AUGMENTED SYSTEMS-BIOLOGY INFORMED NEURAL NETWORKS FOR PARAMETER IDENTIFICATION OF THE NOTCH MODEL

ALEX HUANG, KARTIK RAMACHANDRULA, AGNIV SARKAR, AND LU LU

Abstract. The process of neurogenesis in the mammalian brain is controlled by the Notch signaling pathway, which can be modeled with a system of ordinary differential equations relating the concentrations of species. However, this system contains a relatively large number of state variables (species) and parameters; as such, it is computationally costly to model the system, even with current techniques. In this paper, we describe a neural network pipeline to elucidate properties of the system as well as forecast species. First, we extensively discuss the use of identifiability analysis in systems biology problems to offer guidance in modeling. We show the utilization Systems-Biology Informed Neural Networks (SBINNs) architecture to extract values of ODE parameters as well as model the dynamics of the chemical species. In addition, we describe the implementation of additions to SBINNs such as warm-starting and considering sensitivity of parameters that enhance the learning of the model. Our results should provide accurate predictions of the biochemical dynamics in the Notch signaling pathway and help neuroscientists in the field better understand the formation of neurons. We also describe how we can further this technique and evaluate other modern architectures such as PINNformers and KANs to enhance predictions.

1. Introduction

Systems biology is the holistic study of complex interactions within biological systems using computational and mathematical tools. Although it is sometimes possible to model the system of interest using ordinary differential equations (ODEs), some parameters (e.g. rate constants) may be unknown from current experimental procedures or require numerical derivation due to careful estimation of state variables, which are often unavailable due to technical limitations [7]. In addition, accurate models of the dynamics of the biochemical species are of particular interest for biological applications and research. As such, we investigate methods for extracting both the equations and the variable dynamics of biological systems.

The Notch signaling pathway is an evolutionarily conserved pathway in multicellular organisms that regulates the determination of cell fate during development while simultaneously maintaining adult tissue homeostasis [1]. This biological process regulates juxtacrine cellular signaling, in which both signaling molecules and the receptor are affected by ligand-receptor crosstalk [1]. Our goal is to determine the underlying ODE system of the Notch signaling model, predict the dynamics of biochemical species, and help explain the factors that lead to neurogenesis.

The reconstruction of the ODE system is an example of the inverse design paradigm. The Physics-Informed Neural Network (PINN) architecture has been especially successful in solving inverse design problems [22]. Using the PINN methodology in the form of systems biology neural networks (SBINN) [39], we can determine the parameters of the ODE systems and model the system using Neural Networks. Our goal is to implement and evaluate the efficacy of the SBINN architecture in determining the Notch signaling pathway.

With the large parameter space to search and ill-conditioning of the problem, many parameters cannot be immediately identifiable by the SBINN. So we propose implement a pipeline for the development and identification of systems biological models introduced in [7]. The workflow we employ is:

- **Step 1:** Data acquisition and development of systems-biological models. This is not the primary focus of our work and we use the underlying ODE system from [24] and the data provided from numerous literature sources [12, 15, 17, 19, 33, 37].
- **Step 2:** Structural identifiability analysis. We consider the identifiability of each unknown parameter of the Notch pathway ODE system. If none of the parameters are identifiable, then more data is needed to solve the system. If parameters are locally identifiable, we need to impose search ranges for the model to converge to a solution. Global identifiability is optimal as it gaurantees a convergence.
- **Step 3:** Implementation of SBINN. We use a modified SBINN architecture to estimate the identifiable parameters. Specifically, we implemented transfer learning technique and used sensitivity to efficiently train the fitting of parameters and ensure convergence.
- **Step 4:** Practical identifiability analysis. We check the sensitivity and quality of our estimates of the parameter values from SBINN model to determine if the parameters are practically identifiable. If sensitivity is high, then more data or conditions are needed for the system.
- **Step 5:** Using the inferred parameters and the SBINN model, we are able to approximate the dynamics of the Notch biochemical species. As of now, we are still striving for accurate approximation of the model as well as recovery of all 22 structurally-identifiable parameters.

2. Notch Signaling Pathway

2.1. **Notch mutation in the common fruit fly *Drosophila melanogaster*.**

The first alleles of Notch in history arose as spontaneous dominant mutations in a genus of fruit flies known as *Drosophila* [14]. It was not difficult to recover them because Notch is haploinsufficient in Drosophila, meaning that a heterozygous combination of a wild allele is insufficient to produce a wild phenotype. Continuing work on Notch—beginning with one of the first characterized chromosomal deficiencies, O. L. Mohr in the 1970s strove to establish the context of advancing concepts regarding the nature of genes [23]. The various kinds of alleles of Notch generated during this era became a priority for molecular biologists when cloning and sequencing became applicable tools in the 1980s.

Notch proteins in vertebrates consist of four single-pass transmembrane receptor proteins (Notch-1 to Notch-4) that contain multiple epidermal growth factor-like repeats followed by conserved cysteine-rich Notch/Lin12 repeats in their extracellular domain and six cdc10/ankyrin repeats in their intracellular domain [9]. The Notch ligands (Jagged-1, Jagged-2, and Delta-1 to Delta-3) represent transmembrane proteins that, like Notch, contain multiple epidermal growth processes in their extracellular domain. Ligand binding induces proteolytic cleavage and release of the intracellular domain in the C-terminus, the

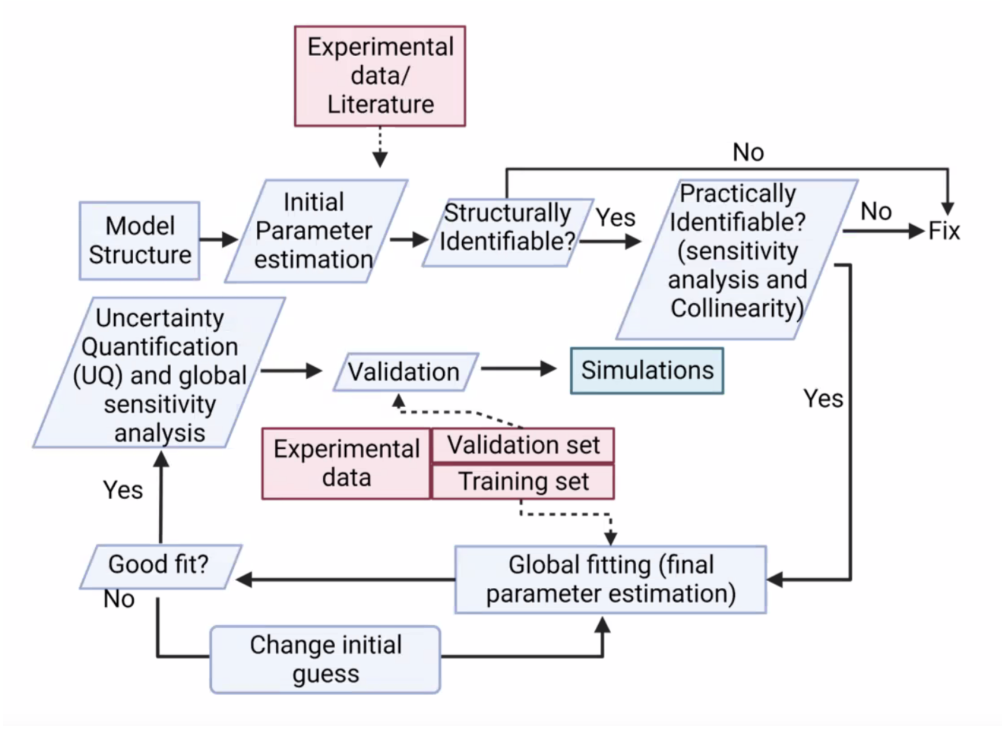

Experimental data/ Literature
Model Structure
Initial Parameter estimation
Structurally Identifiable?
Yes
No
Practically Identifiable? (sensitivity analysis and Collinearity)
No
Fix
Yes
Uncertainty Quantification (UQ) and global sensitivity analysis
Validation
Simulations
Experimental data
Validation set
Training set
Yes
Good fit?
No
Global fitting (final parameter estimation)
Change initial guess

FIGURE 1. **Process map of solving ODE-Based Mechanistic Model of Biological Signaling Pathways.** Figure is adapted with permission [24].

end of an amino acid chain that retains signals for protein sorting. Consequently, the signaling molecule crosses into the intracellular domain (Notch-IC) followed by being nuclear translocated.

A primary target of activated Notch-1 is the ubiquitous DNA binding protein RBP-Jκ/CBF-1. Activated Notch proteins interact with RBP-Jκ/Su(H) primarily through the RAM23 domain, a sequence that was identified N-terminus, the start of a protein or polypeptide. This transactivation domain is localized adjacent to the ankyrin repeats of Notch-1-IC, resulting in the activation of transcription. Downstream targets of Notch signaling such as Enhancer of split [E(spl)] complex genes and mammalian homologs of Hairy and E(spl) genes, HES-1 and HES-5, have been identified. These basic helix-loop-helix (bHLH) proteins antagonize other bHLH factors like myoblast determination protein 1 (MyoD) that induce differentiation.

2.2. Modeling Biological Pathways.

Biological pathways are ubiquitous to the study of cellular biology. These pathways can be studied in a systems-biology lens by constructing lumped-element model that, although idealistic, can capture much of the interaction quantities of interest with tuning of parameters. However, measurements of parameters are often difficult or not possible during reactions. Therefore, biologists seek to determine these parameters (namely reaction rates).

ODE models of interest to us involve a cell releasing a stimulus, such as a vascular endothelial growth factor (VEGF), that binds to the receptors of endothelial cells. The reactions caused by this process can be tracked by rate factors, as in *production rate constant* =

rate of VEGF. We set our parameters as these rate factors, which we then reference in our ODE systems. The complexity of the framework increases exponentially with the addition of parameters, so the 26 parameters in our problem lead to considerable computational complexity.

In the context of our work, we follow the mechanistic model development and evaluation schemes followed by systems-biologists as in Fig. 1. However, our methodology differs from the general workflows namely due to our implementations of SBINNs for parameter estimation and incorporating various methods like sensitivity analysis directly into the fitting stage.

2.3. **Notch Pathway Data.**

In order to train our neural networks and validate our pipeline, we require data on the state variables. Indeed, our system aims to recover the dynamics of the state variables from the observable values. However, due to difficulties in experimental measurement, our empirical dataset is limited to the following Table 2.

Reference	**Time**	**mDll4**	**$\mathbf{NICD_{Notch}}$**	**Hes**	**NICD2**	**R2**
Fish (2017) [12]	0	0.169				
Izumi (2012) [15]	0				0.678480181	
Fish (2017) [12]	900	0.2394				
Izumi (2012) [15]	900				0.720351573	
Takeshita (2007)	900		0.530068885	0.48525544		
Fearnley (2016)[11]	1200					
Fish (2017) [12]	1800	0.6338				
Izumi (2012) [15]	1800				1	
Takeshita (2007) [33]	1800		1	1		
Fish (2017) [12]	3600	1				
Izumi (2012) [15]	3600				0.920314286	
Takeshita (2007)	3600		0.699141704	0.50385688		
Fish (2017) [12]	7200	0.352				
Izumi (2012) [15]	7200				0.68713585	
Takeshita (2007)	7200		0.509760438	0.206604465		
Fearnley (2016) [11]	0					1
Fearnley (2016) [11]	300					1.007343
Fearnley (2016) [11]	900					0.852512
Fearnley (2016) [11]	1800					0.716371
Fearnley (2016)[11]	3600					0.680827
Fearnley (2016) [11]	7200					0.67802

TABLE 2. **Experimental data of Notch model species.**

Such data is not sufficient to train models, and as here we are validating the efficacy of our models, we generate synthetic data from the inferred parameter values given in Appendix B. Substituting the nominal parameter values into the ODE system with initial values $\mathbf{x}(0) \approx 0$, we use standard numerical methods to retrieve the training data fulfilling Step 1 of this workflow. For training purposes, we only use the synthetic data generated for the observables, as in general application, this is the data present. See Fig. 2 for data generation for the dynamics of pR2 species.

