Neural Ordinary Differential Equations for Predicting Metabolic Pathway Dynamics from Time-Series Multiomics Data

Udesh Habaraduwa

I. Introduction

For most of history human life span was limited and one would be considered lucky to reach the age of 35. As our understanding of biological processes gradually improved (e.g., the germ theory of disease, antibiotics, antiseptics, etc.), so too the length and quality of life around the world [1]. Today, the expected lifespan of an adult is nearly twice that of 200 years ago [1]. The bottlenecks that stand in the way of yet another doubling may be qualitatively different in nature, necessitating a deeper understanding of the complex dynamical systems at the heart of biological processes [2].

Modern bio-engineering techniques (e.g., genomic engineering with CRISPR [3] and activation of limb regrowth in species without the innate capacity to do so [4]) have opened up promising avenues for altering the "natural" course of biology to achieve more desirable outcomes. However, there is progress yet to be made in predicting the behavior of biological systems as a whole when a change is introduced [5].

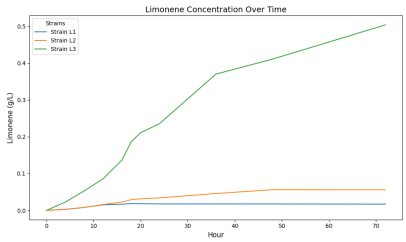
Most systems in biology are observed as short, noisy, time-series recordings (e.g., concentrations of a metabolites or proteins over time) that can be time consuming and expensive to measure [1]. However recent advances in high-throughput techniques (e.g., at the genome, protieome, transcriptome, and metabolome levels) [6] have resulted in a drastic increase in usable data. Thus, the application of modern machine learning (ML) pipelines (e.g., automated model selection and hyper-parameter tuning [7]) is becoming increasingly feasible [8].

In the present work, drawing inspiration from work done at the transcriptome level [9], neural ordinary differential equations (NODEs) are introduced as a promising approach for learning the complex interplay between the proteome (proteins that are created and modified by an organism [10]) and the metabolome (the collection of metabolites that are responsible for life [11], often controlled by a protein) from time-series data.

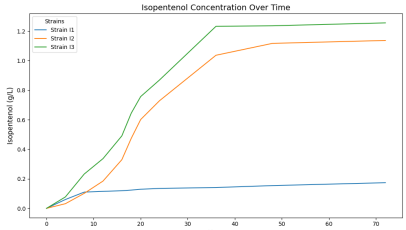
II. Materials and Methods

A. Data

The present study aims to investigate if NODEs can outperform traditional machine learning methods [5] in capturing the dynamics of metabolic pathways. To that end, a proteomic and metabolimic time-series dataset is utilized [5]. The dataset was derived from an experiment in metabolic engineering [12]. Here, different strains of the Escherichia coli bacterium were engineered to be low, medium, or high producers of isopentenol or limonene. The measured time-series for these two metabolites is shown in Figure 1.



(a)



(b)

Fig. 1: Time series measurements for metabolite concentrations. (a) Limonene concentrations across strains L1, L2, and L3. (b) Isopentenol concentrations across strains I1, I2, and I3.

While limonene and isopentenol are the metabolites of interest, the original dataset contains 91 metabolomic and 23 proteomic features, each recorded over 72 hours in 14 intervals. However, [5] only use a subset of these features (Table I). The original time-series was interpolated to generate a time-series of 200 intervals. Finally, the two datasets are combined as follows:

- Controls - These represent the concentrations of proteins (e.g., AtoB, GPPS, etc.) that regulate the production of metabolites at a given time point. The dataset contains one time-series of concentrations for each protein. Proteins play an important regulatory role within biological systems [13]. Depending on the protein and the context, they maybe involved in a range of functions from determining cellular structure to facilitating transport [14].
- States - These represent the concentration of a metabolite of interest (e.g., limonene or isopentenol). Metabolites are the result of metabolic processes and usually the end product of protein metabolism [13]. The dataset contains one time-series of concentrations for each metabolite.

The controls and states from the data are listed in Table I. Each metabolite and protein is represented in a time-series similar to examples shown in Figure 1, not shown here for brevity.

