An Empirical Investigation of Neural ODEs and Symbolic Regression for Dynamical Systems

Panayiotis Ioannou
Department of Physics
University of Cambridge
pi225@cam.ac.uk

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pl219@cam.ac.uk

Pietro Cicuta
Department of Physics
University of Cambridge
pc245@cam.ac.uk

Abstract

Accurately modelling the dynamics of complex systems and discovering their governing differential equations are critical tasks for accelerating scientific discovery. Using noisy, synthetic data from two damped oscillatory systems, we explore the extrapolation capabilities of Neural Ordinary Differential Equations (NODEs) and the ability of Symbolic Regression (SR) to recover the underlying equations. Our study yields three key insights. First, we demonstrate that NODEs can extrapolate effectively to new boundary conditions, provided the resulting trajectories share dynamic similarity with the training data. Second, SR successfully recovers the equations from noisy ground-truth data, though its performance is contingent on the correct selection of input variables. Finally, we find that SR recovers two out of the three governing equations, along with a good approximation for the third, when using data generated by a NODE trained on just 10% of the full simulation. While this last finding highlights an area for future work, our results suggest that using NODEs to enrich limited data and enable symbolic regression to infer physical laws represents a promising new approach for scientific discovery.

1 Introduction

Being able to accurately model the dynamics of complex systems or find the differential equations that govern them can help us in understanding those systems better and accelerate scientific discovery. Given the explosion of data and advances in machine learning, a key question arises: how can we best leverage experimental data to infer the dynamics of complex systems? Neural Ordinary Differential Equations (NODEs) (1) are a powerful tool for modelling dynamical systems, as their continuous-time nature mirrors that of differential equations. As a result, NODEs have been successfully applied across diverse scientific fields (2; 3). Much of the existing research has focused on improving NODE architectures (4) and assessing their robustness (5). However, a significant gap remains in evaluating their performance under noisy synthetic data or real-world conditions. Specifically, their interpolation and extrapolation capabilities with respect to boundary conditions have been under-explored. This work addresses this gap by systematically investigating NODE performance using noise-infused synthetic data from two dynamical systems that exhibit damped oscillations: the cart-pole (6) and a biological model of bacterial adaptation to changing nutrient environments (7).

While NODEs excel at modelling dynamics, their black-box nature limits interpretability. In contrast, Symbolic Regression (SR) (8) can discover the exact governing equations; however, it often requires

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

large datasets, which can be a significant challenge in experimental science. To address this, we investigate a pipeline that uses a trained NODE as a data augmentation tool for SR, assessing its ability to recover ground-truth equations from limited initial data. Our evaluation compares the performance of SR on direct ground-truth data (with and without noise) to its performance on a full dataset generated by a NODE trained on only 10% of the original simulation.

2 Methods

This work investigates the extrapolation and interpolation capabilities of NODEs using noisy synthetic data generated from two dynamical systems: the cart-pole (6) and a biological system (which we refer to as the Bio-model) (7). We also employ SR to determine if the original equations can be recovered directly from the noisy simulation data, as well as from the output generated by the NODE models. The NODE results were obtained using code adapted from the JAX-based (9) `Diffrax` library (10). `PySR` was used for the SR analysis (8), and the `phaseportrait` library (11) was used to create the phase space plot. All the details to reproduce this work are provided in the remainder of the methods section or in sections A, B and C of the appendix.

Cart-pole. Equation 1 shows the angle dynamics of the cart-pole system assuming no friction between the cart and the rail (6):

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \ddot{\theta} = \frac{g\sin\theta + \cos\theta(\frac{-F - m_p l\dot{\theta}^2 \sin\theta}{m_c + m_p}) - \frac{\mu_p \dot{\theta}}{m_p l}}{l(\frac{4}{3} - \frac{m_p \cos^2\theta}{m_c + m_p})} \tag{1}$$

where θ is the angle between the vertical and the pole, $\dot{\theta} = \frac{\mathrm{d}\theta}{\mathrm{d}t}$, F is the force acting on the cart, l is half of the length of the pole, m_p the mass of the pole, m_c the mass of the cart, μ_p is the friction in the articulation between the cart and the pole and g the gravity. Using a forward simulation, equation 1 was solved numerically for many different initial conditions. This was done using the `SciPy` library, more specifically the `integrate.odeint` python function. The first data point $[\theta_0, \omega_0]$ was at t = 0. The simulation ran until t = 10 seconds with 0.01 second time-steps. Noise of magnitude -5 to 5% (of the data point value) was added to each point using a uniform distribution. For the results in this report, the simulation was solved using: F = 0, l = 10 cm, m_p = 0.1 kg, m_c = 1 kg and μ_p = 0.0008. This corresponds to an unforced cart-pole with friction only between the pole and the cart.

