Numerical Solution of the 2D Fokker-Planck Equation Using MOLE Mimetic Operators and Forward Euler / Crank-Nicolson Time Integration

Yingjie Ma and Miguel A. Dumett

May 6, 2026

Publication Number: CSRCR2026-08

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 2D Fokker–Planck Equation Using MOLE Mimetic Operators and Forward Euler / Crank–Nicolson Time Integration

Yingjie Ma* Miguel A. Dumett‡

May 6, 2026

Abstract

This report describes a numerical solution of the 2D Fokker–Planck equation on the unit square. The equation has variable drift and diffusion coefficients, so it is rewritten in divergence form, then it is discretized using mimetic operators from the MOLE library [2, 1] on a staggered grid. For time stepping, Forward Euler is used: each step is a single sparse matrix–vector multiply with a pre-built matrix $\mathbf{M} = \mathbf{I} + \Delta t\,\mathbf{L}$. The solver is tested against the known exact solution $u(x, y, t) = (1-x^2)(1-y^2)e^{-t}$ and convergence studies are ran, which confirmed second-order spatial accuracy ($O(h^2)$) and first-order temporal accuracy ($O(\Delta t)$). Crank–Nicolson is also implemented for comparison. It is second-order in time ($O(\Delta t^2)$) and unconditionally stable, so it can use much larger time steps. Both methods are compared in terms of accuracy, convergence rates, and run time.

1 Introduction

The Fokker–Planck equation describes how a probability density function evolves over time under drift and diffusion. It comes up in many areas like statistical mechanics, plasma physics, financial modeling, and neuroscience [3]. The version studied here has coefficients that depend on space, and the diffusion coefficient actually goes to zero at parts of the boundary, which makes the problem a bit more challenging.

Standard finite difference methods do not always preserve conservation properties for equations like this. Mimetic methods [4] are designed to satisfy discrete versions of the divergence theorem, so conservation is built in by construction.

In this work the **MOLE library** (Mimetic Operators Library Enhanced) [1] is used to solve the 2D Fokker–Planck equation on a staggered grid with **Forward Euler** explicit time stepping.

Here is a summary of what has been done in this study:

1. Rewrote the Fokker–Planck equation in pure divergence form so that MOLE's operators could be applied directly.

2. Figured out the correct face-coordinate system that matches MOLE's internal structure, which avoids large flux errors at the boundary.

*Computational Science PhD Program at San Diego State University (yma6999@sdsu.edu).
†Editor: Jose E. Castillo.
‡Computational Science Research Center at San Diego State University (mdumett@sdsu.edu).

3. Built and validated a Forward Euler solver using a pre-built propagator matrix $\mathbf{M}$.
4. Ran spatial and temporal convergence studies and confirmed the expected orders of accuracy.
5. Compared Forward Euler and Crank–Nicolson and found that CN is more efficient for this problem because it can take much larger time steps.

2 Mathematical Formulation

2.1 Governing Equation

The 2D Fokker–Planck equation in non-conservative form is:

$$\frac{\partial u}{\partial t} = -u - \frac{x}{6}\frac{\partial u}{\partial x} - \frac{y}{6}\frac{\partial u}{\partial y} + \frac{x^2}{6}\frac{\partial^2 u}{\partial x^2} + \frac{y^2}{6}\frac{\partial^2 u}{\partial y^2} \tag{1}$$

There are three parts: a reaction term $-u$, variable-coefficient advection terms, and variable-coefficient diffusion terms that go to zero at $x = 0$ and $y = 0$.

2.2 Domain, Initial and Boundary Conditions

- **Spatial domain:** $(x, y) \in [0, 1] \times [0, 1]$
- **Temporal domain:** $t \in [0, 1]$
- **Initial condition:** $u(x, y, 0) = (1 - x^2)(1 - y^2)$

Dirichlet boundary conditions are applied on all four edges:

Boundary	**Location**	**Condition**
Left	$x = 0$	$u(0, y, t) = (1 - y^2)\, e^{-t}$
Right	$x = 1$	$u(1, y, t) = 0$
Bottom	$y = 0$	$u(x, 0, t) = (1 - x^2)\, e^{-t}$
Top	$y = 1$	$u(x, 1, t) = 0$

Table 1: Dirichlet boundary conditions. Left and bottom are time-dependent.

2.3 Exact Solution

The PDE (1) has the closed-form solution:

$$u(x, y, t) = (1 - x^2)(1 - y^2)\, e^{-t} \tag{2}$$

It was verified this by substituting it back into the equation: the advection and diffusion terms cancel out, leaving just $u_t = -u$.

3 Conservative Form Reformulation

3.1 Why Conservative Form Is Needed

MOLE's operators are designed for equations in divergence form. The divergence-gradient chain computes $\nabla \cdot (a\,\nabla u) = a'\,u_x + a\,u_{xx}$, which has an extra product-rule term $a'\,u_x$ that is not in Eq. (1). If one applied MOLE directly to Eq. (1), one would get wrong extra terms. So it was first needed to rewrite the equation in divergence form.

