# Numerical Solution of the 2D Poisson Equation

Jesse RONG, Tommso Melotti, Gwendolyn Gillian Glodt

project of Numerical Analysis in , University of luxembourg

January 30, 2025

# Outline

# Problem Statement

- Solve the 2D Poisson equation:

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f(x, y), \quad \text{on } \Omega = [0, 1] \times [0, 1].$$

- Boundary conditions:

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

- Exact solution for validation:

$$u(x, y) = \sin^2(\pi x)\sin^2(\pi y).$$

# Finite Difference Method

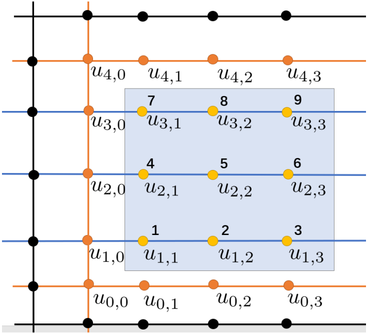
- Discretize the domain into a uniform grid with spacing $h = \frac{1}{N+1}$.
- Approximate derivatives using central differences:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}, \quad \frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}.$$

- Discretized Poisson equation:

$$-4u_{i,j} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} = h^2 f_{i,j}.$$

- Diagram



# Matrix Formulation

- Numbering the grid points in column-major order, we obtain a linear system,The equation is given as:

$$Du_{j-1} + Cu_j + Du_{j+1} = f_j, \quad 1 \leq j \leq n-1.$$

where

# Matrix Formulation

$$C = \begin{pmatrix} 2\left(\frac{1}{h_1^2}+\frac{1}{h_2^2}\right) & -\frac{1}{h_2^2} & 0 & \cdots & 0 \\ -\frac{1}{h_2^2} & 2\left(\frac{1}{h_1^2}+\frac{1}{h_2^2}\right) & -\frac{1}{h_2^2} & \cdots & 0 \\ 0 & -\frac{1}{h_2^2} & 2\left(\frac{1}{h_1^2}+\frac{1}{h_2^2}\right) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -\frac{1}{h_2^2} \\ 0 & 0 & 0 & -\frac{1}{h_2^2} & 2\left(\frac{1}{h_1^2}+\frac{1}{h_2^2}\right) \end{pmatrix},$$

$$D = \begin{pmatrix} -\frac{1}{h_2^2} & 0 & 0 & \cdots & 0 \\ 0 & -\frac{1}{h_2^2} & 0 & \cdots & 0 \\ 0 & 0 & -\frac{1}{h_2^2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{h_2^2} \end{pmatrix},$$

# Matrix Formulation

$$f_j = f + b = \begin{pmatrix} f(x_1, y_j) + \frac{1}{h_1^2}\varphi(x_0, y_j) \\ f(x_2, y_j) \\ \vdots \\ f(x_{m-2}, y_j) \\ f(x_{m-1}, y_j) + \frac{1}{h_1^2}\varphi(x_m, y_j) \end{pmatrix}.$$

Equation can be further written as:

$$\begin{pmatrix} C & D & 0 & \cdots & 0 \\ D & C & D & \cdots & 0 \\ 0 & D & C & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & D \\ 0 & 0 & 0 & D & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 - Du_0 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} - Du_n \end{pmatrix}.$$

Since $u(x,1) = u(x,0) = u(1,y) = u(0,y) = 0$, the vector $b$ is a zero-vector the size of $(N_y - 1)^2$. For a procedure with a non-zero boundary, the vector resembles:

# Matrix Formulation

$$A = \begin{pmatrix} C & D & 0 & \cdots & 0 \\ D & C & D & \cdots & 0 \\ 0 & D & C & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & D \\ 0 & 0 & 0 & D & C \end{pmatrix}$$

$$b + f = \begin{pmatrix} f_1 - Du_0 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} - Du_n \end{pmatrix}.$$

# Validation of the Implementation

- The exact solution we are solving is given by:

$$u(x, y) = \sin^2(\pi x)\sin^2(\pi y).$$

- The corresponding Poisson equation:

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

- Substituting the exact solution $u(x, y)$ into the Laplacian, the right-hand side $f(x, y)$ is derived as:

$$f(x, y) = 2\pi^2 \left(\cos(2\pi x)\sin^2(\pi y) + \cos(2\pi y)\sin^2(\pi x)\right).$$