Our analysis uses two separate models, each trained on a different set of initial conditions. Model A was trained on 35 boundary conditions, which are all combinations of initial angles from {0,0.6,1.2,1.8,2.4,3.14} and angular speeds from {0,2,4,6,8,10}, excluding the unstable equilibrium at [0,0]. The training data for Model A consists of 26 points from the first second of simulation for both the angle and angular speed, corresponding to a sampling rate of 25 Hz. The results in Figure 1a were generated by Model B, which was trained exclusively on the set of boundary conditions highlighted by the red box in the figure. Its training data was sampled at the same rate and time points as Model A.

Bio-model. The system's evolution is governed by the following ordinary differential equations (7), where each is presented in both a simplified and an expanded form using only the three state variables (ψ_A, ϕ_R, χ_R) and constants ($\phi_{Rmax}, \tilde{\epsilon}, k_\alpha, A, A_2, \tau_\chi$):

$$\frac{\mathrm{d}\psi_A}{\mathrm{d}t} = (\phi_{Rmax} - \phi_R)\nu_f - \lambda[\psi_A, \phi_R](1 + \psi_A) = (\phi_{Rmax} - \phi_R)k_f - \phi_R\tilde{\epsilon}\frac{\psi_A(\psi_A + 1)}{\psi_A + k_\alpha} \tag{2}$$

$$\frac{\mathrm{d}\phi_R}{\mathrm{d}t} = \lambda[\psi_A, \phi_R](\chi_R - \phi_R) = \phi_R(\chi_R - \phi_R)\tilde{\epsilon}\frac{\psi_A}{\psi_A + k_\alpha} \tag{3}$$

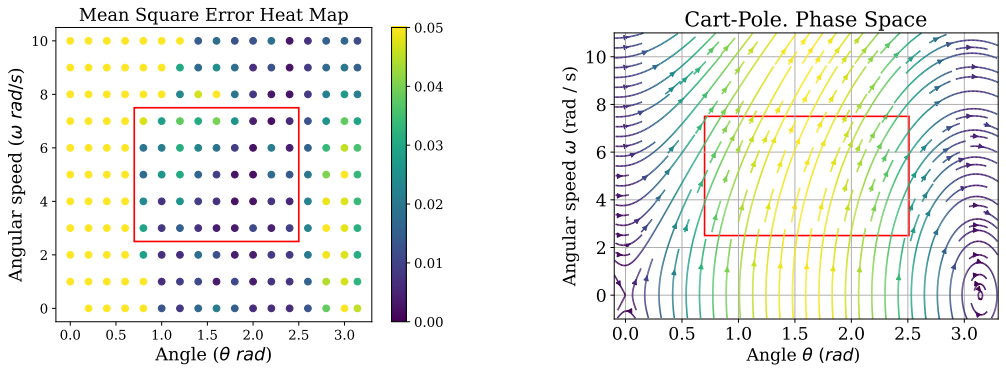
$$\frac{\mathrm{d}\chi_R}{\mathrm{d}t} = \frac{1}{\tau_\chi}(\omega_R[\psi_A] - \chi_R) = \frac{1}{\tau_\chi}(\frac{\psi_A A}{\psi_A A + A_2} - \chi_R) \tag{4}$$

where the simplified form uses $\lambda[\psi_A, \phi_R]$ which is a function of ψ_A and ϕ_R. The complete model derivation and a comprehensive description of the variables are available in (7). Appendix C provides a brief overview of the variables and constants. The system's evolution is driven by a shift in nutrient quality, represented by a change in the parameter ν. Each value of ν corresponds to a unique steady state solution for the state variables (ψ_A, ϕ_R, χ_R). For an example of these values, see Table A1 in the appendix. Experiments begin with the system in a steady state at an initial nutrient quality, ν_i. At t = 0, ν is abruptly changed to a final value, ν_f, and the system evolves toward a new steady state. This is an up-shift if $\nu_f > \nu_i$ and a down-shift if $\nu_f < \nu_i$. All experiments use a fixed ν_f = 3.78.

For the numerical simulations, we solved the model for 8 hours with a 0.01-hour time step and added noise to the data in a way similar to the cart-pole. For Model 2A, the training set consisted of 33 data points per hour over the first 4 hours, generated from $\nu_i = 2.22$ and 3.465. To generate the results in Figure 3, we trained six models, each with a different sampling rate (5, 10, 20, 33, 50, and 100 data points per hour) using data from the first hour of simulation. These models were trained on 12 distinct shifts (6 up-shifts, 6 down-shifts) with ν_i values ranging from 0.36 to 7.19 in steps of 0.62. The NODE model for SR was trained on the same 12 shifts, but with a sampling frequency of 10 data points per hour from the first 2 hours of each simulation.

