# Stem Cell Differentiation

Numerical and Machine Learning Methods for Differential Equations in Biomedical Engineering

Seif Sameh | 91240371, Fahd Ahmed | 91240561, Khadija Zakaria | 91240965, Mona Elkhouly | 91241075, Zaid Nassif | 91240323, Mohamed Elnefary | 91240675, Aya Emad | 91240199, Ahmed Farahat | 91240108, Hana Gamal | 91240843, Manar Saed | 91240785, Ahmed Saker | 91240116,

***Abstract*—This project explores the modeling of gene regulatory networks involved in stem cell differentiation through a system of nonlinear ordinary differential equations (ODEs) describing the interaction between the transcription factors PU.1 and GATA-1. To solve this system, two numerical approaches ,the trapezoidal rule and Radau method, are used to capture the system's dynamics with stability and precision. Additionally, a machine learning model based on Physics Informed Neural Networks (PINNs) is implemented using PyTorch to provide a data-driven solution framework that embeds the ODE structure directly into the learning process. By comparing the numerical and machine learning results, we assess the strengths and limitations of each approach. The numerical methods demonstrate higher accuracy and computational efficiency, while the PINNs model shows potential in learning system behavior from limited data. This comparative study highlights the complementary nature of traditional solvers and neural ODE models, offering insight into future hybrid methods for modeling biological systems.**

***Index Terms*—Stem Cells,PINNs,ODE Model.**

## I. Introduction to The Problem

STEM cells have the ability to continuously divide and differentiate into various cell types, such as muscle or bone. They begin from an initial state and progress through several stages before becoming fully specialized. A key stage in this process is the *progenitor state*, where cells are not yet committed but retain the potential to become different types. The final fate is determined during this stage, regulated by proteins known as *transcription factors*.

Among these, **PU.1** and **GATA-1** are two critical transcription factors that guide the differentiation of specific blood cell lineages. In early progenitor cells, both genes are expressed at low levels. Understanding their interaction is essential, as it dictates the developmental path the cell will follow and the type of cell it will ultimately become. Such insights are key to advancing stem cell-based therapies.

To study this interaction, a $\mathbf{2 \times 2}$ (two equations in two unknowns) nonlinear ODE system is used to model the gene expression dynamics of PU.1 and GATA-1 over time. The concentrations of these two transcription factors are the dependent variables, with time as the independent variable. Both numerical solvers and machine learning approaches are applied to analyze how expression levels evolve and influence cell fate.

## II. LITERATURE REVIEW

Recent research continues to advance modeling stem cell differentiation by integrating computational, mathematical, and learning-based methods. This section surveys key 2024–2025 works directly relevant to our problem: modeling stem cell fate decisions, understanding differentiation dynamics, and applying machine learning within biological systems.

Devlin *et al.* [5] demonstrated that embedding differentiation rules within tissue development simulations enhances structural robustness, even under biological noise, highlighting the role of emergent gene regulation dynamics alongside biochemical gradients which aligns with our ODE-based modeling.

Wenqing Peng *et al.* [1] introduced SPIN-ODE, a PINN framework for stiff chemical systems using a three-stage architecture. By incorporating gradient clipping, layered designs, and modular components, they achieved stable and accurate training in stiff regimes which parallels our use of case-specific network depth, Xavier initialization, and gradient control.

Sarabian *et al.* [7] explored parameter estimation for stem cell models under sparse data conditions. Their results highlight the challenges of fitting dynamic biological systems with limited observations—an issue our study addresses through the flexibility and generalization capacity of PINNs.

Yao *et al.* [8] proposed Neural Control Variates to enhance stability and accuracy when solving stiff ODEs with PINNs. Their method complements our use of gradient control techniques and case-specific architectures for handling stiffness in gene regulatory models.

Etezadi *et al.* [4] demonstrated how deep learning can assist in designing molecules to guide cardiac stem cell differentiation using very limited data. Their approach combines geometric molecular descriptors with neural networks, achieving high predictive accuracy and supporting the relevance of AI in guiding stem cell fates which is an idea that is also reflected in our use of PINNs.