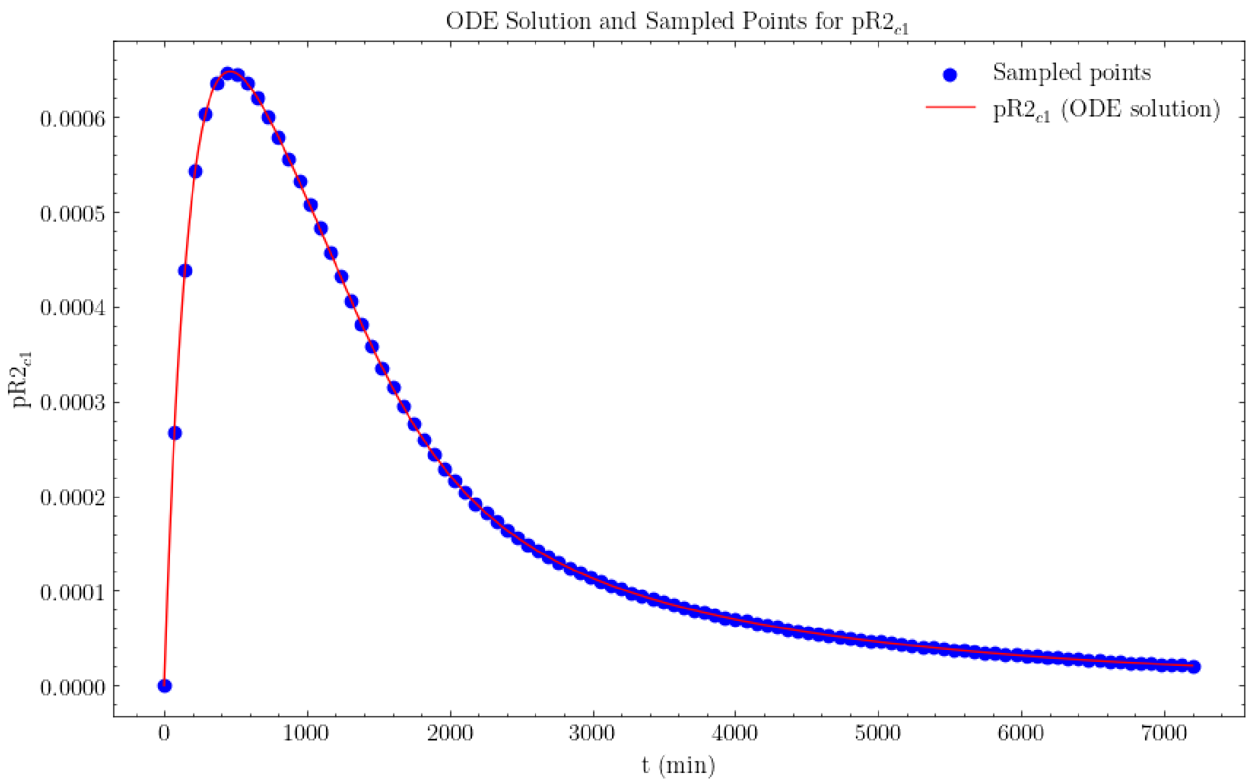


FIGURE 2. **Notch signaling model observation data for parameter inference** 100 measurements on ribonucleotide reductase (pR2) levels are randomly sampled in the time window of $(0, 7200)$ minutes ($\sim$ five days).

3. Structural Identifiability

When analyzing the fit of a neural network model to the equations and parameters of a dynamic system, issues arise in the identification of the parameters. The parameter p is identifiable if the confidence interval of the estimate of $\hat{p}$, the value of the inferred parameter, is a finite interval. The two classes of problems that may occur are *structural* and *practical*, not necessarily disjointly. In this section, we detail the theory and application of several methods for discerning and addressing structural non-identifiability *a priori*, so that our models will not run into issues with impossible or incorrect convergence.

3.1. **Structural Identifiability.** A model is structurally non-identifiable when multiple solutions of $\mathbf{y}$, the observable state variables, appear due to insufficient mappings, denoted $\mathbf{h}$, of the state variables $\mathbf{x}$ to $\mathbf{y}$. Structural identifiability, hence, is performed *a priori* in terms of fitting parameters. Such analysis is necessary for two reasons: one is that we are interested in seeing what biochemical species can be modeled given experimental data and current knowledge of reactions, and the other is avoid fitting problems in the subsequent applications of SBINNs. There are two solutions to resolve this: one is acquiring more data for more species, the other is to fix a subset of the parameters to nominal values while fitting the model to the complement set of parameters.

Consider a dynamical system given by,

$$\mathbf{X}' = \mathbf{f}(\mathbf{X}, \mathbf{\Theta}, \mathbf{u}), \qquad \mathbf{y} = \mathbf{g}(\mathbf{X}, \mathbf{\Theta}, \mathbf{u}), \tag{1}$$

where $\mathbf{X} = (X_1, \cdots, X_n)$ represents the state variables, $\mathbf{y} = (y_1, \cdots, y_m)$ represents the observable state variables, $\mathbf{\Theta} = (\theta_1, \cdots, \theta_k)$ are the parameters, and $\mathbf{u}$ represents the input variable to the system.

Definition 1. A parameter set Θ is called structurally *globally identifiable* if

$$g(\mathbf{X}, \mathbf{\Theta}, \mathbf{u}) = g(\mathbf{X}, \mathbf{\Phi}, \mathbf{u}) \implies \mathbf{\Theta} = \mathbf{\Phi} \tag{2}$$

for every $\mathbf{\Phi} = (\phi_1, \cdots, \phi_k)$ in the same space as $\mathbf{\Theta}$. *Local identifiability* requires Eq. (2) to hold in a neighborhood of $\mathbf{\Theta}$.

If the set is locally identifiable, this suggests a search range to train the model. In this section, we determine the structural identifiability of the Notch signaling pathway using three different methods: the Julia library *StructuralIdentifiability* [8], the *Generating Series test for Structural Identifiability* [6, 5] (a.k.a GenSSI), and *Structural Identifiability Taken as Extended-Generalized Observability with Lie Derivatives and Decomposition* [35] (a.k.a STRIKE-GOLDD).

3.1.1. *Structural Identifiability: Julia Library.* The first method to realize structural non-identifiabilities is using the Julia Library *Structural Identifiability.* The code relies on the method outlined in Dong et al. [8], viewing structural identifiability as the central *differential elimination problem.* In an arbitrary scientific problem, there exist state variable $\mathbf{X}$, observable (or output), $\mathbf{y}$, and input $\mathbf{u}$ related by Eq. (1). Of principal interest is the determination of the *input-output relations* which is performed by an elimination algorithm. In addition, the structural identifiability algorithm (Algorithm 5.3 [8]) probabilistically determines the sufficiency of the relations for local and global identifiability.

3.1.2. *GenSSI.* GenSSI is a Matlab tool box using the generating series method [36] of structural identifiablity coupled with the accessible interpretation of identifiability tableaus [2]. Generating series method is an extension of the Taylor Series method [26] approach. The Taylor series approach seeks to determine the Taylor series of the observable in terms of the inputs around the initial values. Given a system of equations for these coefficients, if there does not exist a unique-solution (rank deficiency), then non-identifiability is detected. However the complexity of the algebraic parameters relations, the failure to know to what extend the Taylor series must be taken, and the lack of insight into local identifiability render this method not preferable. The generating series method similarly calculates coefficients of the observables in series expansion with respect to time and inputs in such a way that the coefficients of this series are the output functions $\mathbf{h}(\mathbf{x}, \mathbf{\Theta}, t)$ and their Lie derivatives with respect to $\mathbf{f}, \mathbf{g}$ from Eq. (1).

Definition 2. The *Lie derivative* of a function $\mathbf{h}\colon M \to \mathbb{R}$ on manifold M with respect to vector field $\mathbf{Y}$ of a point $p \in M$ on flow $\gamma(t)$ is given by,

$$(\mathcal{L}_Y \mathbf{h})(p) = \lim_{t\to 0} \frac{h(\gamma(t)) - h(p)}{t} = \sum_{i=1}^{n_X} \mathbf{Y}_i \frac{\partial \mathbf{h}}{\partial x_i} \tag{3}$$

Calculating the Lie derivatives and the resulting set of non-linear algebraic equations will require systemic computation of *identifiability tableaus* to discern identifiability problems. The tableau is simply the Jacobian of the coefficients of the generating series expansion (the Lie derivatives taken to an a priori order) taken over the parameters, and the non-zero elements are shaded. If the Jacobian is rank-deficient, this indicates that the corresponding system of equations does not have a unique solution, causing non-identifiability. Such a deficiency can be visually seen by an "empty-column" of a parameter. On the other hand, if there is a unique element in a column, this means that the parameter is globally structurally identifiable. As such, continuing a series of reductions will elucidate the globally identifiable and non-identifiable solutions. Once the final tableau is reached, there are either several meaningful solutions, indicating local identifiability, or none, indicating non-identifiability.

3.1.3. *STRIKE-GOLDD.* STRIKE-GOLDD is another Matlab toolbox that determine identifiability using the concept of observability, introduced by Kalman for linear models [16]. For a non-linear vector system, we use Lie derivatives (Definition 2) creating observablity matrix [35] for a system Eq. (1):

$$\mathcal{O}(x) = \begin{pmatrix} \left(\frac{\partial}{\partial x}\mathbf{g}(x)\right)^{\intercal} \\ \left(\frac{\partial}{\partial x}(\mathcal{L}_{\mathbf{f}}\,\mathbf{g}(x))\right)^{\intercal} \\ \left(\frac{\partial}{\partial x}(\mathcal{L}^2_{\mathbf{f}}\,\mathbf{g}(x))\right)^{\intercal} \\ \vdots \\ \left(\frac{\partial}{\partial x}(\mathcal{L}^{n-1}_{\mathbf{f}}\,\mathbf{g}(x))\right)^{\intercal} \end{pmatrix}$$

for n state variables. Then the *Observability Rank Condition* (ORC) formulates that the parameters of the system are locally identifiable if $\text{rank}(\mathcal{O}(x)) = n$ [34]. To determine which parameters are causing non-identifiability, we sequentially remove them from the observability matrix and recalculate rank.

3.2. **Results of Structural Identifiability Analysis for Notch Model.** We implement the aformentioned structural identifiability methods to the ODE system governing the notch model (enumerated in Appendix C) using the corresponding python and MATLAB tools. Due to the probabilistic nature of the Julia method and the necessity of immense compute to calculate higher Lie series in the other two methods, we compare the results of all three to derive theoretically sound council before proceeding.

First consider the use of StructrualIdentifiability.jl. We provide the results given all of the observables and fixing subsets of non-identifiable parameters. Note that the parameters are provided in two lines. The results are Table 3:

Parameter	kf_{dllN}	kp_{R2}	kdp_{R2}	kr_{dllN}	Km	kcat	$kdeg_{NICD}$	$kdeg_{Notch}$	$kdeg_{Dll4}$	kp_{Dll}	teta	$kdeg_{Hes1}$	Kp_{Hes}
	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
kcat fixed	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓
kcat, $kdeg_{iR2}$ fixed	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓

Parameter	tetaHe	kon_{cis}	$kdeg_{Jag}$	$kr_{jagNotch}$	kr_{cis}	$kf_{jagNotch}$	Kp_{Jag}	tetaJag	$kdeg_{pR2}$	$kdeg_{iR2}$	Gs	$kform_{Notch}$	kp
	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗
kcat fixed	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗
kcat, $kdeg_{iR2}$ fixed	✓	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	✓

TABLE 3. **Identifiability results of the Notch model fixing different observables and parameters.** ✓ indicates local identifiability; ✗ indicates non-identifiability.

Observe that initially, the system is structurally non-identifiabiale. However after fixing both kcat, $kdeg_{iR2}$, the system becomes identifiable. Hence, for the purposes of the SBINN model, to have a fit of the other parameters we must use the nominal values for kcat, $kdeg_{iR2}$ found in [25] and [32] respectively.

Now consider the results of the GenSSI implementation. We choose eight orders of Lie Derivatives to determine the Jacobian to have maximal accuracy, and note that further orders do not imply much change. Running identifiability on all of the parameters initially, we observe the following identifiability tableau Fig. 3.

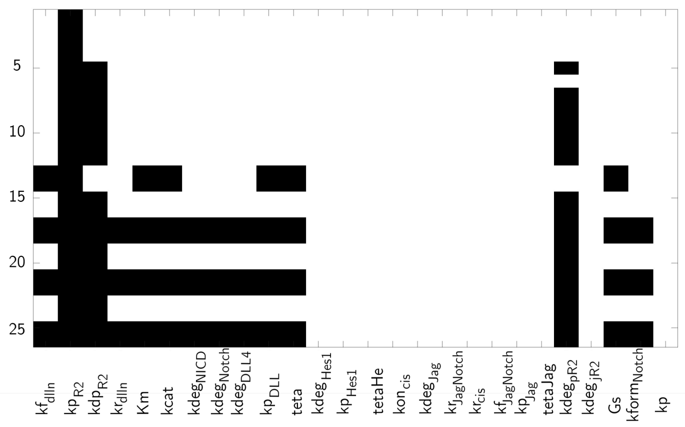


FIGURE 3. **Identifiability Tableau from GenSSI.** The parameters are given on the horizontal axis and their appearance in the Lie derivative expansion is indicated by shading of the Jacobian matrix.

