Physics-Constrained Learning for Scientific Discovery: Inference in Differential Equations and Inverse Design via Generative Models

by

Lena Podina

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2026

© Lena Podina 2026

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Hessam Babaee Professor, Dept. of Mechanical Engineering and Materials Science, University of Pittsburgh
Supervisor(s):	Mohammad Kohandel Professor, Dept. of Applied Mathematics, University of Waterloo Ali Ghodsi Professor, Cheriton School of Computer Science, University of Waterloo
Internal-External Member:	David Del Rey Fernández Professor, Dept. of Applied Mathematics, University of Waterloo
Other Member(s):	Justin Wan Professor, Cheriton School of Computer Science, University of Waterloo Pascal Poupart Professor, Cheriton School of Computer Science, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis is based on four published articles [92, 129, 130, 131], and one preprint [127] which I have coauthored. My contributions in each chapter are detailed below:

Chapter 2: Universal Physics-Informed Neural Networks. This chapter is based on an article published at ICML 2023 [92]. This was a collaboration with Brydon Eastman and Mohammad Kohandel, my supervisor. The high-level concept emerged from discussion with Brydon Eastman and Mohammad Kohandel. Brydon performed initial proof-of-concept experiments that compare UPINNs and UDEs, and he performed the experiments pertaining to Burger's PDE. I performed all other experiments in this chapter. Brydon and I wrote the paper together. Mohammad Kohandel provided high-level guidance on the project and comments on the paper.

Chapter 3: Learning Chemotherapy Drug Action via Universal Physics-Informed Neural Networks. This chapter is based on an article published in *Pharmaceutical Research* [129]. This was a collaboration with my supervisors Mohammad Kohandel and Ali Ghodsi. I performed all experiments and analyses in this work, and wrote the paper. Mohammad Kohandel and Ali Ghodsi provided high-level guidance on the project and comments on the paper.

Chapter 4: Conformalized Physics-Informed Neural Networks. This chapter is based on an article published in the ICLR 2024 AI for Differential Equations in Science Workshop [131]. This was a collaboration with Mahdi Torabi Rad, and Mohammad Kohandel. Through discussions between Mahdi Torabi Rad, Mohammad Kohandel and I, we came up with the high-level concept. I performed all experiments and analyses. Mahdi and I wrote the paper together. Mohammad Kohandel provided high-level guidance on the project and comments on the paper.

Chapter 5: Enhancing Symbolic Regression with Dimensional Analysis. This chapter is based on a preprint [127], which is currently under submission to a journal. This work was a collaboration with Diba Darooneh, Joshveer Grewal, and Mohammad Kohandel. Diba Darooneh performed some symbolic regression analyses in an early version of the paper, but these were omitted from the current version. Mohammad Kohandel and I came up with the conceptual framework for the approach, and I performed and analyzed all experiments that are included in this chapter. All authors contributed to writing the paper. Mohammad Kohandel provided high-level guidance on the project and comments on the paper.

Chapter 6: Catalyst GFlowNet for Electrocatalyst Design. This chapter is based on an article published in the NeurIPS 2025 AI for Materials Workshop [130]. This

is a joint work with Christina Humer, Alexandre Duval, Victor Schmidt, Ali Ramlaoui, Shahana Chatterjee, Yoshua Bengio, Alex Hernández-García, David Rolnick, Ali Ghodsi, Mohammad Kohandel and Félix Therrien. This work was done while a research intern at the Mila Quebec AI Institute between Sept 2024 and April 2025, and the collaboration was continued remotely throughout the rest of my PhD. I worked extensively on the GFlowNet framework, improving its performance and efficiency. I added the relaxation step to the framework, implemented the composite reward, implemented the charge check for OER, and trained the OER DepFAENet proxy. I performed and analyzed all experiments included in this chapter. Félix closely supervised this work, but all authors offered guidance on this project. I wrote the bulk of the manuscript, and Christina Humer, Alexandre Duval, Victor Schmidt, Ali Ramlaoui, Shahana Chatterjee, Yoshua Bengio, Alex Hernández-García, David Rolnick, and Félix Therrien improved the draft and offered comments on the writing.

Abstract

Scientific discovery increasingly relies on machine learning (ML), but many scientific problems involve sparse data, physical constraints, and large combinatorial search spaces. In these regimes, ML can suffer from generalization and robustness issues, including generating outputs that violate physics constraints known *a priori*. Physics-informed machine learning aims to integrate physical constraints into ML frameworks with the goal of guaranteeing physically sound outputs, especially to scientific problems. In supervised machine learning, physics-informed neural networks (PINNs) have been influential in handling problems that traditional differential equation solvers struggle with; in model discovery, symbolic regression algorithms can discover closed-form models that best describe a dataset; within generative models, generative flow networks (GFlowNets) are used to perform inverse design of drugs, antibiotics, and materials.

This thesis develops physics-informed machine learning methods that integrate prior scientific knowledge into predictive, inverse, and generative models. We make three main contributions to physics-informed machine learning: a contribution in the domain of PINNs; a contribution in the domain of symbolic regression; a contribution to materials discovery via GFlowNets. In the domain of PINNs, we introduce Universal PINNs, which can be used to learn unknown components of differential equations from sparse or noisy data, and we apply them to discover the best form for the drug action of a chemotherapeutic, testing the method both on synthetic and experimental data (Chapters 2 and 3). Then, we integrate PINNs with conformal prediction, enabling PINNs to output confidence intervals with provable guarantees on both parameter fits and differential equation solutions (Chapter 4). For our contribution in the domain of symbolic regression (Chapter 5), we augment the efficiency of symbolic regression algorithms with dimensional analysis, and showcase the improvement in the performance of a well-known symbolic regression algorithm, PySR. In the domain of materials discovery (Chapter 6), we build a framework for catalyst discovery, for the application of hydrogen energy storage. In this work, we integrate ML-based relaxation, reward shaping, and action space constraints to generate stable and efficient catalysts. For two separate chemical reactions, we rediscover the best known catalysts within a constrained search space.

Taken together, these contributions show that physical constraints can improve sample efficiency and reliability in scientific machine learning. Future work will integrate Universal PINNs and uncertainty quantification more tightly; GFlowNets and differential equation models could be applied together to a new scientific problem; the catalyst discovery framework could be tested experimentally and augmented with an active learning loop, in order to discover truly new materials.

Acknowledgements

I would like to sincerely thank my supervisors Dr. Mohammad Kohandel and Dr. Ali Ghodsi for their invaluable guidance, support and expertise throughout the last 5 years. You have greatly shaped my path as a researcher, and you've helped me learn how to think critically about new ideas. Thank you as well for connecting me with other academics, and providing me with opportunities for attending conferences and giving talks. I also appreciate the sense of community that you created within your respective labs.

Drs. David Rolnick, Yoshua Bengio, Félix Therrien, Alex Hernández-García: I'm grateful I had the opportunity to work with you. It was very rewarding to collaborate with you for the past two years, and I've learned so much. Thank you. Thanks as well to everyone I worked with in David's and Yoshua's labs: Divya, Shahana, Christina, Alexandre, Ali, Victor and Sasha.

Thank you to my other coauthors: Brydon, Dorsa, Johanna, Diba, Josh, Mahdi, Yuannong, Jiwook, Sebastian, Dr. Brian Ingalls, and others. I really appreciated working with all of you – science is a collaborative effort, and working together makes it more fun too. Thank you Cam and Michelle for your guidance and encouragement – we didn't directly collaborate but I appreciate your perspective and the several chats we had about career and research.

Thank you to my friends: Ati, Mariia, and others. Your support has been invaluable. I wish you all the best in your careers and I hope we can stay in touch despite pursuing our separate paths.

Dr. Kevin Leyton-Brown, Greg d'Eon, and Dr. Hedayat Zarkoob: Thank you for your guidance and mentorship during my time at the GT-DT lab. This time was influential in my decision to go to grad school. Kevin and Greg, glad we had a chance to catch up at Upper Bound.

Finally, thank you to my family: Yulia, Stanislav, Larisa, Valerii. Thank you for your support. Also, I am so glad to be in Canada and to have had the opportunity to pursue a PhD here. Thank you as well for providing me with a roof over my head and food during the last few thesis-writing months, after a fire swept through my building in February 2026.

Dedication

To my family

Table of Contents

Examining Committee Membership ii

Author's Declaration iii

Statement of Contributions iv

Abstract vi

Acknowledgements vii

Dedication viii

List of Figures xiii

List of Tables xviii

1 Introduction: Physics-Informed Machine Learning 1

1.1 Thesis summary 4

2 Universal physics-informed neural networks: symbolic differential operator discovery with sparse data 7

2.1 Introduction 8

2.2 Background 9

2.3 Methods 10

2.4 Results . . . 12
2.4.1 Lotka-Volterra System . . . 13
2.4.2 Viscous Burger's Equation . . . 16
2.4.3 Cell Apoptosis Model . . . 16
2.5 Discussion and Conclusion . . . 18

3 Learning Chemotherapy Drug Action via Universal Physics-Informed Neural Networks 29
3.1 Introduction . . . 30
3.2 Background . . . 32
3.2.1 Physics-informed Neural Networks . . . 32
3.2.2 Chemotherapy modelling . . . 33
3.3 Methods . . . 34
3.3.1 General approach . . . 34
3.3.2 Application to the identification of a cell-kill strategy . . . 36
3.3.3 Application to parameter identification for several drug dosages . . . 39
3.3.4 Application to in-vitro experimental data . . . 40
3.4 Results . . . 42
3.4.1 Identification of drug action (Log-kill, NS, E-max) from synthetic data 42
3.4.2 Inference and interpolation of dose-dependent logisitic growth parameters . . . 46
3.4.3 Learning doxorubicin dynamics . . . 50
3.5 Discussion . . . 56
3.6 Conclusion . . . 57

4 Conformalized Physics-Informed Neural Networks 63
4.1 Introduction . . . 63
4.2 Background . . . 64
4.3 Methodology . . . 66

4.3.1 Conformalizing the PINN surrogate solution (forward problem) . . 66
4.3.2 ODE parameter estimate (inverse problem) 68
4.3.3 Model architecture and training 68
4.4 Results 69
4.4.1 Forward problem 69
4.4.2 Inverse problem 71
4.5 Conclusion and Discussion 72

5 Enhancing Symbolic Regression with Dimensional Analysis 75
5.1 Introduction 76
5.2 Background 78
5.2.1 Symbolic Regression 78
5.2.2 Buckingham π Theorem 79
5.2.3 Ipsen's Method of Dimensional Analysis 81
5.3 Methods 82
5.3.1 Full Discovery of Algebraic Equations 82
5.3.2 Discovery of Differential Equation Hidden Terms 84
5.4 Results 91
5.4.1 PySR with Buckingham π theorem 91
5.4.2 Symbolic Regression with Ipsen's method and UPINNs 92
5.5 Conclusion and Discussion 98

6 Catalyst GFlowNet for Electrocatalyst Design 104
6.1 Introduction 105
6.2 Background 106
6.2.1 Catalysts for hydrogen energy storage 107
6.2.2 Generative Flow Networks 108
6.2.3 Machine learning for catalyst discovery 110

6.3 Methodology . . . 111
6.3.1 Architecture . . . 111
6.3.2 Case study: Hydrogen Evolution Reaction . . . 113
6.3.3 Case study: Oxygen Evolution Reaction . . . 116
6.4 Results . . . 119
6.4.1 Effect of relaxation on lattice parameters . . . 119
6.4.2 Hydrogen Evolution Reaction: Adsorption Energy Reward . . . 119
6.4.3 Hydrogen Evolution Reaction: Composite Reward . . . 121
6.4.4 Oxygen Evolution Reaction: Composite Reward . . . 122
6.4.5 Ablation study . . . 128
6.5 Future Work . . . 130
6.6 Conclusion . . . 131

7 Conclusion **135**

References **139**

List of Figures

2.1 Overview of the structure of the UPINN method as applied to (2.3), which shows inputs and outputs of all known and unknown components, as well as losses. U is the neural network that provides the surrogate solution $\hat{U}$, and takes time t as input. Both $\mathcal{N}_K$ (the known component of the differential equation) and $\mathcal{F}$ (the unknown component, to be fit by the UPINN) take in time and $\hat{U}$, the prediction of U, as input. $\mathcal{N}_K$ and $\mathcal{F}$, along with U_t (the autodifferentiated derivative of U w.r.t. time) and are passed as input to the ODE loss. Then, the ODE loss computes the error between U_t and $\mathcal{N}_K + \mathcal{F}$. The MSE loss computes the error between the surrogate solution ($\hat{U}$) and the data. 20

2.2 Sparse data regime 21

2.3 Noisy data regime 21

2.4 UDE method performance on the Lotka-Volterra model. Noisy data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term. 22

2.5 UDE method performance on the Lotka-Volterra model. Sparse data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term. 23

2.6 UPINN performance on the Lotka-Volterra model. Noisy data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term. 24

2.7 UPINN performance on the Lotka-Volterra model. Sparse data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term. 25

2.8 The reconstructed solution of Burgers' equation. The two vertical dashed white lines indicate the noisy experimental data that were sampled for the algorithm. 26

2.9 Learning the v_1 term using UPINNs. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term. 27

2.10 Learning the v_2 term using UPINNs. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term. 28

3.1 Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). Data was collected with a fixed time period (0.15 days) elapsing between each data point. The drug actions are the functions $G(N)$ in Eq (3.2), replaced by one of: log-kill, Norton-Simon and E_{max}. 43

3.2 Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data is noiseless but collected such that there is one datapoint for each 0.05-interval decrease in N. Although there are half as many points collected than in the equispaced case, the MSE between the true drug action and the predicted drug action is still on the order of 10^{-4} as shown by Table 3.1 45

3.3 Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data has a noise level of 0.03 added proportionally to the mean of the variables. 46

3.4 Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data is spaced so that times of high cell decline have proportionally more observations (one datapoint for each 0.05-interval decrease in N). It also has a noise level of 0.03 added proportionally to the mean of the variables. 47

3.5 (a) Equispaced noiseless data with Norton-Simon drug action, (b) Learned drug action after the parameter β was fit from the first 12 days of noiseless data (final estimate: 0.999, MSE 1×10^{-6}),(c) Equispaced noisy (0.03) data with Norton-Simon drug action, (d) Learned drug action after the parameter β was fit from the first 12 days of noisy data (final estimate: 0.997, MSE 9×10^{-6}) 48

3.6 (a) Surrogate solution for row 4 of Table 3.3. (b) Fit and interpolated parameters per dosage. Blue is θ, and red is k_p. (c) Data was simulated using the inferred parameters for each of the three dosages. (d)-(f) Same as (a)-(c) but for row 7 of Table 3.3. 52

3.7 (a) Fit of $F_D(N,t)$ to data from eq 3.7 with parameters $k_{d,A} = 0.03, k_p = 0.0354$ (b) ODE solution generated using the inferred $F_D(N,t)$, along with the data for comparison. F_D is recovered well using the UPINN method, and the resulting model using the recovered F_D matches the data well. 54

3.8 **(a)** Fit of $F_D(N,t)$ (eq (3.9)) to data from row 2 of Table 3.5 (eq. 3.8, parameters $r = 0.017, k_{d,B} = 0.03$) **(b)** ODE solution generated using the inferred $F_D(N,t)$, along with the data for comparison **(c), (d)** Same as (a), (b) but $G_D(t)$ is fit. F_D in (a) is recovered with a solution MSE of 10^{-6}, but G_D in (c) appears to not be fully identifiable from the data. 59

3.9 **(a)** Fit of $F_D(N,t)$ (eq (3.9)) to experimental data: 312nM, 6h exposure **(b)** ODE solution generated using the inferred $F_D(N,t)$, along with the data for comparison **(c), (d)** Same as (a), (b) but $G_D(t)$ (eq (3.10)) is fit. The overall curve of (a) is similar in shape visually to Fig. 9 (a), indicating that the growth is captured to some extent by Model (3.6). Additionally, (c) shows a plateau in the function before an increase in cell killing, which cannot be captured by a linear combination of (3.7) and (3.8). 60

4.1 (a) The performance of a single PINN on non-noisy data. (b) Using the PINN and noiseless data from Fig 4.1(a), coverage estimated through repeated construction of an interval, for different calibration and validation sets. The running average coverage is computed over 10000 trials. 69

4.2 (a) The performance of a single PINN on noisy data (noise level 0.08). (b) Using the PINN and noisy data from Fig 4.2(a), coverage estimated through repeated construction of an interval and validation, for different calibration and validation sets. 70

4.3 (a) The performance of a single PINN in approximating the Buckley-Leverett equation (eq. 4.2). (b) Using the PINN and data from Fig 2a, coverage estimated through repeated construction of an interval and validation, for different calibration and validation sets. 71

4.4 (a) Plot of the inferred $\hat{\beta}$ vs. true β for 1000 datasets generated from sampled values of β (noiseless data) (b) Estimated coverage using the holdout set of $\{\beta, \hat{\beta}\}$ from Fig 4.4(a). 72

4.5 (a) Plot of the inferred vs. true β for 1000 datasets generated from sampled values of β (data noise level is 0.03) (b) Estimated coverage using the holdout set of $\{\beta, \hat{\beta}\}$ from Fig 4a. 73

4.6 (a) Plot of the inferred vs. true β for 100 different values of β, equispaced from 0 to 0.5. The data is noiseless. (b) Using the β estimates from noiseless data, coverage estimated through repeated construction of an interval, for different calibration and validation sets. The running average coverage is computed over 10000 trials. 74

4.7 (a) Plot of the inferred vs. true β for 100 different values of β, equispaced from 0 to 0.5. The data is noisy with noise level 0.08. (b) Using the β estimates and noisy data, coverage estimated through repeated construction of an interval, for different calibration and validation sets. The running average coverage is computed over 10000 trials. 74

5.1 Comparison of train and test MSE for each algebraic equation. 101

5.2 Solution and unknown term fit for the logistic equation, for one set of parameters. UPINNs do very well in recovering the unknown term in both cases of the original and the dimensionless equation. (a) and (b) are the original equation, whereas (c) and (d) are the fits for the dimensionless equation. 102

5.3 The recovered UPINN fit for both the original and dimensionless rotating bead DEs. (a) and (b) are the original equation, whereas (c) and (d) are the fits for the dimensionless equation. 103

6.1 Overview of the Catalyst GFlowNet framework. The leftmost section, Catalyst GFlowNet, samples the catalyst surface. The middle section determines atom positions, relaxes the structure, and converts to a graph. The rightmost section obtains the adsorption energy from a predictive model and the reward function to train the GFlowNet. 106

6.2 For the HER case study, the platinum samples generated by the GFlowNet relax to the minimum energy every time. The green line represents the total energy of the system. 120

6.3 Proportion of sampled structures (right) for the hydrogen evolution reaction case study. Low overpotential (left) is a predictor for efficient catalysts and aligns with the higher sampling rates. The values are detailed in Table 6.5 in the appendix. Pd(229) does not have experimental or DFT overpotentials, corresponding values for Pd(225) are displayed with shaded colors for reference. 121

6.4 A linear fit that maps log of exchange current density to experimental overpotential using data from [157] and [45]. These values are used column 3 (from the left) of Table 6.5. 123

6.5 Adsorption energy of samples from R_{comp} and $R_{uniform}$, within the OER search space. 126

6.6 Formation energy of samples from R_{comp} and $R_{uniform}$, within the OER search space. 127

6.7 Energy above hull of samples from R_{comp} and $R_{uniform}$, within the OER search space. 128

6.8 Force estimate of samples from R_{comp} and $R_{uniform}$, within the OER search space. 129

6.9 Ablation study with the HER search space. 130

List of Tables

2.1 MSE between F and the true hidden target after training for various values of n and n_P – noiseless data 14

2.2 MSE between F and the true hidden target after training for various values of n and n_P – noisy ($\epsilon = 5 \times 10^{-3}$) data 14

2.3 Coefficients (with MSE) recovered by AI Feynman from the approximations of F_1, comparing over datasets (rows) and method of finding F_1 (columns). The true coefficient is -0.9. 15

2.4 Coefficients (with MSE) recovered by AI Feynman from the approximations F_2, comparing over datasets (rows) and method of finding F_2 (columns). The true coefficient is 0.8 for F_2. 15

3.1 MSE of the drug action predictions (compared to the ground truth drug action) for different types of data. Equal spacing means that the data was equally spaced in time (one datapoint per 0.15 days). Adjusted spacing means that the data was collected with proportionally more datapoints when the slope of the decreasing cells is high in magnitude. When noise is added, the noise level is 0.03, added proportionally to the mean of the data. The lowest value in each column is bolded. 49

3.2 MSEs for each experiment (noiseless data). The row of θ and k_p indicates the values used for each dosage (low, medium, and high dosages respectively). Model MSE is the MSE between the data and the model. 50

3.3 MSEs for each experiment (noisy data). The row of θ and k_p indicates the values used for each dosage (low, medium, and high dosages respectively). The noisy data was created using a noise level of 0.03. Model MSE is the MSE between the data and the model. 51

3.4 Fit of $F_D(N,t)$ using data generated via eq. 3.7, with MSE computed between the inferred solution (using the learned hidden term) and the data. Each experiment was run 5 times; the error of the best fit is shown, along with the mean and standard deviation of all 5 runs. 53

3.5 $F_D(N,t)$ (eq 3.9) using data generated via eq. 3.8, with MSE computed between the inferred solution (using the learned hidden term) and the data. Each experiment was run 5 times; the error of the best fit is shown, along with the mean and standard deviation of all 5 runs. The first row of the table shows a significantly higher error compared to the rest because one of the five runs failed, which skewed the mean and standard deviation of the MSE. 55

3.6 Fits of $F_D(N,t)$ to in-vitro SUM-149 time-series data from [111]. 61

3.7 Fits of $F_D(N,t)$ to in-vitro SUM-149 time-series data from [111] (continued from Table 3.6). 62

5.1 Analysis of PySR-discovered algebraic equations with traditional and dimensionless forms (first 4 of 7 equations). 93

5.2 Analysis of PySR-discovered algebraic equations with traditional and dimensionless forms (last 3 of 7 equations). 94

5.3 Parameter values and MSE for original and nondimensionalized fits of the unknown term for the logistic growth equation. 95

5.4 Symbolic expressions returned by PySR for the logistic growth DE (original equation, data from 10 ODE solves). 95

5.5 Symbolic expressions returned by PySR for the extended logistic growth model (nondimensionalized, data from 10 ODE solves). *inv* represents the function $1/x$. The true discovered equation is bolded, and is algebraically equivalent to the true unknown term, $\alpha^2/(1+\alpha^2)$. 96

5.6 Symbolic expressions returned by PySR for the logistic growth DE (original equation, data from 50 ODE solves). 96

5.7 Parameter values and MSE for original and nondimensionalized fits for the rotating bead equation. Note that $g = 9.8$ and $w = 1.0$ are fixed across all runs and omitted from the table. 97

5.8 Symbolic expressions returned by PySR for the rotating bead DE (original equation, data from 10 ODE solves). 97

5.9 Symbolic expressions returned by PySR for the rotating bead problem (nondimensionalized, 10 ODE solves). The correct expression of the candidate expressions is bolded. 98

6.1 Templates used to set the atomic positions for HER and OER. The materials can be looked up on Materials Project based on their ID. 113

6.2 Hydrogen evolution reaction catalyst search space. 116

6.3 Oxygen evolution reaction catalyst search space. 117

6.4 Oxidation states of elements in OER. 118

6.5 Proportions of sampled structures for the case study: hydrogen evolution reaction. 122

6.6 Inferred space groups for the OER experiment with R_{comp} reward. 124

6.7 Proportions of sampled structures for the case study: oxygen evolution reaction. For the samples from the uniform reward run, we only list the compositions that were also sampled by the composite-reward run. Additionally, we only list those structures that were sampled at least 3 times by the composite-reward run. 125

6.8 Proportions of sampled structures for the case study: oxygen evolution reaction (continued). For completeness, here we list the remainder of the sampled structures, by either the uniform run and the composite-reward run. 133

6.9 Ablation study with the HER search space. 134

Chapter 1

Introduction: Physics-Informed Machine Learning

Machine learning (ML) has been instrumental in solving many complex tasks, such as natural language processing and question answering [162], image generation via diffusion models [43], and control problems and robot movement tasks using reinforcement learning [54]. ML applications in science have been developing rapidly, with ML being used widely in drug discovery [38, 81, 108], protein folding prediction [84], weather prediction [22], and other applications. Traditional ML paradigms such as supervised learning hinge on being able to reproduce and model data distributions. In essence, the model learns the relationships between various aspects of the data implicitly. This is a significant advantage of ML, since it allows ML to be used to solve tasks where the rules are too complex to write down, or unknown.