3 Results and Discussion

NODE Extrapolation: (A) Cart-Pole. Figure 1a displays a mean squared error (MSE) heatmap for Model B, where each point represents the initial conditions of a time series. As expected, points within the training region (inside the red rectangle) exhibit a low MSE. Surprisingly, we observe low-MSE regions outside these boundaries, with values comparable to the training set. An analysis of the system's phase space reveals a strong correlation with this phenomenon (Figure 1b). Points on the same phase space trajectory as training data points tend to have a lower MSE. This indicates that NODEs can generalise effectively to regions that share the same dynamical properties as the training set. This finding emphasizes a key insight: when designing a training set for dynamical systems, the goal should be to sample diverse dynamics rather than simply a large number of initial conditions. Figure 2a displays the results of model A on an interpolated, [1.4, 5], initial condition. Trained on data only from the first second, the model accurately captures the dynamics of the system and successfully extrapolates over a duration five times longer than its training period. **(B) Bio-model.** Figure 2b displays the performance of Model 2A, which was trained exclusively on only two up-shift simulations. The model accurately predicts the down-shift ($\nu_i = 5.95$ to $\nu_f = 3.78$) response, with the error for each point being less than 5%, despite having no down-shift data included in the training.

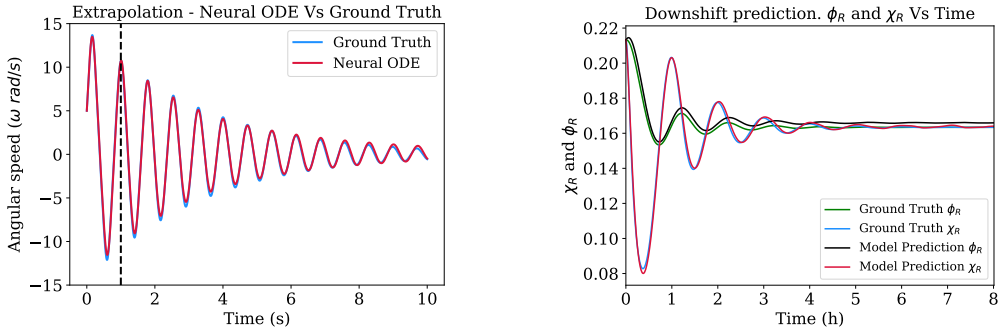


(a) (b)

Figure 1: The red rectangle in both plots represents the training region. **(a) The MSE heatmap for Model B (cart-pole).** The low-error zones outside the training region highlight the model's ability to extrapolate. This occurs because the model learned the underlying dynamics from trajectories within the training set, allowing it to accurately predict other points along the same dynamic path. **(b) The cart-pole phase space.** The colour represents the magnitude of the angular speed and acceleration.

The Impact of Training Data Sampling Frequency on Model Performance. Figure 3 shows the MSE for models trained with varying sampling frequencies on only the first hour of data. The most significant finding is the remarkable consistency of the 8-hour MSE, which shows no significant difference across models. This suggests that high-quality long-term predictions can be achieved with a minimal amount of training data. In contrast, the 1-hour MSE for the two lowest sampling frequencies is notably higher. This is a direct result of the extreme sparsity of the training data; with as few as six training points per variable per shift, noise disproportionately affects the model's fit, leading to a higher interpolation error. The MSE values are averaged over 10 independent runs for each model and across 19 shifts (ν_i from 0.98 to 6.57 in steps of 0.31).

Symbolic Regression. Our SR analysis, presented in Table 1, was conducted on two distinct datasets: ground truth simulation data and data generated by a trained NODE model. First, using ground truth



(a) (b)

Figure 2: **(a) Cart-pole. Model A Time Extrapolation Performance.** Model A, trained on data from only the first second, accurately predicts the system's dynamics for an interpolated initial condition, [1.4, 5]. The model demonstrates strong generalization by successfully extrapolating five times the length of the training data. **(b) Bio-model. Results from Model 2A,** which was trained solely on two up-shift datasets. The model successfully predicts a down-shift (ν_i = 5.95 to ν_f = 3.78), an event it was never exposed to in training. While small amplitude errors (less than 5%) are observed, the model effectively learns and extrapolates the underlying shift dynamics.

data from a single up-shift simulation (ν_i = 2.53 to ν_f = 3.78), we successfully recovered all three target equations when λ was included as an input variable. However, when the analysis was limited to the three main state variables (ψ_A, ϕ_R, χ_R), only Equation 4 was recovered. One of the primary reasons for this failure is due to the the rational term within $\lambda = 10.04 \cdot \frac{\psi_A \phi_R}{\psi_A + k_a}$ being effectively masked by the data. With ψ_A being, on average, 8 times larger than k_a, the denominator $(\psi_A + k_a)$ is dominated by ψ_A. This makes the rational term $\frac{\psi_A \phi_R}{\psi_A + k_a}$ approximately equal to ϕ_R, hiding the true structure and the constant k_a from the SR algorithm.