Due to the lack of shaded squares in the columns of Fig. 3 corresponding to: $kdeg_{Hes1}$, Kp_{Hes}, tetaHe, kon_{cis}, $kdeg_{Jag}$, $kr_{jagNotch}$, kr_{cis}, $kf_{jagNotch}$, Kp_{Jag}, tetaJag, $kdeg_{iR2}$, kp we conclude their non-identifiability. Given this information, we can eliminate these parameters and re-frame the system down for the reduced first-order tableau in Fig. 4.

In Fig. 4, observe there are no further non-identifiable parameters. Moreover, consider parameters: kp_{R2}, kdp_{R2}, and $kdeg_{pR2}$. kp_{R2} is globally identifiable, as it is the only parameter that appears in the degree one Lie derivative Jacobian. Removing this parameter,

$\mathrm{kdp_{R2}}$ is identifiable as it is the only one that appears in degree three. Again removing this parameter, $\mathrm{kdeg_{pR2}}$ is identifiable as it appears in the degree 2 coeffecient. Hence we see that these three parameters are globally identifiable. The final resultant tableau is the second-order tableau Fig. 4.

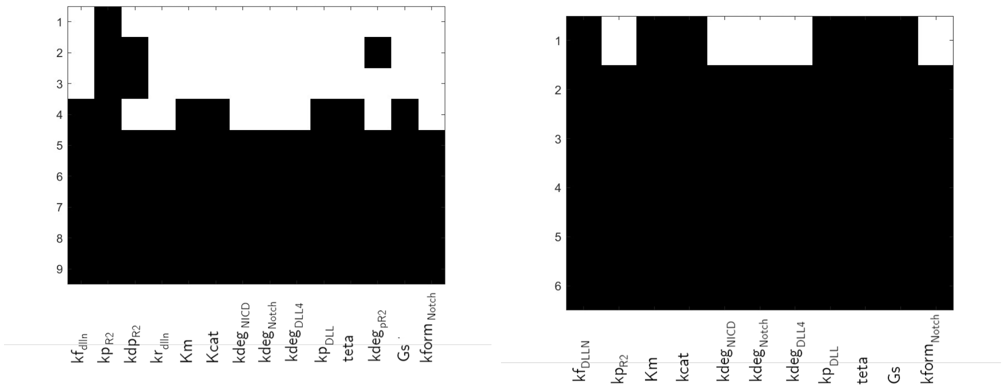


FIGURE 4. **Reduced First and Second Order Identifiability Tableau from GenSSI.** The resulting shading of the Jacobian matrix once non-identifiable parameters from the previous order tableau are fixed and Lie derivatives are recalculated.

There are no new clear non-identifiable or globally identifiable parameters, and the rest of the parameters are not locally identifiable due to a rank deficient system of equations. Hence, the identifiability summary of GenSSI, becomes Table 4:

Parameter	$\mathrm{kf_{dllN}}$	$\mathrm{kp_{R2}}$	$\mathrm{kdp_{R2}}$	$\mathrm{kr_{dllN}}$	Km	kcat	$\mathrm{kdeg_{NICD}}$	$\mathrm{kdeg_{Notch}}$	$\mathrm{kdeg_{Dll4}}$	$\mathrm{kp_{Dll}}$	teta	$\mathrm{kdeg_{Hes1}}$	$\mathrm{Kp_{Hes}}$
Identifiability	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗
Parameter	tetaHe	$\mathrm{kon_{cis}}$	$\mathrm{kdeg_{Jag}}$	$\mathrm{kr_{jagNotch}}$	$\mathrm{kr_{cis}}$	$\mathrm{kf_{jagNotch}}$	$\mathrm{Kp_{Jag}}$	tetaJag	$\mathrm{kdeg_{pR2}}$	$\mathrm{kdeg_{iR2}}$	Gs	$\mathrm{kform_{Notch}}$	kp
Identifiability	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗

TABLE 4. **Structural identifiability results from GenSSI.**

Finally we consider the application of STRIKE-GOLDD software with summary results in Table 5:

Parameter	$\mathrm{kf_{dllN}}$	$\mathrm{kp_{R2}}$	$\mathrm{kdp_{R2}}$	$\mathrm{kr_{dllN}}$	Km	kcat	$\mathrm{kdeg_{NICD}}$	$\mathrm{kdeg_{Notch}}$	$\mathrm{kdeg_{Dll4}}$	$\mathrm{kp_{Dll}}$	teta	$\mathrm{kdeg_{Hes1}}$	$\mathrm{Kp_{Hes}}$
Identifiability	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Parameter	tetaHe	$\mathrm{kon_{cis}}$	$\mathrm{kdeg_{Jag}}$	$\mathrm{kr_{jagNotch}}$	$\mathrm{kr_{cis}}$	$\mathrm{kf_{jagNotch}}$	$\mathrm{Kp_{Jag}}$	tetaJag	$\mathrm{kdeg_{pR2}}$	$\mathrm{kdeg_{iR2}}$	Gs	$\mathrm{kform_{Notch}}$	kp
Identifiability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗

TABLE 5. **Structural Identifiability of Notch model with STRIKE-GOLDD.**

Curiously, we observe that the results of STRIKE-GOLDD and StructuralIdentifiability.jl are the exact same when all assumed observables are given and no additional parameters are fixed. However, there are severe differences between GenSSI and the other two. We suspect this may be due to the initial conditions and non enough orders of Lie derivative calculated (an issue that requires significantly more computational power), hence we proceed with the results of StructuralIdentifiability.jl and STRIKE-GOLDD methods.

4. Parameter Estimation from Systems-Biology Informed Neural Networks

4.1. **Neural Networks in PDE Modeling.** Partial Differential Equations (PDEs) are ubiquitous in governing physical systems and are integral to modeling problems in diverse fields from systems-biology to economics [10]. However, standard numerical methods are either inaccurate or too computationally intensive, and machine learning (ML) has shown promise in universal function approximation. Specifically, physics-based methods such as Physics-Informed Neural Networks embedding physical constraints have shown wide implications. Here we introduce the field of physics-based ML of relevance to our model.

4.1.1. *Deep Learning.* There are several architectures in Deep Neural Networks that maybe employed, however we begin with a consideration of feed-forward Neural Networks (FNNs). We denote a L-layer FNN (see Fig. 5), with $(L-1)$ hidden layers and an output layer, by $\mathcal{N}^L(x) : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$, where $d_{\text{in}}, d_{\text{out}}$ are the dimensions of the input and output, respectively. Layer l contains N_l neurons (note $N_0 = d_{\text{in}}, N_L = d_{\text{out}}$). Each layer is fed input from the previous layer and is transformed according to affine transformation $T^l(x) = \mathbf{W}^l\mathbf{x} + \mathbf{b}^l$ and non-linear function σ. That is,

$$\mathcal{N}^l(x) = T^L \circ \sigma \circ T^{(L-1)} \circ \cdots \circ \sigma \circ T^1(x).$$

There are several possible choices for the activation function such as hyperbolic tangent or sigmoidal. We employ the swish activation function [28] in our methodology.

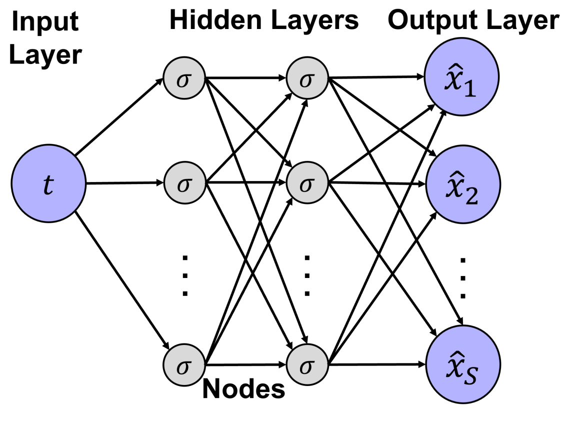


Figure 5. **FNN Architecture**. Input layer, t; hidden layers, with weights $\mathbf{W}^l$, bias $\mathbf{b}^l$, and activation σ); and output layer. Figure adapted with permission [7].

4.1.2. *Physics-Informed Neural Networks (PINNs).* Physics-informed Neural Networks (PINNs) were introduced in [27] by Raissi et al., employing neural network architecture to learn the underlying physics of a system through data and partial differential equations (PDEs). PINNs embed the PDE residual as a soft constraint by including it to the loss function of the base neural network, and the gradient is computed via automatic differentiation.

PINNs are well suited for so-called "inverse problems." Inverse problems in PDEs consist of reconstructing some part of a PDE (such as a coefficient, a boundary condition, an initial condition, the shape of a domain, or a singularity) from partial knowledge of solutions to the PDE [3]. Begin by defining an inverse problem over domain $\Omega \subset \mathbb{R}^d$,

$$\mathcal{F}[\mathbf{u}(\mathbf{x}); \gamma(\mathbf{x})] = 0, \qquad \mathbf{x} \in \Omega$$

with boundary conditions,

$$\mathcal{B}[u(\mathbf{x})] = 0, \qquad \mathbf{x} \in \partial\Omega$$

where $\mathcal{F}$ is a system of PDE operators and $\mathcal{B}$ is the boundary operator. The system yields vector solution $\mathbf{u}(\mathbf{x})$ and some optimal γ. This function, mapping, or parametrization γ is the core of the inverse paradigm and the quantity of interest (QoI). We search for the best γ by minimizing an objective function $\mathcal{J}$ that depends on $\hat{\mathbf{u}}$ and $\hat{\gamma}$. The inverse design problem is formulated as an optimization problem:

$$\min_{\mathbf{u},\gamma} \mathcal{J}(\mathbf{u}; \gamma).$$

To use PINNs, we employ n FNNs $\hat{\mathbf{u}}(\mathbf{x}; \boldsymbol{\theta}_u), \hat{\gamma}(\mathbf{x}; \boldsymbol{\theta}_\gamma)$ to approximate $\mathbf{u}, \gamma$ where $\boldsymbol{\theta}_u, \boldsymbol{\theta}_u$ is the set of trainable parameters in the network. The network takes the coordinates $\mathbf{x}$ as the input and outputs the approximate solution $\hat{\mathbf{u}}(\mathbf{x})$. The loss quantifies the innate constraint of the PDE through sampling of M residual points over the N PDEs as,

$$\mathcal{L}_{\mathcal{F}}(\boldsymbol{\theta}_u, \boldsymbol{\theta}_\gamma) = \frac{1}{MN} \sum_{j=1}^{M} \sum_{i=1}^{N} \left| \mathcal{F}_i \left[\hat{\mathbf{u}}(\mathbf{x}_j); \hat{\gamma}(\mathbf{x}_j) \right] \right|^2,$$

where $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ are a set of M residual points in the domain Ω, and $|\mathcal{F}_i [\hat{\mathbf{u}}(\mathbf{x}_j); \hat{\gamma}(\mathbf{x}_j)]|$ measures the discrepancy of the i-th PDE $\mathcal{F}_i[\mathbf{u}; \gamma] = 0$ at the residual point $\mathbf{x}_j$.

The training of such a neural network follows identical to a regular FNNs with appropriate optimizers (SGD, Adam, L-BFGS, etc...) and regularization techniques prevalent in standard machine learning to approximate $u(\mathbf{x})$. Through classical methods, we recover an approximation of the PDE solution $\mathbf{u}$ and determine the unknown quantity of interest γ to arbitrary convergence.

4.2. **Systems-Biology Informed Neural Network (SBINN).**

SBINN was proposed in [39], using systems-biological models as ODE systems for PINN architectures. Given a system described in Eq. (2), the aim is to use neural networks of parameters $\boldsymbol{\theta}$ (not to be confused with parameters $\boldsymbol{\Theta}$ of the dynamic system) that take time t as input and act as a surrogate model, outputting state variables $\hat{\mathbf{x}}(t; \boldsymbol{\theta}) = (\hat{x}_1(t; \boldsymbol{\theta}), \hat{x}_2(t; \boldsymbol{\theta}), \cdots, \hat{x}_n(t; \boldsymbol{\theta}))$ in place of ODE solution $\mathbf{x}(t)$. In addition to the core DNN, the SBINN architecture also has an Input-scaling layer, a Feature layer, and an Output-scaling layer (Fig. 6).