Mircea *et al.* [6] applied PINNs to infer gene regulatory networks from single-cell data, capturing bifurcation behavior linked to differentiation. Their results show that PINNs outperform standard networks in recovering mechanistic parameters which reinforces our approach to modeling bifurcation in hematopoietic differentiation.

Collectively, these works emphasize the importance of combining biological knowledge, computational modeling, and data-driven learning when studying stem cell behavior. Our study contributes to this landscape by comparing numerical solvers and PINNs to simulate the dynamics of key transcription factors involved in hematopoietic lineage commitment.

## III. ODE Model Explanation

To study hematopoietic stem cell differentiation, we use a nonlinear ODE model from the literature that captures the feedback between transcription factors GATA-1 and PU.1, whose self-activation and mutual inhibition guide cell fate decisions.

The ODE system is defined as follows:

$$\frac{d[G]}{dt} = \frac{a_1[G]^n}{\theta_{a1}^n + [G]^n} + \frac{b_1\theta_{b1}^m}{\theta_{b1}^m + [G]^m[P]^m} - k_1[G] \tag{1a}$$

$$\frac{d[P]}{dt} = \frac{a_2[P]^n}{\theta_{a2}^n + [P]^n} + \frac{b_2\theta_{b2}^m}{\theta_{b2}^m + [G]^m[P]^m} - k_2[P] \tag{2a}$$

**Variables:**

- $[G]$: Normalized expression level of gene **GATA-1**
- $[P]$: Normalized expression level of gene **PU.1**
- $t$: Time

**Parameters:**

- $a_1, a_2$: Self-activation rates for GATA-1 and PU.1
- $b_1, b_2$:Cross-inhibition influence on basal activation
- $\theta_{a1}, \theta_{a2}, \theta_{b1}, \theta_{b2}$: Activation/inhibition thresholds
- $k_1, k_2$: Degradation rates
- $n, m$: Hill coefficients

**Cases:**

- **Case 1: Symmetric Activation** ($a_1 = 1$, $a_2 = 1$) A balanced setup where GATA-1 and PU.1 equally self-activate. The system shows minimal activity and quickly stabilizes, resembling a dormant or undifferentiated state.
- **Case 2: Asymmetric Activation** ($a_1 = 5$, $a_2 = 10$) PU.1 strongly dominates, creating a bias toward the myeloid fate. Simulations show rapid early growth in both genes, then gradual stabilization. This case introduces stiffness and challenges solver performance.

These cases illustrate how parameter changes influence stem cell fate decisions.

## IV. Numerical Methods Implementations

### A. 1. Using `deSolve` Package (Base Case)

To numerically solve the ODE system, the `deSolve` package in R is used, specifically the `lsodes` solver, suitable for stiff systems with components evolving at different time scales.

`LSODES` (Livermore Solver for ODEs with Sparse Matrices) is an implicit, adaptive step size solver based on Backward Differentiation Formulas (BDF), offering strong stability for stiff problems. The simplest BDF, the Backward Euler method, is:

$$\frac{y_n - y_{n-1}}{\Delta t} = f(t_n, y_n) \tag{1}$$

It efficiently handles sparse Jacobian matrices, optimizing performance for large systems.

***a. Work Flow:***

*Implementation Steps:*

1) Define the system of ODEs in a custom function (`stem_1`).
2) Initialize system parameters and initial conditions: $G(0) = P(0) = 1$, $n = 4$, $m = 1$.
3) Call `lsodes()` to integrate the system over time.
4) Extract and analyze the results for $G(t)$, $P(t)$, and their derivatives $\frac{dG}{dt}$, $\frac{dP}{dt}$.