However, for scientific applications, (generative or discriminative) machine learning models that are trained on data only may lead to unrealistic or infeasible solutions, because the underlying physical rules of the data are not respected [69]. Additionally, many supervised ML models may not perform well on data outside of the training distribution [62] due to a difficulty generalizing from sparse and noisy data, which is common in scientific applications. For example, time series *in vitro* data of cell counts as they are affected by chemotherapy can be sparse and noisy due to the time-consuming data collection procedure [129]. In generative and reinforcement-learning settings, data-driven objectives can also lead to physically infeasible candidates if constraints are not enforced. This motivates the use of hard constraints and reward shaping in scientific design problems. Finally, in settings where differential equation constraints are available, traditional solvers and parameter fitting approaches may have difficulty handling missing parameters, incomplete boundary/initial

conditions, or sparse observations. However, pure machine-learning based tools for these problems could struggle with adhering to the differential equation constraints.

This thesis asks how physical constraints can be integrated into machine learning models to improve scientific inference and design when data are sparse, models are partially known, or the search space is large. As many constraints can be written as a system of equations or differential equations (DEs), Physics-informed Neural Networks (PINNs) [140], which leverage prior knowledge in the form of DE constraints, have been particularly influential. Although DEs have been traditionally solved with the finite element method (FEM), PINNs excel at handling ill-posed problems and parameter inference [154]. They have been applied extensively to many real-world problems to either solve differential equations (forward problem) [28, 29, 44, 86], or to infer their parameters (inverse problem) [48, 78, 86]. Methods such as the finite volume method and the FEM [6] are computationally expensive because they involve discretizing the solution, potentially in a high number of dimensions [4]. Unlike mesh-based solvers, PINNs approximate the solution with a neural network and enforce the governing equations through the loss function. This mesh-free formulation can be advantageous in inverse problems, irregular domains, high-dimensional settings, or cases with sparse observations, although PINNs can still suffer from optimization difficulties and may not outperform classical solvers when accurate discretization is available [67]. PINNs have been used to solve many PDEs with real-world applications, such as the model of forward wave propagation introduced by [4]. For inferring parameters, works such as [182] employed PINNs rather than conventional approaches to estimate diffusion parameters in a biomedical flow problem, because conventional approaches would require careful discretization of the brain geometry, and making assumptions on the boundary conditions of the problem. Due to their ability to leverage temporally sparse data, PINNs are able to match the diffusion parameter recovered from conventional FEM-based approaches well in this work [182]. Due to their flexibility, the architecture itself has been extended in various ways to suit different problems. For example, the A-PINN is an extension of the PINN to integro-differential equations [181] and X-PINNs which leverage domain decomposition for more accurate PDE solutions [77]. Bayesian PINNs integrate a Bayesian neural network [179] with PINNs in order to quantify uncertainty and reduce overfitting.

The original PINNs paper gave rise to a wide variety of follow-up work in physics-informed ML that extends PINNs in areas that they struggle in: handling model misspecification, improving PINN interpretability, solving entire classes of PDEs at once. Model discovery and model misspecification can be handled via Universal Differential Equations [138]; DeepONets [104] and physics-informed DeepONets [170] introduced operator learning and applied it in the context of PDE solutions; physics-informed Kolmogorov-Arnold Networks [99] were created for better interpretability of the DE solution. There are still

many open challenges in physics-informed ML, such as: improving PINN performance in stiff or chaotic systems; improving optimization and hyperparameter tuning to enable PINNs to become competitive with traditional PDE solvers [154]; improving the handling of model misspecification [187]; integrating uncertainty quantification frameworks into PINNs to better handle low-fidelity data [179, 187]. In this thesis, our contributions to PINNs aim to improve real-world applicability of PINNs in different ways such as: extending and applying PINNs to tackle model discovery under sparse and noisy data (Chapter 2 and 3) and integrating PINNs with conformal prediction for uncertainty quantification with measurable guarantees (Chapter 4).

While PINNs and UPINNs can learn unknown components of differential equations, the learned components are represented by neural networks. Symbolic regression provides a complementary step: converting these learned relationships into interpretable closed-form expressions. Symbolic regression is the problem of finding a closed-form model for a dataset, trading off simplicity and accuracy. Although symbolic regression is an NP-hard problem [164], the motivation for symbolic regression is to create an interpretable model for a dataset, overcoming the "black-box" interpretability issues of neural networks [66]. Recent techniques introduced for symbolic regression have taken the form of genetic programming [42, 90] and transformers [46, 159]. A number of works use physics-based rules to simplify the search [151, 158]. In Chapter 5, we integrate dimensional analysis and the Buckingham Π theorem [27] with symbolic regression in order to improve its efficiency. Additionally, we apply the method under a partial knowledge setting where some of the DE constraint is known, but some must be learned.

In contrast to Chapters 2–5, which focus on inference and model discovery from sparse data, Chapter 6 addresses inverse design in a large combinatorial search space. Here, the role of physics is not to constrain a DE, but to restrict actions, shape rewards, and filter physically meaningful candidates. Within reinforcement learning (RL) settings, physics information can be enforced to yield more realistic and robust agent policies. Reward-shaping can enforce soft constraints, and action space constraints can be used to enforce hard constraints and ensure safety [13]. Works such as [53] enforce constraints within a graph neural network via frame averaging; this is in order to predict material properties that are equivariant/invariant to 3D transformations [53]. Generative Flow Networks (GFlowNets, GFN) [19] are a novel method closely related to reinforcement learning that is well-suited to scientific exploration: it is equivalent to entropy-regularized RL [47], which allows it to sample proportionally to a reward function rather than simply maximizing it. This feature of the approach is well-suited to scientific discovery, where discovered objects (e.g. materials or drugs) need to undergo downstream experimental testing, so optimizing for a diversity of potential candidates may yield a successful drug or material faster. In Chapter

6, we create a GFlowNet pipeline to sample effective catalyst materials for hydrogen energy storage. With our method, we show that we can successfully rediscover the best known catalysts for the two reactions in the water electrolyzer.

In this thesis, we make three main contributions to physics-informed ML:

1. We develop and apply several extensions of PINNs. We introduce Universal PINNs for learning unknown terms in differential equations from sparse and noisy data (Chapter 2), and apply the method to chemotherapy modelling (Chapter 3). We also integrate PINNs with uncertainty quantification (Chapter 4).

2. We improve symbolic model discovery by integrating dimensional analysis with symbolic regression and UPINNs (Chapter 5).

3. We develop a constrained GFlowNet framework for catalyst discovery, incorporating physical constraints at the level of actions and rewards (Chapter 6).

In the remainder of this thesis, each chapter is a separate project. The necessary background for each project is included in the *Background* section of each project. Additionally, in this thesis, we use the term "incorporating physics into a model" to mean both incorporating physical constraints that yield a realistic output, but we also mean the integration of broader constraints into the model. The constraints (whether soft or hard) should in general be known *a priori* but may not be trivial to evaluate. The constraints that we work with include: DE constraints, invariance/equivariance, requirements on material stability and predicted electrocatalyst performance, requirements on uncertainty quantification, and others.

1.1 Thesis summary

Chapters 2-5 focus primarily on PINNs and extensions to them to enhance their effectiveness on real-world problems (e.g. models of chemotherapy response, species populations, and programmed cell death). In Chapter 6, the application domain shifts to catalyst discovery with generative flow networks. This work features multiple complex and interacting constraints (on material stability, material complexity, overall charge, and predicted efficiency of the catalyst). In both PINNs and GFlowNets, constraint specification and enforcement is of paramount importance to resolving both the predictive (forward) or inverse design problems.

Chapter 2

Universal Physics-Informed Neural Networks (UPINNs) [128] extend the paradigm of PINNs to learn unknown components of differential equations in sparse data. The work in this chapter was published in *ICML 2023 Main Conference*. UPINNs outperform a similar method, Universal Differential Equations (UDEs) [138] on sparse and noisy data, and integration with symbolic regression allows this method to discover the underlying full form of differential equations from data.

Chapter 3

We applied UPINNs to both simulated and *in vitro* data from [111] to learn the underlying drug action of chemotherapy. The work in this chapter was published in *Pharmaceutical Research* [129], and reproduced with permission from Springer Nature. We showed that UPINNs can fit drug-action terms not captured by the originally proposed ODE model, and that this might shed light into the biological mechanism of the drug action in future work. Additionally, we showed that the method can be used to infer multiple sets of parameters simultaneously, for which purpose separate PINNs would need to be employed otherwise.

Chapter 4

In this chapter, we integrated conformal prediction with PINNs in order to yield finite-sample coverage guarantees for uncertainty intervals of their predictions, for both the forward problem and the inverse problem. The work in this chapter was published in *ICLR 2024, Workshop on AI for Differential Equations in Science* [131]. We detail the methods we used to do this, and show that the guarantees hold. This is a step forward for the scientific machine learning community in terms of uncertainty quantification for PINNs, which may have weaker accuracy guarantees compared to traditional numerical solvers [154].

Chapter 5

In this chapter, we examined the increase in performance that symbolic regression can gain when integrated with dimensionality reduction through either the Buckingham Π [27] theorem or Ipsen's method [59]. This work is under submission. We find that Ipsen's method is particularly suitable for integration with UPINNs, and can be used to reduce the number of input variables to the unknown inferred differential equation term. These novel combinations of techniques enhance the efficiency of both symbolic regression and UPINNs due to the reduction in the number of variables.

Chapter 6

In Chapter 6, the focus shifts towards catalyst discovery via generative flow networks. In this project, we use GFlowNets to explore and sample within the space of possible crystal

surfaces that can act as catalysts. This chapter is based on work published at *NeurIPS 2025, AI for Materials Workshop* [130]. We successfully rediscover the best known catalysts for the hydrogen evolution reaction and the oxygen evolution reaction, which are both used for hydrogen energy storage. We add in various physics-informed constraints on the level of the action space, reward functions, and the level of the pipeline architecture.

Chapter 2

Universal physics-informed neural networks: symbolic differential operator discovery with sparse data

Chapter Summary

In this work we perform symbolic discovery of differential operators in a situation where there is sparse experimental data. This small data regime in machine learning can be made tractable by providing our algorithms with prior information about the underlying dynamics. PINNs have been very successful in this regime (reconstructing entire ODE solutions using only a single point or entire PDE solutions with very few measurements of the initial condition). The Universal PINN approach (UPINN) adds a neural network that learns a representation of unknown hidden terms in the differential equation. The algorithm yields both a surrogate solution to the differential equation and a black-box representation of the hidden terms. These hidden term neural networks can then be converted into symbolic equations using symbolic regression techniques like AI Feynman. In order to achieve convergence of the neural networks, we provide our algorithms with (noisy) measurements of both the initial condition as well as (synthetic) experimental data obtained at later times. We demonstrate strong performance of UPINNs even when provided with very few measurements of noisy data in both the ODE and PDE regime.

2.1 Introduction

ML algorithms, for instance neural networks (NN), are particularly helpful in representing unknown quantities in a data-driven way [17]. NNs with a wide enough hidden layer can be used to approximate any function [125] by tuning its parameters (henceforth 'NN parameters'). Hence, NNs have been used to infer DE parameters or even entire DE models, due to their ability to approximate functions. Many recent applications use NNs augmented with prior knowledge in order to learn underlying DE models[1] from data [33, 102, 103, 112, 138, 140]. However, acquiring sufficient data to fit these values accurately using NNs is difficult. A method that can function in low-data regimes by leveraging the known structure of the model is needed. However, a drawback of this method is that NN outputs are often not meaningfully interpretable. Furthermore, the relation learned by the NN may be better described by a simpler mathematical expression. Genetic algorithms [32] avoid this issue by optimizing over mechanistic models directly, but they are computationally expensive [153]. Moreover, it is possible to simplify NN outputs via symbolic regression algorithms [42, 158, 159], which search for the mathematical model that best fits a dataset.

Two prominent NN-based methods that learn DE models from data are PINNs [140] and universal differential equations (UDE) [138]. In UDEs, each unknown component of the DE model is approximated by a NN and a hard DE constraint is employed. That is, the best-fit DE is satisfied at all times during training. However, UDEs are not robust to noise, require a lot of data, and SINDy, as employed in [138], does not succeed in finding the true mechanistic model reliably. PINNs assume the form of the true DE and fits its parameters via a soft constraint (relaxing the requirement that the NN should satisfy the best-fit DE exactly), which is added to the NN loss function as an additional loss term referred to as the 'PINN loss'. A drawback of PINNs is that the structure of the DE model must be determined in advance, and there is no way to learn its unknown components using the method as originally proposed. Additionally, as iterative optimization is computationally expensive, PINN loss can fail on stiff DEs [169].

Our approach, Universal PINNs (UPINNs), bridges the limitations of both PINNs (cannot be used when the structure of the DE is not fully known) [140] and UDEs (not robust to noise and requires lots of data) [138]. To address this, we replace the hard constraint of the UDE with that of PINN loss, which allows the approach to learn unknown components of the DE model from data. This approach is robust to noise and performs well in low-data regimes. Additionally, using the AI Feynman algorithm [158] yields good

[1]The (underlying or true) DE model of a process refers to the DEs that produced the experimental data.

results in identifying the underlying hidden terms of the DE model.

In this chapter, we claim the following contributions:

1. We propose a novel training methodology, Universal PINN, which combines PINN loss with the UDE framework to allow a PINN-based approach to learn unknown parts of the DE model from data
2. Our method is robust to noise and learns the unknown DE model components to significantly higher accuracy in the Lotka-Volterra model compared to the state of the art (UDE approach)
3. UPINNs perform very well in systems biology ODEs, and the Viscous Burgers' equation partial differential equation
4. Using a symbolic regression algorithm, we reached better identification of the tested systems than in the original UDE paper

2.2 Background

Since our method, UPINNs, is based on the PINN paradigm, we will give a technical overview of PINNs [140]. Suppose that the following DE governs a physical process. $u(t, x)$ is an unknown real-valued function of time (t) and position (x). Its time derivative is related to its value for each tuple (t, x) with a known function $\mathcal{N}$, and unknown vector of parameters θ. Furthermore, there are N potentially noisy DE measurements $\{t_i, x_i, u_i\}$.

$$\frac{\partial u(t,x)}{\partial t} = \mathcal{N}[u;\theta], x \in \Omega, \Omega \in R^D, t \in [0,T]. \tag{2.1}$$

Although θ is required in order to find a numerical or analytical function u that satisfies 2.1, θ is unknown in this setup. Using the given data, the PINN method from [140] can estimate θ and u simultaneously. Its key component is a neural network U, which predicts u given any tuple (t, x). The following loss function is used to train U:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} |U(t^i, x^i) - u^i| + \frac{1}{M}\sum_{j=1}^{M} \left| \frac{dU}{dt_j} - \mathcal{N}[u;\theta] \right|, \tag{2.2}$$

where $\frac{dU}{dt_j}$ is auto-differentiated through the neural network and evaluated at time t_j and position x_j. The first term penalizes U for making predictions that do not match the DE at a predefined set of M collocation points $\{t_j, x_j\}$. The second term penalizes U for making predictions that do not match the data. Note that the second term contains θ, which allows parameter estimate $\hat{\theta}$ to be updated using its gradient with respect to $\mathcal{L}$. At every optimization iteration, the parameters of U are updated along with $\hat{\theta}$.

Finally, since we aim to connect the hidden terms learned by UPINNs with symbolic regression, we will introduce a short overview of symbolic regression. Symbolic regression is a general technique of finding a model that fits data while balancing the simplicity of the model with its accuracy. This problem has been solved with various genetic algorithms (see, among others, [110, 144]) but since this method is computationally expensive, newer techniques are becoming more popular. For example, AI Feynman [158] leverages NNs and symmetry, units, compositionality, etc. and finally returns a list of potential models ranked by error and complexity. Some work [85, 159] makes use of transformers to find the correct functional form. In our work, we only use them at the final stage after our method has learned an approximate representation of the missing components.

2.3 Methods

Our proposed method, UPINNs, is a modification of the PINNs to discover the functional form of an unknown term within a differential equation. Suppose $\vec{u}(\vec{x}, t) \in \mathbb{R}^m$ for $\vec{x} \in \mathbb{R}^d$. Let $\mathcal{N}$ be a (potentially non-linear) differential operator, then consider time t in the domain $[0, T] \subset \mathbb{R}$ along with a d dimensional, bounded spatial domain $\Omega \subset \mathbb{R}^d$ where $\partial\Omega$ denotes the boundary of Ω. Notably, if $\mathcal{N}$ contains any derivatives, we assume that those derivatives are with respect to the spatial variables only. We then consider problems (partial and ordinary differential equations) of the form

$$\frac{\partial}{\partial t}\vec{u}(\vec{x}, t) = \mathcal{N}[\vec{u}](\vec{x}, t), \quad t \in [0, T], \quad \vec{x} \in \Omega,$$

subject to initial condition

$$\vec{u}(\vec{x}, 0) = \vec{u}_0(\vec{x}), \quad \vec{x} \in \Omega,$$

and boundary conditions

$$\beta[\vec{u}](\vec{x}, t) = 0, \quad \vec{x} \in \partial\Omega, \quad t \in [0, T],$$

where β is a (potentially non-linear) differential operator whose derivatives are only with respect to the spatial variables.

Further, suppose

$$\mathcal{N}[\vec{u}](\vec{x},t) = \mathcal{N}_K[\vec{u}](\vec{x},t) + \mathcal{F}[\vec{u}](\vec{x},t),$$

where $\mathcal{N}_K$ is some differential operator with known functional form and $\mathcal{F}$ represents some unknown, target differential operator. Similarly, suppose $\beta = \beta_K + \mathcal{B}$ for some known β_K and some unknown $\mathcal{B}$.

Finally, one can consider $\Omega = \varnothing$, in which case the underlying differential law is governed by an ordinary differential equation (ODE). In this situation, there is no boundary condition and so no need for β (or, equivalently, β is the empty function). The differential equation constraint simplifies to the following, and is visualized in Fig. 2.1:

$$\frac{d}{dt}\vec{u}(t) = \mathcal{N}_K[\vec{u}](t) + \mathcal{F}[\vec{u}](t). \tag{2.3}$$

Returning to the general (PDE) case, suppose we have n data points $D = \{(t_i, \vec{x}_i, \vec{u}_i)\}_{i=0}^{n-1}$ where $\vec{u}_i = \vec{u}(t_i, \vec{x}_i) + \epsilon_i$ where ϵ_i is some noise term (potentially $\epsilon_i = 0$). We will use this measured data to fit the parameters of (up to) three neural networks. The first network, $F(\vec{u}; \theta_F)$, will approximate the target differential operator $\mathcal{F}[\vec{u}]$ by using a neural network with parameters θ_F. The second network, $U(\vec{x}, t; \theta_U)$, will approximate the value of $\vec{u}(x,t)$ by a neural network with parameters θ_U. The third network, $B(\vec{u}; \theta_B)$, will approximate the value of $\mathcal{B}[\vec{u}]$, the unknown target for the boundary condition, with a neural network parameterized by θ_B. To fit these networks, we consider another two sets of collocation points: these sets are $X_P = \{(\vec{x}_j, t_j)\}_{j=0}^{n_P-1} \subset (\Omega \setminus \partial\Omega) \times (0, T]$ and $X_B = \{(\vec{x}_k, t_k)\}_{k=0}^{n_B-1} \subset (\partial\Omega) \times (0, T]$. These sets correspond to locations in the space-time domain where we enforce that our network U satisfies the underlying differential equation (in the case of X_P) and the boundary conditions (in the case of X_B).

