Mathematical Modeling of a Bi-factor Stem Cell Differentiation System

Yuanchuan Shao, Fangrui Liu, Zhixiang Yao, Youyuan Hu

Department of Biomedical Engineering, Duke University, Durham 27708, USA; {ys379; fl118; zy154; yh403}@duke.edu

Abstract

This report presents the mathematical modeling and analysis of a bi-factor stem cell differentiation system, with a focus on hematopoietic stem cells (HSCs) and the important transcription factors GATA-1 and PU.1. By integrating deep learning, symmetry analysis, and nondimensionalization, we enhanced existing models in order to gain a more comprehensive understanding of the regulatory mechanisms that control stem cell development. Our project entails the simplification of the mathematical model, the generation of a realistic dataset with diverse parameters, and the utilization of machine learning approaches to classify and predict steady states. The outcomes of our project emphasize the efficacy of machine learning in examining intricate biological systems and emphasize the possibility of enhanced predictive modeling in biomedical research.

Introduction

In exploring degenerative diseases, the role of hematopoietic stem cells (HSCs) is fundamental. Central to their behavior are two transcription factors: GATA-1 and PU.1, which govern their differentiation.[1] Notably, each of these factors not only regulates its own expression — a process known as autoregulation — but also participates in a unique interaction known as mutual antagonism.[2] By integrating deep learning with a refined mathematical model with symmetry and nondimensionalization, we aim to gain a deeper understanding of these regulatory mechanisms. This report presents a detailed overview of our methodology and the insights gained from this approach.

Symmetry analysis is a crucial technique for simplifying complex simulations.[3] It's particularly effective because symmetry aligns closely with natural phenomena, making it an ideal choice for identifying patterns and forecasting results.[4] This alignment with nature's intrinsic properties enhances its effectiveness in various scientific and predictive applications. Additionally, nondimensionalization plays a key role in reducing the number of parameters in a system by consolidating and scaling variables of the original.[5] As a result, it enhances the model's generalizability and simplifies its analysis, ensuring a more efficient and comprehensible approach to understanding complex systems.[6, 7]

Machine learning (ML) is emerging as a power approach that can devise effective solutions for a variety of real-world challenges.[8] Within machine learning, classification stands out as a supervised learning method, referring to a problem of predictive modeling as well, where a class label is predicted for a given example.[9] An integral component of machine learning, artificial neural networks (ANNs) draw inspiration from biological neural networks.[10] Typically comprising input nodes, output nodes, and intermediary "hidden layers," ANNs mirror the complexity of

biological systems. This architecture enables them to process information and make predictions, contributing to the versatility and adaptability of machine learning algorithms in solving complex problems.

In this project, we refined Duff et al.'s mathematical model, focusing on efficiency by incorporating symmetry and nondimensionalization, which reduced simulation parameters. We then created a dataset featuring varied parameters and initial conditions, derived from simulations using this simplified model. The next step involved a thorough evaluation of this dataset to confirm its accuracy and relevance. Finally, we applied machine learning techniques, both to classify the differentiation system based on the count of stable steady states and to forecast these steady states under diverse conditions.

Methods

Mathematic Model Describing GATA-1 and PU.1 Gene Regulatory System

The bi-transcriptional-factor system is comprised of PU.1 and GATA-1, operating on a complex yet structured basis. Both PU.1 and GATA-1 engage in autoregulation, controlling their own production. Additionally, they inhibit each other's production through a PU.1–GATA-1 heterodimer, with separate binding sites designated for autoregulation and this mutual inhibition. Duff et al. assumes that such inhibition influences a baseline expression level. Notably, the system differentiates between the Hill coefficient, responsible for autoregulation, and the antagonism coefficient. To streamline the analysis, the model is designed to be structurally symmetrical, meaning that PU.1 and GATA-1 share identical Hill coefficients. This results in equations (1) and (2) for the system dynamics.[11]

$$\frac{d[G]}{dt} = a_1 \frac{[G]^n}{\theta_{a_1}^n + [G]^n} + b_1 \frac{\theta_{b_1}^m}{\theta_{b_1}^m + [G]^m[P]^m} - k_1[G] \quad (1)$$

$$\frac{d[P]}{dt} = a_2 \frac{[P]^n}{\theta_{a_2}^n + [P]^n} + b_2 \frac{\theta_{b_2}^m}{\theta_{b_2}^m + [G]^m[P]^m} - k_2[P] \quad (2)$$

Simplification of the Model

Duff et al.'s model consists of two ordinary differential equations (ODEs) encompassing a total of 14 parameters. Utilizing a deep learning approach on such a large parameter set would likely demand an extensive dataset and could result in considerable computational requirements. Additionally, their analysis was predominantly focused on scenarios involving symmetrical parameters, which were only a fraction of the possible cases.