# Validation of Boundary Conditions

- The exact solution is:

$$u_{ex}(x, y) = \sin^2(\pi x) \sin^2(\pi y).$$

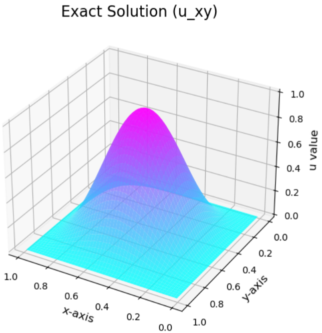
- Verify boundary conditions:
  1. At $x = 0$:
  $$u_{ex}(0, y) = \sin^2(\pi \cdot 0) \sin^2(\pi y) = 0.$$
  2. At $x = 1$:
  $$u_{ex}(1, y) = \sin^2(\pi \cdot 1) \sin^2(\pi y) = \sin^2(\pi) \sin^2(\pi y) = 0.$$
  3. At $y = 0$:
  $$u_{ex}(x, 0) = \sin^2(\pi x) \sin^2(\pi \cdot 0) = 0.$$
  4. At $y = 1$:
  $$u_{ex}(x, 1) = \sin^2(\pi x) \sin^2(\pi \cdot 1) = \sin^2(\pi x) \sin^2(\pi) = 0.$$

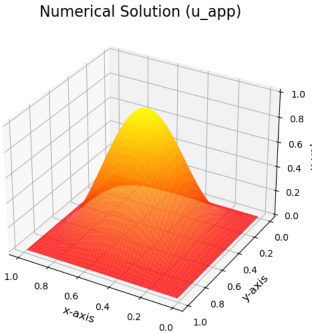
- Hence, $u_{ex}(x, y)$ satisfies:

$$u(x, 0) = u(x, 1) = u(0, y) = u(1, y) = 0.$$

# Validation of the Implementation

- result using numpy.linalg.solve:



# 2.1 Direct Methods

- For the linear system, the matrix is tri-diagonal and sparse.
- Using a sparse representation, only non-zero entries are stored.
- Computational cost:
  - Gaussian elimination for $n \times n$ matrix: $O(n^3)$.
  - For a sparse matrix: $O(n^2)$ for the 2D Laplacian.
- Compare numerical and exact solutions.
- Plot the error:

$$\|u - u_h\|_\infty = \max |u(x_i, y_j) - u_h(x_i, y_j)|.$$

# 2.2 Iterative Methods: Finite Difference Approximation

- The second derivative approximation:

$$D_2 = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix},$$

  where $h$ is the grid spacing.

- Eigenvalues of $D_2$:

$$\lambda_k = -\frac{4}{h^2} \sin^2\left(\frac{k\pi}{2n}\right), \quad k = 1, 2, \ldots, n-1.$$

# Chebyshev Polynomials

- Chebyshev polynomials $T_n(x)$ satisfy:

$$T_0(x) = 1, \quad T_1(x) = x,$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1.$$

- Eigenvalues of the second derivative matrix relate to zeros of $T_n(x)$:

$$x_k = \cos\left(\frac{(2k-1)\pi}{2n}\right), \quad k = 1, 2, \ldots, n.$$

# Iterative Method with Recurrence Relation

- In 1D discrete case with Dirichlet boundary conditions:

$$\frac{v_{k+1} - 2v_k + v_{k-1}}{h^2} = \lambda v_k, \quad k = 1, \ldots, n, \quad v_0 = v_{n+1} = 0.$$

- Rearranging terms:

$$v_{k+1} = (2 + h^2\lambda)v_k - v_{k-1}.$$

- Let $2\alpha = 2 + h^2\lambda$. Then:

$$v_0 = 0, \quad v_1 = 1, \quad v_{k+1} = 2\alpha v_k - v_{k-1}.$$

- General solution:

$$v_k = U_k(\alpha),$$

where $U_k$ is the $k$-th Chebyshev polynomial of the second kind.

# Eigenvalues for the 2D Laplacian

- For the 2D Laplacian, eigenvalues are:

$$\lambda_{i,j} = \lambda_i + \lambda_j,$$

  where $\lambda_i$ and $\lambda_j$ are eigenvalues of the 1D Laplacian.