- Input-scaling layer: The input time domain varies my many orders of magnitude, so linearly scaling the time component via $\tilde{t} = t/T$ for some predefined T or normalizing T such that $t \sim \mathcal{O}(1)$.
- Feature layer: In many systems models, the ODE solution will have specific patterns (e.g. periodicity). To construct a better surrogate, it is beneficial to use a feature layer incorporating these patterns as hard conditions. We do this by adding functions

$e_1, e_2, \ldots, e_L$ for corresponding features $e_1(\tilde{t}), e_2(\tilde{t}), \ldots, e_L(\tilde{t})$. This method just helps the model fit, and the choice and necessity of the functions is problem-dependent.

- Output-scaling layer: The state variables in the output are also of varying magnitudes, so we scale the output of the hidden layers $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_S$ (of order 1) by $k_1, k_2, \ldots, k_S$ respectively, where k_i is the mean value of x_i.

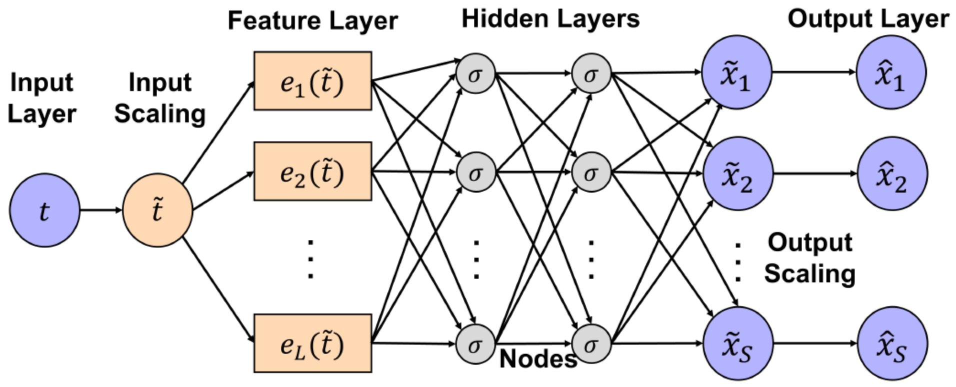


FIGURE 6. **Neural network architecture for SBINNs** [39]. The input is scaled to order one, fed into a feature layer for projection, run through a FNN, and scaled for proper output. Figure adapted with permission [7].

To train the neural network, we must enforce convergence of both the state variables and the ODE predicted dynamics (the systems-biology portion). We define a loss function, computing the mean square error of the neural network predictions and the desired behavior over provided observations at times $t_1, t_2, \ldots, t_{N^{data}}$ and the ODE predictions at times $\tau_1, \tau_2, \ldots, \tau_{N^{data}}$.

For the definitions of the loss functions, we use the notation from Eq. (1). Usually for observation of biological systems, the set of observables is a subset of the state variables, with Gaussian noise. That is, $\mathbf{y} = \mathbf{x}_s + \boldsymbol{\epsilon}$, where $\mathbf{x}_s$ is a subset of the state variables $\mathbf{X}$. Then the output of the neural network model is $\hat{\mathbf{x}}_s$. First, we define $\mathcal{L}^{data}$ for M sets of observations of $\mathbf{y}$:

$$\mathcal{L}^{data}(\boldsymbol{\theta}) = \sum_{m=1}^{M} w_m^{data} \left[\frac{1}{N^{data}} \sum_{n=1}^{N^{data}} \left(\mathbf{y}_m(t_n) - \hat{\mathbf{x}}_{s_m}(t_n; \boldsymbol{\theta})\right)^2 \right].$$

for trainable parameters $\boldsymbol{\theta}$ and weights w_i^{data} for $1 \leq i \leq m$. Similarly, we define a loss for the dynamics of the state variables, $\mathcal{L}^{ode}$, over sampled times,

$$\mathcal{L}^{ode}(\boldsymbol{\theta}, \boldsymbol{\Theta}) = \sum_{i=1}^{S} w_s^{ode} \left[\frac{1}{N^{ode}} \sum_{n=1}^{N^{ode}} \left(\frac{d\hat{x}_i}{dt}|_{\tau_n} - \mathbf{f}_i\left(\hat{x}_i(\tau_n; \boldsymbol{\theta}), \tau_n; \boldsymbol{\Theta}\right) \right)^2 \right].$$

We use Automatic differentiation (AD) [21] to compute the derivative $\frac{d\hat{x}_s}{dt}|_{\tau_n}$. Finally, define the *auxillary* loss function to include additional infromation of system identification. In this case, we assume the measurements of the state variables at two times T_0, T_1, and calculate:

$$\mathcal{L}^{aux}(\boldsymbol{\theta}) = \sum_{i=1}^{S} w_s^{aux} \frac{(x_i(T_0) - \hat{x}_i(T_0;\boldsymbol{\theta}))^2 + (x_i(T_1) - \hat{x}_i(T_1;\boldsymbol{\theta}))^2}{2}.$$

Therefore the final loss function is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Theta}) = \mathcal{L}^{data}(\boldsymbol{\theta}) + w_{ode}\mathcal{L}^{ode}(\boldsymbol{\theta}, \boldsymbol{\Theta}) + w_{aux}\mathcal{L}^{aux}(\boldsymbol{\theta}).$$

Observe that the inclusion of system parameters $\boldsymbol{\Theta}$ allows for the model to train these parameters as well in inverse fashion. Note the weights are chosen such that each loss function has same order of contribution to the total loss function.

Now that the loss function is set up, we want to optimize the parameters: both θ, Θ of the neural networks and the system dynamics respectively. To minimize the loss function, we use a gradient-based optimizer such as Adam [18]:

$$\boldsymbol{\theta}^*, \boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Theta}).$$

This method yields both a PINN prediction solution (using the trained model) as well as an inferred solution from the trained system parameters.

4.2.1. *Warm-Starting.* In this methodology, we employ warm-starting PINNs (WS-PINNs) and transfer learning to achieve better convergence of results in a our high dimensional system. Training of WS-PINNs models are divided into two stages. Initially, a DNN is trained only on our synthetic observable data without considering the constraints form the underlying ODE system. This is the 'warm-up' stage for the data which helps the network converges more quickly since physics-based additional loss terms do not need to be computed while simultaneously approximating a solution. For low resolution data, iterations of data warm-up prevent PINN overfitting. In the case of high resolution data (as is our synthetic), too many iterations of warm-up may compromise the PINN resolution required to elucidate the hidden dynamics. In this work, we use warm the data on the order of magnitude of 10^4 iterations. Once the warm-up stage is complete, we transfer the model to be trained now with the additional loss functions. Here, the model tunes to the physics of the system to improve accuracy and generalization.

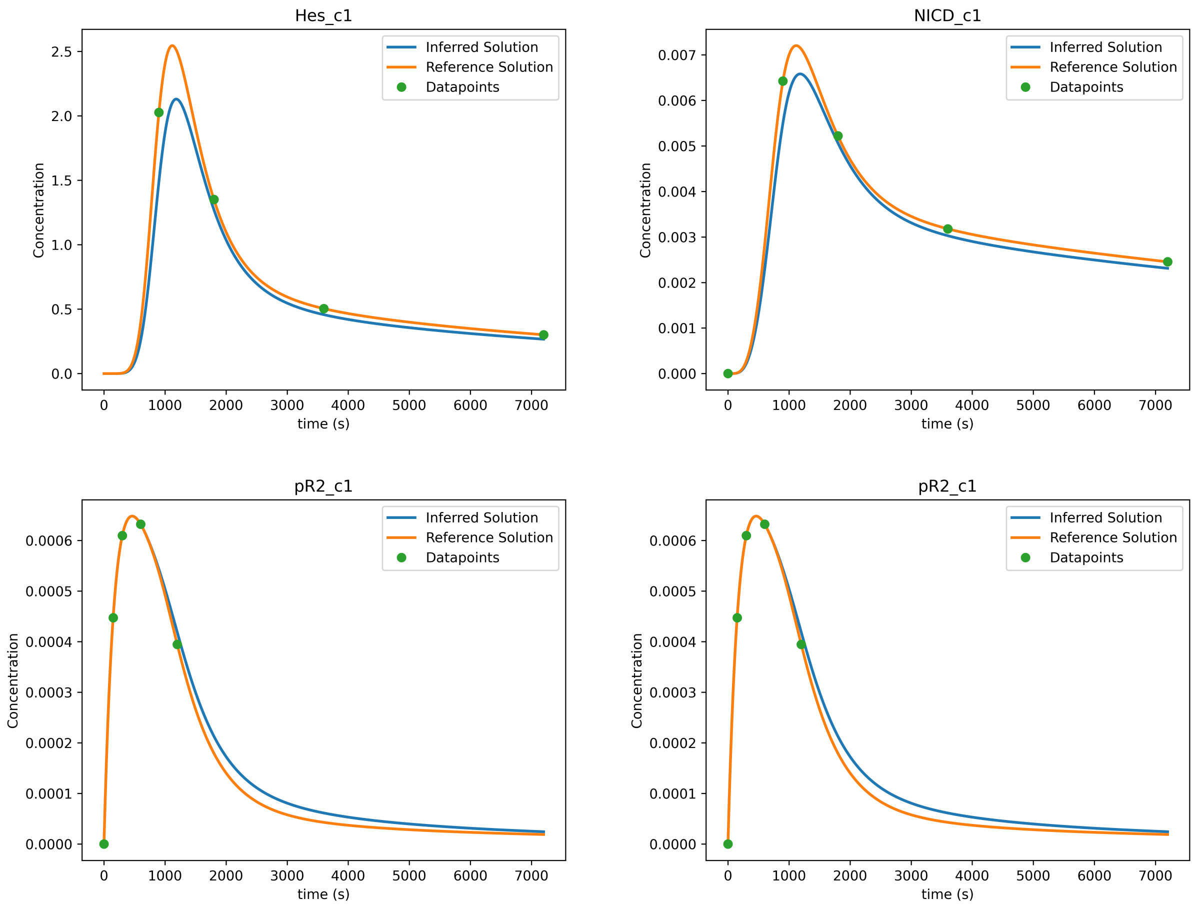
4.3. **Implementation and Results of SBINNs for Notch Model.**

To apply the SBINN methodology to the Notch model, we use the Python open source library DeepXDE [21] with a TensorFlow computational backend.

After obtaining a set of data for training, we begin by employing a transfer learning approach to ensure that our eventual SBINN has weights closer to the solution in the loss landscape. For this paper, we train this model to predict only the structurally and practically identifiable parameters $\mathrm{k_{fDllN}}$, $\mathrm{k_{pR2}}$, $\mathrm{kdeg_{Notch}}$, $\mathrm{kdeg_{Dll4}}$, θ, $\mathrm{kdeg_{Notch}}$, and $\mathrm{kdeg_{pR2}}$. The transfer learning model is a parallel FNN that utilizes independent sub-networks for each network output. The architecture consists of 8 layers: an input layer, 6 hidden layers with 128 neurons each, and an output layer with 22 neurons corresponding to the desired predicted outputs. The swish activation function is employed to mitigate the dying ReLU problem during early training stages. Additionally, L^2 regularization is applied to the data, with a regularization parameter $\lambda = 10^{-8}$, considering the scale of the data. The regularization parameter is kept small, as proper training will be conducted after the completion of this base model.

Data-driven techniques are utilized to shape the feature and output transformations. Prior to passing the input data $\vec{x}$ to the model, it undergoes a transformation defined by $f(\vec{x}) = \ln\left(\vec{x} + \frac{1}{e}\right)$. The logistic function allows for increased differentiation of smaller numbers. For the output transformation, we select a value b from the training data for Dll4$_{c1}$ to implement a hard constraint based on the initial conditions. Before outputting the model's vector $\vec{y}$, it is flattened and transformed using $o(t, \vec{x}) = b + \tanh\left(\frac{t}{500}\right)(\text{abs}(\vec{x}) - (1 + b))$, where $\text{abs}(\vec{x})$ returns the component-wise absolute value of a tensor, as the values of interest cannot be negative. The tanh function is employed to avoid a significant jump at time $t = 1$, ensuring that early time points are not overemphasized.

Finally, the weights are initialized according to the Glorot normal distribution [13]. The model is then trained first in the warm-starting phase and then transferred to the physically constrained phase. The results of training these variables are presented in Appendix A, and we show the ones with data here. We can see that the model fits relatively well in Fig. 6. The loss curves are displayed in Fig. 10 indicating promising results as to the convergence of the system after the initial warm-up phase.



