Numerical Solution of the Biharmonic Equation using Mimetic Differences

Vinayaka Hedge and Miguel A. Dumett

May 8, 2026

Publication Number: CSRCR2026-13

Computational Science & Engineering Faculty and Students Research Articles

Database Powered by the Computational Science Research Center Computing Group & Visualization Lab

COMPUTATIONAL SCIENCE & ENGINEERING

SAN DIEGO STATE UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

Numerical Solution of the Biharmonic Equation using Mimetic Differences

Vinayaka Hedge* Miguel A. Dumett‡

May 8, 2026

Abstract

The biharmonic equation $\Delta^2 u = f$ models the steady-state deflection of a thin, clamped elastic plate. In this report, the equation is solved on the domain of a unit square using the Mimetic Operators Library Enhanced (MOLE) with second-order mimetic operators on a 2D staggered grid. The discrete bilaplacian analog is created by simply doing $A = L_{2D}\, L_{2D}$, and the clamped boundary conditions ($u = 0$, $\partial_n u = 0$) are implemented via this strategy: Dirichlet nodes are eliminated by algebraic substructuring, and inner-ring rows are overwritten using the `robinBC2D` from MOLE. The computed numerical solution was compared with a given exact solution and the L_∞ and L_2 error norms were calculated and were shown to converge as the grid resolution was increased.

1 Introduction

1.1 Physical Context of the PDE:

The biharmonic equation models the steady-state deflection of a thin, clamped elastic plate under a distributed load [4].

1.2 Why using mimetic differences to solve this PDE?

1. The Mimetic Operators Library Enhance (MOLE) [5], makes really easy to use discrete analogs of continuous vector calculus operators like (∇, $\nabla\cdot$, Δ, $\nabla\times$) using sparse matrices. [3]
2. A few important properties of MOLE: [1, 2, 5]:
   - Global and local conservation laws are satisfied.
   - Accuracy is maintained across the interior and boundary of the domain.
   - The identities of vector calculus are maintained.
   - The matrices produced are always sparse.
3. For this 4th-order PDE, the bilaplacian is created by simply doing $\Delta^2 \approx L_{2D}\, L_{2D}$, and all the above properties are still valid.

*Computational Science Master Program at San Diego State University (`vhegde7143@sdsu.edu`).
†Editor: Jose E. Castillo.
‡Computational Science Research Center at San Diego State University (`mdumett@sdsu.edu`).

1.3 Goals of this work

1. Create a 2nd-order mimetic discretization of $\Delta^2 u = f$ on a 2D staggered grid using MOLE.
2. Implement clamped boundary conditions on the domain.
3. Calculate L_∞ and L_2 error metrics as the resolution of the grid is increased using the given exact solution.

2 Details of the PDE:

2.1 PDE:

$$\Delta^2 u(x,y) \;=\; f(x,y), \qquad (x,y) \in \Omega = (0,1)\times(0,1). \tag{1}$$

2.2 Boundary conditions:

$$u = 0 \qquad \text{on } \partial\Omega \quad \text{(Dirichlet Conditions)}, \tag{2}$$
$$\partial_n u = 0 \qquad \text{on } \partial\Omega \quad \text{(Neumann Conditions)}. \tag{3}$$

Physical Context of the "clamped plate" boundary conditions

- Dirichlet ($u = 0$): The plate is fixed at its edges and CANNOT be moved.
- Neumann ($\partial_n u = 0$): The bending of the plate is zero at the edges.

2.3 Exact solution and forcing function

I used an exact solution given to me to calculate error metrics and verify my method.

$$u_{\text{exact}}(x,y) \;=\; x^4(x-1)^2\, y^4(y-1)^2. \tag{4}$$

The forcing function is:

$$\begin{aligned} f(x,y) \;&= 24\,(1-10x+15x^2)(1-y)^2y^4 \\ &\quad + 24\,(1-x)^2x^4(1-10y+15y^2) \\ &\quad + 8\,x^2(6-20x+15x^2)\,y^2(6-20y+15y^2). \end{aligned} \tag{5}$$

3 Methodology

3.1 Utilization of a staggered grid in MOLE [3]

Staggered Grid Layout:

- The domain is divided into a grid of size $m \times n$ ($m = n = M$).
- Scalar quantities (u, f) live at the *cell centers and at the boundary edges*. There are a total of $(m{+}2)(n{+}2)$ scalar nodes on the grid.
- Vector quantities (gradients ∇u) live on the *cell faces*.

