2 + k^2 = 2(\xi + \eta) \tag{2.403}$$
$$4\left(k^2 - h^2\right) = 4(\eta - \xi) \tag{2.404}$$
$$6h^2 - 24ch^2k^2 + 6k^2 = 4(\xi + \eta) \tag{2.405}$$
$$4\left(k^2 - h^2\right) = 4(\eta - \xi) \tag{2.406}$$
$$h^2 + 12ch^2k^2 + k^2 = 2(\xi + \eta) \tag{2.407}$$

Combining (2.403) and (2.405) yields

$$6h^2 - 24ch^2k^2 + 6k^2 = 2\left(h^2 + 12ch^2k^2 + k^2\right) \tag{2.408}$$

Solving c out yields

$$c = \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \tag{2.409}$$

Then (2.403)-(2.407) becomes

$$\xi + \eta = h^2 + k^2 \tag{2.410}$$
$$\xi - \eta = h^2 - k^2 \tag{2.411}$$

i.e.,

$$\xi = h^2 \tag{2.412}$$
$$\eta = k^2 \tag{2.413}$$

Therefore, (2.258)-(2.260) becomes

$$a = \frac{5}{6h^2} - \frac{1}{6k^2} \tag{2.414}$$
$$b = \frac{5}{6k^2} - \frac{1}{6h^2} \tag{2.415}$$
$$c = \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \tag{2.416}$$
$$d = \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) \tag{2.417}$$

and (2.261)-(2.269) becomes

$$\left(\frac{5}{6h^2} - \frac{1}{6k^2}\right)(u(x_{i-1}, y_j) + u(x_{i+1}, y_j) - 2u(x_i, y_j)) \tag{2.418}$$
$$+ \left(\frac{5}{6k^2} - \frac{1}{6h^2}\right)(u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 2u(x_i, y_j)) \tag{2.419}$$
$$+ \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right)(u(x_{i-1}, y_{j-1}) + u(x_{i+1}, y_{j+1}) - 2u(x_i, y_j)) \tag{2.420}$$
$$+ \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right)(u(x_{i-1}, y_{j+1}) + u(x_{i+1}, y_{j-1}) - 2u(x_i, y_j)) \tag{2.421}$$

$$= \Delta u(x_i, y_j) + \frac{1}{12} h^2 u_{xxxx}(x_i, y_j) + \frac{1}{12}\left(h^2 + k^2\right) u_{xxyy}(x_i, y_j) \tag{2.422}$$

$$+ \frac{1}{12} k^2 u_{yyyy}(x_i, y_j) + \frac{5}{6}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) O\left(H^6\right) \tag{2.423}$$

$$= \Delta u(x_i, y_j) \tag{2.424}$$

$$+ \frac{1}{6}\left(\begin{array}{l} \left(h^2 + k^2\right) u_{\alpha\alpha\alpha\alpha}(x_i, y_j) + 2\left(k^2 - h^2\right) u_{\alpha\alpha\alpha\beta} \\ +2\left(h^2 + k^2\right) u_{\alpha\alpha\beta\beta}(x_i, y_j) + 2\left(k^2 - h^2\right) u_{\alpha\beta\beta\beta} \\ +\left(h^2 + k^2\right) u_{\beta\beta\beta\beta}(x_i, y_j) \end{array}\right) \tag{2.425}$$

$$+ \frac{5}{6}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) O\left(H^6\right) \tag{2.426}$$

$$= f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx} + k^2 f_{yy}\right) + \frac{5}{6}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) O\left(H^6\right) \tag{2.427}$$

Finally, we claim that

$$\frac{5}{6}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) O\left(H^6\right) = O\left(H^4\right) \tag{2.428}$$

It suffices to prove

$$\frac{1}{h^2} + \frac{1}{k^2} = O\left(\frac{1}{H^2}\right) \tag{2.429}$$

Indeed, using (1.3) yields

$$h \geq \min\{\alpha_0, 1\} H \tag{2.430}$$

$$k \geq \min\left\{\frac{1}{\beta_0}, 1\right\} H \tag{2.431}$$

Hence,

$$\frac{1}{h^2} + \frac{1}{k^2} \leq \frac{1}{\left(\min\{\alpha_0, 1\}\right)^2 H^2} + \frac{1}{\left(\min\left\{\frac{1}{\beta_0}, 1\right\}\right)^2 H^2} \tag{2.432}$$

$$\leq \left(\frac{1}{\left(\min\{\alpha_0, 1\}\right)^2} + \frac{1}{\left(\min\left\{\frac{1}{\beta_0}, 1\right\}\right)^2}\right) \frac{1}{H^2} \tag{2.433}$$

$$= O\left(\frac{1}{H^2}\right) \tag{2.434}$$

Therefore, (2.418)-(2.427) becomes

$$\left(\frac{5}{6h^2} - \frac{1}{6k^2}\right)\left(u(x_{i-1}, y_j) + u(x_{i+1}, y_j) - 2u(x_i, y_j)\right) \tag{2.435}$$

$$+ \left(\frac{5}{6k^2} - \frac{1}{6h^2}\right)\left(u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 2u(x_i, y_j)\right) \tag{2.436}$$

$$+ \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right)\left(u(x_{i-1}, y_{j-1}) + u(x_{i+1}, y_{j+1}) - 2u(x_i, y_j)\right) \tag{2.437}$$

$$+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)\left(u\left(x_{i-1},y_{j+1}\right)+u\left(x_{i+1},y_{j-1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.438}$$

$$=f\left(x_i,y_j\right)+\frac{1}{12}\left(h^2f_{xx}+k^2f_{yy}\right)+O\left(H^4\right) \tag{2.439}$$

After a long and hard journey, we now define the 9-point stencil in rectangular uniform mesh as the left-hand side of (2.435)-(2.439).

$$\widehat{\nabla}_9^2u_{ij}:=\left(\frac{5}{6h^2}-\frac{1}{6k^2}\right)\left(u\left(x_{i-1},y_j\right)+u\left(x_{i+1},y_j\right)-2u\left(x_i,y_j\right)\right) \tag{2.440}$$

$$+\left(\frac{5}{6k^2}-\frac{1}{6h^2}\right)\left(u\left(x_i,y_{j-1}\right)+u\left(x_i,y_{j+1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.441}$$

$$+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)\left(u\left(x_{i-1},y_{j-1}\right)+u\left(x_{i+1},y_{j+1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.442}$$

$$+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)\left(u\left(x_{i-1},y_{j+1}\right)+u\left(x_{i+1},y_{j-1}\right)-2u\left(x_i,y_j\right)\right) \tag{2.443}$$

or equivalently,

$$\widehat{\nabla}_9^2u_{ij}=\left(\frac{5}{6h^2}-\frac{1}{6k^2}\right)u\left(x_{i-1},y_j\right)+\left(\frac{5}{6h^2}-\frac{1}{6k^2}\right)u\left(x_{i+1},y_j\right) \tag{2.444}$$

$$+\left(\frac{5}{6k^2}-\frac{1}{6h^2}\right)u\left(x_i,y_{j-1}\right)+\left(\frac{5}{6k^2}-\frac{1}{6h^2}\right)u\left(x_i,y_{j+1}\right) \tag{2.445}$$

$$+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u\left(x_{i-1},y_{j-1}\right)+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u\left(x_{i+1},y_{j+1}\right) \tag{2.446}$$

$$+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u\left(x_{i-1},y_{j+1}\right)+\frac{1}{12}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u\left(x_{i+1},y_{j-1}\right) \tag{2.447}$$

$$-\frac{5}{3}\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u\left(x_i,y_j\right) \tag{2.448}$$

Then,

$$\widehat{\nabla}_9^2u_{ij}=f\left(x_i,y_j\right)+\frac{1}{12}\left(h^2f_{xx}\left(x_i,y_j\right)+k^2f_{yy}\left(x_i,y_j\right)\right)+O\left(H^4\right) \tag{2.449}$$

Done. □

At this stage, everything is easier. The rest argument is the same as square uniform mesh case.

Explicitly, if we are solving $\Delta u=f$, we can compute the dominant term

$$\frac{1}{12}\left(h^2f_{xx}\left(x_i,y_j\right)+k^2f_{yy}\left(x_i,y_j\right)\right) \tag{2.450}$$

in the truncation error easily from the known function f without knowing the true solution u to the problem.

In particular, if we are solving Laplace's equation, where $f=0$, or more generally if f is a harmonic function, then this term in the local truncation error vanish and the 9-point Laplacian would give a fourth order accurate discretization of the differential equation.

More generally, we can obtain a fourth order accurate method of the form

$$\widehat{\nabla}_9^2 u_{ij} = f_{ij} \tag{2.451}$$

for arbitrary smooth functions $f(x,y)$ by defining

$$f_{ij} = f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx}(x_i, y_j) + k^2 f_{yy}(x_i, y_j)\right) \tag{2.452}$$

We can view this as deliberately introducing an $O\left(h^2\right)$ error into the right-hand side of the equation that is chosen to cancel the $O\left(h^2\right)$ part of the local truncation error. Taylor series expansion shows that the local truncation error of the method (2.451) is now $O\left(h^4\right)$.

$$\tau_{ij} = \widehat{\nabla}_9^2 u_{ij} - f_{ij} \tag{2.453}$$

$$= \left(f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx}(x_i, y_j) + k^2 f_{yy}(x_i, y_j)\right) + O\left(H^4\right)\right) \tag{2.454}$$

$$- \left(f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx}(x_i, y_j) + k^2 f_{yy}(x_i, y_j)\right)\right) \tag{2.455}$$

$$= O\left(H^4\right) \tag{2.456}$$

If we have only data $f(x_i, y_j)$ at the grid points (but we know that the underlying function is sufficiently smooth), then we can still achieve fourth order accurate by using

$$f_{ij} = f(x_i, y_j) + \frac{1}{12}\begin{pmatrix} f(x_{i-1}, y_j) + f(x_{i+1}, y_j) + f(x_i, y_{j-1}) \\ +f(x_i, y_{j+1}) - 4f(x_i, y_j) \end{pmatrix} \tag{2.457}$$

instead of (2.452). Indeed, using Taylor series expansion yields

$$f(x_{i-1}, y_j) + f(x_{i+1}, y_j) - 2f(x_i, y_j) = h^2 f_{xx}(x_i, y_j) + O\left(h^4\right) \tag{2.458}$$

$$f(x_i, y_{j-1}) + f(x_i, y_{j+1}) - 2f(x_i, y_j) = k^2 f_{xx}(x_i, y_j) + O\left(k^4\right) \tag{2.459}$$

Hence,

$$f(x_{i-1}, y_j) + f(x_{i+1}, y_j) + f(x_i, y_{j-1}) + f(x_i, y_{j+1}) - 4f(x_i, y_j) \tag{2.460}$$

$$= h^2 f_{xx}(x_i, y_j) + O\left(h^4\right) + k^2 f_{xx}(x_i, y_j) + O\left(k^4\right) \tag{2.461}$$

$$= h^2 f_{xx}(x_i, y_j) + k^2 f_{xx}(x_i, y_j) + O\left(H^4\right) \tag{2.462}$$

and (2.457) becomes

$$f_{ij} = f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx}(x_i, y_j) + k^2 f_{xx}(x_i, y_j)\right) + O\left(H^4\right) \tag{2.463}$$

Thus, the local truncation error is

$$\tau_{ij} = \widehat{\nabla}_9^2 u_{ij} - f_{ij} \tag{2.464}$$

$$= \left(f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx}(x_i, y_j) + k^2 f_{yy}(x_i, y_j)\right) + O\left(H^4\right)\right) \tag{2.465}$$

$$- \left(f(x_i, y_j) + \frac{1}{12}\left(h^2 f_{xx}(x_i, y_j) + k^2 f_{xx}(x_i, y_j)\right) + O\left(H^4\right)\right) \tag{2.466}$$

$$= O\left(H^4\right) \tag{2.467}$$

We have used the same trick as in the square uniform mesh case - introducing an "error" into the equations that is carefully chosen to cancel some other error.

2.6.3 Matrix for The 9-Point Laplacian Under Rowwise Ordering

Problem 2.20. *Find an explicit formula for the matrix equation for the 9-point Laplacian under the natural rowwise ordering.*

SOLUTION. We rewrite (2.440)-(2.443) as

$$\widehat{\nabla}_9^2 u_{ij} = \left(\frac{5}{6h^2} - \frac{1}{6k^2}\right) u\left(x_{i-1}, y_j\right) + \left(\frac{5}{6h^2} - \frac{1}{6k^2}\right) u\left(x_{i+1}, y_j\right) \tag{2.468}$$

$$+ \left(\frac{5}{6k^2} - \frac{1}{6h^2}\right) u\left(x_i, y_{j-1}\right) + \left(\frac{5}{6k^2} - \frac{1}{6h^2}\right) u\left(x_i, y_{j+1}\right) \tag{2.469}$$

$$+ \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) u\left(x_{i-1}, y_{j-1}\right) + \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) u\left(x_{i+1}, y_{j+1}\right) \tag{2.470}$$

$$+ \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) u\left(x_{i-1}, y_{j+1}\right) + \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) u\left(x_{i+1}, y_{j-1}\right) \tag{2.471}$$

$$- \frac{5}{3}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) u\left(x_i, y_j\right) \tag{2.472}$$

The matrix for 9-point Laplacian under rowwise ordering is

$$A_9 = \begin{bmatrix} R^{h,k} & S^{h,k} & & & \\ S^{h,k} & R^{h,k} & S^{h,k} & & \\ & \ddots & \ddots & \ddots & \\ & & S^{h,k} & R^{h,k} & S^{h,k} \\ & & & S^{h,k} & R^{h,k} \end{bmatrix} \tag{2.473}$$

where

$$R^{h,k} = \begin{bmatrix} -\frac{5}{3}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \frac{5}{6h^2} - \frac{1}{6k^2} & & & \\ \frac{5}{6h^2} - \frac{1}{6k^2} & -\frac{5}{3}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \frac{5}{6h^2} - \frac{1}{6k^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{5}{6h^2} - \frac{1}{6k^2} & -\frac{5}{3}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \end{bmatrix} \tag{2.474}$$

and

$$S^{h,k} = \begin{bmatrix} \frac{5}{6k^2} - \frac{1}{6h^2} & \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & & & \\ \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \frac{5}{6k^2} - \frac{1}{6h^2} & \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{12}\left(\frac{1}{h^2} + \frac{1}{k^2}\right) & \frac{5}{6k^2} - \frac{1}{6h^2} & \end{bmatrix} \tag{2.475}$$

Done. □

3 Square Uniform Mesh

This section focuses on solving (1.1) numerically on square uniform mesh. This is a special case $h = k$ of rectangular uniform mesh above as we have mentioned many times above.

We recall that we have also dealt with the 9-point stencil Laplacian on square uniform mesh. But the author have decided to separate the square uniform mesh here for completeness and quick-access purposes further.

And this section is actually retyped in [1], Chapter 3. Notice the differences between rectangular uniform meshes we have dealt with and this square uniform one.

Elliptic Equations

In more than one space dimension, the steady-state equation generalize naturally to *elliptic* partial differential equations. In two space dimensions a constant-coefficient elliptic equation has the form

$$a_1 u_{xx} + a_2 u_{xy} + a_3 u_{yy} + a_4 u_x + a_5 u_y + a_6 u = f \tag{3.1}$$

where the coefficients a_1, a_2, a_3 satisfy

$$a_2^2 - 4a_1 a_3 < 0 \tag{3.2}$$

This equation must be satisfied for all (x, y) in some region of the plane Ω, together with some boundary conditions on $\partial\Omega$, the boundary of Ω. For example, we may have Dirichlet boundary conditions in which case $u(u, y)$ is given at all points $(x, y) \in \partial\Omega$. If the ellipticity condition is satisfied, then this gives a well-posed problem. If the coefficients vary with x and y, then the ellipticity condition must be satisfied at each point in Ω.

3.1 Steady-State Heat Conduction

Equations of elliptic character often arise as steady-state equations in some region of space, associated with some time-dependent physical problem. For example, the diffusion or heat conduction equation in two space dimensions takes the form

$$u_t = (\kappa u_x)_x + (\kappa u_x)_x + \psi \tag{3.3}$$

where $\kappa(x, y) > 0$ is a diffusion or heat conduction coefficient that may vary with x and y, and $\psi(x, y, t)$ is a source term. The solution $u(x, y, t)$ generally will vary with time as well as space. We also need initial conditions $u(x, y, 0)$ in Ω and boundary conditions at each point in time at every point on the boundary of Ω. If the boundary conditions and source terms are independent of time, then we expect a steady state to exist, which we can find by solving the elliptic equation

$$(\kappa u_x)_x + (\kappa u_x)_x = f \tag{3.4}$$

where again we set $f(x, y(= -\psi(x, y)$, together with the boundary conditions. Note that (3.2) is satisfied at each point, provided $\kappa > 0$ everywhere. Indeed

$$a_2^2 - 4a_1 a_3 = -4\kappa^2 < 0 \tag{3.5}$$

We first consider the simplest case where $\kappa \equiv 1$. We then have the *Poisson problem* (1.1)

$$u_{xx} + u_{yy} = f \tag{3.6}$$

In the special case $f \equiv 0$, this reduces to *Laplace's equation*

$$u_{xx} + u_{yy} = 0 \tag{3.7}$$

We also need to specify boundary conditions all around the boundary of the region Ω. These could be Dirichlet conditions, where the temperature $u(x, y)$ is specified at each point on the boundary, or Neumann conditions, where the normal derivative (the heat flux) is specified. We may have Dirichlet conditions specified at some points on the boundary and Neumann conditions at other points.

3.2 The 5-Point Stencil for The Laplacian

To discuss discretizations, first consider the Poisson problem (1.1) on the unit square $0 \leq x \leq 1, 0 \leq y \leq 1$ and suppose we have Dirichlet boundary conditions. We will use a square uniform Cartesian grid consisting of grid points (x_i, y_j), where

$$x_i = ih, i = 0, 1, \ldots, N+1 \tag{3.8}$$
$$y_j = jh, j = 0, 1, \ldots, N+1 \tag{3.9}$$

where $h = \dfrac{1}{N+1}$.

Remark 3.1. This square uniform mesh is a special case of retangular uniform mesh with

$$N_x = N_y = N \tag{3.10}$$
$$h = k = \frac{1}{N+1} \tag{3.11}$$

Let u_{ij} represent an approximation to $u(x_i, y_j)$ as usual. To discretize (1.1) we replace the x- and y-derivatives with centered finite differences, which gives

$$\frac{1}{h^2}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right) = f_{ij} \tag{3.12}$$

This finite difference scheme can be represented by the *5-point stencil* shown in Figure 5.

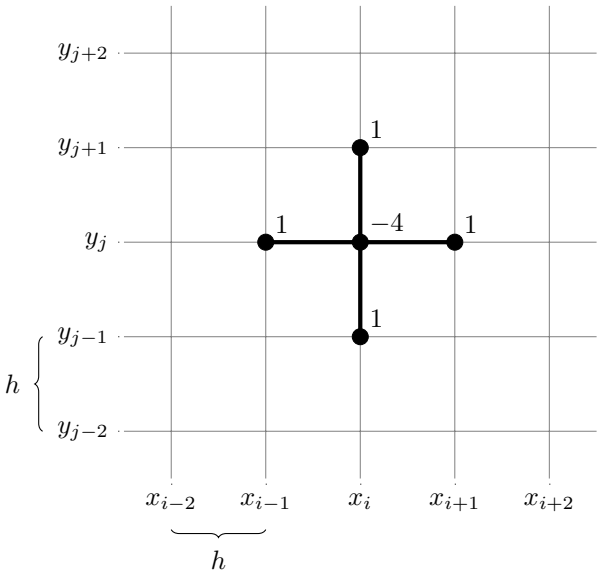

Figure 5: The 5-point stencil for the Laplacian about the point (i, j) is indicated.

We have both an unknown u_{ij} and an equation of the form (3.12) at each of N^2 grid points for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, N$, where $h = \dfrac{1}{N+1}$ as in one dimension. We thus have a linear system of N^2 unknowns. The difference equations at points near the boundary will of course involve the known boundary values, just as in the one-dimensional case, which can be moved to the right-hand side.

3.3 Ordering The Unknowns and Equations

If we collect all these equations together into a matrix equation, we will have an $N^2 \times N^2$ matrix that is very sparse, i.e., most of the elements are zero. Since each equation involves at most five unknowns (fewer near the boundary), each row of the matrix has at most five nonzeros and at least $N^2 - 5$ elements that are zero. This is analogous to the tridiagonal matrix seen in the one-dimensional case, in which each row has at most three nonzeros.

Recall that the structure of the matrix depends on the order we choose to enumerate the unknowns. Unfortunately, in two space dimensions the structure of the matrix is not as compact as in one dimension, no matter how we order the unknowns, and the nonzeros cannot be as nicely clustered near the main diagonal.

One obvious choice is the *natural rowwise ordering*, where we take the unknowns along the bottom row, $u_{11}, u_{21}, \ldots, u_{N1}$, followed by the unknowns in the second row, $u_{12}, u_{22}, \ldots, u_{N2}$, and so on, as illustrated in Figure 6.

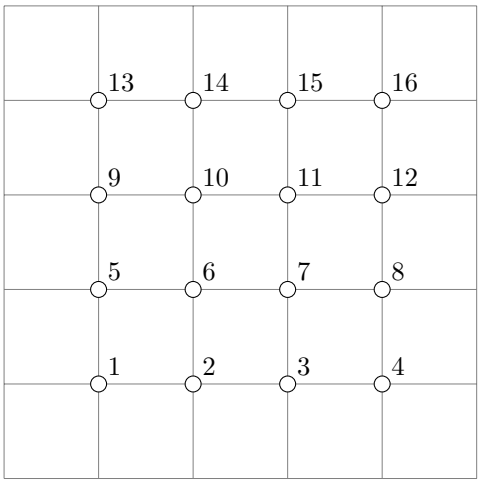

Figure 6: The natural rowwise order of unknowns and equations on a 4×4 grid.

The vector of unknowns is partitioned as

$$u = \begin{bmatrix} u^{[1]} \\ u^{[2]} \\ \vdots \\ u^{[N]} \end{bmatrix} \tag{3.13}$$

where

$$u^{[j]} = \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{Nj} \end{bmatrix} \tag{3.14}$$

This gives a matrix equation where A^h has the form

$$A^h = \frac{1}{h^2} \begin{bmatrix} T^h & I_N & & & \\ I_N & T^h & I_N & & \\ & I_N & T^h & I_N & \\ & & \ddots & \ddots & \ddots \\ & & & I_N & T^h \end{bmatrix} \tag{3.15}$$

which is an $N \times N$ *block tridiagonal matrix* in which each block T^h or I_N is itself $N \times N$ matrix,

$$T^h = \begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & 1 & -4 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -4 \end{bmatrix} \tag{3.16}$$

and I_N is the $N \times N$ identity matrix. While this has a nice structure, the 1 values in the I_N matrices are separated from the diagonal by $N-1$ zeros, since

these coefficients correspond to grid points lying above or below the central point in the stencil and hence are in the next or previous row of unknowns.

Another possibility, which has some advantages in the context of certain iterative methods, is to use the *red-black ordering* (or checkboard ordering) shown in Figure 7.

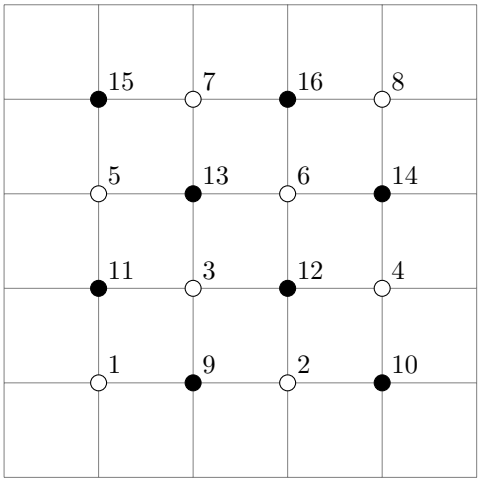

Figure 7: The red-black order of unknowns and equations on a 4×4 grid.

This is the two-dimensional analogue of the odd-even ordering in one dimension. This ordering is significant because all four neighbors of a red grid point are black points, and vice versa, and it leads to a matrix equation with the structure

$$\begin{bmatrix} D & H \\ H^T & D \end{bmatrix} \begin{bmatrix} u_{red} \\ u_{black} \end{bmatrix} = \begin{bmatrix} f_{red} \\ -f_{black} \end{bmatrix} \tag{3.17}$$

where $D = -\dfrac{4}{h^2} I$ is a diagonal matrix of dimension $\dfrac{N^2}{2}$ and H is a banded matrix of the same dimension with four nonzero diagonals.

When direct methods such as Gaussian elimination are used to solve the system, one typically wants to order the equations and unknowns so as to reduce the amount of fill-in during the elimination procedure as much as possible. This is done automatically if the backslash operator in MATLAB is used to solve the system, provided it is set up using sparse storage.

3.4 Accuracy and Stability

The discretization of the two-dimensional Poisson problem can be analyzed using exactly the same approach as we used for the one-dimensional boundary value problem. The local truncation error τ_{ij} at the (i, j) grid point is defined in the obvious way,

$$\tau_{ij} = \frac{1}{h^2} \begin{pmatrix} u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) \\ +u(x_i, y_{j+1}) - 4u(x_i, y_j) \end{pmatrix} - f(x_i, y_j) \tag{3.18}$$

and by splitting this into the second order difference in the x- and y-directions,

$$u(x_{i-1}, y_j) + u(x_{i+1}, y_j) - 2u(x_i, y_j) \tag{3.19}$$

$$= h^2 u_{xx}(x_i, y_j) + \frac{1}{12} h^4 u_{xxxx}(x_i, y_j) + O\left(h^6\right) \tag{3.20}$$

$$u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 2u(x_i, y_j) \tag{3.21}$$

$$= h^2 u_{yy}(x_i, y_j) + \frac{1}{12} h^4 u_{yyyy}(x_i, y_j) + O\left(h^6\right) \tag{3.22}$$

Hence,

$$\tau_{ij} = u_{xx}(x_i, y_j) + u_{yy}(x_i, y_j) - f(x_i, y_j) \tag{3.23}$$

$$+ \frac{1}{12} h^2 u_{xxxx}(x_i, y_j) + \frac{1}{12} h^2 u_{yyyy}(x_i, y_j) + O\left(h^4\right) \tag{3.24}$$

$$= \frac{1}{12} h^2 \left(u_{xxxx}(x_i, y_j) + u_{yyyy}(x_i, y_j)\right) + O\left(h^4\right) \tag{3.25}$$

For this linear system of equations the global error $E_{ij} = u_{ij} - u(x_i, y_j)$ then solves the linear system

$$A^h E^h = -\tau^h \tag{3.26}$$

just as in one dimension, where A^h is now the discretization matrix with mesh spacing h, e.g., the matrix (3.15) if the rowwise ordering is used. The method will be globally second order accurate in some norm provided that it is stable, i.e., that $\left\|\left(A^h\right)^{-1}\right\|$ is uniformly bounded as $h \to 0$.

In the 2-norm this is again easy to check for this simple problem, since we can explicitly compute the spectral radius of the matrix, as we did in one dimension. The eigenvalues and eigenvectors of A can now be indexed by two parameters p and q corresponding to wave numbers in the x- and y-directions for $p, q = 1, 2, \ldots, N$. The (p, q) eigenvector $u^{p,q}$ has the N^2 elements

$$u_{ij}^{p,q} = \sin(p\pi i h) \sin(q\pi j h) \tag{3.27}$$

The corresponding eigenvalue is

$$\lambda_{p,q} = \frac{2}{h^2} \left(\left(\cos(p\pi h) - 1\right) + \left(\cos(q\pi h) - 1\right)\right) \tag{3.28}$$

The eigenvalues are strictly negative (A is negative definite) and the one closest to the origin is

$$\lambda_{1,1} = \frac{2}{h^2} \left(\left(\cos(\pi h) - 1\right) + \left(\cos(\pi h) - 1\right)\right) \tag{3.29}$$

$$= \frac{4}{h^2} \left(-\frac{\pi^2 h^2}{2} + O\left(h^4\right)\right) \tag{3.30}$$

$$= -2\pi^2 + O\left(h^2\right) \tag{3.31}$$

The spectral radius of $\left(A^h\right)^{-1}$, which is also the 2-norm, is thus

$$\left\|\left(A^h\right)^{-1}\right\|_2 = \rho\left(\left(A^h\right)^{-1}\right) \tag{3.32}$$

$$= \max_{1\leq p,q\leq N} \frac{1}{|\lambda_{p,q}|} \tag{3.33}$$

$$= \frac{1}{\min_{1\leq p,q\leq N} |\lambda_{p,q}|} \tag{3.34}$$

$$= \frac{1}{|\lambda_{1,1}|} \tag{3.35}$$

$$= \frac{1}{2\pi^2 + O\left(h^2\right)} \tag{3.36}$$

$$\approx \frac{1}{2\pi^2} \tag{3.37}$$

Hence, the method is stable in the 2-norm. While we are at it, let's also compute the condition number of the matrix A^h, since it turns out that this is a critical quantity in determining how rapidly certain iterative methods converge. Recall that the 2-norm condition number is defined by

$$\kappa_2\left(A\right) = \|A\|_2 \left\|A^{-1}\right\|_2 \tag{3.38}$$

We have just seen that

$$\left\|\left(A^h\right)^{-1}\right\|_2 \approx \frac{1}{2\pi^2} \tag{3.39}$$

for small h, and the norm of A is given by its spectral radius. The largest eigenvalue of A (in magnitude) is

$$\lambda_{N,N} = \frac{2}{h^2}\left(\left(\cos\left(N\pi h\right) - 1\right) + \left(\cos\left(N\pi h\right) - 1\right)\right) \tag{3.40}$$

$$= \frac{4}{h^2}\left(-\frac{N^2\pi^2h^2}{2} + O\left(h^4\right)\right) \tag{3.41}$$

$$= -2N^2\pi^2 + O\left(h^2\right) \tag{3.42}$$

$$= -2\left(\frac{1}{h} - 1\right)^2\pi^2 + O\left(h^2\right) \tag{3.43}$$

Hence,

$$\kappa_2\left(A^h\right) = \left\|A^h\right\|_2 \left\|\left(A^h\right)^{-1}\right\|_2 \tag{3.44}$$

$$\approx \frac{1}{2\pi^2} \cdot 2\left(\frac{1}{h} - 1\right)^2\pi^2 \tag{3.45}$$

$$= \frac{1}{h^2} - \frac{2}{h} + 1 \tag{3.46}$$

$$\leq \frac{1}{h^2} + \frac{2}{h^2} + \frac{1}{h^2} \tag{3.47}$$

$$= O\left(\frac{1}{h^2}\right) \text{ as } h \to 0 \tag{3.48}$$

The fact that the matrix becomes very ill-conditioned as we refine the grid is responsible for the slow-down of iterative methods.

3.5 The 9-Point Laplacian

Above we used the 5-point Laplacian, which we will denote by

$$\nabla_5^2 u_{ij} := \frac{1}{h^2}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}\right) \tag{3.49}$$

Another possible approximation is the 9-point Laplacian

$$\nabla_9^2 u_{ij} := \frac{1}{6h^2}\left(\begin{array}{l} 4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} + u_{i-1,j-1} \\ +u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{ij} \end{array}\right) \tag{3.50}$$

as indicated in Figure 8.

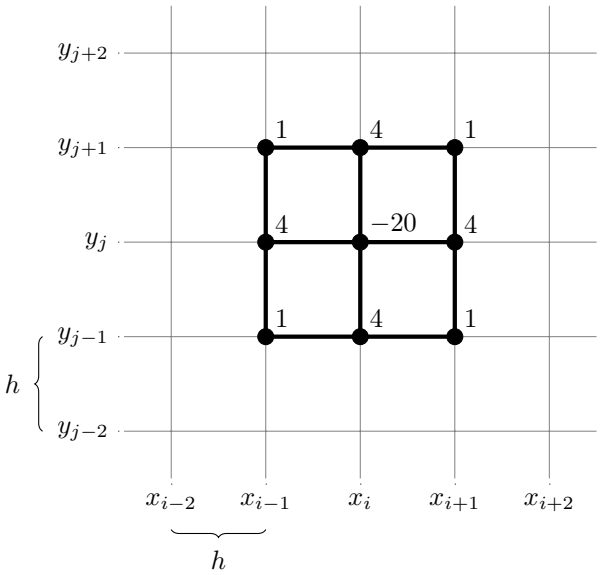

Figure 8: The 9-point stencil for the Laplacian about the point (i, j) is indicated.

If we apply this to the true solution and expand in Taylor series, this is (2.407).

$$\nabla_9^2 u\left(x_i, y_j\right) = \Delta u\left(x_i, y_j\right) + \frac{1}{12}h^2 \Delta^2 u\left(x_i, y_j\right) + O\left(h^4\right) \tag{3.51}$$

At first glance this discretization looks no better than the 5-point discretizaton since the error is still $O\left(h^2\right)$. However, the additional terms lead to a very nice form for the dominant error term, i.e., $\Delta^2 u\left(x_i, y_j\right)$.

If we are solving $\Delta u = f$, then we have

$$\Delta^2 u = \Delta\left(\Delta u\right) \tag{3.52}$$

$$= \Delta f \tag{3.53}$$

Hence, we can compute the dominant term in the truncation error easily from the known function f without knowing the true solution u to the problem.

In particular, if we are solving Laplace's equation, where $f = 0$, or more generally if f is a harmonic function, then this term in the local truncation

error vanishes and the 9-point Laplacian would give a fourth order accurate discretization of the differential equation. More generally, we can obtain a fourth order accurate method of the form

$$\nabla_9^2 u_{ij} = f_{ij} \tag{3.54}$$

for arbitrary smooth functions $f(x, y)$ by defining

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12} \Delta f(x_i, y_j) \tag{3.55}$$

We can view this as deliberately introducing an $O\left(h^2\right)$ error into the right-hand side of the equation that is chosen to cancel the $O\left(h^2\right)$ part of the local truncation error. Taylor series expansion easily shows that the local truncation error of the method (3.54) is now $O\left(h^4\right)$.

$$\begin{aligned}
\tau_{ij} &= \nabla_9^2 u_{ij} - f_{ij} && (3.56)\\
&= \left(\Delta u(x_i, y_j) + \frac{1}{12} h^2 \Delta^2 u(x_i, y_j) + O\left(h^4\right) \right) && (3.57)\\
&- \left(f(x_i, y_j) + \frac{h^2}{12} \Delta f(x_i, y_j) \right) && (3.58)\\
&= \left(\Delta u(x_i, y_j) - f(x_i, y_j) \right) && (3.59)\\
&+ \frac{h^2}{12} \left(\Delta^2 u(x_i, y_j) - \Delta f(x_i, y_j) \right) + O\left(h^4\right) && (3.60)\\
&= O\left(h^4\right) && (3.61)
\end{aligned}$$

This is the two-dimensional analogue of the modification [1], p.55 that gives fourth order accuracy for the boundary value problem $u''(x) = f(x)$.

If we have only data $f(x_i, y_j)$ at the grid points (but we know that the underlying function is sufficiently smooth), then we can still achieve fourth order accuracy by using

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12} \nabla_5^2 f(x_i, y_j) \tag{3.62}$$

instead of (3.55).

Indeed, we have

$$\begin{aligned}
\nabla_5^2 f(x_i, y_j) &= \frac{1}{h^2} \begin{pmatrix} f(x_{i-1}, y_j) + f(x_{i+1}, y_j) + f(x_i, y_{j-1}) \\ + f(x_i, y_{j+1}) - 4f(x_i, y_j) \end{pmatrix} && (3.63)\\
&= \Delta f(x_i, y_j) + O\left(h^2\right) && (3.64)
\end{aligned}$$

Combining (3.62) and (3.63)-(3.64) yields

$$\begin{aligned}
\tau_{ij} &= \nabla_9^2 u_{ij} - f_{ij} && (3.65)\\
&= \left(\Delta u(x_i, y_j) + \frac{h^2}{12} \Delta^2 u(x_i, y_j) + O\left(h^4\right) \right) && (3.66)\\
&- \left(f(x_i, y_j) + \frac{h^2}{12} \nabla_5^2 f(x_i, y_j) \right) && (3.67)
\end{aligned}$$

$$= \Delta u(x_i, y_j) + \frac{h^2}{12}\Delta^2 u(x_i, y_j) + O\left(h^4\right) \tag{3.68}$$

$$- f(x_i, y_j) - \frac{h^2}{12}\left(\Delta f(x_i, y_j) + O\left(h^2\right)\right) \tag{3.69}$$

$$= \left(\Delta u(x_i, y_j) - f(x_i, y_j)\right) \tag{3.70}$$

$$+ \frac{h^2}{12}\left(\Delta^2 u(x_i, y_j) - \Delta f(x_i, y_j)\right) + O\left(h^4\right) \tag{3.71}$$

$$= O\left(h^4\right) \tag{3.72}$$

This is a trick that often can be used in developing numerical methods - introducing an "error" into the equations that is carefully chosen to cancel some other error.

Note that the same trick would not work with the 5-point Laplacian, or at least not as directly. The form of the truncation error in this method depends on $u_{xxxx} + u_{yyyy}$. There is no way to compute this directly from the original equation without knowing u. The extra points in the 9-point stencil convert this into the Laplacian of f, which can be computed if f is sufficiently smooth.

On the other hand, a two-pass approach could be used with the 5-point stencil, in which we first estimate u by solving with the standard 5-point scheme to get a second order accurate estimate of u. We then use this estimate of u to approximate $u_{xxxx} + u_{yyyy}$ and then solve a second time with a right-hand side that is modified to eliminate the dominant term of the local truncation error. This would be more complicated for this particular problem, but this idea can be used much more generally than the above trick, which depends on the special form of the Laplacian. This is the method of *deferred corrections*, see [1], Section 2.20.3.

4 Other Elliptic Equations

In [1] Chapter 2, the author start with the simplest boundary value problem for the constant coefficient problem $u''(x) = f(x)$ but then introduced various, more interesting problems, such as variable coefficients, nonlinear problems, singular perturbation problems, and boundary of interior layers.

In the multidimensional case we have discussed only the simplest Poisson problem, which in one dimension reduces to $u''(x) = f(x)$. All the further complications seen in one dimension can also arise in multidimensional problems.

For example, heat conduction in a heterogeneous two-dimensional domain gives rise to the equation

$$\left(\kappa(x, y) u_x(x, y)\right)_x + \left(\kappa(x, y) u_y(x, y)\right)_y = f(x, y) \tag{4.1}$$

where $\kappa(x, y)$ is the varying heat conduction coefficient. In any number of space dimensions this equation can be written as

$$\nabla \cdot (\kappa \nabla u) = f \tag{4.2}$$

These problems can be solved by generalizations of the one-dimensional methods. The terms $\left(\kappa(x, y) u_x(x, y)\right)_x$ and $\left(\kappa(x, y) u_y(x, y)\right)_y$ can each be discretized as in the one-dimensional case, again resulting in a 5-point stencil in two dimensions.

Nonlinear elliptic equations also arise in multidimensions, in which case a system of nonlinear algebraic equations will result from the discretization. A Newton method can be used as in one dimension, but now in each Newton iteration a large sparse linear system will have to be solved. Typically the Jacobian matrix has a sparsity pattern similar to those seen above for linear elliptic equations. See [1]-Section 4.5, for a brief discussion of Newton-Krylov iterative methods for such problems.