The presence of 5% noise in the ground truth data significantly hindered discovery. As shown in Table 2, SR failed to recover the true structure of Equation (2), finding only the drastic simplification $1.410 - \lambda$. The mean of λ over the shift is equal to 1.410. The target derivatives naturally tend to zero as the system approaches equilibrium, creating a flat fitness landscape; the addition of 5% noise further masks this already-weak signal, making discovery more difficult. Full `PySR` outputs for this equation and dataset can be found in Table A5 in Appendix D.

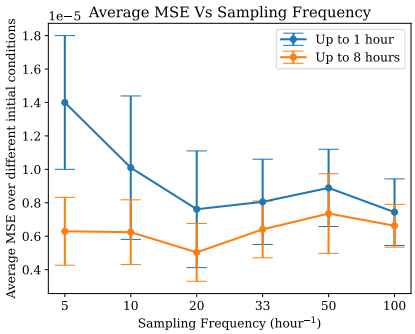


Figure 3: **Impact of sampling frequency on model performance.** The legend shows the time regions for MSE calculation. The 8-hour MSE is consistent across all models, demonstrating that long-term predictions are achievable with sparse data. In contrast, the 1-hour MSE rises sharply for the two lowest sampling frequencies, as the model's fit is highly sensitive to noise when trained on only a handful of points. Errobars indicate the standard deviation over 10 independent runs.

Table 1: **Results of our SR analysis.** On ground truth data (with λ as input), SR recovered all three governing equations but struggled in the presence of 5% noise, finding only one of the three. On NODE-generated data, SR recovered two of three equations and a good approximation for Equation (2). Interestingly, it seems that the NODE acts as a denoising filter, enabling better SR performance on the noisy dataset. * The algorithm does not find the correct equation but it finds a decent approximation (see Table 2).

Equations (with λ)	Ground Truth No noise	Ground Truth 5% noise
2	✓	×
3	✓	✓
4	✓	×

Equations (with λ)	NODE No noise	NODE 5% noise
2	×*	×*
3	✓	✓
4	✓	✓

Table 2: **SR results for Equation** (2). Unlike the other two governing equations (which SR recovered successfully from NODE generated data), this equation's discovery is sensitive to the data source. While SR finds the true structure from Ground Truth data, it fails on NODE-generated data by omitting one of the low signal terms, and it fails completely on the noisy Ground Truth data. The mean value of $\lambda\psi_A$ is 0.045. Loss is the MSE between the SR result and the dataset it was trained on.

Ground Truth for Equation (2)		$2.079 - 3.78 \times \phi_R - \lambda - \lambda \times \psi_A$
Dataset	**Loss (MSE)**	**Symbolic Regression Result**
Ground Truth (No noise)	2.09×10^{-8}	$2.076 - 3.77 \times \phi_R - \lambda - \lambda \times \psi_A$
Ground Truth (5% noise)	8.12×10^{-3}	$1.410 - \lambda$
NODE (No noise)	1.91×10^{-5}	$2.024 - 3.54 \times \phi_R - \lambda - \psi_A$
NODE (5% noise)	2.16×10^{-4}	$2.006 - 3.67 \times \phi_R - \lambda$

SR applied to data from our NODE (trained on ground truth data with or without noise) recovered two of the three governing equations, (3) and (4). It failed to recover the full form of Equation (2) but nonetheless found good approximations (Table 2). This result also demonstrates the NODE's ability to act as a denoiser, enabling SR to achieve a better result on noisy data. The SR solution derived from the NODE trained on noisy data fails to identify the $-\lambda\psi_A$ term. This failure is attributable to a low signal-to-noise ratio: the term's magnitude ($0.01 < \lambda\psi_A < 0.13$, with a mean of 0.045) is negligible compared to the equation's dominant constant, 2.079. Interestingly, `PySR` compensates for this omitted term by absorbing its mean value into the discovered constant: the algorithm finds 2.006 instead of 2.079. Extended results from some of the SR runs are available in Appendix D.

These preliminary findings can be improved, most critically by extending the SR analysis beyond a single-shift simulation to diverse, multi-condition data. Other avenues include improving the NODE model, as its training data was not fully optimised to maximise generalisation in this work, or using more advanced architectures (e.g., Neural CDEs (12)). The SR search itself could also be strengthened with physical priors (e.g., unit matching) or alternative frameworks like SINDy (13).

