PDE Analysis of Pattern Formation in Biological Systems defined by a chemoattractant and a stimulant

Anas Mohamed Abdelaziz Elsheikh*, Ahmed Abdelghaffar Mostafa Shehab†, Sherif Mohamed Mahmoud Elgendy‡, Mostafa Ashraf Mostafa Abdelrahim§, Seif Samaan Ghaly Atia¶, Ziad Ramadan Mohamed Mohamed‖, Mohamed Ashraf Abdelhamid Hagras**, and Mazen Atef Awad Atlam††
Department of Systems and Biomedical Engineering, Faculty of Engineering, Cairo University, Giza, Egypt
Email: *anas.elshaikh@student.cu.edu.eg, †ahmed.shehab@student.cu.edu.eg, ‡sherif.elgendy@student.cu.edu.eg, §mostafa.abdelrahim@student.cu.edu.eg, ¶seif.atia@student.cu.edu.eg, ‖ziad.ramadan@student.cu.edu.eg, **mohamed.hagras@student.cu.edu.eg, ††mazen.atlam04@eng-st.cu.edu.eg

***Abstract*—Pattern formation in biological systems through chemoattractant-cell interactions represents a fundamental mechanism underlying numerous developmental and physiological processes. This paper presents a comprehensive study of partial differential equation (PDE) models describing pattern formation dynamics, focusing on three-PDE system in one spatial dimension. We employ the Method of Lines (MOL) for numerical solutions and compare results with two other numerical approaches and machine learning approach. Our analysis demonstrates the effectiveness of both traditional numerical methods and modern ML techniques in capturing complex spatiotemporal patterns. The study reveals critical parameter regimes for pattern formation and provides insights into the underlying biological mechanisms. Simulation results show good agreement between numerical and ML solutions, with computational efficiency gains observed in the machine learning approach.**

***Index Terms*—pattern formation, partial differential equations, chemoattractant, Method of Lines, machine learning, py-pde, numerical simulation, finite difference method, finite volume method, spatiotemporal dynamics, chemotaxis, biological modeling, PDE analysis, computational biology, neural operators, reaction-diffusion systems**

I. Introduction

Pattern formation in biological systems is a ubiquitous phenomenon that underlies many critical processes, from embryonic development to wound healing and tumor growth. The interaction between cells and chemical signals, particularly chemoattractants and stimulants, plays a crucial role in organizing cellular behavior and generating complex spatiotemporal patterns.

Mathematical modeling of these phenomena typically involves systems of partial differential equations that capture the dynamics of cell density, chemical concentrations, and their mutual interactions. The complexity of these models arises from nonlinear coupling terms, diffusion processes, and reaction kinetics that operate across multiple spatial and temporal scales.

This study focuses on PDE models describing how cells interact with chemoattractants and stimulants in one-dimensional space. We investigate three-PDE system, starting from basic diffusion equations and progressively incorporating nonlinear terms that capture essential biological mechanisms such as chemotaxis, cell proliferation, and chemical degradation.

The primary objectives of this work are: (1) to develop and analyze comprehensive PDE models for chemoattractant-cell interactions, (2) to implement robust numerical solution methods using the Method of Lines, (3) to explore machine learning approaches as alternative solution strategies, and (4) to provide insights into parameter regimes that promote pattern formation.

II. Literature Review

The mathematical study of biological pattern formation has evolved significantly since Turing's introduction of reaction-diffusion models [1]. Key contributions include Murray's extensions incorporating chemotactic behavior [2] and Keller–Segel models that describe cell aggregation via self-produced chemoattractants [3]. Such models now routinely include additional biological processes like proliferation, death, and multi-chemical interactions.

On the computational side, the Method of Lines (MOL) has remained a cornerstone for high-accuracy PDE simulations with adaptive time integration [4]. In parallel, high-order schemes and adaptive mesh refinement methods have been developed to resolve sharp pattern boundaries efficiently.

