INTERPRETING RATIONAL DYNAMICAL SYSTEMS USING KOLMOGOROV-ARNOLD NEURAL ODES (KAN-ODES)

Dennis Manjaly Joshy
Department of Mechanical Engineering
University of California, Santa Barbara
Santa Barbara CA, 93106
dennis00@ucsb.edu

December 6, 2024

ABSTRACT

Effective models built for industrial science-driven objectives demonstrate a balance of generalizability, interpretability, and parsimony. This is particularly relevant in fields that are heavily science-backed. Manufacturing procedures in pharmaceutical industries are rigorously evidence-based. With the arrival of the novel Kolmogorov-Arnold Network architecture, the tuneability of the activation functions vastly improves the expressivity, interpretability and efficiency of neural networks designed with them. Inspired by recent work where researchers used KANs in conjunction with neural ODEs to determine system dynamics, we extended the analysis to systems described as rational functions, as is commonplace in various biochemical systems in both academia and industry. Specifically, we investigate the performance of KAN-ODEs with a simple Monod type CSTR model and compare it to that of neural ODEs designed with the Multi-layer Perceptron (MLP) architecture. We further derive governing equations from the learned activation functions and demonstrate the applicability of this framework for system discovery. We discuss their performance in the context of biological systems and propose ideas and suggestions to physically inform the system to improve performance, interpretability, and scalability.

1 Introduction

System identification is a fundamental process for understanding an unknown system and predicting the system's behavior [1, 2]. The origins of system identification began with efforts to model systems by applying first principles and incorporating all known knowledge of the system into the governing equations. With the development of statistical modeling techniques such as least squares regression and Fourier analysis, the application of empirical methods to perform system identification gathered support in the early 20th century. It was later formalized in the mid-20th century by the advent of control theory [3], the Kalman filter for state estimation [4], and linear system identification techniques.

The steady growth in computing power and the results offered by these approaches in both academia and industry further led to the development of nonlinear system identification techniques [5] and subspace identification algorithms [6]. These leveraged matrix decomposition techniques like SVD to handle large and high dimensional datasets for MIMO (Multi-Input Multi-Output) systems. As the feature space of the systems being addressed further increased with improvement in digital measurement technologies, the field of system identification began to overlap with machine learning and bayesian techniques to quantify uncertainty.

A recent machine learning algorithm implemented for system identification of continuous time ODE systems is the SINDy method (Sparse Identification of Nonlinear Dynamics) [7]. Assuming the governing equations consist of a sparse set of features, it allows the user to specify physical priors as candidate library functions composed of these features. SINDy fits these candidate functions via regularized linear regression. SINDy has evolved to address various challenges faced by machine learning algorithms via ensembling [8], improved coordinate discovery and curse of dimensionality [9], and incorporating control inputs [10]. While SINDy offers a great solution to model systems in a short time mechanistically, its speed and performance are not easily scalable to high-dimensional systems as the size of the candidate library combinatorially increases. Users also might need help understanding the system to select the correct candidate functions, which is an inherent difficulty in any mechanistic approach.

One approach of automating the discovery of these governing equations of ODEs is using Neural ODEs. An evolution of the older Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTMs) notably used in time series modeling, Neural ODEs [11] parameterize the rates in the system of ODEs via a deep neural network. Coupled with an integrator of the user's choice (Forward Euler, RK45), the neural ODE can perform n-step prediction given the initial conditions of the ODE. Furthermore, neural ODEs incorporate adjoint sensitivity analysis which allows addressing irregularly spaced time points and is computationally efficient. Due to the Universal Function Approximation property of the Multi-Layer Perceptron and activation architecture [12], neural ODEs can learn and generalize to highly complex dynamics. However, this predictive power comes at the cost of interpretability of the learned model (in the mechanistic sense), and significant efforts currently take place to address the need for explainability of these models [13, 14].

A recent development in the field that proposes an alternative to the Multi-Layer Perceptron architecture is the Kolmogorov-Arnold Neural network [15] or the KAN network architecture. KANs generalize the application of the Kolmogorov-Arnold representation theorem to a deep neural network format and permit the tuning of the activation functions. The architecture further provides hyperparameters to increase the resolution of these activation functions via a method called Grid extension. While the authors initially used B-splines in their study, several studies have devised means of using other functions such as Radial Basis Functions (RBFs [16]) and rational functions (rKANs [17]) to enhance their expressivity further. The authors also describe a systematic approach to sparsifying and pruning the network to finally obtain an interpretable expression for the data being modeled.