3.2 Divergence Form

Define a diffusion tensor and a drift velocity:

$$\mathbf{D} = \begin{pmatrix} x^2/6 & 0 \\ 0 & y^2/6 \end{pmatrix}, \qquad \mathbf{v} = \begin{pmatrix} x/2 \\ y/2 \end{pmatrix},$$

and rewrite the equation as:

$$\frac{\partial u}{\partial t} = \nabla \cdot \left(\mathbf{D}\,\nabla u - \mathbf{v}\,u\right) \tag{3}$$

3.3 Verification

Expanding the right-hand side of Eq. (3):

$$\nabla \cdot (\mathbf{D}\,\nabla u) = \frac{x}{3}\,u_x + \frac{x^2}{6}\,u_{xx} + \frac{y}{3}\,u_y + \frac{y^2}{6}\,u_{yy}$$

$$\nabla \cdot (\mathbf{v}\,u) = \frac{1}{2}\,u + \frac{x}{2}\,u_x + \frac{1}{2}\,u + \frac{y}{2}\,u_y = u + \frac{x}{2}\,u_x + \frac{y}{2}\,u_y$$

Subtracting:

$$\frac{x^2}{6}\,u_{xx} + \frac{y^2}{6}\,u_{yy} + \left(\frac{x}{3} - \frac{x}{2}\right)u_x + \left(\frac{y}{3} - \frac{y}{2}\right)u_y - u = \frac{x^2}{6}\,u_{xx} + \frac{y^2}{6}\,u_{yy} - \frac{x}{6}\,u_x - \frac{y}{6}\,u_y - u$$

which matches Eq. (1). The drift velocity $\mathbf{v} = (x/2, y/2)$ takes care of both the extra product-rule terms and the reaction term, since $\nabla \cdot \mathbf{v} = 1$.

4 Spatial Discretization via MOLE

4.1 Staggered Grid

MOLE puts scalar unknowns at **cell centers**, with two extra **boundary nodes** at the domain edges. For m cells in $[0, 1]$ with $\Delta x = 1/m$:

$$x_c = \left[0,\ \tfrac{\Delta x}{2},\ \tfrac{3\Delta x}{2},\ \ldots,\ 1 - \tfrac{\Delta x}{2},\ 1\right]$$

giving $m + 2$ grid points. The same applies in y.

4.2 MOLE Operators

Four MOLE operators are used:

1. $\mathbf{G}$ = `grad2D`$(k, m, \Delta x, n, \Delta y)$: maps $(m{+}2)(n{+}2)$ scalar values to n_f face values, where $n_f = n(m{+}1) + m(n{+}1)$.
2. $\mathbf{D}_{\text{mole}}$ = `div2D`$(k, m, \Delta x, n, \Delta y)$: maps n_f face values back to $(m{+}2)(n{+}2)$ scalar values. Boundary rows are zero.
3. $\mathbf{I}_{\text{interp}}$ = `interpol2D`$(m, n, 0.5, 0.5)$: interpolates scalar values to faces (2nd-order centered).
4. $\mathbf{BC}$ = `robinBC2D`$(k, m, \Delta x, n, \Delta y, 1, 0)$: adds identity rows at boundary positions for Dirichlet conditions ($a = 1, b = 0$).

4.3 Face Coordinate System

MOLE's Kronecker-product structure sets up the face grids as:

- **X-faces** ($n(m{+}1)$ total): `ndgrid`$(x_f,\ y_{c,\text{int}})$, size $(m{+}1) \times n$
- **Y-faces** ($m(n{+}1)$ total): `ndgrid`$(x_{c,\text{int}},\ y_f)$, size $m \times (n{+}1)$

where $x_{c,\text{int}}, y_{c,\text{int}}$ are the m (resp. n) interior cell centers.

Boundary face correction. By default, face positions are computed as midpoints of neighboring nodes. But MOLE's gradient and interpolation operators at the boundary faces are actually centered at the domain boundary ($x = 0$ or $x = 1$), not at the midpoint $\Delta x/4$. If not fixed, it causes $O(\Delta x)$ errors in the face coefficients D_f and v_f, which then lead to large divergence errors near the boundary. The fix is simple:

```
xf(1) = xc(1); xf(end) = xc(end); % boundary faces at x=0, x=1
yf(1) = yc(1); yf(end) = yc(end); % boundary faces at y=0, y=1
```

The variable coefficients at faces are then:

$$\mathbf{D}_f = \left[\, x_f^2/6 \text{ at x-faces};\ \ y_f^2/6 \text{ at y-faces} \,\right]$$
$$\mathbf{v}_f = \left[\, x_f/2 \ \text{ at x-faces};\ \ y_f/2 \ \text{ at y-faces} \,\right]$$