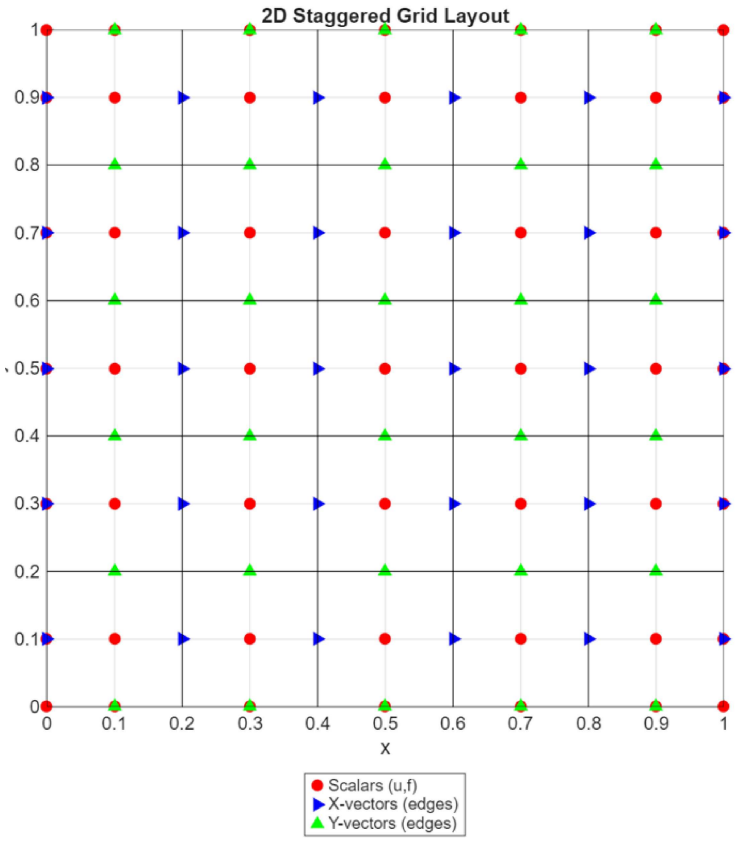

Figure 1: 2D staggered grid layout for $M = 5$. Scalar nodes (red) sit at cell centers and boundary edges; vector components (blue/green) sit on cell edges.

Operators used from MOLE to discretize the domain:

- $L_{2D} = \texttt{lap2D}(k, m, dx, n, dy)$ is a discrete Laplacian operator.
- $B_N = \texttt{robinBC2D}(k, m, dx, n, dy, 0, 1)$ implements the Robin condition $a\,u + b\,\partial_n u = 0$. I have set $a = 0$, $b = 1$, which reduces it to the Neumann condition ($\partial_n u = 0$).
- Order of accuracy is set to $k = 2$ across all operators.

3.2 Creating the Bilaplacian

- The matrix A is the discrete analog of the bilaplacian:

$$A \;=\; L_{2D}\, L_{2D} \;\approx\; \Delta^2. \tag{6}$$

- A inherits all mimetic properties of L_{2D} [3] and is ***sparse***.
- The continuous PDE $\Delta^2 u = f$ has now become a discrete linear system:

$$A\,\mathbf{u} \;=\; \mathbf{f}. \tag{7}$$

3.3 Implementation of the Boundary Conditions

I will be giving an easy explanation by walking through this example using $M = 5$ for the grid, which gives a $7 \times 7 = 49$-node staggered grid. The exact same procedure applies at every grid size we select.

Step 1 - Classification of nodes:

- ***Outer ring*** (24 nodes): They represent the boundary of the plate where we need to implement $u = 0$.
- ***Inner ring*** (16 nodes): They represent the layer just inside the boundary, where we need to implement $\partial_n u = 0$.
- ***Interior*** (9 nodes): These are NOT changed.