As a potential alternative to MLPs, KANs have already been applied to many research questions [18, 19], including system identification using Neural ODEs. The interpretable architecture of the KAN network coupled with neural ODEs can provide interpretable expressions for the parameterized rates/derivatives, in addition to minimizing the number of parameters tuned. In [16], this approach was used to identify the governing equations of the Lotka-Volterra model and multiple PDEs. We extend this analysis to study the performance of this novel architecture on a simplified 2-state ODE model of a Continuously Stirred Tank Bioreactor (commonly referred to as CSTRs), which is characterized by rational functions due to the biochemistry involved in cell growth and substrate consumption. We compare the training and test performance between a KAN-ODE and a neural ODE of equal comparison and complete the procedure documented by [16] to discover interpretable equations that govern this system.

2 Kolmogorov-Arnold Neural Ordinary Differential Equations

For a given dynamical system represented by;

$$\frac{dX}{dt} = f(X,t) \tag{1}$$

A neural ODE parameterizes the rate equation using a neural network [11] using the weights and biases which are tuned in the process of learning the dynamics represented in (1).

$$\frac{dX}{dt} = NN(X,t,W,b) \tag{2}$$

Additionally, the neural ODE implements the highly efficient adjoint sensitivity method to minimize computation time, enabling it to step through varying time steps of discretization. Kolmogorov-Arnold Neural ODEs (KAN-ODEs) are implemented using Neural ODEs using Kolmogorov-Arnold networks in place of the multi-layer perceptron architecture.

$$\frac{dX}{dt} = KAN(X,t,\phi) \tag{3}$$

Where ϕ represents the tunable activation functions used in the model.

As per the Kolmogorov-Arnold Representation Theorem, the rates are in parameterized by the expression according to Liu et. al. [15], who generalized it to use any number of hidden layers and nodes as the user chooses,

$$KAN(X, t, \phi) = f(x_1, \cdots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{4}$$

Where ϕ represents a univariate tuneable activation function and Φ takes the sum of the activations in the previous layer and applies another univariate activation function. KAN-ODEs were first proposed by Koenig et al. [16] where they extensively studied the performance of such models for multiple systems like the Lotka-Volterra system, Fisher-KPP equations and the Schrodinger equation, where they reported orders of magnitude higher performance than NODEs using MLPs, with almost the same or fewer parameters. Next, sparsify the network using L1 regularization, prune the network nodes that do not receive, and calculate an output of strength below a threshold parameter. Ideally, this pruned network should perform as effectively as the unpruned network but have significantly fewer parameters.

Another novel attribute of KAN-ODEs is the flexibility to choose activation functions and the authors of [16] implemented their version of KAN-ODEs using a radial basis function. The activation functions are also calculated on a grid/coordinate axis whose size is tuneable. The grid size is reported to influence the performance of the KAN-ODE and is an essential hyperparameter in this modeling approach. Altogether, the layer width, the grid size, and the number of hidden layers determine the number of parameters the KAN uses. Finally, the authors also implement a new version of the aforementioned adjoint sensitivity method for KAN-ODEs.

3 Methods

A highly idealized and simplified system of equations representing the evolution of cell growth and substrate consumption can be derived using fundamental monod kinetics. We consider one such model for our analysis:

$$\frac{dS}{dt} = 0.8(2 - S) - \frac{S}{0.2 + S} X \tag{5}$$

$$\frac{dX}{dt} = \frac{S}{0.2 + S} X - 0.8X \tag{6}$$

S represents the substrate concentration metabolized by the cell, and X is the biomass concentration inside the bioreactor. In this study, we use one single trajectory of the solution to this system at an arbitrarily chosen initial condition $(0.1, 0.5)$, simulated for 50 days at a resolution of 0.1 days. We used the first 20 days as the training set for our models and the remaining 30 days as the test set (Figure 1).