Controls	States
AtoB	Acetyl-CoA
GPPS	HMG-CoA
HMGR	Mevalonate
HMGS	Mev-P
Idi	IPP/DMAPP
Limonene Synthase	Limonene
MK	OD600
NudB	GPP
PMD	NAD
PMK	NADP
	Acetate
	Pyruvate
	citrate
	Isopentenol

TABLE I: List of Controls and States relevant to the limonene and isopentenol pathways, for a total of 23 features with 600 rows (200 for each strain). The limonene and isopentenol pathways were trained independently.

The dataset can be found by referring to the supplementary material of [5].

B. Methods

To begin, let us define the relationship between metabolites and proteins as a differential equation:

$$\dot{m} = f(p, m) \tag{1}$$

where m and p represent metabolite and protein concentrations respectively. This equation states that the change in m and p is a function of the current concentration of m and p. Such equations are the preferred mathematical representation of bio-chemical kinetics [5], [15].

The models trained by [5] serve as a baseline for testing the proposed implementation. In short, their approach is as follows:

1) For each state and control variable, approximate the derivatives at each time point. The noisy experimental data is first smoothed and the derivative is calculated using a central differencing scheme [5].
2) Find a function that best predicts the approximated derivatives. The authors employ an automatic machine learning approach (with TPOT [7]), searching over a space of machine learning models (e.g., Gradient boosting regressor, random forest regressor, etc.) to find the best one. The models are trained on all the states and controls to minimize the loss on predicting the derivatives of the states alone. A unique model is trained for each state derivative. That is, each model takes the place of the right hand side of equation 1 for one of the metabolites.
3) The trained models, which serves as a derivative function for each state variable, are then integrated using a ODE solver to generate new state values at a given time point.

Using a NODE, the step of approximating derivatives and subsequently fitting an ML model for each feature can be circumvented entirely. Instead, a fully connected neural network is trained to learn the derivative of each state and control variable directly. This is represented in equation 2, where f is a function of the neural network parameters θ and the state variables m and p (collectively defined as u). Once the network has been trained, it serves as the derivative function in the ODE solve (algorithm 1).

$$\dot{u} = f(u, \theta) \tag{2}$$

Algorithm 1 Optimization Process

1: **for each** optimization step **do**
2: Solve $\frac{du}{dt} = f(\theta, u)$ to get a trajectory for each feature by integrating the NODE.
3: Calculate
4: Update θ
5: **end for**

Each solve step of algorithm 1 results in a vector of shape 200×23. Consider one row of the dataset (1×23), representing one time point for each feature. The solve step returns a trajectory matrix of shape 200×23 : The data point evolved for 200 time steps. The last time step from each trajectory is taken to be the NODE prediction for this time point. That is, the neural network is tasked with learning the vector field that evolves a given time point through a trajectory that best approximates the dynamics represented in the training data [16].

For illustration, consider the toy space defined by two features (e.g., Acetyl-CoA and HMG-CoA) in Figure 2. The model is trained to learn the "shape" of this space such that for a given training point (e.g., [HMG-COA$[t_0]$, Acetyl-CoA$[t_0]$]), the point evolves over n time-steps and ends up in the "right place".

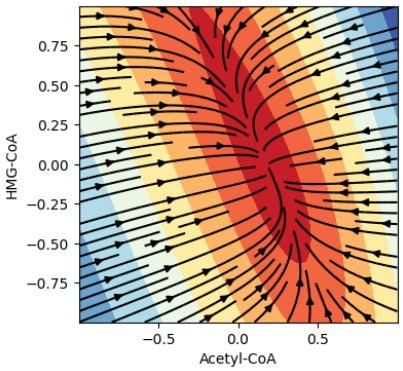


Fig. 2: Visualization of a (toy) learned vector field in two possible dimensions. Color intensity represents the magnitude of the vector field, with red indicating regions of faster dynamics and blue showing slower changes. Black streamlines indicate the direction of system evolution.

A promising approach to training NNs in physical applications is to incorporate domain knowledge into the model [15].

Here, a physiology-informed regularization term (PIR) ($\lambda = [0.01, 1.0, 1000]$) is added to the loss function which penalizes predictions of negative values for metabolite concentrations (equation 3). Optimization is carried out in two stages [15]. First, with ADAM [17] for 300 epochs (learning rate (LR) $= 1e-4$) and BFGS [18](LR $= 1e-4$, maximum iterations $= 1000$, gradient tolerance $= 1e-6$, change tolerance $= 1e-6$).