In multidimensional problems there is an additional potential complication that is not seen in one dimension: the domain Ω where the boundary value problem is posed may not be a simple rectangle as we have supposed in our discussion so far. When the solution exhibits boundary or interior layers, then we would also like cluster grid points or adaptively refine the grid in these regions. This often presents a significant challenge.

5 Appendices

5.1 Determinants of Block Tridiagonal Matrices

See [3].

For ordinary tridiagonal matrices, determinants can be evaluated via multiplication of 2×2 matrices

$$\det \begin{bmatrix} a_1 & b_1 & 0 & \cdots & c_0 \\ c_1 & a_2 & b_2 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & c_{n-2} & a_{n-1} & b_{n-1} \\ b_n & \cdots & 0 & c_{n-1} & a_n \end{bmatrix} \tag{5.1}$$

$$= (-1)^{n+1} \left(b_n \cdots b_1 + c_{n-1} \cdots c_0 \right) \tag{5.2}$$

$$+ tr \left[\begin{pmatrix} a_n & -b_{n-1}c_{n-1} \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_2 & -b_1 c_1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & -b_n c_0 \\ 1 & 0 \end{pmatrix} \right] \tag{5.3}$$

and

$$\det \begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-2} & a_{n-1} & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix} \tag{5.4}$$

$$= \left[\begin{pmatrix} a_n & -b_{n-1}c_{n-1} \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_2 & -b_1 c_1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & 0 \\ 1 & 0 \end{pmatrix} \right]_{11} \tag{5.5}$$

Usage 5.1. (5.4)-(5.5) is used to compute the determinant of the main diagonal block matrices, i.e., compute $\det T^{h,k}$ and $\det T^{h}$.

5.1.1 The Duality Relation

Consider the following block-tridiagonal matrix $M(z)$ with blocks A_i, B_i and C_{i-1}, $i = 1, 2, \ldots, n$ that are complex $m \times m$ matrices. It is very useful to

introduce also a complex parameter z in the corner blocks.

$$M(z) = \begin{bmatrix} A_1 & B_1 & 0 & \cdots & \frac{1}{z}C_0 \\ C_1 & A_2 & B_2 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & C_{n-2} & A_{n-1} & B_{n-1} \\ zB_n & \cdots & 0 & C_{n-1} & A_n \end{bmatrix} \tag{5.6}$$

It is required that off-diagonal blocks are nonsingular:

$$\det B_i \neq 0, \det C_{i-1} \neq 0, \forall i \tag{5.7}$$

The matrix is naturally associated with a *transfer matrix*, built as the product of n matrices of size $2m \times 2m$.

$$T = \begin{bmatrix} -B_n^{-1}A_n & -B_n^{-1}C_{n-1} \\ I_m & 0 \end{bmatrix} \cdots \begin{bmatrix} -B_1^{-1}A_1 & -B_1^{-1}C_0 \\ I_m & 0 \end{bmatrix} \tag{5.8}$$

where I_m is the $m \times m$ unit matrix. The transfer matrix is nonsingular, since

$$\det T = \prod_{i=1}^{n} \det \left[B_i^{-1} C_{i-1} \right] \tag{5.9}$$

Lemma 5.2.

$$\det M(z) = \frac{(-1)^{nm}}{(-z)^m} \det (T - zI_{2m}) \det (B_1 \cdots B_n) \tag{5.10}$$

Theorem 5.3 (Duality Relation).

$$\det (\lambda I_{nm} - M(z)) = (-z)^{-m} \det (T(\lambda) - zI_{2m}) \det (B_1 \cdots B_n) \tag{5.11}$$

It shows that the parameter z, which enters in $M(z)$ as a boundary term, is related to eigenvalues of the matrix $T(\lambda)$ that connects the eigenvector of $M(z)$ at the boundaries.

5.1.2 Block Tridiagonal Matrix with No Corners

By a modification off the proof of the Lemma 4.1, one obtains an identity for the determinant of block-tridiagonal matrices $M^{(0)}$ with no corners $B_n = C_0 = 0$ in the matrix (5.6).

Theorem 5.4.

$$\det M^{(0)} = (-1)^{nm} \det T_{11}^{(0)} \det (B_1 \cdots B_{n-1}) \tag{5.12}$$

where $T_{11}^{(0)}$ is the upper left block of size $m \times m$ of the transfer matrix

$$T^{(0)} = \begin{bmatrix} -A_n & -C_{n-1} \\ I_m & 0 \end{bmatrix} \begin{bmatrix} -B_{n-1}^{-1}A_{n-1} & -B_{n-1}^{-1}C_{n-2} \\ I_m & 0 \end{bmatrix} \cdots \begin{bmatrix} -B_1^{-1}A_1 & -B_1^{-1} \\ I_m & 0 \end{bmatrix} \tag{5.13}$$

Usage 5.5. (5.12)-(5.13) is used to compute the determinant of tridiagonal block matrices $A^{h,k}$ and A^h.

The formula for the evaluation of $\det M^{(0)}$ requires $n-1$ inversions B_k^{-1}, multiplication of n matrices of size $2m \times 2m$, and the final evaluation of a determinant. Salkuyeh proposed a different procedure for the evaluation of the same determinant.

$$\det M^{(0)} = \prod_{k=1}^{n} \det \Lambda_k \tag{5.14}$$

$$\Lambda_1 = A_1 \tag{5.15}$$

$$\Lambda_k = A_k - C_{k-1}\Lambda_{k-1}^{-1}B_{k-1} \tag{5.16}$$

Usage 5.6. (5.14)-(5.16) is used to compute the determinant of tridiagonal block matrices $A^{h,k}$ and A^h.

It requires $n-1$ inversions of matrices of size $m \times m$, and the evaluation of their determinants. [3] show that the two procedures are related.

The transfer matrix $T^{(0)} = T(n)$ is the product of n matrices. Let $T(k)$ be the partial product of k matrices. Then

$$T(k) = \begin{bmatrix} -B_k^{-1}A_k & -B_k^{-1}C_{k-1} \\ I_m & 0 \end{bmatrix} T(k-1) \tag{5.17}$$

This produces a two-term recurrence relation for blocks

$$T(k)_{11} = -B_k^{-1}A_k T(k-1)_{11} - B_k^{-1}C_{k-1}T(k-2)_{11} \tag{5.18}$$

with $T(1)_{11} = -B_1^{-1}A_1, T(0)_{11} = I_m$. The equations by Salkuyeh result for

$$\Lambda_k = -B_k T(k)_{11}[T(k-1)_{11}]^{-1} \tag{5.19}$$

5.2 Analytical Inversion of General Tridiagonal Matrices

See [4].

5.2.1 Block Case

Consider the block tridiagonal matrix

$$A = \begin{bmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & A_{n-1} & B_{n-1} & C_{n-1} \\ & & & A_n & B_n \end{bmatrix} \tag{5.20}$$

and A_j, B_j and C_j are $m \times m$ matrices.

Theorem 5.7. *Define the second-order block recurrences*

$$C_i Z_i = B_i Z_{i-1} - A_i Z_{i-2}, \quad i = 2, 3, \ldots, n \tag{5.21}$$

where $Z_0 = I, Z_1 = C_1^{-1}B_1$ *and*

$$A_jY_j = B_jY_{j+1} - C_jY_{j+2}, j = n-1, n-2, \ldots, 1 \tag{5.22}$$

where $Y_{n+1} = I, Y_n = A_n^{-1}B_n$. *The inverse matrix* $A^{-1} = \{\Phi_{i,j}\}\,(1 \le i, j \le n)$ *can be expressed as*

$$\Phi_{j,j} = \left(B_j - A_jZ_{j-2}Z_{j-1}^{-1} - C_jY_{j+2}Y_{j+2}^{-1}\right)^{-1} \tag{5.23}$$

where $j = 1, 2, \ldots, n, A_1 = 0, C_n = 0$ *and*

$$\Phi_{i,j} = \begin{cases} -Z_{i-1}Z_i^{-1}\Phi_{i+1,j}, i < j \\ -Y_{i+1}Y_i^{-1}\Phi_{i-1,j}, i > j \end{cases} \tag{5.24}$$

Corollary 5.8. *The inverse matrix* $A^{-1} = \{\Phi_{i,j}\}$ *can be expressed as*

$$\Phi_{j,j} = \left(B_j - A_jZ_{j-2}Z_{j-1}^{-1} - C_jY_{j+2}Y_{j+2}^{-1}\right)^{-1} \tag{5.25}$$

where $j = 1, 2, \ldots, n, A_1 = 0, C_n = 0$ *and*

$$\Phi_{i,j} = \begin{cases} (-1)^{j-i}Z_{i-1}Z_{j-1}^{-1}\Phi_{j,j}, i < j \\ (-1)^{i-j}Y_{i+1}Y_{j+1}^{-1}\Phi_{j,j}, i > j \end{cases} \tag{5.26}$$

5.2.2 Block Toeplitz Case

As a straightforward extension of Theorem 5.7, we now consider the inversion of a block matrix

$$T = \begin{bmatrix} B & C & & & \\ A & B & C & & \\ & \ddots & \ddots & \ddots & \\ & & A & B & C \\ & & & A & B \end{bmatrix} \tag{5.27}$$

where A, B and C are $m \times m$ matrices. Let Λ_1 and Λ_2 be $m \times m$ matrices such that

$$\Lambda_1 + \Lambda_2 = C^{-1}B \tag{5.28}$$

$$\Lambda_1\Lambda_2 = C^{-1}A \tag{5.29}$$

Let Δ_1 and Δ_2 be $m \times m$ matrices such that

$$\Delta_1 + \Delta_2 = A^{-1}B \tag{5.30}$$

$$\Delta_1\Delta_2 = A^{-1}C \tag{5.31}$$

Define sequences Z_i and $Y_i, i = 0, 1, \ldots, n$ to be

$$Z_i = \sum_{k=0}^{i} \Lambda_2^{i-k}\Lambda_1^k \tag{5.32}$$

and

$$Y_i = \sum_{k=0}^{n+1-i} \Delta_2^{n+1-i-k} \Delta_1^k \tag{5.33}$$

then we have the following result.

Theorem 5.9. *The inverse of matrix T can be expressed in the following explicit way*

$$\left(T^{-1}\right)_{j,j} = \left(B - AZ_{j-2}Z_{j-1}^{-1} - CY_{j+2}Y_{j+1}^{-1}\right)^{-1} \tag{5.34}$$

where $j = 1, 2, \ldots, n$ and

$$\left(T^{-1}\right)_{i,j} = \begin{cases} (-1)^{j-i} Z_{i-1} Z_{j-1}^{-1} \left(T^{-1}\right)_{j,j}, i < j \\ (-1)^{i-j} Y_{i+1} Y_{j+1}^{-1} \left(T^{-1}\right)_{j,j}, i > j \end{cases} \tag{5.35}$$

When $A = \epsilon C$, where ϵ is a constant factor, we have the following results.

Theorem 5.10. *If $A = \epsilon C$, the inverse of the matrix T can be determined by the following explicit formula*

$$\left(T^{-1}\right)_{i,j} = (-1)^{i-j} \left(\Psi_1^j - \Psi_2^j\right) \left(\Psi_1^{n+1-i} - \Psi_2^{n+1-i}\right) \times \tag{5.36}$$

$$\times \left(\Psi_1 - \Psi_2\right)^{-1} \left(\Psi_1^{n+1} - \Psi_2^{n+1}\right)^{-1} C^{-1}, i < j \tag{5.37}$$

$$\left(T^{-1}\right)_{i,j} = (-\epsilon)^{i-j} \left(\Psi_1^i - \Psi_2^i\right) \left(\Psi_1^{n+1-j} - \Psi_2^{n+1-j}\right) \times \tag{5.38}$$

$$\times \left(\Psi_1 - \Psi_2\right)^{-1} \left(\Psi_1^{n+1} - \Psi_2^{n+1}\right)^{-1} C^{-1}, i \geq j \tag{5.39}$$

where ϵ_1 and ϵ_2 are $m \times m$ matrices that satisfy

$$\Psi_1 + \Psi_2 = C^{-1}B \tag{5.40}$$

$$\Psi_1 \Psi_2 = \epsilon I \tag{5.41}$$

Solution of Equation (5.40)-(5.41). We now find the explicit solution of equation (5.40)-(5.41) in Theorem 5.10. For simplicity, suppose $C^{-1}B$ is non-defective. Let $\lambda_1, \lambda_2, \ldots, \lambda_m$ be the eigenvalues of $C^{-1}B$ and Q be the matrix of eigenvectors such that

$$C^{-1}B = Q diag_m \left(\lambda_k\right) Q^{-1} \tag{5.42}$$

where $diag_m \left(\lambda_k\right)$ is defined to be $m \times m$ diagonal matrix with $\lambda_k, k = 1, 2, \ldots, m$ being the diagonal element. Set the solution of (5.40)-(5.41) to be

$$\Psi_1 = Q diag \left(\lambda_k^+\right) Q^{-1} \tag{5.43}$$

$$\Psi_2 = Q diag \left(\lambda_k^-\right) Q^{-1} \tag{5.44}$$

then we have

$$\lambda_k^+ + \lambda_k^- = \lambda_k \tag{5.45}$$

$$\lambda_k^+ \lambda_k^- = \epsilon \tag{5.46}$$

or in other words, λ_k^+ and λ_k^- are the roots of

$$r^2 - \lambda_k r + \epsilon = 0 \tag{5.47}$$

We have the following results.

1. Case $\lambda_k^2 > 4\epsilon$. We have

$$\left(\lambda_k^+\right)^i - \left(\lambda_k^-\right)^i = 2\epsilon^{\frac{i}{2}} \sinh i\theta_k \tag{5.48}$$

where

$$2\epsilon^{\frac{1}{2}} \cosh\theta_k = \lambda_k \tag{5.49}$$

2. Case $\lambda_k^2 \leq 4\epsilon$. We have

$$\left(\lambda_k^+\right)^i - \left(\lambda_k^-\right)^i = 2\epsilon^{\frac{i}{2}} \sin i\theta_k \tag{5.50}$$

where

$$2\epsilon^{\frac{1}{2}} \cos\theta_k = \lambda_k \tag{5.51}$$

Theorem 5.11. *If $A = \epsilon C$, the inverse of the matrix T can be determined by the following explicit formula.*

$$\left(T^{-1}\right)_{i,j} \tag{5.52}$$

$$= (-1)^{j-i}\epsilon^{\frac{j-i-1}{2}} Qdiag_m \left\{ \frac{\sinh\left(j\theta_k\right)\sinh\left(n+1-i\right)\theta_k}{\sinh\theta_k \sinh\left(\left(n+1\right)\theta_k\right)} \right\} Q^{-1}C^{-1}, \tag{5.53}$$

$$\text{if } i < j \tag{5.54}$$

$$\left(T^{-1}\right)_{i,j} \tag{5.55}$$

$$= (-1)^{i-j}\epsilon^{\frac{i-j-1}{2}} Qdiag_m \left\{ \frac{\sinh\left(i\theta_k\right)\sinh\left(n+1-j\right)\theta_k}{\sinh\theta_k \sinh\left(\left(n+1\right)\theta_k\right)} \right\} Q^{-1}C^{-1}, \tag{5.56}$$

$$\text{if } i \geq j \tag{5.57}$$

where Q and θ_k satisfy

$$C^{-1}B = Qdiag_m\left\{\lambda_k\right\}Q^{-1} \tag{5.58}$$

$$2\epsilon^{\frac{1}{2}} \cosh\theta_k = \lambda_k \tag{5.59}$$

If $\lambda_k^2 \leq 4\epsilon, 1 \leq k \leq m$, the hyperbolic sines and cosines in (5.52) *and* (5.57) *becomes sines and cosines, respectively.*

Usage 5.12. Thereom 5.8 is used to compute the inverse matrix of the tridiagonal block matrices $A^{h,k}$ and A^h.

We can rewrite (5.52)-(5.57) more compactly as the following formula.

$$\left(T^{-1}\right)_{i,j} \tag{5.60}$$

$$= (-1)^{\max\{i,j\}-\min\{i,j\}} \varepsilon^{\frac{\max\{i,j\}-\min\{i,j\}-1}{2}} Q\times \tag{5.61}$$

$$\times\, diag_m \left\{ \frac{\sinh\left(\max\{i,j\}\,\theta_k\right)\sinh\left(n+1-\min\{i,j\}\,\theta_k\right)}{\sinh\theta_k \sinh\left((n+1)\,\theta_k\right)} \right\} Q^{-1}C^{-1} \tag{5.62}$$

$$= (-1)^{\max\{i,j\}+\min\{i,j\}} \varepsilon^{\frac{\max\{i,j\}-\min\{i,j\}-1}{2}} Q\times \tag{5.63}$$

$$\times\, diag_m \left\{ \frac{\sinh\left(\max\{i,j\}\,\theta_k\right)\sinh\left(n+1-\min\{i,j\}\,\theta_k\right)}{\sinh\theta_k \sinh\left((n+1)\,\theta_k\right)} \right\} Q^{-1}C^{-1} \tag{5.64}$$

$$= (-1)^{i+j} \varepsilon^{\frac{\max\{i,j\}-\min\{i,j\}-1}{2}} Q\times \tag{5.65}$$

$$\times\, diag_m \left\{ \frac{\sinh\left(\max\{i,j\}\,\theta_k\right)\sinh\left(n+1-\min\{i,j\}\,\theta_k\right)}{\sinh\theta_k \sinh\left((n+1)\,\theta_k\right)} \right\} Q^{-1}C^{-1} \tag{5.66}$$

Can you continue to simplify (5.60)-(5.66)?

5.3 Tridiagonal Toeplitz Matrices

See [5].

The tridiagonal Toeplitz matrix

$$T = \begin{bmatrix} \delta & \tau & & & \\ \sigma & \delta & \tau & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma & \delta & \tau \\ & & & \sigma & \delta \end{bmatrix} \in \mathbb{C}^{n\times n} \tag{5.67}$$

is denoted by $T = (n; \sigma, \delta, \tau)$, and we let

$$\alpha = \arg\sigma \tag{5.68}$$

$$\beta = \arg\tau \tag{5.69}$$

$$\gamma = \arg\delta \tag{5.70}$$

It is well that the eigenvalues of $T = (n; \sigma, \delta, \tau)$ are given by

$$\lambda_h(T) = \delta + 2\sqrt{\sigma\tau}\cos\frac{h\pi}{n+1}, \quad h = 1, 2, \ldots, n \tag{5.71}$$

Usage 5.13. (5.71) is used to compute the eigenvalues of the main diagonal block matrices $T^{h,k}$ and T^h.

and using (5.68)-(5.70), we obtain

$$\lambda_h(T) = \delta + 2\sqrt{|\sigma\tau|}e^{\frac{i(\alpha+\beta)}{2}}\cos\frac{h\pi}{n+1}, h = 1, 2, \ldots, n \tag{5.72}$$

In particular, if $\sigma\tau \neq 0$, the matrix (5.67) has n simple eigenvalues, which lie on the closed line segment

$$\mathcal{S}_{\lambda(T)} = \left\{ \delta + te^{\frac{i(\alpha+\beta)}{2}} : t \in R, |t| \leq 2\sqrt{|\sigma\tau|}\cos\frac{\pi}{n+1} \right\} \subset \mathbb{C} \tag{5.73}$$

Usage 5.14. (5.73) is used to compute the spectrum of the main diagonal block matrix $T^{h,k}$ and T^h.

The eigenvalues are allocated symmetrically with respect to δ.
The spectral radius of the matrix (5.67) is given by

$$\rho(T) = \max\left\{ \left| \delta + 2\sqrt{|\sigma\tau|}e^{\frac{i(\alpha+\beta)}{2}}\cos\frac{\pi}{n+1} \right|, \left| \delta + 2\sqrt{|\sigma\tau|}e^{\frac{i(\alpha+\beta)}{2}}\cos\frac{n\pi}{n+1} \right| \right\} \tag{5.74}$$

Usage 5.15. (5.74) is used to compute spectral radius of $T^{h,k}$ and T^h.

and, if T is nonsingular, i.e., $\lambda_h(T) \neq 0$ for all $h = 1, 2, \ldots, n$, taking (5.72) into account, one has

$$\rho\left(T^{-1}\right) = \max_{1\leq h\leq n} \left| \delta + 2\sqrt{|\sigma\tau|}e^{\frac{i(\alpha+\beta)}{2}}\cos\frac{h\pi}{n+1} \right|^{-1} \tag{5.75}$$

Usage 5.16. (5.75) is used to compute the spectral radius of the inverse matrix of the main diagonal block matrix $T^{h,k}$ and T^h.

When $\sigma\tau \neq 0$, the components of the $x_h = [x_{h,1}, x_{h,2}, \ldots, x_{h,n}]^T$ associated with the eigenvalue $\lambda_h(T)$ are given by

$$x_{h,k} = \left(\frac{\sigma}{\tau}\right)^{\frac{k}{2}} \sin\frac{hk\pi}{n+1}, \quad k = 1:n, h = 1:n \tag{5.76}$$

Usage 5.17. (5.76) is used to compute the right eigenvector of the main diagonal block matrix $T^{h,k}$ and T^h.

and the corresponding left eigenvector $y_h = [y_{h,1}, y_{h,2}, \ldots, y_{h,n}]^T$ has the components

$$y_{h,k} = \left(\frac{\bar{\tau}}{\bar{\sigma}}\right)^{\frac{k}{2}} \sin\frac{hk\pi}{n+1}, k = 1:n, h = 1:n \tag{5.77}$$

where the bar denotes complex conjugation.

Usage 5.18. (5.77) is used to compute the right eigenvector of the main diagonal block matrix $T^{h,k}$ and T^h.

If $\sigma = 0$ and $\tau \neq 0$ (or $\tau = 0$ and $\sigma \neq 0$), then the matrix (5.67) has the unique eigenvalues δ of geometric multiplicity one. The right and left eigenvectors are the first and last columns (or the last and first columns) of the identity matrix, respectively.

Note that, given the dimension of the matrix, knowing the ratio $\dfrac{\sigma}{\tau}$ is enough to uniquely determine all the right and left eigenvectors of T up to a scaling factor.

Theorem 5.19. *The matrix* (5.67) *is normal if and only if*

$$|\sigma| = |\tau| \tag{5.78}$$

Proof. The condition (5.78) is equivalent to the equality

$$T^H T = T T^H \tag{5.79}$$

Done. □

Usage 5.20. Theorem 5.19 deduces that both $T^{h,k}$ and T^h are normal.

The above theorem shows that a normal tridiagonal Toeplitz matrix can be written in the form

$$T' = \left(n; \rho e^{i\alpha'}, \delta, \rho e^{i\beta'}\right) \tag{5.80}$$

$$= \begin{bmatrix} \delta & \rho e^{i\beta'} & & & \\ \rho e^{i\alpha'} & \delta & \rho e^{i\beta'} & & \\ & \ddots & \ddots & \ddots & \\ & & \rho e^{i\alpha'} & \delta & \rho e^{i\beta'} \\ & & & \rho e^{i\alpha'} & \delta \end{bmatrix} \tag{5.81}$$

where $\delta \in \mathbb{C}, \rho \geq 0$ and $\alpha', \beta' \in \mathbb{R}$. It follows from (5.72) that the eigenvalues of (5.80)-(5.81) are given by

$$\lambda_h(T') = \delta + 2\rho e^{i\frac{\alpha'+\beta'}{2}} \cos\frac{h\pi}{n+1}, \quad h = 1:n \tag{5.82}$$

In particular, the eigenvalues lie on the closed line segment

$$S_{\lambda(T')} = \left\{ \delta + te^{i\frac{\alpha'+\beta'}{2}} : t \in \mathbb{R}, |t| \leq 2\rho\cos\frac{h\pi}{n+1} \right\} \subset \mathbb{C} \tag{5.83}$$

5.4 Strictly Diagonally Dominant Matrices

Definition 5.21. See [7].

1. **Weak row diagonal dominance.** A square matrix $A = (a_{ij})$ is said to be *diagonally dominant* if for every row of the matrix, the magnitude of the diagonal entry in a row is larger than or equal to the sum of the magnitudes

o all the other (non-diagonal) entries in that row. More precisely, the matrix A is strictly diagonally dominant if

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \forall i \tag{5.84}$$

2. **Strict row diagonal dominance.** A square matrix $A = (a_{ij})$ is said to be *strictly diagonally dominant (SDD)* if for every row of the matrix, the magnitude of the diagonal entry in a row is larger than the sum of the magnitudes o all the other (non-diagonal) entries in that row. More precisely, the matrix A is strictly diagonally dominant if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \forall i \tag{5.85}$$

3. **Weak column diagonal dominance.** A square matrix $A = (a_{ij})$ is said to be *diagonally dominant* if for every column of the matrix, the magnitude of the diagonal entry in a column is larger than or equal to the sum of the magnitudes o all the other (non-diagonal) entries in that column. More precisely, the matrix A is strictly diagonally dominant if

$$|a_{jj}| \geq \sum_{i \neq j} |a_{ij}|, \forall i \tag{5.86}$$

4. **Strict column diagonal dominance.** A square matrix $A = (a_{ij})$ is said to be *strictly diagonally dominant* if for every column of the matrix, the magnitude of the diagonal entry in a column is larger than or equal to the sum of the magnitudes o all the other (non-diagonal) entries in that column. More precisely, the matrix A is strictly diagonally dominant if

$$|a_{jj}| > \sum_{i \neq j} |a_{ij}|, \forall i \tag{5.87}$$

Theorem 5.22 (Levy-Desplanques). *A strictly diagonally dominant matrix (or an irreducibly diagonally dominant matrix) is nonsingular.*

Usage 5.23. Levy-Desplanques theorem is used to deduce the invertibility (or nonsingularity) of the main diagonal block matrix $T^{h,k}$ and T^h.

Theorem 5.24. *A Hermitian diagonally dominant matrix A with real nonnegative diagonal entries is positive semidefinite.*

5.5 Gershgorin Circle Theorem

See [8]. In mathematics, the *Gershgorin circle theorem* may be used to bound the spectrum of a square matrix. It was first published by the Soviet mathematican Semyon Aronovich Gershgorin in 1931. Let A be a complex $n \times n$ matrix, with entries a_{ij}. For $i = 1, 2, \ldots, n$ let

$$R_i = \sum_{j \neq i} |a_{ij}| \tag{5.88}$$

be the sum of the absolute values of the non-diagonal entries in ith row. Let $D(a_{ii}, R_i)$ be the closed disc centered at a_{ii} with radius R_i. Such a disc is called a *Gershgorin disc*.

Theorem 5.25. *Every eigenvalue of A lies within at least one of the Gershgorin discs $D(a_{ii}, R_i)$.*

PROOF. Let λ be an eigenvalue of A and let $x = (x_j)$ be a corresponding eigenvector. Let $i \in \{1, 2, \ldots, n\}$ be chosen so that

$$|x_i| = \max_j |x_j| \tag{5.89}$$

That is to say, choose i so that x_i is the largest number in absolute value in the the vector x. Then $|x_i| > 0$, otherwise $x = 0$. Since x is an eigenvector, $Ax = \lambda x$, and thus

$$\sum_j a_{ij} x_j = \lambda x_i, \quad i = 1, 2, \ldots, n \tag{5.90}$$

So, splitting the sum, we get

$$\sum_{j \neq i} a_{ij} x_j = \lambda x_i - a_{ii} x_i \tag{5.91}$$

We may then divide both sides by $x_i \neq 0$ and take the absolute value to obtain

$$|\lambda - a_{ii}| = \left| \frac{\sum_{j \neq i} a_{ij} x_j}{x_i} \right| \tag{5.92}$$

$$\leq \sum_{j \neq i} \left| \frac{a_{ij} x_j}{x_i} \right| \tag{5.93}$$

$$\leq \sum_{j \neq i} |a_{ij}| \tag{5.94}$$

$$= R_i \tag{5.95}$$

where the last inequality is valid because

$$\left| \frac{x_j}{x_i} \right| \leq 1, \forall j \neq i \tag{5.96}$$

Done. □

Usage 5.26. We now use Gershgorin circle theorem to deduce some information about eigenvalues of the matrix $A^{h,k}$ and A^h in particular.

All $N_x N_y$ Gershgorin discs are identical to

$$D(a_{ii}, R_i) = D\left(-2\left(\frac{1}{h^2} + \frac{1}{k^2} \right), \frac{1}{h^2} + \frac{1}{h^2} + \frac{1}{k^2} + \frac{1}{k^2} \right) \tag{5.97}$$

$$= D\left(-2\left(\frac{1}{h^2} + \frac{1}{k^2} \right), 2\left(\frac{1}{h^2} + \frac{1}{k^2} \right) \right), \quad i = 1, 2, \ldots, N_x N_y \tag{5.98}$$

Using the Gershgorin circle theorem, we deduce that all eigenvalues of $A^{h,k}$ lie in this Gershgorin circle, i.e.,

$$\left|\lambda_i\left(A^{h,k}\right)+2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)\right| \leq 2\left(\frac{1}{h^2}+\frac{1}{k^2}\right), \quad i=1,2,\ldots,N_xN_y \tag{5.99}$$

or equivalently,

$$-4\left(\frac{1}{h^2}+\frac{1}{k^2}\right) \leq \lambda_i\left(A^{h,k}\right) \leq 0, i=1,2,\ldots,N_xN_y \tag{5.100}$$

Thus, we have bounded all eigenvalues of $A^{h,k}$. These bounds can used to estimate $\left\|A^{h,k}\right\|_2$ and estimate the condition number $\kappa\left(A^{h,k}\right)$ without knowing any information about $\lambda_i\left(A^{h,k}\right)$.

More explicitly, we have

$$\left\|A^{h,k}\right\|_2 = \max_{1\leq i\leq N_x, 1\leq j\leq N_y}\left|\lambda_i\left(A^{h,k}\right)\right| \tag{5.101}$$

$$\leq 4\left(\frac{1}{h^2}+\frac{1}{k^2}\right) \tag{5.102}$$

$$= O\left(\frac{1}{H^2}\right) \tag{5.103}$$

Corollary 5.27. *The eigenvalues of A must also lie within the Gershgorin discs C_j corresponding to the columns of A.*

PROOF. Apply Theorem 5.25 to A^T. □

For a diagonal matrix, the Gershgorin discs coincide with the spectrum. Conversely, if the Gershgorin discs coincide with the spectrum, the matrix is diagonal.

Theorem 5.28 (Strengthening of Gershgorin Circle Theorem). *If the union of k discs is disjoint from the union of the other $n-k$ discs then the former union contains exactly k and the latter $n-k$ eigenvalues of A.*

PROOF. Let D be the diagonal matrix with entries equal to the diagonal entries of A and let

$$B(t) = (1-t)D + tA \tag{5.104}$$

We will use the fact that the eigenvalues are continuous in t, and show that if any eigenvalue moves from one of the unions to the other, then it must be outside all the discs for some t, which is a contradiction.

The statement is true for $D+B(0)$. The diagonal entries of $B(t)$ are equal to that of A, thus the centers of the Gershgorin circles are the same, however their radii are t times that of A. Therefore the union of the corresponding k discs of $B(t)$ is disjoint from the union of the remaining $n-k$ for all $t\in[0,1]$. The discs are closed, so the distance of the two unions for A is $d>0$. The distance for $B(t)$ is a decreasing function of $t\in[0,1]$, so it is always at least d. Since the eigenvalues of $B(t)$ are continuous functions of t, for any eigenvalue

$\lambda(t)$ of $B(t)$ in the union of the k discs its distance $d(t)$ from the union of the other $n-k$ discs is also continuous. Obviously, $d(0) \geq d$, and assume $\lambda(1)$ lies in the union of the $n-k$ discs. Then $d(1) = 0$, so there exists $0 < t_0 < 1$ such that $0 < d(t_0) < d$.

But this means $\lambda(t_0)$ lies outside the Gershgorin discs, which is impossible. Therefore $\lambda(1)$ lies in the union of the k discs, and the theorem is proven. □

5.6 Normal Matrices

See [9].

Definition 5.29. A complex square matrix A is *normal* if

$$A^*A = AA^* \tag{5.105}$$

where A^* is the conjugate transpose of A. That is, a matrix is normal if it commutes with its conjugate transpose. A real square matrix A satisfies $A^* = A^T$, and is therefore normal if

$$A^TA = AA^T \tag{5.106}$$

A matrix is normal if and only if it is unitarily similar to a diagonal matrix, and therefore any matrix A satisfying (5.105) is diagonalizable. Among complex matrices, all unitary, Hermitian and skew-Hermitian matrices are normal. Likewise, among real matrices, all orthogonal, symmetric and skew-symmetric matrices are normal. However, it is not the case that all normal matrices are either unitary or skew-Hermitian. For example,

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \tag{5.107}$$

is neither unitary, Hermitian, nor skew-Hermitian, yet it is normal because

$$AA^* = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} = A^*A \tag{5.108}$$

Usage 5.30. Since $A^{h,k}$ is symmetric, $A^{h,k}$ is normal. Therefore all following properties of normal matrices can be applied to deduce some information of $A^{h,k}$.

Theorem 5.31. *A normal triangular matrix is diagonal.*

Proof. Let A be a normal upper triangular matrix. Since $[A^*A]_{ii} = [AA^*]_{ii}$, one has

$$\langle e_i, A^*Ae_i \rangle = \langle e_i, AA^*e_i \rangle \tag{5.109}$$

i.e., the first row must have the same norm as the first column.

$$\|Ae_1\|^2 = \|A^*e_1\|^2 \tag{5.110}$$

The first entry of row 1 and column 1 are the same and the rest of column 1 is zero. This implies the first row must be zero for entries 2 through n. Continuing this argument for row-column pairs 2 through n shows A is diagonal. □

The concept of normality is important because normal matrices are precisely those to which the spectral theorem applies.

Theorem 5.32. *A matrix A is normal if and only if there exists a diagonal matrix Λ and a unitary matrix U such that*

$$A = U\Lambda U^* \tag{5.111}$$

Usage 5.33. Theorem 5.32 is used to deduce that the main diagonal block matrices $T^{h,k}$ and T^h are normal.

The diagonal entries of Λ are the eigenvalues of A, and the columns of U are the eigenvectors of A. The matching eigenvalues in Λ come in the same order as the eigenvectors are ordered as columns of U.

Another way of stating the spectral theorem is to say that normal matrices are precisely those matrices that can be represented by a diagonal matrix with respect to a properly chosen orthonormal basis of $\mathbb{C}^n$.

Phrased differently: *A matrix is normal if and only if its eigenspaces span $\mathbb{C}^n$ and are pairwise orthogonal with respect to the standard inner product of $\mathbb{C}^n$.*

The spectral theorem for normal matrices is a special case of the more general *Schur decomposition* which holds for all square matrices. Let A be a square matrix. Then by Schur decomposition it is unitary similar to an upper-triangular matrix, say, B. If A is normal, so is B. But then B must be diagonal, for, as noted above, a normal upper-triangular matrix is diagonal. The spectral theorem permits the classification of normal matrices in terms of their spectral.

Theorem 5.34. *A normal matrix is unitary if and only if its spectrum is contained in the unit circle of the complex plane.*

Theorem 5.35. *A normal matrix is self-adjoint if and only if its spectrum is contained in $\mathbb{R}$. In other words, a normal matrix is Hermitian if and only if all its eigenvalues are real.*

In general, the sum or product of two normal matrices need not be normal. However, the following holds.

Theorem 5.36. *If A and B are normal with $AB = BA$, then both AB and $A + B$ are also normal. Furthermore there exists a unitary matrix U such taht UAU^* and UBU^* are diagonal matrices. In other words A and B are simultaneously diagonalizable.*

In this special case, the columns of U^* are eigenvectors of both A an B and form an orthonormal basis in $\mathbb{C}^n$. This follows by combining the theorems that, over an algebraically closed fields, commuting matrices are simultaneously triangularizable and a normal matrix is diagonalizable - the added result is that

these can both be done simultaneously.

Theorem 5.37 (Equivalent definitions of Normal matrices). *Let A be a $n \times n$ complex matrix. Then the following are equivalent.*

1. *A is normal.*
2. *A is diagonalizable by a unitary matrix.*
3. *The entire space is spanned by some orthonormal set of eigenvectors of A.*
4. $$\|Ax\| = \|A^* x\|, \forall x \tag{5.112}$$
5. *The Frobenius norm of A can be computed by the eigenvalues of A.*
$$tr\left(A^* A\right) = \sum_j |\lambda_j|^2 \tag{5.113}$$
6. *The Hermitian part $\frac{1}{2}\left(A + A^*\right)$ and skew-Hermitian part $\frac{1}{2}\left(A - A^*\right)$ of A commute.*
7. *A^* is a polynomial of degree $\leq n-1$ in A.*
8. *$A^* = AU$ for some unitary matrix U.*
9. *U and P commute, where we have the polar decomposition $A = UP$ with a unitary matrix U and some positive semidefinite matrix P.*
10. *A commutes with some normal matrix N with distinct eigenvalues.*
11. *$\sigma_i = |\lambda_i|$ for all $1 \leq i \leq n$ where A has singular values $\sigma_1 \geq \ldots \geq \sigma_n$ and eigenvalues $|\lambda_1| \geq \cdots \geq |\lambda_n|$.*
12. *The operator norm of a normal matrix A equals the numerical and spectral radius of A. This fact generalizes to normal operators. Explicitly, this means*
$$\sup_{\|x\|=1} \|Ax\| = \sup_{\|x\|=1} |\langle Ax, x\rangle| \tag{5.114}$$
$$= \max\left\{|\lambda| : \lambda \in \sigma\left(A\right)\right\} \tag{5.115}$$

Usage 5.38. Theorem 5.37 is used to deduce some properties of the matrix $T^{h,k}$ and $A^{h,k}$, or T^h and A^h in particular.

Some but not all of the above generalize to normal operators on infinite-dimensional Hilbert spaces. For example, a bounded operator satisfying **9** in Theorem 5.37 is only quasinormal.

6 Practical Problems

Let $\Omega = (0,1) \times (0,1) \subset \mathbb{R}^2$ and $f \in L^2(\Omega)$.

$$-\Delta u(x,y) = f(x,y) \text{ in } \Omega \tag{6.1}$$

Problem 6.1 (Dirichlet boundary condition).

1. *Solve equation* (6.1) *with uniform mesh subject to a Dirichlet boundary condition*

$$u(x,y) = g(x,y), \forall (x,y) \in \partial\Omega \tag{6.2}$$

2. *Solve equation* (6.1) *subject to a Dirichlet boundary condition*

$$u(x,y) = g(x,y), \forall (x,y) \in \partial\Omega \tag{6.3}$$

with following mesh

$$\{x_i\}_{i\in[0,N_x]} \text{ with } x_i = ih \text{ and } h = \frac{1}{N_x} \tag{6.4}$$

$$\{y_j\}_{j\in[0,N_y]} \text{ with } y_j = jk \text{ and } k = \frac{1}{N_y} \tag{6.5}$$

Noting that there exist positive constants α, β *such that*

$$\alpha \leq \frac{h}{k} \leq \beta \tag{6.6}$$

Problem 6.2 (Dirichlet-Neumann boundary condition). *Solve equation* (6.1) *with uniform mesh subject to a Dirichlet Neumann boundary condition*

$$u(x,0) = g_1(x) \tag{6.7}$$

$$u(0,y) = g_2(y) \tag{6.8}$$

$$u(x,1) = g_3(x) \tag{6.9}$$

$$\frac{\partial u}{\partial x}(1,y) = g_4(x) \tag{6.10}$$

Problem 6.3 (Neumann boundary condition). *Solve equation* (6.1) *with condition*

$$\int_\Omega f(x,y)\,dxdy = 0 \tag{6.11}$$

and with uniform mesh subject to a Neumann boundary condition

$$\nabla u \cdot \vec{n}(x,y) = 0, \forall (x,y) \in \partial\Omega \tag{6.12}$$

where $\vec{n}$ *is unit normal vector to boundary* $\partial\Omega$.