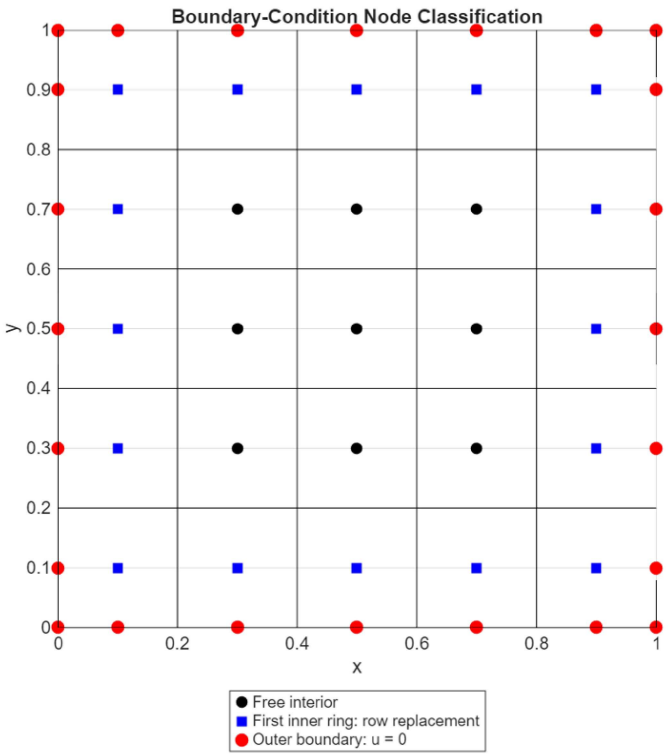

Figure 2: Boundary-condition node classification on the $M = 5$ grid. Red circles are the outer-boundary nodes ($u = 0$, to be eliminated). Blue squares are inner ring nodes ($\partial_n u = 0$, rows to be replaced). The Black circles are interior nodes (unchanged).

Step 2 - Implementing the Neumann boundary condition on the 16 inner-ring nodes:

- For each of the 16 inner-ring nodes, we consider its boundary node adjacent to it.
- We will then **overwrite** that row of A with the mimetic values obtained from the ***robinBC2D*** operator from MOLE.

$$A_{\text{row, inner}} \longleftarrow B_{N,\text{row, boundary}}.$$

- We will also replace the corresponding value in matrix $\mathbf{f}$ with 0.
- Therefore, we have now replaced 16 rows in the matrix A.

Step 3 - Implementing the Dirichlet boundary condition on the 24 outer ring nodes:

- We can eliminate the 24 outer ring nodes since $u = 0$.
- To do this we just remove the 24 outer-ring rows AND columns from A:
  - Rows are removed since we already know the value is zero.
  - Columns are removed since they would be multiplying with zero, and are of no use here.
- We also remove the corresponding 24 entries of matrix $\mathbf{f}$.
- We have now reduced the matrix A from : $49 \times 49 \longrightarrow 25 \times 25$. Let us call this reduced matrix A_{FF}.

The final matrix system is:

$$A_{FF}\,\mathbf{u}_F \;=\; \mathbf{f}_F, \qquad A_{FF} \in \mathbb{R}^{25\times 25}. \tag{8}$$

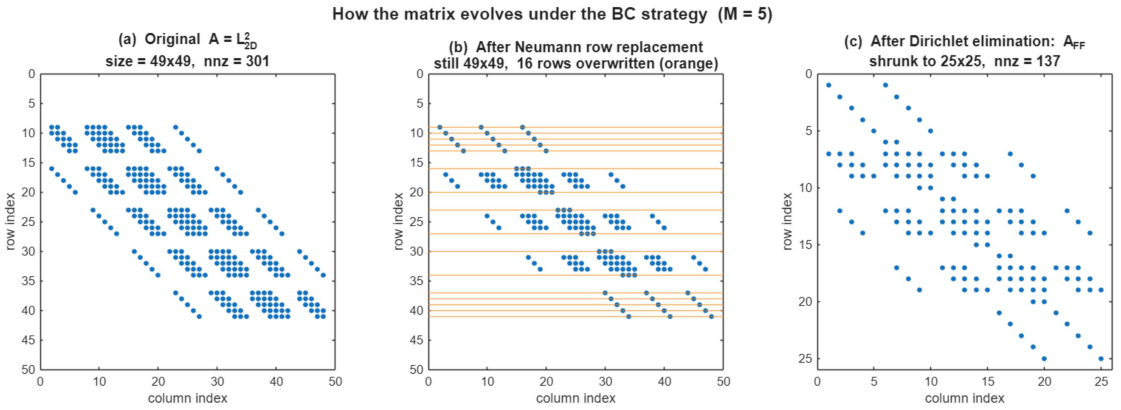

Figure 3: (a) Original $A = L_{2D}^2$ (49×49) ; (b) After Neumann row replacement, it is still 49×49, but 16 rows (highlighted in orange) have been overwritten with robinBC2D operator. (c) After Dirichlet elimination, the matrix has shrunk to 25×25, and now we solve this final system A_{FF}.