To calculate the gradients for fitting these networks, we consider the loss function

$$L(\theta_U, \theta_B, \theta_F) = L_M(\theta_U) + L_B(\theta_U, \theta_B) + L_P(\theta_U, \theta_F).$$

The first component of the loss is the Mean Squared Error (MSE) loss. This loss is the difference in MSE between the measurement value of $\vec{u} \approx \vec{u}_i$ from the input data with the neural network approximation of $\vec{u} \approx U(\vec{x}_i, t_i)$, evaluated at the same space-time location and is given by

$$L_M(\theta_U) = \frac{1}{n} \sum_{(\vec{x}_i, t_i, \vec{u}_i) \in D} (U(\vec{x}_i, t_i; \theta_U) - \vec{u}_i)^2.$$

The second component of the loss is the boundary loss. This loss is the mean squared value of the approximated value of the boundary condition and is given by

$$L_B(\theta_U, \theta_B) = \frac{1}{n_B} \sum_{(\vec{x}_k, t_k) \in X_B} (\beta_K[U](\vec{x}_k, t_k; \theta_U) + B(U(\vec{x}_k, t_k; \theta_U); \theta_B))^2.$$

The final component of the loss is the PINN loss. This loss is the MSE between the value U_t, the time derivative of the neural network approximation of U, and the value $\mathcal{N}_K[U] + F(U)$.

$$L_P(\theta_U, \theta_F) = \frac{1}{n_P} \sum_{(\vec{x}_j, t_j) \in X_P} (\mathcal{N}_\mathcal{K}[U](\vec{x}_j, t_j; \theta_U) + F(U(\vec{x}_j, t_j; \theta_U); \theta_F) - U_t(\vec{x}_j, t_j; \theta_U))^2.$$

This loss function is quite similar to the loss function for PINNs given in [140], however here we insert two additional neural networks into the loss function corresponding to the unknown parts of the underlying dynamics in the boundary conditions and the differential equation. To compensate for these additional parameters, we extend the first component of the loss to include more than just initial data (but solution data as well). In this way, D could contain data from the initial condition, data from the boundary, or data from the interior of the domain.

Practically, one way to select X_P is to simply choose n_P and use Latin hypercube sampling to select n_P points in $(\Omega \setminus \partial\Omega) \times (0, T]$. A similar construction works for selecting X_B. In this way, we are sampling the domain in a space-filling manner.

The architecture of the fully-connected neural networks is as follows, for each of our models examined in Section 2.4:

1. **Burgers**: two inputs for t and x followed by scaling layer; 8 hidden layers of 20 units for the surrogate network and the hidden component network; sigmoid activation

2. **Lotka-Volterra** and **Apoptosis model**: one input for t followed by a scaling layer; 2 hidden layers of 64 units for the surrogate solution; 2 hidden layers of 16 units for the hidden component approximation; sigmoid activation

2.4 Results

To demonstrate our approach, we show high accuracy in identifying the hidden terms in three test-cases: the Lotka-Volterra equations (an ODE model), and the viscous Burgers' equation (a parabolic PDE model), and a model for cell apoptosis (an ODE model). Additionally, we show that AI Feynman is able to correctly identify the functional form of hidden terms within the Lotka-Volterra model from the output of our model.

2.4.1 Lotka-Volterra System

We begin our analysis by testing our method on the Lotka-Volterra (LV) model [21] of predator-prey interactions. The DE is formulated as follows:

$$\begin{aligned}\frac{dx}{dt} &= \alpha x - \beta xy \ , \\ \frac{dy}{dt} &= -\delta y + \gamma xy \ .\end{aligned}$$

We take the known portion of the differential equation as $\mathcal{N}_{\mathcal{K}}[U] = [\alpha\, x, -\delta\, y]$ for known parameters α and δ, and seek to learn $F = [F_1, F_2] \approx [-\beta\, x\, y, \gamma\, x\, y]$ from data only, without knowing the target form and without knowing β and γ.

To generate the synthetic data, $\alpha, \beta, \gamma, \delta$ were fixed at $(1.3, 0.9, 0.8, 1.8)$ respectively, with initial conditions at $(x_0, y_0) = (0.44249296, 4.6280594)$ just as in [138]. The time interval was chosen as $[0, 3]$ and stayed the same throughout every LV experiment.

An ODE solver was used to generate data satisfying the LV equations. This yields a set of points $\{t_i, x_i, y_i\}$. Then, Gaussian noise is added to each x_i and y_i. Given a particular noise level ϵ, Gaussian noise was added to the data as follows:

$$\begin{aligned}(x_i)_{noise} &= x_i + \epsilon \cdot \bar{x} \cdot N(0,1) \ , \\ (y_i)_{noise} &= y_i + \epsilon \cdot \bar{y} \cdot N(0,1) \ ,\end{aligned}$$

where $\bar{x}$ denotes the element-wise mean of x_i over all i (similarly for y).

First, we demonstrate our approach on noise-free data (Table 2.1) and data with $\epsilon = 5 \times 10^{-3}$ noise (Table 2.2) for various values of n (number of data points) and n_P (number of collocation points). We want to show how the hard-to-acquire data can be augmented by taking more collocation points which require no experiments/measurements and come at only the cost of increased computing power. We see that, in contrast to a standard PINN approach, we need to provide more data than just the initial condition. However, even with very sparse measurement data, we can acquire a good discovery by only increasing the number of collocation points. The additional benefit gained from increasing the collocation points is only realized when there is already ample enough experimental data for the algorithm to leverage.

Next, we compare UPINN performance to the UDE method. We test the two methods on noiseless sparse data (2.2) and on noisy data (Fig 2.3). The error is computed as the MSE taken with respect to the true interaction. At minimal noise level, the UDE approach

Table 2.1: MSE between F and the true hidden target after training for various values of n and n_P – noiseless data

n \ n_P	10^2	10^3	10^4
1	2×10^{1}	2×10^{1}	2×10^{1}
5	9×10^{-4}	1×10^{-3}	9×10^{-4}
10	2×10^{-4}	4×10^{-5}	5×10^{-6}

Table 2.2: MSE between F and the true hidden target after training for various values of n and n_P – noisy ($\epsilon = 5 \times 10^{-3}$) data

n \ n_P	10^2	10^3	10^4
1	2×10^{1}	2×10^{1}	2×10^{1}
5	6×10^{-2}	4×10^{-3}	5×10^{-3}
10	1×10^{-3}	6×10^{-4}	8×10^{-4}

and UPINN approach perform similarly and for the densest data UDEs slightly outperform UPINNs. Although increasing either noise or sparsity degrades the performance of both methods, the UPINN method consistently attains a lower MSE compared to the UDE method as noise or sparsity increases.

Figures 2.4, 2.5, 2.6, and 2.7 show the surrogate solution and hidden terms as recovered by the UDE and UPINN methods. The noise level of the noisy data was set at 0.1 and, for the noiseless sparse data, there were 5 points each 0.6 units apart. It is clear that UPINNs are quite robust to noise and perform well in low-data regimes. The UDE approach performs reasonably on sparse data, but is not robust to noise.

Finally, AI Feynman symbolic regression is run on the neural network output from both our approach and the UDE approach, in order to find the best functional form. These results are presented in Table 2.3. A dash indicates AI Feynman did not recover the functional form Cxy. The best performance between the two methods is bolded. In cases of both sparse and noisy data, AI Feynman correctly recovers the hidden interaction terms more often for our method than it does for the UDE method. If a formula is recovered for both methods, the one recovered for the PINN method is often more accurate.

The terms $\gamma\, x\, y$ and $-\beta\, x\, y$ in the LV equations correspond to the predator's uptake

Spacing	Noise Level	F_1 (UDE)	F_1 (UPINN)
0.1	0	**-0.901 (2.8e-7)**	–
0.2	0	–	–
0.3	0	–	**-0.897 (4e-6)**
0.4	0	–	**-0.888 (8.2e-5)**
0.5	0	–	**-0.889 (8.9e-5)**
0.6	0	-0.892 (4e-3)	**-0.890 (1e-5)**
0.1	8e-3	-9.25 (1.8e-3)	**-0.906 (1e-5)**
0.1	1e-2	–	**-0.911 (3.45e-5)**
0.1	3e-2	–	**-0.960 (1e-3)**
0.1	5e-2	–	–
0.1	8e-2	–	–
0.1	1e-1	–	–

Table 2.3: Coefficients (with MSE) recovered by AI Feynman from the approximations of F_1, comparing over datasets (rows) and method of finding F_1 (columns). The true coefficient is -0.9.

Spacing	Noise Level	F_2 (UDE)	F_2 (UPINN)
0.1	0	**0.802 (1.1e-6)**	0.797 (2.5e-6)
0.2	0	0.797 (3.4e-6)	**0.799 (3.8e-7)**
0.3	0	–	**0.798 (1.9e-6)**
0.4	0	–	**0.797 (5.2e-6)**
0.5	0	**0.760 (1e-3)**	–
0.6	0	–	**0.800 (1e-32)**
0.1	8e-3	–	**0.798 (3e-5)**
0.1	1e-2	**0.791 (2.3e-5)**	0.777 (1.5e-4)
0.1	3e-2	–	**0.777 (1.5e-4)**
0.1	5e-2	–	**0.740 (1.1e-3)**
0.1	8e-2	–	–
0.1	1e-1	**0.887 (2e-3)**	–

Table 2.4: Coefficients (with MSE) recovered by AI Feynman from the approximations F_2, comparing over datasets (rows) and method of finding F_2 (columns). The true coefficient is 0.8 for F_2.

function in the ecological model. This represents the predators' feeding habits as a function of prey population and its resulting effect on both the prey population and the predator's population. The actual form of these functions can take various forms in predator-prey

models (see, for instance, [24, 70]). While we initially modelled this as two unknown, decoupled functions F_1 and F_2 and learned them independently, we could also have modeled them by a single function with an additional learned parameter as a scaling factor. That is, we could assume that $F_1 = -\phi\, F_2$ and then only explicitly learn F_2 and a single parameter ϕ. This results in regressions that are near identical to the ones presented above, but showcases an important modelling methodology that our method is amenable to and, for more complicated models than LV, may be necessary in order to achieve a high-quality regression.

2.4.2 Viscous Burger's Equation

Finally, our method is easily applied to PDEs (as in the original PINN implementation). Here we present the discovery of both the solution to the PDE where the underlying hidden dynamics of the operator were partially hidden. This reconstruction used only noisy ($\epsilon = 5 \times 10^{-3}$) data obtained from two time points (the initial condition, $t = 0$, and a later time at $t = 0.5$). While this method can be used to discover the form of the boundary condition as well, here we assume that the homogeneous Dirichlet boundary conditions are known. The PDE in question is

$$\frac{\partial u}{\partial t} = -u\,\frac{\partial u}{\partial x} + \nu\,\frac{\partial^2 u}{\partial x^2},\ \nu = \frac{1}{1000\,\pi},\ u(x,0) = -\sin(\pi\, x)\ .$$

Here we took $\mathcal{N}_{\mathcal{K}} = \nu\, u_{xx}$ and let the algorithm learn the hidden term $-u\, u_x$. To do this, we gave the F network u, u_x, and u_t as inputs. This represents an inductive prior where we are assuming that the hidden term depends on first order and lower derivatives of the solution. In our approach, such a prior is necessary (that is, the algorithm cannot learn what order of derivatives to include or not include, it can merely choose which inputs presented to it to utilize). For collocation data we used $n_P = 10^4$ and $n_B = 10^2$ points sampled from the appropriate parts of the domain $[-1, 1] \times [0, 1]$ via Latin hypercube sampling. The PDE solution was reconstructed with MSE of 3×10^{-4} and the hidden term was discovered with MSE of 2×10^{-2}. The resulting solution is visualized in Figure 2.8.

2.4.3 Cell Apoptosis Model

We also test the method on a biological application, which is the Q1 cell apoptosis model from [173]. This is an ODE with three variables, serine-threonine kinase Akt_s (active Akt), Akt (inactive Akt) and tumour suppressor protein $p53$. p53 promotes cell apoptosis, or

programmed cell death, and Akt inhibits it. Here, we only focus on the system of ODEs and learning its nonlinear terms. We refer readers to the paper for the biological motivation and discussion. We denote the concentrations of p53, active Akt and inactive Akt as x, y, z respectively. The equations are given by

$$\begin{aligned}\frac{dx}{dt} &= v_0 - v_2 - k_d \cdot x,\\ \frac{dy}{dt} &= v_1 - v_{m1} - v_{m3},\\ \frac{dz}{dt} &= \frac{-dy}{dt},\end{aligned}$$

where

$$\begin{aligned}v_0 &= k_0,\\ v_1 &= k_1 \cdot z \cdot (j_1 + y),\\ v_{m1} &= \frac{k_{m1} \cdot y}{j_{m1} + y},\\ v_2 &= \frac{k_2 \cdot y \cdot x}{j_2 + x},\\ v_{m3} &= \frac{k_{m3} \cdot x \cdot y}{j_{m3} + y},\end{aligned}$$

All parameter values were taken from the paper. With the initial condition $(x, y, z) = (0.248, 0.0973, 0.0027)$ and 30 noiseless datapoints, both the v_1 and v_2 interactions (including the parameters) were learned to a high degree of accuracy. In Figures 2.9 and 2.10 it can be seen that although the general shape does not match the true interaction 100%, the MSE between the true interaction and the learned function is in fact very small (on the order of 10^{-4}) and the surrogate solution fits the data very well. This case study reveals a key trait of the method – in some DE's, the hidden interaction is not unique given a particular trajectory and data. Furthermore, when the derivatives of the trajectory are very small (as can be seen by the saturation past $t = 100$) the method can have difficulty learning the hidden term. As is done in [180], if this method were augmented to handle very small and very large values through scaling, more accurate learning of the interaction would be possible.

2.5 Discussion and Conclusion

In conclusion, the Universal PINN approach is able to recover, with a great degree of accuracy, the symbolic functional form of hidden terms within a differential operator using very sparse measurements of noisy data. This approach is robust to both noise and sparsity of the data by increasing the number of collocation points (an operation that doesn't require any additional experimentation, just stronger compute capacities). This approach can be applied to discovering the functional form of an unknown ODE as well as both the functional form of a partial differential operator in a PDE and unknown terms in the boundary condition of a PDE. However, we do have to assume that the ODE and PDEs in question have unique solutions. In all of our experiments in this chapter (and in the subsequent QSP application chapter), this is the case.

The question of why UPINNs perform better than UDEs on sparse and noisy datasets is worth investigating in follow-up work. One hypothesis is that the neural network which approximates the solution within the PINN paradigm regularizes the solution and prevents it from overfitting to noise. Notably, MLPs have been noted to perform implicit regularization [132, 142]. We postulate that in the absence of additional regularization that enforces smoothness of the DE solution, UDEs will tend to overfit to noise, whereas MLPs would not; this potential overfitting behaviour can be quantified within a simple (potentially linear system) and investigated further in future work. Another possible difference between UPINNs and UDEs concerns the optimization landscape. UPINNs have two soft constraints (PINN loss and MSE loss), whereas UDEs treat the DE constraint as exactly satisfied at all times, and thereby operate with the MSE loss only. This may lead to suboptimal solutions since the two soft constraints may create a more favorable loss landscape. This general advantage of UPINNs over UDEs could fail when the dataset is large and noiseless, since there are strictly fewer parameters to optimize for UDEs compared to UPINNs. While we have provided some intuition and justification for why UPINNs perform better than UDEs for sparse and noisy data, a rigorous theoretical understanding of this behaviour is still lacking and remains an important direction for future research.

Regarding scalability of the UPINN approach, we claim that it is very similar to that of PINNs. In particular, the number of collocation points grows exponentially with the dimension of the problem (e.g. 1000 collocation points in a 1D problem, 1000^2 in a 2D problem, 1000^3 in a 3D problem, etc.) However, as a 1D PINN can be run on only a single GPU, and PINNs can be significantly advantageous to running the FEM in high dimensions, it would be an interesting problem to investigate exactly how computationally expensive UPINNs are in higher dimensions, and the applicability of mitigation strategies such as parallelization over multiple GPUs.

Although PINNs have been noted to perform sub-optimally on stiff equations without modification [82, 114], we have noted promising results in this direction. However, more investigation is needed. Finally, we note that in previous work by Karniadakis et al. [141], UPINNs were employed as a natural extension of PINNs. In [180], unobserved variables were inferred as part of the ODE solution using a deep NN. In this chapter, one of our main contributions is the analysis of UPINNs alongside UDEs, and their application to various noisy and sparse datasets.

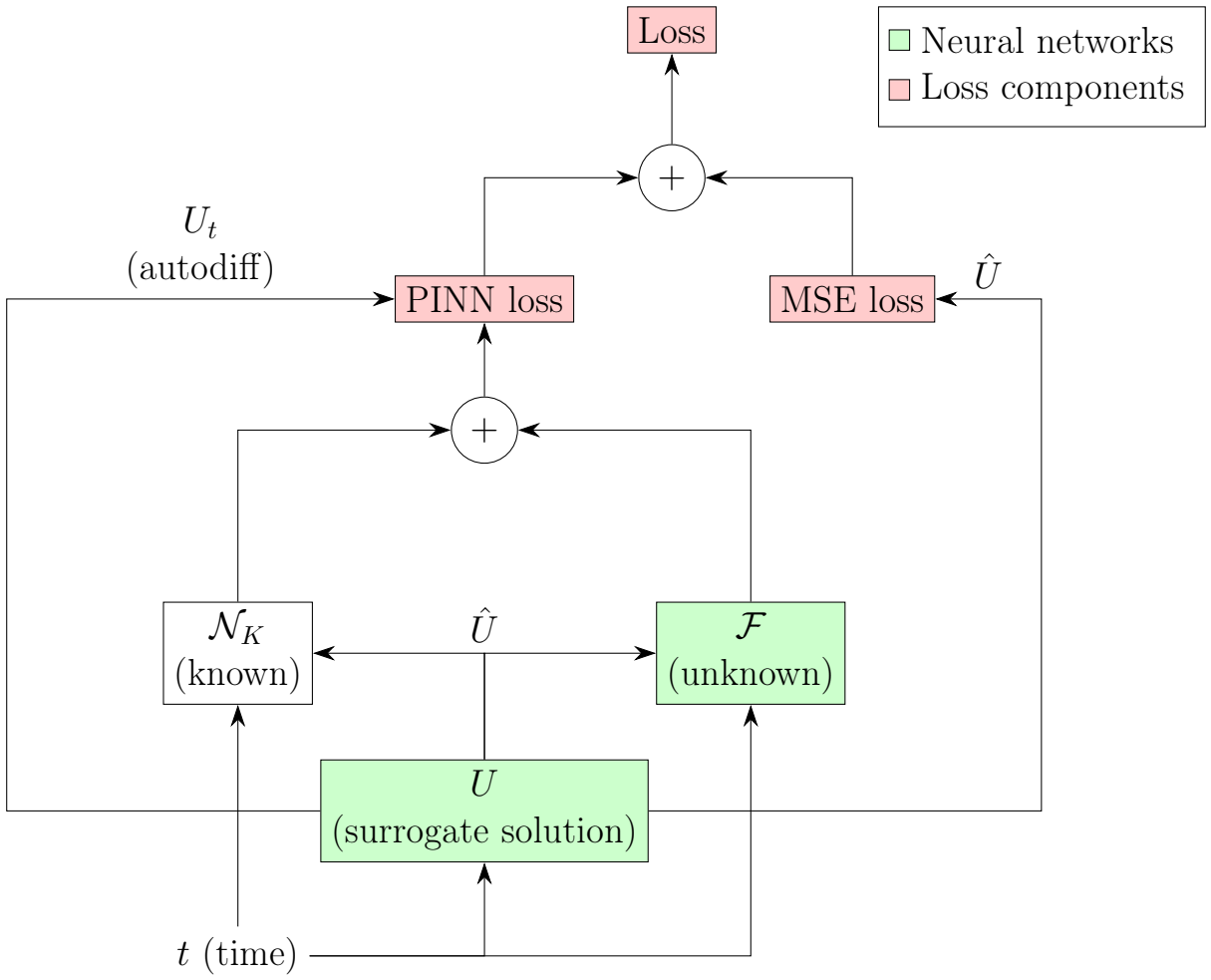


Figure 2.1: Overview of the structure of the UPINN method as applied to (2.3), which shows inputs and outputs of all known and unknown components, as well as losses. U is the neural network that provides the surrogate solution $\hat{U}$, and takes time t as input. Both $\mathcal{N}_K$ (the known component of the differential equation) and $\mathcal{F}$ (the unknown component, to be fit by the UPINN) take in time and $\hat{U}$, the prediction of U, as input. $\mathcal{N}_K$ and $\mathcal{F}$, along with U_t (the autodifferentiated derivative of U w.r.t. time) and are passed as input to the ODE loss. Then, the ODE loss computes the error between U_t and $\mathcal{N}_K + \mathcal{F}$. The MSE loss computes the error between the surrogate solution ($\hat{U}$) and the data.