We now use MATLAB to implement these pratical problems with

1. The 5-point Laplacian.
2. The 9-point Laplacian.

7 Matlab Implementation Using 5-Point Laplacian

7.1 Matlab Implementation for Problem 6.1.1

7.1.1 Matlab Scripts

g.m

```
function g = g(x,y,k);

if (k == 1)
    g = x + y;
end

if(k==2)
    g = x^2 + y^2;
end

if(k==3)
    g = sin(x) + sin(y);
end

if(k==4)
    g = x^2 + y^3;
end

if(k==5)
    g = sin(x) + cos(y);
end
```

exact_solution2D.m

```
function u_ex=exact_solution2D(x,y,k);

if (k==1)
    u_ex = x + y + 30*x*y*(x - 1)*(y - 1);
end

if(k==2)
    u_ex = x^2 + y^2 + 30*x^2*(1-x)^2*y^2*(1-y)^2;
end

if(k==3)
    u_ex = sin(x) + sin(y) + 30*x*(x-1)^2*y^3*(y-1)^4;
end

if(k==4)
    u_ex = x^2 + y^3 + 50*sin(x^2*(x-1)^2*y^2*(y-1)^2);
end
```

```
if(k==5)
    u_ex = sin(x) + cos(y) + sin(x^4*(x-1)^4*y^4*(y-1)^4);
end
```

functionf2D.m

```
function f=functionf2D(x,y,k);

if (k == 1)
    f =  - 60*x*(x - 1) - 60*y*(y - 1);
end

if(k==2)
    f = - (60*x^2*y^2*(x - 1)^2 + 60*x^2*y^2*(y - 1)^2 + ...
        60*x^2*(x - 1)^2*(y - 1)^2 + 60*y^2*(x - 1)^2*(y - 1)^2 + ...
        120*x*y^2*(2*x - 2)*(y - 1)^2 + ...
        120*x^2*y*(2*y - 2)*(x - 1)^2 + 4);
end

if(k==3)
    f = -(60*y^3*(2*x - 2)*(y - 1)^4 - sin(y) - sin(x) + ...
        60*x*y^3*(y - 1)^4 + 720*x*y^2*(x - 1)^2*(y - 1)^3 + ...
        360*x*y^3*(x - 1)^2*(y - 1)^2 + 180*x*y*(x - 1)^2*(y - 1)^4);
end

if(k==4)
    f = -(6*y + 100*x^2*cos(x^2*y^2*(x - 1)^2*(y - 1)^2)* ...
        (x - 1)^2*6*y^2 - 6*y + 1) + 100*y^2*cos(x^2*y^2*(x - 1)^2* ...
        (y - 1)^2)*(y - 1)^2*(6*x^2 - 6*x + 1) - 200*x^2*y^4* ...
        sin(x^2*y^2*(x - 1)^2*(y - 1)^2)*(y - 1)^4* ...
        (2*x^2 - 3*x + 1)^2 - 200*x^4*y^2*sin(x^2*y^2*(x - 1)^2* ...
        (y - 1)^2)*(x - 1)^4*(2*y^2 - 3*y + 1)^2 + 2);
end

if(k==5)
    f = cos(y) + sin(x) + 16*x^6*y^8*sin(x^4*y^4*(x - 1)^4* ...
        (y - 1)^4)*(2*x - 1)^2*(x - 1)^6*(y - 1)^8 + 16*x^8* ...
        y^6*sin(x^4*y^4*(x - 1)^4*(y - 1)^4)*(2*y - 1)^2*(x - 1)^8* ...
        (y - 1)^6 - 4*x^2*y^4*cos(x^4*y^4*(x - 1)^4*(y - 1)^4)* ...
        (x - 1)^2*(y - 1)^4*(14*x^2 - 14*x + 3) - ...
        4*x^4*y^2*cos(x^4*y^4*(x - 1)^4*(y - 1)^4)* ...
        (x - 1)^4*(y - 1)^2*(14*y^2 - 14*y + 3);
end
```

main2D_1a.m

```
%% solve equation -u_xx(x,y)-u_yy(x,y)=f(x,y) with the
%% Dirichlet boundary condition
clear all
close all
clc
```

```
format long

tic

%% Initial informations
ax=0.0;
bx=1.0;
ay=0.0;
by=1.0;
cases=1;
N=4;% number of mesh points of first mesh
number_mesh=3;
number_mesh_point=zeros(number_mesh,1);
norm_max=zeros(number_mesh,1);
norm_l2=zeros(number_mesh,1);
norm_maxh1=zeros(number_mesh,1);
norm_h1=zeros(number_mesh,1);

%% Solve discrite solution and refine mesh
for inumber_mesh=1:number_mesh

    h=(bx-ax)/N;
    number_mesh_point(inumber_mesh)=N;

    %% Create mesh point
    x=zeros(N+1,1);
    for i_iter=1:N+1
        x(i_iter)=(i_iter-1)*h;
    end
    y=zeros(N+1,1);
    for i_iter=1:N+1
        y(i_iter)=(i_iter-1)*h;
    end

    %% Create matrix A
    A=sparse((N-1)*(N-1),(N-1)*(N-1));
    B=sparse(N-1,N-1);
    I_h=sparse(N-1,N-1);
    for i=1:N-1
        I_h(i,i)=1;
    end
    for i=1:N-1
        if(i==1)
            B(i,i)=4;
            B(i,i+1)=-1;
        else
            if(i==N-1)
                B(i,i)=4;
                B(i,i-1)=-1;
            else
```

```
            B(i,i)=4;
            B(i,i-1)=-1;
            B(i,i+1)=-1;
        end
    end
end
for i=1:N-1
    if(i==1)
        A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B;
        A((i-1)*(N-1)+1:i*(N-1),i*(N-1)+1:(i+1)*(N-1))=-I_h;
    else
        if(i==N-1)
            A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B;
            A((i-1)*(N-1)+1:i*(N-1),(i-2)*(N-1)+1:(i-1)*(N-1))=-I_h;
        else
            A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B;
            A((i-1)*(N-1)+1:i*(N-1),i*(N-1)+1:(i+1)*(N-1))=-I_h;
            A((i-1)*(N-1)+1:i*(N-1),(i-2)*(N-1)+1:(i-1)*(N-1))=-I_h;
        end
    end
end
A=(1/h^2)*A;

%% Create vector F
F=zeros((N-1)*(N-1),1);
for i1=1:N-1
    if (i1 == 1)
        for i2=1:N-1
            if (i2 == 1)
                F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                (g(h,0)+g(0,h))/h^2;
            else
                if (i2 == N-1)
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    (g(h,1)+g(0,(N-1)*h))/h^2;
                else
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    g(0,i2*h)/h^2;
                end
            end
        end
    else
        if (i1 == N-1)
            for i2=1:N-1
                if (i2 == 1)
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    (g((N-1)*h,0)+g(1,h))/h^2;
                else
                    if (i2 == N-1)
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
```

```
                        (g((N-1)*h,1)+g(1,(N-1)*h))/h^2;
                    else
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                        g(1,i2*h)/h^2;
                    end
                end
            end
        else
            for i2=1:N-1
                if (i2 == 1)
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    (g(i1*h,0))/h^2;
                else
                    if (i2 == N-1)
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                        (g(i1*h,1))/h^2;
                    else
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases);
                    end
                end
            end
        end
    end
end

%% Solve discrete solution
u=A\F;

%% Get exact solution
u_exact=zeros((N+1),(N+1));
for i1=1:N+1
    for i2=1:N+1
        u_exact(i1,i2)=exact_solution2D(x(i1),y(i2),cases);
    end
end

%% Create discrete solution with boundary
u_dis=u_exact;
for i1=1:N-1
    for i2=1:N-1
        u_dis(i1+1,i2+1)=u((i2-1)*(N-1)+i1);
    end
end

%% Reshape u_dis and u_exact
udis=reshape(u_dis,size(u_dis,1)*size(u_dis,2),1);
uexact=reshape(u_exact,size(u_exact,1)*size(u_exact,2),1);

%% Calculate the error on L^infinity
norm_max(inumber_mesh)=0.0;
```

```
    for i=1:length(udis)
        if (abs(u_dis(i)-u_exact(i)) > norm_max(inumber_mesh))
            norm_max(inumber_mesh)=abs(u_dis(i)-u_exact(i));
        end
    end
    fprintf('error on L^infinity: %df\n',norm_max(inumber_mesh));

    %%  Calculate the error on L^2
    norm_l2(inumber_mesh)=0;
    for i=1:length(udis)
        norm_l2(inumber_mesh)=norm_l2(inumber_mesh)+ ...
        (u_dis(i)-u_exact(i))^2*h;
    end
    norm_l2(inumber_mesh)=(norm_l2(inumber_mesh))^(1/2);
    fprintf('error on L^2: %df\n',norm_l2(inumber_mesh));

    %% Figure exact and dicrete solutions
    figure
    surf(x,y,u_dis)
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Discrete solutions with N=%d',N);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1])
    figure
    surf(x,y,u_exact)
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Exact solutions with N=%d',N);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1]);

    %% Refine mesh
    N=4*N;
end

%% Figure for errors respect to number of mesh point
figure
plot(log(number_mesh_point), log(norm_max),'blue', ...
     log(number_mesh_point), log(norm_l2), log(number_mesh_point), ...
     2*log(number_mesh_point),'black');
xlabel('Log(MeshPoint)');ylabel('-Log(Error)');
title('Errors');
legend('norm_max','norm_l2','2x','Location','NorthEastOutside');
print('-r300','-djpeg');
toc
```

7.1.2 Results

TEST 1.

$$g = x + y \tag{7.1}$$
$$u_{ex} = x + y + 30xy\,(x-1)\,(y-1) \tag{7.2}$$
$$f = -60x\,(x-1) - 60y\,(y-1) \tag{7.3}$$

Run these scripts for `cases = 1`, MATLAB returns

```
N = 4
error on L^infinity: 4.440892e-16f
error on L^2: 4.440892e-16f

N = 8
error on L^infinity: 1.776357e-15f
error on L^2: 2.645093e-15f

N = 16
error on L^infinity: 3.552714e-15f
error on L^2: 3.869614e-15f

N = 32
error on L^infinity: 2.264855e-14f
error on L^2: 5.587052e-14f

N = 64
error on L^infinity: 1.145750e-13f
error on L^2: 4.679732e-13f
```

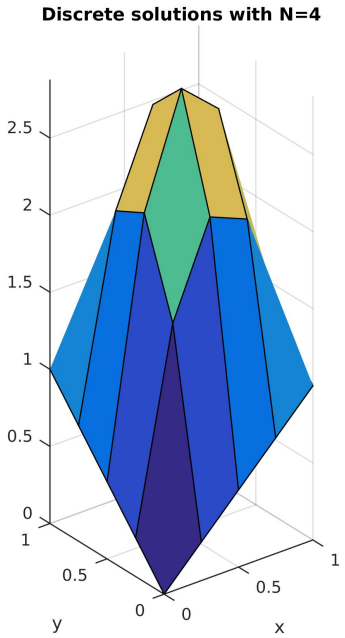

Figure 9: Discrete Solutions: Problem 1, Test 1, $N = 4$.

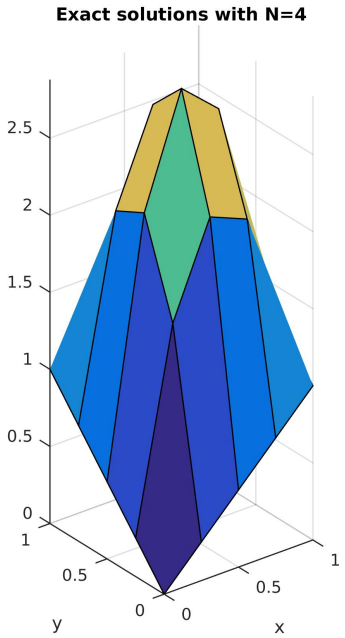

Figure 10: Exact Solutions: Problem 6.1.1, Test 1, $N = 4$.

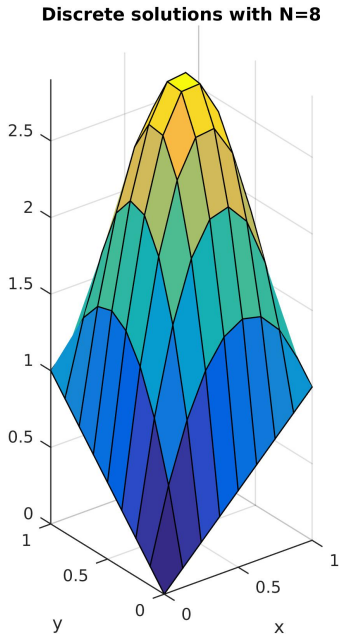

Figure 11: Discrete Solutions: Problem 6.1.1, Test 1, $N = 8$.

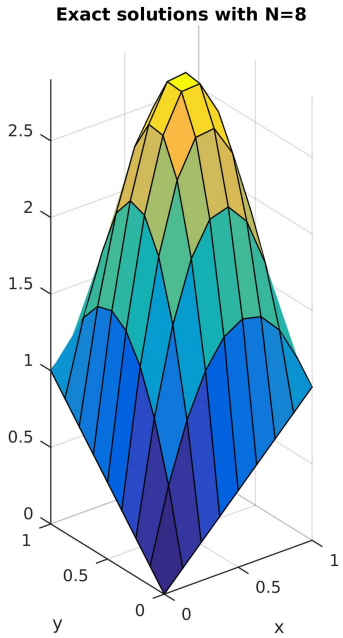

Figure 12: Exact Solutions: Problem 6.1.1, Test 1, $N = 8$.

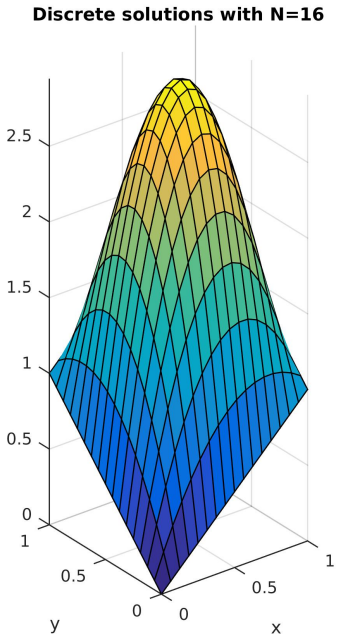

Figure 13: Discrete Solutions: Problem 6.1.1, Test 1, $N = 16$.

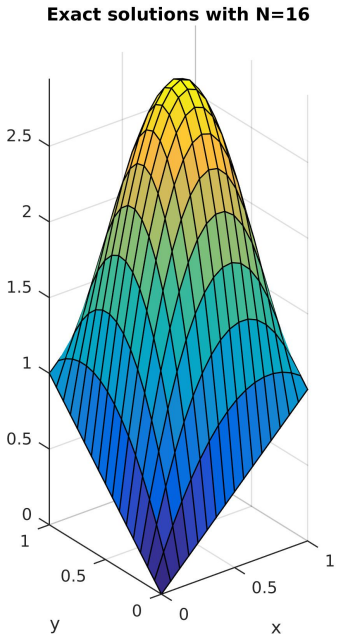

Figure 14: Exact Solutions: Problem 6.1.1, Test 1, $N = 16$.

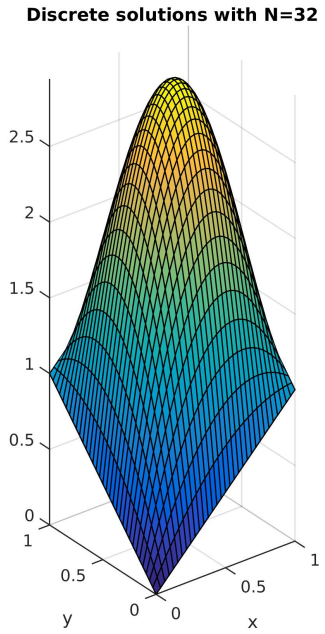

Figure 15: Discrete Solutions: Problem 6.1.1, Test 1, $N = 32$.

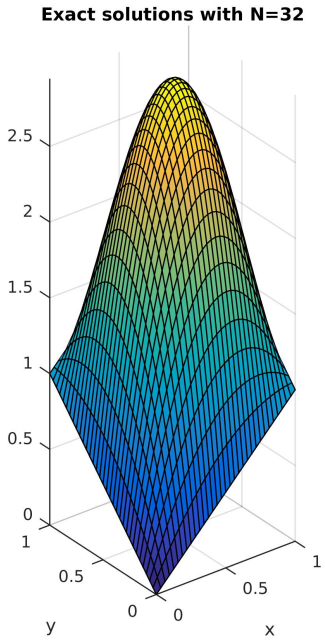

Figure 16: Exact Solutions: Problem 6.1.1, Test 1, $N = 32$.

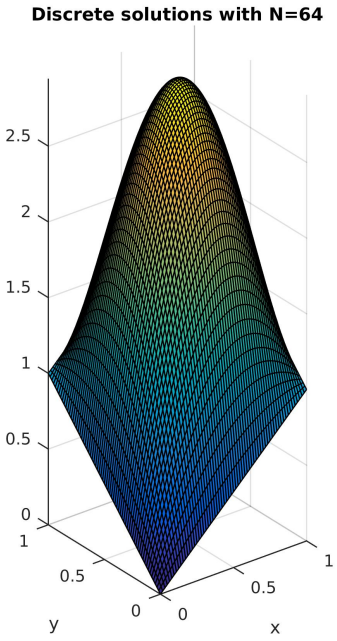

Figure 17: Discrete Solutions: Problem 6.1.1, Test 1, $N = 64$.

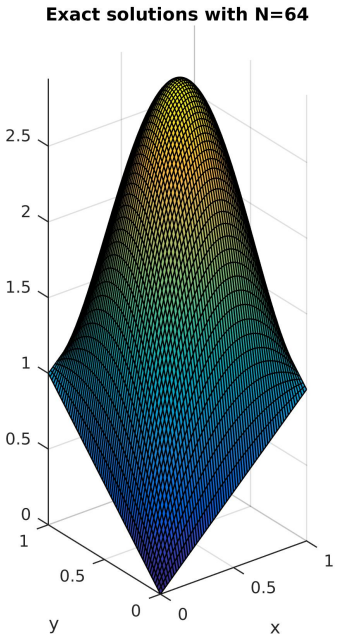

Figure 18: Exact Solutions: Problem 6.1.1, Test 1, $N = 64$.

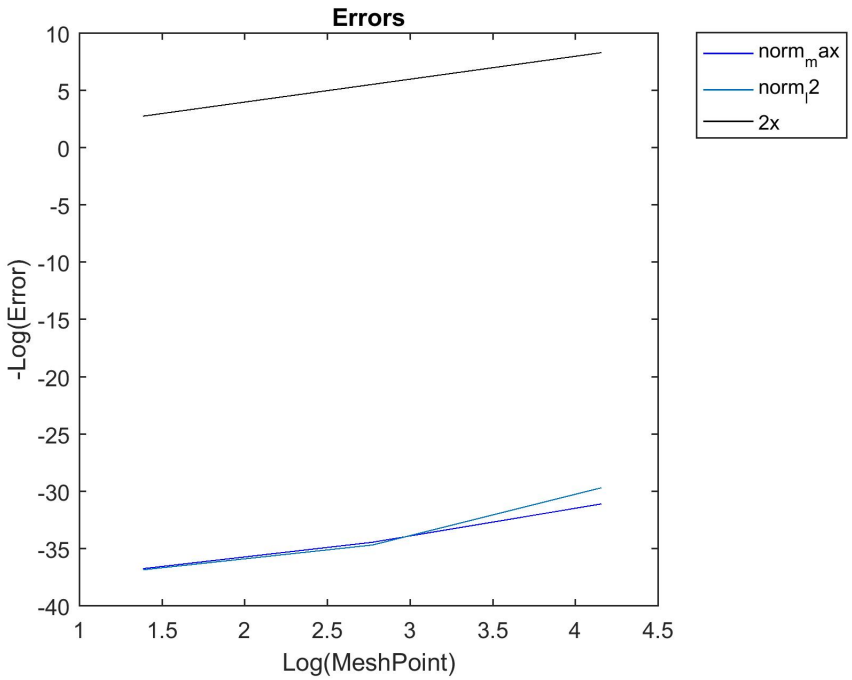

Figure 19: Error on a log-log scale: Problem 6.1.1, Test 1.

TEST 2.

$$g = x^2 + y^2 \tag{7.4}$$

$$u_{ex} = x^2 + y^2 + 30x^2y^2(x-1)^2(y-1)^2 \tag{7.5}$$

$$f = -\Delta\left(x^2 + y^2 + 30x^2y^2(x-1)^2(y-1)^2\right) \tag{7.6}$$

Run the above scripts for `cases = 2`, MATLAB returns

```
N = 4
error on L^infinity: 2.655029e-02f
error on L^2: 2.710100e-02f

N = 8
error on L^infinity: 6.543538e-03f
error on L^2: 9.532194e-03f

N = 16
error on L^infinity: 1.631785e-03f
error on L^2: 3.369234e-03f

N = 32
error on L^infinity: 4.077211e-04f
error on L^2: 1.191197e-03f

N = 64
error on L^infinity: 1.019167e-04f
error on L^2: 4.211525e-04f
```

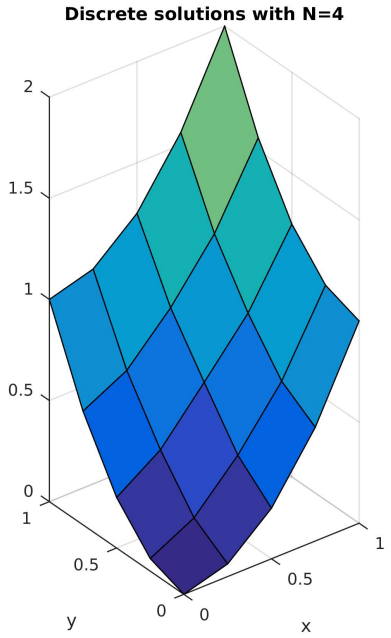

Figure 20: Discrete Solutions: Problem 6.1.1, Test 2, $N = 4$.

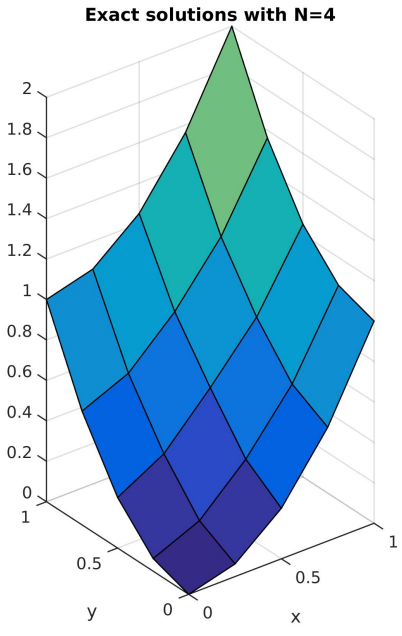

Figure 21: Exact Solutions: Problem 6.1.1, Test 2, $N = 4$.

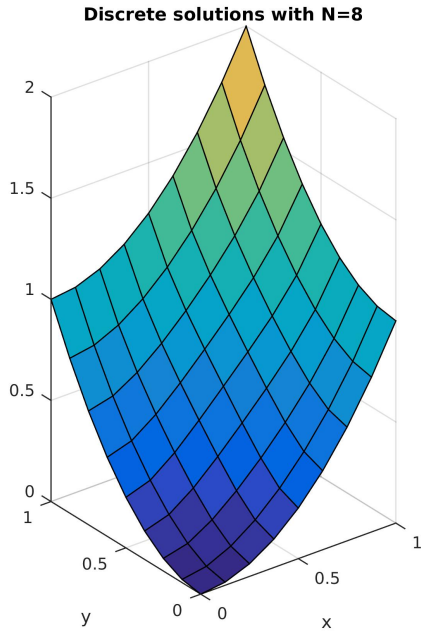

Figure 22: Discrete Solutions: Problem 6.1.1, Test 2, $N = 8$.

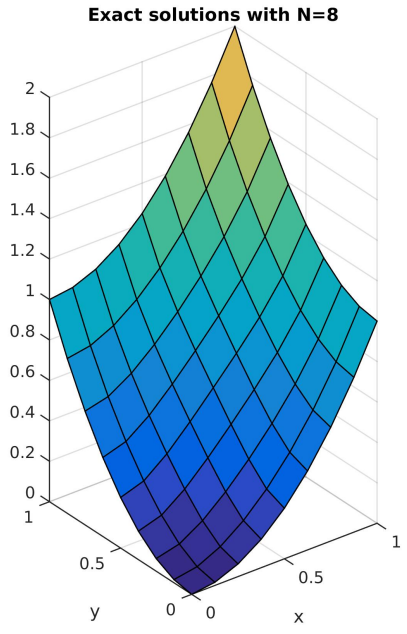

Figure 23: Exact Solutions: Problem 6.1.1, Test 2, $N = 8$.

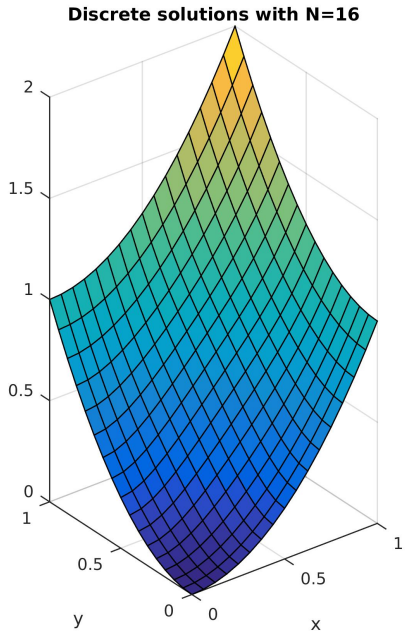

Figure 24: Discrete Solutions: Problem 6.1.1, Test 2, $N = 16$.

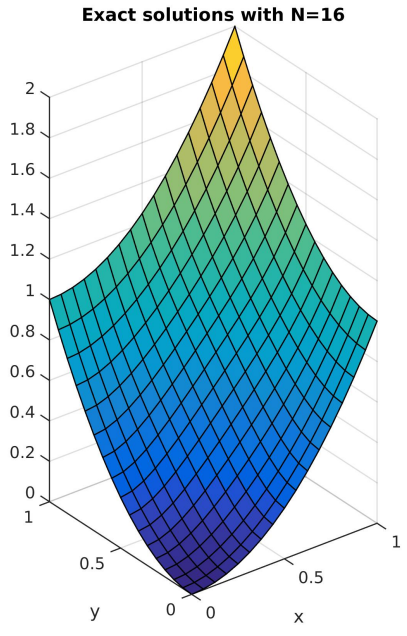

Figure 25: Exact Solutions: Problem 6.1.1, Test 2, $N = 16$.

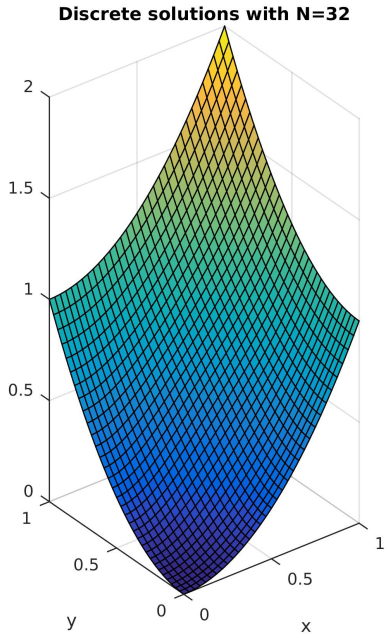

Figure 26: Discrete Solutions: Problem 6.1.1, Test 2, $N = 32$.

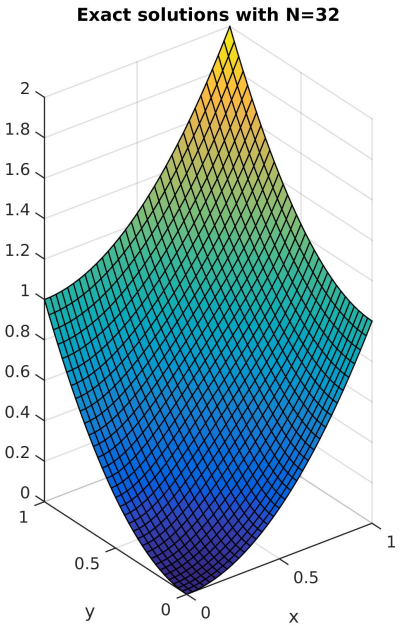

Figure 27: Exact Solutions: Problem 6.1.1, Test 2, $N = 32$.

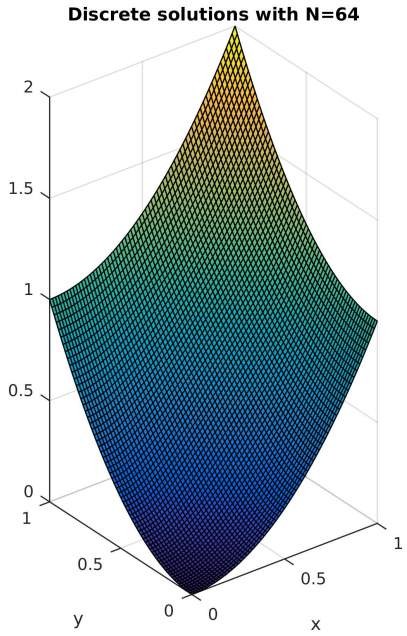

Figure 28: Discrete Solutions: Problem 6.1.1, Test 2, $N = 64$.

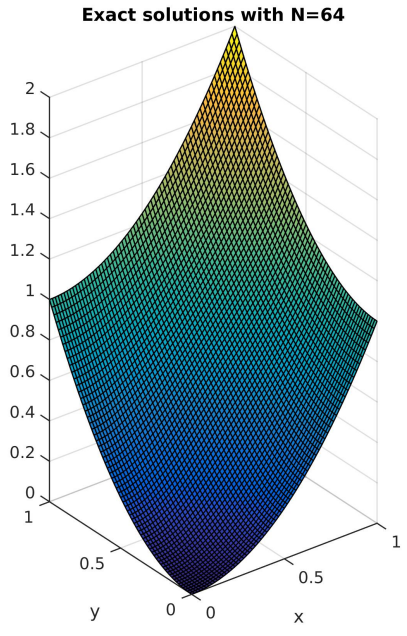

Figure 29: Exact Solutions: Problem 6.1.1, Test 2, $N = 64$.

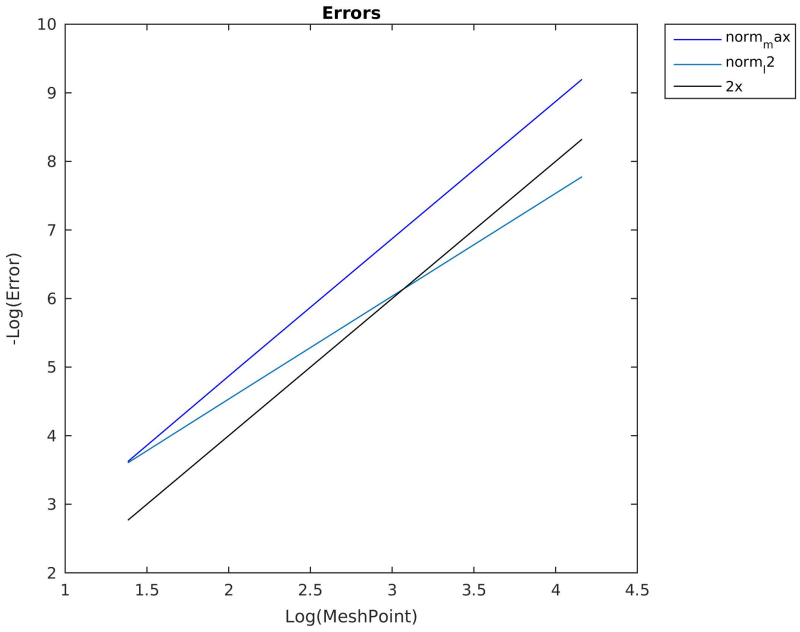

Figure 30: Error on a log-log scale: Problem 6.1.1, Test 2.

TEST 3.

$$g = \sin x + \sin y \tag{7.7}$$

$$u_{ex} = \sin x + \sin y + 30x(x-1)^2y^3(y-1)^4 \tag{7.8}$$

$$f = -\Delta\left(\sin x + \sin y + 30x(x-1)^2y^3(y-1)^4\right) \tag{7.9}$$

Run the above scripts for `cases = 3`, MATLAB returns

```
N = 4
error on L^infinity: 4.408222e-03f
error on L^2: 4.663096e-03f

N = 8
error on L^infinity: 1.239465e-03f
error on L^2: 1.377622e-03f

N = 16
error on L^infinity: 2.871268e-04f
error on L^2: 4.800258e-04f

N = 32
error on L^infinity: 7.868488e-05f
error on L^2: 1.688302e-04f

N = 64
error on L^infinity: 1.978660e-05f
error on L^2: 5.959994e-05f
```

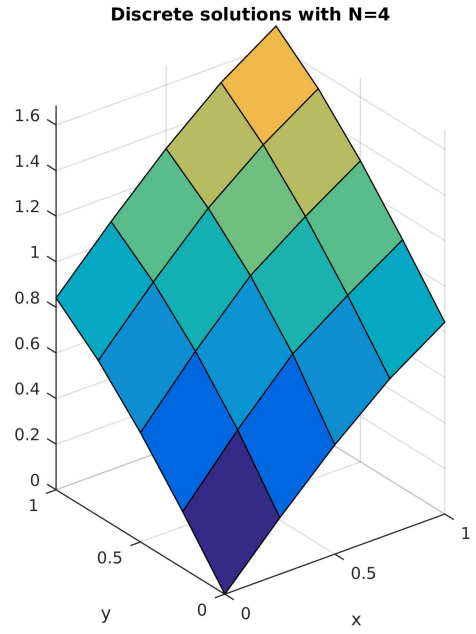

Figure 31: Discrete Solutions: Problem 6.1.1, Test 3, $N = 4$.

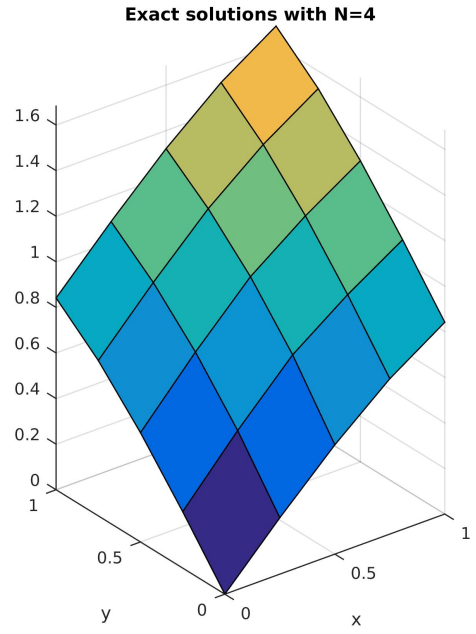

Figure 32: Exact Solutions: Problem 6.1.1, Test 3, $N = 4$.

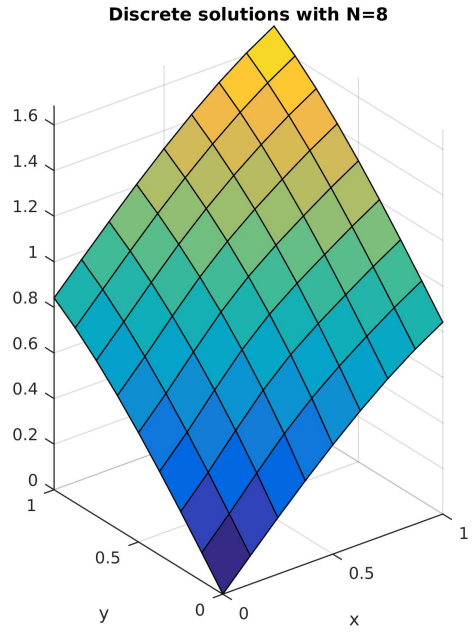

Figure 33: Discrete Solutions: Problem 6.1.1, Test 3, $N = 8$.

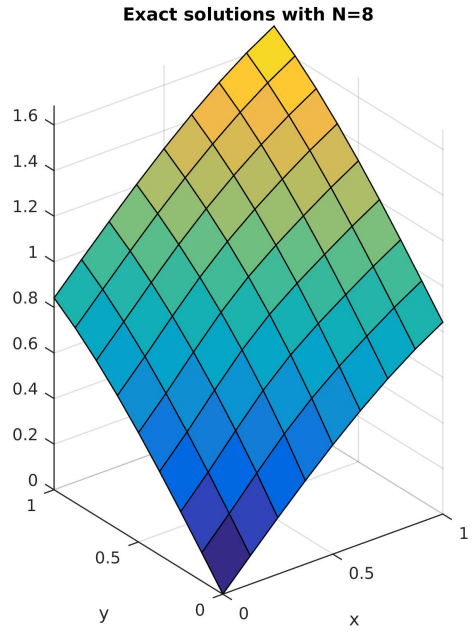

Figure 34: Exact Solutions: Problem 6.1.1, Test 3, $N = 8$.

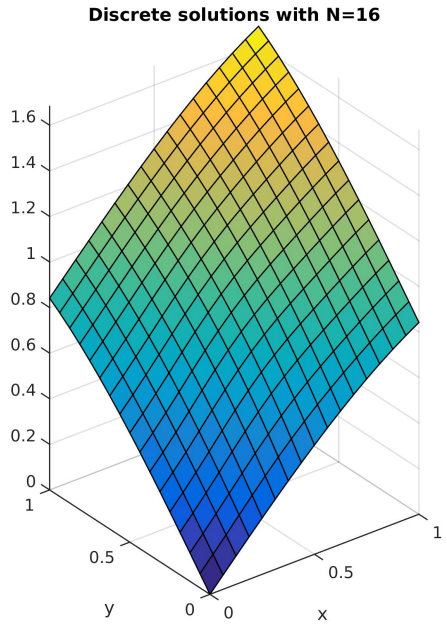

Figure 35: Discrete Solutions: Problem 6.1.1, Test 3, $N = 16$.

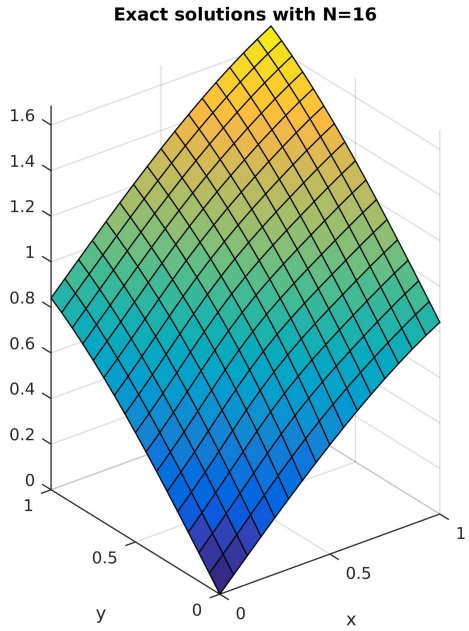

Figure 36: Exact Solutions: Problem 6.1.1, Test 3, $N = 16$.