4 Conclusions and Future Work

This study investigated the generalization properties of Neural Ordinary Differential Equations (NODEs) and their potential to augment data for symbolic regression (SR). Using noisy, synthetic data from two damped oscillatory systems, we demonstrated that NODEs can learn the underlying dynamics and generalize well to novel initial conditions and time periods, provided the trajectories for the new conditions are dynamically similar to those in the training data. Our results, based on sparse and noisy data, highlight that training on diverse dynamics is crucial for generalization, surpassing the need for dense sampling. We also demonstrated that SR can recover the true governing equations from this noisy data, although its performance is sensitive to the choice of input variables and equation complexity. We tested a pipeline for data-scarce settings: a NODE, trained on only 10% of the data, generated a full dataset for SR. This pipeline successfully recovered two of the three governing equations and found a good approximation for the third. This pipeline is promising, as these outcomes can be improved with future work, such as implementing techniques like unit matching within the SR framework, using more augmented data or exploring other SR frameworks. Ultimately, this research lays the groundwork for a promising scientific discovery tool, particularly for data-scarce domains, by using NODEs to enrich limited experimental data and enable symbolic regression to infer underlying physical laws.

Acknowledgments and Disclosure of Funding

P.I. was supported by the Engineering and Physical Sciences Research Council Centre for Doctoral Training in Sensor Technologies for a Healthy and Sustainable Future [EP/S023046/1]. We would like to thank Rossana Droghetti and Marco Cosentino Lagomarsino for helpful discussions regarding the Bio-model. Large language models, such as Gemini and ChatGPT, were used to enhance clarity and improve language use.

References

[1] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural Ordinary Differential Equations," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[2] G. D. Portwood, P. P. Mitra, M. D. Ribeiro, T. M. Nguyen, B. T. Nadiga, J. A. Saenz, M. Chertkov, A. Garg, A. Anandkumar, A. Dengel, *et al.*, "Turbulence forecasting via neural ODE," *arXiv preprint arXiv:1911.05180*, 2019.

[3] J. Lu, K. Deng, X. Zhang, G. Liu, and Y. Guan, "Neural-ODE for pharmacokinetics modeling and its advantage to alternative machine learning models in predicting new dosing regimens," *Iscience*, vol. 24, no. 7, 2021.

[4] T. Zhang, Z. Yao, A. Gholami, J. E. Gonzalez, K. Keutzer, M. W. Mahoney, and G. Biros, "ANODEV2: A coupled neural ODE framework," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] H. Yan, J. Du, V. Y. Tan, and J. Feng, "On robustness of neural ordinary differential equations," *arXiv preprint arXiv:1910.05513*, 2019.

[6] R. V. Florian, "Correct equations for the dynamics of the cart-pole system," *Center for Cognitive and Neural Studies (Coneural), Romania*, vol. 63, 2007.

[7] R. Droghetti, P. Fuchs, I. Iuliani, V. Firmano, G. Tallarico, L. Calabrese, J. Grilli, B. Sclavi, L. Ciandrini, and M. Cosentino Lagomarsino, "Incoherent feedback from coupled amino acids and ribosome pools generates damped oscillations in growing E. coli," *Nature communications*, vol. 16, no. 1, p. 3063, 2025.

[8] M. Cranmer, "Interpretable machine learning for science with PySR and SymbolicRegression. jl," *arXiv preprint arXiv:2305.01582*, 2023.

[9] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[10] P. Kidger, *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[11] H. L. Victor and F. L. Unai, "Make phase portraits in 2d and 3d in Python." `https://github.com/phaseportrait/phaseportrait`.

[12] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 6696–6707, Curran Associates, Inc., 2020.

[13] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

Appendix

The Appendix includes: (A) details on the training methodology for the Neural ODE models; (B) a description of the setup and parameters used for the symbolic regression (SR) analysis; (C) additional details on the Bio-model, including definitions of key variables, equations, and constants; and (D) the SR outputs from PySR using the data generated from the NODE model that was trained on the ground truth data with 5% noise and for equation 2 also the noisy ground truth data.

A Neural ODE Training Details

All models were trained using a standard multi-layer perceptron architecture with two hidden layers, each containing 20 nodes. The `Adabelief` optimizer from the `optax` library was used for all training runs.

Cart-Pole System. For the cart-pole experiments, all models were trained for 100,000 iterations with a learning rate of 0.003. Model A utilized a batch size of 16, whereas Model B used a batch size of 20.

Bio-model. For the biological system models, Model 2A was trained for 100,000 iterations with a learning rate of 0.001 and a batch size of 1. The Models used to generate the results in Figure 3 were trained for 100,000 iterations with a learning rate of 0.003 and a batch size of 10. To generate the data for the SR analysis, the Neural Ordinary Differential Equation (NODE) models were trained for 150,000 iterations with a learning rate of 0.003 and a batch size of 10. To help prevent convergence to a local minimum, all Bio-model training included an initial 500 iterations using only the first 10% of the data and a learning rate of 0.003.