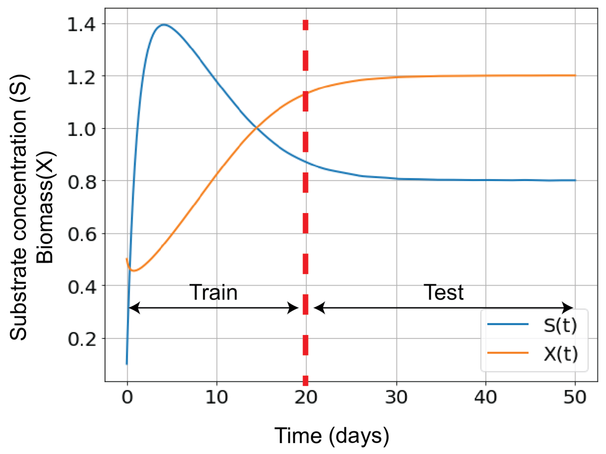

Figure 1: Simulated truth trajectories for a Monod-Type CSTR

Using the code base programmed in Julia in [16], we (code) initialize one KAN-ODE of 240 parameters (2 layers of layer width 10 and 5 grid points) and one neural ODE of 242 parameters for a fair comparison. We use the training set of points to calculate the training loss using the mean absolute error. We simulate from $t = 0$ till day 50 and calculate the mean absolute error to determine the total loss. The networks simulate the trajectory using the Tsit5() ODE solver function from DifferentialEquations.jl.

We enforced sparsity in the network using L1 regularization with a value of $1e-6$. We trained the CSTR KAN-ODE with regularization and the CSTR neural ODE for 100,000 iterations. We then used the sparse KAN-ODE as a starting point for the pruning experiment and pruned the nodes with a threshold value of $\theta = 0.25$. We used the same scheme in [16] to remove inconsequential nodes in the sparse KAN-ODE. This involves removing the nodes for which the maximum input and output values are below θ. We next retrained the pruned network with the same regularization level for another 100,000 iterations.

We next investigated the activation functions learned by the KAN-ODE while training in the pruned network. We fit univariate functions to each activation function and used symbolic regression via a genetic algorithm run for 100 iterations to determine each activation function's most accurate functional form. We then used the Kolmogorov-Arnold representation form (Equation 4) to calculate expressions for both $\dot{X}$ and $\dot{S}$.