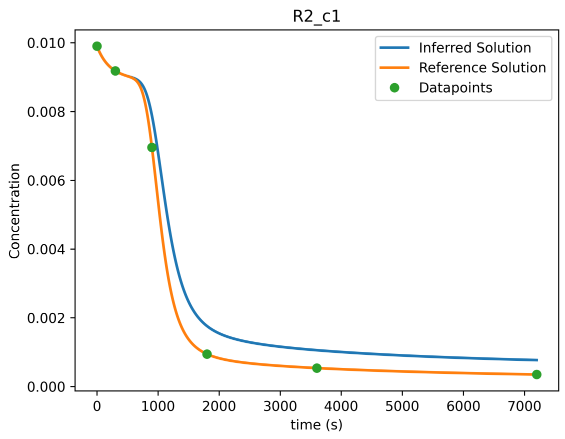


FIGURE 6. Forecasts of (A) $\mathrm{Hes}_{\mathrm{c1}}$, (B) $\mathrm{NICD}_{\mathrm{c1}}$, (C) $\mathrm{Notch1}_{\mathrm{c1}}$, (D) $\mathrm{pR2}_{\mathrm{c1}}$, and (E) $\mathrm{R2}_{\mathrm{c1}}$ predicted by the SBINN model.

Parameter extraction has not yet reached desired precision. Due to the large magnitude of the system in question, as compared to previously successful parameter identification tasks [7], we conclude that our methods here are not yet adequate to tackle the larger problem. We are actively pursuing such a solution expanding upon this outlined methodology.

Practical Identifiability

Practical identifiability concerns the quality of the training data. After training, the inferred parameters may have infinite confidence intervals. Practical identifiability specifically determines the sensitivity of the parameter values to noise in the data and is performed *a posteriori*.

Definition 3. *Fisher Information* is the measure of the amount of information that an observable random variable X carries about an unknown parameter Θ upon which the probability of X depends:

$$\mathcal{I}(\theta) = -\operatorname{E}\left[\frac{\partial^2}{\partial\theta^2}\log f(X;\theta)\,\middle|\,\theta\right],$$

given probability density function $f(X;\theta)$. The *Fisher Information Matrix* (FIM) is given as:

$$\big[\mathcal{I}(\theta)\big]_{i,j} = \operatorname{E}\left[\left(\frac{\partial}{\partial\theta_i}\log f(X;\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log f(X;\theta)\right)\,\middle|\,\theta\right].$$

FIM can be used in two analyses: first is determine the correlation matrix of parameters and the second is to determine null eigenvalues (those with zero eigenvector) of the FIM.

The correlation matrix, R, is calculated from the FIM matrix, called FIM, as:

$$R_{ij} = \frac{\mathrm{FIM}_{ij}^{-1}}{\mathrm{FIM}_{ii}^{-1}}.$$

$|R_{ij}| \approx 1$ indicates high correlation between two parameters and and indicates that the parameters are not individually identifiable between each other.

Next, we compute null eigenvalues and eigenvectors of FIM. If the value of a component and all other components are approximately zero, the associated parameter has little to no effect on state variables and is therefore practically unidentifiable from the dataset.

4.4. **Sensitivity.** Sensitivity is another method of practical identifiability analysis that determines the impact of uncertainty in the inputs of a model on quantities of interest. In this study, we consider the impact of artificial noise in the observable data on the parameter specification to understand how well a parameter is able to converge. There are two classifications of sensitivity analysis methods: local and global.

The most common local sensitivity analysis method is One-at-a-time (OAT) analysis. Local sensitivity for some parameter θ_k for a model $f(x, \theta)$ is generally approximated via $\frac{\partial f}{\partial \theta_k}$ already at the pre-determined solution using stanard OAT methods.

However, when we are in midst of training and only have access to a range of possible parameter solutions, global sensitivity must be employed. We use variance decompositions of the parameter estimates (using ANOVA or the likes) and calculate Sobol coefficients [31]. The higher the coefficient, the more a parameter is impacted by noise in input data and thereby it has a higher ability to train.

We propose two ideas for employing sensitivity, not only as a *posteriori* measure, but as an integrated portion of the pipeline. First is to vary the learning rate of various parameters in accordance to their Sobol coefficient in order to accelerate training of certain parameters while preventing non-convergence of others. The second is to halt training of certain parameters once convergence has been achieved in order to devote computational power to slower parameters.

4.5. **Practical Identifiability Analysis of Notch Results.** We implement Practical Identifiability in Julia and generate the FIM and correlation matrix. The parameter in Fig. 7 are enumerated starging from 0: $\mathrm{kf}_{\mathrm{dllN}}$, $\mathrm{kp}_{\mathrm{R2}}$, $\mathrm{kdp}_{\mathrm{R2}}$, $\mathrm{kr}_{\mathrm{dllN}}$, Km, kcat, $\mathrm{kdeg}_{\mathrm{NICD}}$, $\mathrm{kdeg}_{\mathrm{Notch}}$, $\mathrm{kdeg}_{\mathrm{Dll4}}$, $\mathrm{kp}_{\mathrm{Dll}}$, teta, $\mathrm{kdeg}_{\mathrm{Hes1}}$, $\mathrm{Kp}_{\mathrm{Hes}}$, tetaHe, $\mathrm{kon}_{\mathrm{cis}}$, $\mathrm{kdeg}_{\mathrm{Jag}}$, $\mathrm{kr}_{\mathrm{jagNotch}}$, $\mathrm{kr}_{\mathrm{cis}}$, $\mathrm{kf}_{\mathrm{jagNotch}}$, $\mathrm{Kp}_{\mathrm{Jag}}$, tetaJag, $\mathrm{kdeg}_{\mathrm{pR2}}$, $\mathrm{kdeg}_{\mathrm{iR2}}$, Gs, $\mathrm{kform}_{\mathrm{Notch}}$, kp

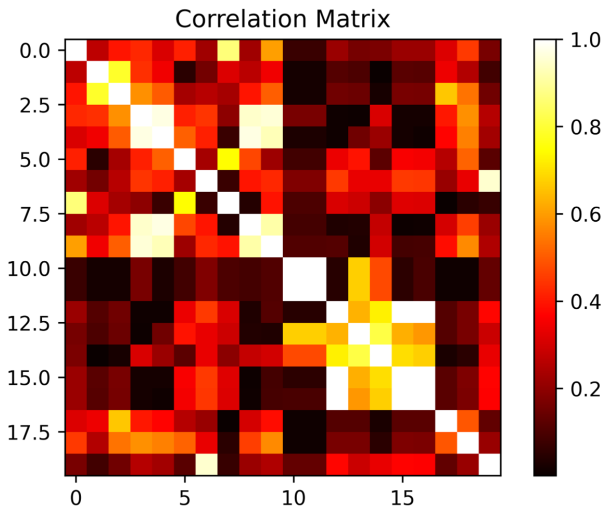


FIGURE 7. **Correlation matrix of parameters from FIM.**

From the correlation matrix, we see that there exist certain pairs of parameters that are correlated (e.g $\text{kon}_{\text{cis}}, \text{kdeg}_{\text{jag}}$). This allows us to infer that if a parameter fit to the assumed value (via other literature and knowledge of the system), then the paired value can be inferred to fit as well. For the general practical identifiability of the system, we consider the Eigenvalue analysis of the FIM, Fig. 8.

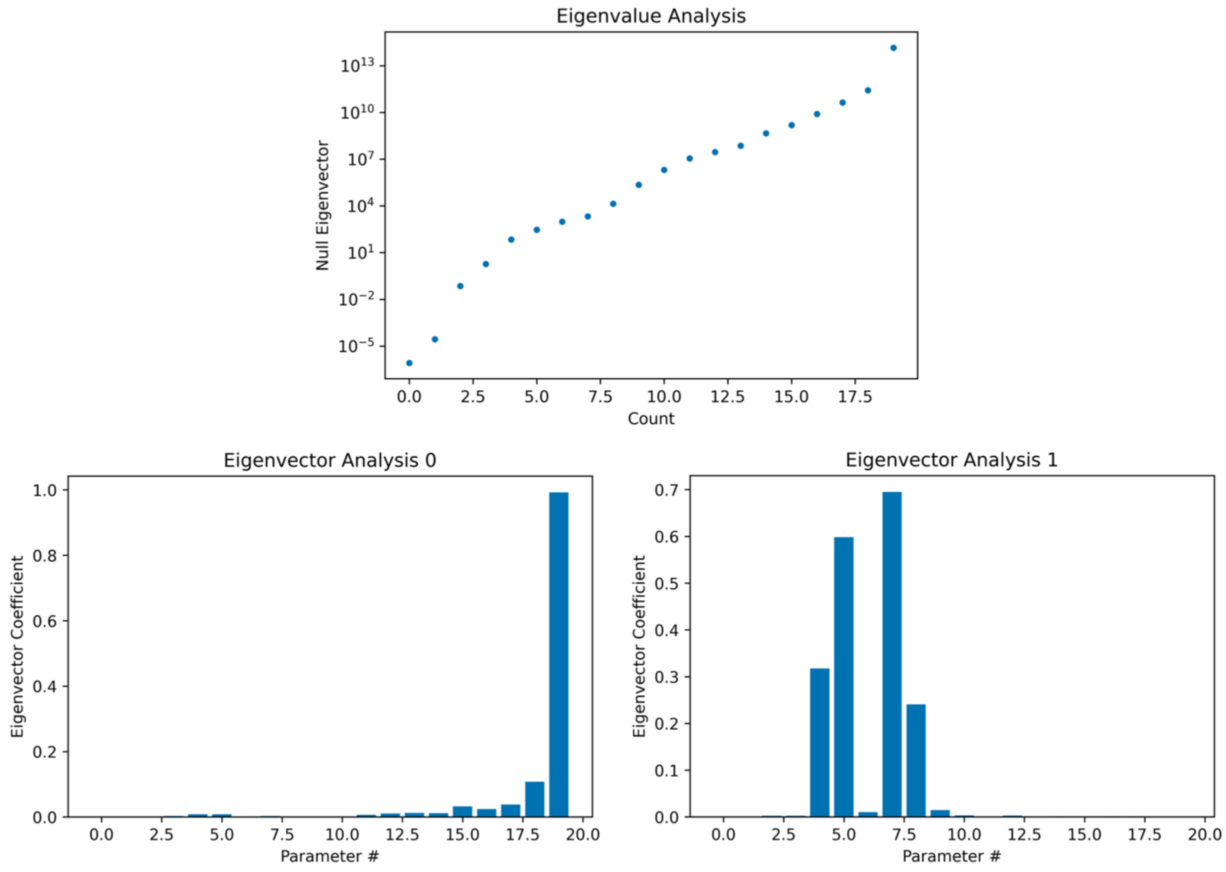


FIGURE 8. **Null eigenvector analysis.** (A) Eienvalues of the FIM, (B) Eigenvector for first null eigenvalue, (C), Eigenvector for second null eigenvalue.

From the upper graph of Fig. 8, we observe only two null eigenvalues. From there, we determine the eigenvectors as the left and right figures. $\text{Kform}_{\text{Notch}}$ may not be identifiable according to Eigenvector Analysis 0. Due to the large component in the eigenvector, we can infer the sensitivity is smaller. Furthermore, $\text{Kdeg}_{\text{NICD}}$ may also not be identifiable, but other components are also quite large so it is hard to confirm. For the purpose of this analysis, we take $\text{Kform}_{\text{Notch}}$ as the practically non-identifiable prediction. Indeed, thus far, we have not been able to train to recover this parameter.

5. Conclusion

Solving the inverse problem of the Notch model, we recover parameter governing the neurogenesis processes. This information has the potential to identify key factors influencing neurogenesis, such as gene expression patterns or signaling pathways, which can guide experimental design and lead to targeted interventions.

We invite the community to continue our research in forecasting and estimating the parameters of the Notch model using the SBINN and employing other machine learning techniques to improve accuracy. We are considering methods such as: additionally sensitivity-based methods, regularization, and adding artificial constraints that have proven to increase training accuracy and efficiency in other contexts.

While the current SBINN architecture has shown promise in modeling neurogenesis, it is important to explore modern architectures, such as Physically Informed Transformers (PINNformers) [40] and Kolmogorov-Arnold Networks (KANs) [20], to enhance the modeling capabilities. PINNformers, which combine physics-informed neural networks with transformer architectures, have shown potential for modeling sequential time-based data. This type of work has been considered in [30], where they specifically evaluated the performance and output of KANs against PINNs. They utilize a new architecture to solve these problems. However, KANs currently face challenges in training speed due to CUDA not being optimized to train them in parallel compared to the current state-of-the-art MLPs. Combining different architectures and techniques could lead to a more comprehensive and accurate modeling approach for neurogenesis.