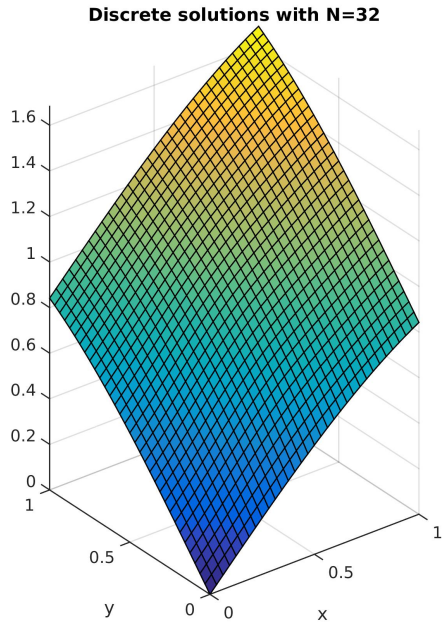

Figure 37: Discrete Solutions: Problem 6.1.1, Test 3, $N = 32$.

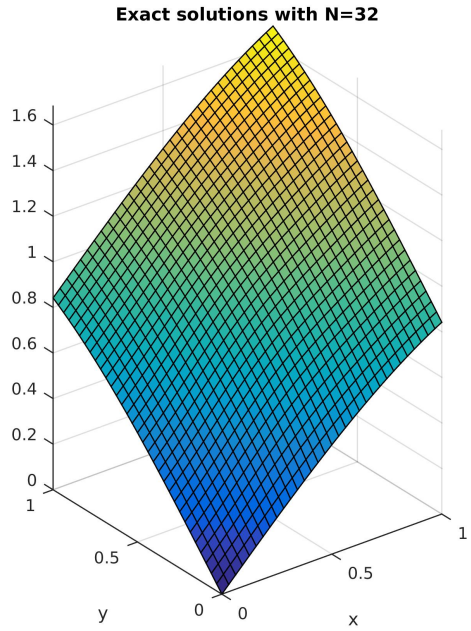

Figure 38: Exact Solutions: Problem 6.1.1, Test 3, $N = 32$.

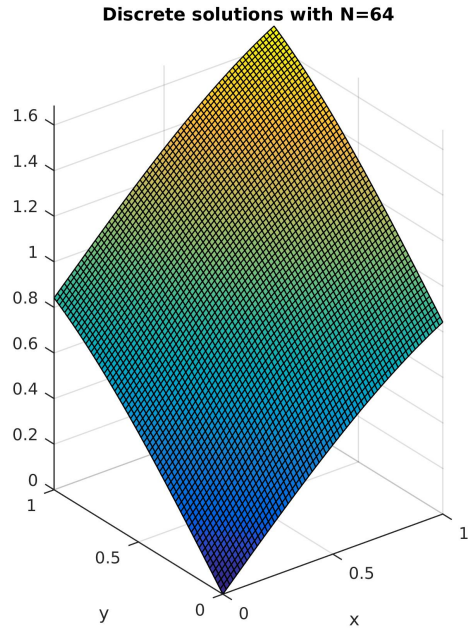

Figure 39: Discrete Solutions: Problem 6.1.1, Test 3, $N = 64$.

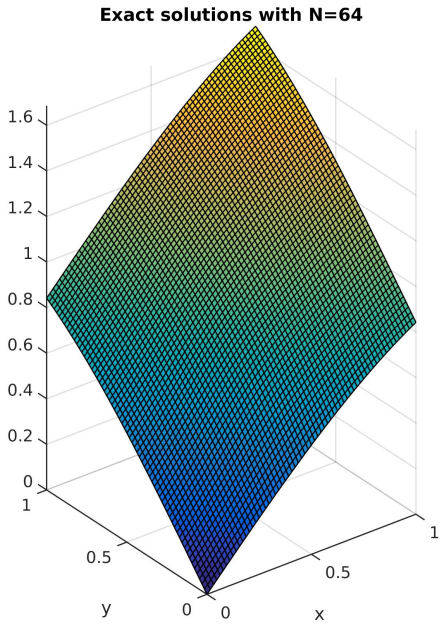

Figure 40: Exact Solutions: Problem 6.1.1, Test 3, $N = 64$.

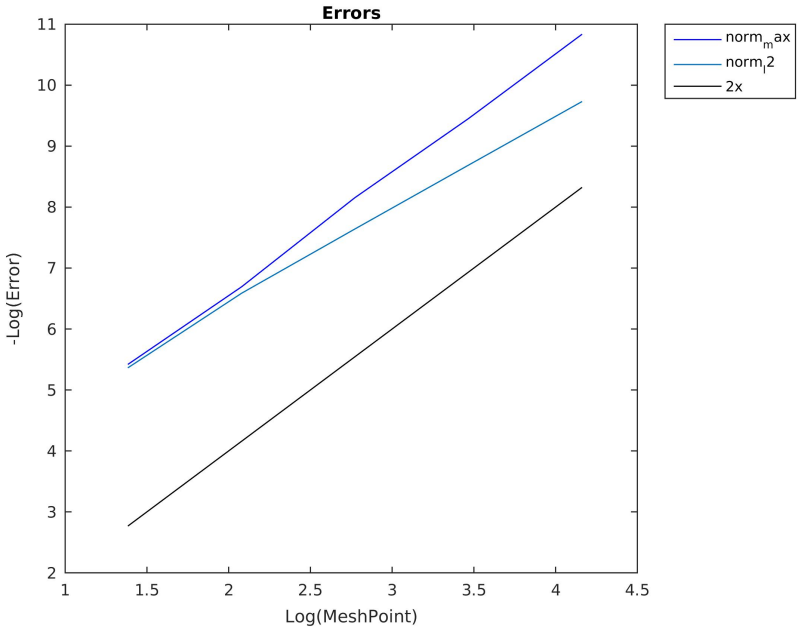

Figure 41: Error on a log-log scale: Problem 6.1.1, Test 3.

TEST 4.

$$g = x^2 + y^3 \tag{7.10}$$

$$u_{ex} = x^2 + y^3 + 50 \sin\left(x^2(x-1)^2 y^2 (y-1)^2\right) \tag{7.11}$$

$$f = -\Delta\left(x^2 + y^3 + 50 \sin\left(x^2(x-1)^2 y^2 (y-1)^2\right)\right) \tag{7.12}$$

Run the above scripts for `cases = 4`, MATLAB returns

```
N = 4
error on L^infinity: 4.425022e-02f
error on L^2: 4.516836e-02f

N = 8
error on L^infinity: 1.090585e-02f
error on L^2: 1.588699e-02f

N = 16
error on L^infinity: 2.719630e-03f
error on L^2: 5.615389e-03f

N = 32
error on L^infinity: 6.795323e-04f
error on L^2: 1.985327e-03f

N = 64
error on L^infinity: 1.698605e-04f
error on L^2: 7.019206e-04f
```

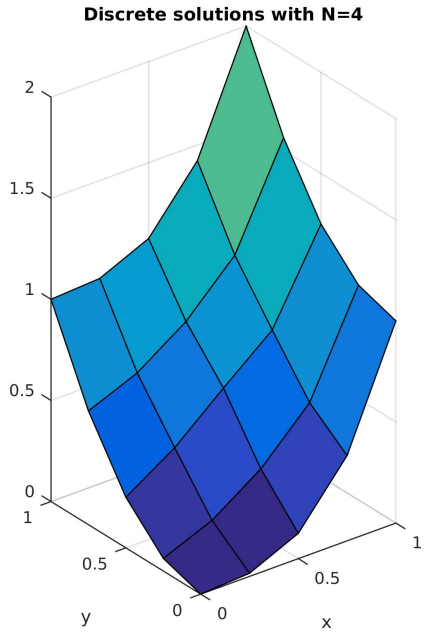

Figure 42: Discrete Solutions: Problem 6.1.1, Test 4, $N = 4$.

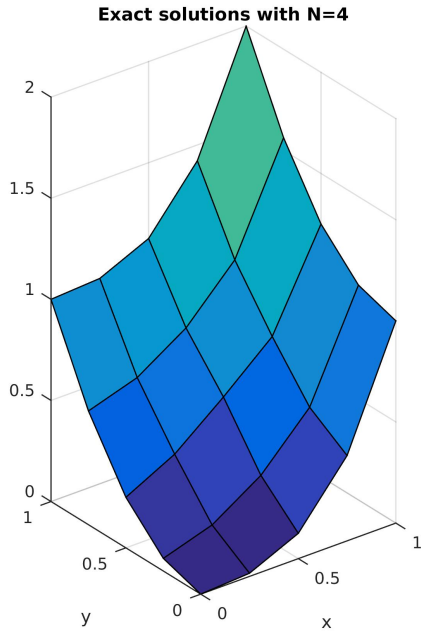

Figure 43: Exact Solutions: Problem 6.1.1, Test 4, $N = 4$.

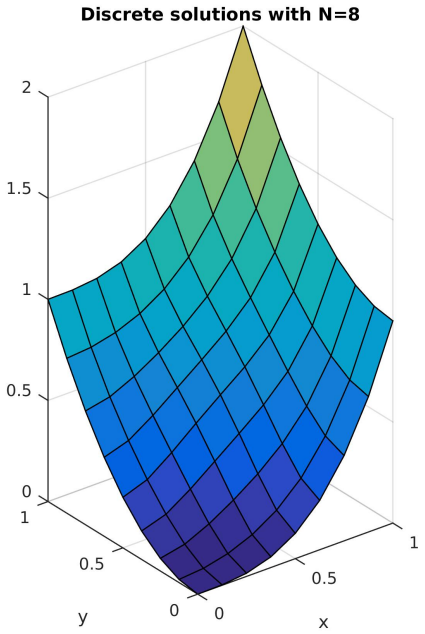

Figure 44: Discrete Solutions: Problem 6.1.1, Test 4, $N = 8$.

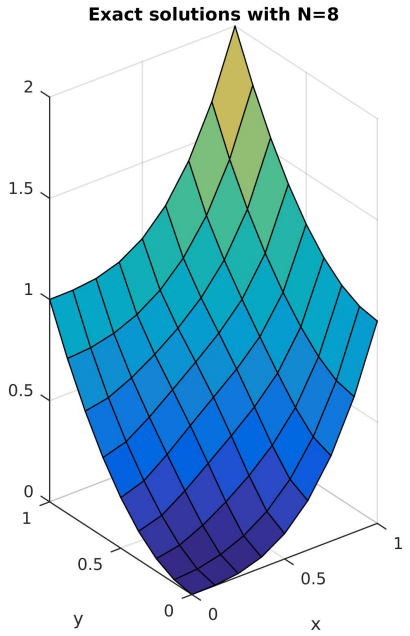

Figure 45: Exact Solutions: Problem 6.1.1, Test 4, $N = 8$.

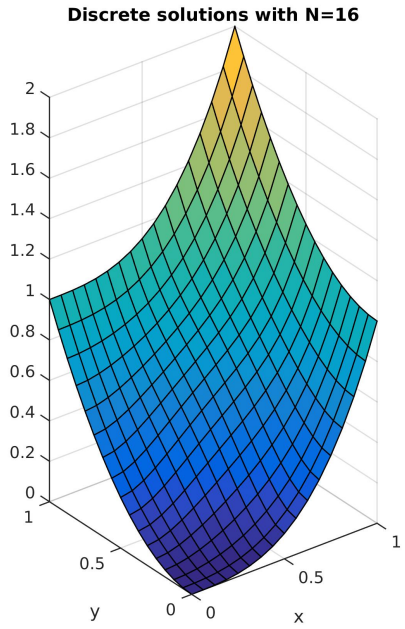

Figure 46: Discrete Solutions: Problem 6.1.1, Test 4, $N = 16$.

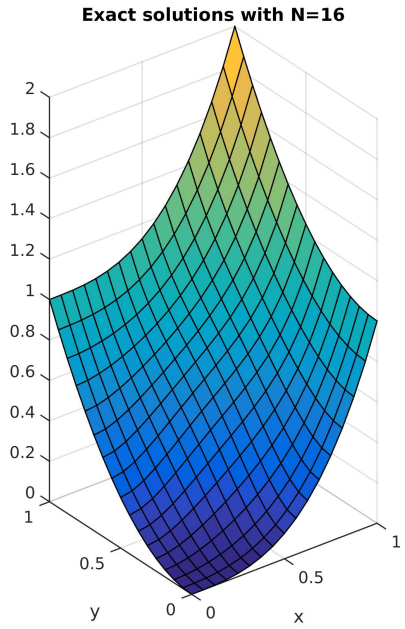

Figure 47: Exact Solutions: Problem 6.1.1, Test 4, $N = 16$.

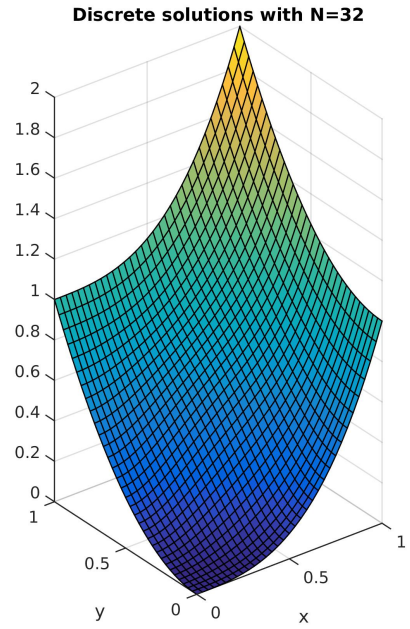

Figure 48: Discrete Solutions: Problem 6.1.1, Test 4, $N = 32$.

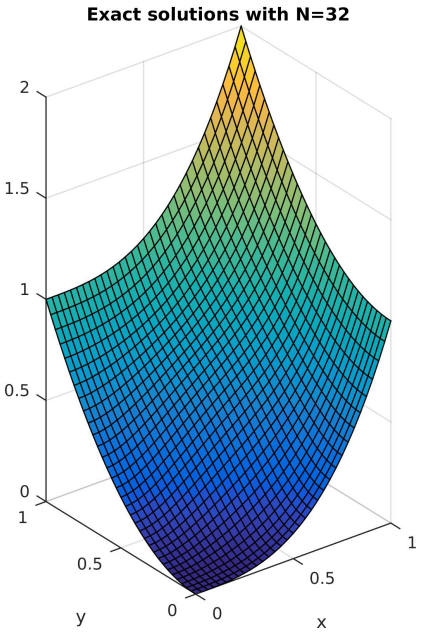

Figure 49: Exact Solutions: Problem 6.1.1, Test 4, $N = 32$.

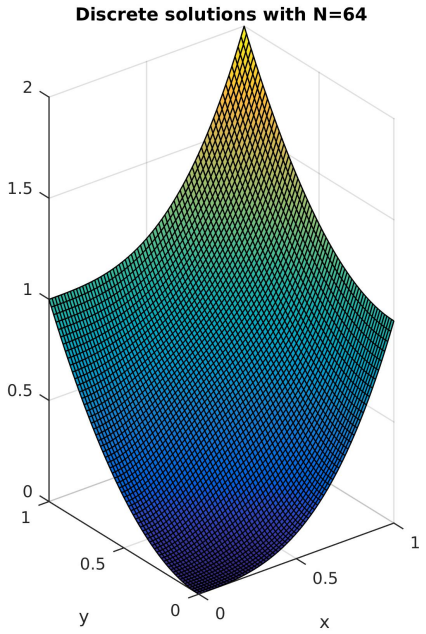

Figure 50: Discrete Solutions: Problem 6.1.1, Test 4, $N = 64$.

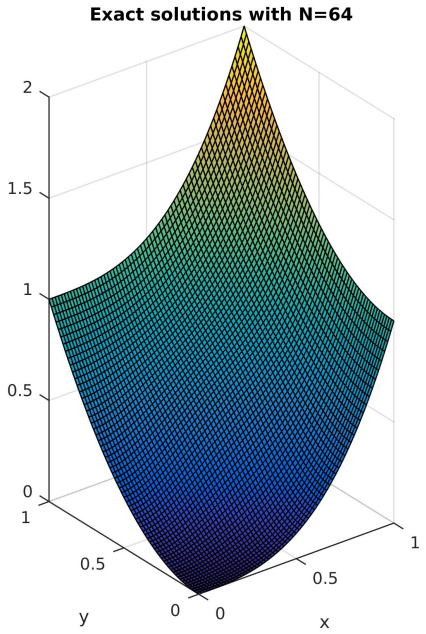

Figure 51: Exact Solutions: Problem 6.1.1, Test 4, $N = 64$.

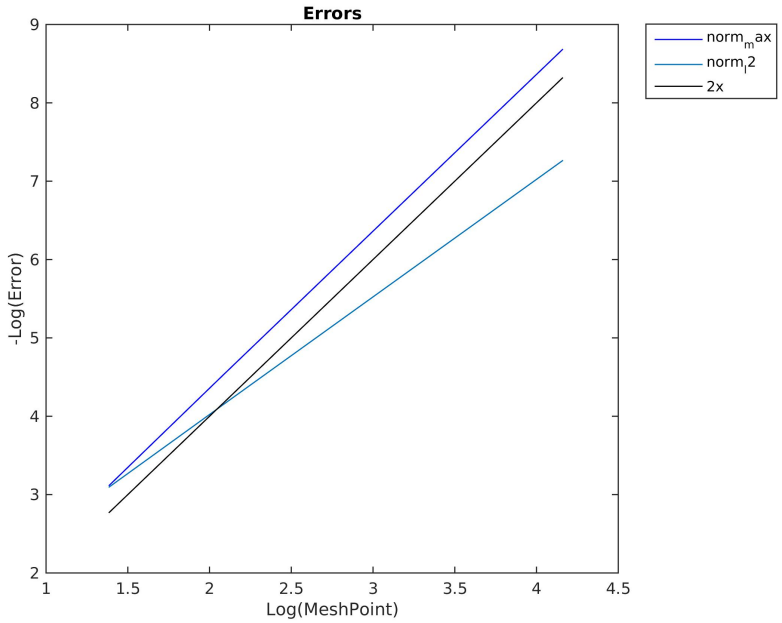

Figure 52: Error on a log-log scale: Problem 6.1.1, Test 4.

TEST 5.

$$g = \sin x + \cos y \tag{7.13}$$

$$u_{ex} = \sin x + \cos y + \sin\left(x^4(x-1)^4y^4(y-1)^4\right) \tag{7.14}$$

$$f = -\Delta\left(\sin x + \cos y + \sin\left(x^4(x-1)^4y^4(y-1)^4\right)\right) \tag{7.15}$$

Run the above scripts for `cases = 3`, MATLAB returns

```
N = 4
error on L^infinity: 4.930484e-04f
error on L^2: 5.402760e-04f

N = 8
error on L^infinity: 1.271770e-04f
error on L^2: 1.993044e-04f

N = 16
error on L^infinity: 3.225832e-05f
error on L^2: 7.116878e-05f

N = 32
error on L^infinity: 8.102735e-06f
error on L^2: 2.522322e-05f

N = 64
error on L^infinity: 2.028070e-06f
error on L^2: 8.923146e-06f
```

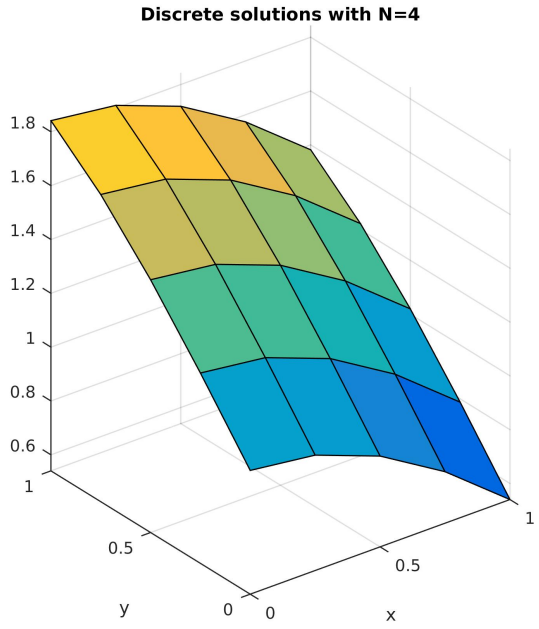

Figure 53: Discrete Solutions: Problem 6.1.1, Test 5, $N = 4$.

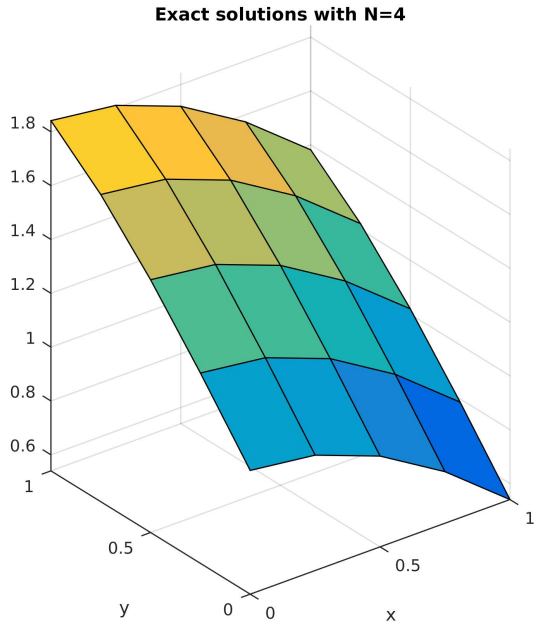

Figure 54: Exact Solutions: Problem 6.1.1, Test 5, $N = 4$.

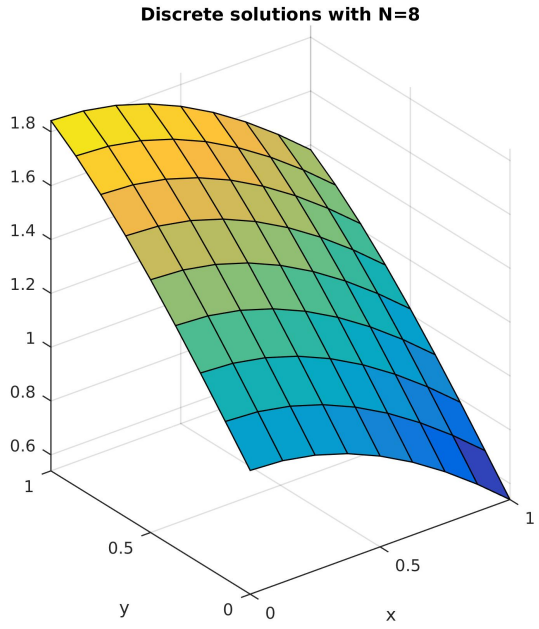

Figure 55: Discrete Solutions: Problem 6.1.1, Test 5, $N = 8$.

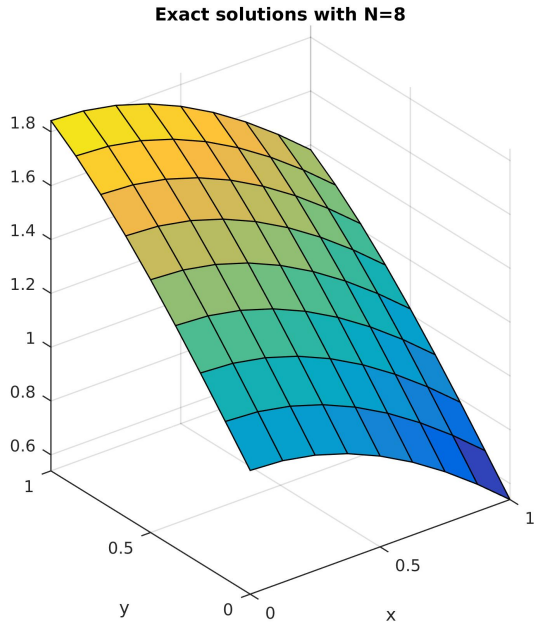

Figure 56: Exact Solutions: Problem 6.1.1, Test 5, $N = 8$.

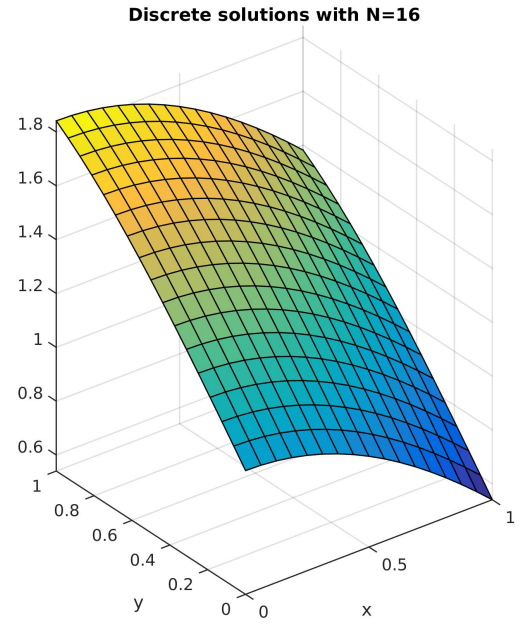

Figure 57: Discrete Solutions: Problem 6.1.1, Test 5, $N = 16$.

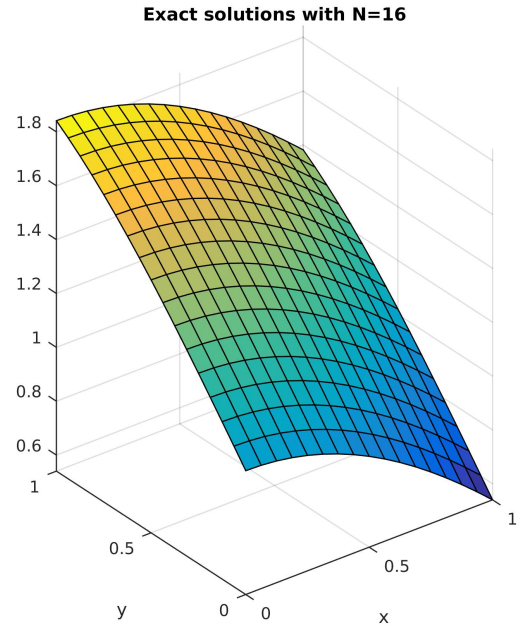

Figure 58: Exact Solutions: Problem 6.1.1, Test 5, $N = 16$.

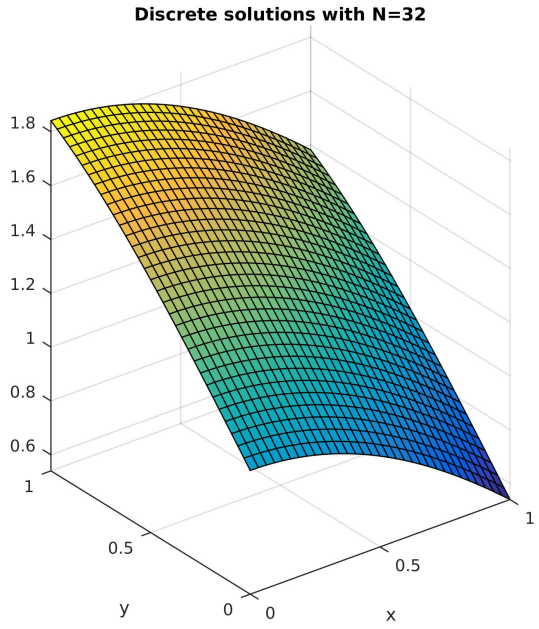

Figure 59: Discrete Solutions: Problem 6.1.1, Test 5, $N = 32$.

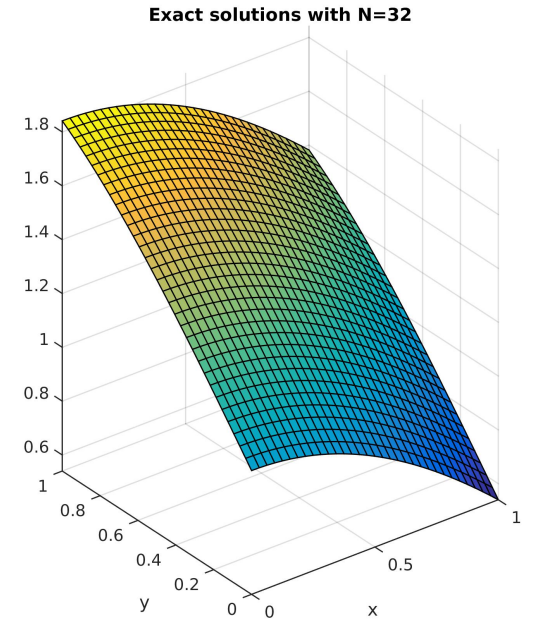

Figure 60: Exact Solutions: Problem 6.1.1, Test 5, $N = 32$.

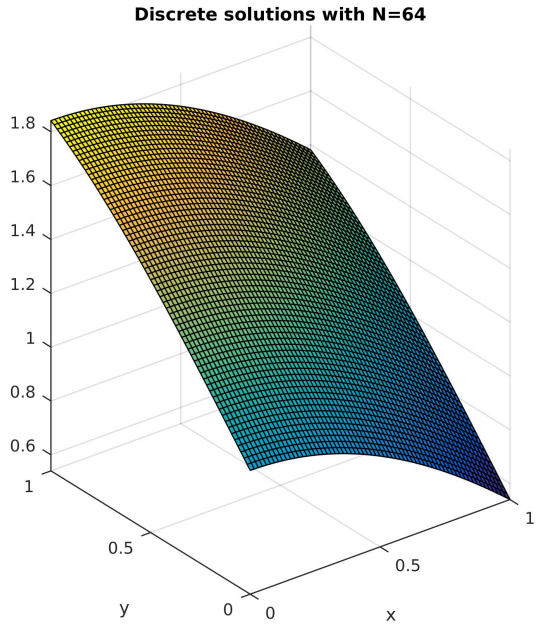

Figure 61: Discrete Solutions: Problem 6.1.1, Test 5, $N = 64$.

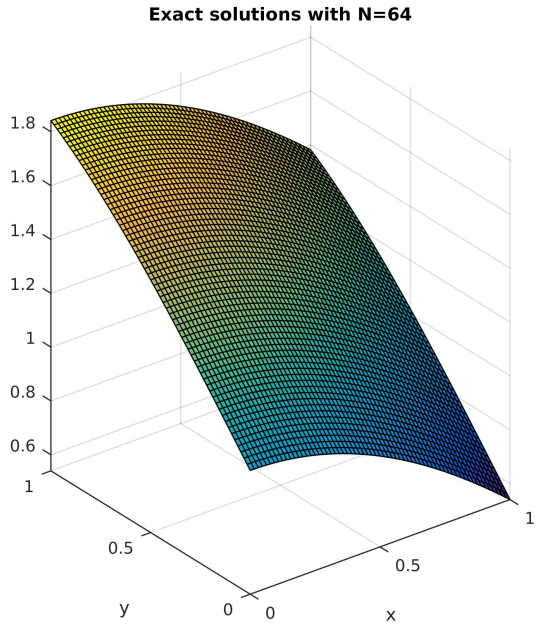

Figure 62: Exact Solutions: Problem 6.1.1, Test 5, $N = 64$.

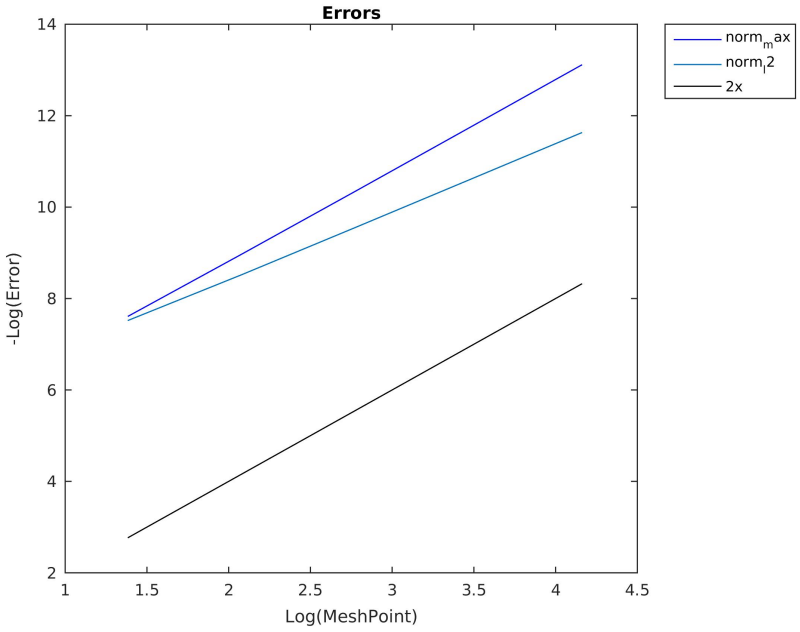

Figure 63: Error on a log-log scale: Problem 6.1.1, Test 5.

7.2 Matlab Implementation for Problem 6.1.2

7.2.1 Matlab Scripts

exact_solution.m

```
function u_ex=exact_solution2D(x,y,k);

if(k==1)
    u_ex=10^4*sin(x^3*(x-1)^3*y^3*(y-1)^3);
end

if(k==2)
    u_ex=8*x*(1-x)*y*(1-y);
end

if(k==3)
    u_ex=30*x^2*y*(x - 1)*(y - 1)^3;
end
```

functionf2D.m

```
function f=functionf2D(x,y,k);

if(k==1)
    f=-10^4*(cos(x^3*y^3*(x - 1)^3*(y - 1)^3)*(6*x*y^3*(x - 1)^3* ...
    (y - 1)^3 + 3*x^3*y^3*(2*x - 2)*(y - 1)^3 + ...
    18*x^2*y^3*(x - 1)^2*(y - 1)^3) - ...
    sin(x^3*y^3*(x - 1)^3*(y - 1)^3)*(3*x^3*y^2*(x - 1)^3*(y - 1)^3 + ...
    3*x^3*y^3*(x - 1)^3*(y - 1)^2)^2 - ...
    sin(x^3*y^3*(x - 1)^3*(y - 1)^3)*(3*x^2*y^3*(x - 1)^3*(y - 1)^3 + ...
    3*x^3*y^3*(x - 1)^2*(y - 1)^3)^2 + ...
    cos(x^3*y^3*(x - 1)^3*(y - 1)^3)*(6*x^3*y*(x - 1)^3*(y - 1)^3 + ...
    3*x^3*y^3*(2*y - 2)*(x - 1)^3 + 18*x^3*y^2*(x - 1)^3*(y - 1)^2));
end

if(k==2)
    f = - 16*x*(x - 1) - 16*y*(y - 1);
end

if(k==3)
    f = - 180*x^2*(x - 1)*(y - 1)^2 - 120*x*y*(y - 1)^3 - ...
     60*y*(x - 1)*(y - 1)^3 - 90*x^2*y*(2*y - 2)*(x - 1);
end
```

main2D_1b.m

```
%% solve equation -u_xx(x,y)-u_yy(x,y)=f(x,y)
%% with the Dirichlet boundary condition
clear all
close all
clc
format long
```

```
tic

%% Initial informations
ax=0.0;
bx=1.0;
ay=0.0;
by=1.0;
cases=2;
Nx=3;% number of mesh points of first mesh
Ny=4;% number of mesh points of first mesh
N=max(Nx,Ny);
number_mesh=3;
number_mesh_point=zeros(number_mesh,1);
norm_max=zeros(number_mesh,1);
norm_l2=zeros(number_mesh,1);
norm_maxh1=zeros(number_mesh,1);
norm_h1=zeros(number_mesh,1);

%% Solve discrite solution and refine mesh
for inumber_mesh=1:number_mesh
    h=(bx-ax)/Nx;
    k=(by-ay)/Ny;
    number_mesh_point(inumber_mesh)=Nx;

    %% Create mesh point
    x=zeros(Nx+1,1);
    for i_iter=1:Nx+1
        x(i_iter)=(i_iter-1)*h;
    end
    y=zeros(Ny+1,1);
    for i_iter=1:Ny+1
        y(i_iter)=(i_iter-1)*k;
    end

    %% Create matrix A
    A1=zeros((Nx-1)*(Ny-1),(Nx-1)*(Ny-1));
    A2=zeros((Nx-1)*(Ny-1),(Nx-1)*(Ny-1));
    B1=zeros(Nx-1,Nx-1);
    I_h=sparse(Nx-1,Nx-1);
    for i=1:Nx-1
        I_h(i,i)=1;
    end
    for i=1:Nx-1
        if(i==1)
            B1(i,i)=2;
            B1(i,i+1)=-1;
        else
            if(i==Nx-1)
                B1(i,i)=2;
```

```
            B1(i,i-1)=-1;
        else
            B1(i,i)=2;
            B1(i,i-1)=-1;
            B1(i,i+1)=-1;
        end
    end
end

for i=1:Ny-1
    if(i==1)
        A1((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=B1;
    else
        if(i==Nx-1)
            A1((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=B1;
        else
            A1((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=B1;
        end
    end
end

for i=1:Ny-2
    A2((i-1)*(Nx-1)+1:i*(Nx-1),i*(Nx-1)+1:(i+1)*(Nx-1))=-I_h;
end
A2=A2+A2';
for i=1:(Nx-1)*(Ny-1)
    A2(i,i)=2;
end
A1;
A2;
C2=A1+A2;
A=Nx^2*A1+Ny^2*A2;

%% Create vector F
F2=zeros((Nx-1)*(Ny-1),1);
for i2=1:Ny-1
    for i1=1:Nx-1
        F2((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases);
    end
end

%% Solve discrete solution
u2=A\F2;

%% Get exact solution
u_exact=zeros((Nx+1),(Ny+1));
for i2=1:Ny+1
    for i1=1:Nx+1
        u_exact(i1,i2)=exact_solution2D(x(i1),y(i2),cases);
    end
```

```
end

%% Create discrete solution with boundary
u_dis=zeros((Nx+1),(Ny+1));
for i2=1:Ny-1
    for i1=1:Nx-1
        u_dis(i1+1,i2+1)=u2((i2-1)*(Nx-1)+i1);
    end
end

%% Reshape u_dis and u_exact
udis=reshape(u_dis,size(u_dis,1)*size(u_dis,2),1);
uexact=reshape(u_exact,size(u_exact,1)*size(u_exact,2),1);

%% Calculate the error on L^infinity
norm_max(inumber_mesh)=0.0;
for i=1:length(udis)
    if (abs(u_dis(i)-u_exact(i)) > norm_max(inumber_mesh))
        norm_max(inumber_mesh)=abs(u_dis(i)-u_exact(i));
    end
end
fprintf('error on L^infinity: %df\n',norm_max(inumber_mesh));

%%  Calculate the error on L^2
norm_l2(inumber_mesh)=0;
for i=1:length(udis)
    norm_l2(inumber_mesh)=norm_l2(inumber_mesh)+...
    (u_dis(i)-u_exact(i))^2*h;
end
norm_l2(inumber_mesh)=(norm_l2(inumber_mesh))^(1/2);
fprintf('error on L^2: %df\n',norm_l2(inumber_mesh));

%% Figure exact and dicrete solutions
figure
surf(x,y,u_dis')
xlabel('x')
ylabel('y')
axis equal
str=sprintf('Discrete solutions with N_x=%d, N_y=%d',Nx,Ny);
title(str);
print('-r300','-djpeg');
%    zlim([-1 1])
figure
surf(x,y,u_exact')
xlabel('x')
ylabel('y')
axis equal
str=sprintf('Exact solutions with N_x=%d, N_y=%d',Nx,Ny);
title(str);
print('-r300','-djpeg');
```

```
    %    zlim([-1 1]);

    %% Refine mesh
    Nx=4*Nx;
    Ny=4*Ny;
end

%% Figure for errors respect to number of mesh point
figure
plot(log(number_mesh_point), -log(norm_max),'blue', ...
    -log(number_mesh_point), log(norm_l2), log(number_mesh_point), ...
     2*log(number_mesh_point),'black');
xlabel('Log(MeshPoint)');ylabel('-Log(Error)');
title('Errors');
legend('norm_max','norm_l2','2x','Location','NorthEastOutside');
print('-r300','-djpeg');

toc
```

7.2.2 Results

Test 1.

$$u_{ex}(x,y) = 10^4 \sin\left(x^3(x-1)^3y^3(y-1)^3\right) \tag{7.16}$$

$$f(x,y) = -\Delta\left(10^4 \sin\left(x^3(x-1)^3y^3(y-1)^3\right)\right) \tag{7.17}$$

Run the above scripts, MATLAB returns

```
Nx = 3, Ny = 4
error on L^infinity: 5.966169e-01f
error on L^2: 5.540446e-01f

Nx = 12, Ny = 16
error on L^infinity: 3.733171e-02f
error on L^2: 5.096791e-02f

Nx = 48, Ny = 64
error on L^infinity: 2.250012e-03f
error on L^2: 6.309561e-03f
```

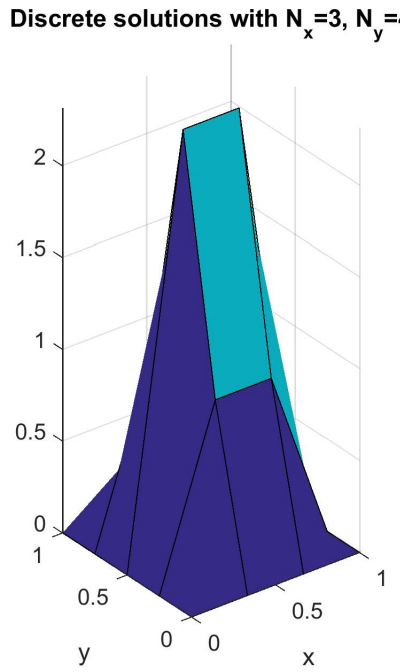

Figure 64: Discrete Solutions: Problem 6.1.2, Test 1, $N_x = 3, N_y = 4$.