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \lambda \cdot \frac{1}{n} \sum_{i=1}^{n} (\min(0, \hat{y}_i))^2 \tag{3}$$

The architecture of the NN component of the NODE is shown in Table II. In terms of architectural decisions, considerations were made regarding the size (depth and width) and activation functions. Though it was assumed that general characteristics of NNs (e.g., depth improves performance [19]) would hold, a relatively small NN was chosen (useful in scientific machine learning applications [20] where the outputs of a NN can be directly linked to, for example, unobserved interactions of metabolites). The chosen activation function should be differentiable and smooth [21], [22], both of which the hyperbolic tangent function satisfy [22].

A prediction of a trajectory requires an ODE solve to approximate the current value of the function (i.e., the approximation of the current concentration of a metabolite given its derivative). A variety of methods exist for obtaining approximate solutions to time-dependent ordinary differential equations (e.g., the Runge-Kutta family of methods [23]). Here, the 8th order Dormand-Prince [23] method is used from the torchdiffeq library [24] with default error tolerance parameters.

TABLE II: Architecture of the NeuralODE Function. Weights are initialized with mean 0 and standard deviation 0.1.

Layer	Type	Details
1	Linear	Input: 23, Output: 10, Bias: True
2	Activation (Tanh)	-
3	Linear	Input: 10, Output: 10, Bias: True
4	Activation (Tanh)	-
5	Linear	Input: 10, Output: 10, Bias: True
6	Activation (Tanh)	-
7	Linear	Input: 10, Output: 10, Bias: True
8	Activation (Tanh)	-
9	Linear	Input: 10, Output: 23, Bias: True

Code is available online [25]. All experiments were conducted on a Google Colab instance with the following specifications:

- CPU: Intel(R) Xeon(R) @ 2.00GHz (8 cores)
- GPU: NVIDIA Tesla T4 (15.36GB VRAM)
- Operating System: Linux-6.1.85+-x86_64
- Python Version: 3.10.12
- PyTorch : 2.5.1+cu121
- torchdiffeq : 0.2.5
- Numpy : 1.26.4

III. Results

The performance compared to the baseline models of [5] are shown in Tables III and IV. Plots of the trajectories generated by the best performing model for Limonene strain dynamics is shown in Figure 3. All models were trained on low and high producing strain data and tested on the held out medium producing strain (as in [5]).

TABLE III: Performance of NODE over baseline on the Limonene test data for varying regularization strengths (λ). Values are root mean squared error (RMSE).

	0.01	**1.00**	**1000.00**	**Baseline**
Acetyl-CoA	0.20	0.09	1.07	0.34
HMG-CoA	0.50	0.20	0.53	0.02
Mevalonate	1.21	0.37	2.38	1.07
Mev-P	1.30	0.17	0.23	3.64
IPP/DMAPP	0.36	0.37	0.32	75.43
Limonene	0.57	0.32	0.15	0.30
OD600	0.91	0.61	2.25	4.30
GPP	1.02	0.54	0.61	0.07
NAD	0.63	0.35	0.16	1.50
NADP	0.07	0.50	1.22	1.26
Acetate	0.44	0.60	1.20	1.89
Pyruvate	0.33	0.15	0.44	0.14
Citrate	0.28	0.83	0.70	0.23
Mean RMSE	**0.60**	**0.39**	**0.87**	**6.94**
% Improvement	**91.35**	**94.38**	**87.46**	**–**

TABLE IV: Performance of NODE over baseline on the Isopentenol test data for varying regularization strengths (λ). Values are RMSE.