3.4 Linear Solver

- I have solved the reduced system using MATLAB's sparse backslash operator:

$$\mathbf{u}_F \;=\; A_{FF} \setminus \mathbf{f}_F. \tag{9}$$

- Then I have reconstructed the full solution vector $\mathbf{u}$ by inserting zeros at the eliminated Dirichlet positions (24 in the $M = 5$ example, $4(M + 1)$ in general).
- I felt this was the best method to use since A_{FF} is sparse and well-structured.

4 Experimental Setup

4.1 Grid Resolutions

- I ran the experiment for grid sizes $M \in \{10, 20, 30, 40, 50, 100, 200, 300, 400\}$, with $h = 1/M$.
- Mimetic order: $k = 2$.

4.2 Error Metrics

To evaluate the numerical method, I compared the computed solution $\mathbf{u}_{\text{num}}$ against the given exact solution $\mathbf{u}_{\text{exact}}$ at every scalar grid node. The discrete error is defined as

$$e_h \;=\; \mathbf{u}_{\text{num}} - \mathbf{u}_{\text{exact}}, \tag{10}$$

and its size is measured by two norms:

- L_∞ **norm** (maximum pointwise error):

$$\|e_h\|_\infty \;=\; \max_{i,j} \big|\, e_h(x_i, y_j) \,\big|. \tag{11}$$

  This metric captures the *worst-case* pointwise deviation and is sensitive to localized failures.

- L_2 **norm** (root-mean-square error, weighted by cell area):

$$\|e_h\|_2 \;=\; \left(\sum_{i,j} e_h(x_i, y_j)^2 \, \Delta x \, \Delta y \right)^{1/2}. \tag{12}$$

  This metric captures *global accuracy* across the domain.

5 Results

5.1 Solution and Pointwise Error

- The maximum pointwise error is $\sim 1.4 \times 10^{-8}$

5.2 Convergence as Grid Resolution Increases

M	h	$\|e_h\|_\infty$	$\|e_h\|_2$
10	0.1000	2.983×10^{-5}	1.369×10^{-5}
20	0.0500	6.672×10^{-6}	2.700×10^{-6}
30	0.0333	2.798×10^{-6}	1.116×10^{-6}
40	0.0250	1.532×10^{-6}	6.107×10^{-7}
50	0.0200	9.617×10^{-7}	3.861×10^{-7}
100	0.0100	2.316×10^{-7}	9.538×10^{-8}
200	0.0050	5.674×10^{-8}	2.388×10^{-8}
300	0.0033	2.505×10^{-8}	1.063×10^{-8}
400	0.0025	1.404×10^{-8}	5.985×10^{-9}

Table 1: L_∞ and L_2 error norms at increasing grid resolutions. Errors decrease as $h \to 0$.

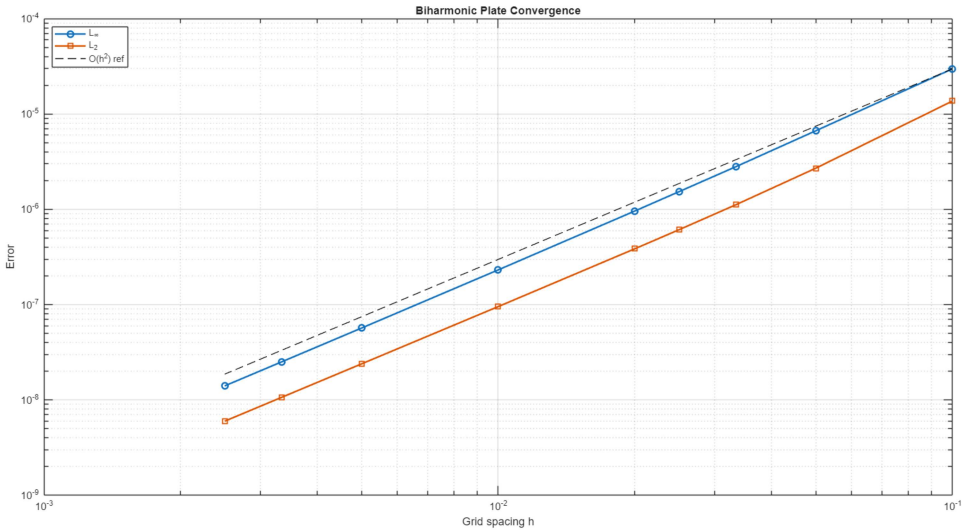

Figure 4: Convergence of L_∞ and L_2 errors versus grid spacing h, on log-log axes, with an $\mathcal{O}(h^2)$ reference line.