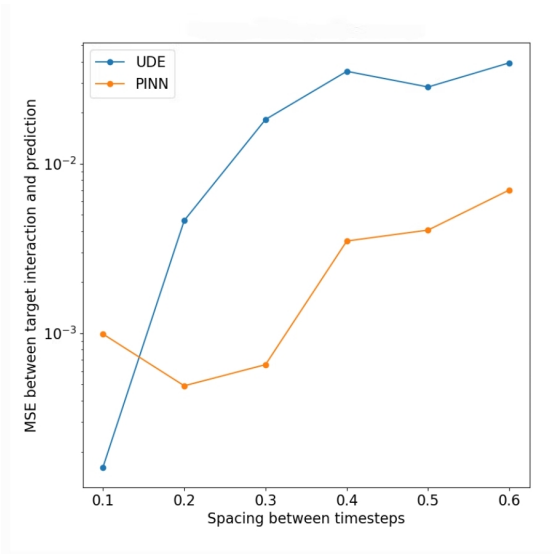


Figure 2.2: Sparse data regime

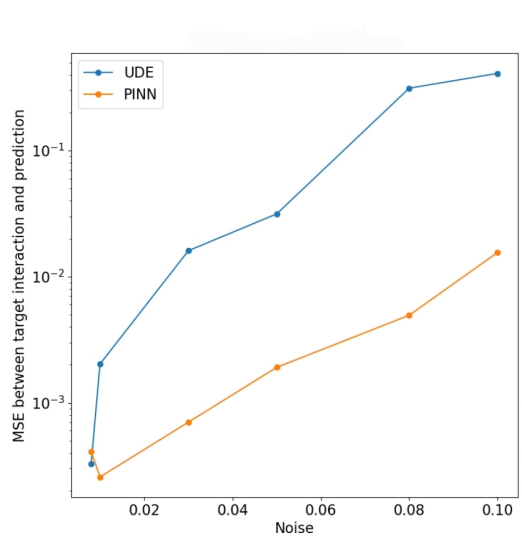


Figure 2.3: Noisy data regime

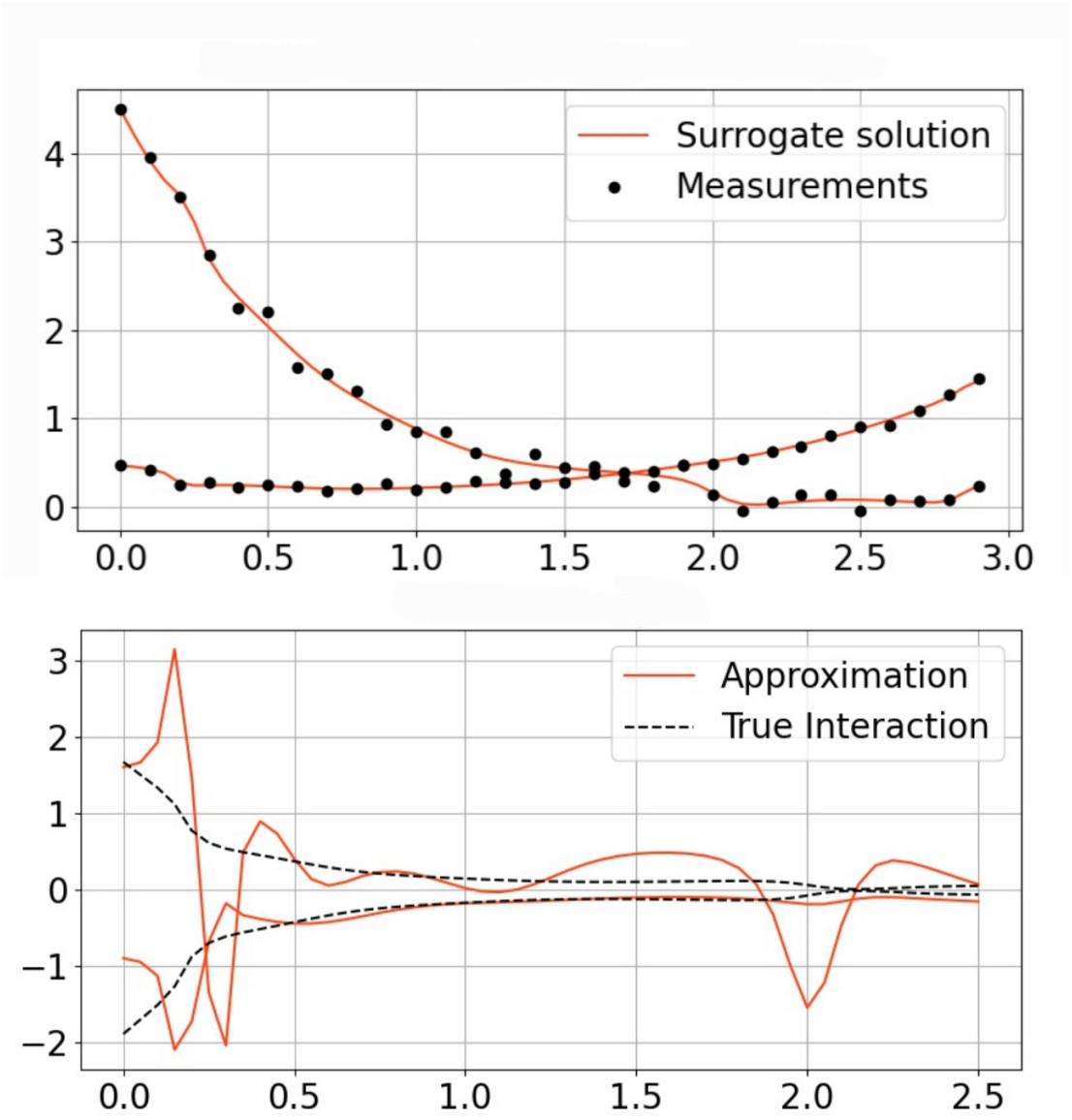


Figure 2.4: UDE method performance on the Lotka-Volterra model. Noisy data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term.

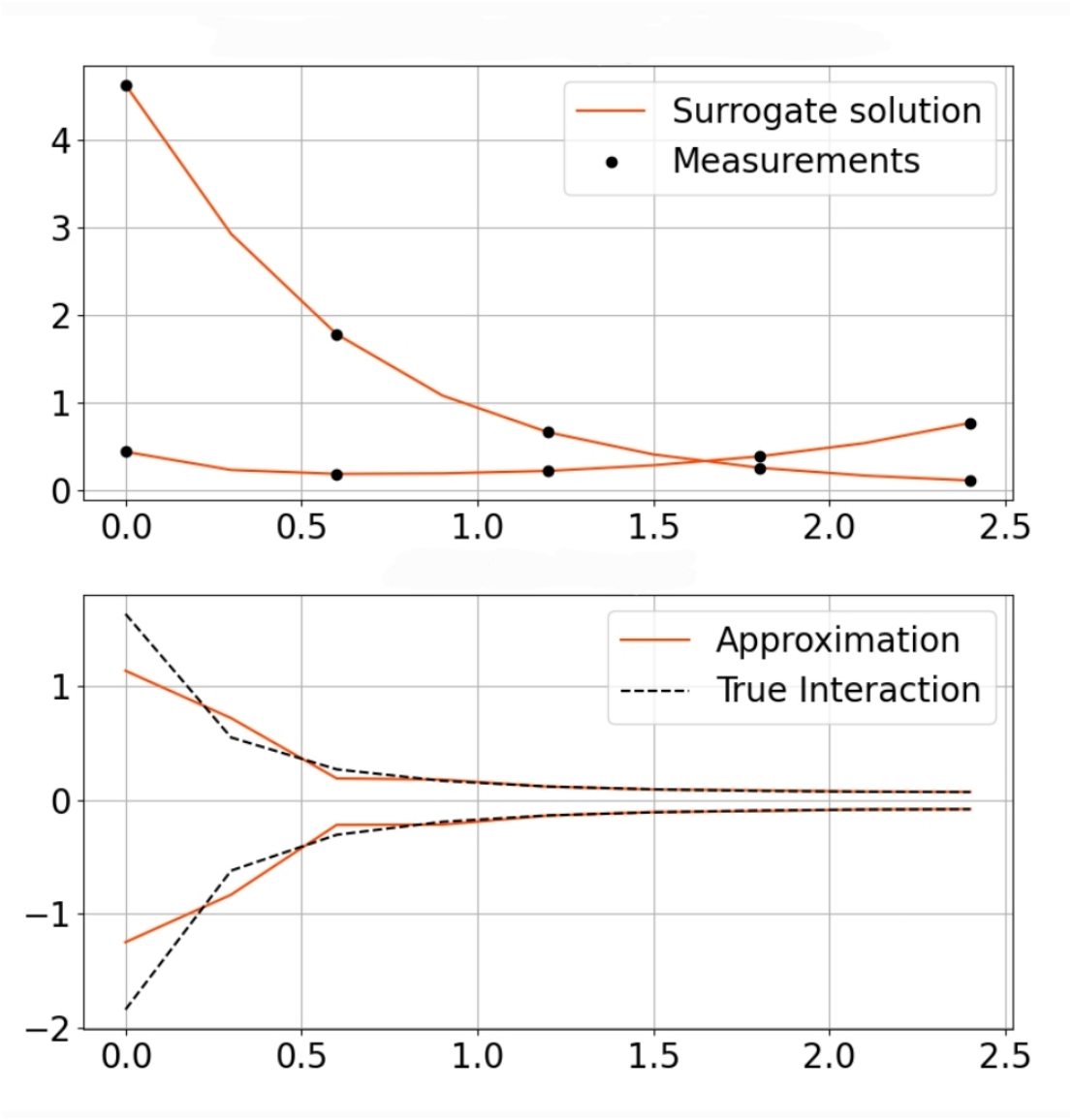


Figure 2.5: UDE method performance on the Lotka-Volterra model. Sparse data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term.

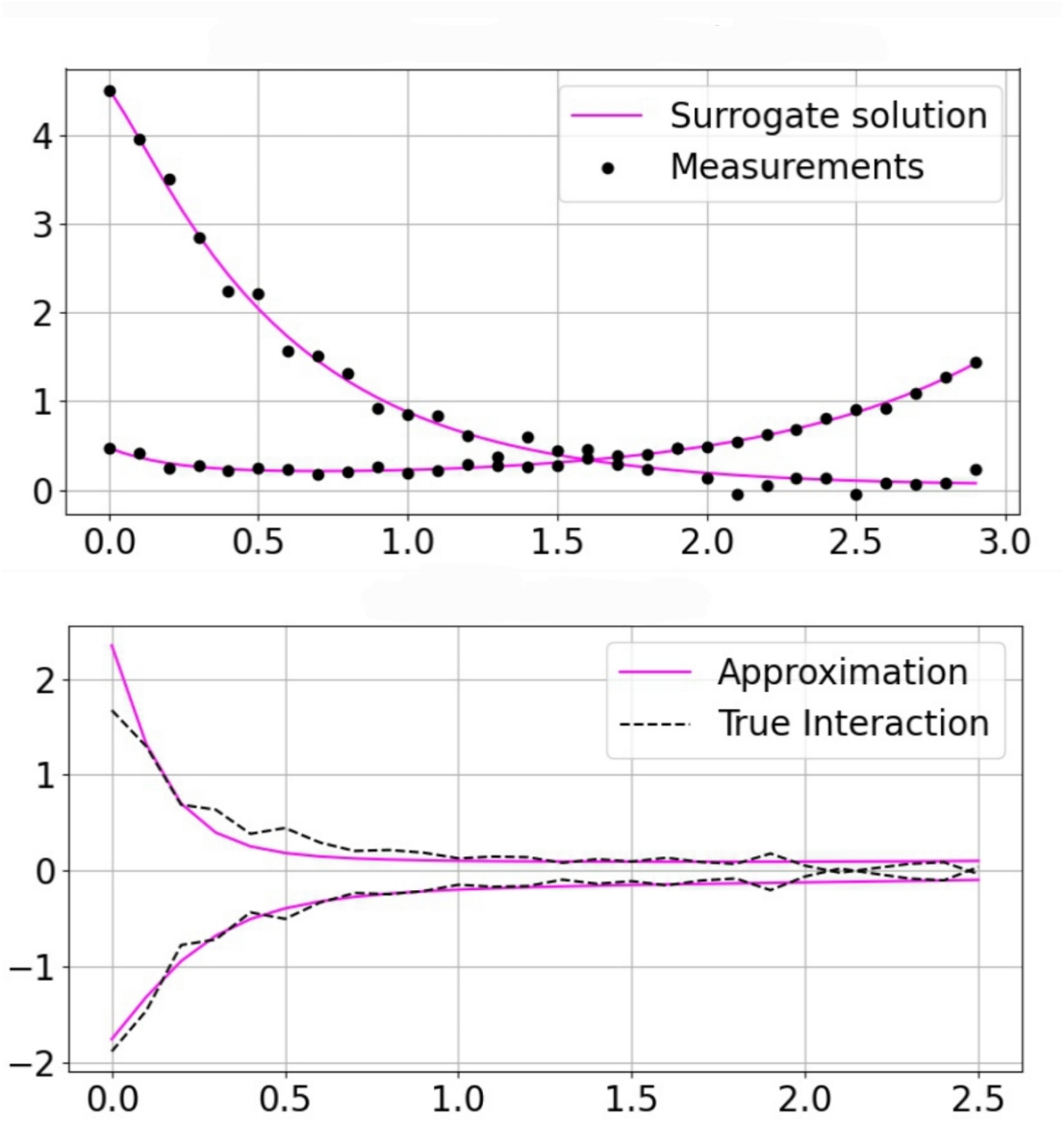


Figure 2.6: UPINN performance on the Lotka-Volterra model. Noisy data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term.

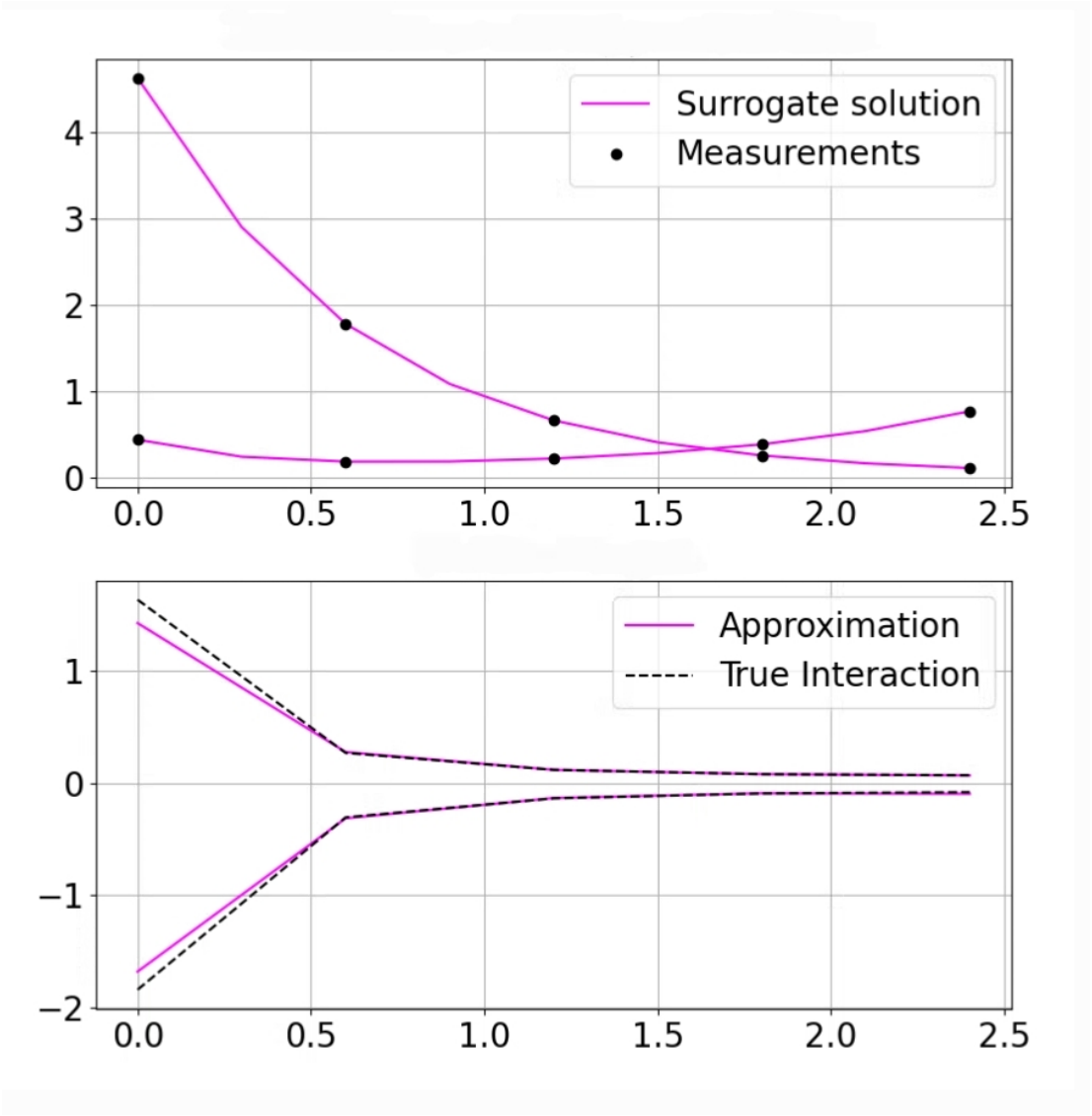


Figure 2.7: UPINN performance on the Lotka-Volterra model. Sparse data regime. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term.

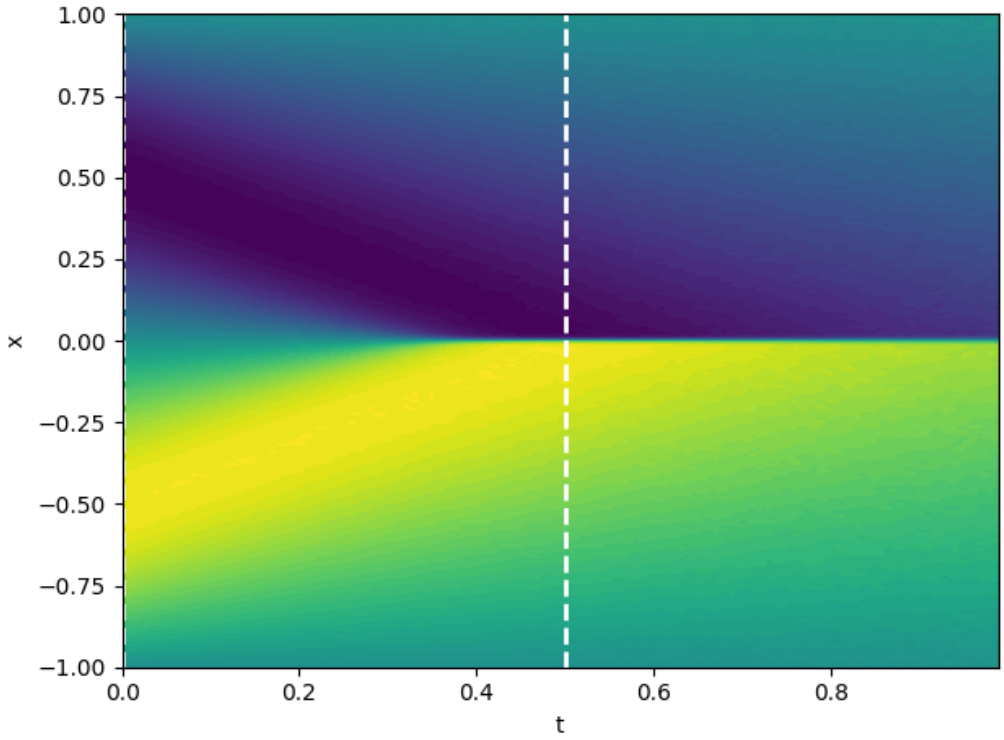


Figure 2.8: The reconstructed solution of Burgers' equation. The two vertical dashed white lines indicate the noisy experimental data that were sampled for the algorithm.

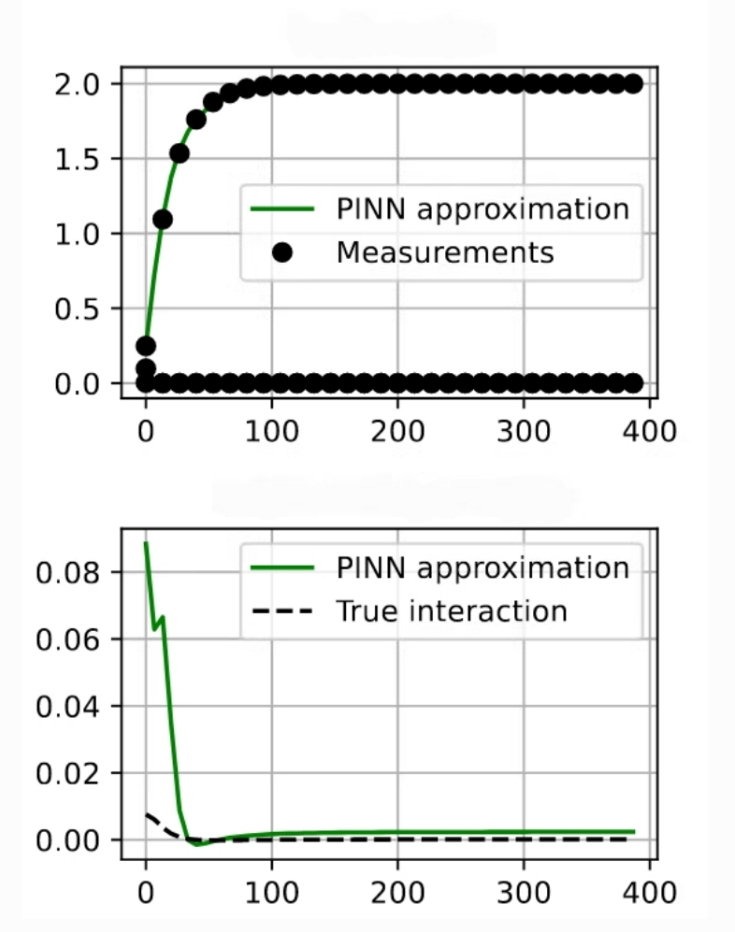


Figure 2.9: Learning the v_1 term using UPINNs. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term.

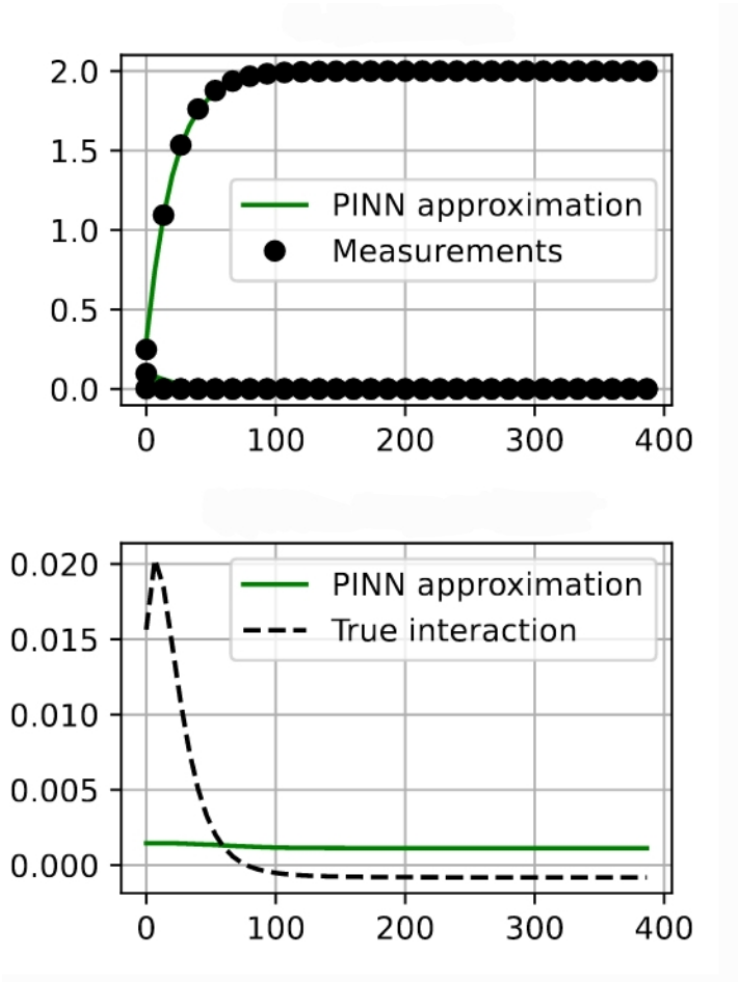


Figure 2.10: Learning the v_2 term using UPINNs. Reconstructed trajectory (top) and learned hidden interaction (bottom). x-axis: time, and y-axis: interaction term.