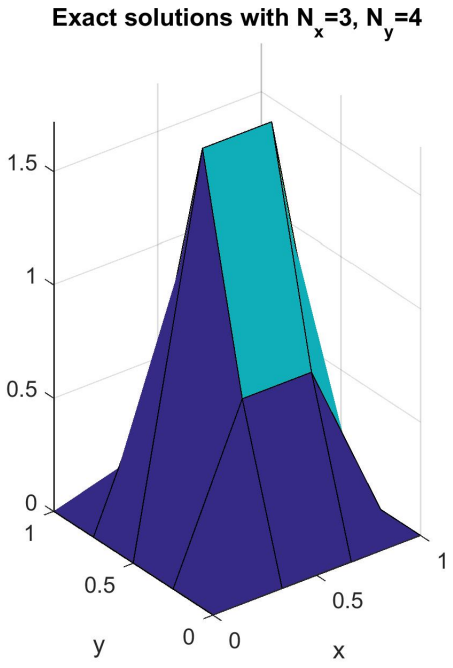

Figure 65: Exact Solutions: Problem 6.1.2, Test 1, $N_x = 3, N_y = 4$.

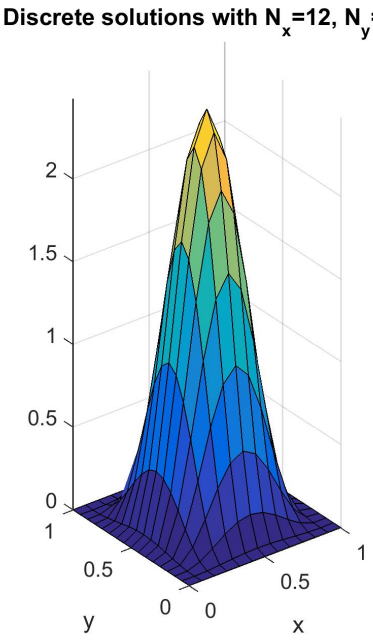

Figure 66: Discrete Solutions: Problem 6.1.2, Test 1, $N_x = 12, N_y = 16$.

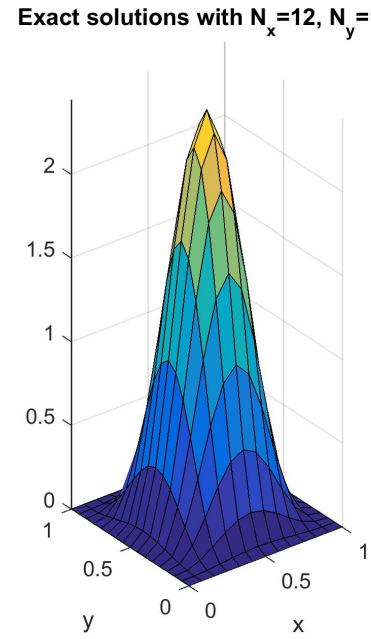

Figure 67: Exact Solutions: Problem 6.1.2, Test 2, $N_x = 12, N_y = 16$.

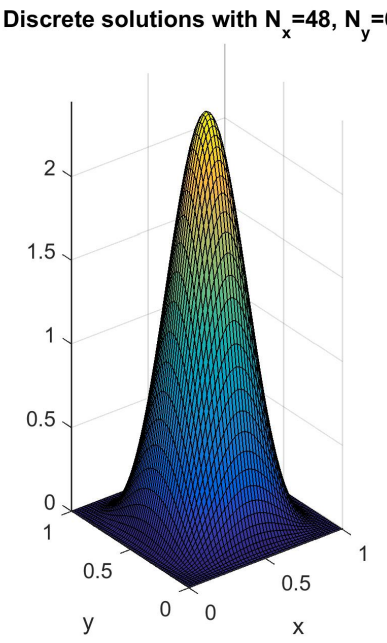

Figure 68: Discrete Solutions: Problem 6.1.2, Test 1, $N_x = 48, N_y = 64$.

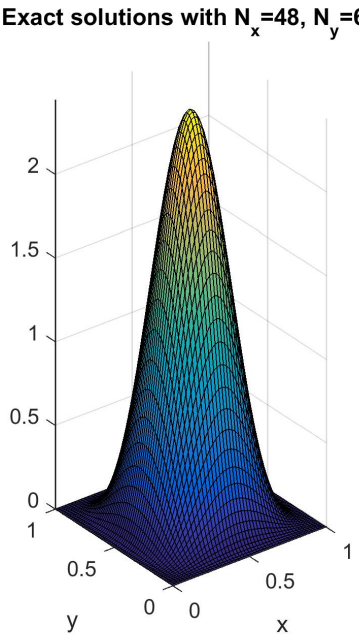

Figure 69: Exact Solutions: Problem 6.1.2, Test 1, $N_x = 48, N_y = 64$.

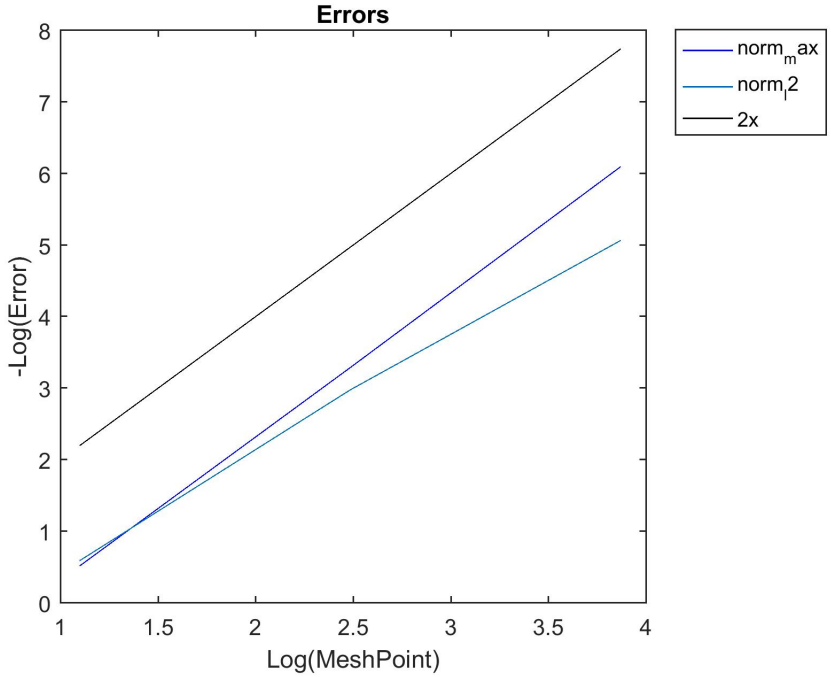

Figure 70: Error on a log-log scale: Problem 6.1.2. Test 1

TEST 2.

$$u_{ex}(x,y) = 8x(1-x)\,y(1-y) \tag{7.18}$$

$$f(x,y) = -\Delta\left(8x(1-x)\,y(1-y)\right) \tag{7.19}$$

Run the above scripts, MATLAB returns

```
Nx = 3, Ny = 4
error on L^infinity: 4.551914e-15f
error on L^2: 1.104501e-14f

Nx = 12, Ny = 16
error on L^infinity: 8.326673e-16f
error on L^2: 1.205209e-15f

Nx = 48, Ny = 64
error on L^infinity: 1.665335e-16f
error on L^2: 2.001483e-16f
```

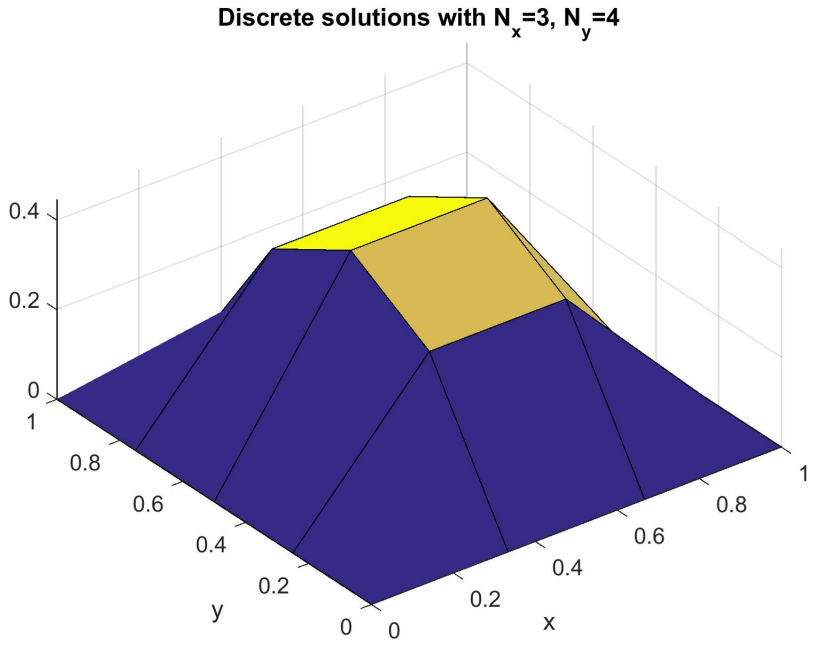

Figure 71: Discrete Solutions: Problem 6.1.2, Test 2, $N_x = 3, N_y = 4$.

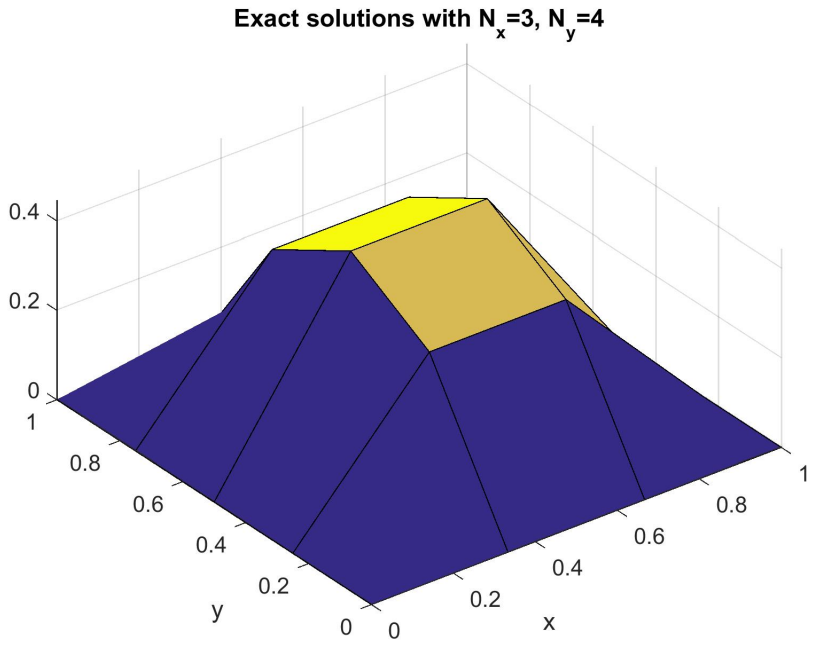

Figure 72: Exact Solutions: Problem 6.1.2, Test 2, $N_x = 3, N_y = 4$.

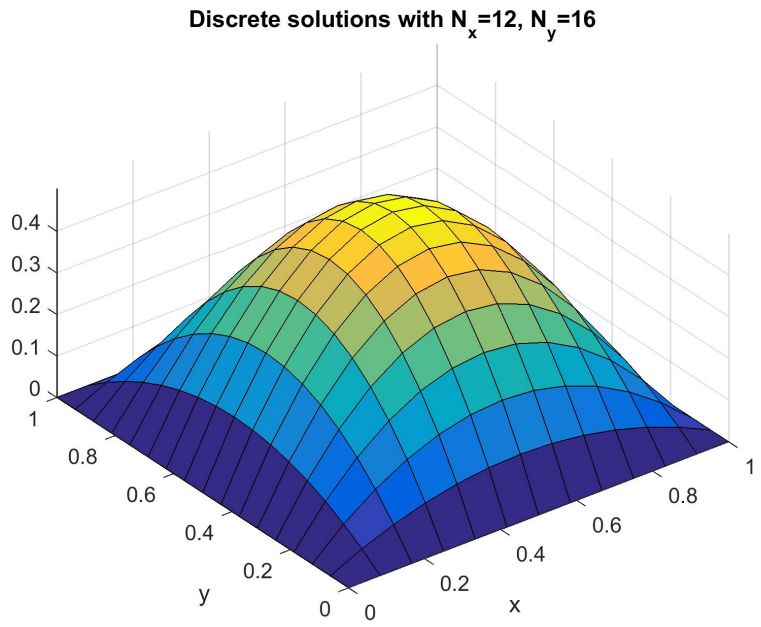

Figure 73: Discrete Solutions: Problem 6.1.2, Test 2, $N_x = 12, N_y = 16$.

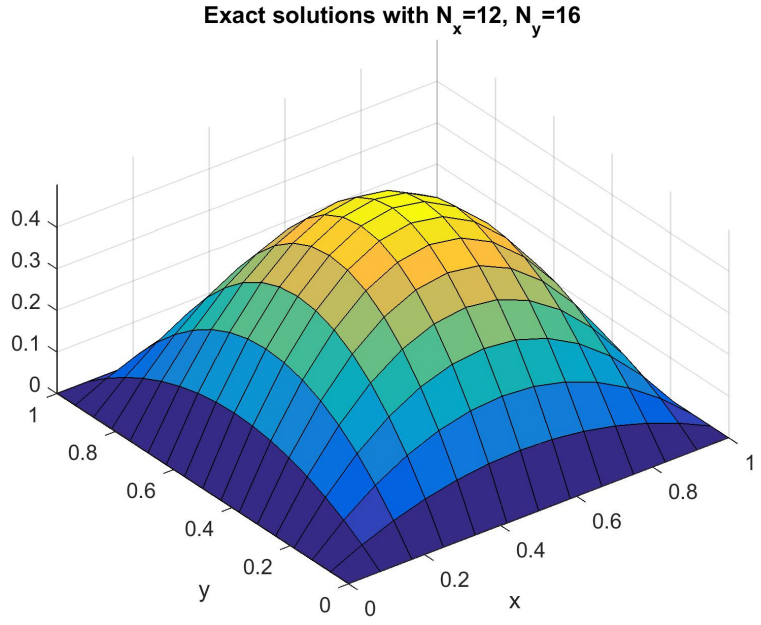

Figure 74: Exact Solutions: Problem 6.1.2, Test 2, $N_x = 12, N_y = 16$.

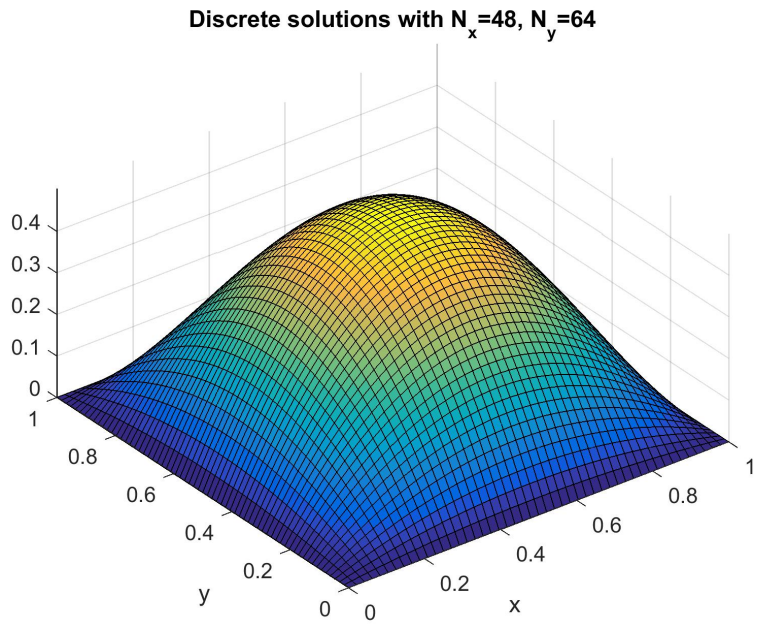

Figure 75: Discrete Solutions: Problem 6.1.2, Test 2, $N_x = 48, N_y = 64$.

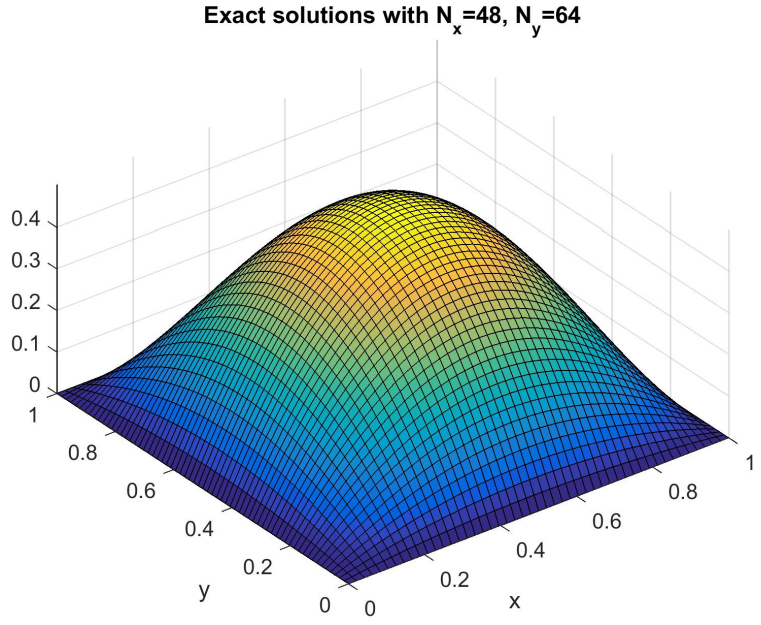

Figure 76: Exact Solutions: Problem 6.1.2, Test 2, $N_x = 48, N_y = 64$.

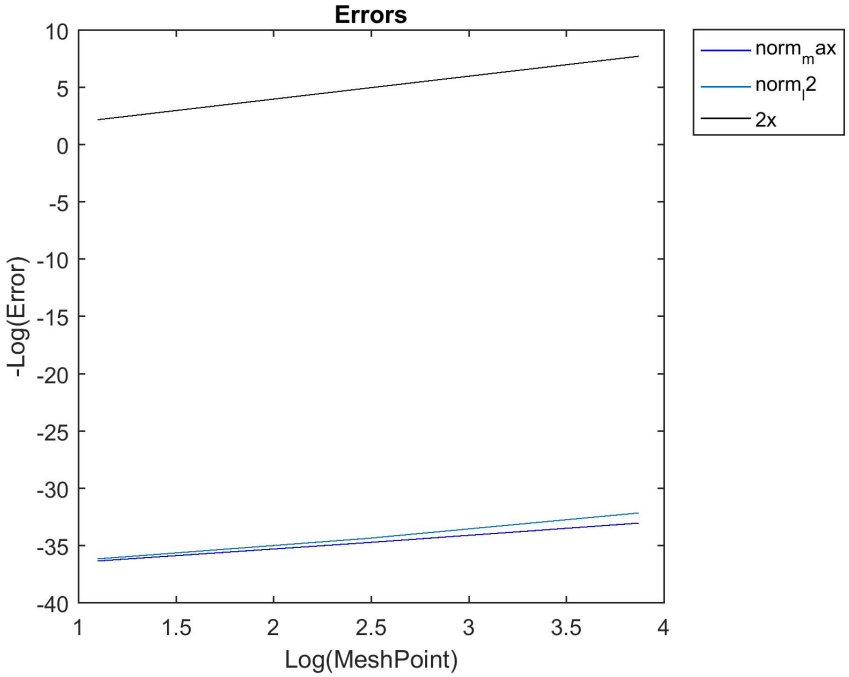

Figure 77: Error on a log-log scale: Problem 6.1.2. Test 2

TEST 3.

$$u_{ex}(x,y) = 10^4 \sin\left(x^3(x-1)^3 y^3 (y-1)^3\right) \tag{7.20}$$

$$f(x,y) = -\Delta\left(10^4 \sin\left(x^3(x-1)^3 y^3 (y-1)^3\right)\right) \tag{7.21}$$

Run the above scripts, MATLAB returns

```
Nx = 3, Ny = 4
error on L^infinity: 3.027634e-02f
error on L^2: 3.209642e-02f

Nx = 12, Ny = 16
error on L^infinity: 1.899742e-03f
error on L^2: 3.960307e-03f

Nx = 48, Ny = 64
error on L^infinity: 1.185024e-04f
error on L^2:4.943346e-04f
```

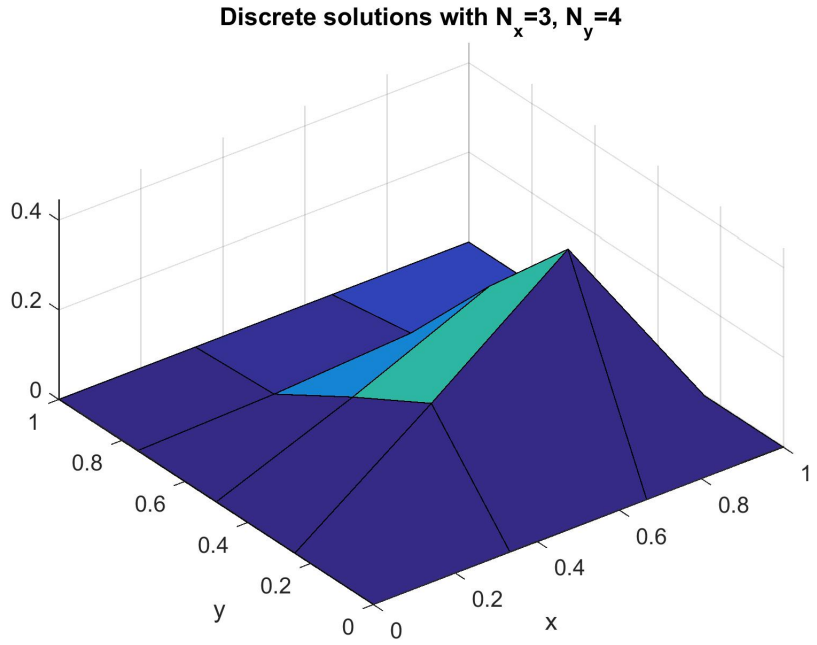

Figure 78: Discrete Solutions: Problem 6.1.2, Test 3, $N_x = 3, N_y = 4$.

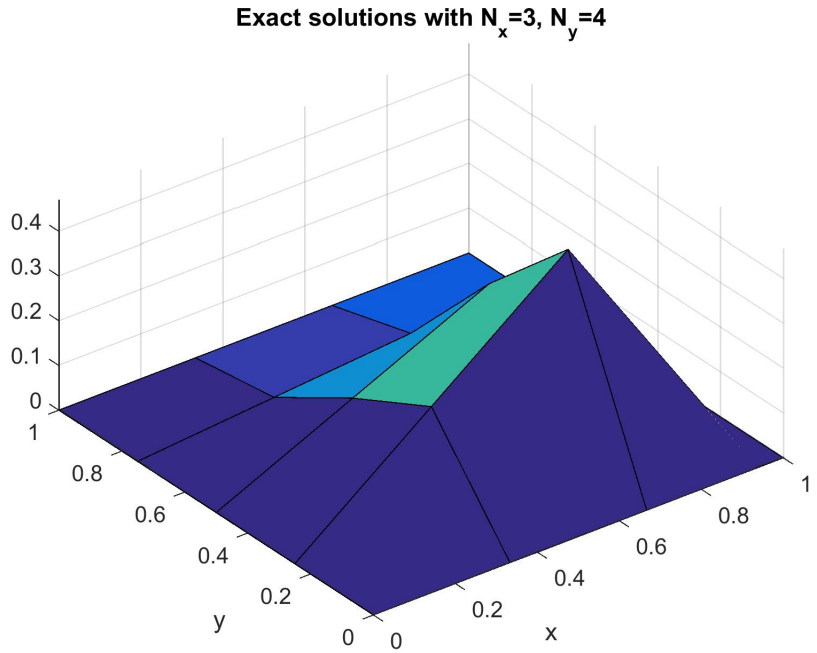

Figure 79: Exact Solutions: Problem 6.1.2, Test 3, $N_x = 3, N_y = 4$.

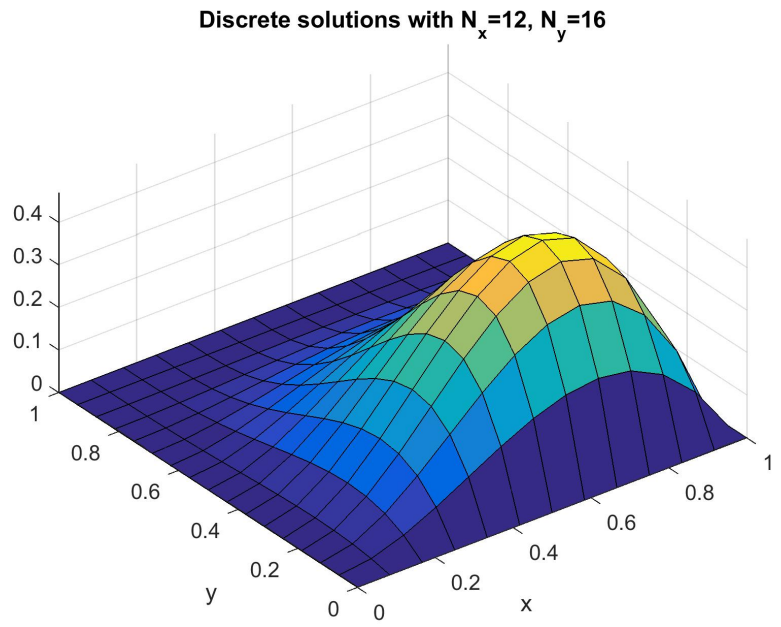

Figure 80: Discrete Solutions: Problem 6.1.2, Test 3, $N_x = 12, N_y = 16$.

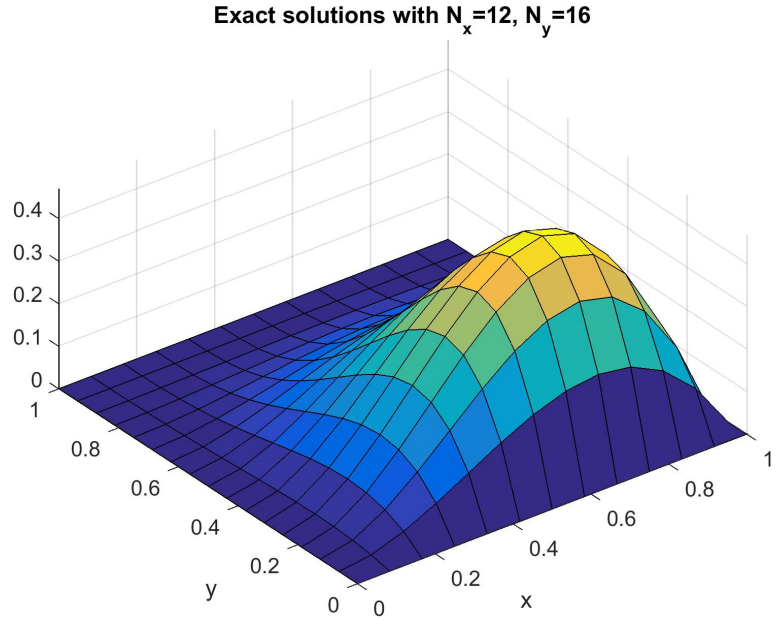

Figure 81: Exact Solutions: Problem 6.1.2, Test 3, $N_x = 12, N_y = 16$.

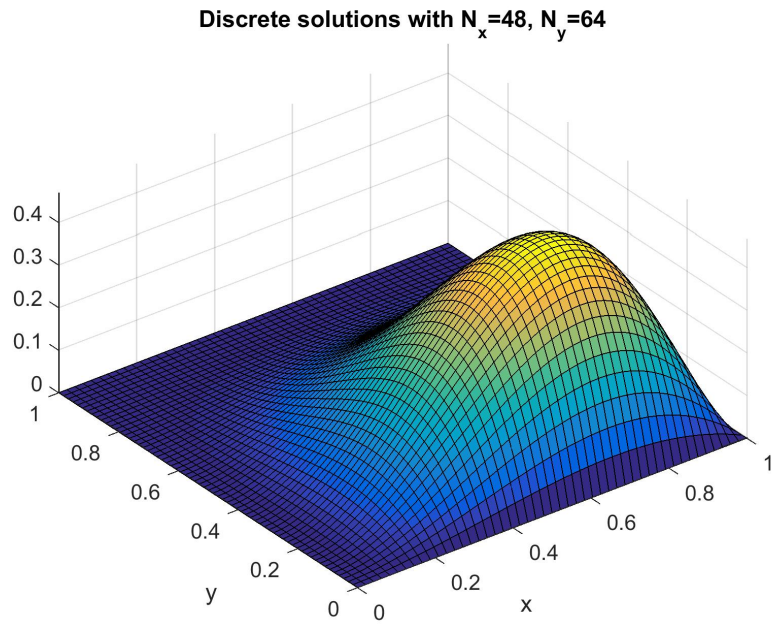

Figure 82: Discrete Solutions: Problem 6.1.2, Test 3, $N_x = 48, N_y = 64$.

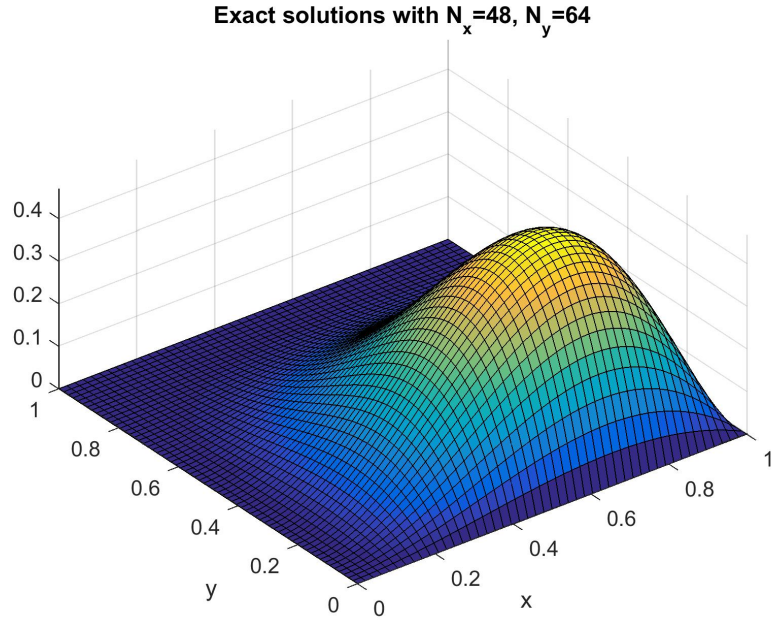

Figure 83: Exact Solutions: Problem 6.1.2, Test 3, $N_x = 48, N_y = 64$.

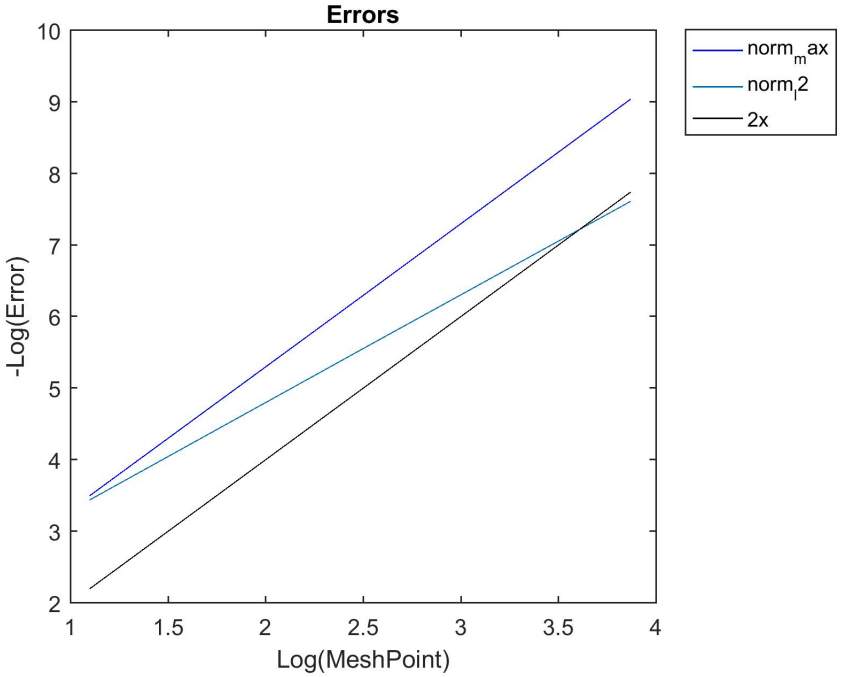

Figure 84: Error on a log-log scale: Problem 6.1.2. Test 3

7.3 Matlab Implementation for Problem 6.2

7.3.1 Matlab Scripts

Warning 7.1. Following codes use Euler differential approximation, thus it significantly reduces the method's convergence rate.