Convergence Rates: The rate between grid resolutions is computed as

$$\text{rate} = \frac{\log\big(\|e_h\|_{\text{coarse}} \,/\, \|e_h\|_{\text{fine}}\big)}{\log\big(h_{\text{coarse}} \,/\, h_{\text{fine}}\big)}. \tag{13}$$

Refinement	L_∞ rate	L_2 rate
$M = 10 \to 20$	2.16	2.34
$M = 20 \to 30$	2.14	2.18
$M = 30 \to 40$	2.09	2.10
$M = 40 \to 50$	2.09	2.05
$M = 50 \to 100$	2.05	2.02
$M = 100 \to 200$	2.03	2.00
$M = 200 \to 300$	2.02	2.00
$M = 300 \to 400$	2.01	2.00

Table 2: Observed convergence rates between consecutive resolutions.

5.3 Sparsity of the Reduced Matrix

- At $M = 400$:
  - $\text{size}(A_{FF}) = 160{,}000 \times 160{,}000$.
  - $\text{nnz}(A_{FF}) \approx 2.06 \times 10^6$ (~ 13 nonzeros per row).
- The reduced matrix is sparse and banded

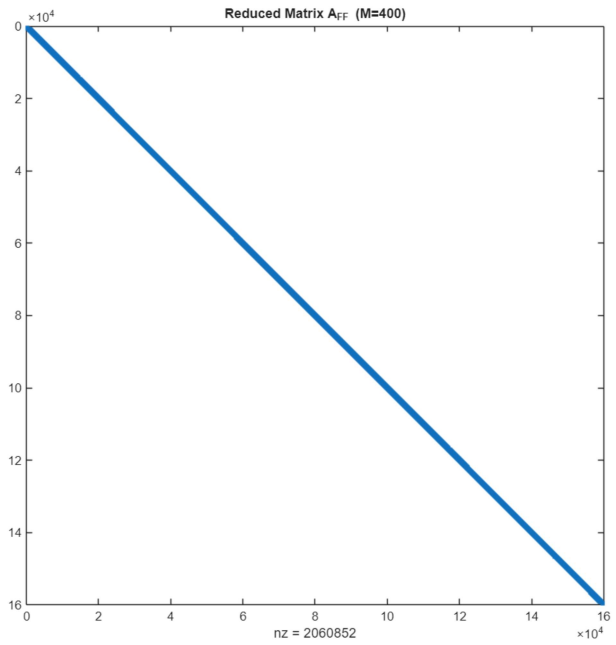

Figure 5: Sparsity pattern of the reduced interior matrix A_{FF} at $M = 400$.

6 Conclusion

- Implemented a 2nd-order mimetic discretization of $\Delta^2 u = f$ on a clamped square plate using the MOLE library.
- Implemented boundary conditions using *algebraic substructuring* (Dirichlet) + *mimetic row replacement* (Neumann).
- Verified $\mathcal{O}(h^2)$ convergence in both L_∞ and L_2.

References

[1] J. Corbino and J.E. Castillo, *High-order mimetic finite-difference operators satisfying the extended Gauss divergence theorem*, J. Comput. Appl. Math., vol. 364, 2020.

[2] J.E. Castillo and R.D. Grone, *A matrix analysis approach to higher-order approximations for divergence and gradients satisfying a global conservation law*, SIAM J. Matrix Anal. Appl., 2003.

[3] J. Corbino and J.E. Castillo, *MOLE: Mimetic Operators Library Enhanced – Documentation*, Computational Science Research Center, SDSU, 2017.

[4] S. Timoshenko and S. Woinowsky-Krieger, *Theory of Plates and Shells*, 2nd ed., McGraw-Hill, 1959.

[5] J. Corbino, M. Dumett, and J. E. Castillo, MOLE: Mimetic Operators Library Enhanced, *Journal of Open Source Software*, 9(99):6288, 2024. https://doi.org/10.21105/joss.06288