- Thus:

$$\lambda_{i,j} = 2 - 2\cos\left(\frac{\pi i}{N+1}\right) - 2\cos\left(\frac{\pi j}{N+1}\right), \quad i, j = 1, 2, \ldots, N.$$

# Convergence of Iterative Methods

- Laplacian matrix properties:
  - Symmetric and semi-positive definite.
  - Eigenvalues are non-negative:

$$\lambda_i = 2 - 2\cos\left(\frac{\pi i}{N+1}\right).$$

- Condition number increases as grid size grows:
  - Smallest eigenvalue approaches zero.
  - Leads to a spectral radius closer to 1, slowing convergence.
- Convergence radius of iterative method:

$$\rho(T) = \max|\lambda_i(T)|,$$

where $\lambda_i(T)$ are eigenvalues of iteration matrix $T$.

# Jacobi Method

- The iteration matrix for Jacobi:

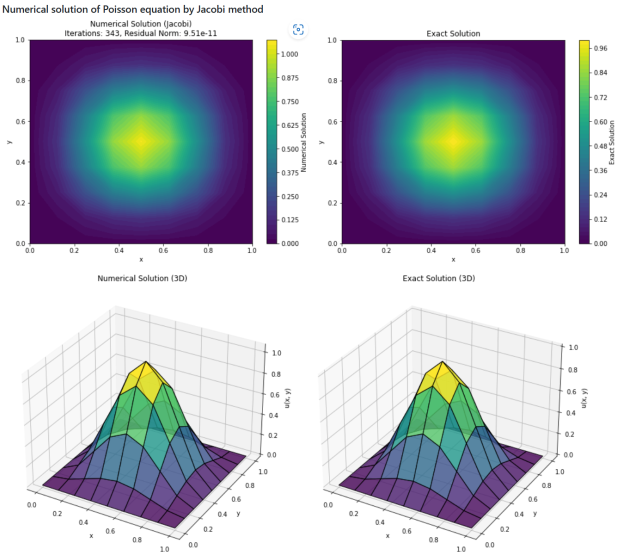
$$T_J = D^{-1}(L + U),$$

  where:
  - $D$: Diagonal part of the system matrix.
  - $L$: Lower triangular part.
  - $U$: Upper triangular part.
- For the discrete Poisson problem, the convergence radius:

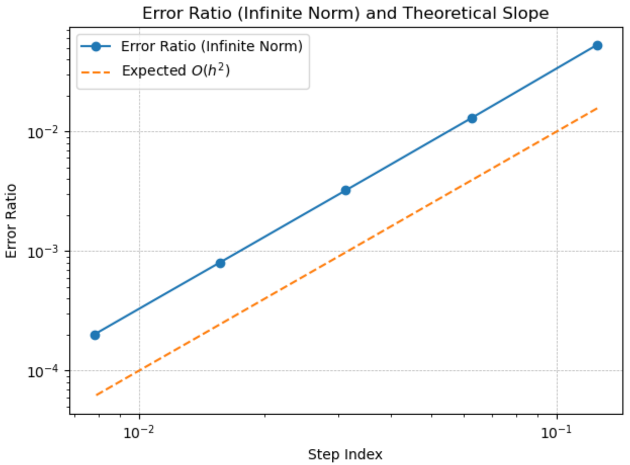
$$\rho(T_J) = \max \left| 1 - \frac{h^2 \lambda_i}{2} \right|.$$

- As grid size increases, smallest eigenvalues $\lambda_i$ approach zero, making $\rho(T_J)$ approach 1, slowing convergence.