Readers can use second order differential approximation for better convergence rate.

exact_solution2D.m

```
function u_ex=exact_solution2D(x,y,k);

if(k==1)
    u_ex=x^2 + y^2 + x*y*(x - 1)*(y - 1);
end
```

functionf2D.m

```
function f=functionf2D(x,y,k);

if (k==1)
    f = - 2*x*(x - 1) - 2*y*(y - 1) - 4;
end
```

gk.m

```
function f = gk(x,y);

f = x^2+y^2;
```

g4.m

```
function f = g4(y);

f = y*(y - 1) + 2;
```

main2D_2.m

```
%% Initial informations
ax=0.0;
bx=1.0;
ay=0.0;
by=1.0;
cases=1;
N=4;% number of mesh points of first mesh
number_mesh=3;
number_mesh_point=zeros(number_mesh,1);
norm_max=zeros(number_mesh,1);
norm_l2=zeros(number_mesh,1);
norm_maxh1=zeros(number_mesh,1);
norm_h1=zeros(number_mesh,1);

%% Solve discrite solution and refine mesh
```

```
for inumber_mesh=1:number_mesh
    h=(bx-ax)/N;
    number_mesh_point(inumber_mesh)=N;

    %% Create mesh point
    x=zeros(N+1,1);
    for i_iter=1:N+1
        x(i_iter)=(i_iter-1)*h;
    end
    y=zeros(N+1,1);
    for i_iter=1:N+1
        y(i_iter)=(i_iter-1)*h;
    end

    %% Create matrix A
    A=sparse((N-1)*(N-1),(N-1)*(N-1));
    B=sparse(N-1,N-1);
    I_h=sparse(N-1,N-1);
    for i=1:N-1
        I_h(i,i)=1;
    end
    for i=1:N-1
        if(i==1)
            B(i,i)=4;
            B(i,i+1)=-1;
        else
            if(i==N-1)
                B(i,i)=3;
                B(i,i-1)=-1;
            else
                B(i,i)=4;
                B(i,i-1)=-1;
                B(i,i+1)=-1;
            end
        end
    end
    for i=1:N-1
        if(i==1)
            A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B;
            A((i-1)*(N-1)+1:i*(N-1),i*(N-1)+1:(i+1)*(N-1))=-I_h;
        else
            if(i==N-1)
                A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B;
                A((i-1)*(N-1)+1:i*(N-1),(i-2)*(N-1)+1:(i-1)*(N-1))=-I_h;
            else
                A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B;
                A((i-1)*(N-1)+1:i*(N-1),i*(N-1)+1:(i+1)*(N-1))=-I_h;
                A((i-1)*(N-1)+1:i*(N-1),(i-2)*(N-1)+1:(i-1)*(N-1))=-I_h;
            end
        end
```

```
end
A=(1/h^2)*A;

%% Create vector F
F=zeros((N-1)*(N-1),1);
for i1=1:N-1
    if (i1 == 1)
        for i2=1:N-1
            if (i2 == 1)
                F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                (gk(h,0)+gk(0,h))/h^2;
            else
                if (i2 == N-1)
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    (gk(h,1)+ gk(0,(N-1)*h))/h^2;
                else
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    gk(0,i2*h)/h^2;
                end
            end
        end
    else
        if (i1 == N-1)
            for i2=1:N-1
                if (i2 == 1)
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    gk((N-1)*h,0)/h^2 + g4(h)/h;
                else
                    if (i2 == N-1)
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                        gk((N-1)*h,1)/h^2 + g4((N-1)*h)/h;
                    else
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                        g4(i2*h)/h;
                    end
                end
            end
        else
            for i2=1:N-1
                if (i2 == 1)
                    F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                    (gk(i1*h,0))/h^2;
                else
                    if (i2 == N-1)
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases)+ ...
                        (gk(i1*h,1))/h^2;
                    else
                        F((i2-1)*(N-1)+i1)=functionf2D(i1*h,i2*h,cases);
                    end
                end
```

```
            end
        end
    end
end

%% Solve discrete solution
u=A\F;


%% Get exact solution
u_exact=zeros((N+1),(N+1));
for i1=1:N+1
    for i2=1:N+1
        u_exact(i1,i2)=exact_solution2D(x(i1),y(i2),cases);
    end
end

%% Create discrete solution with boundary
u_dis=u_exact;
for i1=1:N-1
    for i2=1:N-1
        u_dis(i1+1,i2+1)=u((i2-1)*(N-1)+i1);
    end
end
%% Reshape u_dis and u_exact
udis=reshape(u_dis,size(u_dis,1)*size(u_dis,2),1);
uexact=reshape(u_exact,size(u_exact,1)*size(u_exact,2),1);
%% Calculate the error on L^infinity
norm_max(inumber_mesh)=0.0;
for i=1:length(udis)
    if (abs(u_dis(i)-u_exact(i)) > norm_max(inumber_mesh))
        norm_max(inumber_mesh)=abs(u_dis(i)-u_exact(i));
    end
end
fprintf('error on L^infinity: %df\n',norm_max(inumber_mesh));
%%  Calculate the error on L^2
norm_l2(inumber_mesh)=0;
for i=1:length(udis)
    norm_l2(inumber_mesh)=norm_l2(inumber_mesh)+ ...
    (u_dis(i)-u_exact(i))^2*h;
end
norm_l2(inumber_mesh)=(norm_l2(inumber_mesh))^(1/2);
fprintf('error on L^2: %df\n',norm_l2(inumber_mesh));
%% Figure exact and dicrete solutions
figure
surf(x,y,u_dis)
xlabel('x')
ylabel('y')
axis equal
str=sprintf('Discrete solutions with N=%d',N);
```

```
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1])
    figure
    surf(x,y,u_exact)
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Exact solutions with N=%d',N);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1]);

    %% Refine mesh
    N=4*N;
end
%% Figure for errors respect to number of mesh point
figure
plot(log(number_mesh_point), -log(norm_max),'blue', ...
     log(number_mesh_point), -log(norm_l2),...
     log(number_mesh_point), 2*log(number_mesh_point),'black');
xlabel('Log(MeshPoint)');ylabel('-Log(Error)');
title('Errors');
legend('norm_max','norm_l2','2x','Location','NorthEastOutside');
print('-r300','-djpeg');
```

7.3.2 Results

TEST 1.

$$u_{ex}(x,y) = 30x(1-x)\,y(1-y) \tag{7.22}$$
$$f(x,y) = -60x(x-1) - 60y(y-1) \tag{7.23}$$
$$g_1(x) = x^2 \tag{7.24}$$
$$g_2(y) = y^2 \tag{7.25}$$
$$g_3(x) = x^2 + 1 \tag{7.26}$$
$$g_4(y) = y(y-1) + 2 \tag{7.27}$$

Run the above scripts, MATLAB returns

```
Nx = 3, Ny = 4
error on L^infinity: 4.951642e-02f
error on L^2: 4.133113e-02f

Nx = 12, Ny = 16
error on L^infinity: 1.657805e-02f
error on L^2: 2.239739e-02f

Nx = 48, Ny = 64
error on L^infinity: 4.420174e-03f
error on L^2: 1.126896e-02f
```

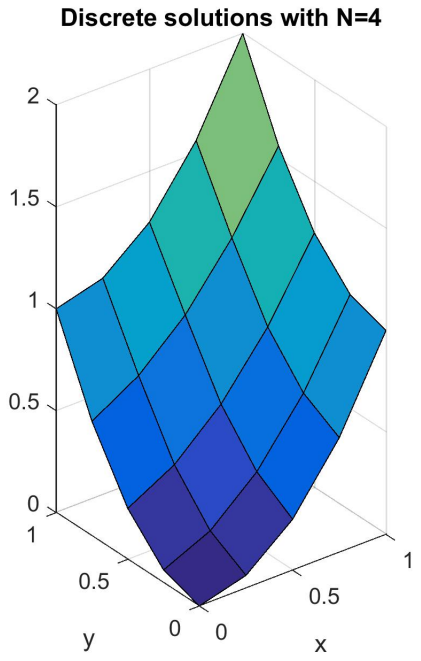

Figure 85: Discrete Solutions: 6.2, $N = 4$

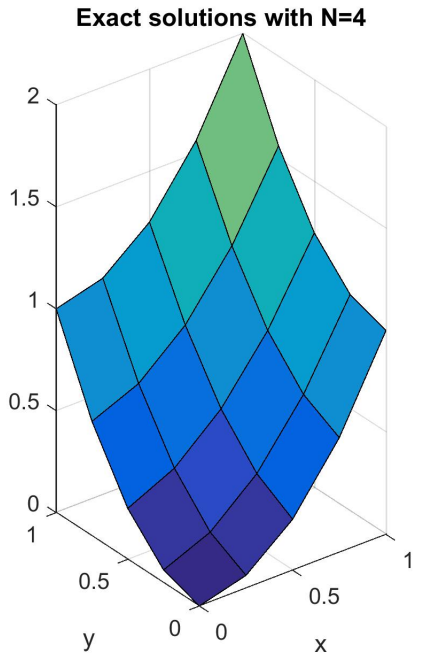

Figure 86: Exact Solutions: 6.2, $N = 4$

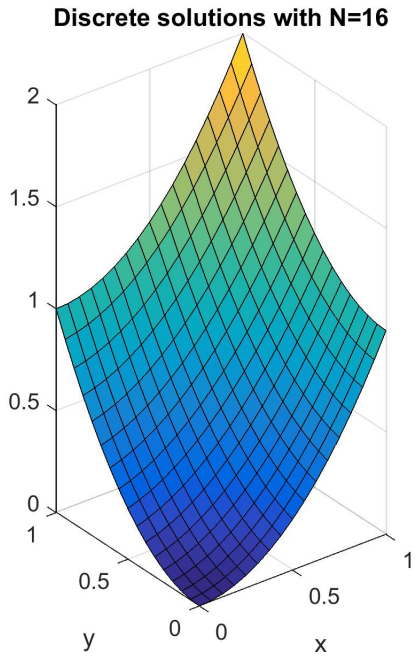

Figure 87: Discrete Solutions: 6.2, $N = 16$

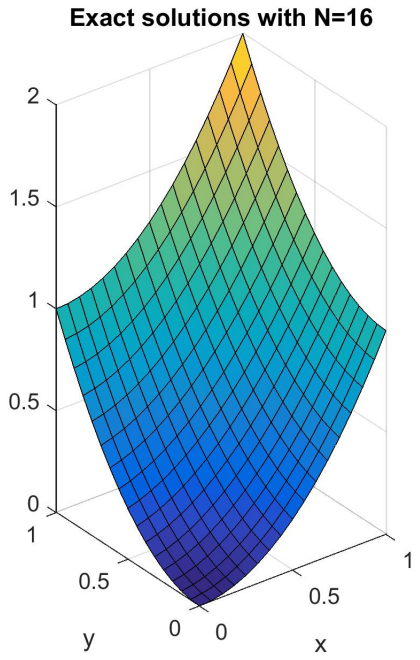

Figure 88: Exact Solutions: 6.2, $N = 16$

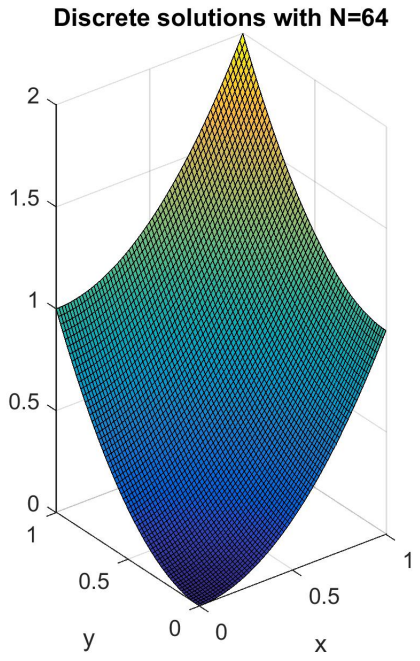

Figure 89: Discrete Solutions: 6.2, $N = 64$

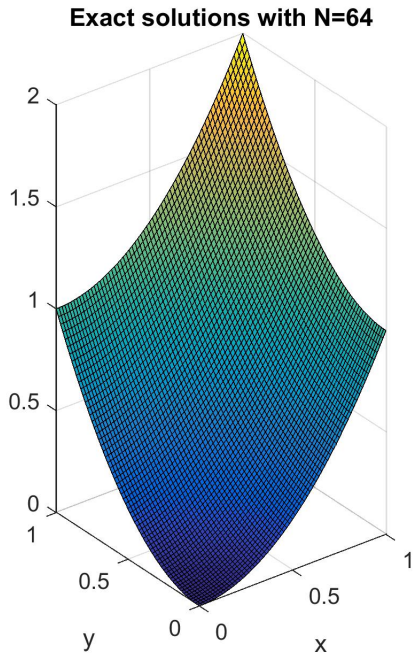

Figure 90: Exact Solutions: Problem 6.2 , $N = 64$

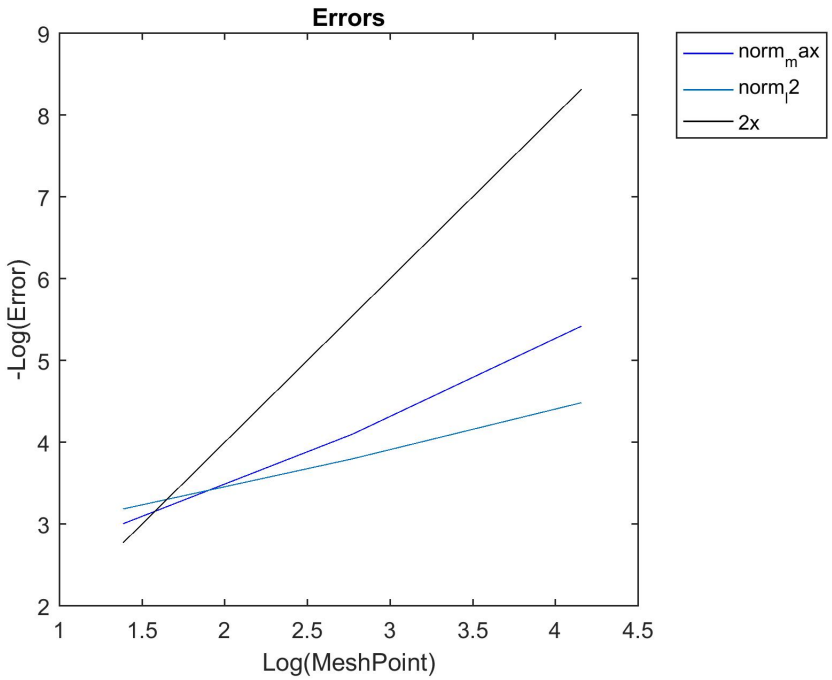

Figure 91: Error on a log-log scale: Problem 6.2

7.4 Matlab Implementation for Problem 6.3

7.4.1 Matlab Scripts

Warning 7.2. Following codes use Euler differential approximation, thus it significantly reduces the method's convergence rate.

Readers can use second order differential approximation for better convergence rate.

With pure Neumann boundry condition, the matrix A can be singular, following codes and example does not get this problem, but in other cases, users can have it.

exact_solution2D.m

```
function u_ex=exact_solution2D(x,y,k);

if(k==1)
    u_ex=sin(x^3*(x-1)^3*y^3*(y-1)^3);
end
```

functionf2D.m

```
function f=functionf2D(x,y,k);

if(k==1)
    f=sin(x^3*y^3*(x - 1)^3*(y - 1)^3)*(3*x^2*y^3*(x - 1)^3*(y - 1)^3 + ...
      3*x^3*y^3*(x - 1)^2*(y - 1)^3)^2 + ...
      sin(x^3*y^3*(x - 1)^3*(y - 1)^3)*(3*x^3*y^2*(x - 1)^3*(y - 1)^3 + ...
      3*x^3*y^3*(x - 1)^3*(y - 1)^2)^2 - ...
      cos(x^3*y^3*(x - 1)^3*(y - 1)^3)*(6*x*y^3*(x - 1)^3*(y - 1)^3 + ...
      3*x^3*y^3*(2*x - 2)*(y - 1)^3 + 18*x^2*y^3*(x - 1)^2*(y - 1)^3) - ...
      cos(x^3*y^3*(x - 1)^3*(y - 1)^3)*(6*x^3*y*(x - 1)^3*(y - 1)^3 + ...
      3*x^3*y^3*(2*y - 2)*(x - 1)^3 + 18*x^3*y^2*(x - 1)^3*(y - 1)^2);
end
```

main2D_3.m

```
clear all
close all
clc
format long

tic

%% Initial informations
ax=0.0;
bx=1.0;
ay=0.0;
by=1.0;
cases=1;
N=4;% number of mesh points of first mesh
number_mesh=3;
number_mesh_point=zeros(number_mesh,1);
```

```
norm_max=zeros(number_mesh,1);
norm_l2=zeros(number_mesh,1);
norm_maxh1=zeros(number_mesh,1);
norm_h1=zeros(number_mesh,1);

%% Solve discrite solution and refine mesh
for inumber_mesh=1:number_mesh
    h=(bx-ax)/N;
    number_mesh_point(inumber_mesh)=N;

    %% Create mesh point
    x=zeros(N+1,1);
    for i_iter=1:N+1
        x(i_iter)=(i_iter-1)*h;
    end
    y=zeros(N+1,1);
    for i_iter=1:N+1
        y(i_iter)=(i_iter-1)*h;
    end

    %% Create matrix A
    A=sparse((N-1)*(N-1),(N-1)*(N-1));
    B1=sparse(N-1,N-1);
    B2=sparse(N-1,N-1);
    I_h=sparse(N-1,N-1);
    for i=1:N-1
        I_h(i,i)=1;
    end
    for i=1:N-1
        if(i==1)
            B1(i,i)=2;
            B1(i,i+1)=-1;
        else
            if(i==N-1)
                B1(i,i)=2;
                B1(i,i-1)=-1;
            else
                B1(i,i)=4;
                B1(i,i-1)=-1;
                B1(i,i+1)=-1;
            end
        end
    end
    for i=1:N-1
        if(i==1)
            B2(i,i)=3;
            B2(i,i+1)=-1;
        else
            if(i==N-1)
                B2(i,i)=3;
```

```
            B2(i,i-1)=-1;
        else
            B2(i,i)=4;
            B2(i,i-1)=-1;
            B2(i,i+1)=-1;
        end
    end
end
for i=1:N-1
    if(i==1)
        A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B1;
        A((i-1)*(N-1)+1:i*(N-1),i*(N-1)+1:(i+1)*(N-1))=-I_h;
    else
        if(i==N-1)
            A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B1;
            A((i-1)*(N-1)+1:i*(N-1),(i-2)*(N-1)+1:(i-1)*(N-1))=-I_h;
        else
            A((i-1)*(N-1)+1:i*(N-1),(i-1)*(N-1)+1:i*(N-1))=B2;
            A((i-1)*(N-1)+1:i*(N-1),i*(N-1)+1:(i+1)*(N-1))=-I_h;
            A((i-1)*(N-1)+1:i*(N-1),(i-2)*(N-1)+1:(i-1)*(N-1))=-I_h;
        end
    end
end
A=(1/h^2)*A;

%% Create vector F
F=zeros((N-1)*(N-1),1);
for i1=1:N-1
    for i2=1:N-1
        F((i1-1)*(N-1)+i2)=functionf2D(i1*h,i2*h,cases);
    end
end

%% Solve discrete solution
u=A\F;


%% Get exact solution
u_exact=zeros((N+1),(N+1));
for i1=1:N+1
    for i2=1:N+1
        u_exact(i1,i2)=exact_solution2D(x(i1),y(i2),cases);
    end
end

%% Create discrete solution with boundary
u_dis=u_exact;
for i1=1:N-1
    for i2=1:N-1
        u_dis(i1+1,i2+1)=u((i1-1)*(N-1)+i2);
```

```
        end
    end

    %% Reshape u_dis and u_exact
    udis=reshape(u_dis,size(u_dis,1)*size(u_dis,2),1);
    uexact=reshape(u_exact,size(u_exact,1)*size(u_exact,2),1);

    %% Calculate the error on L^infinity
    norm_max(inumber_mesh)=0.0;
    for i=1:length(udis)
        if (abs(u_dis(i)-u_exact(i)) > norm_max(inumber_mesh))
            norm_max(inumber_mesh)=abs(u_dis(i)-u_exact(i));
        end
    end
    fprintf('error on L^infinity: %df\n',norm_max(inumber_mesh));

    %%  Calculate the error on L^2
    norm_l2(inumber_mesh)=0;
    for i=1:length(udis)
        norm_l2(inumber_mesh)=norm_l2(inumber_mesh)+ ...
        (u_dis(i)-u_exact(i))^2*h;
    end
    norm_l2(inumber_mesh)=(norm_l2(inumber_mesh))^(1/2);
    fprintf('error on L^2: %df\n',norm_l2(inumber_mesh));

    %% Figure exact and dicrete solutions
    figure
    surf(x,y,u_dis)
    xlabel('x')
    ylabel('y')
    %     axis equal
    str=sprintf('Discrete solutions with N=%d',N);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1])
    figure
    surf(x,y,u_exact)
    xlabel('x')
    ylabel('y')
    %     axis equal
    str=sprintf('Exact solutions with N=%d',N);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1]);

    %% Refine mesh
    N=4*N;
end

%% Figure for errors respect to number of mesh point
```

```
figure
plot(log(number_mesh_point), -log(norm_max),'blue', ...
     log(number_mesh_point), -log(norm_l2),...
    log(number_mesh_point), 2*log(number_mesh_point),'black');
xlabel('Log(MeshPoint)');ylabel('-Log(Error)');
title('Errors');
legend('norm_max','norm_l2','2x','Location','NorthEastOutside');
print('-r300','-djpeg');
```

7.4.2 Results

TEST.

$$u_{ex}(x,y) = \sin\left(x^3(x-1)^3y^3(y-1)^3\right) \tag{7.28}$$

$$f(x,y) = -\Delta\left(\sin\left(x^3(x-1)^3y^3(y-1)^3\right)\right) \tag{7.29}$$

Run the above scripts, MATLAB returns

```
Nx = 3, Ny = 4
error on L^infinity: 4.596710e-04f
error on L^2: 6.159250e-04f

Nx = 12, Ny = 16
error on L^infinity: 8.337595e-06f
error on L^2: 1.662004e-05f

Nx = 48, Ny = 64
error on L^infinity: 5.823727e-07f
error on L^2: 1.923342e-06f
```

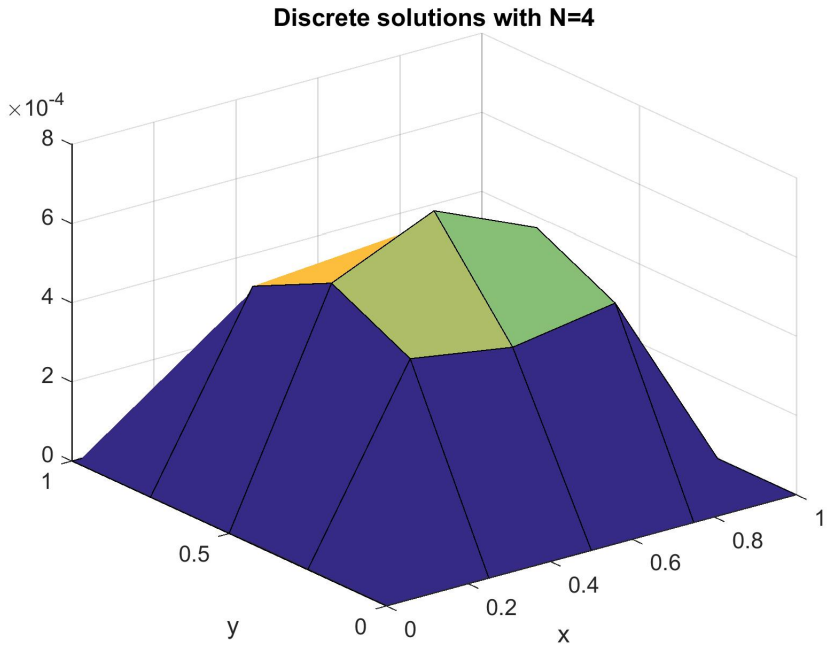

Figure 92: Discrete Solutions: 6.3, $N = 4$

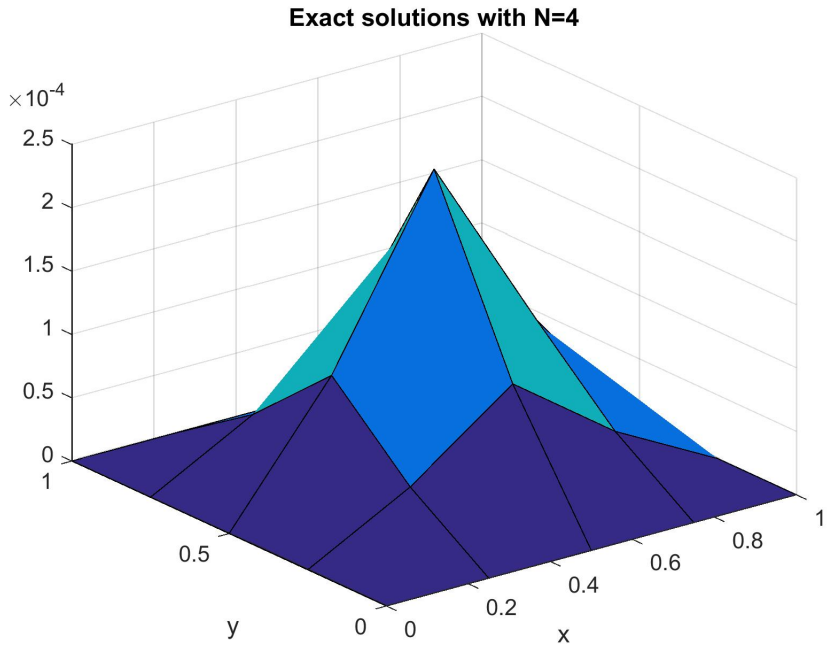

Figure 93: Exact Solutions: 6.3, $N = 4$

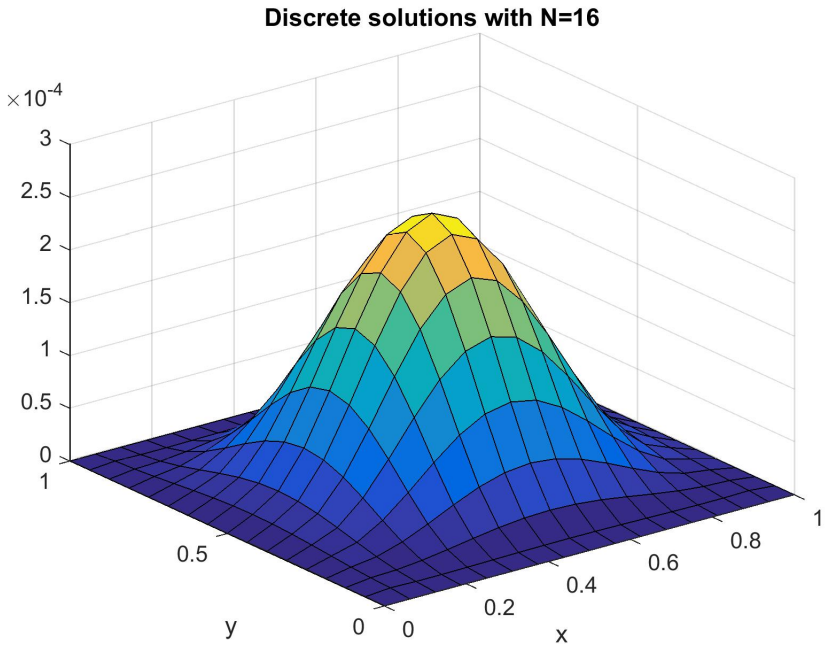

Figure 94: Discrete Solutions: 6.3, $N = 16$

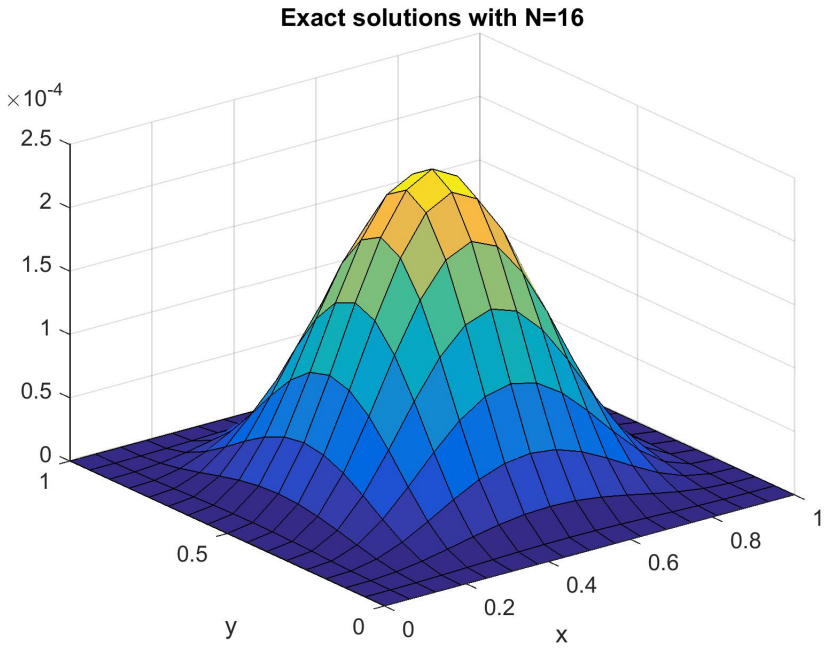

Figure 95: Exact Solutions: 6.3, $N = 16$

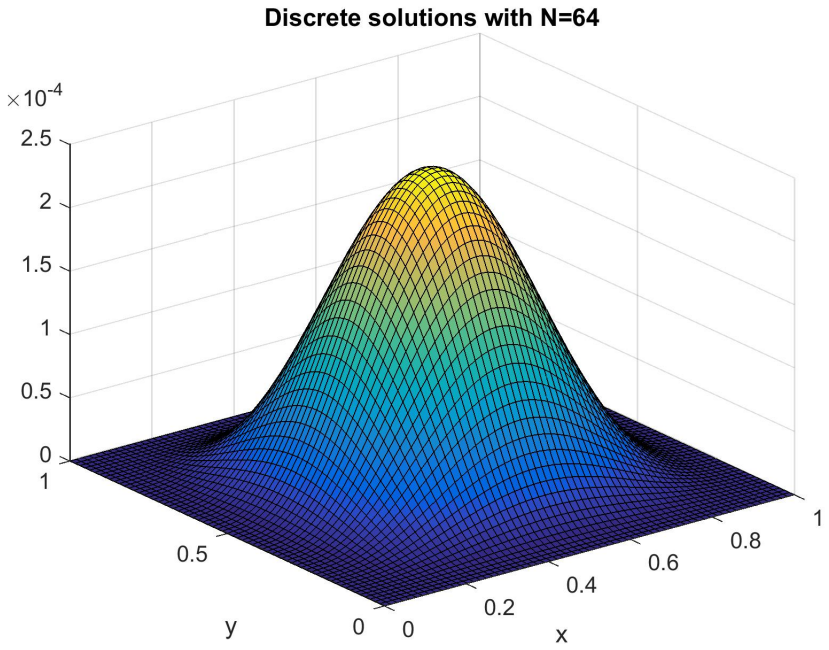

Figure 96: Discrete Solutions: 6.3, $N = 64$

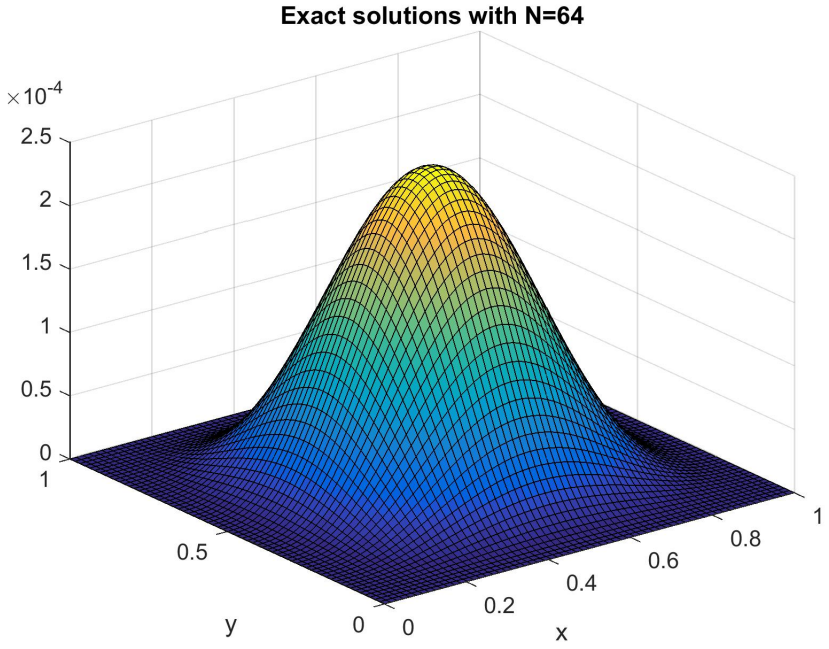

Figure 97: Exact Solutions: Problem 6.3 , $N = 64$

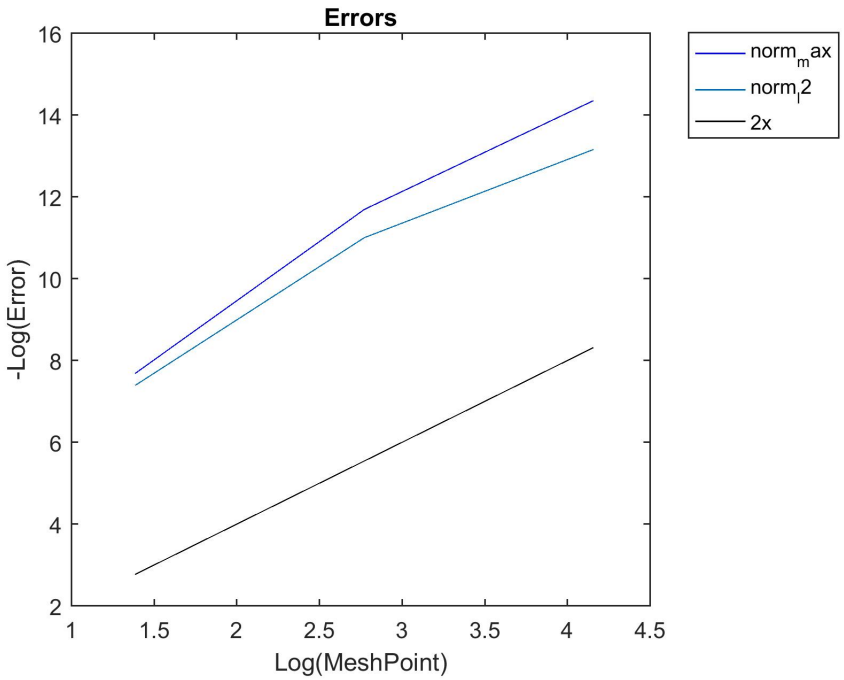

Figure 98: Error on a log-log scale: Problem 6.3

8 Matlab Implementation using 9-point Laplacian

All "main" codes use following test functions

exact_solution2D.m

8.1 Matlab Implementation for Problem 6.1.1

8.1.1 Matlab Scripts

```
function u_ex=exact_solution2D(x,y,k);
if (k==1)
    u_ex = x + y + 30*x*y*(x - 1)*(y - 1);
end

if(k==2)
    u_ex = x^2 + y^2 + 30*x^2*(1-x)^2*y^2*(1-y)^2;
end

if(k==3)
    u_ex = sin(x) + sin(y) + 30*x*(x-1)^2*y^3*(y-1)^4;
end

if(k==4)
    u_ex = x^2 + y^3 + 50*sin(x^2*(x-1)^2*y^2*(y-1)^2);
end

if(k==5)
    u_ex = sin(x) + cos(y) + sin(x^4*(x-1)^4*y^4*(y-1)^4);
end

if(k==6)
    u_ex=x^2 + y^2 + x*y*(x - 1)*(y - 1);
end
```

functionf2D.m

```
function f=functionf2D(x,y,k);

if (k == 1)
    f =   60*x*(x - 1) + 60*y*(y - 1);
end

if(k==2)
    f = (60*x^2*y^2*(x - 1)^2 + 60*x^2*y^2*(y - 1)^2 + ...
        60*x^2*(x - 1)^2*(y - 1)^2 + ...
        60*y^2*(x - 1)^2*(y - 1)^2 + ...
        120*x*y^2*(2*x - 2)*(y - 1)^2 + ...
        120*x^2*y*(2*y - 2)*(x - 1)^2 + 4);
end
```

```
if(k==3)
    f = (60*y^3*(2*x - 2)*(y - 1)^4 - sin(y) - sin(x) + ...
         60*x*y^3*(y - 1)^4 + 720*x*y^2*(x - 1)^2*(y - 1)^3 + ...
         360*x*y^3*(x - 1)^2*(y - 1)^2 + 180*x*y*(x - 1)^2*(y - 1)^4);
end

if(k==4)
    f = (6*y + 100*x^2*cos(x^2*y^2*(x - 1)^2*(y - 1)^2)*(x - 1)^2* ...
        (6*y^2 - 6*y + 1) + 100*y^2*cos(x^2*y^2*(x - 1)^2*(y - 1)^2)* ...
        (y - 1)^2*(6*x^2 - 6*x + 1) - ...
        200*x^2*y^4*sin(x^2*y^2*(x - 1)^2* ...
        (y - 1)^2)*(y - 1)^4*(2*x^2 - 3*x + 1)^2 - ...
        200*x^4*y^2*sin(x^2*y^2*(x - 1)^2*(y - 1)^2)* ...
        (x - 1)^4*(2*y^2 - 3*y + 1)^2 + 2);
end

if(k==5)
    f = cos(y) + sin(x) + 16*x^6*y^8*sin(x^4*y^4*(x - 1)^4* ...
        (y - 1)^4)*(2*x - 1)^2*(x - 1)^6*(y - 1)^8 + ...
        16*x^8*y^6*sin(x^4*y^4*(x - 1)^4*(y - 1)^4)*(2*y - 1)^2* ...
        (x - 1)^8*(y - 1)^6 - 4*x^2*y^4*cos(x^4*y^4*(x - 1)^4* ...
        (y - 1)^4)*(x - 1)^2*(y - 1)^4*(14*x^2 - 14*x + 3) - ...
        4*x^4*y^2*cos(x^4*y^4*(x - 1)^4*(y - 1)^4)*(x - 1)^4* ...
        (y - 1)^2*(14*y^2 - 14*y + 3);
end

if(k==6)
    f = 2*x*(x - 1) + 2*y*(y - 1) + 4;
end
```

g.m

```
function g = g(x,y,k);

if (k == 1)
    g = x + y;
end

if(k==2)
    g = x^2 + y^2;
end

if(k==3)
    g = sin(x) + sin(y);
end

if(k==4)
    g = x^2 + y^3;
end
```

```
if(k==5)
    g = sin(x) + cos(y);
end
```

main2D_9p.m

```
clear all
close all
clc
format long

tic

%% Initial informations
ax=0.0;
bx=1.0;
ay=0.0;
by=1.0;
cases=2;
Nx=10;% number of mesh points of first mesh
Ny=15;% number of mesh points of first mesh
N=max(Nx,Ny);
number_mesh=3;
number_mesh_point=zeros(number_mesh,1);
norm_max=zeros(number_mesh,1);
norm_l2=zeros(number_mesh,1);
norm_maxh1=zeros(number_mesh,1);
norm_h1=zeros(number_mesh,1);

%% Solve discrite solution and refine mesh
for inumber_mesh=1:number_mesh
    h=(bx-ax)/Nx;
    k=(by-ay)/Ny;
    number_mesh_point(inumber_mesh)=Nx;

    %% Create mesh point
    x=zeros(Nx+1,1);
    for i_iter=1:Nx+1
        x(i_iter)=(i_iter-1)*h;
    end
    y=zeros(Ny+1,1);
    for i_iter=1:Ny+1
        y(i_iter)=(i_iter-1)*k;
    end

    %% Create matrix A
    a1 = (-5/3)*(1/h^2+1/k^2);
    a2 = (1/12)*(1/h^2+1/k^2);
    a3 = 5/(6*h^2)-1/(6*k^2);
    a4 = 5/(6*k^2)-1/(6*h^2);
```

```
A=zeros((Nx-1)*(Ny-1),(Nx-1)*(Ny-1));
R=zeros(Nx-1,Nx-1);
S=sparse(Nx-1,Nx-1);
for i=1:Nx-1
    if(i==1)
        R(i,i)=(-5/3)*(1/h^2+1/k^2);
        R(i,i+1)=5/(6*h^2)-1/(6*k^2);
    else
        if(i==Nx-1)
            R(i,i)=(-5/3)*(1/h^2+1/k^2);
            R(i,i-1)=5/(6*h^2)-1/(6*k^2);
        else
            R(i,i)=(-5/3)*(1/h^2+1/k^2);
            R(i,i-1)=5/(6*h^2)-1/(6*k^2);
            R(i,i+1)=5/(6*h^2)-1/(6*k^2);
        end
    end
end

for i=1:Nx-1
    if(i==1)
        S(i,i)=5/(6*k^2)-1/(6*h^2);
        S(i,i+1)=(1/12)*(1/h^2+1/k^2);
    else
        if(i==Nx-1)
            S(i,i)=5/(6*k^2)-1/(6*h^2);
            S(i,i-1)=(1/12)*(1/h^2+1/k^2);
        else
            S(i,i)=5/(6*k^2)-1/(6*h^2);
            S(i,i-1)=(1/12)*(1/h^2+1/k^2);
            S(i,i+1)=(1/12)*(1/h^2+1/k^2);
        end
    end
end

for i=1:Ny-1
    if(i==1)
        A((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=R;
        A((i-1)*(Nx-1)+1:i*(Nx-1),i*(Nx-1)+1:(i+1)*(Nx-1))=S;
    else
        if(i==Ny-1)
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=R;
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-2)*(Nx-1)+1:(i-1)*(Nx-1))=S;
        else
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=R;
            A((i-1)*(Nx-1)+1:i*(Nx-1),i*(Nx-1)+1:(i+1)*(Nx-1))=S;
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-2)*(Nx-1)+1:(i-1)*(Nx-1))=S;
        end
    end
end
```

```
%% Create vector F
F=zeros((Nx-1)*(Ny-1),1);
for i2=1:Ny-1
    if (i2 == 1)
        for i1=1:Nx-1
            if (i1 == 1)
                F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                a2*g(0,0,cases)-a4*g(h,0,cases)-a2*g(2*h,0,cases)- ...
                a3*g(0,k,cases)-a2*g(0,2*k,cases);
            else
                if (i1 == Nx-1)
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    a2*g((Nx-2)*h,0,cases)-a4*g((Nx-1)*h,0,cases)- ...
                    a2*g(Nx*h,0,cases)-a3*g(Nx*h,k,cases)- ...
                    a2*g(Nx*h,2*k,cases);
                else
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    a2*g((i1-1)*h,0,cases)-a4*g(i1*h,0,cases)- ...
                    a2*g((i1+1)*h,0,cases);
                end
            end
        end
    else
        if (i2 == Ny-1)
            for i1=1:Nx-1
                if (i1 == 1)
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    a2*g(0,Ny*k,cases)-a4*g(h,Ny*k,cases)- ...
                    a2*g(2*h,Ny*k,cases)-a3*g(0,(Ny-1)*k,cases)- ...
                    a2*g(0,(Ny-2)*k,cases);
                else
                    if (i1 == Nx-1)
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                        a2*g((Nx-2)*h,Ny*k,cases)-a4*g((Nx-1)*h,Ny*k,cases)- ...
                        a2*g(Nx*h,Ny*k,cases)-a3*g(Nx*h,(Ny-1)*k,cases)- ...
                        a2*g(Nx*h,(Ny-2)*k,cases);
                    else
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                        a2*g((i1-1)*h,Ny*k,cases)-a4*g(i1*h,Ny*k,cases)- ...
                        a2*g((i1+1)*h,Ny*k,cases);
                    end
                end
            end
        else
            for i1=1:Nx-1
                if (i1 == 1)
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    a2*g(0,(i2-1)*k,cases)-a3*g(0,i2*k,cases)- ...
```

```
                    a2*g(0,(i2+1)*k,cases);
                else
                    if (i1 == Nx-1)
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                        a2*g(Nx*h,(i2-1)*k,cases)-a3*g(Nx*h,i2*k,cases)- ...
                        a2*g(Nx*h,(i2+1)*k,cases);
                    else
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases);
                    end
                end
            end
        end
    end
end

%% Solve discrete solution
u=A\F;

%% Get exact solution
u_exact=zeros((Nx+1),(Ny+1));
for i2=1:Ny+1
    for i1=1:Nx+1
        u_exact(i1,i2)=exact_solution2D(x(i1),y(i2),cases);
    end
end

%% Create discrete solution with boundary
u_dis=u_exact;
for i2=1:Ny-1
    for i1=1:Nx-1
        u_dis(i1+1,i2+1)=u((i2-1)*(Nx-1)+i1);
    end
end

%% Reshape u_dis and u_exact
udis=reshape(u_dis,size(u_dis,1)*size(u_dis,2),1);
uexact=reshape(u_exact,size(u_exact,1)*size(u_exact,2),1);

%% Calculate the error on L^infinity
norm_max(inumber_mesh)=0.0;
for i=1:length(udis)
    if (abs(u_dis(i)-u_exact(i)) > norm_max(inumber_mesh))
        norm_max(inumber_mesh)=abs(u_dis(i)-u_exact(i));
    end
end
fprintf('error on L^infinity: %df\n',norm_max(inumber_mesh));

%%  Calculate the error on L^2
norm_l2(inumber_mesh)=0;
for i=1:length(udis)
```

```
        norm_l2(inumber_mesh)=norm_l2(inumber_mesh)+ ...
        (u_dis(i)-u_exact(i))^2*h;
    end
    norm_l2(inumber_mesh)=(norm_l2(inumber_mesh))^(1/2);
    fprintf('error on L^2: %df\n',norm_l2(inumber_mesh));

    %% Figure exact and dicrete solutions
    figure
    surf(x,y,u_dis')
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Discrete solutions with N_x=%d, N_y=%d',Nx,Ny);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1])
    figure
    surf(x,y,u_exact')
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Exact solutions with N_x=%d, N_y=%d',Nx,Ny);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1]);

    %% Refine mesh
    Nx=2*Nx;
    Ny=2*Ny;
end

%% Figure for errors respect to number of mesh point
figure
plot(log(number_mesh_point), -log(norm_max),'blue', ...
     log(number_mesh_point), -log(norm_l2),...
     log(number_mesh_point), 4*log(number_mesh_point),'black');
xlabel('Log(MeshPoint)');ylabel('-Log(Error)');
title('Errors');
legend('norm_max','norm_l2','4x','Location','NorthEastOutside');
print('-r300','-djpeg');
```

8.1.2 Results

TEST 1.

$$g = x + y \tag{8.1}$$
$$u_{ex} = x + y + 30xy\,(x-1)\,(y-1) \tag{8.2}$$
$$f = -60x\,(x-1) - 60y\,(y-1) \tag{8.3}$$

Run these scripts for `cases = 1`, MATLAB returns

```
N = 4
error on L^infinity: 4.440892e-16f
error on L^2: 4.440892e-16f

N = 8
error on L^infinity: 1.776357e-15f
error on L^2: 2.645093e-15f

N = 16
error on L^infinity: 3.552714e-15f
error on L^2: 3.869614e-15f

N = 32
error on L^infinity: 2.264855e-14f
error on L^2: 5.587052e-14f

N = 64
error on L^infinity: 1.145750e-13f
error on L^2: 4.679732e-13f
```

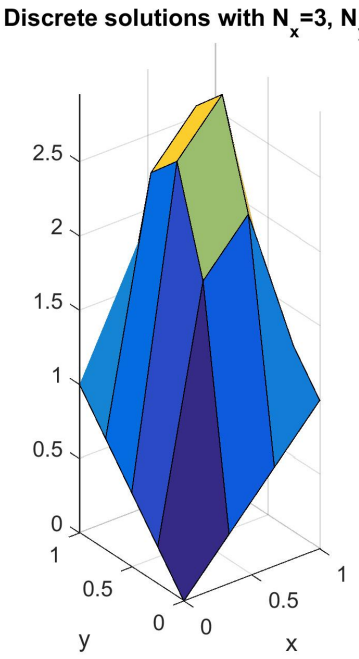

Figure 99: Discrete Solutions: Problem 6.1.1, Test 1, $Nx = 3, Ny = 4$.