In our approach to simplifying the model developed by Duff et al., we adopted two primary strategies: assuming a symmetrical system and applying nondimensionalization.

Symmetry Assumptions

To reduce the model's complexity, we assumed uniformity in several parameters across the genes involved: a) self-induction strengths ($a = a_1 = a_2$) and deactivation rates ($k = k_1 = k_2$)

are identical for each gene; b) cross-inhibition rates ($b = b_1 = b_2$) are the same between the two genes; and c) for gene regulatory processes, we assume concentration thresholds for self-induction ($\theta_a = \theta_{a_1} = \theta_{a_1}$) are equivalent and concentration thresholds for cross-inhibition ($\theta_b = \theta_{b_1} = \theta_{b_2}$) are identical. This approach results in a reduction of the parameter set.

Nondimensionalization

By introducing nondimensional variables $X = \frac{[G]}{\theta_a}$, $Y = \frac{[P]}{\theta_a}$, $\theta = \frac{\theta_a^2}{\theta_b}$, $\tau = k \cdot t$, $\alpha = \frac{a}{k \cdot \theta_a}$, and $\beta = \frac{b}{k \cdot \theta_a}$., and normalizing the equations with $k \cdot \theta_a$, the model is simplified into equations (3) and (4). This simplified model retains the core dynamics of the original system while being more amenable to analysis and simulation.

$$\frac{dX}{d\tau} = \alpha \frac{X^n}{1+X^n} + \beta \frac{1}{1+(\theta XY)^m} - X \tag{3}$$

$$\frac{dY}{d\tau} = \alpha \frac{Y^n}{1+Y^n} + \beta \frac{1}{1+(\theta XY)^m} - Y \tag{4}$$

A more detailed description of the derivation on the model simplification is provided in Supplementary Material 1.

Data Generation and Normalization Process

The simplified model comprises of 5 distinct parameters: α, β, θ, m, and n. The machine learning dataset was generated via a brute-force approach. We simply ran a loop, exploring all possible parameter (with unit) combinations, and computed average values after the Ordinary Differential Equations (ODEs) solver reached a stable steady state. Since certain parameter combinations had multiple stable steady states, we created a grid of (G, P) pairs (5 by 5) as initial conditions for each parameter combination to search for all these steady states. To remove duplicated steady states, we set a tolerance in advance and for steady states with distance smaller than the tolerance, we only picked one of them for the data set. Additionally, floating-point errors in Python are so small that the ODEs could be trapped by unstable steady states (saddle points), we added a small random perturbation to both ODEs in our model to skip these unwanted results. Then all the parameters, including the values of the steady states, in the dataset were nondimensionalized based on the equations mentioned in the previous section. After dataset generation, we reviewed all the data points to identify and eliminate weird data points, ensuring the dataset's overall quality and reliability for subsequent machine learning model development.

Data augmentation was performed after testing the entire machine learning pipeline. Points with single steady state took up 90% in the original dataset. The classifier in the machine learning pipeline overfitted this dataset easily. The accuracy of predicting data points with single steady state could reach 0.95, while for other data points, the accuracy could hardly reach 0.3. The aim of data augmentation was to balance the size of the data points with different number of steady states. For parameter settings that have multiple steady states, we purposely sampled more parameter combinations around these data points, thus generating more data points with the same number of

steady states as these center data points. After data augmentation, data points with different number of steady states had roughly balanced size and we could observe a great improvement in the classifier's training result.

Normalization is a critical preprocessing step in machine learning, aimed at standardizing the scale of input features to ensure uniform contributions during model training. To ensure that data was both centered around zero and brought to a comparable scale, we centered the data by subtracting the mean value of the training dataset and subsequently scaled the data by dividing it through the standard deviation of the training data.