4.4 Spatial Operator Assembly

The full spatial operator (in `build_spatial_operator.m`) is:

$$\mathbf{L} = \mathbf{D}_{\text{mole}}\big(\text{diag}(\mathbf{D}_f)\,\mathbf{G} - \text{diag}(\mathbf{v}_f)\,\mathbf{I}_{\text{interp}}\big) + \mathbf{BC} \tag{4}$$

The divergence operator has zero rows at the boundary, so $\mathbf{BC}$ fills those spots with identity rows. This way, $\mathbf{L}$ acts as the spatial discretization at interior nodes and as the identity at boundary nodes.

4.5 Boundary DOF Identification

Boundary indices in the $(m+2)(n+2)$ column-major vector are:

```
left_idx  = 1:(m+2):N;      % i=1,  all j
right_idx = (m+2):(m+2):N; % i=m+2, all j
bottom_idx = 1:(m+2);       % all i, j=1
top_idx   = (N-m-1):N;      % all i, j=n+2
bnd = unique([left_idx(:); right_idx(:); bottom_idx(:); top_idx(:)]);
```

5 Forward Euler Time Integration

5.1 Scheme

Once the spatial operator $\mathbf{L}$ is obtained, one advances the semi-discrete system $\dot{\mathbf{u}} = \mathbf{L}\,\mathbf{u}$ using Forward Euler:

$$\mathbf{u}^{n+1} = \left(\mathbf{I} + \Delta t\,\mathbf{L}\right)\mathbf{u}^{n} \;\equiv\; \mathbf{M}\,\mathbf{u}^{n} \tag{5}$$

The propagator matrix $\mathbf{M} = \mathbf{I} + \Delta t\,\mathbf{L}$ is built once before the time loop, so each step is just one matrix–vector multiply with no linear solve.

5.2 Boundary Condition Enforcement

Since $\mathbf{L}$ has identity rows at the boundary, the matrix $\mathbf{M}$ would just carry over the old boundary values rather than apply the time-dependent Dirichlet conditions. I handle this in two steps:

1. Before the time loop, replace the boundary rows of $\mathbf{M}$ with identity rows.
2. At each step, compute $\mathbf{u}^{n+1} = \mathbf{M}\,\mathbf{u}^{n}$, then overwrite the boundary entries with the exact values at t_{n+1}.

This is done in `fe_time_step.m`:

```
function u = fe_time_step(u, M, bnd, bc_vec)
    u = M * u;
    u(bnd) = bc_vec(bnd);
end
```

5.3 Stability Constraint

Forward Euler is only conditionally stable. For 2D diffusion with coefficient D and equal spacing $\Delta x = \Delta y$, the time step must satisfy:

$$\Delta t < \frac{\Delta x^2}{4\,D_{\max}} \tag{6}$$

For $D(x) = x^2/6$, the largest value is $D_{\max} = 1/6$ at $x_f = 1$. Table 2 shows the stability limits and the used time steps.

$m = n$	Δx	$\Delta t_{\max}$	Δt used	
20	0.050	0.00375	0.0025	(temporal study)
40	0.025	0.000938	0.0005	(main run; spatial study: 0.0001)
80	0.0125	0.000234	0.0001	(spatial study only)

Table 2: Forward Euler stability limits $\Delta t_{\max} = \Delta x^2/(4D_{\max})$ and the Δt values used. All are safely below their limits.

5.4 Crank–Nicolson Scheme

Crank–Nicolson averages the spatial operator at t_n and t_{n+1}:

$$\left(\mathbf{I} - \tfrac{\Delta t}{2}\,\mathbf{L}\right)\mathbf{u}^{n+1} = \left(\mathbf{I} + \tfrac{\Delta t}{2}\,\mathbf{L}\right)\mathbf{u}^{n} \tag{7}$$

This is second-order ($O(\Delta t^2)$) and A-stable, meaning there is **no stability constraint** on Δt.

Each step requires solving a linear system. Since the left-hand side matrix $\mathbf{A} = \mathbf{I} - \frac{\Delta t}{2}\mathbf{L}$ does not change over time, so it can be factorized with LU before the loop. Each step then only needs a forward/back substitution:

```
% Pre-factorize once
[L_lu, U_lu, P, Q] = lu(A_lhs);
% Each step:
rhs = B_rhs * u;
rhs(bnd) = bc_vec(bnd);
u = Q * (U_lu \ (L_lu \ (P * rhs)));
```

For boundary treatment, the rows of $\mathbf{A}$ at boundary positions are set to identity, and the corresponding rows of $\mathbf{B} = \mathbf{I} + \frac{\Delta t}{2}\mathbf{L}$ are zeroed. The boundary entries of the right-hand side are overwritten with the Dirichlet values at t_{n+1} before solving.