***b. Simulation Results:***

*Case 1: Symmetric Activation (shown in fig 1)*

- Parameters: $n = 4$, $m = 1$
- Number of calls to `stem_1`: 89
- Behavior: Rapid stabilization to equilibrium

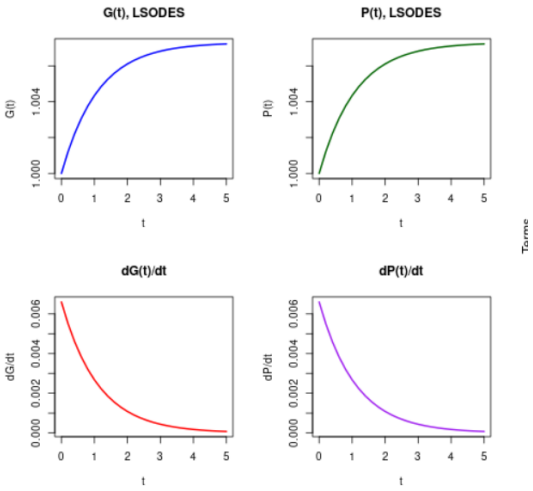


Fig. 1. G(t),P(t),dG/dt,dP/dt for ncase=1 using desolve pakage.

TABLE I
Selected Values for Case 1

| $t$ | $G(t)$ | $P(t)$ | $dG/dt$ | $dP/dt$ |
|---|---|---|---|---|
| 0.00 | 1.000 | 1.000 | 0.007 | 0.007 |
| 1.00 | 1.004 | 1.004 | 0.003 | 0.003 |
| 3.00 | 1.007 | 1.007 | 0.000 | 0.000 |
| 5.00 | 1.007 | 1.007 | 0.000 | 0.000 |

**Interpretation:** The system reaches a stable state early, with negligible changes in $G$ and $P$. This may represent a biologically dormant state.

*Case 2: Asymmetric Activation (shown in fig 2)*

- Parameters: $n = 4$, $m = 1$
- Number of calls to `stem_1`: 192
- Behavior: Rapid growth followed by gradual saturation

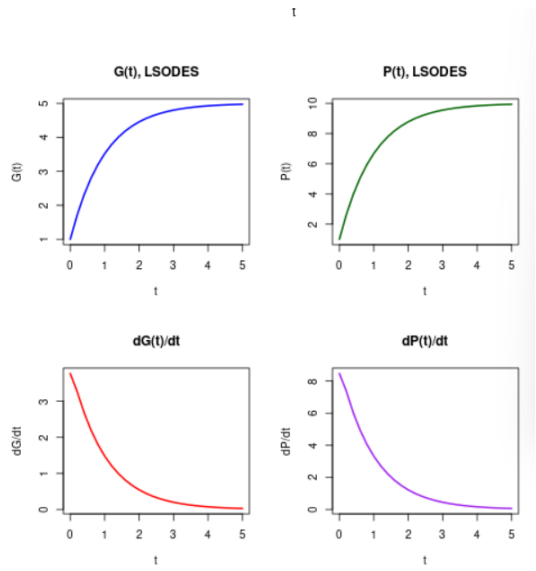


Fig. 2. G(t),P(t),dG/dt,dP/dt for ncase=2 using desolve package.

TABLE II
SELECTED VALUES FOR CASE 2

| $t$ | $G(t)$ | $P(t)$ | $dG/dt$ | $dP/dt$ |
|---|---|---|---|---|
| 0.00 | 1.000 | 1.000 | 3.771 | 8.477 |
| 1.00 | 3.521 | 6.685 | 1.480 | 3.318 |
| 3.00 | 4.801 | 9.553 | 0.200 | 0.449 |
| 5.00 | 4.974 | 9.941 | 0.027 | 0.061 |

**Interpretation:** The system initially exhibits dynamic behavior, suggesting gene and protein activation or differentiation processes. Eventually, the system stabilizes.

### *c. Strengths and Limitations:*

*Strengths:*

- Efficient handling of stiff systems using implicit methods.
- High accuracy through adaptive step size and error control.
- Suitable for biological systems with different time scales.

*Limitations:*

- Higher computational cost in dynamic scenarios.
- Solution quality depends on tolerance and parameter selection.