A new frontier has emerged at the intersection of machine learning and PDEs. Physics-Informed Neural Networks (PINNs) and related frameworks integrate governing equations directly into neural network training, offering mesh-free representations capable of modeling complex boundary conditions [5], [6]. Operator learning approaches, such as DeepONet and Fourier Neural Operators (FNO), generalize the concept by learning mappings between function spaces, enabling fast, discretization-invariant prediction of PDE solutions across parameter regimes [7], [8].

Hybrid operator-learning frameworks, like deep branch–trunk networks, now come with rigorous error bounds for nonlinear PDEs [9], while Gaussian Process–based architectures (e.g., kernel-weighted corrective-residual models, CoRes) integrate model interpretability and stability into ML-driven PDE solvers [10]. Additionally, reinforcement of domain-decomposition solvers via graph convolutional neural networks (GCNNs) has shown promising scalability improvements on both structured and unstructured grids [11].

These developments suggest a burgeoning toolkit for PDE simulation that combines the numerical rigor of classical discretization with the flexibility and efficiency of data-driven solvers. In particular, the ability of neural operators to generalize across resolutions and parameter regimes makes them highly suitable for modeling complex spatiotemporal biological patterns.

III. Mathematical Model

A. PDE Model Description

A three-component system describing the interaction between cell density $u_1(x,t)$ and chemoattractant concentration $u_2(x,t)$ in one spatial dimension extended to include a stimulant $u_3(x,t)$:

$$\frac{\partial u_1}{\partial t} = D_1\nabla^2 u_1 - \nabla\left[\frac{k_1 u_1}{(k_2+u_2)^2}\nabla u_2\right] + k_3 u_1\left(\frac{k_4 u_3^2}{k_9+u_3^2} - u_1\right) \quad (1)$$

$$\frac{\partial u_2}{\partial t} = D_2\nabla^2 u_2 + k_5 u_3\left(\frac{u_1^2}{k_6+u_1^2} - k_7 u_1 u_2\right) \quad (2)$$

$$\frac{\partial u_3}{\partial t} = D_3\nabla^2 u_3 - k_8 u_1\left(\frac{u_3^2}{k_9+u_3^2}\right) \quad (3)$$

This extended model allows for more complex interactions, including competition between different chemical signals and their differential effects on cell behavior.

B. Initial and Boundary Conditions

We consider no-flux (Neumann) boundary conditions for all components:

$$\left.\frac{\partial u_1}{\partial x}\right|_{x=0,L} = \left.\frac{\partial u_2}{\partial x}\right|_{x=0,L} = \left.\frac{\partial u_3}{\partial x}\right|_{x=0,L} = 0 \quad (4)$$

and Gaussian profiles centered at $t=0$:

$$u_1(x,0) = 10^8 e^{-5x^2} \text{ cells/ml} \quad (5)$$

$$u_2(x,0) = 5\times 10^{-6} e^{-5x^2} \text{ M} \quad (6)$$

$$u_3(x,0) = 10^{-3} e^{-5x^2} \text{ M} \quad (7)$$

IV. Numerical Solution Methods

A. Finite Difference Method

The Finite Difference Method solves PDEs by replacing all continuous derivatives with discrete approximations on a grid. This transforms the PDE system into a set of algebraic equations that can be solved iteratively.

1) Step 1: Spatio-Temporal Grid Discretization: We discretize both space and time. The spatial domain $x \in [0, L]$ is divided into $N+1$ points $x_j = j\cdot\Delta x$, and time is divided into steps $t_n = n\cdot\Delta t$. The solution at a grid point (x_j, t_n) is denoted $u_j^n \approx u(x_j, t_n)$.

2) Step 2: Finite Difference Approximation: We use the Forward-Time Central-Space (FTCS) explicit scheme.

- **Time Derivative** (Forward Difference):

$$\left.\frac{\partial u_k}{\partial t}\right| j,n \approx \frac{uk, j^{n+1} - u_{k,j}^n}{\Delta t} \quad (8)$$

- **Spatial Derivative** (Central Difference):

$$\left.\frac{\partial^2 u_k}{\partial x^2}\right| j,n \approx \frac{uk, j+1^n - 2u_{k,j}^n + u_{k,j-1}^n}{(\Delta x)^2} \quad (9)$$

3) Step 3 : The Explicit FDM Update Equation: Substituting these into the PDE and rearranging gives a direct update rule to find the solution at the next time step, $n+1$, using only known values from the current step, n:

$$u_{k,j}^{n+1} = u_{k,j}^n + \frac{D_k\Delta t}{(\Delta x)^2}(u_{k,j+1}^n - 2u_{k,j}^n + u_{k,j-1}^n) + \Delta t\cdot f_k^n \quad (10)$$

This equation is applied at each grid point to "march" the solution forward in time.