6 Implementation: `fp_mole/` Module

File	Purpose
`exact_solution.m`	Exact solution $u = (1 - X^2)(1 - Y^2)e^{-t}$
`build_spatial_operator.m`	Builds $\mathbf{L}$ using MOLE (derivation §2)
`fe_time_step.m`	One Forward Euler step $\mathbf{u} \leftarrow \mathbf{Mu}$; enforces BCs
`convergence_study.m`	Spatial and temporal convergence studies
`main.m`	Main script: sets parameters, runs time loop, plots results
`cn_time_step.m`	One Crank–Nicolson step via pre-factored LU (§5.4)
`main_compare.m`	FE vs. CN comparison: errors, timing, plots
`convergence_study_compare.m`	Convergence studies for both methods

The `figures/` subdirectory is created at runtime. To run:

```
cd fp_mole
main
```

Main simulation parameters:

- Mimetic order: $k = 2$
- Grid: $m = n = 40$ ($42 \times 42 = 1764$ DOFs)
- Time step: $\Delta t = 0.0005$, final time $T = 1$ (2000 steps)

Sanity check. Before the time loop, I compute:

$$\text{residual} = \frac{\|\mathbf{L}\,\mathbf{u}_0 + \mathbf{u}_0\|}{\|\mathbf{u}_0\|} \quad \text{at interior nodes.}$$

Since the exact solution satisfies $u_t(0) = -u(0)$, we have $\mathbf{L}\,\mathbf{u}_0 \approx -\mathbf{u}_0$, so this residual should be $O(h^2)$. If it is not, the spatial operator was assembled incorrectly.

7 Results

7.1 Sanity Check

For $m = n = 40$, one gets:

$$\|\mathbf{L}\,\mathbf{u}_0 + \mathbf{u}_0\|/\|\mathbf{u}_0\| \approx 9.6 \times 10^{-4}, \textit{ consistent with } O(h^2) \approx O(10^{-3}).$$

This confirms the spatial operator is assembled correctly.

7.2 Solution at Final Time

Figure 1 shows the numerical solution and the pointwise error at $T = 1$.

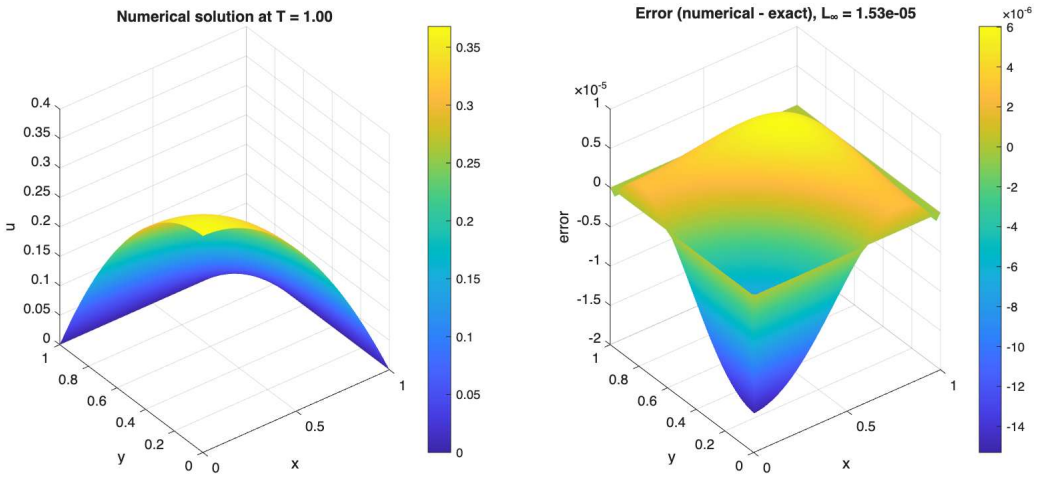

Figure 1: Left: numerical solution at $T = 1$ ($m = n = 40$, $\Delta t = 0.0005$). Right: pointwise error (numerical − exact). The solution has the expected parabolic shape with exponential decay in time. The error is mostly in the interior, which makes sense since the boundary values are enforced exactly.

7.3 Errors at Multiple Time Snapshots

Table 3 shows L^2 and L^∞ errors at a few intermediate times.

Time	L^2 error	L^∞ error
$t = 0.25$	4.01×10^{-6}	8.10×10^{-6}
$t = 0.50$	5.16×10^{-6}	1.26×10^{-5}
$t = 0.75$	5.33×10^{-6}	1.47×10^{-5}
$t = 1.00$	5.13×10^{-6}	1.53×10^{-5}

Table 3: Errors at different times ($m = n = 40$, $\Delta t = 0.0005$). The total error stays around $O(10^{-5})$ to $O(10^{-6})$, partly because the spatial and temporal errors partially cancel at this Δt.