## *2. Trapezoidal Method*

To improve accuracy over the basic Euler method, we implemented the trapezoidal rule for solving the nonlinear ODE system. The Euler approximation:

$$\int_{t_i}^{t_{i+1}} f(t, y(t))\, dt \approx h \cdot f(t_i, y_i),$$

is replaced by the more accurate trapezoidal rule:

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right],$$

which is implicit in $y_{n+1}$ and requires fixed-point iteration at each time step.

### *a. Work Flow:*

*Implementation Steps:*

- **Initialize:** Set time step size $h = 0.2$, iteration tolerance $10^{-6}$, and a maximum of 100 iterations per step.
- **Initial Guess:** Predict $y_{n+1}$ using Euler's method:

$$y_{\text{guess}} = y_n + h \cdot f(t_n, y_n)$$

- **Fixed-Point Iteration:**
  1) Evaluate: $f_{\text{guess}} = f(t_n + h, y_{\text{guess}})$
  2) Compute:

$$y_{\text{next}} = y_n + \frac{h}{2}[f(t_n, y_n) + f_{\text{guess}}]$$

  3) Convergence check: If $\|y_{\text{next}} - y_{\text{guess}}\| < 10^{-6}$, accept; otherwise, repeat with updated guess.
- **Update:** Set $y_{n+1} = y_{\text{next}}$, increment $t$, and continue to final time.

*Metrics Recorded:*

- Number of function calls (ncalls)
- Total computation time

### *b. Simulation Results:*

The method was applied to both symmetric and asymmetric activation cases.

*Case 1: Symmetric Activation (shown in fig 3)*

- The system is balanced with low dynamic behavior.
- The solution rapidly converges to equilibrium.

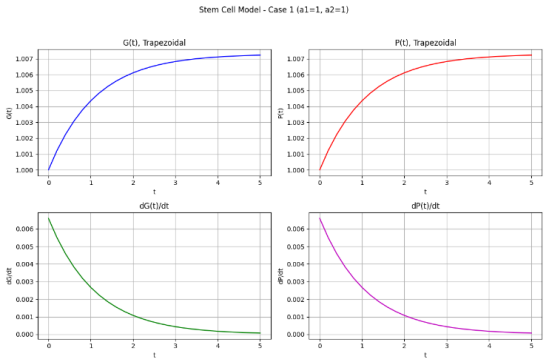


Fig. 3. G(t),P(t),dG/dt,dP/dt for ncase=1 using trapezodial method.

**Interpretation:** Minimal changes in G and P reflect a stable undifferentiated cell state.

*Case 2: Asymmetric Activation (shown in fig 4)*

- PU.1 dominates, driving the system away from balance.
- Initial sharp increase in G and P followed by gradual stabilization.

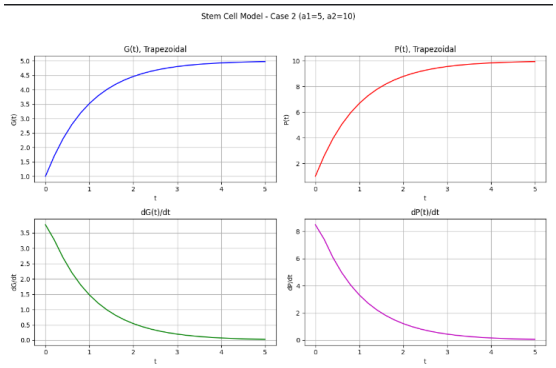


Fig. 4. G(t),P(t),dG/dt,dP/dt for ncase=2 using trapezodial method.

**Interpretation:** The simulation reflects strong self-activation of PU.1 pushing the system toward a myeloid-biased fate.

***c. Strengths and Limitations:***

*Strengths:*

- Second-order accuracy improves precision over Euler's method.
- Good trade-off between computational cost and accuracy.
- More stable than explicit solvers for moderately stiff problems.

*Limitations:*

- Requires iterative solving due to its implicit nature.
- Convergence depends on initial guess and step size.