Chapter 3

Learning Chemotherapy Drug Action via Universal Physics-Informed Neural Networks

Chapter Summary

Quantitative Systems Pharmacology (QSP) is widely used to assess drug effects and toxicity before the drug goes to clinical trial. However, significant manual distillation of the literature is needed in order to construct a QSP model. Parameters may need to be fit, and simplifying assumptions of the model need to be made. In this work, we apply UPINNs to learn unknown components of various DEs that model chemotherapy pharmacodynamics. Using UPINNs, we learn three commonly employed chemotherapeutic drug actions (log-kill, Norton-Simon, and E_{max}) from synthetic data. Then, we use the UPINN method to fit the parameters for several synthetic datasets simultaneously. Finally, we learn the net proliferation rate in a model of doxorubicin (a chemotherapeutic) pharmacodynamics. We show that the UPINN can successfully learn the hidden terms and unknown parameters in a variety of DEs (with differing time and variable scales) that model the effect of chemotherapeutics on cancer cells. As the examples we study are only toy examples, we highlight the usefulness of UPINNs in learning unknown terms in pharmacodynamic and pharmacokinetic models. UPINNs can be used to find these terms and analyze them further to understand new chemotherapeutics and biological mechanisms that interact with them.

3.1 Introduction

Mathematical models that simulate the passage and effect of drugs within the body, otherwise known as *quantitative systems pharmacology (QSP)*, are a commonly used approach to assess drugs before they go to clinical trial [172]. Many drugs fail phase II and III clinical trials [176] due to inadequate understanding of the mechanism of action. However, via QSP models, which integrate biological, physiological and pharmacological data [149], it is possible to understand the underlying biological mechanisms and optimize drug dose and schedule. Additionally, QSP models can be used to predict drug toxicity or efficacy [133], as well as individual patient response to the drug. QSP model types range from systems of ODEs, compartmental models, cellular automata, and agent-based models [7].

Despite its benefits, QSP suffers from several limitations which make the models costly and time-consuming to develop. First, many QSP models are built by making simplifying assumptions about the underlying biology. In general, manual distillation of the literature (and potentially large amounts of data) is done in order to build these assumptions accurately, rather than building them directly or automatically in a data-driven way. Secondly, building a model fully (including parameter estimation) can be very time-consuming due to the model complexity and manual distillation of the literature [183]. QSP modelling is especially affected by these issues because a QSP model may comprise more than a dozen dynamical variables, and even more parameters [183]. Recently, ML has been used to improve and simplify the QSP model-building process. Decision trees, regression, and NNs are some of the tools that have been used to predict drug toxicity and patient response in a data-driven way [30]. A major benefit of ML is the almost entirely automatic learning of patterns in the data by a model, and potentially robust generalization to unseen data [30]. However, purely ML-based approaches often need large amounts of data to make accurate predictions, and are often uninterpretable. This means that although the model may produce reasonable predictions, the mechanism by which it does so is opaque. For this reason, an ML model may be sometimes called a "black box". In systems pharmacology, it is often important to know which features are the most important for prediction and why a certain outcome is expected. These findings shed light on underlying biological mechanisms, and inform drug development. This is also key to a sanity-check of the model prediction, and trust in the output.

Integrating ML and QSP is a promising research direction which has promise in binding together their strengths while compensating for their respective weaknesses. ML can be used to automate the learning of complex patterns, or abstract away the function of less important mechanisms. QSP, on the other hand, can provide structure to the ML model and inform it with prior knowledge. An integration of these two methods means that some

assumptions will be built in or enforced, but other interactions and mechanisms will be learned directly from the data. Due to the additional structure imposed by assumptions, potentially less data will be needed to learn the unknown components. The components of the model that need potentially more manual distillation can be learned more automatically from the data. Finally, the interpretability of the ML components may be augmented due to the surrounding structure imposed by the QSP model.

Existing integrations of QSP and ML tackle problems such as parameter inference [140], inference of model structure [55, 138], dimension reduction [49], and creation of virtual patients [5]. A prominent method to identify parameters from data is PINNs. PINNs are a flexible framework and have been extended in various ways [77, 134, 179] for the purposes of model fitting and inference, which are key problems in systems pharmacology. Given a DE that models a particular biological process, PINNs use a NN to fit a surrogate solution. The unknown parameters are treated as additional weights to be optimized. However, PINNs cannot be used to identify entire unknown components of the model. Inference of model structure, as described in [183], has followed several different directions, such as logic-based modeling [55], integration of QSP with genome-scale computational models [135], and UDEs [138]. UDE's are a popular method to identify unknown components of DEs. However, they are not robust to noise, as we showed in Chapter 2 and [128], and the UPINN method is more robust. In this chapter, we apply the UPINN method to chemotherapy modelling, as a proof of concept of what the method can accomplish in the realm of pharmacokinetic and pharmacodynamic modelling. We have applied the method to both simulated and in-vitro data, and showed high accuracy of identification of the unknown components of DEs.

Our contributions in this work are:

- We integrate ML in the form of PINNs with QSP models in order to identify unknown drug actions within QSP models. To illustrate, we apply the UPINN method to identify hidden terms in QSP models in cases of both synthetic and in-vitro experimental data.

- Via simulations, we show that three different types of drug action (Log-Kill, Norton-Simon, E_{max}) can be identified from the chemotherapy concentration and number of cells over time. We also employ our approach to recover dose-dependent parameters from several sets of data simultaneously and interpolate these parameters between dosages. This could potentially replace the repeated application of a standard PINN.

- We show that our approach can successfully identify the time-dependent net proliferation rate in cases of both synthetic and in-vitro experimental doxorubicin data.

In the remaining sections, we start with an overview of existing modeling approaches to chemotherapy modelling, PINNs and QSP structure identification methods. Following that, we use the UPINN method to identify the hidden components of various ODEs that model chemotherapy dynamics. We show the performance of our method on both synthetic and in-vitro data. For the experiments on in-vitro data, we learn the time-dependent effect of a chemotherapeutic, doxorubicin, on cell growth (proliferation).

3.2 Background

3.2.1 Physics-informed Neural Networks

PINNs [140] were developed by Raissi et al. for the purposes of solving forward and inverse problems of DEs (partial (PDE) and ordinary (ODE)). For the forward problem, a NN is used to provide a numerical solution to a PDE using data that satisfies the PDE. In this case, the data generating process is fully known. For the inverse problem, the DE is known except for certain parameters, which take on a real value. Using data, PINNs can be used to identify the parameters that best fit the data.

A drawback of PINNs is their inability to identify entire hidden terms within DEs. Originally developed to solve DEs and fit unknown parameters, PINNs require that the entire DE be explicitly specified within the loss function. However, when a fully specified model does not exist for the available data, it becomes challenging to uncover individual unknown components. Addressing this gap, UPINNs were introduced in [128] as an innovative extension designed to discover hidden terms in a data-driven manner. UPINNs enhance the PINN framework by integrating fully-connected NNs that represent each hidden term to be learned. The method utilizes loss functions akin to those employed in PINNs, including a data loss that ensures adherence to the known data and a DE loss that maintains conformity with the governing equations. These functions facilitate the simultaneous learning of both unknown terms and parameters. Thus, UPINNs extend the capabilities of PINNs by enabling the learning of hidden terms, enhancing their practical applicability in scenarios where the model is partially unknown.

The ability of UPINNs to recover hidden terms from data was demonstrated on the Lotka-Volterra system of ODEs, on a model of cell apoptosis with bistability, and on Burger's equation, a PDE. This underscores the broad applicability of the method. A comparison between UPINNs and UDEs [138] demonstrated that UPINNs excel in identifying hidden terms within the Lotka-Volterra model, particularly when the data is noisy. This capability

makes UPINNs particularly effective for models with sparse or noisy data, enhancing their suitability for complex fields such as pharmacodynamics. In this chapter, we explore further the power of UPINNs in identifying unknown terms from data, thereby avoiding modelling assumptions. We base our work on the models in [121] and [88] and we show that in every combination of drug action, we are able to make an approximation of the drug action using a NN. Furthermore, we can combine the UPINNs approach from [128] with the traditional PINN approach to identify parameters first, and then identify the hidden terms given the constraints imposed by these parameters.

3.2.2 Chemotherapy modelling

In many studies, the modelling of the drug is performed using assumptions on how the drug acts on the cells, and afterwards, the parameters of the model may be fit from data. For example, in [155], a Norton–Simon model of drug action and Gompertzian growth was assumed and then fit to data; the drug schedule can subsequently be perturbed and the best dosing schedule identified from model predictions. In [50], a reaction-kinetics analogy is assumed for the model and the parameters are fit to tumours in untreated mice; fits are then evaluated for treated mice. [16] used assumptions based on paclitaxel's effect on cells in different phases of the cell cycle, yielding a compartment-based model with DEs governing the transition of cells from one phase to the next. In [80], tumour growth was modelled with logistic growth, but the effect of the chemotherapeutic (also paclitaxel) is captured only through its effect on the carrying capacity. In [25], a novel fractional-order Gompertz model is introduced and determined to be a better fit for experimental mouse tumour data. UPINNs can refine the model's assumptions about tumour growth as well as assumptions about the effect of the drug.

Optimizing the order and sequence of chemotherapy and surgery is also an important problem. In [88], a mathematical model of ovarian cancer was developed to determine whether chemotherapy followed by surgery, or surgery followed by chemotherapy, was the superior treatment sequence. The authors examined both Gompertzian and logistic growth, and three different cell-kill models: log-kill, Norton–Simon, and $E_{\max}$. Both Gompertzian growth and logistic growth assume that cells grow until a carrying capacity, but the rate of growth is modelled differently by the two. Logistic growth posits a linear decline in growth rate as tumour size increases, whereas Gompertz growth models the per-capita growth rate as decreasing exponentially with size [65]. The *log-kill* model assumes that chemotherapy eradicates a constant proportion of tumour cells, regardless of tumour size. Conversely, the *Norton–Simon* model proposes that the rate of tumour regression is directly proportional to the tumour's current growth rate. Finally, the E_{max} model specifies the

maximum effect that a drug can achieve, occurring at saturating tumour volume. In all cases examined by [88], the sequencing of chemotherapy followed by surgery led to better outcomes than surgery followed by chemotherapy. However, it is key to note that in order to make a conclusive recommendation, all reasonable possibilities and assumptions need to be examined. When data is available, these assumptions can be learned from data, as we endeavour to show by adapting the UPINN method to this problem.

3.3 Methods

We build on the methods in Chapter 2 and [128]. In this work, we apply UPINNs in three different scenarios. First, we apply it to learn the drug action of a cancer growth ODE, modifying the method for better performance. Using the method, we are able to obtain an approximation of the drug action in three ODEs with three different drug actions. Secondly, we apply the method to a situation where the parameters vary with dose. We learn the parameters as a function of the dosage, and interpolate between observed dosages. Finally, we learn the net proliferation rate of doxorubicin from in-vitro data, after validating the method on similar synthetic data.

In the following subsections, we first describe how to apply the method generally, and in the subsequent sections, we detail the specific applications.

3.3.1 General approach

As introduced in the previous chapter, consider an ODE in the following form, with m variables, where $\vec{u}(t) \in \mathbb{R}^m$ and let $\mathcal{F}, \mathcal{G}$ be differential operators:

$$\frac{d}{dt}\vec{u}(t) = \mathcal{F}[\vec{u}](t) + \mathcal{G}[\vec{u}](t) \ . \tag{3.1}$$

Suppose furthermore that $\mathcal{F}$ can be evaluated for every possible input $\vec{u}, t$, but $\mathcal{G}$ is unknown and so cannot be. Then given a dataset of n points $\{t_i, \vec{u}_i\}$ which satisfy the DE, we can use the UPINN method to approximate $\mathcal{G}$ at these points, and at points that are linear combinations of pairs $(t_j, \vec{u}_j)$ in the dataset. We do this by representing $\mathcal{G}$ with a NN $G_{NN}(\vec{u}, t)$ returning a vector-valued output for each such combination $(\vec{u}, t)$. However, some of the outputs of $\mathcal{G}$ may be *a priori* known to be zero, and as such don't need to be modeled by the NN. A case of this can be seen in the following sections. Furthermore, although in (3.1), $\mathcal{F}$ and $\mathcal{G}$ are assumed to be summed, this assumption need not hold in order for

the method to learn $\mathcal{G}$ through G_{NN}. Instead, there may be a different (known) function combining $\mathcal{F}$ and $\mathcal{G}$ on the right-hand side of (3.1) (e.g. $\mathcal{F} \cdot \mathcal{G}$) and if $\mathcal{G}$ is identifiable given the data, the UPINN method should still be able to learn $\mathcal{G}$. However, more complex expressions tend to be more difficult for the method to learn. The term *identifiable* refers to the uniqueness of $\mathcal{G}$ that fits both the data and the DE well. While not a strict definition, if the UPINN method is able to learn two distinctly different functions for G_{NN} (within different trials) that both adequately fit the data, it suggests that G_{NN} (and hence $\mathcal{G}$) is not identifiable. This implies that multiple solutions may exist that satisfy the constraints, making it challenging to determine a unique G_{NN}.

As per the standard PINN approach, along with an approximation of $\mathcal{G}$, we obtain an approximation of the DE solution $\vec{u}$ using a NN U_{NN}. U_{NN} and G_{NN} are trained by minimizing the following loss:

$$L = L_{MSE} + L_{ODE} \ .$$

The MSE loss ensures that the output of the surrogate solution U_{NN} adheres to the data $\{t_k, \vec{u}_k\}$:

$$L_{MSE} = \frac{1}{n} \sum_{k=1}^{n} (U_{NN}(t) - \vec{u}_k)^2 \ .$$

As for the ODE loss, U_{NN} is autodifferentiated[1] with respect to t and evaluated at a set of collocation points $\{t_j\}$. This loss minimizes the difference between the derivative of $U_{NN}(t)$ and the right hand side of eq. 3.1. In effect, this loss component enforces the DE to hold.

$$\mathcal{L}_{ODE} = \frac{1}{K} \sum_{j=1}^{K} \left(\left. \frac{dU_{NN}(t)}{dt} \right|_{t_j} - (\mathcal{F}[U_{NN}](t_j) + G_{NN}(U_{NN}(t_j), t_j)) \right) \ .$$

G_{NN} is trained through the ODE loss component, and U_{NN} is trained through both loss components.

The necessity for a surrogate solution, rather than direct integration of the hidden term, arises from findings in [128], which demonstrate that the UPINN approach outperforms the

[1] A common operation in the training of NNs, which in this case computes the derivative of the output of the NN.

UDE method in terms of robustness to noisy data and efficacy in sparse data scenarios. It is important to acknowledge that the ODE loss is unlikely to reach exactly zero, indicating that the surrogate solution we derive is an approximation of the true solution.

In our work, we have achieved favorable outcomes by assigning equal weights to all losses, avoiding any bias towards one over another. However, previous studies, such as those documented in [75], have explored the impact of loss weights on model accuracy. It is important to note that when the intrinsic characteristics of the model and data cause the losses to differ by more than an order of magnitude, employing calibrated weights can significantly enhance the efficiency of the optimization process.

The selection of collocation points is a crucial methodological decision that can significantly impact results. Research documented in [128] indicates that between 10^3 and 10^4 uniformly distributed collocation points can effectively recover the hidden term with an MSE of 10^{-4}, even in the presence of sparse and noisy data. We follow this guideline in our current work. Employing more than 10^4 collocation points might enhance the quality of the solution and the accuracy of the learned hidden term but introduces higher computational costs. Collocation points can be adaptively selected, denser in areas where the solution is challenging, but as they serve as a data set for the ODE loss, managing large numbers can resemble processing extensive datasets. Thus, data may need to be batch-processed if too large for memory. Consequently, while increasing collocation points in specific areas can aid learning, their use should not be restricted due to their relatively low computational overhead.

3.3.2 Application to the identification of a cell-kill strategy

We are interested in applying the above method to identify the kind of drug action that a specific chemotherapeutic has. Using an ODE, we simulate a situation where the chemotherapeutic is applied in a single dose to a culture of cells, and the cell numbers are measured at intervals after the application of the chemotherapeutic. We choose the following DE to simulate the concentration and cell numbers over time:

$$\frac{dN}{dt} = \beta N(1-N) - C(t)G(N) \ , \tag{3.2}$$

$$\frac{dC}{dt} = -\gamma NC \ , \tag{3.3}$$

where $C(t)$ is the concentration of the drug (between 0 and 1), and $N(t)$ is normalized cell count, and as such takes values between 0 and 1. The drug C is assumed to be consumed by the cancer cells (N), so the decay is proportional to both N and C.

The first term governing the tumour dynamics is a growth term and is known. In general, we assume to have no information about $G(N)$, the drug action of the chemotherapeutic. However, for the purposes of testing the method, we choose G to have one of the three following forms, as per [88, 121]:

1. $a_1 N$ (Log-kill model)
2. $a_2 \beta N(1-N)$ (Norton-Simon model)
3. $a_3 N/(N+\delta)$ (E_{max} model)

where a_j are constants.

Data generation: We generate synthetic data $\{t_i, N_i, C_i\}$ satisfying the above DE, with initial condition $N(0) = 0.01$. C is assumed to be zero for $t \in [0, 12]$, and at day 12, the drug is instantaneously added to create a concentration of 1. From there the drug decays depending on the parameter $\gamma = 1.0$. For the E_{max} case, we have $\delta = 0.55$. β is assumed to be known and equal to 1.0, but can be fit using a regular PINN from untreated growth data. Finally, the constants $a_{1,2,3}$ are set as $a_1 = 2.8, a_2 = 11.0, a_3 = 2.4$. An example of this workflow is shown in the results. It is important to allow the cells to grow to their carrying capacity to ensure the full range of N is observed. If not, the learned drug action may fail to extrapolate over the entire range of N. To effectively learn the cell kill function, the constants should be set to observe the complete dynamic range of N as the cells respond to chemotherapy. Specifically, if the number of live cells decreases and then increases post-chemotherapy, a comprehensive understanding of the cell kill term might not be achieved. While the cells' behavior post-chemotherapy is sensitive to these parameters, we demonstrate the UPINN method using a parameter set that maximizes the observed range of N during chemotherapy, to highlight the method's capabilities. Specific sensitivity analyses were not conducted but our observations have shown that the model's performance is notably sensitive to the parameters a_1 a_2, and a_3. Future work should include a detailed sensitivity analysis to thoroughly explore the influence of these parameters on the model's predictions.

UPINN setup: We set up three NNs as follows: let $G(t, N) = C(t)G(N)$ and we approximate $G(t, N)$ with a NN G_{NN} with two inputs (t and N) and one output. Separately, we approximate $C(t)$ by C_{NN} with one input, t, and one output, $C(t)$. Finally, we have a NN U_{NN} with one input (t) and one output (N), which approximates the solution $N(t)$. Since C and N are variables in a system of ODEs, they both need to be represented using a NN. Furthermore, approximating $C(t)$ with a NN $C_{NN}(t)$ will prove useful when we need to interpolate the concentration of the drug at times that it is not observed.

The workflow proceeds in four steps. The input data for this process is a synthetically generated dataset $\{t_i, N_i, C_i\}$ for a specific cell-kill model.

1. Train C_{NN} using only an MSE loss relative to the $\{t_i, C_i\}$ data. This network facilitates the interpolation of the drug concentration between observed time steps.
2. Train $G_{NN}(t, N)$ using the $\{t_i, N_i\}$ data, employing the UPINN method outlined in the previous subsection.
3. Predict outputs from C_{NN} and G_{NN} for a new set of times $\{t_j\}$.
4. Calculate the ratio of G_{NN} predictions to C_{NN} predictions to approximate $G(N)$ for each time step t_j.

Since we also have the solution U_{NN} for every timestep, we are able to get an approximation of $G(N)$ for any N within the bounds of the data. This function cannot extrapolate to unseen N, so in generating $\{t_i, N_i, C_i\}$ it is important that the widest range possible is covered by the values N_i.

This method enables us to create a black-box representation of drug actions. The approximation of $G(N)$ for different timesteps t_j may then be run through a symbolic regression algorithm to obtain a closed-form of the function. For example, AI Feynman [158] can suggest possible closed-form functions linking the t_j and $G(N)$, trading off the simplicity of the function with how well it fits the data. This is different than training a regression model or NN because the symbolic regression algorithm searches a wide space of possible functions (e.g. including mathematical functions such as log or cos directly) rather than restricting the search space to polynomials or multi-layer perceptrons. Although the symbolic regression algorithm can find a closed form, which may shed some light on the underlying biological mechanisms of the hidden term, certain downstream applications for the drug action (e.g., finding an optimal drug schedule using reinforcement learning as in [54]) do not require a closed form.

The UPINN was implemented in PyTorch [123]. The NN which represents the surrogate solution was a fully-connected network with 8 hidden layers with 20 hidden units each, tanh activation. All NNs used in this work are structured similarly. Each NN which represents a hidden term was structured the same way. First, 5000 Adam [87] iterations were run, and then the Limited Memory BFGS optimization algorithm [98] was run until sufficient convergence. The MSE and PINN loss components were weighted equally at all times. The inputs to each NN were scaled such that it would be between 0 and 1 for all inputs (both time and number of cells). This is because the NN is best at approximating functions when the inputs and outputs are normalized [147].

3.3.3 Application to parameter identification for several drug dosages

In some situations, the DEs that govern a specific biological process has parameters that vary not in time, but as a function of chemotherapy dosage. One such example can be created by assuming that the parameters k_p, θ in logistic growth

$$\frac{dN}{dt} = k_p N \left(1 - \frac{N}{\theta}\right) \tag{3.4}$$

are in fact a function of the administered chemotherapy dosage, D. This changes the previous equation to

$$\frac{\partial N(t,D)}{\partial t} = k_p(D) N(t,D) \left(1 - \frac{N(t,D)}{\theta(D)}\right) \ , \tag{3.5}$$

where $k_p(D)$ and $\theta(D)$ are now the dose-dependent growth rate and carrying capacity. An initial condition constraint $N(0,D) = N_0$ can be included as well. In this case, for a higher dosage of chemotherapy, the growth of the cells would remain logistic, but the growth rate and carrying capacity would decrease. Conversely, if the dosage is low then the growth rate and carrying capacity would be higher. We consider this a realistic scenario for modelling the effects of doxorubicin, due to a model employed in a recent work [111].