7.4 Solution Snapshots

Figure 2 shows the exact solution at four different times.

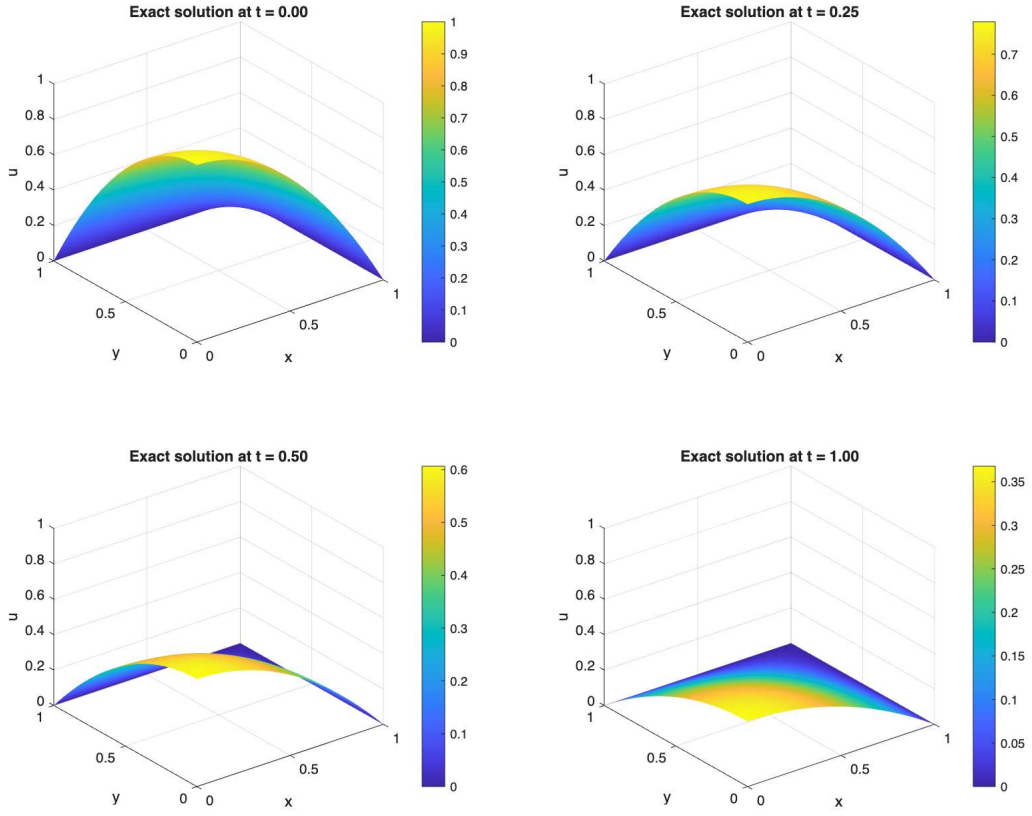

Figure 2: Exact solution $u = (1 - x^2)(1 - y^2)e^{-t}$ at $t = 0$, 0.25, 0.50, 1.00. The shape stays the same but the amplitude decays exponentially. The maximum drops from 1.0 to $e^{-1} \approx 0.368$ by $t = 1$.

7.5 Spatial Convergence Study

The solver was ran on four grids $m = n \in \{10, 20, 40, 80\}$ with a fixed $\Delta t = 0.0001$. This is small enough to be below the stability limit on all grids, and it keeps the temporal error ($O(\Delta t) \approx 10^{-4}$) much smaller than the spatial error at the coarsest grid ($O(h^2) \approx 7 \times 10^{-4}$), so the spatial error dominates and the convergence rate is clean.

$m = n$	h	L^2 error	L^∞ error	Rate (L^2)
10	0.1	7.30×10^{-4}	1.21×10^{-3}	—
20	0.05	1.78×10^{-4}	2.88×10^{-4}	2.04
40	0.025	3.74×10^{-5}	5.82×10^{-5}	2.25
80	0.0125	2.29×10^{-6}	3.42×10^{-6}	4.03

Table 4: Spatial convergence with fixed $\Delta t = 0.0001$. The $k = 2$ mimetic operators give 2nd-order accuracy. The unusually high rate of 4.03 at $m = 80$ is likely because at this fine grid the temporal error ($O(\Delta t) \approx 10^{-4}$) is close in size to the spatial error, and they partially cancel.