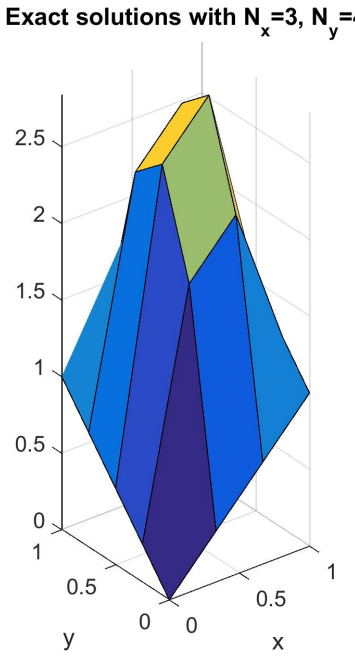

Figure 100: Exact Solutions: Problem 6.1.1, Test 1, $Nx = 3, Ny = 4$.

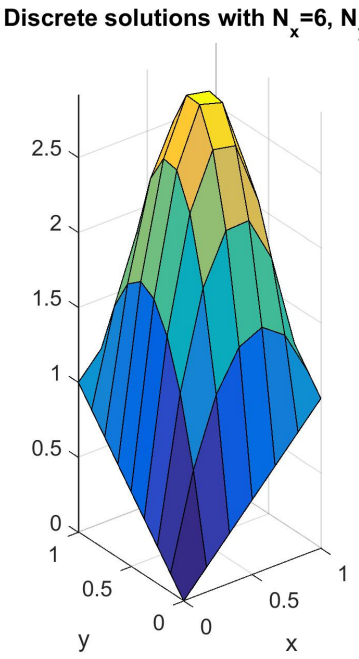

Figure 101: Discrete Solutions: Problem 6.1.1, Test 1, $Nx = 6, Ny = 8$.

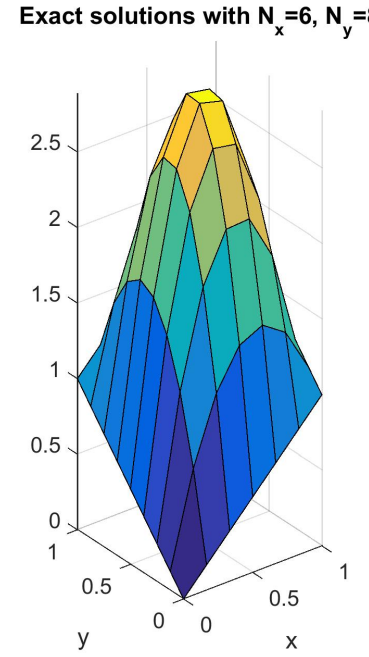

Figure 102: Exact Solutions: Problem 6.1.1, Test 1, $Nx = 6, Ny = 8$.

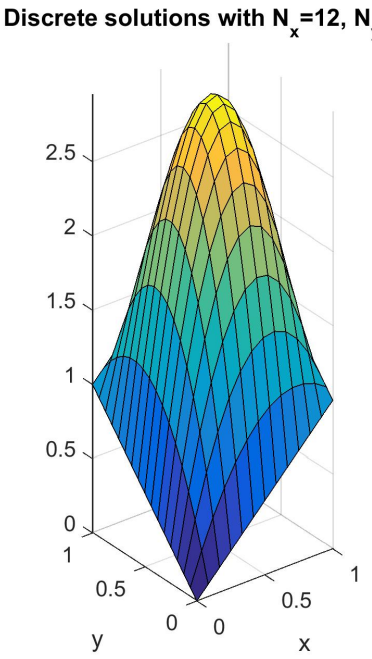

Figure 103: Discrete Solutions: Problem 6.1.1, Test 1, $Nx = 12, Ny = 16$.

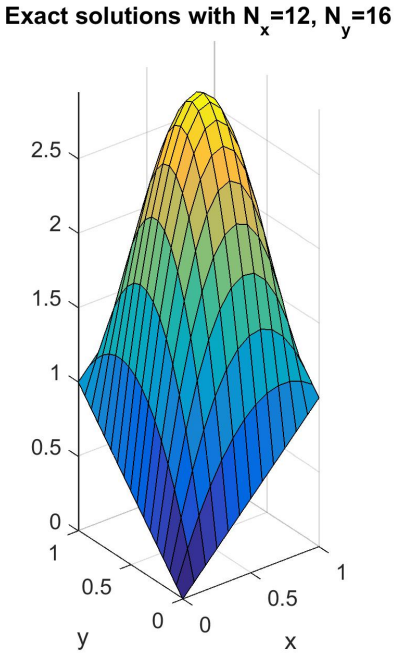

Figure 104: Exact Solutions: Problem 6.1.1, Test 1, $Nx = 12, Ny = 16$.

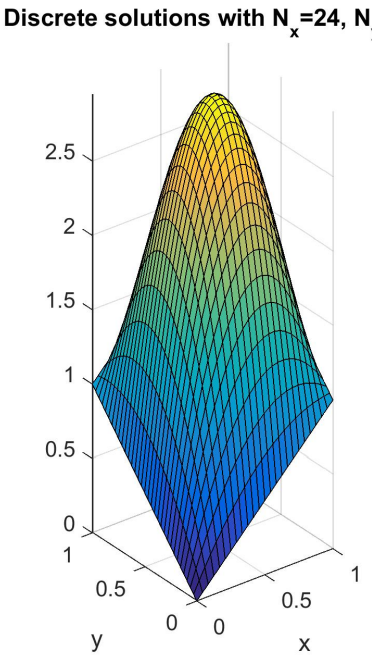

Figure 105: Discrete Solutions: Problem 6.1.1, Test 1, $Nx = 24, Ny = 32$.

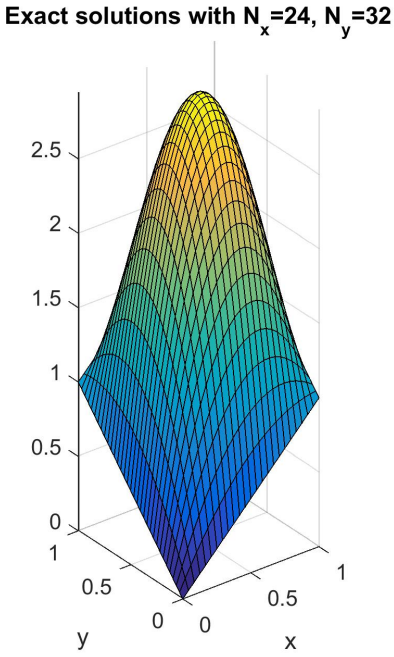

Figure 106: Exact Solutions: Problem 6.1.1, Test 1, $Nx = 24, Ny = 32$.

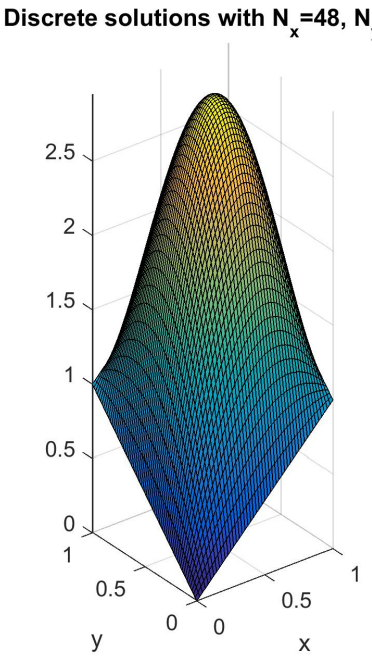

Figure 107: Discrete Solutions: Problem 6.1.1, Test 1, $Nx = 48, Ny = 64$.

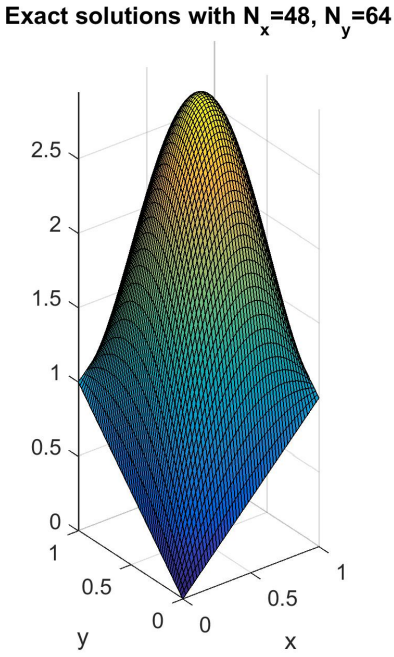

Figure 108: Exact Solutions: Problem 6.1.1, Test 1, $Nx = 48, Ny = 64$.

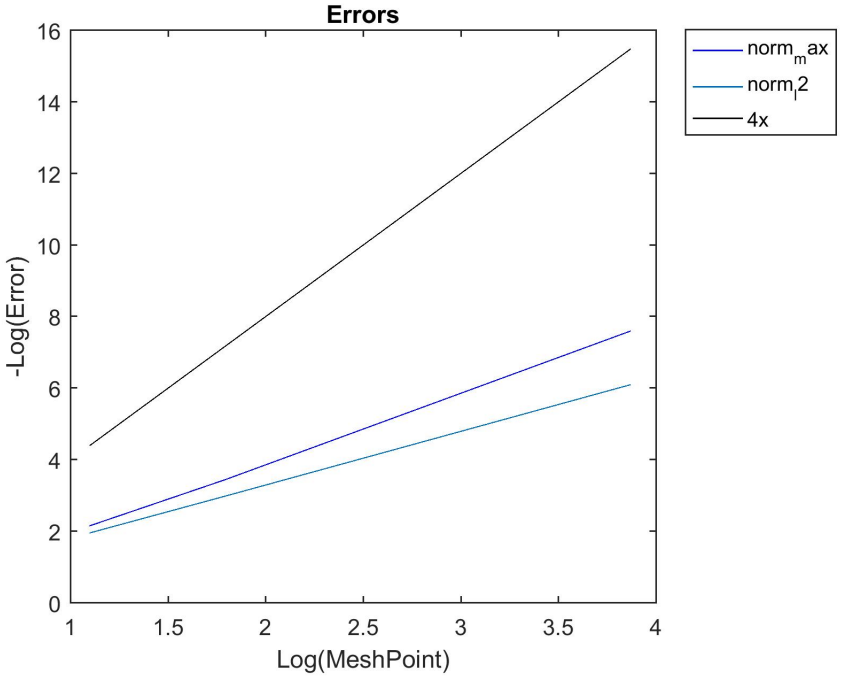

Figure 109: Error on a log-log scale: Problem 6.1.1, Test 1.

TEST 2.

$$g = x^2 + y^2 \tag{8.4}$$

$$u_{ex} = x^2 + y^2 + 30x^2y^2(x-1)^2(y-1)^2 \tag{8.5}$$

$$f = -\Delta\left(x^2 + y^2 + 30x^2y^2(x-1)^2(y-1)^2\right) \tag{8.6}$$

Run the above scripts for `cases = 2`, MATLAB returns

```
N = 4
error on L^infinity: 2.655029e-02f
error on L^2: 2.710100e-02f

N = 8
error on L^infinity: 6.543538e-03f
error on L^2: 9.532194e-03f

N = 16
error on L^infinity: 1.631785e-03f
error on L^2: 3.369234e-03f

N = 32
error on L^infinity: 4.077211e-04f
error on L^2: 1.191197e-03f

N = 64
error on L^infinity: 1.019167e-04f
error on L^2: 4.211525e-04f
```

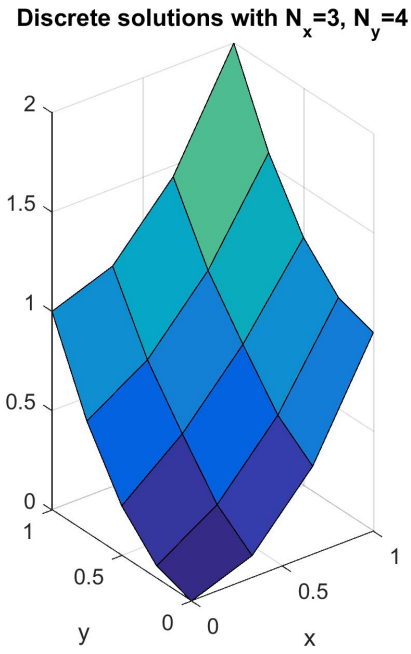

Figure 110: Discrete Solutions: Problem 6.1.1, Test 2, $Nx = 3, Ny = 4$.

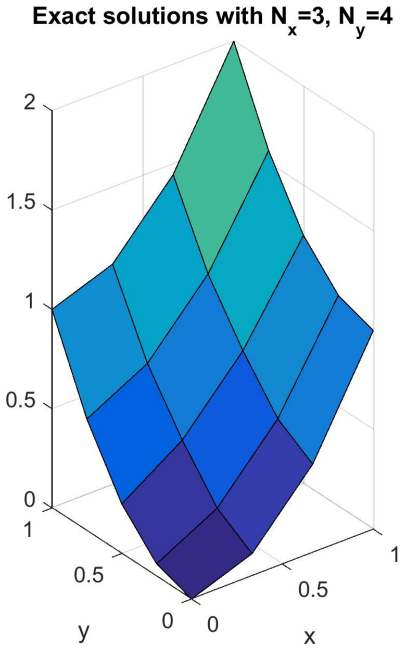

Figure 111: Exact Solutions: Problem 6.1.1, Test 2, $Nx = 3, Ny = 4$.

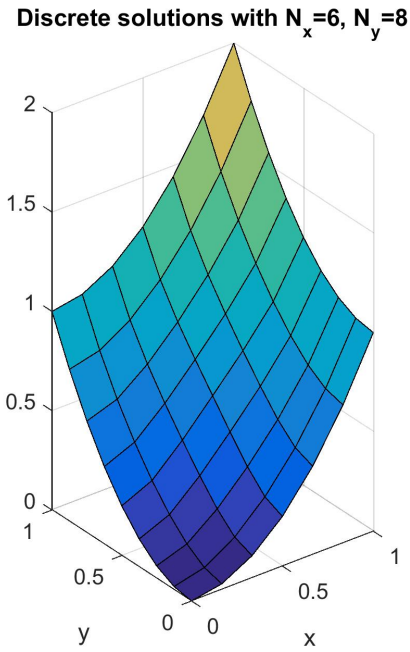

Figure 112: Discrete Solutions: Problem 6.1.1, Test 2, $Nx = 6, Ny = 8$.

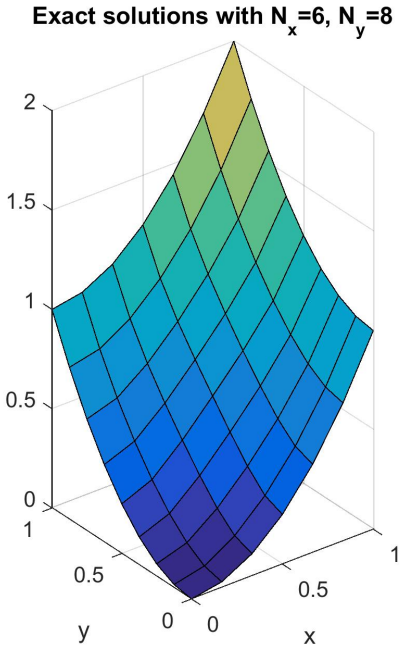

Figure 113: Exact Solutions: Problem 6.1.1, Test 2, $Nx = 6, Ny = 8$.

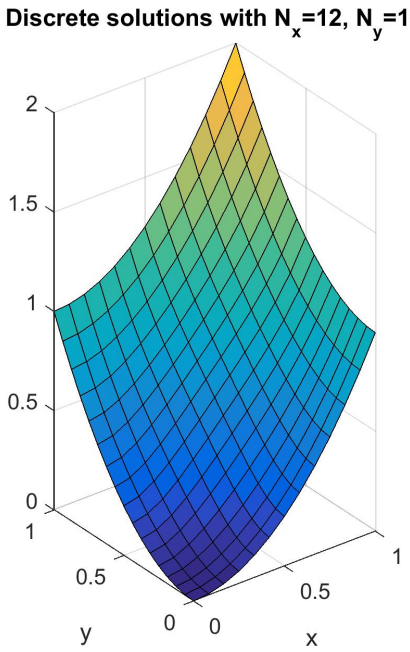

Figure 114: Discrete Solutions: Problem 6.1.1, Test 2, $Nx = 12, Ny = 16$.

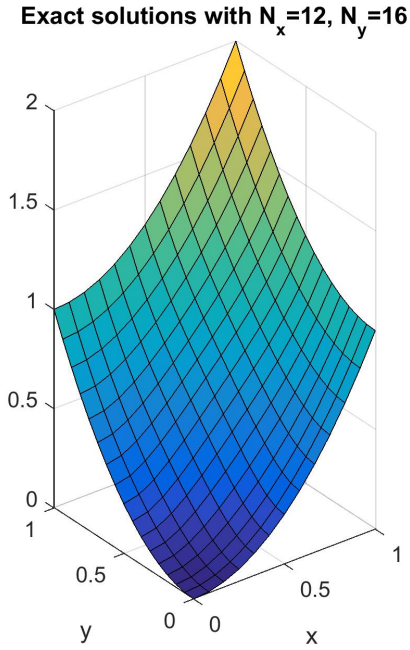

Figure 115: Exact Solutions: Problem 6.1.1, Test 2, $Nx = 12, Ny = 16$.

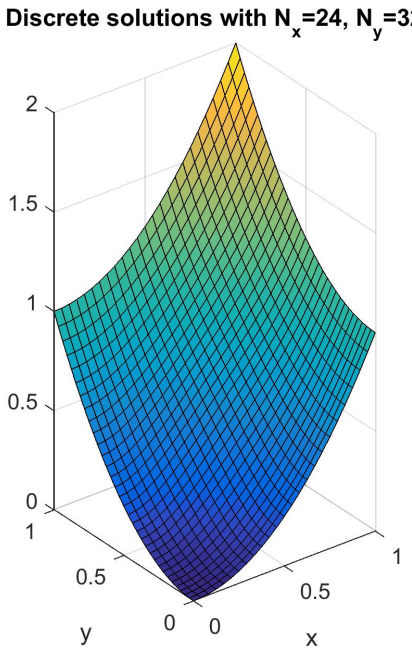

Figure 116: Discrete Solutions: Problem 6.1.1, Test 2, $Nx = 24, Ny = 32$.

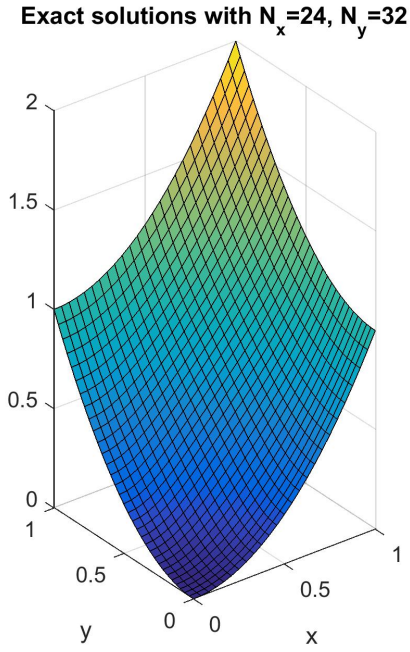

Figure 117: Exact Solutions: Problem 6.1.1, Test 2, $Nx = 24, Ny = 32$.

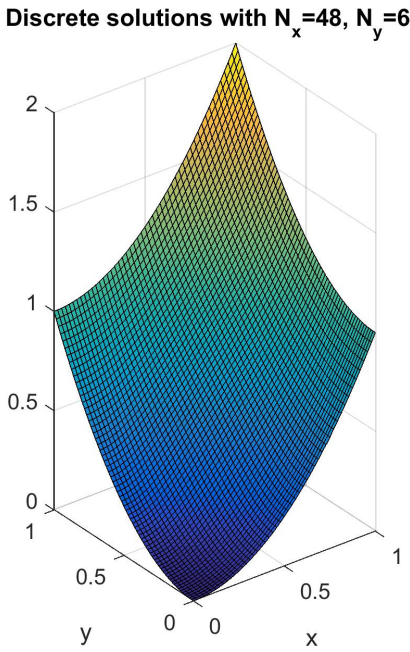

Figure 118: Discrete Solutions: Problem 6.1.1, Test 2, $Nx = 48, Ny = 64$.

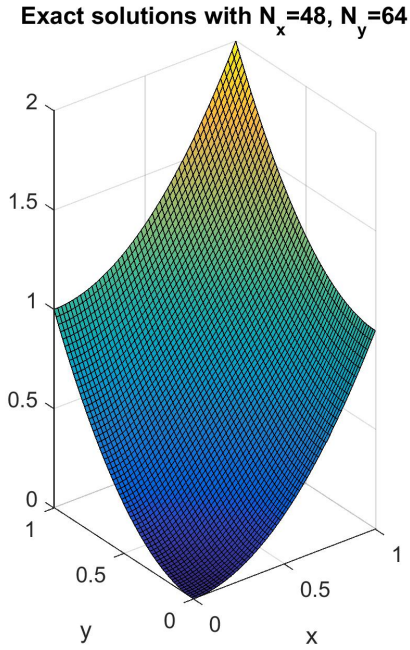

Figure 119: Exact Solutions: Problem 6.1.1, Test 2, $Nx = 48, Ny = 64$.

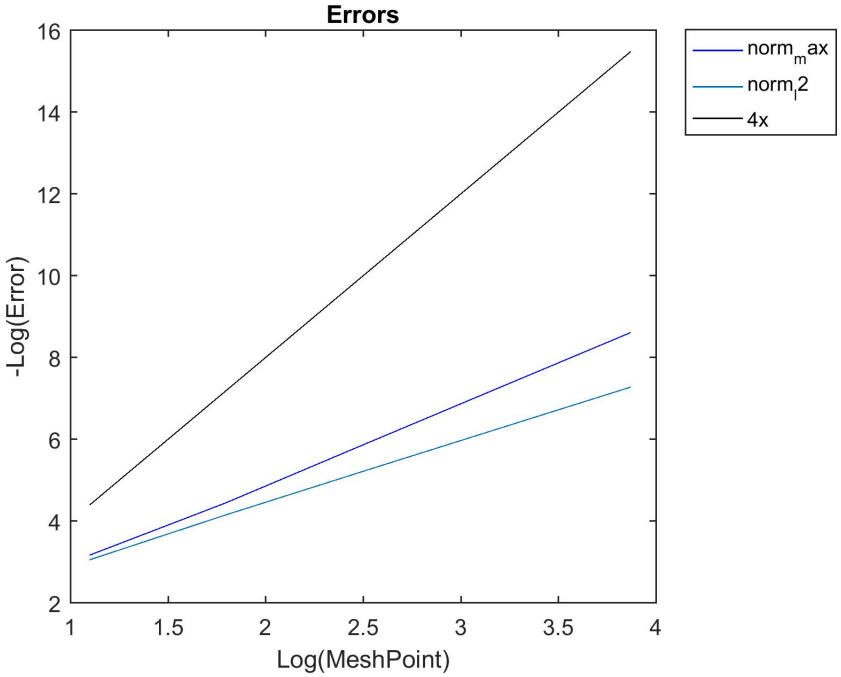

Figure 120: Error on a log-log scale: Problem 6.1.1, Test 2.

TEST 3.

$$g = \sin x + \sin y \tag{8.7}$$

$$u_{ex} = \sin x + \sin y + 30x(x-1)^2y^3(y-1)^4 \tag{8.8}$$

$$f = -\Delta\left(\sin x + \sin y + 30x(x-1)^2y^3(y-1)^4\right) \tag{8.9}$$

Run the above scripts for `cases = 3`, MATLAB returns

```
N = 4
error on L^infinity: 4.408222e-03f
error on L^2: 4.663096e-03f

N = 8
error on L^infinity: 1.239465e-03f
error on L^2: 1.377622e-03f

N = 16
error on L^infinity: 2.871268e-04f
error on L^2: 4.800258e-04f

N = 32
error on L^infinity: 7.868488e-05f
error on L^2: 1.688302e-04f

N = 64
error on L^infinity: 1.978660e-05f
error on L^2: 5.959994e-05f
```

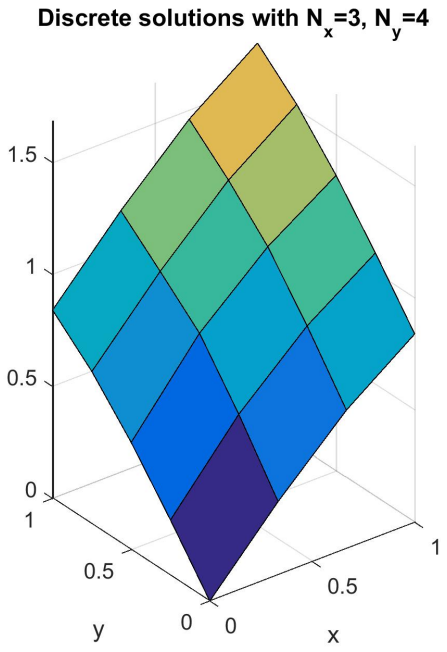

Figure 121: Discrete Solutions: Problem 6.1.1, Test 3, $Nx = 3, Ny = 4$.

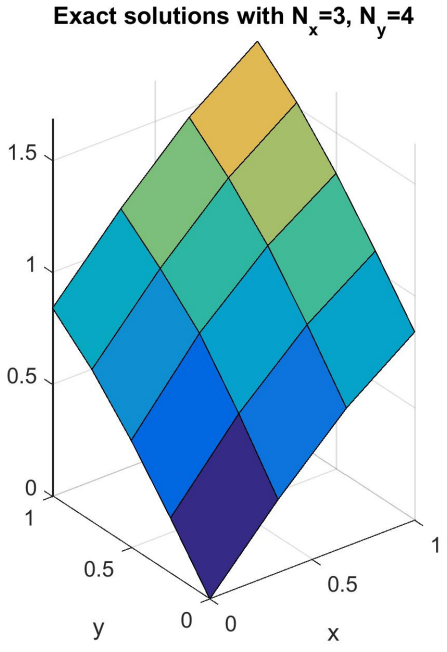

Figure 122: Exact Solutions: Problem 6.1.1, Test 3, $Nx = 3, Ny = 4$.

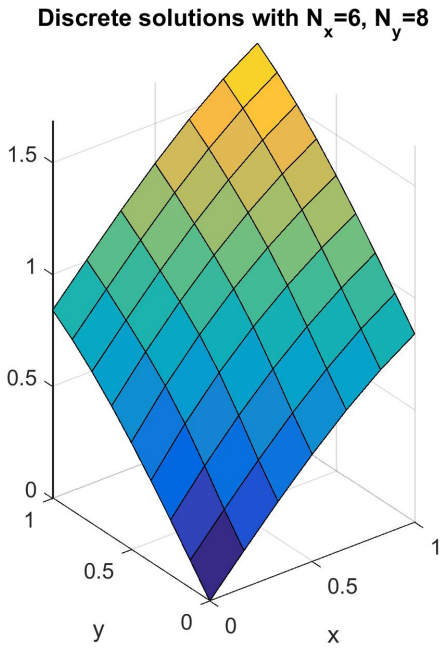

Figure 123: Discrete Solutions: Problem 6.1.1, Test 3, $Nx = 6, Ny = 8$.

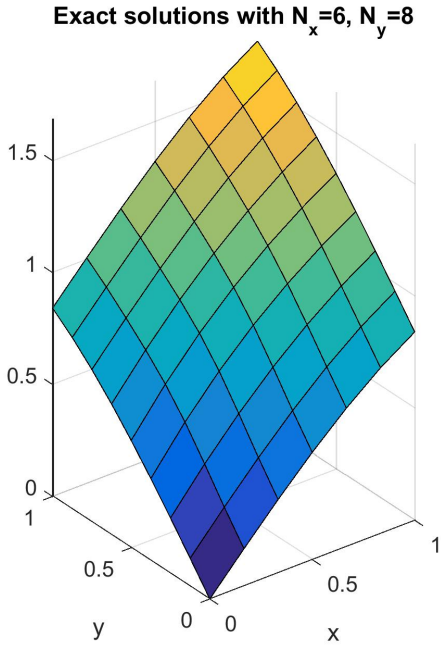

Figure 124: Exact Solutions: Problem 6.1.1, Test 3, $Nx = 6, Ny = 8$.

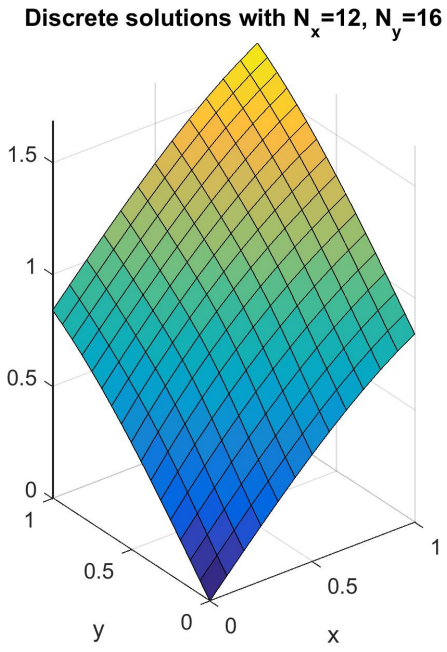

Figure 125: Discrete Solutions: Problem 6.1.1, Test 3, $Nx = 12, Ny = 16$.

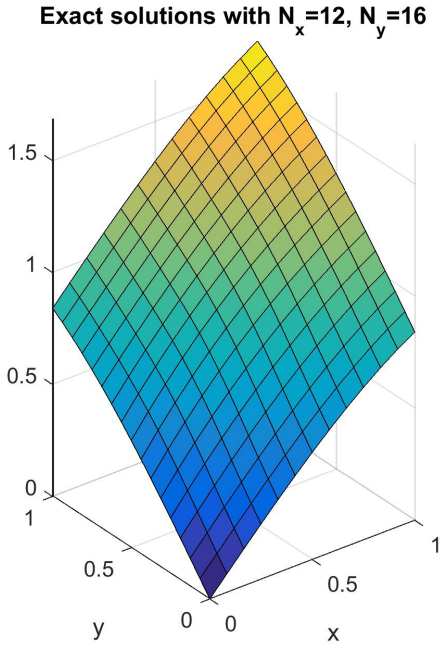

Figure 126: Exact Solutions: Problem 6.1.1, Test 3, $Nx = 12, Ny = 16$.

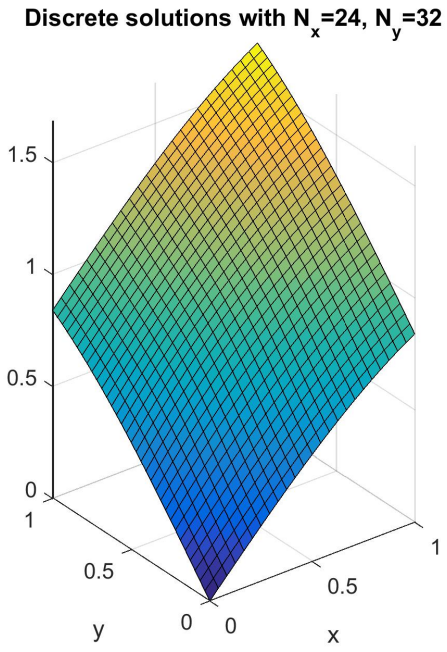

Figure 127: Discrete Solutions: Problem 6.1.1, Test 3, $Nx = 24, Ny = 32$.

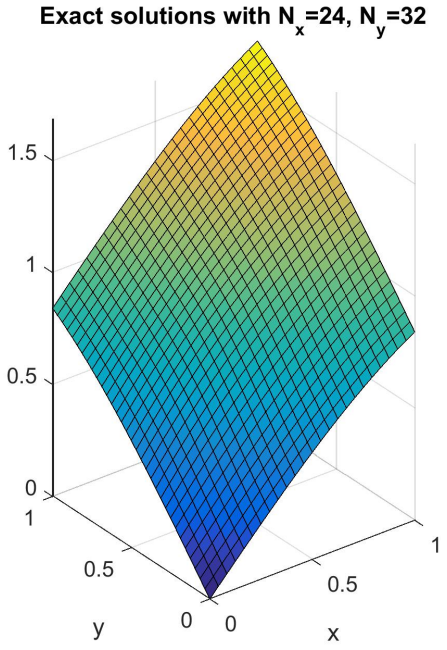

Figure 128: Exact Solutions: Problem 6.1.1, Test 3, $Nx = 24, Ny = 32$.

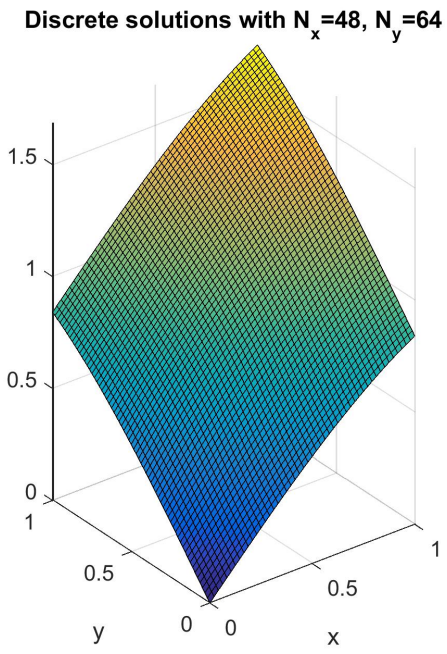

Figure 129: Discrete Solutions: Problem 6.1.1, Test 3, $Nx = 48, Ny = 64$.