4) Step 4 : Boundary Condtions: At the boundaries ($j=0$ and $j=N$), we use "ghost points" to enforce the no-flux condition. For the left boundary ($j=0$), $\frac{\partial u_k}{\partial x} = 0$ implies $u_{k,-1}^n = u_{k,1}^n$. Substituting this into the update equation gives:

$$u_{k,0}^{n+1} = u_{k,0}^n + \frac{D_k\Delta t}{(\Delta x)^2}(2u_{k,1}^n - 2u_{k,0}^n) + \Delta t\cdot f_k^n \quad (11)$$

A similar equation is derived for the right boundary ($j=N$).

B. Finite Volume Method

The Finite Volume Method solves PDEs by discretizing the domain into control volumes and enforcing conservation laws over each volume. This transforms the PDE system into flux balance equations that preserve physical quantities.

1) Step 1: Domain Discretization and Grid Definition: We partition the spatial domain $x\in[0,L]$ into N control volumes (cells) centered at x_j with width Δx. The cell faces are located at $x_{j\pm1/2}$. The solution is represented as cell averages $\bar{u}_{k,j}^n \approx \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} u_k(x,t_n)dx$.

2) Step 2: Integral Formulation and Flux Approximation: We integrate the PDE over each control volume and apply the divergence theorem:

$$\frac{d\bar{u}_{k,j}}{dt} = \frac{1}{\Delta x}\left[F_{k,j+1/2} - F_{k,j-1/2}\right] + \bar{f}_{k,j} \quad (12)$$

where fluxes F are approximated using:

- **Diffusive flux**: Central difference at faces

$$F_{k,j+1/2}^{\text{diff}} = D_k\frac{u_{k,j+1} - u_{k,j}}{\Delta x}$$

- **Advective flux**: Upwind scheme for chemotaxis terms

$$F_{k,j+1/2}^{\text{adv}} = \chi u_{k,j}\frac{c_{j+1} - c_j}{\Delta x}$$

3) Step 3: The Semi-Discrete FVM System: Applying the flux approximations yields the ODE system for each cell:

$$\begin{aligned}\frac{d\bar{u}_{k,j}}{dt} =& \frac{D_k}{\Delta x^2}(\bar{u}_{k,j+1} - 2\bar{u}_{k,j} + \bar{u}_{k,j-1}) \\ & - \frac{\chi}{\Delta x^2}\big[u_{k,j}(c_{j+1} - c_j) \\ & - u_{k,j-1}(c_j - c_{j-1})\big] + \bar{f}_{k,j}\end{aligned} \tag{13}$$

4) Step 4: Boundary Condition Implementation: For no-flux boundaries ($j = 0$ and $j = N$):

- Ghost cell values are set to enforce zero flux:

$$u_{k,0} = u_{k,1}, \quad u_{k,N+1} = u_{k,N}$$

- Boundary fluxes become:

$$F_{k,1/2} = 0, \quad F_{k,N+1/2} = 0$$

5) Step 5: Temporal Discretization: The ODE system is solved using:

- **Time integration**: Adaptive BDF2 method
- **Nonlinear handling**: Newton iterations for implicit steps
- **Stability**: CFL condition $\Delta t \leq \frac{\Delta x^2}{2D_{\max}}$

V. Results and Discussion

A. Finite Difference Method

While this report details a direct FDM implementation, it is crucial to compare it with the Method of Lines (MOL), another common strategy for solving time-dependent PDEs.