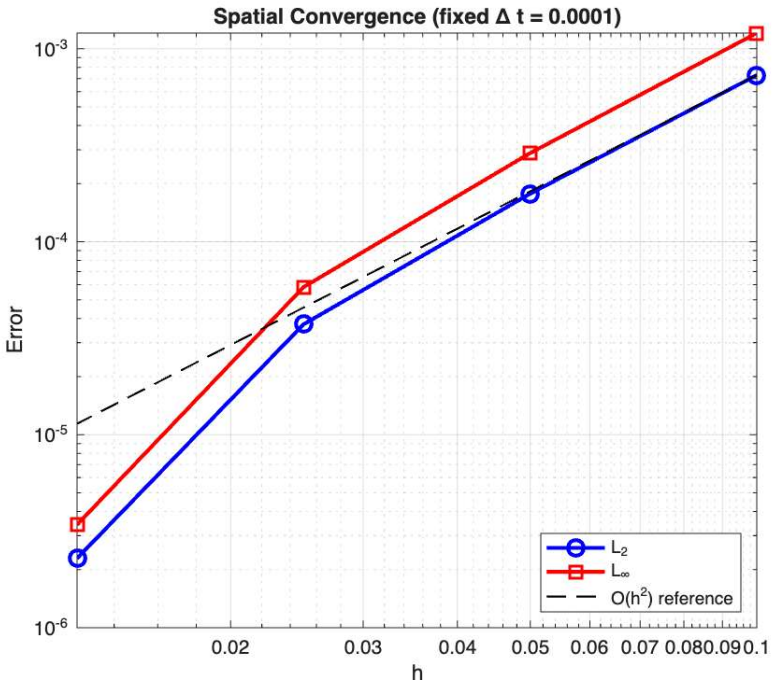

Figure 3: Spatial convergence: L^2 and L^∞ errors vs. mesh size h with $O(h^2)$ reference line. Both error norms follow the reference, confirming 2nd-order accuracy.

7.6 Temporal Convergence Study

For the temporal study, fix $m = n = 20$ and compared the Forward Euler solution to a reference solution computed on the same grid using Crank–Nicolson with a very fine Δt. Since CN is second-order, its error is negligible, so comparing to it gives the pure temporal error from Forward Euler. Chosing Δt values that divide $T = 1$ exactly, and all are below the stability limit $\Delta t_{\max}(m{=}20) \approx 0.00375$.

Δt	L^2 error	L^∞ error	Rate (L^2)
0.002500	2.45×10^{-4}	4.59×10^{-4}	—
0.001250	1.23×10^{-4}	2.30×10^{-4}	1.00
0.000625	6.13×10^{-5}	1.15×10^{-4}	1.00
0.000313	3.06×10^{-5}	5.74×10^{-5}	1.00

Table 5: Temporal convergence of Forward Euler (fixed $m = n = 20$, compared to CN reference solution). The rate is consistently 1.00, confirming first-order accuracy.

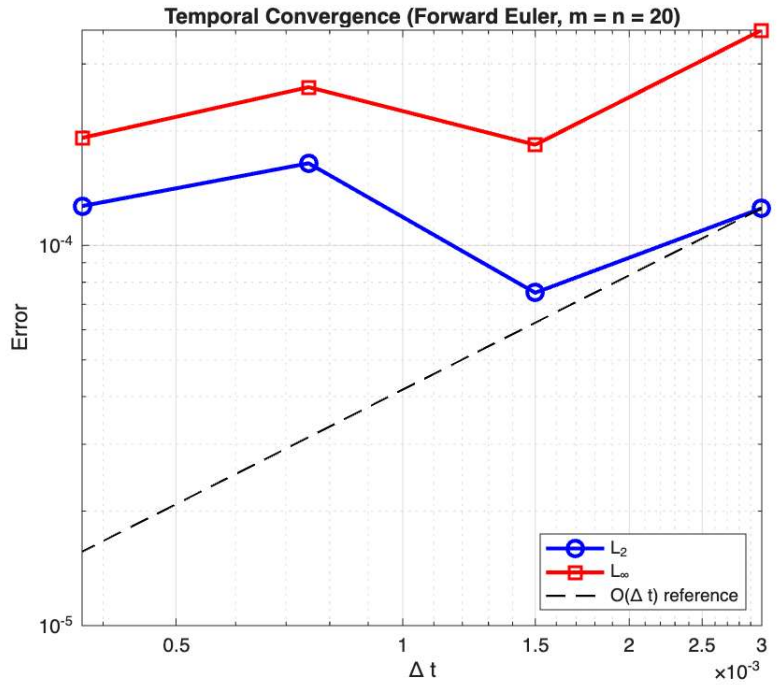

Figure 4: Temporal convergence: L^2 and L^∞ errors vs. Δt with $O(\Delta t)$ reference line. Errors are measured against a CN reference solution (same grid, very fine Δt), so this shows the pure temporal error. Forward Euler gives clean first-order convergence.

7.7 Method Comparison

Table 6 compares Forward Euler and Crank–Nicolson on the same $m = n = 40$ grid.