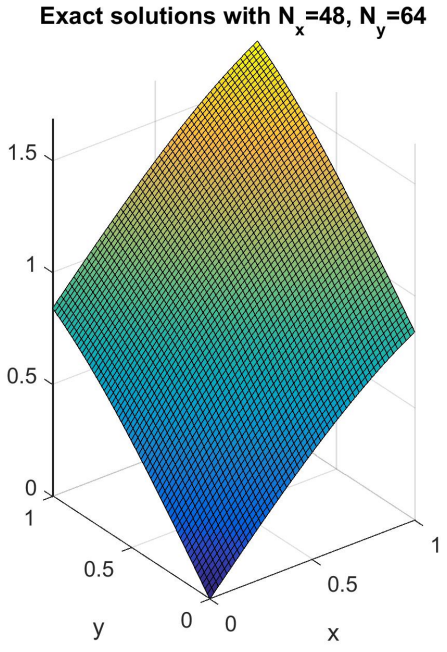

Figure 130: Exact Solutions: Problem 6.1.1, Test 3, $Nx = 48, Ny = 64$.

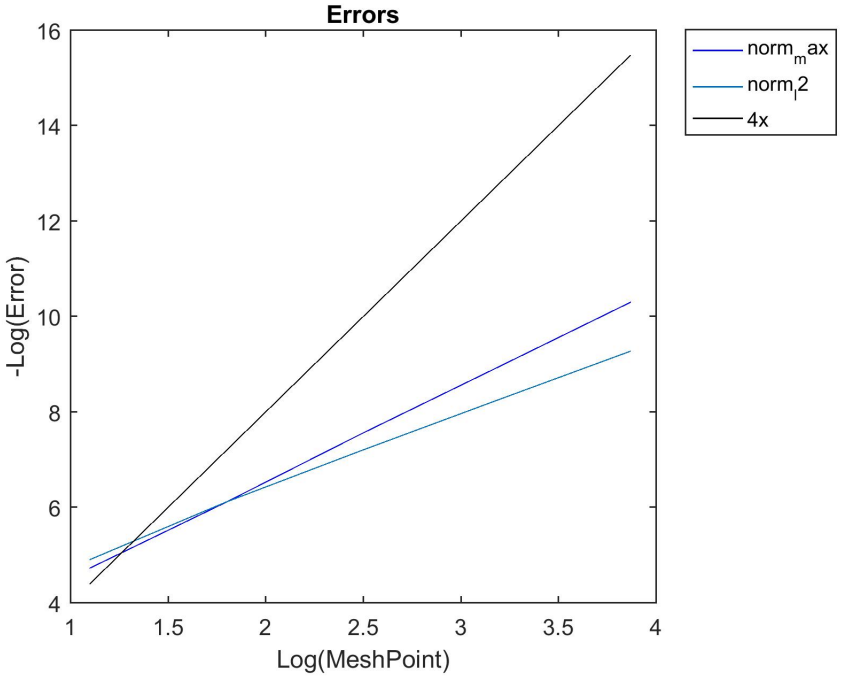

Figure 131: Error on a log-log scale: Problem 6.1.1, Test 3.

TEST 4.

$$g = x^2 + y^3 \tag{8.10}$$

$$u_{ex} = x^2 + y^3 + 50 \sin\left(x^2(x-1)^2 y^2 (y-1)^2\right) \tag{8.11}$$

$$f = -\Delta\left(x^2 + y^3 + 50 \sin\left(x^2(x-1)^2 y^2 (y-1)^2\right)\right) \tag{8.12}$$

Run the above scripts for `cases = 3`, MATLAB returns

```
N = 4
error on L^infinity: 4.425022e-02f
error on L^2: 4.516836e-02f

N = 8
error on L^infinity: 1.090585e-02f
error on L^2: 1.588699e-02f

N = 16
error on L^infinity: 2.719630e-03f
error on L^2: 5.615389e-03f

N = 32
error on L^infinity: 6.795323e-04f
error on L^2: 1.985327e-03f

N = 64
error on L^infinity: 1.698605e-04f
error on L^2: 7.019206e-04f
```

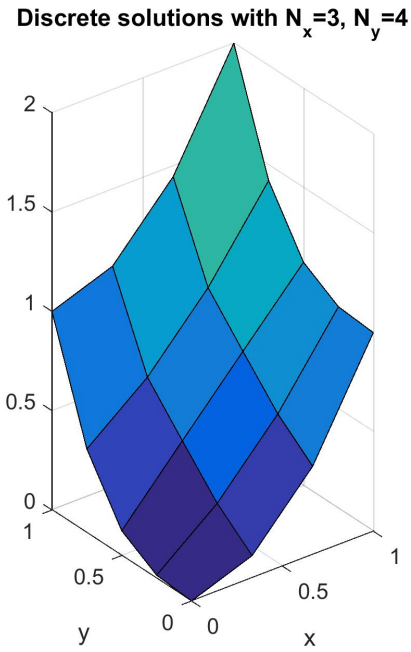

Figure 132: Discrete Solutions: Problem 6.1.1, Test 4, $Nx = 3, Ny = 4$.

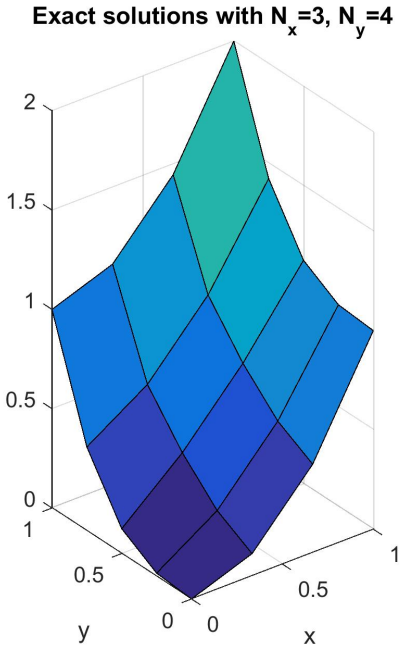

Figure 133: Exact Solutions: Problem 6.1.1, Test 4, $Nx = 3, Ny = 4$.

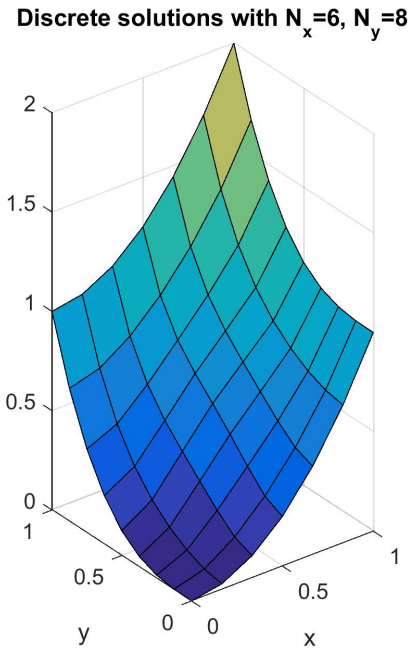

Figure 134: Discrete Solutions: Problem 6.1.1, Test 4, $Nx = 6, Ny = 8$.

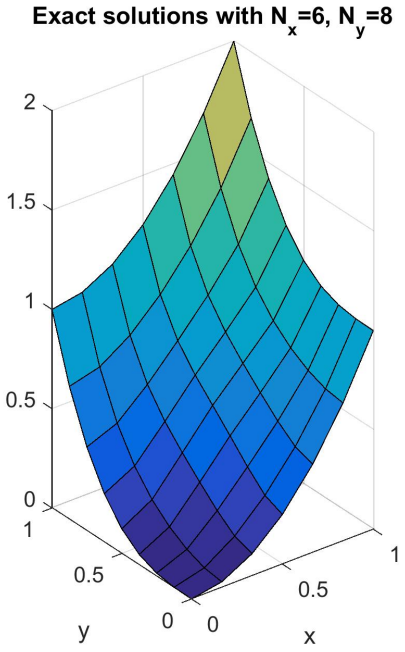

Figure 135: Exact Solutions: Problem 6.1.1, Test 4, $Nx = 6, Ny = 8$.

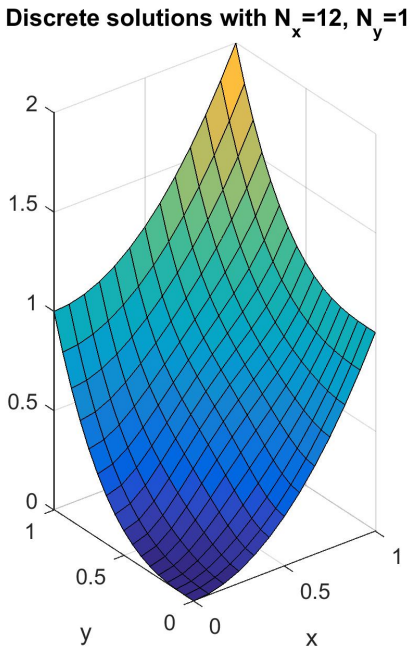

Figure 136: Discrete Solutions: Problem 6.1.1, Test 4, $Nx = 12, Ny = 16$.

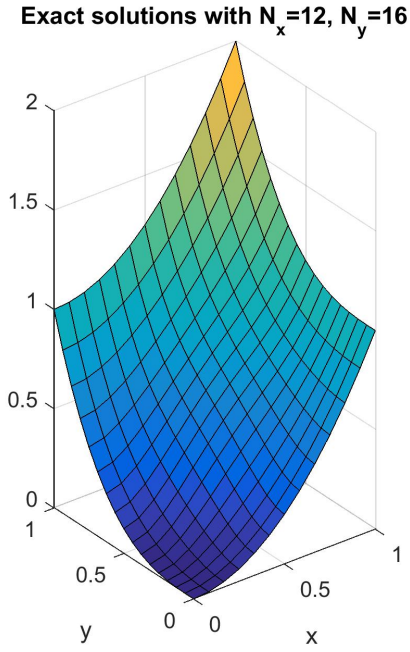

Figure 137: Exact Solutions: Problem 6.1.1, Test 4, $Nx = 12, Ny = 16$.

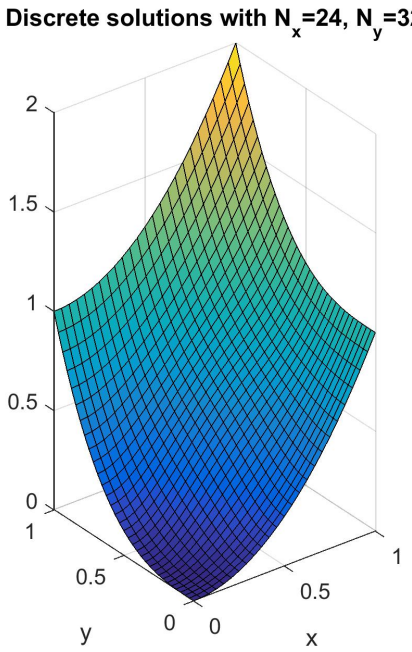

Figure 138: Discrete Solutions: Problem 6.1.1, Test 4, $Nx = 24, Ny = 32$.

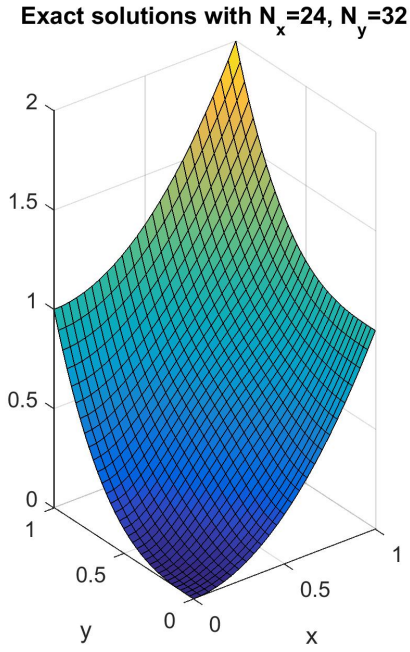

Figure 139: Exact Solutions: Problem 6.1.1, Test 4, $Nx = 24, Ny = 32$.

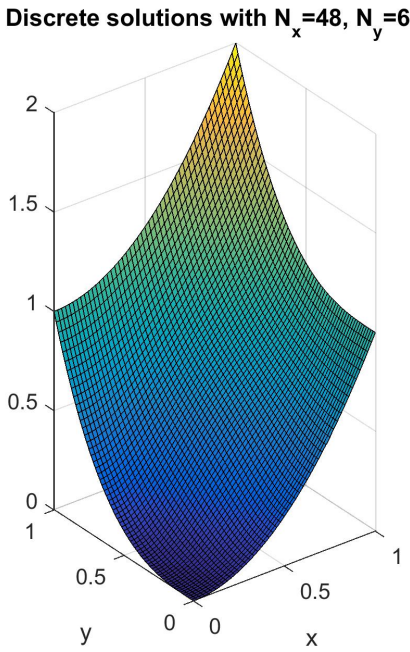

Figure 140: Discrete Solutions: Problem 6.1.1, Test 4, $Nx = 48, Ny = 64$.

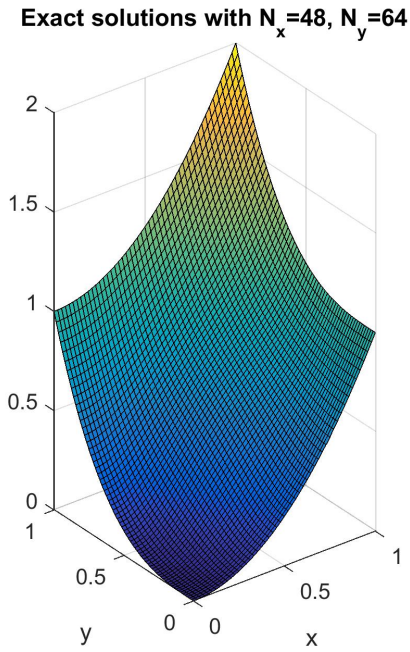

Figure 141: Exact Solutions: Problem 6.1.1, Test 4, $Nx = 48, Ny = 64$.

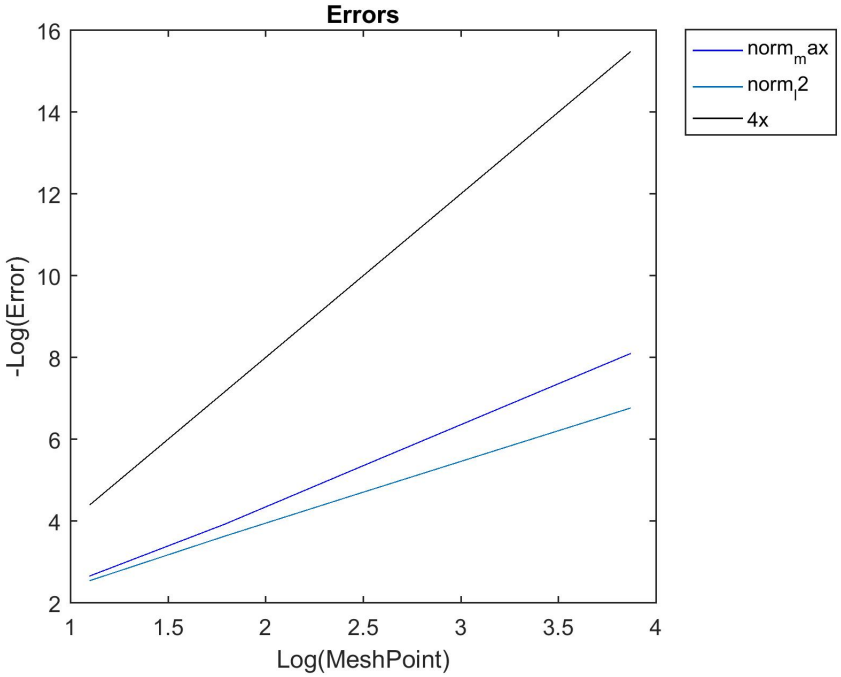

Figure 142: Error on a log-log scale: Problem 6.1.1, Test 4.

8.2 Matlab Implementation for Problem 6.2

8.2.1 Matlab Scripts

Remark 8.1. Following codes use second order differential approximation, thus it significantly reduces the method's convergence rate.

Readers can use fourth order differential approximation for better convergence rate.

gk.m

```
function f = gk(x,y);

f = x^2+y^2;
```

g4.m

```
function f = g4(y);

f = y*(y - 1) + 2;
```

main2D_9p_2.m

```
clear all
close all
clc
format long

tic

%% Initial informations
ax=0.0;
bx=1.0;
ay=0.0;
by=1.0;
cases=6;
Nx=3;% number of mesh points of first mesh
Ny=4;% number of mesh points of first mesh
N=max(Nx,Ny);
number_mesh=5;
number_mesh_point=zeros(number_mesh,1);
norm_max=zeros(number_mesh,1);
norm_l2=zeros(number_mesh,1);
norm_maxh1=zeros(number_mesh,1);
norm_h1=zeros(number_mesh,1);

%% Solve discrite solution and refine mesh
for inumber_mesh=1:number_mesh
    h=(bx-ax)/Nx;
    k=(by-ay)/Ny;
    number_mesh_point(inumber_mesh)=Nx;

    %% Create mesh point
```

```
x=zeros(Nx+1,1);
for i_iter=1:Nx+1
    x(i_iter)=(i_iter-1)*h;
end
y=zeros(Ny+1,1);
for i_iter=1:Ny+1
    y(i_iter)=(i_iter-1)*k;
end

%% Create matrix A
a1 = (-5/3)*(1/h^2+1/k^2);
a2 = (1/12)*(1/h^2+1/k^2);
a3 = 5/(6*h^2)-1/(6*k^2);
a4 = 5/(6*k^2)-1/(6*h^2);

A=zeros((Nx-1)*(Ny-1),(Nx-1)*(Ny-1));
R=zeros(Nx-1,Nx-1);
S=sparse(Nx-1,Nx-1);
for i=1:Nx-1
    if(i==1)
        R(i,i)=a1;
        R(i,i+1)=a3;
    else
        if(i==Nx-1)
            R(i,i)=a1+(4/3)*a3;
            R(i,i-1)=(2/3)*a3;
        else
            R(i,i)=a1;
            R(i,i-1)=a3;
            R(i,i+1)=a3;
        end
    end
end

for i=1:Nx-1
    if(i==1)
        S(i,i)=a4;
        S(i,i+1)=a2;
    else
        if(i==Nx-1)
            S(i,i)=a4+(4/3)*a2;
            S(i,i-1)=(2/3)*a2;
        else
            S(i,i)=a4;
            S(i,i-1)=a2;
            S(i,i+1)=a2;
        end
    end
end
```

```
for i=1:Ny-1
    if(i==1)
        A((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=R;
        A((i-1)*(Nx-1)+1:i*(Nx-1),i*(Nx-1)+1:(i+1)*(Nx-1))=S;
    else
        if(i==Ny-1)
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=R;
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-2)*(Nx-1)+1:(i-1)*(Nx-1))=S;
        else
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-1)*(Nx-1)+1:i*(Nx-1))=R;
            A((i-1)*(Nx-1)+1:i*(Nx-1),i*(Nx-1)+1:(i+1)*(Nx-1))=S;
            A((i-1)*(Nx-1)+1:i*(Nx-1),(i-2)*(Nx-1)+1:(i-1)*(Nx-1))=S;
        end
    end
end


%% Create vector F
%     F=zeros((Nx-1)*(Ny-1),1);
%     for i2=1:Ny-1
%         for i1=1:Nx-1
%             F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k);
%         end
%     end

%% Create vector F
F=zeros((Nx-1)*(Ny-1),1);
for i2=1:Ny-1
    if (i2 == 1)
        for i1=1:Nx-1
            if (i1 == 1)
                F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                a2*gk(0,0)-a4*gk(h,0)-a2*gk(2*h,0)- ...
                a3*gk(0,k)-a2*gk(0,2*k);
            else
                if (i1 == Nx-1)
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    (2/3)*a2*gk((Nx-2)*h,0)-(a4+(4/3)*a2)*gk((Nx-1)*h,0)- ...
                    a2*(2/3)*h*g4(0)-a3*(2/3)*h*g4(k)-a2*(2/3)*h*g4(2*k);
                else
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    a2*gk((i1-1)*h,0)-a4*gk(i1*h,0)-a2*gk((i1+1)*h,0);
                end
            end
        end
    else
        if (i2 == Ny-1)
            for i1=1:Nx-1
                if (i1 == 1)
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
```

```
                    a2*gk(0,Ny*k)-a4*gk(h,Ny*k)-a2*gk(2*h,Ny*k)- ...
                    a3*gk(0,(Ny-1)*k)-a2*gk(0,(Ny-2)*k);
                else
                    if (i1 == Nx-1)
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                        (2/3)*a2*gk((Nx-2)*h,Ny*k)- ...
                        (a4+(4/3)*a2)*gk((Nx-1)*h,Ny*k)- ...
                        a2*(2/3)*h*g4((Ny-2)*k)-a3*(2/3)*h*g4((Ny-1)*k)- ...
                        a2*(2/3)*h*g4(Ny*k);
                    else
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                        a2*gk((i1-1)*h,Ny*k)-a4*gk(i1*h,Ny*k)- ...
                        a2*gk((i1+1)*h,Ny*k);
                    end
                end
            end
        else
            for i1=1:Nx-1
                if (i1 == 1)
                    F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                    a2*gk(0,(i2-1)*k)-a3*gk(0,i2*k)- ...
                    a2*gk(0,(i2+1)*k);
                else
                    if (i1 == Nx-1)
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases)- ...
                        a2*(2/3)*h*g4((i2-1)*k)-a3*(2/3)*h*g4(i2*k)- ...
                        a2*(2/3)*h*g4((i2+1)*k);
                    else
                        F((i2-1)*(Nx-1)+i1)=functionf2D(i1*h,i2*k,cases);
                    end
                end
            end
        end
    end
end

%% Solve discrete solution
u=A\F;

%% Get exact solution
u_exact=zeros((Nx+1),(Ny+1));
for i2=1:Ny+1
    for i1=1:Nx+1
        u_exact(i1,i2)=exact_solution2D(x(i1),y(i2),cases);
    end
end

%% Create discrete solution with boundary
u_dis=u_exact;
for i2=1:Ny-1
```

```
        for i1=1:Nx-1
            u_dis(i1+1,i2+1)=u((i2-1)*(Nx-1)+i1);
        end
    end

    %% Reshape u_dis and u_exact
    udis=reshape(u_dis,size(u_dis,1)*size(u_dis,2),1);
    uexact=reshape(u_exact,size(u_exact,1)*size(u_exact,2),1);

    %% Calculate the error on L^infinity
    norm_max(inumber_mesh)=0.0;
    for i=1:length(udis)
        if (abs(u_dis(i)-u_exact(i)) > norm_max(inumber_mesh))
            norm_max(inumber_mesh)=abs(u_dis(i)-u_exact(i));
        end
    end
    fprintf('error on L^infinity: %df\n',norm_max(inumber_mesh));

    %%  Calculate the error on L^2
    norm_l2(inumber_mesh)=0;
    for i=1:length(udis)
        norm_l2(inumber_mesh)=norm_l2(inumber_mesh)+ ...
                              (u_dis(i)-u_exact(i))^2*h;
    end
    norm_l2(inumber_mesh)=(norm_l2(inumber_mesh))^(1/2);
    fprintf('error on L^2: %df\n',norm_l2(inumber_mesh));
    %% Figure exact and dicrete solutions
    figure
    surf(x,y,u_dis')
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Discrete solutions with N_x=%d, N_y=%d',Nx,Ny);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1])
    figure
    surf(x,y,u_exact')
    xlabel('x')
    ylabel('y')
    axis equal
    str=sprintf('Exact solutions with N_x=%d, N_y=%d',Nx,Ny);
    title(str);
    print('-r300','-djpeg');
    %    zlim([-1 1]);

    %% Refine mesh
    Nx=2*Nx;
    Ny=2*Ny;
end
```

```
%% Figure for errors respect to number of mesh point
figure
plot(log(number_mesh_point), -log(norm_max),'blue', ...
     log(number_mesh_point), -log(norm_l2),...
     log(number_mesh_point), 4*log(number_mesh_point),'black');
xlabel('log(MeshPoint)');ylabel('-log(Error)');
title('Errors');
legend('norm_max','norm_l2','4x','Location','NorthEastOutside');
print('-r300','-djpeg');
```

8.2.2 Results

TEST.

$$u_{ex}(x,y) = 30x(1-x)y(1-y) \tag{8.13}$$
$$f(x,y) = -60x(x-1) - 60y(y-1) \tag{8.14}$$
$$g_1(x) = x^2 \tag{8.15}$$
$$g_2(y) = y^2 \tag{8.16}$$
$$g_3(x) = x^2 + 1 \tag{8.17}$$
$$g_4(y) = y(y-1) + 2 \tag{8.18}$$

Run the above scripts, MATLAB returns

```
Nx = 3, Ny = 4
error on L^infinity: 4.951642e-02f
error on L^2: 4.133113e-02f

Nx = 12, Ny = 16
error on L^infinity: 1.657805e-02f
error on L^2: 2.239739e-02f

Nx = 48, Ny = 64
error on L^infinity: 4.420174e-03f
error on L^2: 1.126896e-02f
```

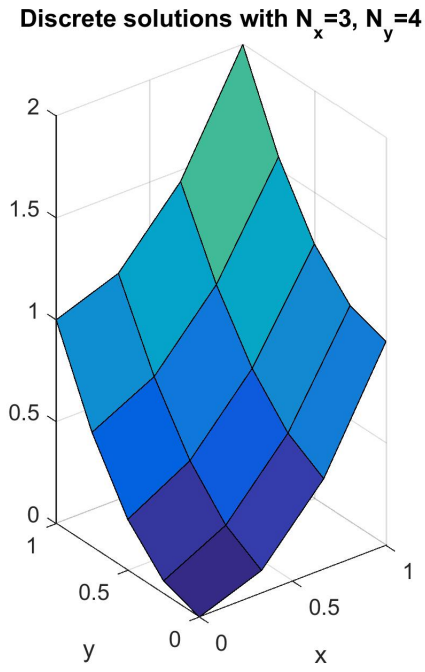

Figure 143: Discrete Solutions: 6.2, $Nx = 3, Ny = 4$

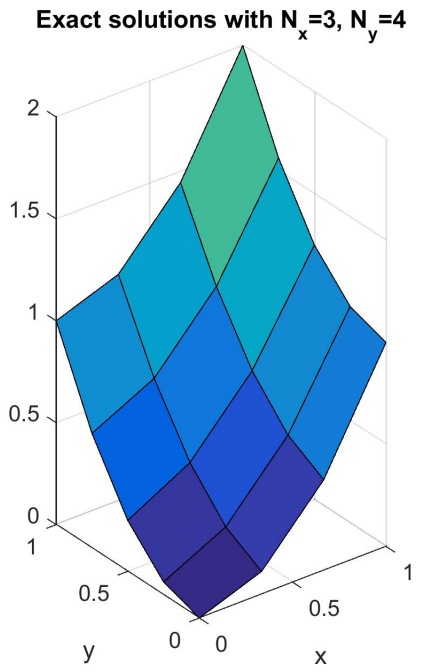

Figure 144: Exact Solutions: 6.2, $Nx = 3, Ny = 4$

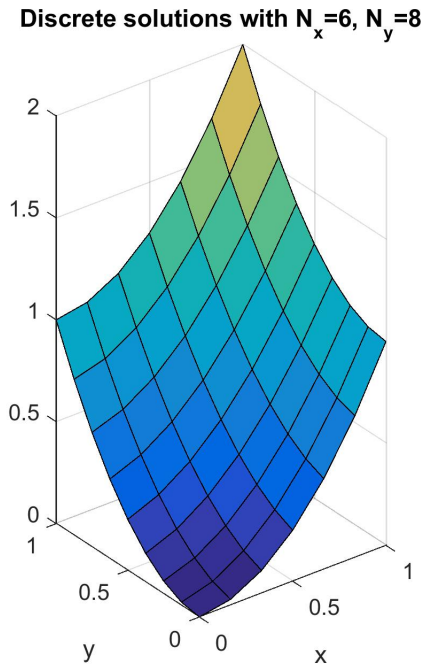

Figure 145: Discrete Solutions: 6.2, $Nx = 6, Ny = 8$

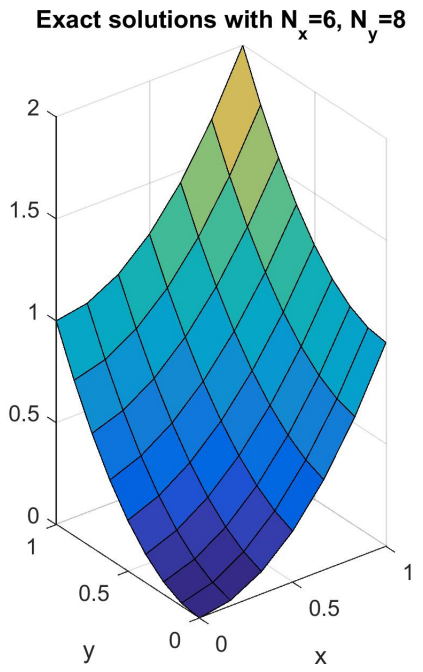

Figure 146: Exact Solutions: 6.2, $Nx = 6, Ny = 8$

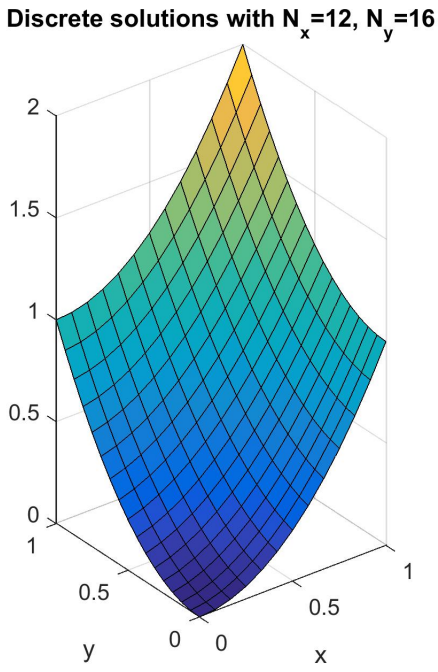

Figure 147: Discrete Solutions: 6.2, $Nx = 12, Ny = 16$

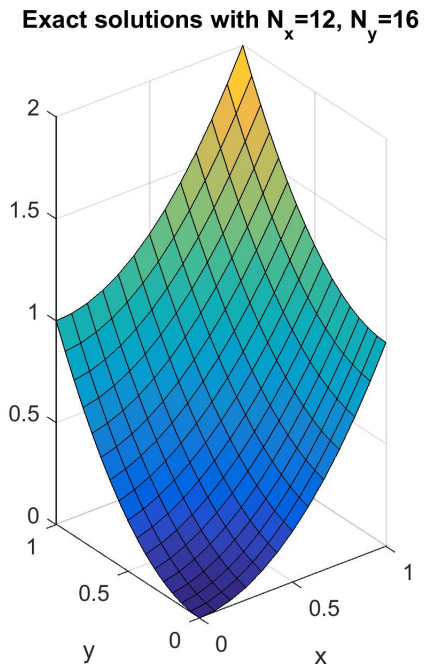

Figure 148: Exact Solutions: Problem 6.2 , $Nx = 12, Ny = 16$

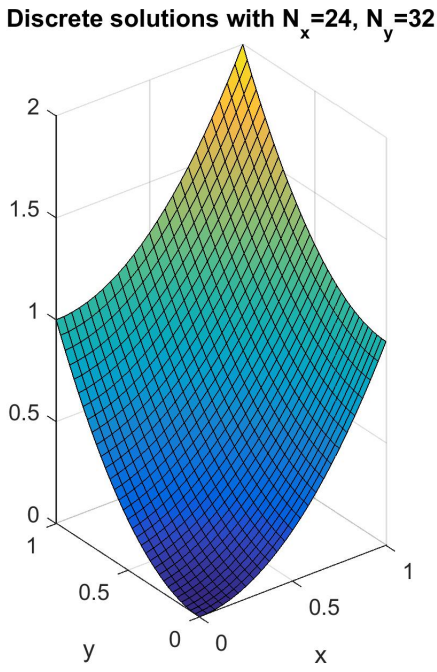

Figure 149: Exact Solutions: Problem 6.2 , $Nx = 24, Ny = 48$

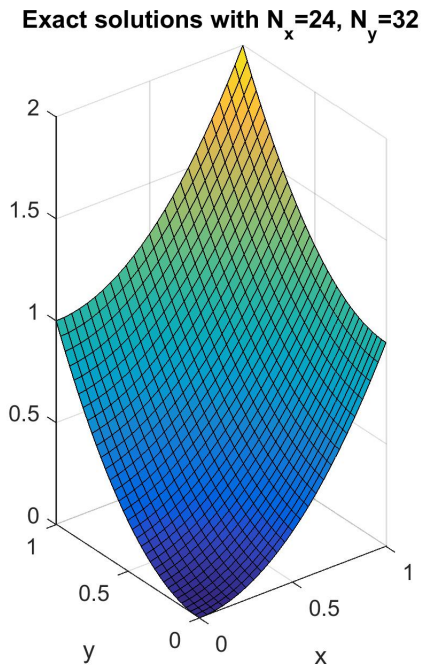

Figure 150: Exact Solutions: Problem 6.2 , $Nx = 24, Ny = 48$

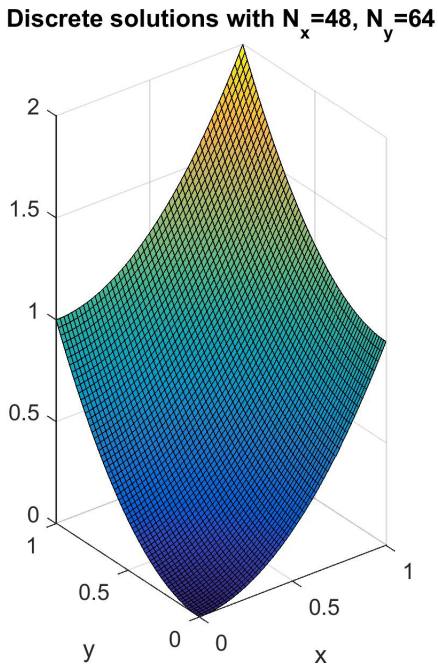

Figure 151: Exact Solutions: Problem 6.2 , $Nx = 48, Ny = 64$

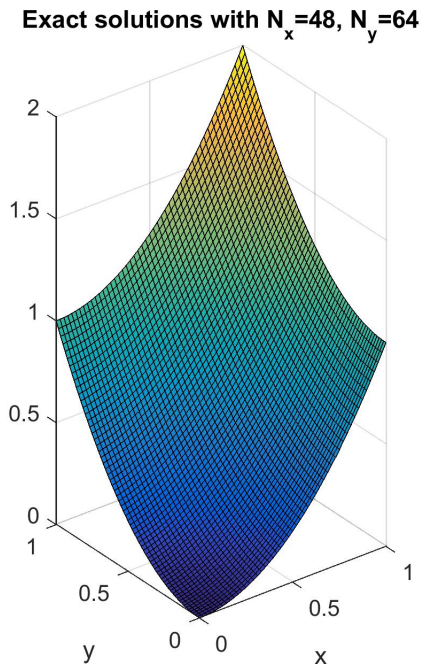

Figure 152: Exact Solutions: Problem 6.2, $Nx = 48, Ny = 64$

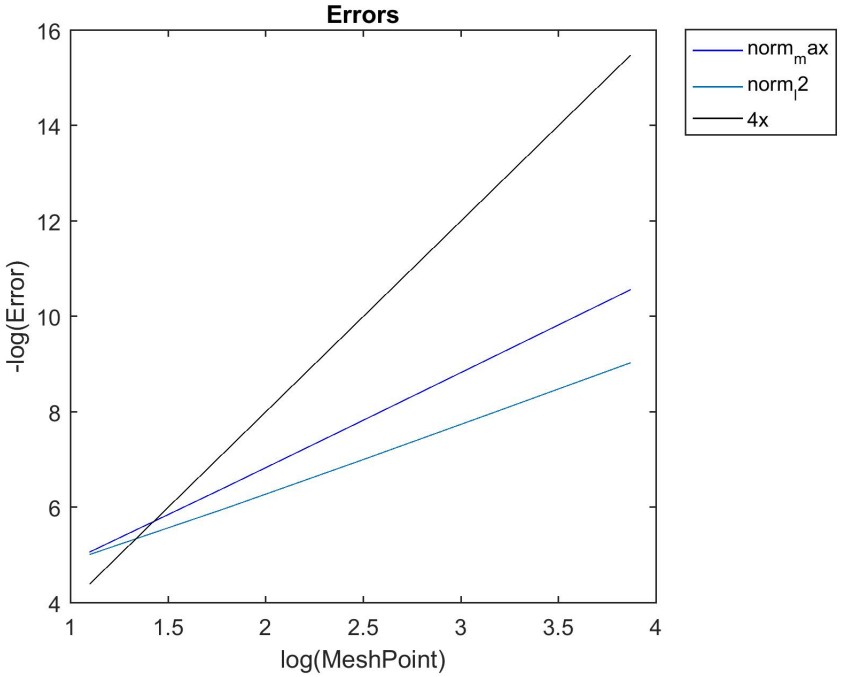

Figure 153: Error on a log-log scale: Problem 6.2

8.3 Matlab Implementation for Problem 6.3

Remark 8.2. Our team current work on pure Neumann boundry condition has a serious problem. The matrix A created is a singular matrix which leads to inaccurate results.

Any readers who have an accurate MATLAB Implement can contact us for more discussion.

THE END

Alphabetical Index

B

C

D

E

F

S

T

U

V

W

References

[1] Randall J. Leveque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM Society for Industrial and Applied Mathematics, 2007.

[2] Nguyen Quan Ba Hong, Doan Tran Nguyen Tung, Nguyen An Thinh, *On Approximating Solutions of One-Dimensional Boundary Value Problems by Using Finite Difference Method on Nonuniform Mesh*, October 17, 2016.
Download. Hong-Tung-Thinh Teamwork, FDM 1D

[3] Luca Guido Molinari, *Determinant of Block Tridiagonal Matrices*, Linear Algebra and its Appllication 429 (2008) 2221-2226, July 27, 2008.

[4] Y Huang, W F McColl, *Analytical Inversion of General Tridiagonal Matrices*, J. Phys. A: Math. Gen. 30 (1997) 79197933. Printed in the UK, June 19, 1997.

[5] Silvia Noschese Lionello Pasquini, Lothar Reichel, *Tridiagonal Toeplitz Matrices: Properties and Novel Applications*, Numer. Linear Algebra Appl. 2006.

[6] V. Prasolov, *Problems and Theorems in Linear Algebra*, 1994.

[7] `https://en.wikipedia.org/wiki/Diagonally_dominant_matrix`

[8] `https://en.wikipedia.org/wiki/Gershgorin_circle_theorem`

[9] `https://en.wikipedia.org/wiki/Normal_matrix`

[10] `https://en.wikipedia.org/wiki/List_of_trigonometric_identities`

[11] `https://en.wikipedia.org/wiki/Commutation_matrix`

[12] `http://math.stackexchange.com/questions/2018162/eigenvalue-decomposition-of-a-triadiagonal-matrix/2018311#2018311`

[13] `http://math.stackexchange.com/questions/2018197/eigenvalues-and-eigenvectors-of-a-tridiagonal-block-matrix/2019766#2019766`