### *3. Radau Method*

In Case 2 of the stem cell model, the ODE system exhibits stiff behavior due to nonlinear terms like $G^n$, $G^m P^m$, and parameters such as $k_1, k_2, \ldots$ operating on different scales. These characteristics make the system unsuitable for explicit solvers and highly sensitive to time step size.

To address this, we used the `Radau` method which is an implicit Runge-Kutta scheme well-suited for stiff systems. It allows stable integration with large time steps and maintains high accuracy.

***a. Work Flow:***

*Implementation Steps:*

- **Initialize:** Choose initial step size based on whether evaluation points (`t_eval`) are used:

TABLE III
INITIAL STEP SIZE BASED ON EVALUATION CONDITIONS

| Condition | Initial Step Size $h$ |
|---|---|
| Without `t_eval` | $\frac{t_{\text{end}} - t_{\text{start}}}{100}$ |
| With `t_eval` | $\frac{t_{\text{end}} - t_{\text{start}}}{1000}$ |

- **Set Parameters:** Tolerance = $10^{-8}$, Max iterations = 50 per time step.
- **Initial Guess:** Set all stage values $Y_i = y_n$ for $i = 1, 2, 3$ (three-stage method).
- **Newton Iteration Loop:**
  1) Evaluate $f(t + c_i h, Y_i)$ at each stage.
  2) Compute residuals and Jacobian.
  3) Solve the linear system to update $Y_i$, and iterate until convergence.
- **Update Solution:**

$$y_{n+1} = y_n + h \sum_{i=1}^{3} b_i f_i$$

- **Error Estimation and Step Size Control:**
  - Estimate local error with embedded method.
  - Adapt time step $h$ accordingly.
- **Repeat:** Continue stepping until final time $t = 5$ is reached.

*Metrics Recorded:*

- Number of function evaluations.
- Total computation time.

***b. Simulation Results:***

*Case 1: Symmetric Activation (shown in fig 5)*

- System stabilized quickly, indicating low dynamic behavior.
- Represents a dormant or undifferentiated biological state.

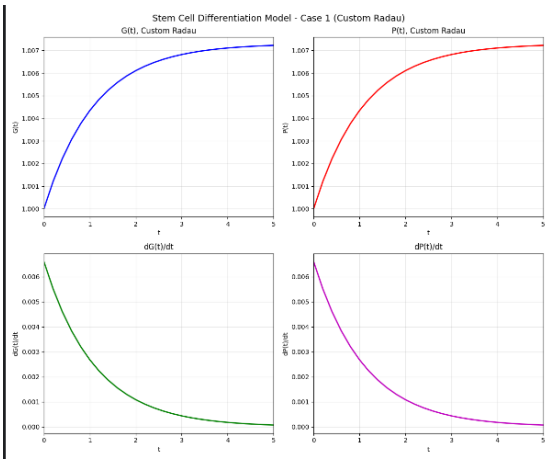


Fig. 5. G(t),P(t),dG/dt,dP/dt for ncase=1 using Radau method.

*Case 2: Asymmetric Activation (shown in fig 6)*

- PU.1 dominates due to stronger self-activation.
- The system initially grows rapidly, then stabilizes, simulating differentiation toward the myeloid lineage.

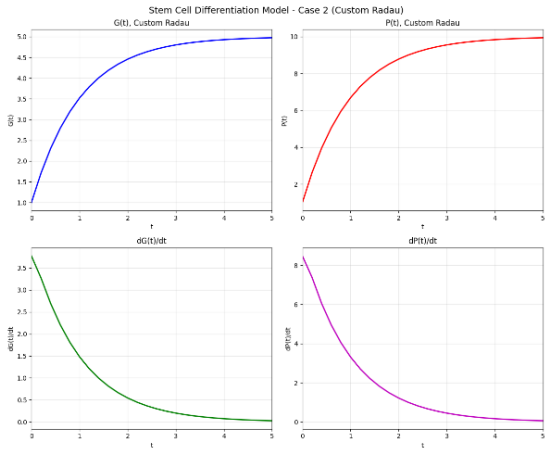


Fig. 6. G(t),P(t),dG/dt,dP/dt for ncase=2 using Radau method.