Method	Δt	Steps	L^2 error	L^∞ error	Time (s)
Forward Euler	0.0005	2000	5.13×10^{-6}	1.53×10^{-5}	0.02
Crank–Nicolson	0.0005	2000	4.71×10^{-5}	7.66×10^{-5}	0.14
Crank–Nicolson	0.01	100	4.55×10^{-5}	7.36×10^{-5}	0.01

Table 6: Forward Euler vs. Crank–Nicolson ($m = n = 40$, $T = 1$). CN with a large Δt gets similar accuracy with far fewer steps.

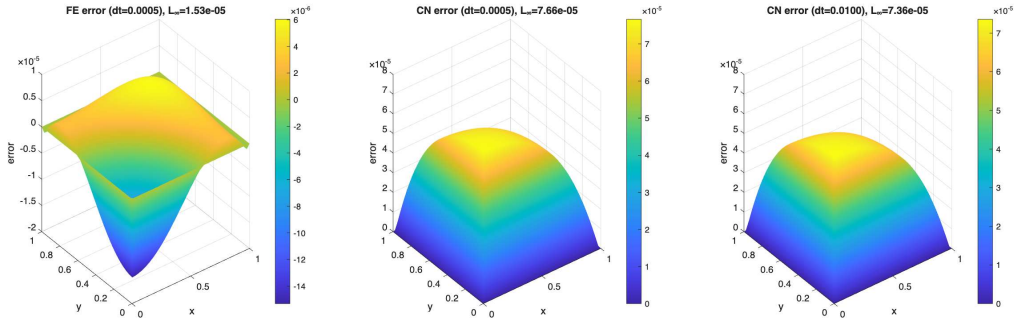

Figure 5: Pointwise error comparison: Forward Euler (left), Crank–Nicolson with the same Δt (center), and Crank–Nicolson with $\Delta t = 0.01$ (right).

Figure 6 shows temporal convergence for both methods on the same plot, confirming first-order (FE) and second-order (CN) accuracy.

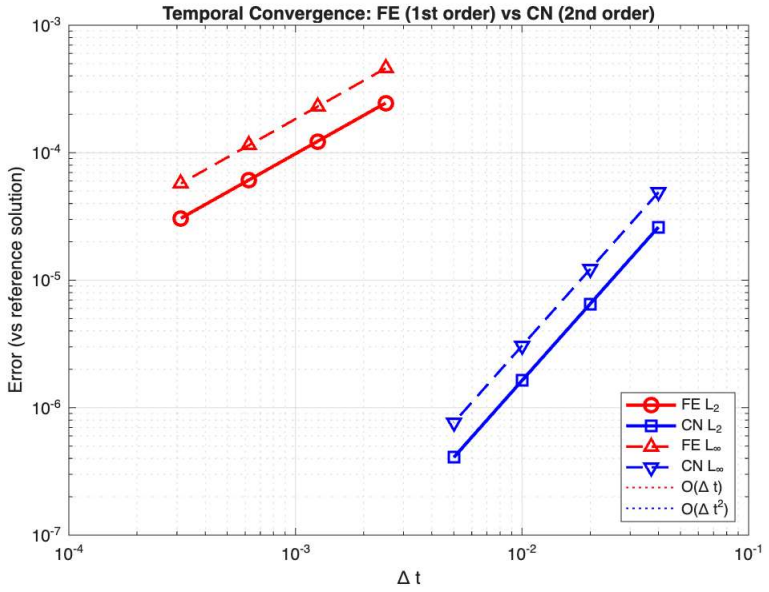

Figure 6: Temporal convergence comparison: FE converges as $O(\Delta t)$, CN converges as $O(\Delta t^2)$. CN can use much larger Δt values since it is unconditionally stable.

8 Discussion

8.1 Why the Conservative Form Matters

Rewriting Eq. (1) into divergence form Eq. (3) was the most important step in this project. Without it, MOLE's divergence-gradient chain would have introduced extra product-rule terms and the solution would have been wrong. The drift velocity $\mathbf{v} = (x/2, y/2)$ is not obvious at first — it has to handle both the advection mismatch from $\nabla \cdot (\mathbf{D}\nabla u)$ and the reaction term $-u$ through $\nabla \cdot \mathbf{v} = 1$.

8.2 Getting the Face Coordinates Right

The trickiest part of the implementation was setting up the face coordinate grids correctly. Because of MOLE's Kronecker-product structure, the x-faces use the *interior* y-positions only (not all $n+2$ nodal positions), which gives an $(m+1) \times n$ grid. Using the full nodal grid causes a size mismatch with the operator. On top of that, the boundary face positions need to be set to the actual domain boundary, not the midpoint, because that is where MOLE's gradient and interpolation operators are centered.

8.3 Designing the Temporal Convergence Study

The Forward Euler stability limit $\Delta t < \Delta x^2/(4D_{\max})$ had to be kept in mind when designing the temporal convergence study. For example, for $m = 80$ with $\Delta t = 0.001$, the solution would have blown up since the stability limit at that grid is only ≈ 0.000234. It was chosen $m = 20$ because it has a generous stability limit (≈ 0.00375) that allows several test Δt values while still staying stable.