4 Results

4.1 KAN-ODEs predict CSTR dynamics with significantly lower training and test loss

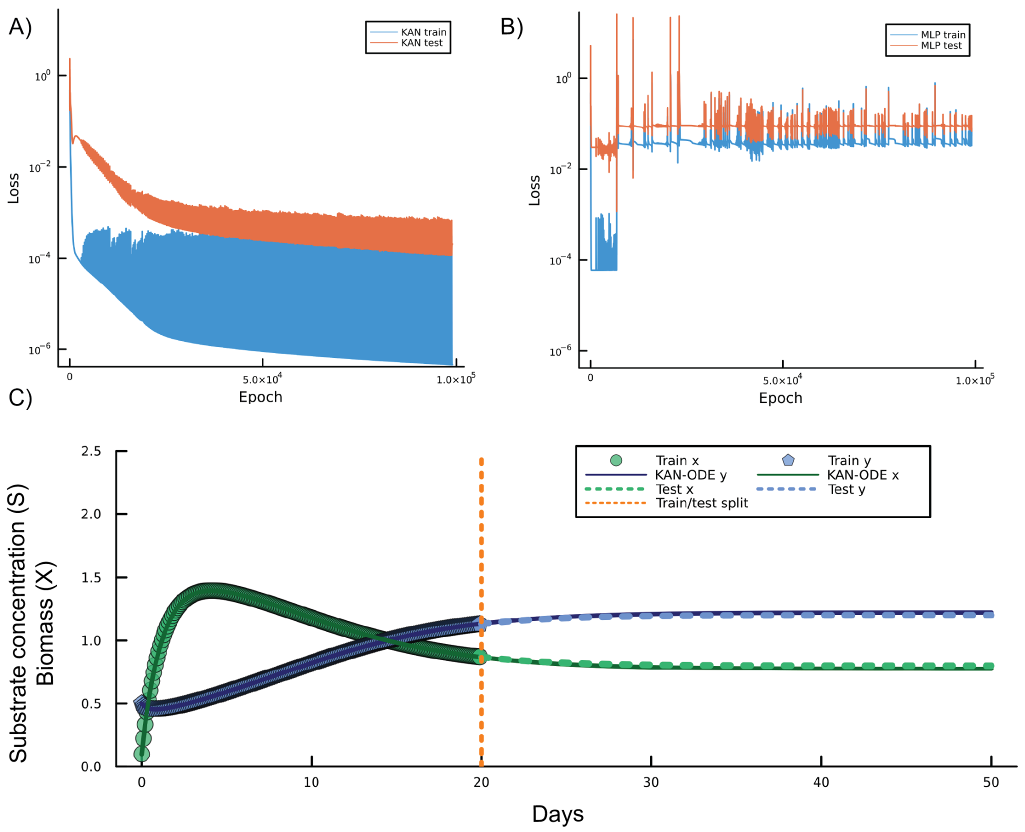

Figure 2: **KAN-ODEs outperform neural ODEs in performance metrics in predicting monod type CSTR dynamics**. 2A) The evolution of the train loss and test loss for KAN-ODEs as a function of epochs. 2B) The evolution of the train loss and test loss for neural ODEs as a function of epochs. 2C) Simulation of KAN-ODE network for given training data and test data.

We first compared the training loss and test loss evolution for an unsparsified KAN-ODE and a neural ODE concerning epoch number (Figure 2). From Figures 2A and 2B, we observe that the KAN-ODE handily outperforms the neural ODE in the training mean absolute error (blue). The minimum training loss for the KAN-ODE was recorded as close to $4.60\text{e}{-7}$, while the neural ODE initially performed well, with a minimum training loss of $5.93\text{e}{-5}$ until approximately 10,000 iterations, but increased to 0.01 and remained steady for the remainder of the 100,000 iterations. The trend is consistent with the test mean absolute error, where the KAN-ODE recorded a minimum loss of $1.1\mathrm{e}{-4}$, while the test loss of the neural ODE reported a higher value at $1.1\mathrm{e}{-3}$.

We also would like to report that the performance of the KAN-ODE was not significantly different from the neural ODE, when the initial training data provided was 25% (corresponds to 12.5 days), highlighting the relevance of capturing important dynamical features and trends in the system rates. Figure 2C indicates that the KAN-ODE simulates the data well and also succesfully predicts the fixed point behavior of the monody type CSTR system, even though the training region did not inform the model of such behavior.