TABLE I: Relative errors at selected spatial points

x (cm)	u_1 error (%)	u_2 error (%)	u_3 error (%)
0.0	0.05	2.98	0.01
0.4	0.03	2.64	0.02
0.8	0.13	1.93	0.03

TABLE II: Performance & Statistics

Metric	FDM	Book Implementation
Runtime in seconds (real)	1.881	1.826
Problem Size (no. of Equations)	153 ODEs	153 ODEs
Time per Equations (s/equ)	0.0123	0.0119
Function Evaluations	1052	1052

B. Finite Volume Method

The FVM solution demonstrates excellent conservation properties, crucial for mass-preserving systems like chemotaxis models. The MOL provides greater flexibility in time stepping and easier implementation of complex boundary conditions. For the Patlak-Keller-Segel model:

- FVM better captures sharp gradients and maintains physical bounds
- MOL allows efficient handling of stiff systems through adaptive ODE solvers
- Both methods show good agreement in smooth regions of the solution

TABLE III: Relative Errors at Selected Spatial Points

x (cm)	u_1 Error (%)	u_2 Error (%)	u_3 Error (%)
0.0	0.07	2.99	0.02
0.4	0.25	2.77	0.12
0.8	0.59	1.61	0.17

TABLE IV: Performance Comparison Between FVM and Reference Implementation

Metric	FVM	Reference
Runtime (s)	1.132	1.826
Problem Size	153 ODEs	153 ODEs
Time per Equation ($\mathrm{s\,equ^{-1}}$)	0.0074	0.0119
Function Evaluations	945	1052

C. Machine Learning Solution Results

The PDE system was solved using Python's `py-pde` package, implementing finite difference discretization with 51 spatial points over a 1D domain $x \in [0, 1]$ cm. Key implementation aspects include:

1) Solution Characteristics:

- **Initial Conditions**: Gaussian profiles centered at $t = 0$:

$$u_1(x, 0) = 10^8 e^{-5x^2} \text{ cells/ml} \quad \text{(eq. 5)}$$

$$u_2(x, 0) = 5 \times 10^{-6} e^{-5x^2} \text{ M} \quad \text{(eq. 6)}$$

$$u_3(x, 0) = 10^{-3} e^{-5x^2} \text{ M} \quad \text{(eq. 7)}$$

- **Boundary Conditions**: Zero-flux Neumann ($\partial u_i/\partial x = 0$ at $x = 0, 1$)
- **Temporal Resolution**: Adaptive BDF solver with output at 0.5-hour intervals

2) Accuracy Validation: Table 1 shows relative errors versus reference solutions at $t = 5$ hours:

TABLE V: Relative errors at selected spatial points

x (cm)	u_1 error (%)	u_2 error (%)	u_3 error (%)
0.0	0.43	0.65	0.18
0.4	0.08	0.45	0.02
0.8	0.46	0.72	0.18

3) Time Metrics Validation: Table 2 shows the performance and statistics of the learning-based solution:

TABLE VI: Performance & Statistics

Metric	learning-based	Book Implementation
Runtime in seconds (real)	6.446	1.826
Problem Size (no. of Equations)	153 ODEs	153 ODEs
Time per Equations (s/equ)	0.0452	0.0119
Function Evaluations	3717	1052

D. Summary Discussion

The comprehensive comparison of four numerical approaches—Finite Difference Method (FDM), Finite Volume Method (FVM), Method of Lines (MOL), and Machine Learning solver (py-pde)—reveals significant performance differences across multiple dimensions. FVM demonstrated superior computational efficiency with the fastest runtime (1.132