Consequently, interdisciplinary collaboration between computational modelers and neuroscientists is crucial to advance our understanding of neurogenesis. Close collaboration allows for a better understanding of the biological problem at hand and ensures that computational models are grounded in experimental evidence. Neuroscientists can provide valuable insights into the relevant biological processes, help identify key questions to address, and guide the selection of appropriate modeling approaches. In turn, computational models can generate testable hypotheses and guide the experimental design, leading to a more targeted and efficient biological research process.

Acknowledgements

Alex Huang, Kartik Ramachandrula, Agniv Sarkar, and Lu Lu kindly thank the MIT PRIMES program under which this research was conducted. We also thank the Emergent Ventures fund for their generous grant.

References

[1] Jessica L Ables, Joshua J Breunig, Amelia J Eisch, and Pasko Rakic. Not(ch) just development: Notch signalling in the adult brain. *Nature Reviews Neuroscience*, 5:269–283, 5 2011.

[2] Eva Balsa-Canto, Antonio A Alonso, and Julio R Banga. An iterative identification procedure for dynamic modeling of biochemical networks. *BMC systems biology*, 4:1–18, 2010.

[3] Liliana Borcea, Thorsten Hohage, and Barbara Kaltenbacher. Computational inverse problems for partial differential equations. *Oberwolfach Reports*, 17(4):1903–1954, 2020.

[4] Daipeng Chen, Zary Forghany, Xinxin Liu, Haijiang Wang, Roeland M.H. Merks, and David A. Baker. A new model of notch signaling: Control of notch receptor 2 cis-inhibition via notch ligand dimers. *bioRxiv*, 5 2022.

[5] Oana-Teodora Chis, Julio R Banga, and Eva Balsa-Canto. Genssi: a software toolbox for structural identifiability analysis of biological models. *Bioinformatics*, 27:2610–2611, November 2011.

[6] Oana-Teodora Chis, Julio R Banga, and Eva Balsa-Canto. Structural identifiability of systems biology models: a critical comparison of methods. *PLoS ONE*, 6(11):e27755, November 2011.

[7] Daneker, Mitchell and Zhang, Zhen and Karniadakis, George Em, and Lu, Lu. Systems biology: Identifiability analysis and parameter identification via systems-biology informed neural networks, 2022.

[8] Ruiwen Dong, Christian Goodbrake, Heather A Harrington, and Gleb Pogudin. Differential elimination for dynamical models via projections with applications to structural identifiability, 2022.

[9] Franz Oswald et al. p300 acts as a transcriptional coactivator for mammalian notch-1. *Molecular and Cellular Biology*, 21:7761–7774, 11 2001.

[10] Benjamin Fan, Edward Qiao, Anran Jiao, Zhouzhou Gu, Wenhao Li, and Lu Lu. Deep learning for solving and estimating dynamic macro-finance models. *Computational Economics*, pages 1–37, 2024.

[11] Gareth W Fearnley, Gina A Smith, Izma Abdul-Zani, Nadira Yuldasheva, Nadeem A Mughal, Shervanthi Homer-Vanniasinkam, Mark T Kearney, Ian C Zachary, Darren C Tomlinson, Michael A Harrison, et al. Vegf-a isoforms program differential vegfr2 signal transduction, trafficking and proteolysis. *Biology open*, 5(5):571–583, 2016.

[12] Jason E Fish, Manuel Cantu Gutierrez, Lan T Dang, Nadiya Khyzha, Zhiqi Chen, Shawn Veitch, Henry S Cheng, Melvin Khor, Lina Antounians, Makon-Sébastien Njock, Emilie Boudreau, Alexander M Herman, Alexander M Rhyner, Oscar E Ruiz, George T Eisenhoffer, Alejandra Medina-Rivera, Michael D Wilson, and Joshua D Wythe. Dynamic regulation of vegf-inducible genes by an erk/erg/p300 transcriptional network. *Development*, 144(13), 7 2017.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[14] Iva Greenwald. Notch and the awesome power of genetics. *Genetics*, 191:655–669, 7 2012.

[15] Nanae Izumi, Christian Helker, Manuel Ehling, Axel Behrens, Wiebke Herzog, and Ralf H Adams. Fbxw7 controls angiogenesis by regulating endothelial notch activity. *PloS one*, 7(7):e41116, 2012.

[16] Rudolf E. Kálmán. Contributions to the theory of optimal control. 1960.

[17] B Kiec-Wilk, J Grzybowska-Galuszka, A Polus, J Pryjma, A Knapp, and K Kristiansen. The mapk-dependent regulation of the jagged/notch gene expression by vegf, bfgf or ppar gamma mediated angiogenesis in huvec. *Journal of physiology and pharmacology: an official journal of the Polish Physiological Society*, 61(2):217–25, 4 2010.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Zhao-Jun Liu et al. Regulation of notch1 and dll4 by vascular endothelial growth factor in arterial endothelial cells: Implications for modulating arteriogenesis and angiogenesis. *Molecular and Cellular Biology*, 23(1):14–25, 2003. PMID: 12482957.

[20] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[21] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[22] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[23] Otto L. Mohr. Character changes caused by mutation of an entire region of a chromosome in drosophila. *Genetics*, 4:275–282, 5 1919.

[24] Rebeca Hannah de Melo Oliveira, Brian H Annex, and Aleksander S Popel. Endothelial cells signaling and patterning under hypoxia: a mechanistic integrative computational model including the notch-dll4 pathway. *Frontiers in Physiology*, 15:1351753, 2024.

[25] Chelsea Paresi. *The Mechanism of Action of Novel GAMMA-Secretase Modulators (thesis).* PhD thesis, Weill Cornell Graduate School of Medical Sciences, 2016.

[26] Hannu Pohjanpalo. System identifiability based on the power series expansion of the solution. *Mathematical biosciences*, 41(1-2):21–33, 1978.

[27] 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*, 378:686–707, 2019.

[28] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[29] Timothy Sargis, Seock-Won Youn, Krishna Thakkar, L A Naiche, Na Yoon Paik, Kostandin V Pajcini, and Jan K Kitajewski. Notch1 and notch4 core binding domain peptibodies exhibit distinct ligand-binding and anti-angiogenic properties. *Angiogenesis*, 26(2):249–63, 5 2023.

[30] Khemraj Shukla, Juan Diego Toscano, Zhicheng Wang, Zongren Zou, and George Em Karniadakis. A comprehensive and fair comparison between mlp and kan representations for differential equations and operator networks. *arXiv preprint arXiv:2406.02917*, 2024.

[31] I.M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271–280, 2001. The Second IMACS Seminar on Monte Carlo Methods.

[32] Min Song and Stacey D Finley. Mechanistic insight into activation of mapk signaling by pro-angiogenic factors. *BMC systems biology*, 12:1–17, 2018.

[33] Kyosuke Takeshita, Minoru Satoh, Masaaki Ii, Marcy Silver, Florian P Limbourg, Yasushi Mukai, Yoshiyuki Rikitake, Freddy Radtke, Thomas Gridley, Douglas W Losordo, et al. Critical role of endothelial notch1 signaling in postnatal angiogenesis. *Circulation research*, 100(1):70–78, 2007.

[34] Mathukumalli Vidyasagar. *Nonlinear systems analysis.* SIAM, 2002.

[35] Alejandro F. Villaverde, Antonio Barreiro, and Antonis Papachristodoulou. Structural identifiability of dynamic systems biology models. *PLOS Computational Biology*, 12(10):1–22, 10 2016.

[36] Eric Walter and Yves Lecourtier. Global approaches to identifiability testing for linear and nonlinear state space models. *Mathematics and Computers in Simulation*, 24(6):472–482, 1982.

[37] Chongjuan Wei, Varun Kumar Bhattaram, John C. Igwe, Elizabeth Fleming, and Jennifer S. Tirnauer. The lkb1 tumor suppressor controls spindle orientation and localization of activated ampk in mitotic epithelial cells. *PLOS ONE*, 7(7):1–10, 07 2012.

[38] Qianhui Wu and Stacey D Finley. Mathematical model predicts effective strategies to inhibit vegf-enos signaling. *Journal of Clinical Medicine*, 9(5):1255, 4 2020.

[39] Alireza Yazdani, Lu Lu, Maziar Raissi, and George Em Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLOS Computational Biology*, 16(11):1–19, 11 2020.

[40] Zhiyuan Zhao, Xueying Ding, and B Aditya Prakash. Pinnsformer: A transformer-based framework for physics-informed neural networks. *arXiv preprint arXiv:2307.11833*, 2023.

The Harker School, San Jose, CA 95129
Email address: 25alexh@students.harker.org

Brookfield Central High School, Brookfield, WI 53005
Email address: 68kartik@gmail.com

University of Oxford, Oxford OX1 2JD, UK
Email address: agni.spare@gmail.com

Department of Statistics and Data Science, Yale University, New Haven, CT 06511
Email address: lu.lu@yale.edu

Appendix A. Figures

Here, the inferred solution is the solution obtained after the forward pass of a model solver. PINN prediction is the forward pass of our model. Reference solution is generated from our data, and inferred solution is the solution that the final neural approximation of the system.

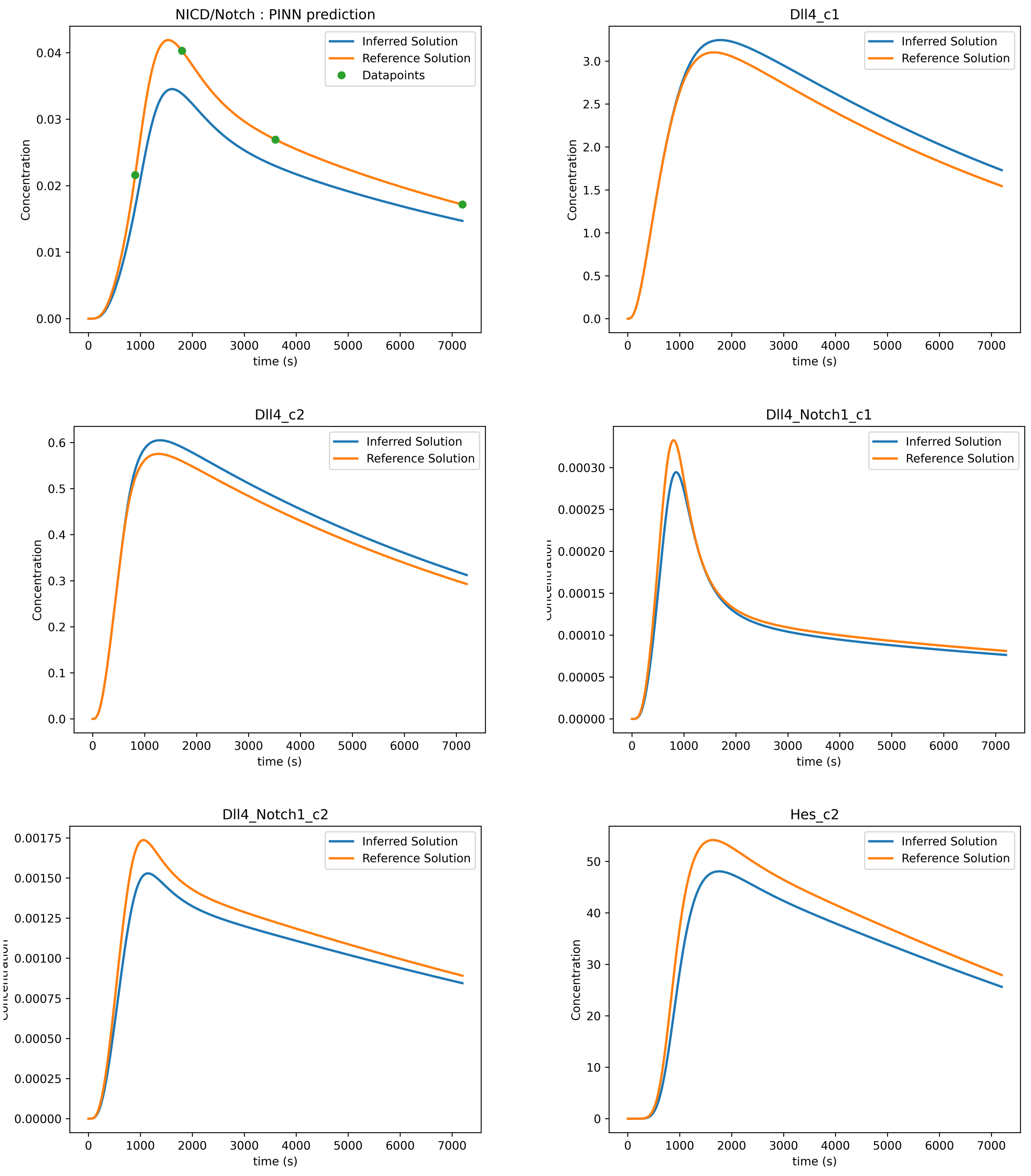


Figure 9. Predictions for other parameters which describe the system.