Data generation: Rather than solving (3.5), it is simpler to use an ODE solver to solve (3.4) for different (k_p, θ). This generates several time series datasets $\{t_i, n_i\}_D$ of the number of tumour cells n_i over time (t), one for each chemotherapy dosage condition D. It is possible to aggregate them into one dataset and fit $k_p(D)$ and $\theta(D)$ simultaneously to all available data. In our UPINNs setup, we simulate three different datasets using logistic growth, under low, medium and high dosage conditions. Three different dosages were considered in each experiment (low, medium and high), given as 15.0, 25.0, and 45.0 nM respectively to the model. Since this data is synthetic and the dosage is not used in the ODE directly, the numerical values of the drug concentration are somewhat arbitrary. 48 datapoints were used in total for each experiment, which means there were 16 datapoints per dosage. We added noise proportionally to the mean of the data, as per synthetic experiments in [128] at a noise level of 0.03. We aggregate the data for all dosages, and model the system via eq. 3.5.

UPINN setup: We employ three NNs: one, $U_{NN}(t, D)$, models the surrogate solution and takes both time and dosage as inputs. The other two, $k_{p,NN}(D)$ and $\theta_{NN}(D)$, each take dosage as the only input and produce a single output. These networks are trained in a single step following eq. 3.5. Given that k_p and θ need to be estimated for only three dosages, the NNs designed for $k_p(D)$ and $\theta(D)$ can interpolate parameters for unobserved dosages. Additional constraints for interpolation may be enforced by incorporating an extra loss term.

It is worth noting that if standard PINNs were utilized to fit these dose-dependent parameters, a separate PINN would have to be fit for the dataset corresponding to each dose. This is a more significant computational burden, and also not scalable when more than a few dosages are available. It also does not allow interpolation between dosages. By modelling eq (2) using UPINNs, we are able to fit one surrogate solution N to all the datasets and obtain all the dose-dependent parameters simultaneously, provided they are identifiable.

The architecture of all NNs was the same as in the previous section. Training proceeded very similarly, with 5000 Adam iterations at first. Then, the data loss was weighted with a value $\lambda = 0.001$ and Adam training was continued for 1000 iterations. Then, L-BFGS finished the optimization. This training method was used in order to ensure that the PINN loss was minimized sufficiently during training. Similar scaling was employed on the input.

3.3.4 Application to in-vitro experimental data

We now apply our method to identify the drug action of a real chemotherapeutic from in-vitro cell counts. For this purpose, we gathered the data used in [111]. In this series of experiments, four different cell lines (MDA-MB-468 (basal-like 1), SUM-149PT (basal-like 2), MDA-MB-231 (mesenchymal), and MDA-MB-453 (luminal expressing androgen receptor)) were first allowed to grow undisturbed for at least three days, and then they were exposed to different concentrations of doxorubicin. The cells were exposed to doxorubicin for 6, 12 or 24 hours, after which the medium was changed. The concentrations of doxorubicin that were used by [111] are: 10nM, 20nM, 39nM, 78nM, 156nM, 312nM, 625nM, 1250nM, 2500nM.

In this work, the time-series cell count data is described by the following model:

$$\frac{dN_{TC}}{dt} = (k_p - k_d(t, D))N_{TC}(t)\left(1 - \frac{N_{TC}}{\theta(D)}\right) \ , \tag{3.6}$$

$$k_d(t, D) = k_{d,A}(D) \ , \tag{3.7}$$

$$k_d(t, D) = k_{d,B}(D)r(D)te^{1-r(D)t} \ , \tag{3.8}$$

where $N_{TC}(t)$ is the number of cells over time, k_p is the constant growth rate under control conditions, $k_d(D, t)$ is the death rate, dependent on dosage and time, $\theta(D)$ is the dose-dependent carrying capacity, and $r(D)$ is also a function of dosage. These parameters also take on different values per cell line. In [111], eq 3.7 and eq 3.8 are fit separately, and the final prediction for each dosage is a weighted combination of the predictions of the two models. The authors have fit $k_{d,A}(D), r(D), k_{d,B}(D)$ as constant parameters per dosage using a nonlinear least squares approach. No time-dependent or dose-dependent functions are fit. The parameters A and B represent that $k_{d,A}$ and $k_{d,B}$ are different functions that depend on dosage only.

We note that if k_p is allowed to take on any value, then fitting $k(D, t)$ and $\theta(D)$ simultaneously using our method is not identifiable. Hence, we simplified this model to the two following forms:

$$\frac{dN_{TC}(t)}{dt} = F_D(N, t)N_{TC}(t) \ , \tag{3.9}$$

$$\frac{dN_{TC}(t)}{dt} = G_D(t)N_{TC}(t)\left(1 - N_{TC}(t)\right) \ . \tag{3.10}$$

These two equations offer alternative approaches to modeling eq. 3.6, depending on our interest in learning the net proliferation rate $(k_p - k_d(t, D))$ from a given dataset. We choose to use equation 3.9 when modeling assumes exponential-type growth characterized by a single unknown term. Conversely, eq. 3.10 is used when more comprehensive background knowledge suggests that the cells exhibit logistic growth with a time-dependent net proliferation rate. In our experiments, we fit both equation 3.9 and 3.10 to evaluate and compare the outcomes from each modeling approach. The subscript D indicates that the function varies with each dosage. In our implementation, F_D primarily takes time as its input, though it could also consider both the number of cells, N, and time t. These two formulations allow for the incorporation of varying levels of prior knowledge, making them suitable for different applications. We suggest and work with these two formulations of the hidden term, because for different applications, more prior knowledge can be incorporated if it is known.

UPINN setup: Using either simulated or in vitro data of cell counts over time $\{t_i, N_i\}$, we can choose either equation 3.9 and learn $F_D(t)$, or use equation 3.10 and learn $G_D(t)$ for a specific dosage D. Both equations 3.9 and 3.10 are general enough, suitable for various scenarios including simulated data from eq. 3.7, simulated data from eq. 3.8, and actual in vitro data. Initially, we generate synthetic data based on eq. 3.6, eq. 3.7, and eq. 3.8 to verify that UPINNs can effectively identify the hidden term. Subsequently, we apply the models to fit the hidden term using in vitro cell count data. The workflow is described in more detail in the Results section.

Equations 3.9 and 3.10 are certainly identifiable for all tuples of time, cell count and time derivative values, denoted by $\{t_i, N_i, dN_i\}$. This is because for every tuple $\{t_i, N_i, dN_i\}$ at each of the collocation points, there is a unique solution of either eqs. 3.9 and 3.10 for F_D or G_D (respectively) at each dose D.

The loss function was the same as in the previous sections, and the training proceeded similarly. The NN architectures used were of 8 hidden layers of 128 units each, for each fully-connected NN. Training was done such that Adam iterations were stopped at 1500 iterations, or upon reaching a loss of $2 \cdot 10^{-3}$, whichever comes first. Then, L-BFGS iterations were started. This conditional stopping of Adam iterations was done in order to bring the weights sufficiently close to the optimum before L-BFGS starts, but not so close that L-BFGS fails to perform any optimization steps.

3.4 Results

In the following three sections, we detail the performance of the identification of hidden terms in all three of the scenarios. We apply our method to learn the drug action (the model that best describes the cell dynamics after treatment) as a function of the cell count only, then we learn growth rate parameters as a function of the dosage, and finally we learn the net proliferation rate of doxorubicin as a function of time.

3.4.1 Identification of drug action (Log-kill, NS, E-max) from synthetic data

As described in the methods, we generate synthetic data for three different drug actions, and we evaluate how well the underlying true drug action can be recovered. We test our method on sparse noiseless data generated in two different ways, and noisy data. We also

test a two-step procedure where we recover β first from the untreated data using a regular PINN, and then apply our method to find the drug action.

First, we test on sparse data with spacing that is uniform through time. We generate a time series dataset of 54 points for each drug action. Fig 3.1 shows the learned drug action for all three types of cell-kill strategies, when the spacing was uniform in time. In other words, it plots the learned $G(N)$ as a function of different N that appeared in the training data. We note that the MSE between the true drug action and the predicted drug action, as reported in Table 3.1 for all three types of drug action, are all on the order of 10^{-4}. A visual inspection shows that the predicted drug action matches the true drug action for almost all values of N.

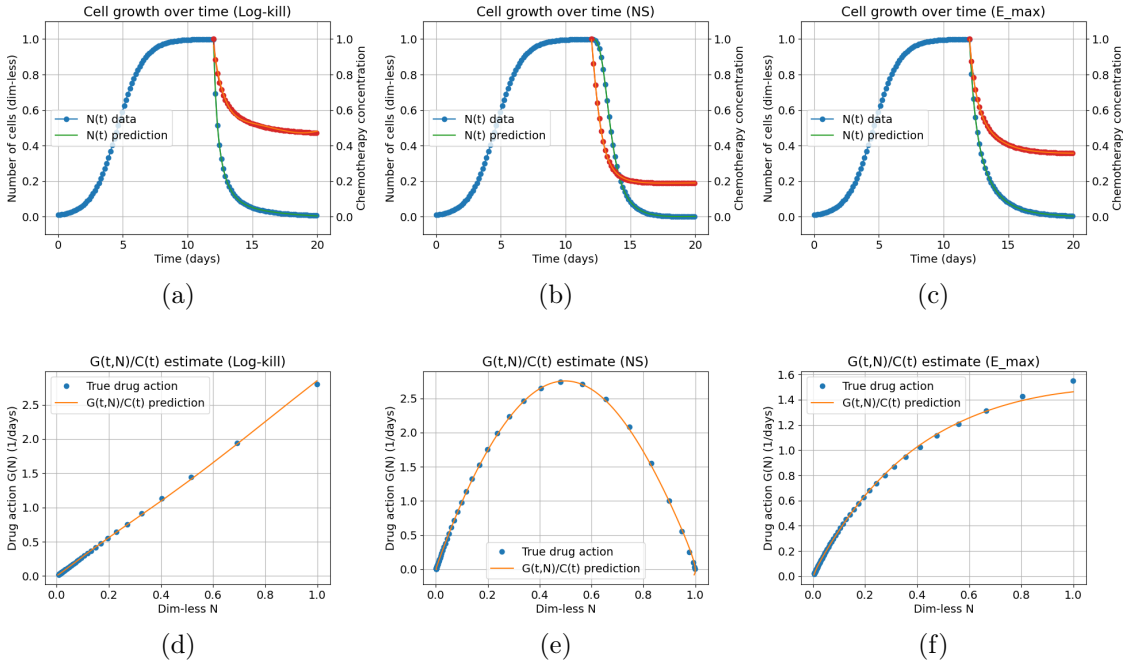


Figure 3.1: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). Data was collected with a fixed time period (0.15 days) elapsing between each data point. The drug actions are the functions $G(N)$ in Eq (3.2), replaced by one of: log-kill, Norton-Simon and E_{max}.

For the second data collection strategy, the spacing of the datapoints was dynamically chosen based on the rate of change of cell populations, so that there are proportionally more points at higher-slope areas compared to low-slope areas. This was done by first solving

each ODE with many timesteps (time-step size 0.001), and then filtering the resulting data. The settings for the filtering were such that 1 point was present for each cell decline of 0.05, and no more than 500 contiguous points were removed. This yielded datasets of 31 points for the log-kill model, and 29 points for the others. The results of running the UPINNs on these datasets can be seen in Fig 3.2. As evidenced by the MSEs in Table 3.1, the learned drug actions for both the log-kill and Norton-Simon model are on the order of 10^{-4} and 10^{-5}, giving comparable or better MSEs than using spacing that is equal in time. Given that the adjusted spacing uses half as much data, but achieves errors on the same order of magnitude (10^{-4}) highlights the importance of data collection in high-slope areas. In practice, this might mean collecting data as frequently as possible when a high cell decline is to be expected, and collecting less frequently when the cell growth has reached a plateau. It is important to note that the number of collocation points remained constant across both experiments, ensuring that variations in error cannot be attributed to differences in collocation point density.

Then, we test the method on noisy data, constructed by adding noise to both dynamically spaced (Fig 3.4) and equally spaced data (Fig 3.3). For each of the drug actions (log-kill, NS, and E_{max}), we find that the method has an MSE of 10^{-4}, 10^{-3}, and 10^{-5} respectively (Table 3.1) under noisy conditions when the spacing is equal in time, but not when the spacing is adjusted dynamically. When the spacing is adjusted dynamically then the errors are an order of magnitude higher (10^{-3}, 10^{-2}, and 10^{-4} respectively). The noise is added proportionally to the mean of the data, with the noise level being 0.03. The predicted drug action is visually quite a bit different from the true $G(N)$ for larger values of N when the spacing is adjusted. More investigation is needed into why the adjusted spacing has a detrimental effect when the data is noisy. In addition, it is worth noting that adding only noise to equally spaced data has a negligible effect on the MSE of all drug actions. The MSEs remain at most 10^{-4} for both noisy and noiseless equispaced data. This highlights the robustness of the method. A comparison of the MSE values of the learned drug actions for all three types of tests can be seen in Table 3.1, with the lowest values for each column bolded.

Finally, we apply a two-step process where first β is learned using a standard PINN from both noiseless and noisy data, and then this β estimate is employed during the estimation of $G(N)$. This showcases a way to apply UPINNs where no prior knowledge of the parameters or drug action is needed. The results of this test can be seen in Fig 3.5. Fig 3.5 (a) is the noiseless data, created using the same parameters as before, using the Norton-Simon model. For noiseless data, we obtain an estimate of $\beta = 0.999$ from the untreated data, which yields an MSE of 1×10^{-6} compared to the true $\beta = 1$, with subsequent MSE of the drug action being 1.3×10^{-4}. For noisy data, we obtain an estimate of $\beta = 0.997$ (MSE of

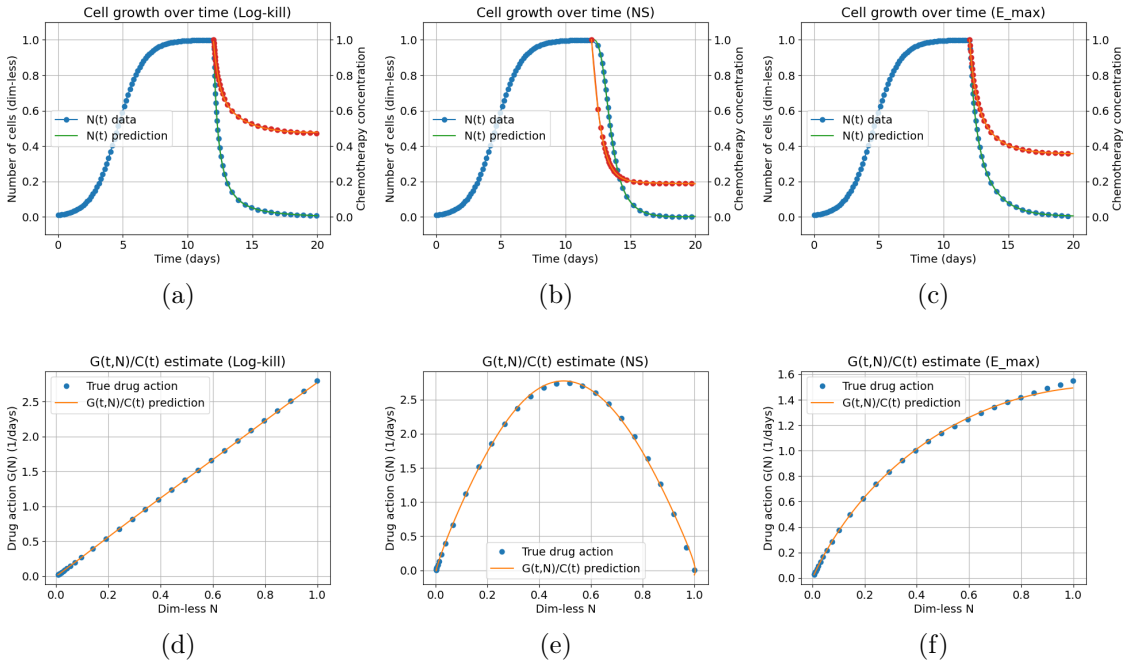


Figure 3.2: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data is noiseless but collected such that there is one datapoint for each 0.05-interval decrease in N. Although there are half as many points collected than in the equispaced case, the MSE between the true drug action and the predicted drug action is still on the order of 10^{-4} as shown by Table 3.1

9×10^{-6}), with subsequent MSE of the drug action being 4.5×10^{-3}. The performance is clearly worse when the data is noisy, but a visual inspection shows that we can recover the curve and β well for both noisy and noiseless data.

Overall, we show that the UPINN method applied to the discovery of chemotherapeutic drug actions is effective in recovering the drug action in three common cases. It recovers the drug action with a low MSE (10^{-3}) in cases of both sparse and noisy data, and can be combined with the standard PINN approach to yield MSEs on the same order of magnitude.

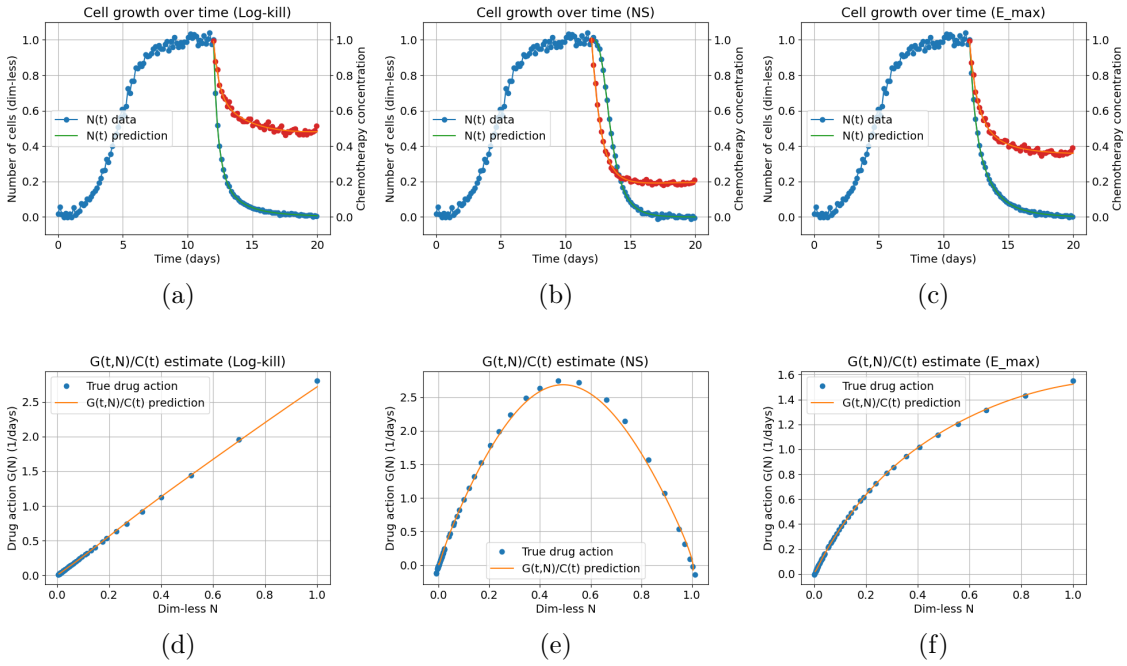


Figure 3.3: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data has a noise level of 0.03 added proportionally to the mean of the variables.

3.4.2 Inference and interpolation of dose-dependent logisitic growth parameters

In this set of synthetic experiments, we show that we attain low MSE (10^{-3} to 10^{-8}) in recovering the true proliferation rate $k_p(D)$ and carrying capacity $\theta(D)$ for known dosages, and interpolate between the dosages. At the low dosage, the growth rate and carrying capacity are set to be the highest ($k_p = 0.03, 0.08$ hr^{-1}, $\theta = 2.0, 4.0$), and at the highest dosage they are the lowest ($k_p = 0.01, 0.06$ hr^{-1}, $\theta = 1.0, 3.0$). The set of values chosen for k_p reflects either low growth rates (0.01-0.03), high growth rates overall (0.06-0.08), or a large range of growth rates (0.01-0.08). Similar parameter choices were made for θ, where low growth rates correspond to θ ranges 1.0-2.0, large range of growth rates correspond to ranges 1.0-4.0, and high growth rates overall correspond to θ between 3.0-4.0. Table 3.2 and 3.3 show the accuracy in recovering the true parameters and fitting the data for both noisy and noiseless data respectively. The true parameters are recovered on the order of

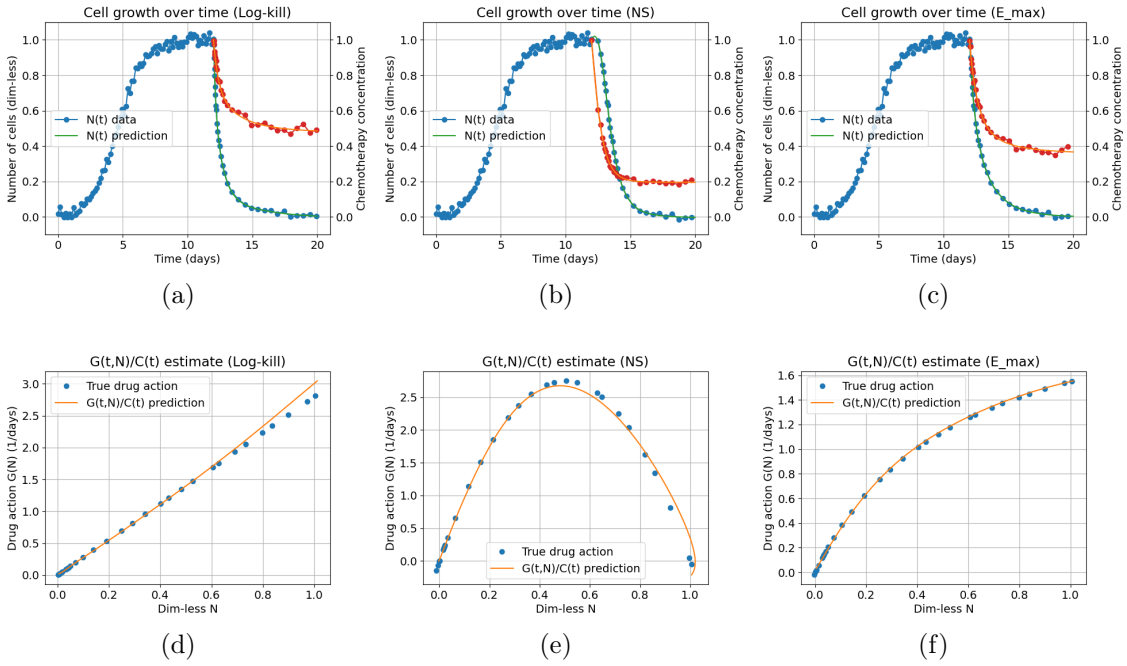


Figure 3.4: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data is spaced so that times of high cell decline have proportionally more observations (one datapoint for each 0.05-interval decrease in N). It also has a noise level of 0.03 added proportionally to the mean of the variables.