Machine learning pipeline and the training process

Our machine learning model consists of Machine Learning Classifier Models and 4 subsequent Neural Network Regressors (Figure 1). The aim of the classifier was to categorize the data points into 4 classes based on the number of steady states they possess. Then the subsequent Neural Networks were used to predict the values of the corresponding steady states.

To train the entire model, we first used the entire dataset to train the classifier with the number of steady states as the outputs and the dimensionless parameters as the inputs. Then we manually separated the dataset into 4 sub-datasets with each dataset containing points with the same number of steady states. Then each neural network was trained with its corresponding dataset. The size ratio between the training dataset and the testing dataset was roughly 9:1.

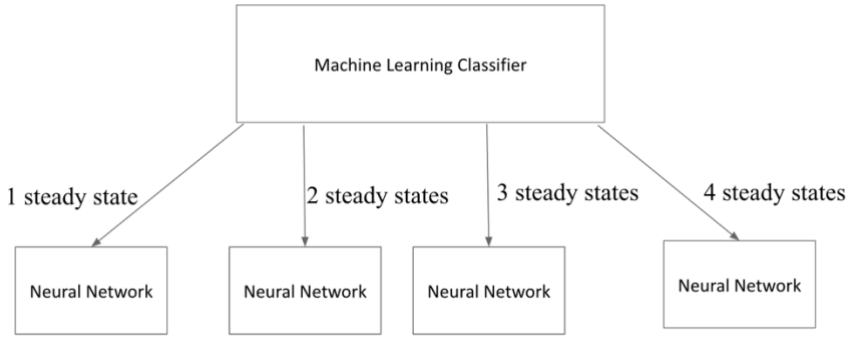


Figure 1: The machine learning pipeline for the dataset

Machine learning Classifier Models

Two traditional machine learning algorithms, Decision Tree and Random Forest were utilized to construct models using the dataset where the independent variables consist of the dimensionless parameters, while the dependent variables are the associated count of steady state. Decision Tree is a popular machine learning for data mining and pattern recognition**.** Random Forest is an ensemble algorithm for classification, which creates many decision trees by bootstrapping training samples and gives the predictions by integrating the outputs of the individual trees. In the case of Decision Tree, default parameters were utilized. As for Random

Forest, the number of decision trees in the forest was configured to 100. All utilized packages are sourced from scikit-learn.

Model performance was assessed in terms of true positive (TP), true negative (TN), false positive (FP), and false negative (FN). In addition, four metrics, including accuracy for the entire dataset, as well as precision, recall, and F1-score for each individual class. These metrics offer a detailed understanding of how well the model performed across different aspects of classification. The metrics were calculated using the following equations.

$$Accuracy = \frac{Correct\ Predictions}{All\ Presictions} \tag{5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

$$F1\ Score = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} \tag{8}$$

Neural Network

The neural network architecture was defined using the nn.Sequential module from PyTorch. The architecture consists of multiple fully connected (linear) layers followed by Rectified Linear Unit (ReLU) activation functions. The structure of the neural network is as follows:

Input Layer: in_length neurons (determined by the number of features in the input data)

Hidden Layer 1: 64 neurons with 'ReLU' activation function

Hidden Layer 2: 32 neurons with 'ReLU' activation function

Hidden Layer 3: 16 neurons with 'ReLU' activation function

Output Layer: out_length neurons (determined by the number of output features in the dataset)

The neural network was trained using the Adam optimizer with a learning rate of 0.0004 and utilized the Mean Squared Error (MSE) loss function. The training loop ran for 20,000 epochs, and during each 100th epoch, the R-squared score, MSE, and MAE were computed for monitoring the training progress.

After training, the model was saved and then loaded for evaluation on both the training and testing datasets. The performance metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared (R2) score, and the time consumed during training, are printed for assessment.