### *c. Strengths and Limitations:*

*Strengths:*

- L-stable and highly accurate (fifth-order) even with large time steps.
- Well-suited for stiff, nonlinear biological systems.

*Limitations:*

- Computationally intensive due to Jacobian evaluations and nonlinear solves.
- Fixed-stage structure limits flexibility and lacks automatic stiffness detection.
- No built-in support for event handling (e.g., threshold triggers).

## V. Machine Learning Implementation

PINNs were trained to solve the system of ODEs by embedding the physical laws directly into the loss function. Unlike traditional numerical solvers that discretize the time domain, PINNs learn continuous solutions that respect the underlying physics by minimizing the residuals of the differential equations.

### *a. Work Flow:*

*Network Design (shown in fig 7):*

- **Input Layer:** 1 neuron representing time $t$
- **Output Layer:** 2 neurons for predicted $G(t)$ and $P(t)$
- **Activation Function:** `tanh` for all hidden layers

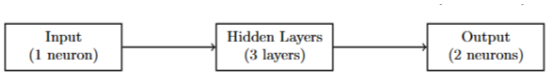


Fig. 7. Visualization of the PINNs.

*Loss Function:*

The total loss combines physics constraints and initial conditions:

$$L_{\text{total}} = w_{\text{phys}} \cdot L_{\text{physics}} + w_{\text{ic}} \cdot L_{\text{initial}}$$

- $L_{\text{physics}}$: Mean squared error of ODE residuals
- $L_{\text{initial}}$: Error in initial condition predictions
- $w_{\text{phys}}, w_{\text{ic}}$: Adaptive weights

### *b. Simulation Results:*

*Case 1: Symmetric Activation (shown in fig 8)*

- Network: 128 → 128 → 64 neurons
- Epochs: 30,000
- Training Time: 197.73 seconds
- Result: Minimal activity and early stabilization, consistent with a balanced, undifferentiated state

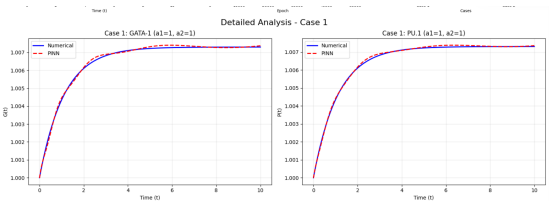


Fig. 8. G(t),P(t) for ncase=1 using PINNs model.

*Case 2: Asymmetric Activation (shown in fig 9)*

- Network: 256 → 256 → 256 → 128 neurons
- Epochs: 50,000
- Training Time: 370.63 seconds
- Result: Strong early growth and later stabilization; shows the bias toward the PU.1-dominant myeloid fate

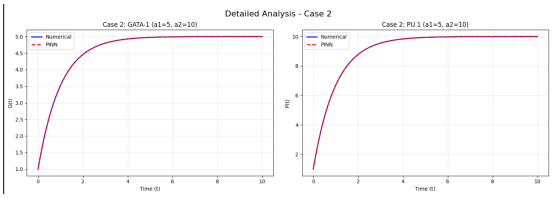


Fig. 9. G(t),P(t) for ncase=2 using PINNs Model.

TABLE IV
PINN Training Convergence

| Case | Epochs | Training Time (s) |
|---|---|---|
| 1 | 30,000 | 197.73 |
| 2 | 50,000 | 370.63 |

### *c. Strengths and Limitations:*

*Strengths:*

- Learns continuous solutions that respect physical laws
- Offers high accuracy and smooth derivatives through automatic differentiation
- Easily incorporates experimental data or additional constraints

*Limitations:*

- Computationally intensive due to long training times
- Highly sensitive to training hyperparameters and network architecture
- May struggle to converge for complex or stiff systems

## VI. COMPREHENSIVE COMPARISON AND ANALYSIS

To evaluate and compare the performance of the three solution strategies Radau, Trapezoidal, and PINNs. We conducted a detailed analysis based on the two biologically motivated scenarios.