B Symbolic Regression Details

We employed the `PySR` library (v1.5.9) to perform symbolic regression and identify the governing equations of the system. The search space for candidate equations included a standard set of primary binary operations (addition, subtraction, multiplication, division) and the unary inverse operator, `inv(x)=1/x`. We ran the search for 800 iterations with a population size of 50, setting the maximum equation size to 25 and constant complexity to 2.

To serve as the target values for the symbolic regression, the time derivatives of the three state variables (ψ_A, ϕ_R, χ_R) were calculated from the data using the `NumPy gradient` function. To improve data fidelity, we truncated the time series by removing the first 10 datapoints (0.1 h) and the final 100 datapoints (1.0 h). This preprocessing mitigates two issues: (1) high numerical error from the finite-difference gradient at the initial points, and (2) uninformative, steady-state data, which introduces a low signal-to-noise ratio that can hinder the SR search. While this approach provided a sufficient approximation, using a more robust gradient estimation technique could further improve the accuracy of our symbolic regression results.

C Bio-model: Additional information

A complete description of the model, along with all variables, constants, and the full derivation, is available in (7). The model, described by equations 2, 3, and 4 , simulates the adaptive behaviour of *E. coli* bacteria in response to changing nutrient conditions. The system is defined by three key state variables:

- ψ_A: The ratio of free amino acid mass to the total protein mass within the bacteria.
- ϕ_R: The mass fraction of the proteome dedicated to all ribosome-related proteins.
- χ_R: The gene regulatory function, which quantifies the fraction of protein synthesis allocated for ribosome production.

The bacterial growth rate, λ, is a key parameter defined by the relationship $\lambda = \epsilon\phi_R$. Here, ϵ represents the translation rate, and is given by the function $\epsilon = \tilde{\epsilon}\frac{\psi_A}{\psi_A + k_a}$. The constant k_a = 0.005 is the amino acid uptake efficiency, and $\tilde{\epsilon}$ = 10.04 h^{-1} is the theoretical maximum elongation rate. The

system's behaviour is governed by the nutrient quality parameter, ν_f, which is solely dependent on the potency of the post-shift nutrient. In this work, we set ν_f = 3.78 h^{-1}. In the steady state, where the time derivatives are zero, each value of ν corresponds to two solutions for the state variables (ψ_A, ϕ_R, χ_R). One of these solutions yields a negative value for ψ_A, while the other is always positive. Since ψ_A represents a physical quantity and must therefore be non-negative, the negative solution is not biologically viable. Consequently, there exists only one unique and physically meaningful steady-state solution for each value of ν, which also corresponds to a unique growth rate, λ. Examples of these values are presented in Table A1.

The RNA Polymerase allocation on ribosomal genes, ω_R, is given by the equation: $\omega_R[\psi_a] = \frac{K_G}{K_G+[G]}$ where $[G]$ is the (p)ppGpp, guanosine tetraphosphate or pentaphosphate, concentration which is given by $[G] = CG_{Ref}(\frac{\psi_a+k_a}{\psi_a} - 1)$. In this formulation, we define the shorthand variables $A = K_G$ and $A_2 = CG_{Ref} * k_a$. For this work, the constants are set to the following values: C = 4.6, G_{Ref} = 101.46 μM, K_G = 14.5 μM, ϕ_{Rmax} = 0.55 and τ_x = 1/6 h.

Note that the values for the constants K_G and G_{Ref} differ from those presented in the original reference (7). The referenced paper used values from newer experimental data, such as K_G = 8.07 μM and G_{Ref} = 55.7 μM. Despite these differences, the core results and conclusions of this work remain unchanged.

Table A1: **Steady-state solutions and their corresponding growth rates.** The table presents the unique, biologically viable steady-state values for the state variables (ψ_A, ϕ_R, χ_R) and their associated growth rates, with each solution determined by a specific value of the parameter ν.

At Steady State			
ν	growth rate (λ)	ψ_a	ϕ_R and χ_R
2.53	1.05	0.0233	0.127
3.78	1.41	0.0314	0.163
5.95	1.92	0.0436	0.213

D Symbolic Regression (PySR) outputs

The following tables detail the `PySR` discovery results using data from the NODE model that was trained on ground truth data with 5% noise. Tables A2, A3, and A4 show the outputs for equations (2), (3), and (4), respectively. Table A5 shows the `PySR` discovery results using the ground truth data with the 5% noise. For clarity, we display equations up to a complexity of 20. The reported `Loss` is the MSE between a discovered equation and the input dataset to the SR, not the ground truth. An asterisk (*) indicates the equation with the highest score, representing `PySR`'s optimal trade-off between accuracy and complexity. **Bold** highlights the equation whose structure is most similar to the true governing equation.