Results

Data Summary

The generated dataset comprises a total of 130,957 instances, categorized based on the number of steady states. Although class 1 (with 1 steady state) has a size of 79,390 instances,

representing the majority of the dataset, classes 2, 3, and 4 are relatively balanced at around 15,000 instances in each category. (Supplementary Material 2)

Evaluation of Machine learning Classifier Models

We examined the precision, recall, and F1 score metrics for each steady state class, which provide insights into the model's ability to correctly classify instances' corresponding counts of steady states. The overall accuracies for Decision Tree and Random Forest models are 89% and 91%, respectively. This indicates that our classifier model is capable of correctly classify instances across different classes and Random Forest model outperforms Decision Tree model. Additionally, given that the F1 scores for each class in the Decision Tree model are consistently above 0.70 and those in the Random Forest model are greater than 0.75 for all classes, it is evident that both models exhibit strong performance. The higher accuracy of the Random Forest model than Decision Tree model suggests that the Random Forest model provides a more accurate and robust classification across the entire dataset, making it a favorable choice for this classification task. (Supplementary Material 3, Suppl 3. Table 1, 2)

Evaluation of Neural Network

The R2 scores further explains that approximately 99% of the variance within the training set and 98% (the specific value may have minor differences in each run; we have chosen the latest result for this report) of that within the testing set can be explained by the Neural Network model trained for 1 steady state. However, it is noteworthy that the performance of models trained for datasets with 2, 3, and 4 steady states experienced a significant drop compared to the performance of the model trained for data with only one steady state. (Supplementary Material 3, Suppl 3. Table 3)

Visualization of Neural Network Models' Performance

The below graphical representations illustrate the performance of the Neural Network models across 4 steady states. Predicted values are plotted against true values, with correct predictions indicated by dots landing on the red reference line ($y = x$).

Figure 2 visualizes the comparison between the predicted outcomes of the neural network model on one steady state with the true value (visualization of other models are provided in Supplementary Material 4). The observed scatter plots reveal a decline in model performance as the number of steady states increases. The dots are gradually less clustered and deviated away from the red reference line, indicating that our model faces limitations, particularly when confronted with a higher number of steady states. This pattern highlights areas for potential future improvements in the model to enhance its capacity to handle datasets with multiple steady states more effectively.

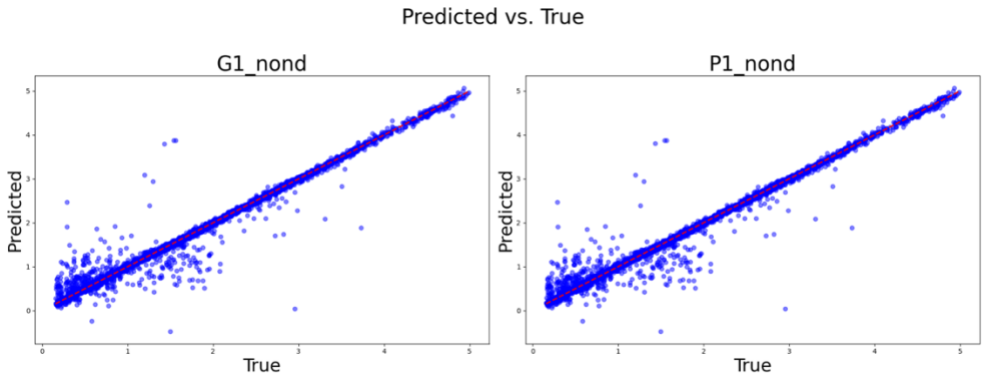


Figure 2: Predicted Values vs. True Values for Neural Network Model (1 Steady State)

Discussion

Based on the results shown in the previous section, it is not difficult to see that the classifier worked well on the dataset. The overall accuracy of the classifier on the testing dataset reached 0.9, which was a very good result. And the accuracy of each category also reached 0.8 separately. As a result, we could draw a conclusion that the classifier performed well on the dataset we generated, and it didn't overfit the dataset.

The performance of the neural network didn't reach our expectations, although it worked well on data points with single steady states. There are several possible reasons accounting for the unreliable behaviors. The first one is the random order of the steady states in the output vector. Neural network is simply a regressor. Namely, it is a map between the input vectors and the output vectors. To ensure that it has a good performance, we have to maintain the continuity among all the columns in the output vector. For data points with multiple steady states, random order of the steady states could possibly introduce sharp changes in the output vectors. That's why we observed a great difference in the accuracy between training dataset and testing dataset. The neural networks simply overfitted the training dataset.

Due to the poor performance of neural network models on datasets with 2, 3, and 4 steady states, we opted not to integrate them with the machine learning classifier model.