	0.01	**1.00**	**1000.00**	**Baseline**
Acetyl-CoA	0.56	0.74	0.16	9.02
HMG-CoA	0.70	0.04	0.34	0.08
Mevalonate	1.03	0.25	0.63	2.55
Mev-P	0.23	0.20	0.64	126.19
IPP/DMAPP	0.22	0.23	0.65	33.77
OD600	0.07	0.31	0.84	2.13
GPP	0.03	0.27	1.24	0.00
NAD	0.29	0.61	0.72	1.96
NADP	1.03	0.16	0.74	0.24
Acetate	0.25	0.69	0.10	0.45
Pyruvate	0.85	0.25	0.56	0.06
Citrate	0.13	0.37	0.33	0.29
Isopentenol	0.10	0.08	0.25	0.32
Mean RMSE	**0.42**	**0.32**	**0.55**	**13.62**
% Improvement	**96.92**	**97.65**	**95.96**	**–**

TABLE V: Training and inference times for baseline and NODE models (averaged over λ values). Times may vary slightly between training runs, particularly for baseline models due to TPOT's genetic recombination process. However, the times for both models remained consistent and comparable across runs and datasets (see [25]).

Model	Training Time (minutes)		Inference Time (minutes)	
	Limonene	Isopentenol	Limonene	Isopentenol
Baseline	52.08	61.79	4.83	4.04
NODE	5.61	5.40	0.004	0.004

IV. Discussion and concluding remarks

The present study evaluated the performance of NODEs against previous generations of ML techniques [5] in learning the dynamics represented in metabolomic and proteomic time-series data.