Table A2: PySR results for Equation (2) (NODE, 5% noise data). **Bold** highlights the structure most similar to the true governing equation; * is PySR's highest score.

Complexity	Loss (MSE)	Score	Equation
1	6.296×10^{-3}	0.000	ψ_A
2	4.548×10^{-3}	0.325	-0.0053729056
3	3.280×10^{-3}	0.327	$\phi_R - \chi_R$
4	1.109×10^{-3}	1.084*	$1.4028348 - \lambda$
6	6.885×10^{-4}	0.238	$(1.5672859 - \phi_R) - \lambda$
8	4.006×10^{-4}	0.271	$((1.7317353 - \phi_R) - \lambda) - \phi_R$
9	2.160×10^{-4}	**0.618**	$\mathbf{2.0058749 - (\lambda + (\phi_R \times 3.6670127))}$
11	9.418×10^{-5}	0.415	$(2.1844785 - (\lambda + (\phi_R \times 4.556726))) - \psi_A$
13	6.408×10^{-5}	0.193	$1.8036835 - ((((\phi_R \times \phi_R) \times 13.594807) + \lambda) + \psi_A)$
14	2.118×10^{-5}	1.107	$((\phi_R \times -5.776956) - (\lambda + (\chi_R + -2.515695))) \times 0.69243747$
16	1.523×10^{-5}	0.165	$(((-5.5431213 + \chi_R) \times \phi_R) - ((\chi_R + -2.4505582) + \lambda)) \times 0.7387968$
17	1.449×10^{-5}	0.050	$((\phi_R \times -5.272122) - (((\chi_R \times 0.789321) + -2.398444) + \lambda)) \times 0.74833924$
18	1.252×10^{-5}	0.146	$((\phi_R \times (\chi_R + -5.5318727)) - ((\chi_R + -2.4487023) + \lambda)) \times (0.7895749 - \psi_A)$
19	1.092×10^{-5}	0.137	$((\phi_R \times -5.1778607) - (\lambda + (-2.376086 + (\chi_R \times 0.7470051)))) \times (0.810564 - \psi_A)$
...	...	...	...

Table A3: PySR results for Equation (3) (NODE, 5% noise data). **Bold** highlights the structure most similar to the true governing equation; * is PySR's highest score.

Complexity	Loss (MSE)	Score	Equation
1	2.042×10^{-3}	0.000	ψ_A
3	1.761×10^{-4}	1.225	$\chi_R - \phi_R$
5	3.123×10^{-5}	**0.865**	$\boldsymbol{(\chi_R - \phi_R) \times \lambda}$
6	2.494×10^{-5}	0.225	$(\phi_R - \chi_R)/ - 0.7338049$
7	2.412×10^{-5}	0.034	$(\lambda - \phi_R) \times (\chi_R - \phi_R)$
8	9.552×10^{-6}	0.926	$(\chi_R - \phi_R)/(0.9726795 - \chi_R)$
9	9.493×10^{-6}	0.006	$(\phi_R \times -1.3376828) + (\chi_R \times 1.3615563)$
10	7.311×10^{-6}	0.261	$(\chi_R \times \psi_A) - ((\phi_R - \chi_R) \times 1.2554784)$
11	2.109×10^{-6}	1.243*	$((-1.1330749 - \chi_R) \times (\phi_R - \chi_R)) - -0.0028907147$
13	1.591×10^{-6}	0.141	$(-1.1011528 - \chi_R) \times (\phi_R + ((\psi_A \times -0.07154481) - \chi_R))$
15	6.273×10^{-7}	0.465	$(\phi_R \times -1.1626418) + (\chi_R \times (((\lambda \times \psi_A) \times \lambda) - -1.1181651))$
16	6.104×10^{-7}	0.027	$(\phi_R - (\chi_R \times (\psi_A + 0.98330754))) \times (-0.6877947 - (\lambda \times 0.3857772))$
17	3.750×10^{-7}	0.487	$(\phi_R \times -1.1737428) + (((\psi_A \times (\lambda \times (\lambda - \psi_A))) - -1.1311564) \times \chi_R)$
18	1.933×10^{-7}	0.663	$(\chi_R \times ((((\lambda \times \lambda) + -0.37667426) \times \psi_A) - -1.1579369)) + (\phi_R \times -1.1897844)$
19	1.800×10^{-7}	0.071	$((((\chi_R \times (\lambda \times \lambda)) - \psi_A) \times \psi_A) + (\phi_R \times -1.1864315)) - (-1.1489322 \times \chi_R)$
...	...	...	...

Table A4: PySR results for Equation (4) (NODE, 5% noise data). **Bold** highlights the structure most similar to the true governing equation; * is PySR's highest score.