Conclusions

The integration of machine learning into mathematical modeling in this project has significantly improved our understanding of stem cell development, particularly in regards to the GATA-1 and PU.1 transcription factors in hematopoietic stem cells. The utilization of symmetry analysis, nondimensionalization, and deep learning in our models has greatly broadened the potential for computational investigation in biological systems. However, challenges in predicting several stable circumstances continue to exist. The upcoming study will prioritize the reorganization of stable conditions to guarantee smoothness, the classification of distribution datasets based on the symmetry of stable conditions, and the integration of machine learning classifiers with simpler artificial neural network models to mitigate overfitting. These efforts are expected to improve the accuracy of our models and expand their applicability in complex biological scenarios.

Reference

1. Roeder, I. & Glauche, I. Towards an understanding of lineage specification in hematopoietic stem cells: A mathematical model for the interaction of transcription factors GATA-1 and PU.1. *J. Theor. Biol.* 241, 852–865 (2006).

2. Huang, S., Guo, Y. P., May, G. & Enver, T. Bifurcation dynamics in lineage-commitment in bipotent progenitor cells. *Dev. Biol.* 305, 695–713 (2007).

3. Guo, H., Xu, Y., Li, Y., Huang, L. & Chen, H. A symmetry analysis methodology for general energy conversion systems. *Commun. Eng.* 2, (2023).

4. Layek, G. C. *An introduction to dynamical systems and chaos*. *An Introduction to Dynamical Systems and Chaos* (2015). doi:10.1007/978-81-322-2556-0.

5. Conesa, M., Sánchez Pérez, J. F., Alhama, I. & Alhama, F. On the nondimensionalization of coupled, nonlinear ordinary differential equations. *Nonlinear Dyn.* 84, 91–105 (2016).

6. Sánchez-Pérez, J. F., Jorde-Cerezo, G., Fernández-Roiz, A. & Moreno-Nicolás, J. A. Mathematical Modeling and Analysis Using Nondimensionalization Technique of the Solidification of a Splat of Variable Section. *Mathematics* 11, (2023).

7. Ganghoffer, J. F., Rahouadj, R. & Cheviakov, A. *Symmetry analysis and equivalence transformations for the construction and reduction of constitutive models*. *Advanced Modeling and Simulation in Engineering Sciences* vol. 8 (Springer International Publishing, 2021).

8. Sarker, I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Comput. Sci.* 2, 1–21 (2021).

9. Han, J., Pei, J. & Tong, H. Classification: basic concepts and methods. in *Data Mining* 239–240 (2022).

10. Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F. & Peter Campbell, J. Introduction to machine learning, neural networks, and deep learning. *Transl. Vis. Sci. Technol.* 9, 1–12 (2020).

11. Duff, C., Smith-Miles, K., Lopes, L. & Tian, T. Mathematical modelling of stem cell differentiation: The PU.1-GATA-1 interaction. *J. Math. Biol.* 64, 449–468 (2012).

Contributions

Yuanchuan (Robert) SHAO:

- Constructed the original model from the selected paper (Duff et al.) and conducted sanity check on the simulation result.
- Evaluated the validity of the simplified mathematical model by replicating results from the selected center paper.
- Generated the dataset and performed data augmentation for machine learning pipeline.

Fangrui (Lori) LIU:

- Preprocessed data to adapt to machine learning applications.
- Constructed both machine learning classifier models (Decision Tree and Random Forest) and artificial neural network prediction models.
- Assessed model performance by analyzing model evaluation metrics, and enhanced interpretability of the classifier models by assessing feature importance of each input variable.

Zhixiang (Carl) YAO:

- Constructed 2D and 3D demonstration on the original model to perform sensitivity analysis on the original model (demonstration with interactive panel available on GitHub repo).
- Derived the simplified model with symmetry analysis and nondimensionalization.
- Examined the reliability of generated data and helped with data augmentation.

Youyuan HU:

- Performed literature review and established sophisticated understanding on the theoretical background.
- Prepared and delivered the introductory sections for presentations, effectively communicating the project's background, objectives and methodologies.

All team members actively participated in the preparation of the presentations and composing the final report. We organized weekly meetings to update the progress of the project and setup GitHub repository. Each member of the team is solely responsible for the description of his/her contribution to the project.

GitHub repository to this project:
https://github.com/ZhixiangYao1999/GeneCircuit
Any further updates on the project will be available on our GitHub repo.