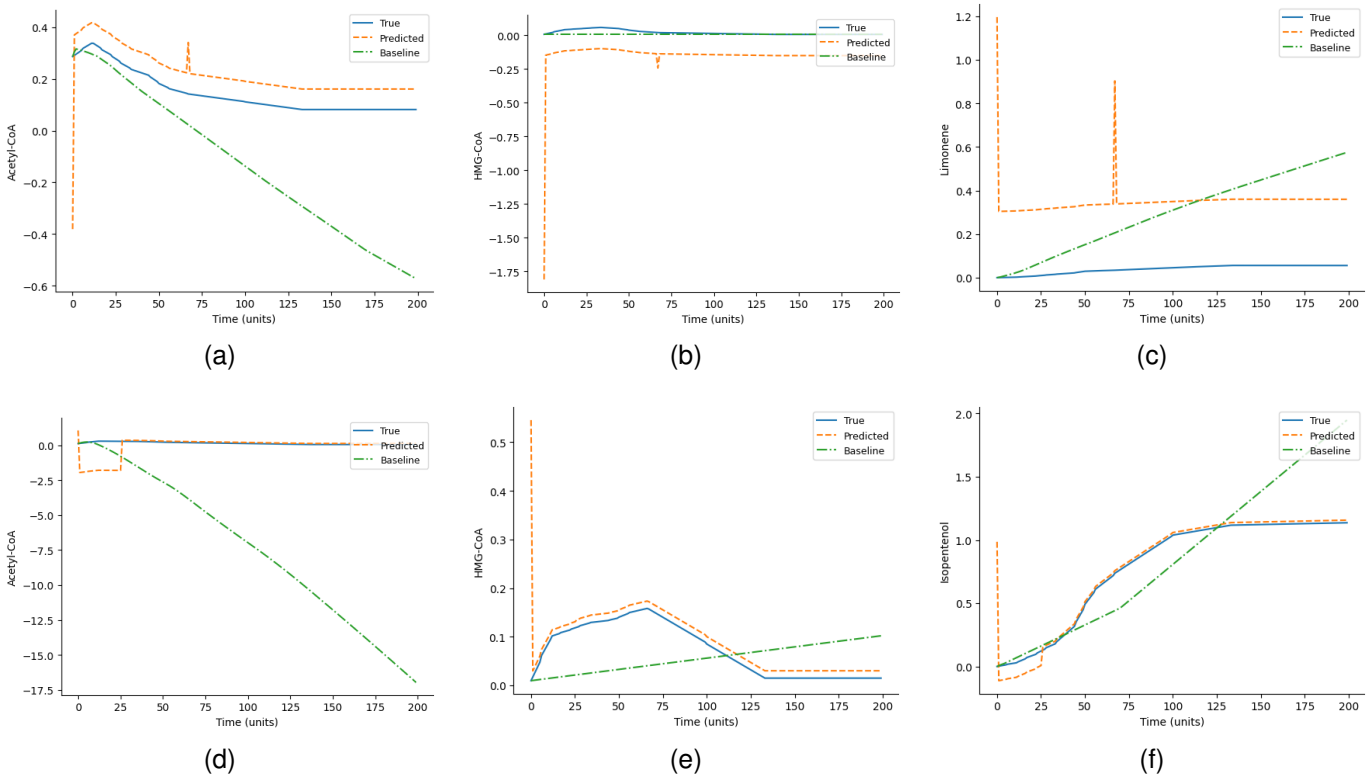


Fig. 3: Prediction of strain dynamics for Limonene and Isopentenol pathways on the held out medium producing strains (I2 and L2). Limonene pathway (top row): (a) Acetyl-CoA concentration, (b) HMG-CoA concentration, (c) Limonene concentration. Isopentenol pathway (bottom row): (d) Acetyl-CoA concentration, (e) HMG-CoA concentration, (f) Isopentenol concentration. All experimental data was recorded over 72 hours, every 2 hours. Data was interpolated to 200 time points[5]. Only a subset of all predicted metabolites shown. Note that the NODE is able to qualitatively capture the underlying pattern even if it is off by a scale factor.

Across both dynamics datasets (limonene and isopentenol), the NODEs out perform the baseline models with 87.5% - 95% improvement in mean RMSE. The models also seem to benefit from a moderate amount of PIR ($\lambda = 1.0$) in learning the dynamics in both datasets. Notably, even when the error between the NODE prediction and actual time series is large, the model appears to qualitatively capture the underlying dynamics, missing only the scale factor (visible in 3a), as in [5]. Indeed, for metabolic engineering purposes the ability to reliably identify the most productive strain is sufficient [5].

The NODE models were approximately 10x faster in training and 1000x faster at inference (Table V). The long training times taken for the baseline models can be attributed to the process of searching through a space of best pipelines with models that do not inherently support hardware acceleration [26]. Nonetheless, even if more modern implementation were to be used, fitting a unique function to serve as the derivative for each state variable may ultimately be unnecessary.

NODEs, much like other ML methods, rely heavily on the quantity and quality of the data, necessitating the interpolation techniques implemented by [5]. It's possible that longer and higher resolution time-series data would enable the use of deeper networks for capturing richer representations. There is also room for improvement in the NODE set up used here in. Stiff (neural) ODEs (i.e., where the time-steps necessary for good approximations of the integrated solutions grows extremely small), pose a problem for the computational efficiency which may be addressed by stabilizing gradient calculations, scaling of network outputs, and using alternative activation functions (or learning the best one for the task and data at hand [27], [28]). Performance may benefit further from NN specific regularization techniques (e.g., enforcing Lipschitz continuity [29]) and incorporating further knowledge of the task domain into the loss function. Finally, given the simplicity of the underlying NN, it may be possible to derive governing equations for the system [30].

The results herein illustrate the potential of NODEs for efficiently and accurately capturing the dynamics of interacting metabolic and proteomic systems. For the future of bioengineering, these techniques may prove to be invaluable for rapidly iterating through design space, control mechanisms in production (e.g., for moving from the lab bench to industrial scale [31]), and generating new insight into the nuanced mechanics of biochemical interaction processes. With better prediction and control of biology, it may be possible to unlock the next stages of longer and healthier lives.

References

[1] N. v. d. Berg, M. Rodríguez-Girondo, A. J. de Craen, J. J. Houwing-Duistermaat, M. Beekman, and P. E. Slagboom, "Longevity around the turn of the 20th century: Life-long sustained survival advantage for parents of today's nonagenarians," *The journals of gerontology: series A*, vol. 73, no. 10, pp. 1295–1302, 2018.

[2] T. Ideker, L. R. Winslow, and D. A. Lauffenburger, "Bioengineering and systems biology," *Annals of biomedical engineering*, vol. 34, pp. 1226–1233, 2006.

[3] F. Ran, P. D. Hsu, J. Wright, V. Agarwala, D. A. Scott, and F. Zhang, "Genome engineering using the crispr-cas9 system," *Nature protocols*, vol. 8, no. 11, pp. 2281–2308, 2013.

[4] N. J. Murugan, H. J. Vigran, K. A. Miller, *et al.*, "Acute multidrug delivery via a wearable bioreactor facilitates long-term limb regeneration and functional recovery in adult xenopus laevis," *Science advances*, vol. 8, no. 4, eabj2164, 2022.

[5] Z. Costello and H. G. Martin, "A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data," *NPJ systems biology and applications*, vol. 4, no. 1, pp. 1–14, 2018.