4.2 Pruned sparse KAN-ODEs maintain consistent performance and learns relevant activation functions

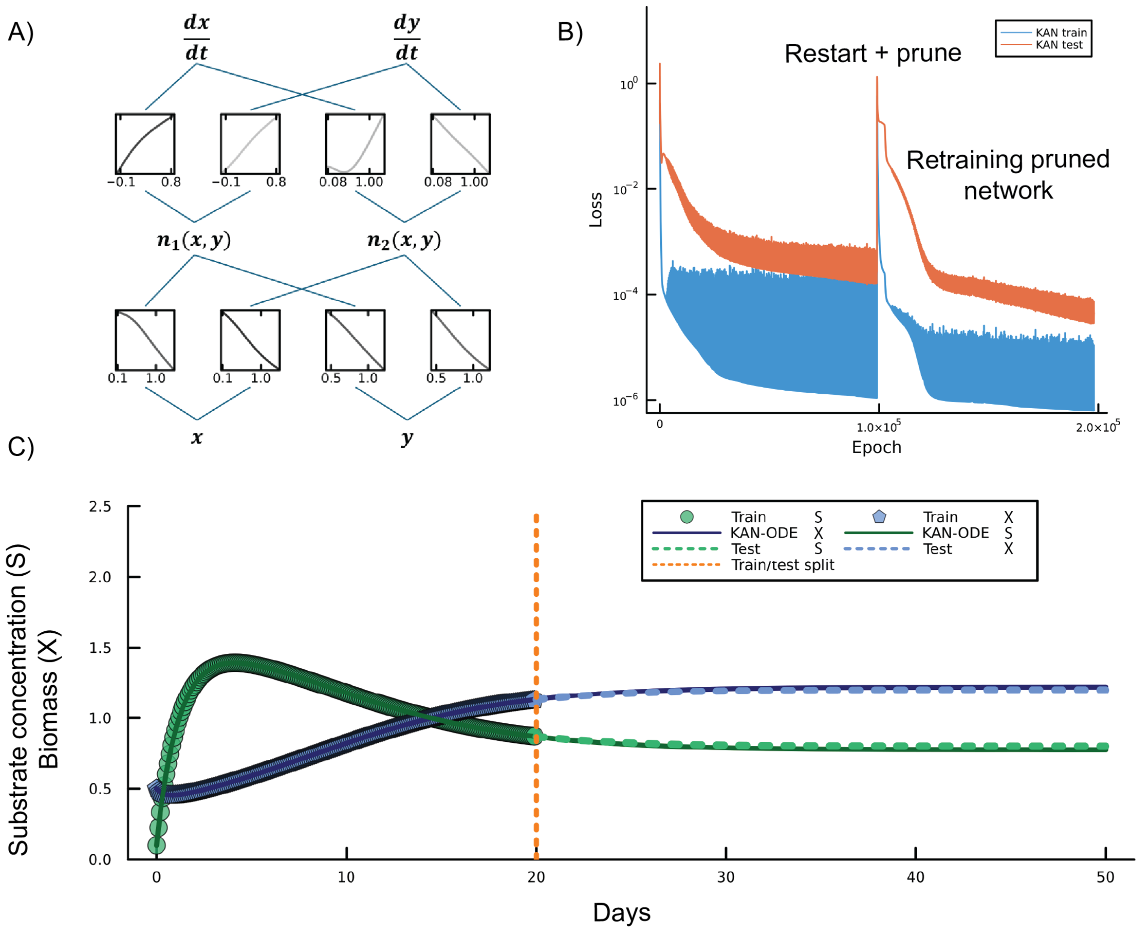

Figure 3: **Pruned KAN-ODE architecture uses 80% fewer parameters, but continues to maintain performance. 3A) KAN-ODE network architecture [2, 2, 5], as per the author's original naming conventions [input, hidden layer width, grid size] with the learned activation functions at each node**. 3B) Training and test loss of the sparsified pruned network appear to depict similar trends as the unsparsified network with a slightly elevated minimum loss (Figure 2A). 3C) Monod-type CSTR prediction of the sparsified pruned KAN-ODE captures system dynamics correctly.

Using the sparsified network, we pruned away the inconsequential nodes to reduce the size of the KAN-ODE from 242 parameters to 48 parameters. This corresponds to a KAN network of two hidden layers, each with two nodes accepting both, accepting the state inputs Figure 3A. As depicted by [16], the strength or influence of each activation function is marked by the darkness of the curves in each box (Figure 3A). The activation functions within these nodes are learned via Symbolic regression (section 4.2) and can be used to derive interpretable governing equations using the Kolmogorov-Arnold Representation theorem (Equation (4)). The pruning metric θ is another hyperparameter in the KAN-ODE framework, and we carefully tuned it (0.01, 0.1, 0.175, and 0.25) to discover the sparsest and the most minimal KAN-ODE without significantly altering the test loss in this example.

The minimum training loss for the KAN-ODE corresponding to $\theta = 0.25$ and L1 regularization of $1.0\text{e}{-6}$ with two hidden layers containing two nodes each, with a grid size of 5 (described as [2, 2, 5]), was recorded as close to $6.38\text{e}{-7}$, while the minimum test loss was $2.83\text{e}{-5}$. The value is slightly higher than the results from 4.1 due to the L1 regularization. We further emphasize that the training and test losses continued to decline steadily even after several hundreds of thousands of iterations, and the predicted trajectories converged to the ground truth trajectories, suggesting an increased expressivity for the KAN-ODE.

4.3 Discovering interpretable systems of equations for Monod CSTR systems

The activation functions were fit to univariate functions using Julia's SymbolicRegression.jl package, which discovered expressions for each activation function at each node in Figure 3A. See supplementary Table S1 for the complete results for each node. Applying the Kolmogorov-Arnold Representation theorem for our network, the expressions for the rates in our example can be calculated as follows;

$$\begin{aligned} n_1(x,y) &= \phi_{1x}(x) + \phi_{1y}(y) \\ n_2(x,y) &= \phi_{2x}(x) + \phi_{2y}(y) \end{aligned}$$

The components or contributing terms in each derivative are then given by;

$$\begin{aligned} \frac{dx}{dt} &= \Phi_{11}(n_1(x,y)) + \Phi_{12}(n_2(x,y)) = F(x,y) \\ \frac{dy}{dt} &= \Phi_{21}(n_1(x,y)) + \Phi_{22}(n_2(x,y)) = G(x,y) \end{aligned}$$