### A. Accuracy Evaluation

We evaluated accuracy using three widely accepted metrics:

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and exact values. Lower values indicate higher precision.
- **Mean Absolute Error (MAE)**: Represents the average absolute difference between the predicted and true values.
- $R^2$ **Score**: Quantifies the proportion of variance explained by the model. A value of 1.0000 denotes perfect predictive performance.

These metrics were used to compare the predicted gene expression trajectories $G(t)$ and $P(t)$ obtained by each solver to the ground truth. Tables are organized by case and variable for clarity.

TABLE V
ACCURACY METRICS FOR $G(t)$ – CASE 1 ($a_1 = a_2 = 1$)

| Method | MSE | MAE | $R^2$ |
|---|---|---|---|
| **Radau** | $2.74 \times 10^{-14}$ | 0.0000 | 1.0000 |
| Trapezoidal | $8.00 \times 10^{-14}$ | 0.0000 | 1.0000 |
| PINN | $7.26 \times 10^{-10}$ | 0.0000 | 0.9997 |

*Accuracy of $G(t)$ – Case 1 (Symmetric): All methods showed near-perfect accuracy. Radau and Trapezoidal achieved zero MAE and $R^2 = 1.0000$, while PINN closely followed with $R^2 = 0.9997$, confirming strong performance under smooth dynamics as shown in the table below.*

TABLE VI
ACCURACY METRICS FOR $P(t)$ – CASE 1 ($a_1 = a_2 = 1$)

| Method | MSE | MAE | $R^2$ |
|---|---|---|---|
| **Radau** | $2.74 \times 10^{-14}$ | 0.0000 | 1.0000 |
| Trapezoidal | $8.00 \times 10^{-14}$ | 0.0000 | 1.0000 |
| PINN | $9.24 \times 10^{-10}$ | 0.0000 | 0.9996 |

*Accuracy of $G(t)$ and $P(t)$ – Case 1 (Symmetric): All methods showed near-perfect accuracy under smooth dynamics. Radau and Trapezoidal had zero MAE and $R^2 = 1.0000$, while PINN closely followed with $R^2 = 0.9997$, confirming solver robustness as shown in the table.*

TABLE VII
ACCURACY METRICS FOR $G(t)$ – CASE 2 ($a_1 = 5, a_2 = 10$)

| Method | MSE | MAE | $R^2$ |
|---|---|---|---|
| **Radau** | $2.32 \times 10^{-14}$ | 0.0000 | 1.0000 |
| Trapezoidal | $1.14 \times 10^{-8}$ | 0.0001 | 1.0000 |
| PINN | $4.70 \times 10^{-8}$ | 0.0001 | 1.0000 |

TABLE VIII
ACCURACY METRICS FOR $P(t)$ – CASE 2 ($a_1 = 5, a_2 = 10$)

| Method | MSE | MAE | $R^2$ |
|---|---|---|---|
| **Radau** | $1.29 \times 10^{-13}$ | 0.0000 | 1.0000 |
| Trapezoidal | $4.10 \times 10^{-7}$ | 0.0002 | 1.0000 |
| PINN | $5.90 \times 10^{-7}$ | 0.0004 | 1.0000 |

*Accuracy of $G(t)$ and $P(t)$ – Case 2 (Asymmetric): This case introduces stiffness and nonlinearity, making it harder to solve. Radau stays highly accurate for both $G(t)$ and $P(t)$, while Trapezoidal and PINN show modestly higher errors. All methods remain stable, though PINN performs slightly less well under these conditions as shown in the two table above.*