seconds), lowest time per equation (0.0074 s/equ), and fewest function evaluations (945), outperforming both FDM (1.881 seconds, 1052 evaluations) and MOL (1.826 seconds, 1052 evaluations). Conversely, the ML approach showed substantially higher computational overhead (6.446 seconds) and required significantly more function evaluations (3717), reflecting current limitations in pure ML implementations for PDE solving. Regarding solution accuracy, the ML solver achieved the smallest relative errors across all variables (0.02-0.72%), followed by FDM (0.01-2.98%) and FVM (0.02-2.99%), suggesting ML's potential for high-precision applications where computational cost is secondary. The FVM exhibited excellent mass conservation properties critical for biological systems, while its low function evaluation count indicates efficient computation. For handling complex boundary conditions, MOL provided greater flexibility through its adaptive ODE solvers, whereas FDM offered the simplest implementation pathway for rapid prototyping. The ML approach demonstrated promising pattern-capturing capabilities without explicit discretization, though its current efficiency limitations and high function evaluation count necessitate further optimization. Regarding solution smoothness, FVM produced the most physically plausible profiles in regions with sharp gradients, while ML showed slight oscillations near boundaries. For large-scale parameter studies, FVM's efficiency and low function evaluation requirement make it particularly suitable, whereas ML's accuracy advantages might justify its cost in critical validation scenarios despite higher computational demands. The MOL implementation served as a reliable reference solution, balancing reasonable speed with good accuracy and moderate function evaluations. FDM's explicit formulation showed limitations in handling stiff terms, requiring stricter stability constraints than the semi-implicit MOL. The ML solver's architecture showed potential for generalization across parameter regimes, though this study focused on single parameter sets and revealed high computational costs. Future work should investigate hybrid FVM-ML approaches where FVM handles bulk simulations and ML refines critical regions to balance efficiency and accuracy. Overall, method selection depends on application priorities: FVM for efficient conservation-critical systems, ML for high-accuracy requirements, MOL for stiff systems with complex boundaries, and FDM for educational purposes or rapid testing.

VI. Future Work and Improvements

A. Suggestions for Improvements

1) Algorithmic Enhancements:

- **Adaptive Mesh (h Refinement)**: Increase number of support points in the grid
- **High-Order Time Integration (p Refinement)**: Use 6th order FD approximations instead of 4th order ones

2) Model Extensions:

- **Spatial Dimension Expansion**: Extend to 2D/3D domains for realistic biological pattern simulation
- **Stochastic Terms**: Introduce noise terms for $\partial u_i/\partial t$ to model microenvironment variability
- **Parameter Estimation**: Integrate neural ODEs for data-driven parameter calibration

B. Future Research Directions

1) Deep Learning Integration:

- **Hybrid Solver**: Show the computer microscope photos of real cells to set up better simulations, then use CNN encoder to generate initial conditions from microscopy images
- **Operator Learning**: Train Fourier Neural Operators (FNOs) to predict patterns without solving complex equations
- **Uncertainty Quantification**: Implement Bayesian PINNs for solution confidence intervals

2) Biological Applications:

- **Cancer Research**: Simulate how tumor cells invade healthy tissue
- **Drug Delivery**: revolutionize drug design by simulating how smart medicines navigate biological systems

Acknowledgment

The authors would like to express their sincere gratitude to their course supervisor, Professor Muhammad Rushdi, for his invaluable guidance, encouragement, and support throughout this project, which was conducted as part of the Numerical Methods course at Cairo University.

References

[1] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 237, no. 641, pp. 37–72, 1952.

[2] J. D. Murray, *Mathematical Biology II: Spatial Models and Biomedical Applications*, 3rd ed., ser. Interdisciplinary Applied Mathematics. New York: Springer, 2003.

[3] E. F. Keller and L. A. Segel, "Initiation of slime mold aggregation viewed as an instability," *Journal of Theoretical Biology*, vol. 26, no. 3, pp. 399–415, 1970.

[4] W. E. Schiesser, *The Numerical Method of Lines: Integration of Partial Differential Equations*. San Diego: Academic Press, 1991.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[6] ——, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[7] Y. Yang, "Deeponet for solving nonlinear partial differential equations with physics-informed training," *arXiv preprint arXiv:2410.04344*, 2024.

[8] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *arXiv preprint arXiv:2010.08895*, 2020.

[9] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart, "Learning nonlinear operators with deep neural networks and error analysis," *Journal of Computational Physics*, vol. 449, p. 110768, 2022.

[10] R. Tripathy and I. Bilionis, "Corrective residual modeling for physics-informed deep learning of pdes: Theory and applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 398, p. 115248, 2022.

[11] J. Brandstetter, M. Welling, and T. Kipf, "Graph neural operator for pdes on irregular domains," *arXiv preprint arXiv:2206.11871*, 2022.