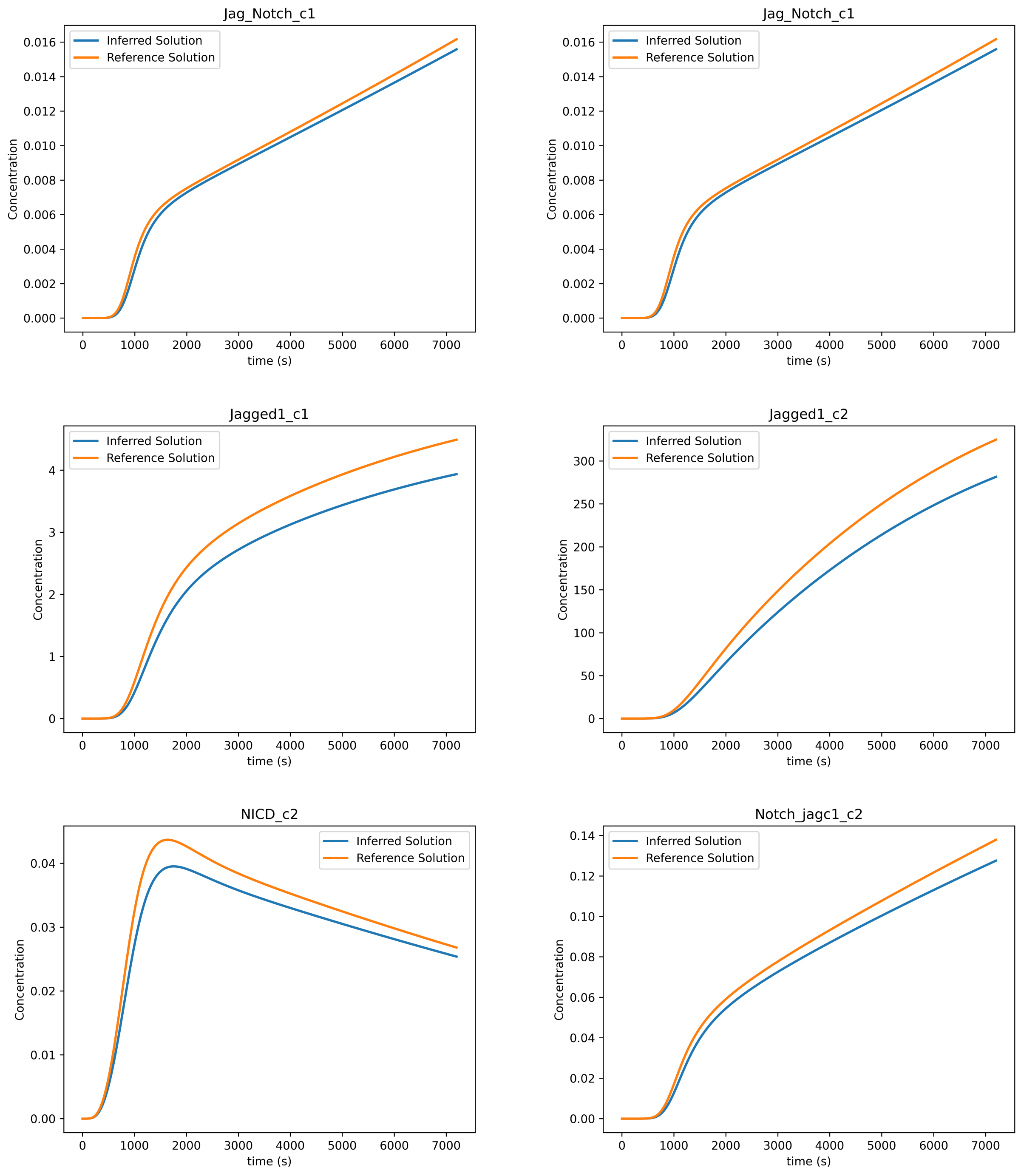


FIGURE 9. Predictions for other parameters which describe the system.

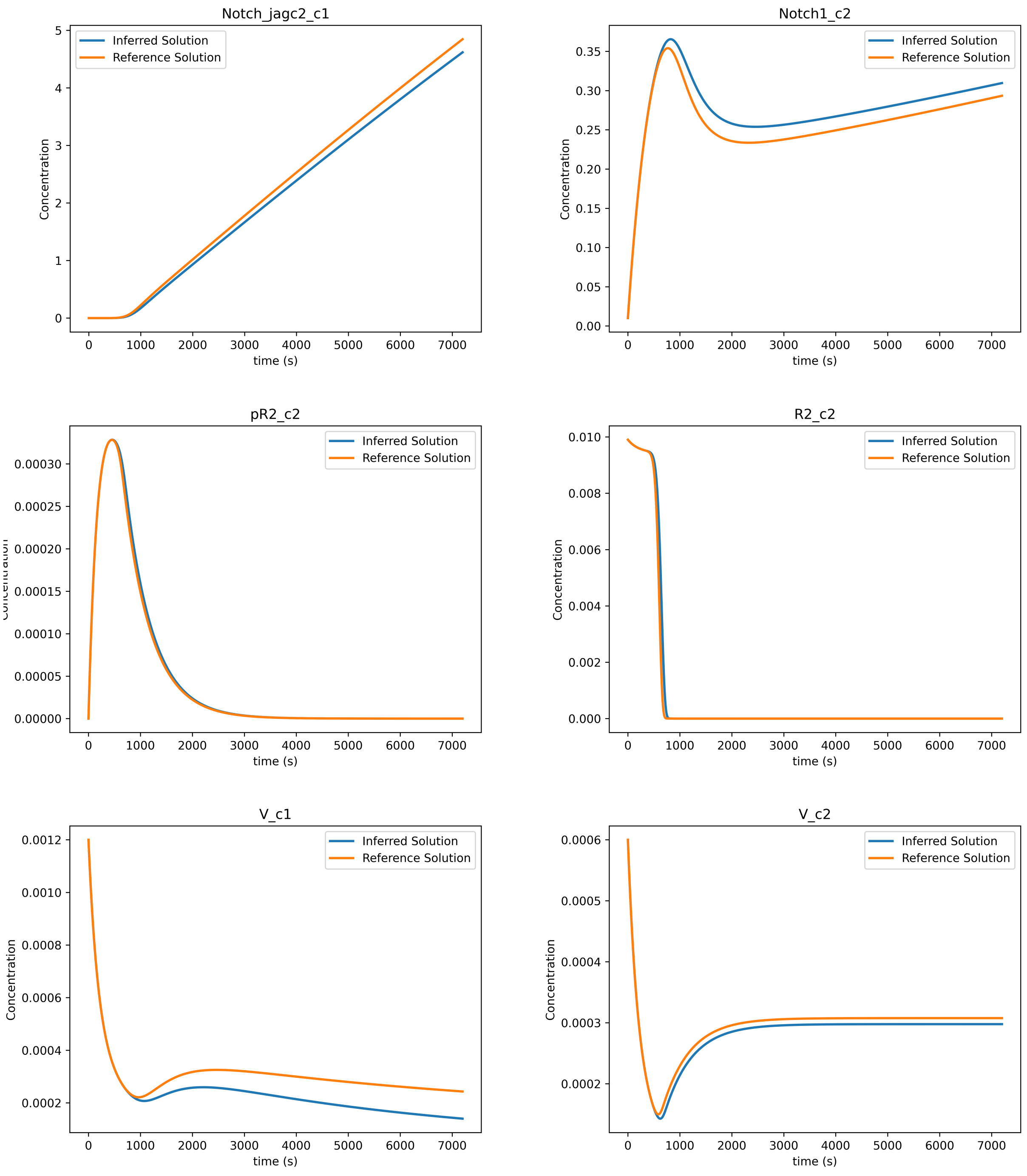
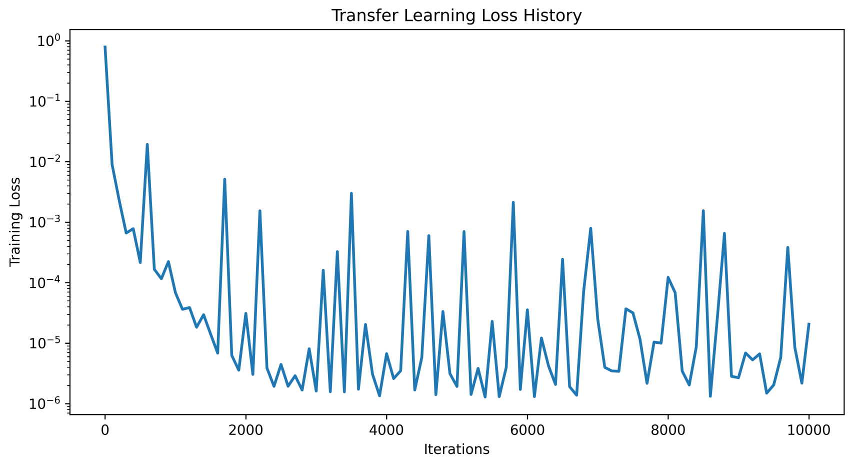


Figure 9. Predictions for other parameters which describe the system.

The loss curve during the training process of the transfer learning model is illustrated in Fig. 10. After completing the transfer learning phase, we proceeded to train our final SBINN model. The architecture of this model is the same as the transfer learning model, though this time with the added loss from the ODE residuals. The training process of the final SBINN model yielded promising results. The loss curve during training of the SBINN is presented in Fig. 10.



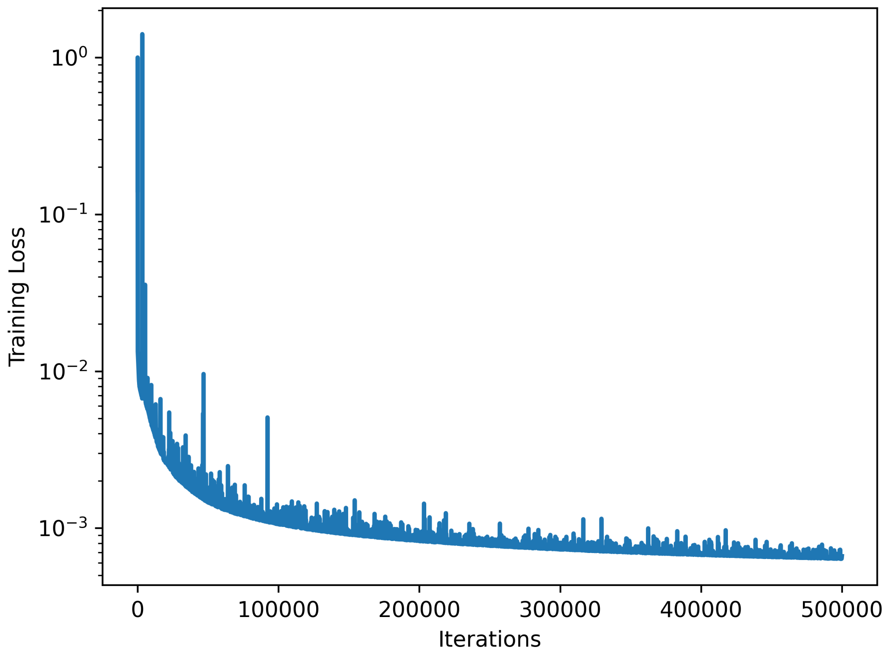


FIGURE 10. **Loss Curves** During the training of the transfer learning model (left) and during the training of the SBINN (right).

APPENDIX B. NOTCH MODEL PARAMETERS, OBSERVABLES, AND STATE VARIABLES

The Notch model has five observables, 22 state variables, and 26 parameters. As Notch is a singaling pathway, half of the state variables correspond to cell one (c1) while the other half correspond to cell two (c2) indicated by subscript.