As mentioned earlier, the values of both ϕ and Φ in each layer are determined by symbolic regression. Figure 4A) shows a list of candidate expressions for the expression ϕ_{1x} (where x1 is a placeholder variable name). It's suggested from the original study that the choice of the expression from Table 4A should balance both parsimony and performance in the loss metric to get the best interpretable results. Similarly, a prudent choice must be made for the remainder of the activation functions. For example;

$$\begin{aligned} \phi_{1x}(x) &= -x + \frac{x}{0.16 + x} \\ \phi_{1y}(y) &= -y + 0.88 \end{aligned}$$

$$\begin{aligned} \phi_{2x}(x) &= -0.77(x-1) \\ \phi_{2y}(y) &= 0.74 - 0.63y \end{aligned}$$

Consequently,

$$\begin{aligned} n_1(x,y) &= -(x+y) + \frac{x}{0.16 + x} + 0.88 \\ n_2(x,y) &= -0.77(x-1) + 0.74 - 0.63y \end{aligned}$$

Simplifying, we get;

$$\begin{aligned} n_1(x,y) &= -(x+y) + \frac{x}{0.16 + x} + 0.88 \\ n_2(x,y) &\approx -0.7(x+y) + 1.5 \end{aligned}$$

In the second layer,

$$\Phi_{11}(n_1) = 1.46n_1 - n_1^2$$
$$\Phi_{12}(n_2) = 0.23n_2^2 + 0.1$$

$$\Phi_{21}(n_1) = 0.17\frac{n_1^2 + 0.11}{0.55n_1^2 + 0.47n_1 + 0.77}$$
$$\Phi_{22}(n_2) = -0.28n_2$$

These expressions represent only one of the possible pathways to take and result in a complex expression for $F(x, y)$ and $G(x, y)$, which can still be simulated using conventional solvers such as Python's $solve\_ivp$ (code). The results of these expressions are shown in Figure 4B, and we quickly discover that they do not perform as well as expected.

We then decided to avoid the uncertainty of approximating and selecting candidate functions. We used the expressions with the highest complexity and the least loss (an example corresponds to the last row in the table in Figure 4). Consequently, although the resulting expressions are complex and cumbersome compared to the initial expression, they still represent an interpretable model of the Monod-type CSTR dynamics. Figure 4C demonstrates that the simulated curves from the resulting ODE do a far better job of approximating the system dynamics.

A)

Complexity	Loss	Rounded Equation
1	0.029612	-0.078
3	0.001088	0.8504 - x1.0
5	0.000171	(x1.0 * -0.85) + 0.7112
7	0.000171	((x1.0 * -0.85) + -0.2415) +0.9527
9	7.76E-05	(x1.0 * -0.9949) + (x1.0 / (x1.0 +0.1624))
11	2.76E-05	x1.0 * ((0.84 - x1.0) / ((x1.0 + 0.0722) / 0.9168))
13	2.17E-05	((x1.0 / (x1.0 +0.5057)) / (x1.0 +0.3011)) + (x1.0 * -0.6522)
15	5.99E-06	((((x1.0 * (x1.0 * 0.3388)) + 0.088) * x1.0) +0.4312) - (x1.0 * x1.0)
17	2.70E-06	(((((x1.0 * x1.0) * 0.419) + 0.2488) * x1.0) +0.3945) - ((x1.0 * 1.2066) * x1.0)
21	2.70E-06	(((((((x1.0 * x1.0) * 0.419) + 0.2488) * x1.0) +0.3945) - (x1.0 * (x1.0 * 1.2066))) + x1.0) - x1.0
23	2.20E-06	(x1.0 * (((((x1.0 * 0.3537) + 0.2318) * (x1.0 * x1.0)) +0.4892) - (x1.0 * (x1.0 * 1.2219)))) / (x1.0 +0.0213)