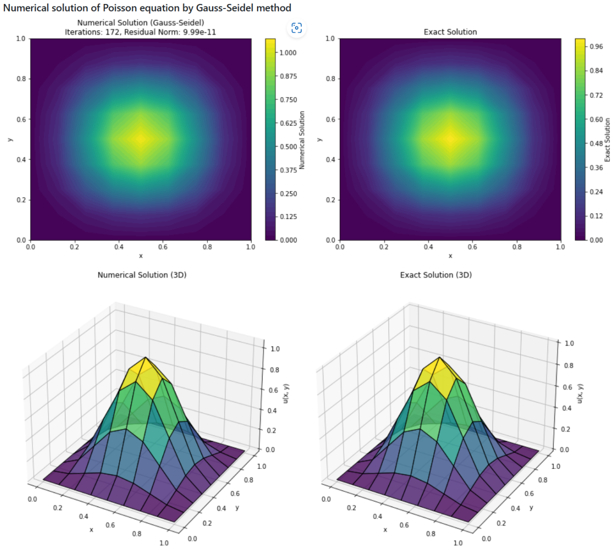
# Jacobi Method



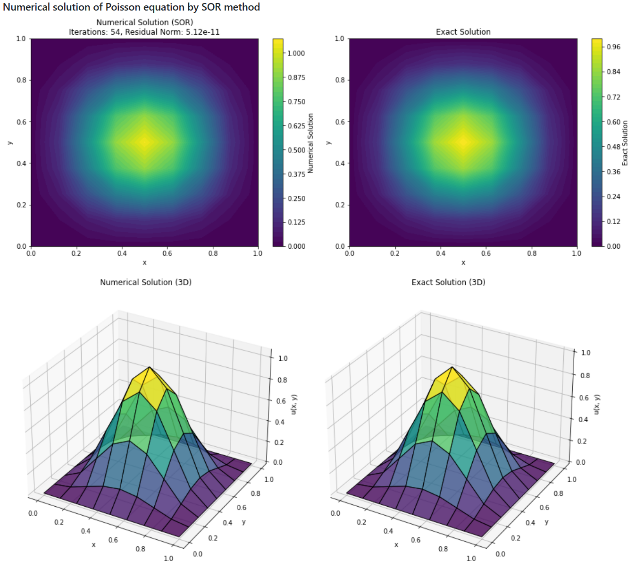
# Jacobi Method



# Gauss-Seidel method



# SOR method



Numerical solution of Poisson equation by SOR method
Numerical Solution (SOR)
Iterations: 54, Residual Norm: 5.12e-11
Numerical Solution
Exact Solution
Numerical Solution (3D)
Exact Solution (3D)
u(x, y)
x
y

# Cost of an Iterative Solver

- The cost of an iterative solver depends on:
  1. **Cost per iteration**: Computational cost of a single iteration.
  2. **Number of iterations**: Total iterations required for convergence.

**Example: Jacobi Method**

- Sparse system matrix $A$ (e.g., discrete Laplacian) for $N \times N$ grid:
  - $O(N^2)$ unknowns.
  - $O(N^2)$ nonzero entries for a 5-point stencil.
- **Cost per iteration**:
  - Sparse matrix-vector multiplication: $O(N^2)$.
  - Additional vector operations (addition, scaling): $O(N^2)$.
- Total cost per iteration: $O(N^2)$.

$$\text{Total Cost} = \text{Cost per Iteration} \times \text{Number of Iterations}.$$

# Successive Over-Relaxation (SOR) Method

- **Iteration matrix for SOR**:

$$T_{\mathrm{SOR}} = (D - \omega L)^{-1}[(1 - \omega)D + \omega U].$$

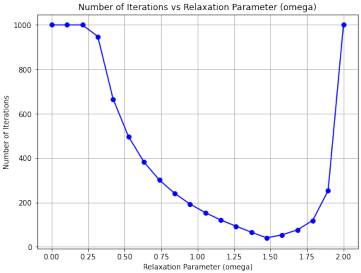
- Convergence rate depends on spectral radius $\rho(T_{\mathrm{SOR}})$.
- The smaller the spectral radius, the faster the convergence.

# Optimal Relaxation Parameter $\omega_{\text{opt}}$

- For SOR, the optimal relaxation parameter $\omega_{\text{opt}}$ minimizes the spectral radius.
- It is given by:

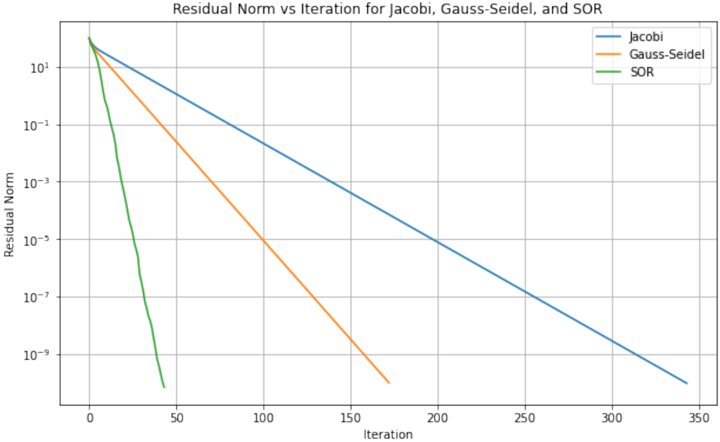
$$\omega_{\text{opt}} = \frac{2}{1 + \sin\left(\frac{\pi}{n+1}\right)}.$$

- This value ensures faster convergence for the SOR method.