Observability	**State Variable**	**Description**
✓	pR2	Production rate of R2 non-long terminal repeat retrotransposons
✓	R2	Steady-state concentration of R2
✓	Notch1	Concentration of the Neurogenic locus notch homolog protein 1 (NOTCH1) receptor
✓	Hes	Expression level of the *Hes* gene
✓	NICD	Total concentration of NICD
✗	Dll4	Concentration of the *DLL4* (Delta-like 4) ligand [12]
✗	$\text{Dll4}_{\text{Notch1}}$	Complex formed by *DLL4* ligand and NOTCH1 receptor
✗	Jagged1	Concentration of Jagged1 [17]
✗	$\text{Jag}_{\text{Notch}}$	Complex formed by Jagged ligand and Notch receptor
✗	$\text{Notch}_{\text{jagc2}}$	Activation level of Notch receptor on cell 1 by Jagged ligand from cell 2
✗	V	Volume of cell

TABLE 6. **Observables and non-observable state variables of the Notch model.** There are 11 distinct species, and each occur in both of the cells of the signaling pathway. The species are differentiated by subscript in subsequent equations.

Name	Value	Units	Definition
$\mathrm{k}_{\mathrm{f}_{\mathrm{dllN}}}$	0.0001 [29]	$\mathrm{mmol}^{-1}\mathrm{s}^{-1}$	Forward binding rate constant for the Delta-like (Dll) ligand binding onto the Notch receptor (N)
$\mathrm{k}_{\mathrm{p}_{\mathrm{R2}}}$	0.3874 [38]	$\mathrm{mmol}^{-1}\mathrm{s}^{-1}$	Production rate constant for the R2 DNA element
$\mathrm{k}_{\mathrm{dp}_{\mathrm{R2}}}$	0.001 [38]	s^{-1}	Degradation rate constant for R2
$\mathrm{k}_{\mathrm{r}_{\mathrm{dllN}}}$	0.019 [29]	s^{-1}	Reverse binding rate constant describing the dissociation of the Delta-Notch complex
K_{m}	0.045 [25]	mmol	Michaelis constant, representing the substrate concentration at which the reaction is half its maximum rate
$\mathrm{k}_{\mathrm{cat}}$	0.2386 [25]	s^{-1}	Catalytic rate constant, representing the maximal number of molecules of substrate converted to product per active site per unit time
$\mathrm{kdeg}_{\mathrm{NICD}}$	0.002677 [4]	s^{-1}	Degradation rate of the Notch Intracellular Domain (NICD)
$\mathrm{kdeg}_{\mathrm{Notch}}$	0.001995 [4]	s^{-1}	Degradation rate of Notch receptors on the cell membrane
$\mathrm{kdeg}_{\mathrm{Dll4}}$	0.0001	s^{-1}	Degradation rate of the Delta-like 4 (Dll4) ligand
$\mathrm{k}_{\mathrm{p}_{\mathrm{Dll}}}$	0.0161	mmol	Production rate of the Dll protein
teta	2.3607	mmols^{-1}	Threshold concentration for Notch signaling activation
$\mathrm{kdeg}_{\mathrm{Hes1}}$	0.8	s^{-1}	Degradation rate of hairy and enhancer of split-1 (HES1), a transcriptional target of NICD
$\mathrm{k}_{\mathrm{p}_{\mathrm{Hes}}}$	0.04996	mmol	Production rate of Hes1
tetaHe	100.0019	mmols^{-1}	Threshold for HES1 activation, indicating the concentration required to initiate *Hes* transcription
$\mathrm{k}_{\mathrm{on}_{\mathrm{cis}}}$	$8.4 \cdot 10^{-4}$ [4]	$\mathrm{mmol}^{-1}\mathrm{s}^{-1}$	Binding rate constant for cis-interactions between ligands and receptors on the same cell
$\mathrm{kdeg}_{\mathrm{Jag}}$	$1.82 \cdot 10^{-5}$ [4]	s^{-1}	Degradation rate of the Jagged ligand on the cell membrane
$\mathrm{k}_{\mathrm{f}_{\mathrm{jagNotch}}}$	0.019 [29]	s^{-1}	Forward binding rate constant for Jagged binding to Notch receptors in trans-interactions
$\mathrm{kr}_{\mathrm{cis}}$	0.033238524 [4]	s^{-1}	Reverse binding rate constant for cis-interactions
$\mathrm{k}_{\mathrm{p}_{\mathrm{Jag}}}$	0.1	mmol	Production rate of the Jagged ligand
tetaJag	0.4999	mmols^{-1}	Threshold for Jagged-Notch signaling, signifying the concentration required for signal transduction
$\mathrm{kdeg}_{\mathrm{pR2}}$	0.0009 [38]	s^{-1}	Degradation rate of mRNA
$\mathrm{kdeg}_{\mathrm{iR2}}$	$5.06 \cdot 10^{-4}$ [32]	$\mathrm{mmol}^{-1}\mathrm{s}^{-1}$	Degradation rate of the immediate cytosolic protein iR2
Gs	0.01495	mmol	Concentration of gamma secretase, an integral membrane protein
$\mathrm{kform}_{\mathrm{Notch}}$	0.001	mmols^{-1}	Rate constant for Notch receptor formation on the cell membrane
kp	1	mmol	A general production rate

Table 7. **The initial estimates of the 26 parameters in the Notch model**, with units and definitions included for concreteness. Parameters without citations were fit by conventional numerical methods.

Appendix C. Notch Model Equations

Here we provide details on the ODE system governing the Notch signaling pathway. The rate of change of the species (state variables) are given as functions of other state variables as shown here:

$$
\begin{aligned}
\frac{d[\mathrm{Dl4}_{c_1}]}{dt} &= -\Big(k_f \cdot \mathrm{Dl4}_{c_1} \cdot \mathrm{Notch1}_{c_2} + k_r \cdot \mathrm{Dll1}_{\mathrm{Notch1}_{c_2}}\Big) \\
&\quad - \Big(k_f \cdot \mathrm{Dl4}_{c_1} \cdot \mathrm{Notch1}_{c_1} + k_r \cdot \mathrm{Dll4}_{\mathrm{Notch1}_{c_1}}\Big) \\
\frac{d[\mathrm{Notch1}_{c_1}]}{dt} &= -\Big(k_f \cdot \mathrm{Dl4}_{c_2} \cdot \mathrm{Notch1}_{c_1} + k_r \cdot \mathrm{Dll4}_{\mathrm{Notch1}_{c_1}}\Big) \\
&\quad - \Big(k_f \cdot \mathrm{Dl1}_{c_2} \cdot \mathrm{Notch1}_{c_1} + k_r \cdot \mathrm{Dll1}_{\mathrm{Notch1}_{c_1}}\Big) \\
\frac{d[\mathrm{Dl4}_{\mathrm{Notch1}_{c_1}}]}{dt} &= \frac{G_s \cdot k_{\mathrm{cat}} \cdot \mathrm{Dl4}_{c_1} \cdot \mathrm{Notch1}_{c_1}}{K_m + \mathrm{Dl4}_{\mathrm{Notch1}_{c_1}}} - \frac{\mathrm{tetAhe} \cdot \mathrm{Hes1}_{c_1 c_2}}{K_p + \mathrm{NICD}_{c_1 c_2}} \\
\frac{d[\mathrm{NICD}_{c_1}]}{dt} &= -\frac{\mathrm{tetAhe} \cdot \mathrm{Hes1}_{c_1 c_2}}{K_p + \mathrm{NICD}_{c_1 c_2}} \\
\frac{d[\mathrm{Jagged1}_{c_1}]}{dt} &= -k_{\mathrm{degJag}} \cdot \mathrm{Jagged1}_{c_1} - \big(k_{\mathrm{oncis}} \cdot \mathrm{Jagged1}_{c_1} \cdot \mathrm{Notch1}_{c_1} + k_{\mathrm{oncis}} \cdot \mathrm{JagNotch}_{c_1}\big) \\
\frac{d[\mathrm{Jag}_{\mathrm{Notch}_{c_1}}]}{dt} &= k_{\mathrm{oncis}} \cdot \mathrm{Jagged1}_{c_1} \cdot \mathrm{Notch1}_{c_1} + k_{\mathrm{oncis}} \cdot \mathrm{JagNotch}_{c_1} \\
\frac{d[\mathrm{Notch}_{\mathrm{jag2}_{c_1}}]}{dt} &= k_f \cdot \mathrm{JagNotch}_{\mathrm{jagc1}_{c_1}} \cdot \mathrm{Notch1}_{c_2} + k_r \cdot \mathrm{JagNotch}_{\mathrm{jagc1}_{c_1}} \\
&\quad - \Big(k_f \cdot \mathrm{Dl1}_{c_2} \cdot \mathrm{Notch1}_{c_1} + k_r \cdot \mathrm{Dll1}_{\mathrm{Notch1}_{c_1}}\Big) \\
\frac{d[\mathrm{pR2}_{c_1}]}{dt} &= k_p \cdot R_2^2 \cdot V - k_{\mathrm{dp}} \cdot R_2 \cdot \mathrm{pR2}_{c_1} - k_{\mathrm{degpR2}} \cdot \mathrm{pR2}_{c_2} \\
\frac{d[V_{c_1}]}{dt} &= k_p \cdot R_2^2 \cdot V - k_{\mathrm{dp}} \cdot R_2 \cdot \mathrm{pR2}_{c_1} \\
\frac{d[\mathrm{Dl4}_{c_2}]}{dt} &= -\Big(k_f \cdot \mathrm{Dl4}_{c_2} \cdot \mathrm{Notch1}_{c_1} + k_r \cdot \mathrm{Dll4}_{\mathrm{Notch1}_{c_1}}\Big) \\
&\quad - \Big(k_f \cdot \mathrm{Dl4}_{c_1} \cdot \mathrm{Notch1}_{c_2} + k_r \cdot \mathrm{Dll4}_{\mathrm{Notch1}_{c_2}}\Big) \\
\frac{d[\mathrm{Notch1}_{c_2}]}{dt} &= -\Big(k_f \cdot \mathrm{Dl4}_{c_2} \cdot \mathrm{Notch1}_{c_2} + k_r \cdot \mathrm{Dll4}_{\mathrm{Notch1}_{c_2}}\Big) \\
&\quad - \Big(k_f \cdot \mathrm{Dl1}_{c_2} \cdot \mathrm{Notch1}_{c_1} + k_r \cdot \mathrm{Dll1}_{\mathrm{Notch1}_{c_1}}\Big) \\
\frac{d[\mathrm{Dl4}_{\mathrm{Notch1}_{c_2}}]}{dt} &= \frac{G_s \cdot k_{\mathrm{cat}} \cdot \mathrm{Dl4}_{c_2} \cdot \mathrm{Notch1}_{c_2}}{K_m + \mathrm{Dl4}_{\mathrm{Notch1}_{c_2}}} - \frac{\mathrm{tetAhe} \cdot \mathrm{NICD}_{c_2 c_2}}{K_p + \mathrm{NICD}_{c_2 c_2}} \\
\frac{d[\mathrm{NICD}_{c_2}]}{dt} &= \frac{\mathrm{tetAhe} \cdot \mathrm{NICD}_{c_2 c_2}}{K_p + \mathrm{NICD}_{c_2 c_2}} - k_{\mathrm{cat}} \cdot \mathrm{Dl4}_{c_2} \cdot \mathrm{Notch1}_{c_2} \\
\frac{d[\mathrm{Jagged1}_{c_2}]}{dt} &= -k_{\mathrm{degJag}} \cdot \mathrm{Jagged1}_{c_2} - \big(k_{\mathrm{oncis}} \cdot \mathrm{Jagged1}_{c_2} \cdot \mathrm{Notch1}_{c_2} + k_{\mathrm{oncis}} \cdot \mathrm{JagNotch}_{c_2}\big) \\
\frac{d[\mathrm{Jag}_{\mathrm{Notch}_{\mathrm{jagc1}_{c_2}}}]}{dt} &= k_f \cdot \mathrm{JagNotch}_{\mathrm{jagc1}_{c_2}} \cdot \mathrm{Notch1}_{c_2} + k_r \cdot \mathrm{JagNotch}_{\mathrm{jagc1}_{c_2}} \\
&\quad - \Big(k_f \cdot \mathrm{Dl1}_{c_1} \cdot \mathrm{Notch1}_{c_2} + k_r \cdot \mathrm{Dll1}_{\mathrm{Notch1}_{c_2}}\Big) \\
\frac{d[\mathrm{pR2}_{c_2}]}{dt} &= k_p \cdot R_2^2 \cdot V - k_{\mathrm{dp}} \cdot R_2 \cdot \mathrm{pR2}_{c_2} - k_{\mathrm{degpR2}} \cdot \mathrm{pR2}_{c_1} \\
\frac{d[V_{c_2}]}{dt} &= k_p \cdot R_2^2 \cdot V - k_{\mathrm{dp}} \cdot R_2 \cdot \mathrm{pR2}_{c_2}
\end{aligned}
$$