[6] H.-F. Juan and H.-C. Huang, "Quantitative analysis of high-throughput biological data," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 13, no. 4, e1658, 2023.

[7] R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating machine learning," in *Workshop on automatic machine learning*, PMLR, 2016, pp. 66–74.

[8] M. Mowbray, T. Savage, C. Wu, *et al.*, "Machine learning for biochemical engineering: A review," *Biochemical Engineering Journal*, vol. 172, p. 108 054, 2021.

[9] R. Erbe, G. Stein-O'Brien, and E. J. Fertig, "Transcriptomic forecasting with neural ordinary differential equations," *Patterns*, vol. 4, no. 8, 2023.

[10] E. J. Dupree, M. Jayathirtha, H. Yorkey, M. Mihasan, B. A. Petre, and C. C. Darie, "A critical review of bottom-up proteomics: The good, the bad, and the future of this field," *Proteomes*, vol. 8, no. 3, p. 14, 2020.

[11] B. Peng, H. Li, and X.-X. Peng, "Functional metabolomics: From biomarker discovery to metabolome reprogramming," *Protein & cell*, vol. 6, no. 9, pp. 628–637, 2015.

[12] E. Brunk, K. W. George, J. Alonso-Gutierrez, *et al.*, "Characterizing strain variation in engineered e. coli using a multi-omics-based workflow," *Cell systems*, vol. 2, no. 5, pp. 335–346, 2016.

[13] J. Shi, X. Wu, H. Qi, X. Xu, and S. Hong, "Application and discoveries of metabolomics and proteomics in the study of female infertility," *Frontiers in Endocrinology*, vol. 14, p. 1 315 099, 2024.

[14] B. Alberts, "The cell as a collection of protein machines: Preparing the next generation of molecular biologists," *cell*, vol. 92, no. 3, pp. 291–294, 1998.

[15] M. de Rooij, B. Erdos, N. van Riel, and S. O'Donovan, "Physiology-informed regularization enables training of universal differential equation systems for biological applications," *bioRxiv*, pp. 2024–05, 2024.

[16] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[17] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2000.

[19] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *International conference on machine learning*, PMLR, 2017, pp. 2979–2987.

[20] C. Rackauckas, Y. Ma, J. Martensen, *et al.*, "Universal differential equations for scientific machine learning," *arXiv preprint arXiv:2001.04385*, 2020.

[21] Anonymous, "Exploring the impact of activation functions in training neural ODEs," in *Submitted to The Thirteenth International Conference on Learning Representations*, under review, 2024. [Online]. Available: https://openreview.net/forum?id=AoraWUmpLU.

[22] P. Kidger, *On neural differential equations*, 2022. arXiv: 2202.02435 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2202.02435.

[23] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[24] R. T. Q. Chen, *Torchdiffeq*, 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq.

[25] U. Habaraduwa, *Deep learning - learndynamics.ipynb*, Accessed: 2024-12-09, 2024. [Online]. Available: https://github.com/onedeeper/deeplearning/blob/main/LearnDynamics.ipynb.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, *Scikit-learn: Machine learning in python - faq*, Accessed: 2024-11-25, 2011. [Online]. Available: https://scikit-learn.org/stable/faq.html#will-you-add-gpu-support.

[27] H. Wang, L. Lu, S. Song, and G. Huang, "Learning specialized activation functions for physics-informed neural networks," *arXiv preprint arXiv:2308.04073*, 2023.

[28] W. Cui, H. Zhang, H. Chu, P. Hu, and Y. Li, "On robustness of neural odes image classifiers," *Information Sciences*, vol. 632, pp. 576–593, 2023.

[29] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *Machine Learning*, vol. 110, pp. 393–416, 2021.

[30] A. Forootani, P. Goyal, and P. Benner, *A robust sindy approach by combining neural networks and an integral form*, 2023. arXiv: 2309.07193 [math.DS]. [Online]. Available: https://arxiv.org/abs/2309.07193.

[31] V. Chubukov, A. Mukhopadhyay, C. J. Petzold, J. D. Keasling, and H. G. Martín, "Synthetic and systems biology for microbial production of commodity chemicals," *NPJ systems biology and applications*, vol. 2, no. 1, pp. 1–11, 2016.