10^{-3} to 10^{-6} even in cases of sparse and noisy data. The specific values of these parameters were selected to be similar to those fit by [111]. The resulting surrogate solution MSE for noiseless and noisy data is at most 10^{-3}.

Figure 3.6, (a)-(c) shows the results for row 4 of the noisy data (Table 3.3), and Figure 3.6, (d)-(f) shows the results for row 7 of the noisy data. It can be seen that the interpolation is performed on a smooth continuous curve, the surrogate solution is correct for each dosage, and the resulting equations, when solved using the estimated parameters, fit the data very well visually. In addition, the model MSE (the MSE between the data and the model) is on the order of 10^{-3}. It is worth noting that sometimes, the surrogate solution returns and the fit implied by the parameters does not agree. The surrogate solution may agree with the data but the inferred parameters, when substituted into the DE, do not show an accurate fit. This may happen because the PINN loss is not sufficiently minimized compared to the MSE loss. For this reason, for this set of experiments, we perform both checks to ensure

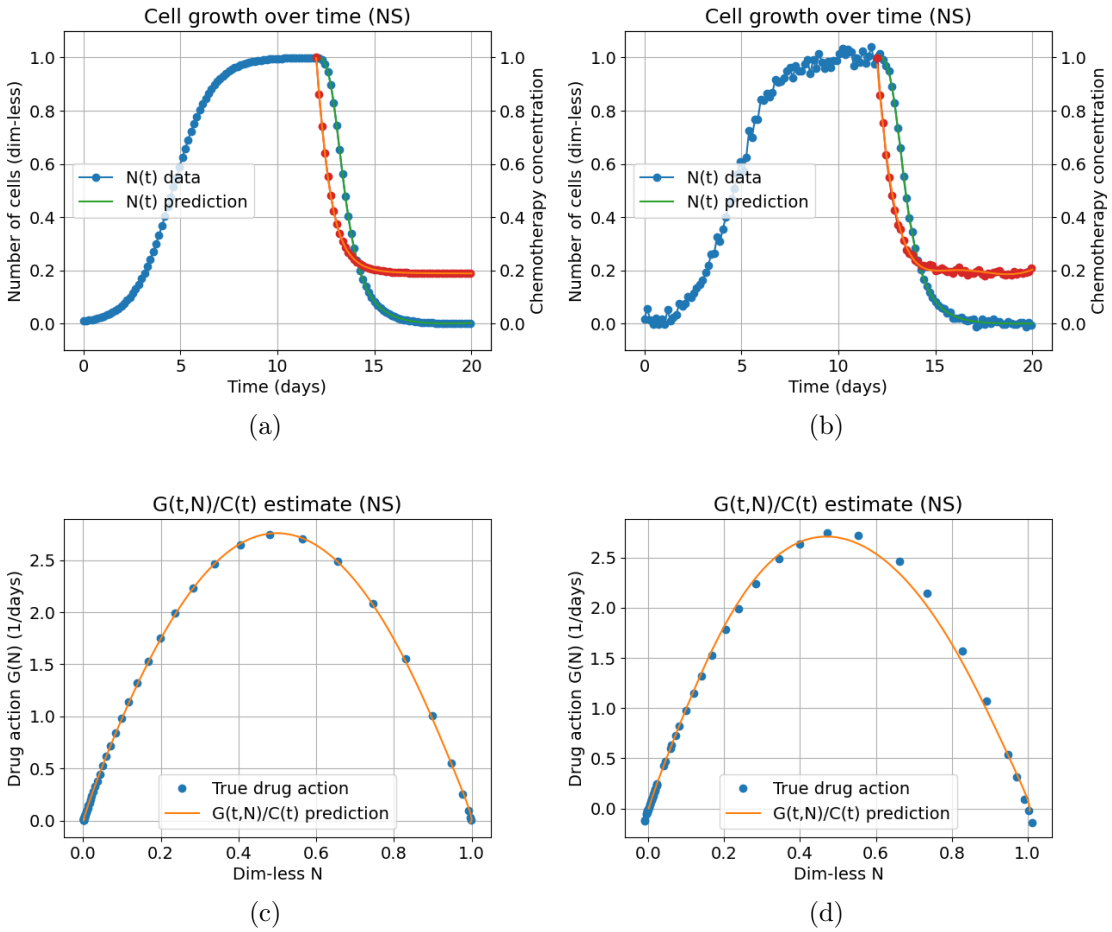


Figure 3.5: (a) Equispaced noiseless data with Norton-Simon drug action, (b) Learned drug action after the parameter β was fit from the first 12 days of noiseless data (final estimate: 0.999, MSE 1×10^{-6}),(c) Equispaced noisy (0.03) data with Norton-Simon drug action, (d) Learned drug action after the parameter β was fit from the first 12 days of noisy data (final estimate: 0.997, MSE 9×10^{-6})

Drug action / Type of data	Log-kill	Norton-Simon	E_{max}
Equal spacing	1.00×10^{-4}	4.13×10^{-4}	1.85×10^{-4}
Adjusted spacing	$\mathbf{6.30 \times 10^{-5}}$	$\mathbf{3.84 \times 10^{-4}}$	2.00×10^{-4}
Equal spacing + noise	1.66×10^{-4}	4.17×10^{-3}	$\mathbf{7.32 \times 10^{-5}}$
Adjusted spacing + noise	4.29×10^{-3}	1.22×10^{-2}	1.96×10^{-4}

Table 3.1: MSE of the drug action predictions (compared to the ground truth drug action) for different types of data. Equal spacing means that the data was equally spaced in time (one datapoint per 0.15 days). Adjusted spacing means that the data was collected with proportionally more datapoints when the slope of the decreasing cells is high in magnitude. When noise is added, the noise level is 0.03, added proportionally to the mean of the data. The lowest value in each column is bolded.

that the results are reasonable. Overall these results show that the UPINN method may provide an alternative to fitting multiple PINNs, with the added capability of interpolating between observations.

$k_p(D)$	$\theta(D)$	MSE of θ	MSE of k_p	Model MSE
$[0.03, 0.02, 0.01]$	$[2.0, 1.5, 1.0]$	0.00012	6.3e-7	2.4e-05
$[0.08, 0.07, 0.06]$	$[2.0, 1.5, 1.0]$	3.678e-06	6.5e-05	0.0002
$[0.08, 0.05, 0.01]$	$[2.0, 1.5, 1.0]$	0.00014	0.0029	0.00154
$[0.03, 0.02, 0.01]$	$[4.0, 3.5, 3.0]$	0.000465	4.456e-08	2.8e-05
$[0.08, 0.07, 0.06]$	$[4.0, 3.5, 3.0]$	4.94e-05	6.69e-05	0.0004
$[0.08, 0.05, 0.01]$	$[4.0, 3.5, 3.0]$	0.00042	1.67e-05	0.006
$[0.03, 0.02, 0.01]$	$[4.0, 2.5, 1.0]$	0.0011	5.8e-07	1.9e-05
$[0.08, 0.07, 0.06]$	$[4.0, 2.5, 1.0]$	4.35e-05	7.01e-05	0.0002
$[0.08, 0.05, 0.01]$	$[4.0, 2.5, 1.0]$	0.0036	0.00089	0.0035

Table 3.2: MSEs for each experiment (noiseless data). The row of θ and k_p indicates the values used for each dosage (low, medium, and high dosages respectively). Model MSE is the MSE between the data and the model.

3.4.3 Learning doxorubicin dynamics

In the next set of experiments, we aim to learn the dynamics of doxorubicin from in vitro data, as collected and modeled by [111]. Specifically, we seek to learn $(k_p - k_d)$ in equation 3.6. Since [111] modeled k_d as a weighted sum of two models, eq. 3.7 and eq. 3.8, this formulation inherently involves assumptions and modeling choices, particularly when the death and proliferation rates are dosage-dependent. To better assess these assumptions, we use UPINNs to learn $(k_p - k_d)$ directly from the data, allowing us to determine whether the structure imposed by eq. 3.7 and eq. 3.8 emerges naturally from the data or if an alternative formulation provides a better fit.

Due to the absence of the ground truth net proliferation rate for in-vitro experimental data, we first validate the method using simulated data. We use [111]'s ODE to generate data from eq. 3.6 and each of 3.7 and 3.8. We then learn $F_D(N, t)$ and $G_D(t)$ separately as per the methodology for eq. 3.8. Hence, for model validation on simulated data, we consider four experiments:

1. Generate data from eq. 3.6 and eq. 3.7, then learn $F_D(N, t)$ using UPINNs.

$k_p(D)$	$\theta(D)$	MSE of θ	MSE of k_p	Model MSE
$[0.03, 0.02, 0.01]$	$[2.0, 1.5, 1.0]$	0.0006	3.72e-06	0.001
$[0.08, 0.07, 0.06]$	$[2.0, 1.5, 1.0]$	0.0001	7.97e-05	0.0017
$[0.08, 0.05, 0.01]$	$[2.0, 1.5, 1.0]$	0.00024	0.0029	0.0015
$[0.03, 0.02, 0.01]$	$[4.0, 3.5, 3.0]$	0.0029	9.0e-07	0.0044
$[0.08, 0.07, 0.06]$	$[4.0, 3.5, 3.0]$	0.00016	6.6e-05	0.0086
$[0.08, 0.05, 0.01]$	$[4.0, 3.5, 3.0]$	0.025	0.00094	0.0095
$[0.03, 0.02, 0.01]$	$[4.0, 2.5, 1.0]$	0.0011	2.7e-07	0.0033
$[0.08, 0.07, 0.06]$	$[4.0, 2.5, 1.0]$	0.00023	7.1e-05	0.0055
$[0.08, 0.05, 0.01]$	$[4.0, 2.5, 1.0]$	0.0058	0.0013	0.0092

Table 3.3: MSEs for each experiment (noisy data). The row of θ and k_p indicates the values used for each dosage (low, medium, and high dosages respectively). The noisy data was created using a noise level of 0.03. Model MSE is the MSE between the data and the model.

2. Generate data from eq. 3.6 and eq. 3.7, then learn $G_D(t)$ using UPINNs.
3. Generate data from eq. 3.6 and eq. 3.8, then learn $F_D(N, t)$ using UPINNs.
4. Generate data from eq. 3.6 and eq. 3.8, then learn $G_D(t)$ using UPINNs.

We note that the second experiment would entail learning a constant value using a NN, so we exclude this case from further analysis. Subsequently, we learn both $F_D(N, t)$ (eq. 3.9) and $G_D(t)$ (eq. 3.10) from the in-vitro SUM-149 data.

For our in-silico analysis, we generated time series data $\{t_i, n_i\}$ (number of tumour cells over time) either via equation 3.7 or 3.8, for different sets of realistic parameters. Since $k_{d,A}(D), k_{d,B}(D), \theta(D), r(D)$ take on scalar real values for a given dosage, it suffices to generate a dataset with one parameter combination for each "dosage". We generated several such datasets. By comparing it to the ground truth net proliferation rate, we are able to validate the performance of the model. Parameter values for these data were chosen to be similar to the values obtained via fitting by [111]. k_p was selected to be 0.0354 hr^{-1}. This value was obtained by fitting a standard continuous PINN [140] to the control data

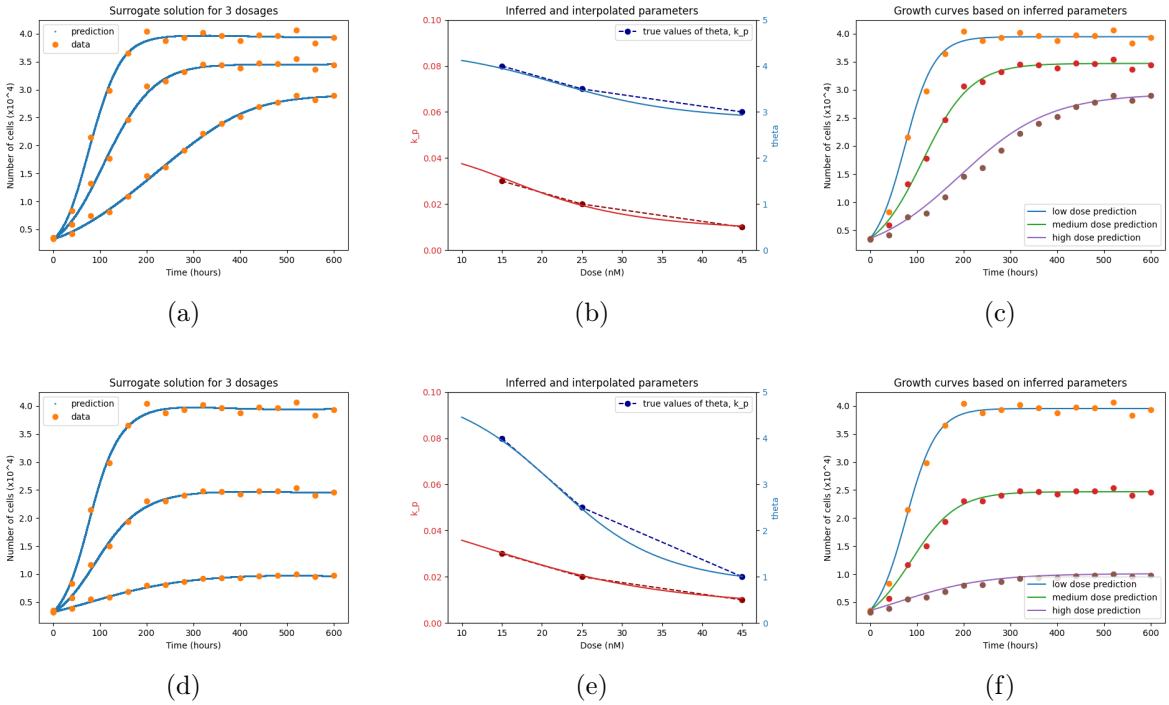


Figure 3.6: (a) Surrogate solution for row 4 of Table 3.3. (b) Fit and interpolated parameters per dosage. Blue is θ, and red is k_p. (c) Data was simulated using the inferred parameters for each of the three dosages. (d)-(f) Same as (a)-(c) but for row 7 of Table 3.3.

provided by the authors of [111]. Finally, we added noise to the synthetic data, as per synthetic experiments in [128] at a noise level of 0.03. The data generated for each dosage was normalized such that both the time and cell counts range from 0 to 1. In other words, the carrying capacity information and the timescale information is lost at this stage, but can be recovered later after $F_D(N,t)$ and $G_D(t)$ are fit, by undoing the scaling. This initial scaling improves the performance of the UPINN method.

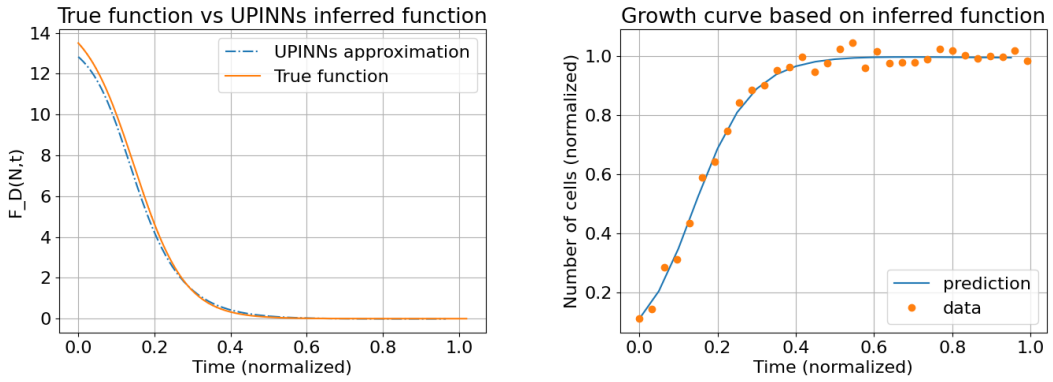
First, data from eq. 3.6 combined with 3.7 was generated, utilizing different parameter values to create datasets for different "dosages". Table 3.4 summarizes the different parameters and noise levels tested. For the same dataset, the UPINN was used to fit $F_D(N,t)$ from eq 3.9 5 times, and the best, mean and standard deviation MSE is reported. The MSE is computed between the predicted cell count and the actual cell count. The predicted cell count was obtained by substituting the learned $F_D(N,t)$ into eq 3.9 and solving the equation numerically. Figure 3.7 shows a particular fit of $F_D(N,t)$ using data

generated from $\theta = 1.0$, $k_{d,A} = 0.03$ hr^{-1}. The method performs well in identifying $F_D(N,t)$ for different realistic values of the parameters, as evidenced by the MSE model error being at most 10^{-4}. Given that the standard deviation of the fits is on the order of 10^{-5} and 10^{-6}, we can conclude that for this equation, the UPINN method produces reproducible fits.

θ	$k_{d,A}$	noise	Best solution fit MSE	Mean MSE	St. Dev. of MSE
1.0	0.05	0.0	1.11e-05	4.17e-05	2.31e-05
1.0	0.03	0.0	6.03e-06	1.42e-05	1.48e-05
1.0	0.01	0.0	2.20e-07	3.24e-06	2.99e-06
1.0	0.05	0.03	5.83e-04	5.98e-04	2.41e-05
1.0	0.03	0.03	5.02e-04	5.11e-04	1.30e-05
1.0	0.01	0.03	2.15e-04	2.17e-04	1.71e-06

Table 3.4: Fit of $F_D(N,t)$ using data generated via eq. 3.7, with MSE computed between the inferred solution (using the learned hidden term) and the data. Each experiment was run 5 times; the error of the best fit is shown, along with the mean and standard deviation of all 5 runs.

Next, data from eq. 3.6 combined with 3.8 was generated. In this case, the net proliferation rate is time-dependent, whereas in eq. 3.7 it is constant. Table 3.5 summarizes the MSE of the learned $F_D(N,t)$ when the data was generated according to eq 3.8. We can see that the model MSE is on the order of at most 10^{-4} for most parameter combinations. In this case, several different $(r_d, k_{d,B})$ combinations are chosen, and 5 runs are performed. The MSE of the best solution and the mean and standard deviation of all 5 runs is reported. Since the standard deviation is at most 10^{-3} (unless one of the runs completely fails to learn the underlying function), we do not anticipate that adding more runs would change these MSEs significantly. The values chosen for these parameters were informed in part by [111] such that a variety of realistic growth curves were generated. Carrying capacity θ is set to 1 to avoid the additional step of rescaling the cell counts. Table 3.5 reports the results for both noisy and noiseless data. It can be seen that the very first row had a run that has an error on the order of 10^1, but the MSE for the best model run is still 10^{-5}. This indicates that for this particular parameter combination, many of the model fits were inaccurate. However, due to the inherent stochasticity of the NN fitting process (such as random initialization and optimizer step choices), it is still possible to achieve a



(a) (b)

Figure 3.7: (a) Fit of $F_D(N,t)$ to data from eq 3.7 with parameters $k_{d,A} = 0.03, k_p = 0.0354$ (b) ODE solution generated using the inferred $F_D(N,t)$, along with the data for comparison. F_D is recovered well using the UPINN method, and the resulting model using the recovered F_D matches the data well.

model with an MSE below 10^{-3} within five trials, or random seeds. While it is possible to mitigate this variation by fixing random seeds in PyTorch and NumPy, we deliberately conduct multiple trials to illustrate the impact of stochasticity. In real-world applications of UPINNs, NN initialization and optimization steps will naturally introduce some randomness. Importantly, we emphasize that UPINN training remains robust to this stochasticity in most cases. Figure 3.8 (a), (b) shows a fit of $F_D(N,t)$ to row 2 of Table 3.5. The solution shows a time-dependent net proliferation rate, which is learned with a model MSE of 10^{-6}.

We also demonstrate that $G_D(t)$ can be learned well from equation (3.8). Figure 3.8 (c), (d) shows the fit of the same row 2, with parameters $r_d = 0.017$ and $k_{d,B} = 0.03$, but this time fitting $G_D(t)$. In this case, the solution matches the ground truth very closely except for $t \in [0.6, 1.0]$. At this point, the predictions start to diverge. This is likely because the function ceases to be identifiable in this time interval.

Since the UPINN method is able to accurately recover the hidden terms $F_D(N,t)$ and $G_D(t)$ for both model 3.7 and 3.8, we apply the UPINN method to learn $F_D(N,t)$ and $G_D(t)$ for in-vitro data, where no ground truth is available. Figure 3.9 shows the learned time-dependent term $F_D(N,t)$ for the application of the doxorubicin chemotherapeutic to SUM-149 cells, at 312nM concentration, for an exposure time of 6hrs. After 6 hours, the medium was changed such that there is no more chemotherapeutic affecting the cells past this point. The cells were then allowed to grow undisturbed. This curve is very similar in

r_d	θ	$k_{d,B}$	noise	Best solution MSE	Mean MSE	St. Dev. of MSE
0.017	1.0	0.05	0.0	1.83e-05	23.0	46.0
0.017	1.0	0.03	0.0	4.38e-06	8.06e-05	6.14e-05
0.017	1.0	0.01	0.0	1.35e-05	2.59e-05	9.87e-06
0.017	1.0	0.05	0.03	2.28e-04	1.03e-02	6.56e-03
0.017	1.0	0.03	0.03	4.37e-04	4.99e-04	7.25e-05
0.017	1.0	0.01	0.03	5.00e-04	5.11e-04	8.82e-06

Table 3.5: $F_D(N,t)$ (eq 3.9) using data generated via eq. 3.8, with MSE computed between the inferred solution (using the learned hidden term) and the data. Each experiment was run 5 times; the error of the best fit is shown, along with the mean and standard deviation of all 5 runs. The first row of the table shows a significantly higher error compared to the rest because one of the five runs failed, which skewed the mean and standard deviation of the MSE.

shape to the one shown in Fig. 3.8 (a). The peak occurs at approximately the same height as well. Figure 3.9 (c), (d) shows the time-dependent net proliferation, $G_D(t)$, learned for the same in-vitro dataset. We can see that this curve is not possible to construct using equation 3.7 or 3.8 alone, or even with a weighted average of these equations. This is because the function shown is very flat up until $t = 0.4$, but the derivative increases after this point. In contrast, a weighted average of 3.7 and 3.8 could not have a derivative of zero for any finite time interval given their formulations. Hence, fitting the UPINN to find $G_D(t)$ provides insights into the true net proliferation rate as a function of time. Initially, the net proliferation rate behaves more like 3.7, and subsequently more like 3.8. The large nonlinearities in F_D and G_D also show that a function that is more similar to a sigmoid may fit the data better than the model used in [111]. Tables 3.6, 3.7 show the MSE between the solution using the learned function $F_D(N,t)$ and the data. MSE is computed between the inferred solution (using the learned hidden term) and the in-vitro data. Each experiment was run 10 times; the error of the best run is shown, along with the mean and standard deviation of all 10 runs. The inferred solution fits the data well in most cases, as evidenced by the MSE between the inferred solution and the data being on the order of 10^{-4} for over half of the time-series datasets.