Complexity	Loss (MSE)	Score	Equation
1	3.26×10^{-2}	0.000	ψ_A
3	3.25×10^{-2}	0.001	$\chi_R - \phi_R$
4	3.13×10^{-2}	0.036	$\psi_A + -0.034593984$
5	1.78×10^{-2}	0.568	$(\psi_A/\phi_R) - \chi_R$
7	4.25×10^{-3}	0.715	$(\phi_R \times -21.217772) + 3.486964$
9	2.91×10^{-3}	0.190	$((\psi_A/(\psi_A + \phi_R)) - \chi_R)/\chi_R$
10	7.57×10^{-4}	1.345	$((\psi_A/(\psi_A + \phi_R)) - \chi_R) \times 4.9153194$
11	9.29×10^{-5}	**2.098***	$\boldsymbol{((\psi_A/(\psi_A + 0.16037777)) - \chi_R) \times 5.927795}$
13	4.36×10^{-5}	0.378	$((\psi_A/(\psi_A + 0.16027561)) - \chi_R) \times (4.554471 + \lambda)$
15	2.95×10^{-5}	0.196	$((\lambda + \lambda) + 3.1795135) \times ((\psi_A/(\psi_A + 0.16015515)) - \chi_R)$
16	2.93×10^{-5}	0.005	$(((\psi_A/(\psi_A + 0.1601735)) - \chi_R) \times 1.8685215) \times (\lambda + 1.7985661)$
17	2.21×10^{-5}	0.282	$(\lambda + ((\lambda - \chi_R) + 3.3858838)) \times ((\psi_A/(\psi_A + 0.16015515)) - \chi_R)$
18	1.54×10^{-5}	0.364	$(((\psi_A/(\psi_A + 0.16009086)) - \chi_R) \times 2.5824206) \times (\lambda + (1.127972 - \chi_R))$
20	1.51×10^{-5}	0.010	$((\psi_A/(\psi_A + 0.16009125)) - \chi_R) \times (((0.9849129 + (\lambda - \chi_R)) \times 2.8481526) - \chi_R)$
...	...	...	...

Table A5: PySR results for Equation (2) (Ground Truth, 5% noise data). **Bold** highlights the structure most similar to the true governing equation; * is PySR's highest score.

Complexity	Loss	Score	Equation
1	1.185×10^{-2}	0.000	ψ_A
2	1.018×10^{-2}	0.152	-0.0044986876
3	9.006×10^{-3}	0.122	$\phi_R - \chi_R$
4	7.639×10^{-3}	**0.165***	$\boldsymbol{1.4092975 - \lambda}$
6	7.289×10^{-3}	0.023	$0.23274253 - (\chi_R \times \lambda)$
7	6.872×10^{-3}	0.059	$((\psi_A/\chi_R) - \chi_R) - \psi_A$
8	6.736×10^{-3}	0.020	$\psi_A + (0.19986038 - (\lambda \times \chi_R))$
9	5.970×10^{-3}	0.121	$(\psi_A - (\chi_R \times (\chi_R + \psi_A)))/\phi_R$
10	5.926×10^{-3}	0.007	$(\psi_A - ((\chi_R \times \chi_R) \times 1.1969346))/\phi_R$
11	5.817×10^{-3}	0.019	$(\psi_A - (\chi_R \times (\chi_R + (\chi_R \times \chi_R))))/\phi_R$
12	5.734×10^{-3}	0.014	$(\psi_A - (\lambda \times ((\chi_R \times 0.83034015) \times \chi_R)))/\phi_R$
14	5.609×10^{-3}	0.011	$(\psi_A - (((\chi_R + \psi_A) \times \lambda)/(1.460721/\chi_R)))/\phi_R$
15	5.608×10^{-3}	0.000	$(((\psi_A - (\chi_R \times (\chi_R \times \lambda)))/\phi_R) \times 0.7639032) - -0.03282497$
16	5.562×10^{-3}	0.008	$(((\chi_R \times ((\lambda \times \lambda) \times (-0.09359126/\phi_R))) \times \chi_R) + \psi_A)/\phi_R$
17	5.547×10^{-3}	0.003	$((((\psi_A - ((\chi_R \times \lambda) \times \chi_R)) \times 0.1317986)/\phi_R) - -0.0053570317)/\phi_R$
18	5.519×10^{-3}	0.005	$((\psi_A + (\lambda \times ((\lambda \times ((-0.080160506/\phi_R) \times \chi_R)) \times \chi_R)))/\phi_R) - \psi_A$
19	5.472×10^{-3}	0.009	$(\psi_A + (\chi_R \times (((\lambda \times (\lambda \times -0.027622199))/\phi_R) + (0.30923742 - \chi_R))))/\phi_R$
...	...	...	...