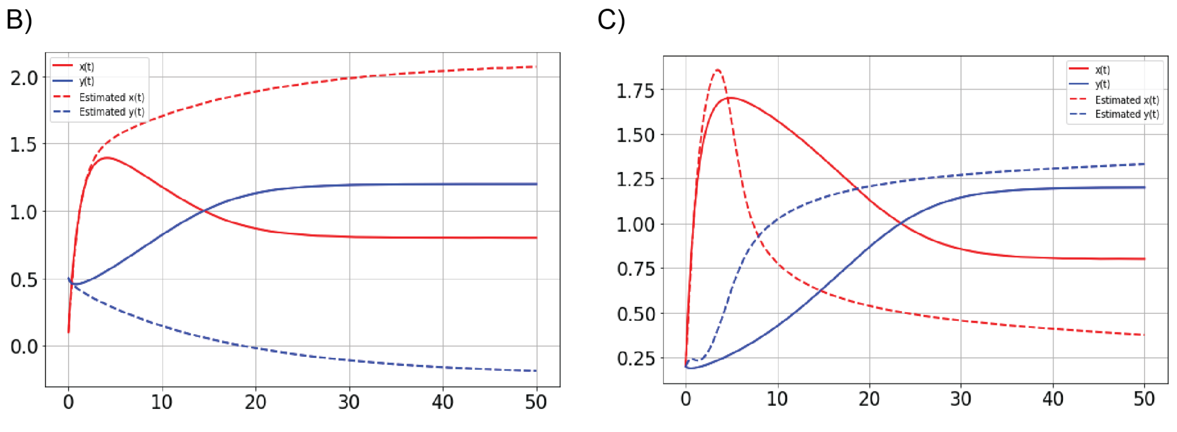

Figure 4: **Candidate symbolic expressions for activation function $\phi_{1x}(x)$ and simulated symbolic expressions for two interpretable models provided by the KAN-ODE**. A) Results for the univariate symbolic regression performed for the activation function $\phi_{2x}(x)$. The complexity metric specifies the parsimony of the expression, and the loss refers to the prediction error resulting from the regression process. B) Simulation results from a manual selection of candidate activation functions. C) Simulation results by selecting the most complex candidate function for each activation function.

5 Discussion

Our analysis shows that KAN-ODEs perform remarkably better than neural ODEs for this simple Monod-CSTR system composed of rational functions using this training methodology. We implemented a systematic approach to interpret the functions learned by the KAN-ODE to discover a system of equations from very minimal that performed well to predict the general trends of the Monod-CSTR dynamics. While we did not recover the exact equations (5, 6), the symbolic regression discovered an important functional form ubiquitous in biochemical kinetics. The rational functional form is given by

$$\phi_{1x}(x) = -x + \frac{x}{0.16 + x}$$

Indicates that the activation functions could approximate and encode an activation trend by the substrate. The function almost correctly estimated the dissociation constant (K_m) that governs the substrate's rate of association and dissociation within the cell's metabolic machinery. Knowledge of these constants gives significant insight into each state's contribution in the system. There are genetic circuits that involve repression dynamics, which are governed by functions such as;

$$\frac{1}{K_R + x}$$

It would be interesting to investigate the performance of this framework with systems such as the oscillating repressilator [20] or the activator repressor clock [21], since the approach showed great success with the oscillating Lotka-Volterra system.

While the KAN network could predict the system's long-term behavior, it did not perform as well when the training set was reduced to 25% of the complete dataset. We began our analysis with this threshold since it was the amount used by the original study's authors. However, after trial and error, we determined that at least 40% of the data was required for this system. We hypothesize that this is because the first derivative of the system initiates its long-term behavior (especially the biomass concentration) after 12.5 days in this system. This is a potentially important hyperparameter to tune to promote generalizability.

We, however, find it rather contradictory that even though we trained a generalizable model with testing losses of the order of -7, the activation functions did not easily reflect the system dynamics. We'd like to suggest that a pertinent choice of activation functions will facilitate the discovery and interpretability of these equations since it is already well-understood that specific activation functions improve the performance of MLPs on different tasks. This furthermore provides us a vehicle to build in physical priors to the KAN-ODE model, much like specifying candidate function libraries with SINDy. For instance, one recent extension of KANs is the use of rational functions [17] called rational KANs, which implemented Jacobi polynomials in one layer and a layer of Pade type functions in a subsequent layer to improve the KAN's expressivity as rational functions capture far more complex patterns. We debated using these activation functions in this study. We decided to investigate this in detail in a later manuscript. We believe this approach might hold great promise when systems are composed of multiple functional forms, including rational functions and other polynomial or exponential forms.