- finding the best omega by plot:

# Comparison between the three methods

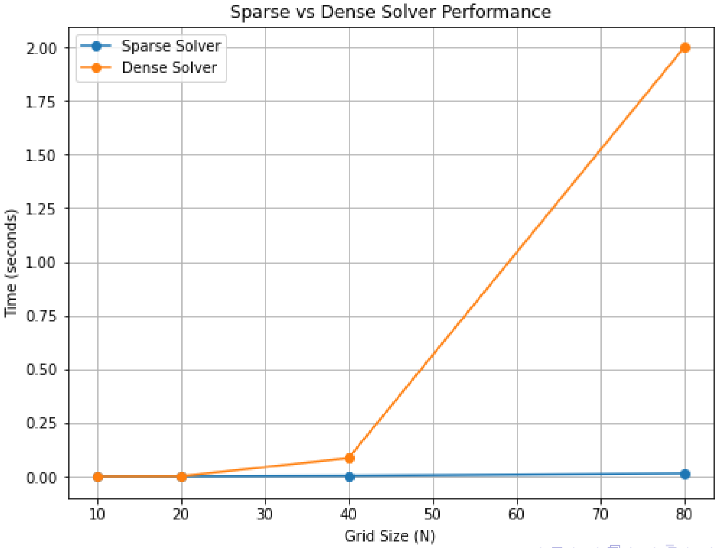


- The coefficient matrix $A$ has the form:

$$A = I_n \otimes T + T \otimes I_n,$$

where $\otimes$ is the Kronecker product. so we can use scipy.kron to create sparse matrix and use spsolve in scipy.sparse.linalg to solve the linear system.

# Computational Time Between The Dense and Sparse Matrix Solvers



# Richardson Extrapolation

**Solution Representation:**

$$u_{ij}(h_1, h_2) \quad \text{approximates} \quad u(x, y).$$

**Theorem: Problem Definition**

$$-\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = \frac{1}{12}\frac{\partial^4 u(x, y)}{\partial x^4}, \quad (x, y) \in \Omega, \tag{1}$$

$$v = 0, \quad (x, y) \in \Gamma.$$

$$-\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) = \frac{1}{12}\frac{\partial^4 u(x, y)}{\partial y^4}, \quad (x, y) \in \Omega, \tag{2}$$

$$w = 0, \quad (x, y) \in \Gamma.$$

# Existence of a Smooth Solution

**Error Bound:**

$$\max_{\substack{1\le i\le m-1\\ 1\le j\le n-1}} \left| u(x_i, y_j) - \left[\frac{4}{3} u_{2i,2j}\left(\frac{h_1}{2}, \frac{h_2}{2}\right) - \frac{1}{3} u_{ij}(h_1, h_2)\right]\right| = O(h_1^4 + h_2^4), \quad (3)$$

where:

$$h_1 = \frac{b-a}{m}, \quad h_2 = \frac{d-c}{n}.$$

**Error Equation:**

$$-(\delta_x^2 e_{ij}+\delta_y^2 e_{ij}) = -\frac{h_1^2}{12}\frac{\partial^4 u(x_i,y_j)}{\partial x^4}-\frac{h_2^2}{12}\frac{\partial^4 u(x_i,y_j)}{\partial y^4}-\frac{h_1^4}{360}\frac{\partial^6 u(\xi_{ij},y_j)}{\partial x^6}-\frac{h_2^4}{360}\frac{\partial^6 u(}{\partial} \tag{4}$$

$$e_{ij}=0, \quad (i,j)\in\gamma.$$

**Difference Schemes:**

$$-(\delta_x^2 v_{ij}+\delta_y^2 v_{ij}) = \frac{1}{12}\frac{\partial^4 u(x_i,y_j)}{\partial x^4}, \quad v_{ij}=0, \quad (i,j)\in\gamma, \tag{5}$$

$$-(\delta_x^2 w_{ij}+\delta_y^2 w_{ij}) = \frac{1}{12}\frac{\partial^4 u(x_i,y_j)}{\partial y^4}, \quad w_{ij}=0, \quad (i,j)\in\gamma. \tag{6}$$