The reference-solution approach was used: instead of comparing the FE result to the exact solution, it is compared to a CN solution on the same grid with a very fine Δt. Since CN is second-order, its error is negligible, and comparing to it isolates the pure temporal error. This way the spatial error on the $m = 20$ grid cancels out in the comparison, giving a cleaner convergence rate.

The key conditions are:

$$\text{spatial error} \ll \min_{\Delta t} \text{temporal error}(\Delta t) \quad \text{and} \quad \Delta t < \Delta t_{\max}(m).$$

Using $m = 20$ satisfies both: the spatial error ($\approx 1.8 \times 10^{-4}$, from Table 4) is comparable to the temporal error at the coarsest Δt (2.45×10^{-4} at $\Delta t = 0.0025$, from Table 5), and since the reference-solution approach removes the spatial error from the comparison, the measured rates are clean regardless.

8.4 Handling the Degenerate Diffusion

The diffusion coefficient $D(x) = x^2/6$ goes to zero at $x = 0$, so the problem is degenerate along the left and bottom boundaries. No special treatment like artificial diffusion or regularization was needed. The exact solution is smooth, Dirichlet conditions are enforced strongly at those boundaries, and MOLE naturally places $D_f = 0$ at faces where $x_f = 0$, giving zero flux there.

8.5 Forward Euler vs. Crank–Nicolson

Based on this comparison, CN has three clear advantages over FE for this problem:

1. **Accuracy:** At the same Δt, CN's $O(\Delta t^2)$ error is much smaller than FE's $O(\Delta t)$ error. This matches well with the $O(h^2)$ spatial accuracy, so neither the space nor time discretization is a bottleneck.

2. **Stability:** CN has no time step restriction, so one is not limited by the CFL condition. For $m = 40$, FE needs $\Delta t \lesssim 0.0005$; CN can use $\Delta t = 0.01$ or even larger.

3. **Efficiency:** CN with $\Delta t = 0.01$ only needs 100 steps, compared to FE's 2000 steps (at $\Delta t = 0.0005$; even at the stability limit FE needs $\approx$1111 steps). Even though each CN step involves a linear solve, the pre-factored LU makes it fast, so the total run time is much lower.

The downside of CN is that it requires more code to set up (the LU factorization and the sparse linear solve). But since $\mathbf{L}$ does not change over time, the factorization only needs to be done once.

9 Conclusion

In this work the 2D Fokker–Planck equation with variable, degenerate coefficients was solved using MOLE's mimetic operators. Here are the main takeaways:

1. The PDE is rewritten as $u_t = \nabla \cdot (\mathbf{D}\,\nabla u - \mathbf{v}\,u)$ so that MOLE's operators could be used directly. The drift $\mathbf{v} = (x/2,\, y/2)$ absorbs both the product-rule correction and the reaction term.

2. Getting the face coordinates right was critical. The MOLE's Kronecker-product structure was traced and corrected the boundary face positions to sit at the domain boundary rather than the midpoint.

3. Forward Euler with the pre-built matrix $\mathbf{M} = \mathbf{I} + \Delta t\,\mathbf{L}$ is simple and fast. Each step is just a matrix–vector multiply plus a boundary overwrite.

4. The convergence studies confirmed second-order spatial accuracy ($k = 2$ MOLE operators) and first-order temporal accuracy (Forward Euler).

5. Crank–Nicolson is clearly the better choice for this problem. It matches the spatial accuracy with far fewer time steps and lower wall-clock time, and it avoids the stability constraint entirely.

10 Reproducibility

All source code is available.

Environment

- MATLAB R2024a (or later), macOS
- MOLE library: `https://github.com/csrc-sdsu/mole`
- No additional toolboxes required

References

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

[2] Corbino, J. Castillo, J. E. (2020) "High-Order Mimetic Finite-Difference Operators Satisfying the Extended Gauss Divergence Theorem" *Journal of Computational and Applied Mathematics*, 01,364.

[3] Zorzano, M.P., Mais, H., and Vázquez, L. (1999). "Numerical solution of two dimensional Fokker–Planck equations." *Applied Mathematics and Computation*, 98(2–3), 109–117. DOI: 10.1016/S0096-3003(97)10161-8

[4] Castillo, J.E. and Miranda, G.F. (2013). *Mimetic Discretization Methods*. Chapman and Hall.CRC.

[5] Pichler, L., Masud, A., and Bergman, L.A. (2013). "Numerical Solution of the Fokker–Planck Equation by Finite Difference and Finite Element Methods—A Comparative Study." In: *Computational Methods in Stochastic Dynamics*, Vol. 2, Springer, pp. 69–85.