SUPPLEMENTARY MATERIALS TABLE OF CONTENTS

Supplementary Material 1: Simplification of the Mathematical Model

Original Mathematical Model

Based on the conclusion from Duff et al., the system comprises of GATA-1 and PU.1 can be described by the following ordinary differential equations (ODEs):

1. $\frac{d[G]}{dt} = a_1 \frac{[G]^n}{\theta_{a_1}^n + [G]^n} + b_1 \frac{\theta_{b_1}^m}{\theta_{b_1}^m + [G]^m[P]^m} - k_1[G]$
2. $\frac{d[P]}{dt} = a_2 \frac{[P]^n}{\theta_{a_2}^n + [P]^n} + b_2 \frac{\theta_{b_2}^m}{\theta_{b_2}^m + [G]^m[P]^m} - k_2[P]$

Assumptions for Symmetry

Characterized by the following equalities, the symmetrical system assumption simplifies the model by reducing 5 parameters out of 12.

In order to simplify the model, we assume a symmetrical system characterized by the following equalities:

- $a = a_1 = a_2$
- $b = b_1 = b_2$
- $k = k_1 = k_2$
- $\theta_a = \theta_{a_1} = \theta_{a_2}$
- $\theta_b = \theta_{b_1} = \theta_{b_2}$

Simplified ODEs under Symmetrical Assumptions

Under these assumptions, the ODEs are simplified to:

1. $\frac{d[G]}{dt} = a \frac{[G]^n}{\theta_a^n + [G]^n} + b \frac{\theta_b^m}{\theta_b^m + [G]^m[P]^m} - k[G]$
2. $\frac{d[P]}{dt} = a \frac{[P]^n}{\theta_a^n + [P]^n} + b \frac{\theta_b^m}{\theta_b^m + [G]^m[P]^m} - k[P]$

Units of Parameters

The units for the parameters in the ODEs are described as follows:

$[G], [P], \theta_a$: Units for concentrations

θ_b: Squared units for concentrations

a, b: Concentration divided by time

k: Inverse of time

m, n: Dimensionless (Hill coefficients)

Nondimensionalization

To further simplify the model for analytical or numerical treatment, we introduce nondimensional variables:

- $X = \frac{[G]}{\theta_a}$
- $Y = \frac{[P]}{\theta_a}$
- $\theta = \frac{\theta_a^2}{\theta_b}$
- $\tau = kt$
- $\alpha = \frac{a}{k\theta_a}$
- $\beta = \frac{b}{k\theta_a}$

Final Form of Simplified ODEs

Using the nondimensional variables, the final form of the simplified ODEs is:

1. $\frac{dX}{d\tau} = \alpha \frac{X^n}{1+X^n} + \beta \frac{1}{1+(\theta XY)^m} - X$
2. $\frac{dY}{d\tau} = \alpha \frac{Y^n}{1+Y^n} + \beta \frac{1}{1+(\theta XY)^m} - Y$

This reformulation allows for a more manageable analysis of the system, retaining the essential dynamics while reducing complexity.

Supplementary Material 2: Data Generation Summary

Suppl 2. Table 1: Data Distribution table

Number of Steady State (Class)	1	2	3	4	Total
Data Size	79,390	16,221	22,839	12,507	130,957

Supplementary Material 3: Model Performance Summary

Suppl 3. Table 1: Decision Tree Performance Breakdown

	Precision	Recall	F1 Score
1 (Steady State)	0.96	0.96	0.96
2 (Steady States)	0.72	0.72	0.72
3 (Steady States)	0.81	0.81	0.81
4 (Steady States)	0.82	0.80	0.81
Overall Accuracy	0.89		