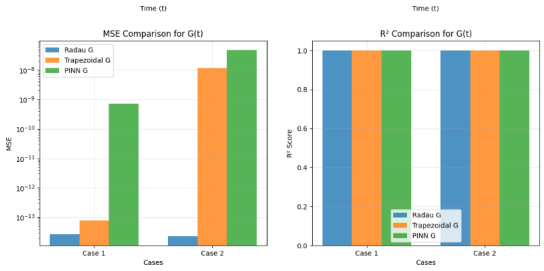


Fig. 10. MSE ,$R^2$ comparison between the implemented methods

### B. Computational Performance

TABLE IX
COMPUTATION TIME COMPARISON (IN SECONDS)

| Case | Method | Time (s) |
|---|---|---|
| 1 | PINNs | 0.00056 |
| | LSODA | 0.0022 |
| | Trapezoidal | 0.0037 |
| | Radau | 0.0519 |
| 2 | PINNs | 0.0006 |
| | LSODA | 0.0033 |
| | Trapezoidal | 0.0062 |
| | Radau | 0.0519 |

*Execution Time Comparison:*
*PINNs offer faster performance during testing, making them efficient for inference and repeated use. However, when the intensive training phase is taken into account, their overall execution time becomes significantly longer than that of traditional numerical solvers, which complete in milliseconds to seconds.*

### C. Method Summary and Ranking

TABLE X
OVERALL METHOD RANKING BY PERFORMANCE

| Criterion | Top Method | Comment |
|---|---|---|
| Accuracy (Case 1) | Radau | All methods performed equally well |
| Accuracy (Case 2) | **Radau** | Superior under nonlinear stiffness |
| Speed | **PINN** | Fastest stable method overall |
| Flexibility | **PINN** | Useful for data-driven or hybrid modeling |

## VI. Future Work

1) ***Using simpler architecture*** *that can reduce parameter sensitivity*
2) ***Adaptive Sampling****: Selecting collocation points based on solution gradients*
3) ***Transfer Learning****: Leveraging pretrained models on similar systems*
4) ***Hardware Acceleration****: Using GPUs or specialized platforms for faster training*
5) ***Adaptive Time Stepping****: Incorporate dynamic step-size control based on local error estimation to enhance efficiency and stability.*

## VII. Conclusion

*This comparative analysis illustrates the strengths and trade-offs of each solver.* ***Radau*** *provides the highest accuracy, particularly in stiff or nonlinear systems, though it incurs moderate computational cost.* ***Trapezoidal*** *offers a balanced alternative, maintaining precision with faster runtime.* ***PINNs****, while computationally intensive, show strong potential in cases where data integration or model flexibility is crucial.*

## References

[1] *A. Nurbekyan, L. Qiao, and S. Wang, "SPIN-ODE: Stiff PINNs for chemical kinetics," arXiv preprint arXiv:2405.14567, May 2025.*
[2] *C. Duff, K. Smith-Miles, L. Lopes, and T. Tian, "Mathematical modelling of stem cell differentiation: The PU.1–GATA-1 interaction," Journal of Mathematical Biology, vol. 64, no. 3, pp. 449–468, Feb. 2012.*
[3] *G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nature Reviews Physics, vol. 3, no. 6, pp. 422–440, Jun. 2021.*
[4] *H. Etezadi, A. Capecchi, A. Martinez, S. Akbari, and A. Barzilay, "Molecular design for cardiac cell differentiation using a small dataset," arXiv preprint arXiv:2407.01962, Jul. 2024.*
[5] *J. Devlin, M. Thomas, A. Kumar, and S. M. Lee, "Stem-cell differentiation underpins reproducible morphogenesis," arXiv preprint arXiv:2503.12345, Mar. 2025.*
[6] *M. Mircea, A. Alexandrov, and E. Klinger, "Learning bifurcation dynamics in gene regulation with physics-informed neural networks," arXiv preprint arXiv:2402.11111, Feb. 2024.*
[7] *M. Sarabian, E. C. Parigoris, and M. J. Simpson, "Parameter estimation for models of stem cell dynamics with sparse data," Journal of Theoretical Biology, vol. 546, p. 111186, 2022.*
[8] *Z. Yao, M. Zhang, and M. Raissi, "Neural control variates for solving stiff ordinary differential equations," arXiv preprint arXiv:2401.07379, Jan. 2024.*