# Theorem Results and Extrapolation

**Theorem Results:**

$$v(x_i, y_j) - v_{ij}(h_1, h_2) = O(h_1^2 + h_2^2), \quad (i,j) \in \omega, \tag{7}$$

$$w(x_i, y_j) - w_{ij}(h_1, h_2) = O(h_1^2 + h_2^2), \quad (i,j) \in \omega. \tag{8}$$

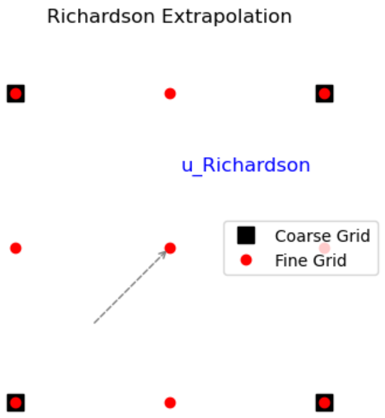
**Final Approximation:**

$$u_{ij}(h_1, h_2) = u(x_i, y_j) + h_1^2 v(x_i, y_j) + h_2^2 w(x_i, y_j) + O(h_1^4 + h_2^4), \quad (i,j) \in \omega. \tag{9}$$

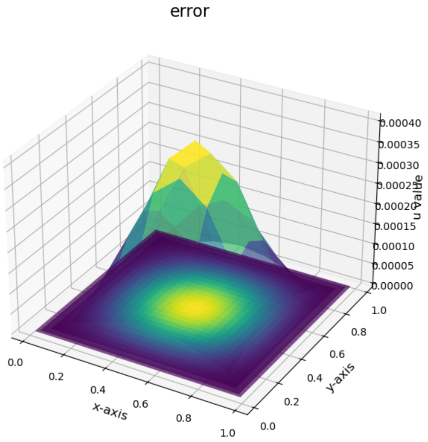
# Richardson Extrapolation Formula

**Higher Accuracy Approximation:**

$$\frac{4}{3}u_{2i,2j}\left(\frac{h_1}{2},\frac{h_2}{2}\right)-\frac{1}{3}u_{ij}(h_1,h_2)=u(x_i,y_j)+O(h_1^4+h_2^4),\quad (i,j)\in\omega. \tag{10}$$



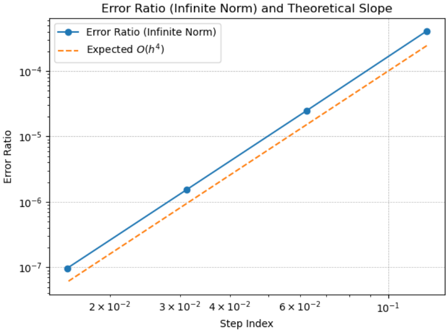
# Richardson Extrapolation Formula




The error plot of Richardson:

# Richardson Extrapolation Formula

Convergence order:



In this section, we establish a difference scheme with an accuracy of $O(h_1^4 + h_2^4)$ for solving the boundary value problems.
Let $v = \{v_{ij} \mid 0 \leq i \leq m, 0 \leq j \leq n\}$. Define the operators as follows:

$$(Av)_{ij} = \begin{cases} -\frac{1}{12}\left(v_{i-1,j} + 10v_{ij} + v_{i+1,j}\right), & 1 \leq i \leq m-1,\ 0 \leq j \leq n, \\ -v_{ij}, & i = 0,\ 0 \leq j \leq n, \end{cases} \tag{11}$$

$$(Bv)_{ij} = \begin{cases} -\frac{1}{12}\left(v_{i,j-1} + 10v_{ij} + v_{i,j+1}\right), & 1 \leq j \leq n-1,\ 0 \leq i \leq m, \\ -v_{ij}, & j = 0,\ 0 \leq i \leq m. \end{cases} \tag{12}$$

At the grid point $(x_i, y_j)$, the differential equation is:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) + \frac{\partial^2 u}{\partial y^2}(x_i, y_j) = f(x_i, y_j), \quad 0 \leq i \leq m, \ 0 \leq j \leq n. \tag{13}$$