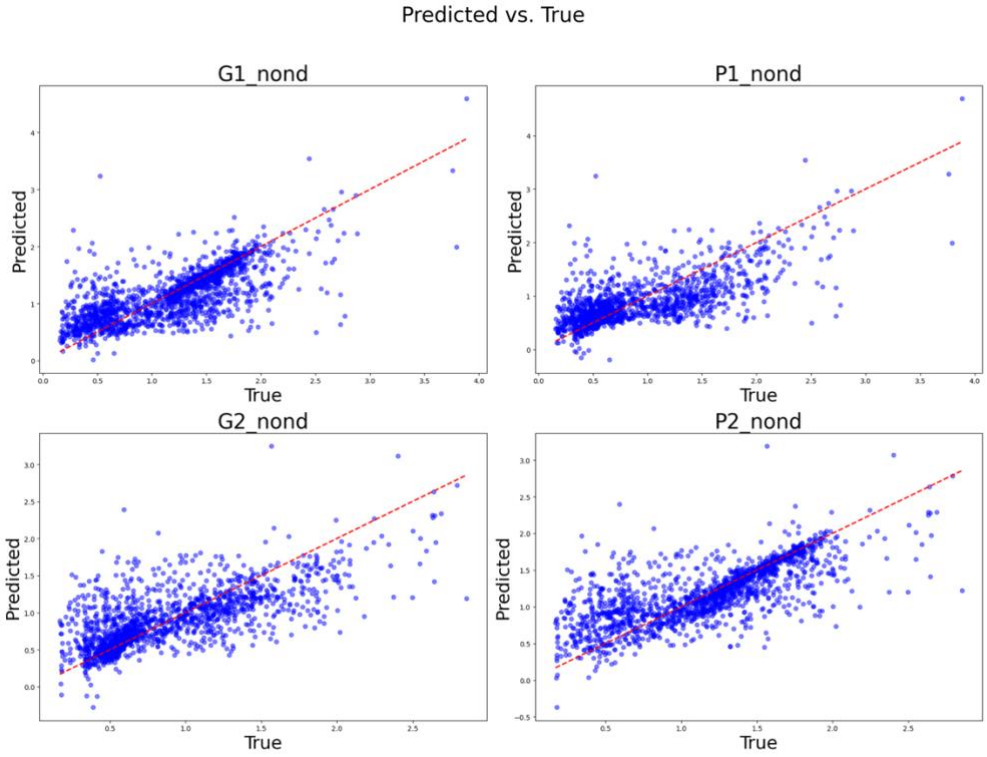
Suppl 3. Table 2: Random Forest Performance Breakdown

	Precision	Recall	F1 Score
1 (Steady State)	0.97	0.96	0.97
2 (Steady States)	0.77	0.78	0.78
3 (Steady States)	0.84	0.85	0.84
4 (Steady States)	0.86	0.84	0.85
Overall Accuracy	0.91		

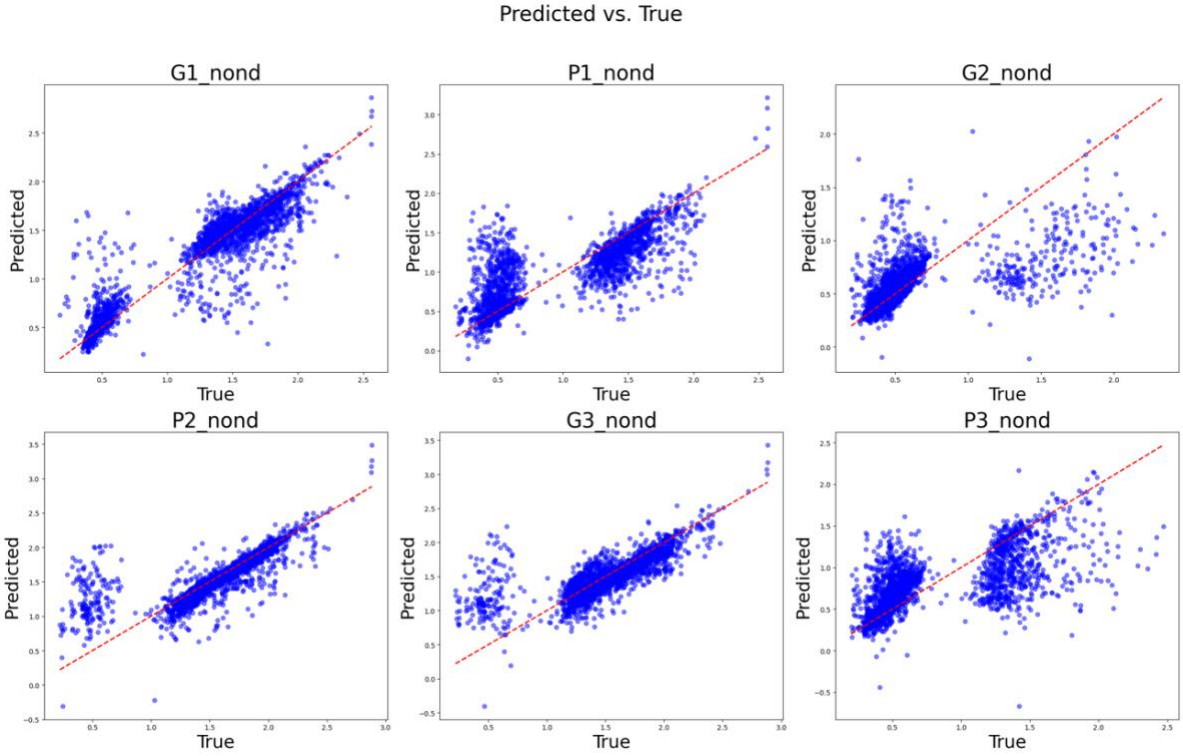
Suppl 3. Table 3: Performance Breakdown of Neural Network Models