The grid size and pruning threshold are two other important hyperparameters in this framework. While we used the default grid size, decreasing the grid size was reported to improve accuracy and learn more relevant activation functions. The finer grid sizes also increase the computational cost. The prune threshold is also to be carefully tuned, and we observed that it is tied to the trends of regularization in this study. This is because the strength of regularization influences the activity of the nodes, and a careful balance of sparsity and performance must be maintained to reduce the network size correctly without compromising generalizability.

As a pilot study, we also investigated the performance of this approach in various other systems, including the chaotic Rossler system of equations. We could extend the method to three dimensions but could not achieve high test performance, although they reported excellent training loss values. It would be interesting to explore such chaotic systems and whether these governing equations can be discovered with this methodology.

Finally, we'd like to comment on the procedure for deriving the governing equations. Firstly, it is a highly manual process, and we propose the definition of a rigorous metric that determines the suitability of the discovered functions for further processing. We foresee that an accurate interpretation of the KAN-ODE dynamics would require physics-informed symbolic regression. For example, one way of doing this is to enforce functional forms, such as rational forms and polynomials, as members of the populations of the genetic algorithm. Alternatively, we can regularize the procedure to promote relevant functional forms. We also anticipate a scalability issue since selecting the candidate functions for interpretability is currently a manual task. Automating this process will be essential for extending the framework to a higher dimensional system, with special consideration for sparsification and pruning.

6 Conclusion

Multi-layer perceptrons have dominated the field of modeling systems using neural networks for several decades. While their predictive power is remarkable, they are inherently not interpretable due to their structure. Kolmogorov-Arnold networks offer a potential alternative to MLPs and can be applied across multiple domains. This is especially relevant in high-dimensional systems with complex interactions between the states but where the system dynamics are not fully explored. The added interpretability, minimalism, and flexibility of user-specified and tuneable activation functions provided by this framework will inform system identification initiatives across various industries, including communications, biopharmaceuticals, robotics, and academic initiatives implementing scientific machine learning approaches.

7 Acknowledgements

I acknowledge the technical discussions with my roommates Rohan Ravula, VB Janitri Babu, and my friends Shashank Acharya and Sai Akhil Rayapudi for insightful and intriguing deep learning and biology conversations.

References

[1] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[2] Lennart Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1–12, 2010.

[3] Karl J Åström and Björn Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[4] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. Kalman filter and its application. In *2015 8th international conference on intelligent networks and intelligent systems (ICINIS)*, pages 74–77. IEEE, 2015.

[5] Georgios B Giannakis and Erchin Serpedin. A bibliography on nonlinear system identification. *Signal Processing*, 81(3):533–580, 2001.

[6] S Joe Qin. An overview of subspace identification. *Computers & chemical engineering*, 30(10-12):1502–1513, 2006.

[7] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[8] Urban Fasel, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260):20210904, 2022.

[9] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[10] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[11] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[12] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[13] Colby Fronk and Linda Petzold. Interpretable polynomial neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(4), 2023.

[14] Colby Fronk, Jaewoong Yun, Prashant Singh, and Linda Petzold. Bayesian polynomial neural networks and polynomial neural ordinary differential equations. *PLOS Computational Biology*, 20(10):e1012414, 2024.

[15] Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. Kan 2.0: Kolmogorov-arnold networks meet science. *arXiv preprint arXiv:2408.10205*, 2024.

[16] Benjamin C Koenig, Suyong Kim, and Sili Deng. Kan-odes: Kolmogorov–arnold network ordinary differential equations for learning dynamical systems and hidden physics. *Computer Methods in Applied Mechanics and Engineering*, 432:117397, 2024.

[17] Alireza Afzal Aghaei. rkan: Rational kolmogorov-arnold networks. *arXiv preprint arXiv:2406.14495*, 2024.

[18] 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.

[19] Chenxin Li, Xinyu Liu, Wuyang Li, Cheng Wang, Hengyu Liu, Yifan Liu, Zhen Chen, and Yixuan Yuan. U-kan makes strong backbone for medical image segmentation and generation. *arXiv preprint arXiv:2406.02918*, 2024.

[20] Michael B Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[21] Domitilla Del Vecchio. Design and analysis of an activator-repressor clock in e. coli. In *2007 american control conference*, pages 1589–1594. IEEE, 2007.