Applying the operator $AB$ gives:

$$AB\frac{\partial^2 u}{\partial x^2}(x_i, y_j) + AB\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = ABf(x_i, y_j), \quad 1 \leq i \leq m-1, \ 1 \leq j \leq n-1. \tag{14}$$

This can be rewritten as:

$$B\left(A\frac{\partial^2 u}{\partial x^2}(x_i, y_j)\right) + A\left(B\frac{\partial^2 u}{\partial y^2}(x_i, y_j)\right) = ABf(x_i, y_j), \quad 1 \leq i \leq m-1, \ 1 \leq j \tag{15}$$

By a Lemma, we have:

$$A\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \delta_x^2 u_{ij} + \frac{h_1^4}{240}\frac{\partial^6 u}{\partial x^6}(\xi_{ij}), \quad 1 \le i \le m-1,\ 0 \le j \le n, \tag{16}$$

$$B\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \delta_y^2 u_{ij} + \frac{h_2^4}{240}\frac{\partial^6 u}{\partial y^6}(\eta_{ij}), \quad 0 \le i \le m,\ 1 \le j \le n-1. \tag{17}$$

Here, $\xi_{ij} \in (x_{i-1}, x_{i+1})$, $\eta_{ij} \in (y_{j-1}, y_{j+1})$.
Define:

$$P_{ij} = \frac{h_1^4}{240}\frac{\partial^6 u}{\partial x^6}(\xi_{ij}), \quad 1 \le i \le m-1,\ 0 \le j \le n, \tag{18}$$

$$Q_{ij} = \frac{h_2^4}{240}\frac{\partial^6 u}{\partial y^6}(\eta_{ij}), \quad 0 \le i \le m,\ 1 \le j \le n-1. \tag{19}$$

From equations (16) and (17), we obtain:

$$A\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \delta_x^2 u_{ij} + P_{ij}, \quad 1 \le i \le m-1,\ 0 \le j \le n, \tag{20}$$

$$B\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \delta_y^2 u_{ij} + Q_{ij}, \quad 0 \le i \le m,\ 1 \le j \le n-1. \tag{21}$$

Substituting into (15):

$$-\left[B\left(\delta_x^2 u_{ij} + P_{ij}\right) + A\left(\delta_y^2 u_{ij} + Q_{ij}\right)\right] = ABf_{ij}, \quad (i,j) \in \omega. \tag{22}$$

Simplifies to:

$$-\left(B\delta_x^2 u_{ij} + A\delta_y^2 u_{ij}\right) = ABf_{ij} + R_{ij}, \quad (i,j) \in \omega, \tag{23}$$

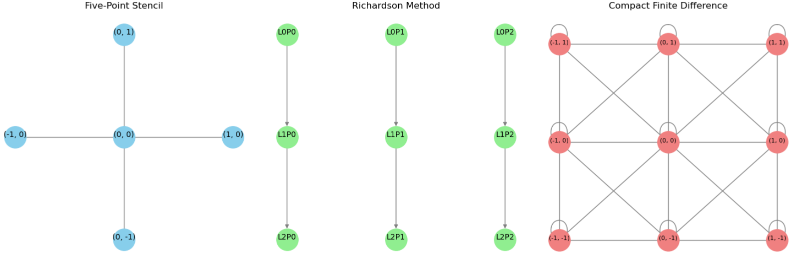
where:

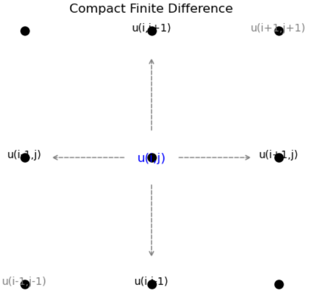
$$R_{ij} = BP_{ij} + AQ_{ij}, \quad (i,j) \in \omega. \tag{24}$$

Define:

$$M_6 = \max\left\{\max_{(x,y)\in\Omega}\left|\frac{\partial^6 u(x,y)}{\partial x^6}\right|, \max_{(x,y)\in\Omega}\left|\frac{\partial^6 u(x,y)}{\partial y^6}\right|\right\}. \tag{25}$$