	Neural Network for 1 Steady State		Neural Network for 2 Steady States		Neural Network for 3 Steady States		Neural Network f or 4 Steady States	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
MSE	0.00990	0.0169	0.0913	0.137	0.0743	0.0835	0.0475	0.0667
MAE	0.0357	0.0406	0.210	0.251	0.170	0.179	0.0917	0.110
R2 Score	0.990	0.983	0.645	0.491	0.604	0.559	0.677	0.562

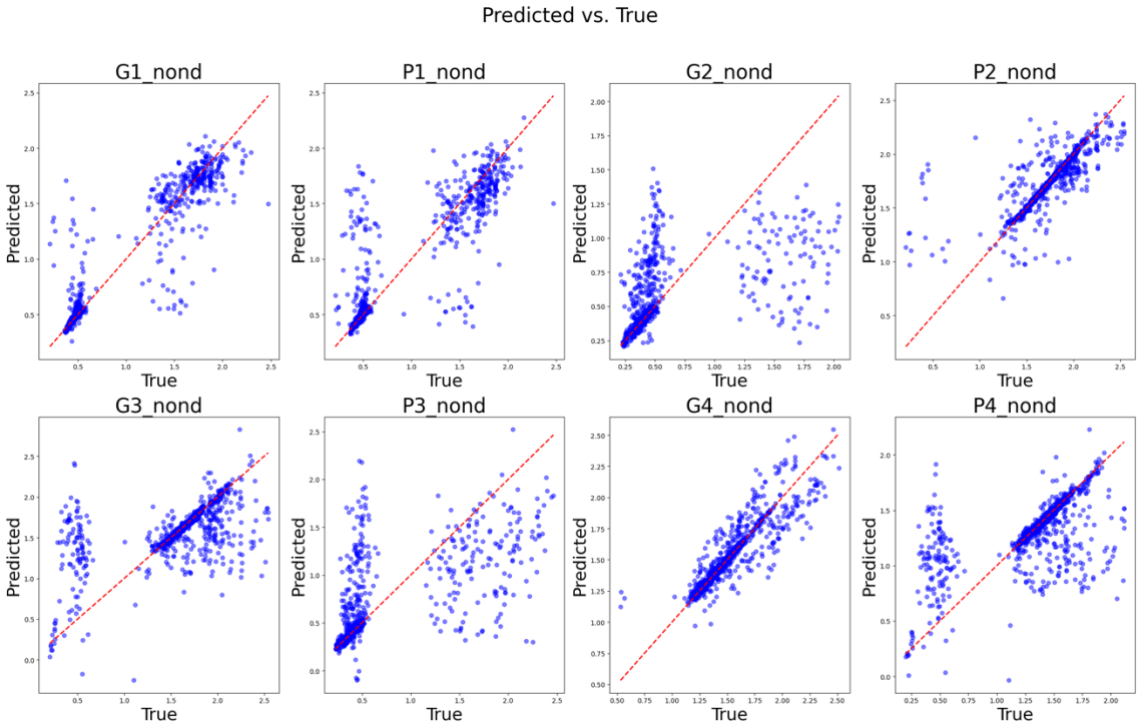
Supplementary Material 4: Additional Prediction Vs. True Value Plots



Suppl 4. Figure 1: Predicted Values vs. True Values for Neural Network Model (2 Steady States)



Suppl 4. Figure 2: Predicted Values vs. True Values for Neural Network Model (3 Steady States)



Suppl 4. Figure 3: Predicted Values vs. True Values for Neural Network Model (4 Steady States)