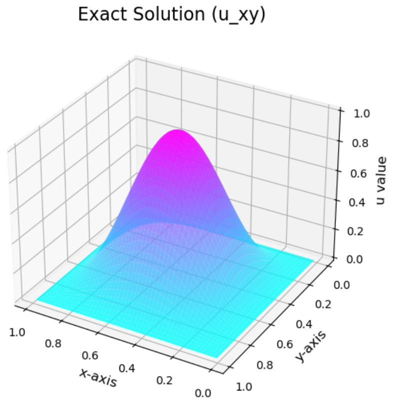
From equations (23), (20), and (21), we know:

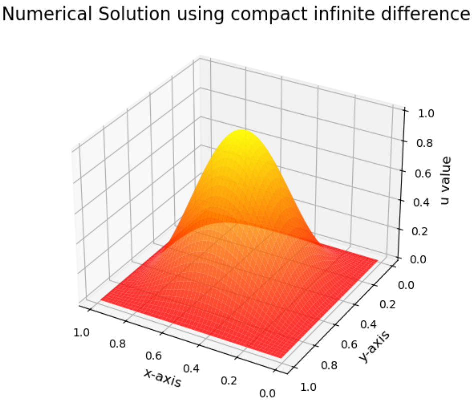


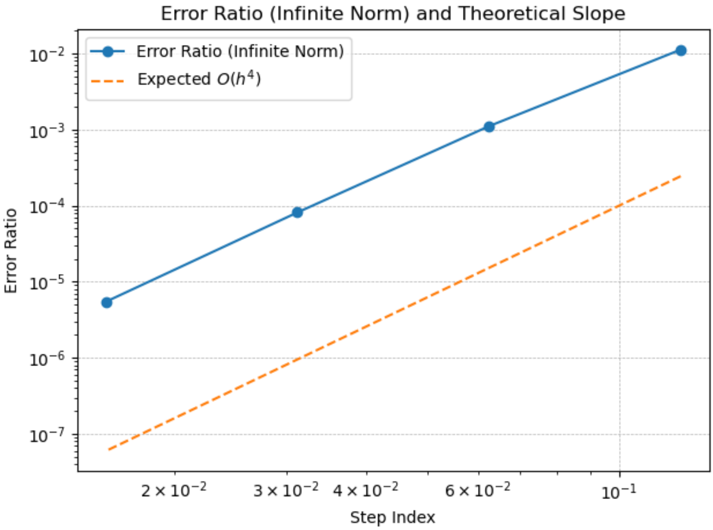

$$|R_{ij}| \leq \frac{1}{240} M_6 \left(h_1^4 + h_2^4\right), \quad (i,j) \in \omega. \tag{26}$$

# Construction of the Compact Difference Scheme



# Construction of the Compact Difference Scheme



# Laplace equation with non-homogeneous boundary condition

- The Laplace equation is given by:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,$$

  for $(x, y) \in (0, 1) \times (0, 1)$.

- The boundary conditions are:

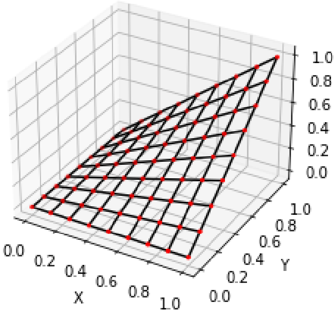
$$u(0, y) = 0, \quad \text{for } y \in [0, 1].$$

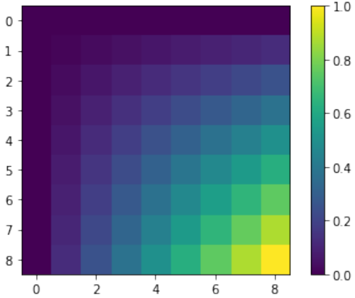
$$u(x, 0) = 0, \quad \text{for } x \in [0, 1].$$

$$u(1, y) = y, \quad \text{for } y \in [0, 1].$$

$$u(x, 1) = x, \quad \text{for } x \in [0, 1].$$

# Laplace equation with non-homogeneous boundary condition



# Conclusion

- Successfully solved the 2D Poisson equation using finite difference methods.
- Validated numerical results against exact solution.
- Explored iterative methods and their convergence properties.
- Constract higher order methods
- Implement our methods on Laplace eqution with non-homogeneous boundary condition.

# Acknowledgment

**Thank You!**


- Thank you for your attention!
- Special thanks to Pro.Hadrien BERIOT, who taught us how to use the difference method and supported this work.
- Questions and discussions are welcome!