3.5 Discussion

In this work, we apply the UPINN method to learn the chemotherapy drug action in several different ODEs, applying the method to both synthetic and in-vitro experimental data. Rather than making assumptions about the drug action, we can learn it from data in order to identify how to best model the effect of the chemotherapeutic on cells over time. In addition, this learned drug action may allow us to learn more about the underlying biological mechanism. For example, symbolic regression can be applied to the functions learned by UPINNs to derive a closed-form expression for the identified nonlinearities. Additionally, one can compare whether the nonlinear functions discovered by UPINNs provide a better fit to the data than corresponding linear models. If the nonlinear model demonstrates higher accuracy, this would suggest the presence of intrinsic nonlinearities in the hidden term that cannot be effectively represented by a linear combination of equations 3.7 and 3.8. Finally, the learned drug action can then be used for downstream tasks such as drug treatment schedule optimization. Current limitations of this method is a lack of uncertainty quantification, which is especially important given the method's stochastic nature and potential high-risk downstream applications, lack of integration with model identifiability workflows, and minimal quantification of DE setups where it is unlikely to perform well.

This work sheds some light on identifiability issues with UPINNs. It is clear that identifiability problems can arise to due "structural" identifiability issues, such as where the inverse problem and neural network setup does not admit a single unique set of hidden term representations that fit the data. Arguably, given noisy data (or even finite noiseless data) there is no guarantee that the hidden term can be successfully learned. These questions should be investigated more rigorously in future work. Before training a UPINN, some analysis of the "structural" identifiability of the hidden terms should be performed. In the case of nonindentifiability at this stage, in some cases the range of the output of the hidden term NNs can be constrained to remove the unidentifiability. Furthermore, there is a different identifiability problem which concerns the optimization itself. As can be seen in the results of this chapter, insufficient minimization of the ODE loss and MSE loss combination can make it difficult for the framework to recover the hidden term at points when the estimated derivatives are close to zero. Possible mitigations of this should be investigated in future work as well. In particular, improved optimization or additional regularization of the hidden term NNs could help recover the hidden terms more accurately.

General future work in this direction would involve understanding the conditions under which this method performs well and when it is not able to learn the drug action effectively (e.g. noise level, shape and properties of the ground truth function). If there is more than one function to be learned, understanding the identifiability of the functions would provide

more guidance on whether the method can successfully recover the ground truth. This could be a mathematical analysis similar to parameter identifiability, or it could be a more nuanced quantification of the uncertainty of the output of the NNs, taking into account the abundance of data. Uncertainty quantification has been proposed for PINNs [179], but it would be especially helpful to quantify the uncertainty for UPINNs. Additionally, it would be beneficial if the uncertainty quantification method has validity guarantees, such as conformal prediction [8]. Finally, applying the method to a drug with a known model of drug induced death (e.g. Norton-Simon) would further validate the method's performance. As an additional step after the functions have been learned, symbolic regression could be performed to find a closed form of the function.

As an extension to learning the dynamics of doxorubicin, the UPINN method can be easily adapted to learn the unknown net proliferation rate $G(t, D)$ as a function of dosage and time (and possibly cells) simultaneously. This would involve treating the number of cells as a function of both dosage and time as well. However, there are several considerations: due to the limited dosage data (only 9 measurements), interpolation between these measurements may not yield good results. However, it should be possible to fit $G(t, D)$ correctly for the available dosages. Secondly, there may be identifiability issues and there may not be a single unique G which satisfies the equation.

3.6 Conclusion

In this chapter, we integrate ML in the form of the UPINN method with QSP models in order to learn the drug action of chemotherapeutics accurately and with minimal computational expenses. We showcase the ability of the method to identify three different well-known drug actions, the Log-kill model, Norton-Simon, and E_{max}. The learned drug actions match the ground truth very well. This method can also be used to infer many parameter sets simultaneously rather than running a separate fitting procedure for each dataset. In addition, the method can interpolate between fitted parameters. Finally, we employ the method to learn the drug action of doxorubicin from time-series data.

Chapter Acknowledgements

We thank Matthew McKenna (Vanderbilt University), Thomas Yankeelov (The University of Texas at Austin), and Ernesto Lima (The University of Texas at Austin) for sharing their data and for informative discussions and comments on the first draft of the manuscript.

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

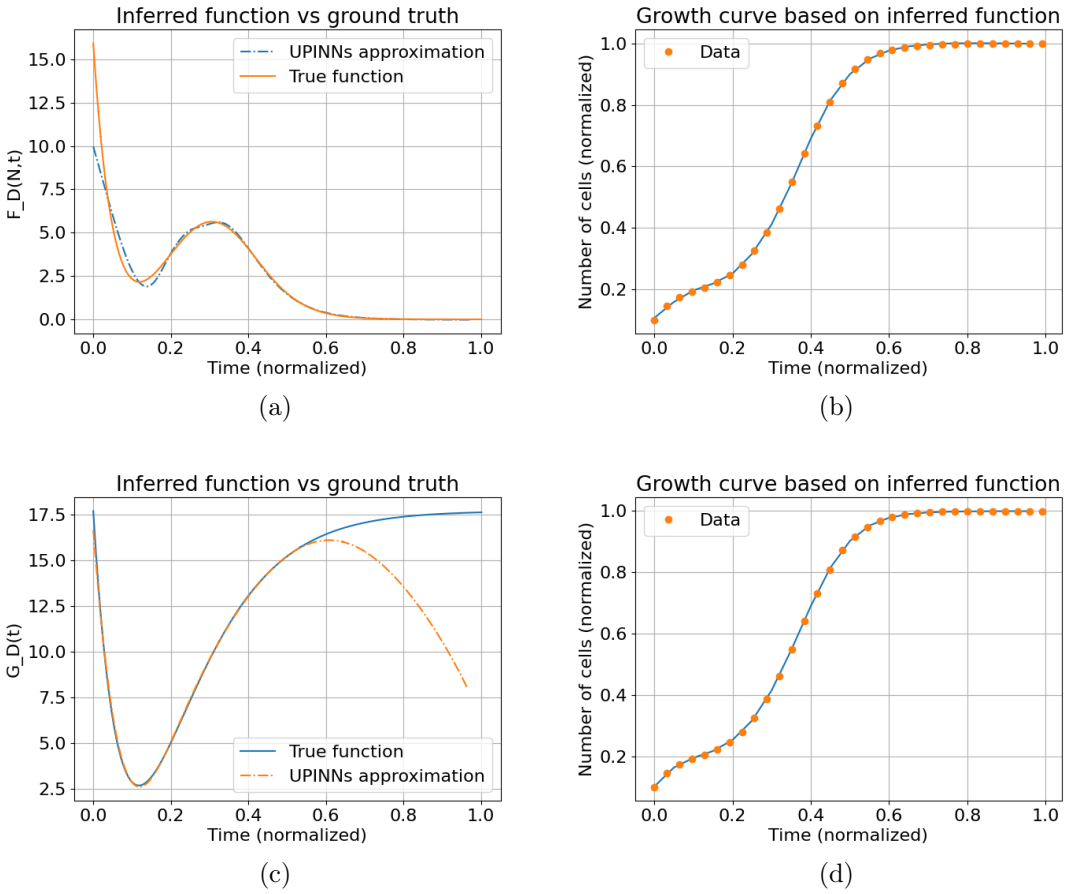
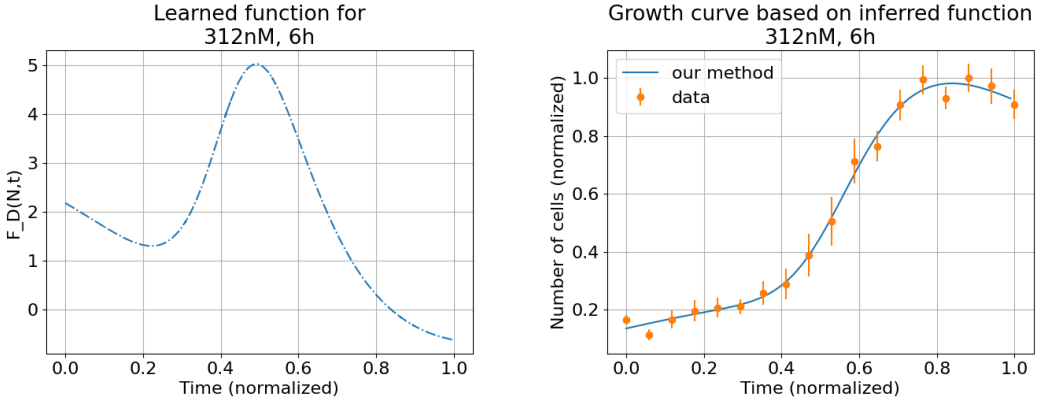


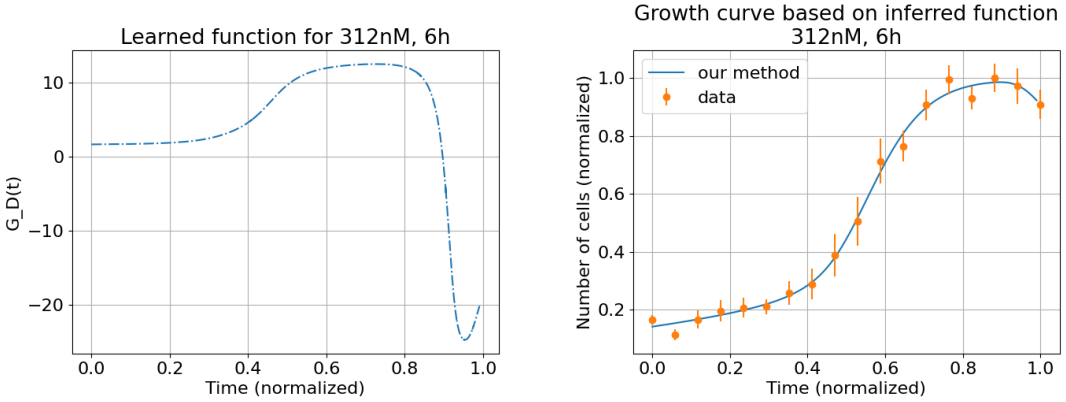
Figure 3.8: **(a)** Fit of $F_D(N,t)$ (eq (3.9)) to data from row 2 of Table 3.5 (eq. 3.8, parameters $r = 0.017, k_{d,B} = 0.03$) **(b)** ODE solution generated using the inferred $F_D(N,t)$, along with the data for comparison **(c), (d)** Same as (a), (b) but $G_D(t)$ is fit. F_D in (a) is recovered with a solution MSE of 10^{-6}, but G_D in (c) appears to not be fully identifiable from the data.

$$\frac{dN_{TC}(t)}{dt} = F_D(N,t)N_{TC}(t)$$



(a) (b)

$$\frac{dN_{TC}(t)}{dt} = G_D(t)N_{TC}(t)\left(1 - N_{TC}(t)\right))$$



(c) (d)

Figure 3.9: **(a)** Fit of $F_D(N,t)$ (eq (3.9)) to experimental data: 312nM, 6h exposure **(b)** ODE solution generated using the inferred $F_D(N,t)$, along with the data for comparison **(c), (d)** Same as (a), (b) but $G_D(t)$ (eq (3.10)) is fit. The overall curve of (a) is similar in shape visually to Fig. 9 (a), indicating that the growth is captured to some extent by Model (3.6). Additionally, (c) shows a plateau in the function before an increase in cell killing, which cannot be captured by a linear combination of (3.7) and (3.8).

Duration	Concentration	Best MSE	Mean MSE	St. Dev. of MSE
6h	10nM	0.00045	0.0022	0.0036
6h	20nM	0.00031	0.0011	0.0013
6h	39nM	7.57e-05	0.024	0.062
6h	78nM	0.00050	0.0013	0.00054
6h	156nM	0.00082	0.00084	8.04e-06
6h	312nM	0.00019	0.00086	0.00049
6h	625nM	0.00061	0.010	0.016
6h	1250nM	0.00038	0.0072	0.0065
6h	2500nM	0.0047	0.0062	0.00097
12h	10nM	0.00022	0.0011	0.0015
12h	20nM	0.00066	0.0021	0.0015
12h	39nM	0.00056	0.0011	0.00020
12h	78nM	0.00084	0.00086	7.50e-06
12h	156nM	0.00079	0.0021	0.0032
12h	312nM	0.00011	0.021	0.036
12h	625nM	0.00043	0.012	0.033
12h	1250nM	0.00086	0.0076	0.0031
12h	2500nM	0.0014	0.011	0.0098

Table 3.6: Fits of $F_D(N,t)$ to in-vitro SUM-149 time-series data from [111].

Duration	Concentration	Best MSE	Mean MSE	St. Dev. of MSE
24h	10nM	0.00050	0.00065	0.00013
24h	20nM	0.00067	0.0014	0.00097
24h	39nM	0.00092	0.0011	0.00010
24h	78nM	0.00043	0.00098	0.00024
24h	156nM	0.00062	0.011	0.019
24h	312nM	0.00031	0.0023	0.0040
24h	625nM	0.00033	0.0063	0.0038
24h	1250nM	0.0048	0.0048	6.66e-05
24h	2500nM	0.0045	0.0078	0.0013

Table 3.7: Fits of $F_D(N,t)$ to in-vitro SUM-149 time-series data from [111] (continued from Table 3.6).

Chapter 4

Conformalized Physics-Informed Neural Networks

Chapter Summary

PINNs are an influential method of solving DEs and estimating their parameters given data. However, since they make use of NNs, they provide only a point estimate of DE parameters, as well as the solution at any given point, without any measure of uncertainty. Ensemble and Bayesian methods have been previously applied to quantify the uncertainty of PINNs, but these methods may require making strong assumptions on the data-generating process, and can be computationally expensive. Here, we introduce Conformalized PINNs (C-PINNs) that, without making any additional assumptions, utilize the framework of conformal prediction to quantify the uncertainty of PINNs by providing intervals that have finite-sample, distribution-free statistical validity.

4.1 Introduction

PINNs are an influential method developed by [140], which employs NNs to solve a DE (forward problem) as well as estimate its parameters (inverse problem). The DEs may be ODEs or PDEs. PINNs have been extended in many ways (e.g. UPINNs by [128] for learning functions, XPINNs by [77] for domain decomposition, and DeepONets by [104] for operator learning), in order to solve increasingly complex problems. However, many of these methods produce only point estimates of the DE solution. This makes it difficult to

apply these methods in any practical situation (e.g. scientific discovery) where the error of the solution needs to be quantified or bounded.

UQ methods for PINNs have been developed in order to fill this gap, such as Bayesian PINNs in [179]. This method involves modelling the uncertainty of model and DE parameters using Bayesian statistics. However, this approach has several significant drawbacks. An appropriate prior distribution on the NN and DE parameters must be determined in advance, and in the case of model misspecification at this stage, the resulting uncertainties may not be calibrated, as shown by [60]. Secondly, as described in [179], the noise distribution within the data needs to be assumed in advance. In [179], a Gaussian noise model is assumed, but may not be an appropriate model in all cases.

Conformal prediction (CP), as introduced by [166], is a user-friendly framework for creating uncertainty intervals on predictions made by any ML model, including ones that are black-box or not inherently interpretable. Its distinctive feature lies in the ability to provide intervals that have finite-sample, distribution-free statistical validity. This implies that the method offers explicit and non-asymptotic guarantees without relying on specific distributional or model-related assumptions. This makes conformal prediction a versatile tool that can be applied to any model, including NNs. The most common flavor of CP methods is the Split Conformal method as outlined by [122], which splits the otherwise training set into training and calibration sets; it then uses the former to train a point predictor and the latter to construct uncertainty intervals.

Here, we introduce Conformalized PINNs (C-PINNs) and show how they can quantify the uncertainty of two common PINN use cases:

- The inferred DE solution (the forward problem).
- Estimated DE parameters (the inverse problem).

We also show how a procedure that utilizes the strong law of large numbers and an underlying property of Split Conformal methods can be used to *rigorously* check the *exact* coverage validity of C-PINN intervals.

4.2 Background

UQ (UQ) for ML (and physics-informed ML) models takes multiple forms, such as ensemble models [139], Bayesian NNs [83], MC Dropout [63], and conformal prediction [8]. The

simplest of these is ensemble models [139], where multiple versions of the same ML model (with differing seeds and initializations) are trained. The predictions are aggregated and depending on the variance of the predictions, confidence intervals on the outputs can be constructed. A downside of the ensemble method is that it can be computationally expensive. Another prominent method is that of Bayesian NNs, and its physics-informed extension B-PINNs [83, 179]. This method treats weights in the network as probability distributions rather than values. Using Bayes' rule, the output acquires a natural UQ through these probability distributions. However, a downside of this method is that priors still have to be assumed on the parameters (and could be assumed incorrectly). Bayesian NNs can also be computationally expensive to train. MC Dropout [63] involves using inference-time dropout to introduce stochasticity into the predictions of an ML model. While simple to implement and less computationally expensive than deep ensembles, there are no guarantees that the uncertainties will be well-matched to actual error rates [163]. In physics-informed ML, many of the general UQ methods are applicable.

Conformal prediction [8] is a method of UQ that leverages held-out data (which comprises the *calibration set*) to construct confidence intervals with provable guarantees on the coverage. None of the previously discussed models have provable guarantees on the coverage, which makes this method particularly valuable for high-stakes predictions, which is a common application in scientific discovery. In our work, we employ split conformal prediction as per [122] in order to have trained models output both a point estimate $\hat{y}$ and a prediction interval Γ_α with confidence level α. This technique requires the use of a holdout *calibration set* in addition to the training set D_{train}. The calibration set D_{cal} is used to construct the intervals, but not used to train the model. In this sense it is analogous to a validation set. The procedure for constructing such an interval with confidence level α is as follows:

1. Train a model that outputs $f(x_i) = \hat{y}_i$ on the training set D_{train}.
2. Define a non-conformity score for the prediction of the model, which is a measure of the discrepancy between the prediction $\hat{y}$ of the model and the true label y. In our case, we will use $R(\hat{y}, y) = |y - \hat{y}|$.
3. Construct the set of nonconformity scores R for all points in the calibration set, D_{cal}.
4. Let $\hat{q}_n$ be the $\lceil (1-\alpha)(n+1) \rceil$ smallest value of R.
5. Then, for a new point (x_{n+1}, y_{n+1}), the prediction interval $[\hat{y} - \hat{q}_n, \hat{y} + \hat{q}_n]$ is guaranteed to contain the true label y_{n+1} with probability $p \in [1-\alpha, 1-\alpha+1/(n+1))$

The coverage guarantee is proven in [122], so the proof is omitted.

In our work C-PINNs, where we bridge conformal prediction and PINNs, we were the first to do so for both the forward and inverse problem [131]. Additionally, we show that the resulting confidence intervals have provable coverage guarantees even when the solution is discontinuous.

4.3 Methodology

For the forward and inverse problems, we consider the logistic growth ODE. This equation is widely used to model logistic growth as described by [88, 174] (e.g. in a context where cells (N) are growing over time (t) but are limited by the nutritional value of the medium):

$$\frac{dN}{dt} = \beta N(1-N); \quad N(0) = N_0 \ . \tag{4.1}$$

For the forward problem only, we also consider the 1D Buckley-Leverett Equation [169]:

$$\begin{cases} u_t - f(u)_x = 0, x \in [-1,1], t \in [0,1] \ , \\ u(0,x) = \begin{cases} -3 & \text{if } x < 0 \\ 3 & \text{if } x > 0 \ , \end{cases} \end{cases} \tag{4.2}$$

taking $f(u) = \dfrac{4u^2}{4u^2 + (1-u)^2}$ [96] showed that a continuous PINN was not able to successfully learn the solution to the equation, likely due to the discontinuous nature of the solution. To see if conformal prediction can still be used to construct intervals with valid coverage when PINNs do not approximate the solution well, we conformalize the forward problem for eq. 4.2 in Section 4.3.1.

4.3.1 Conformalizing the PINN surrogate solution (forward problem)

In the following subsections, we conformalize the forward problem for the logistic growth ODE and a PDE with a discontinuous solution, the 1D Buckley-Leverett equation.

Conformalizing the logistic growth ODE

First, a numerical solver is used to solve Equation 4.1 for the initial condition $N_0 = 0.1$ and $\beta = 0.05$ for time 0 to 150. A dataset of 150 tuples $\{t_i, N_i\}$ are generated in total. Gaussian noise (noise level 0.08 as per [128]) is added to the solution. These are randomly split into holdout set of points (100), training points (25), and test points (25). Then, a single continuous PINN is fit to the training data as per [140], with the parameter value of β given to the PINN in advance. After solving the forward problem, the PINN returns a surrogate solution to the DE. Since the solution is modelled using a NN, it can be evaluated at any timepoint, not just at the datapoints.

We implement split conformal prediction as per [122] on the solution generated by PINNs. Given a prespecified miscoverage level $\alpha \in [0, 1]$, this procedure creates a $(1-\alpha)$-confidence interval for a new test point t_{test} which will contain the true label y_{test} with probability $P \in [1-\alpha, 1-\alpha+\frac{1}{(n+1)}]$ as derived in [8]. This means the constructed interval has *valid coverage*. The 90% and 50% confidence intervals (with miscoverage levels $\alpha = 0.1, 0.5$ respectively) are guaranteed to contain at least 90% and 50% of the true labels respectively.

In order to create a prediction interval (of confidence $1-\alpha$) for a new test point t_{test}, we split the holdout set of 100 tuples $\{t_i, N_i\}$ into $n_c = 80$ calibration points and $n_v = 20$ validation points. We then evaluate a non-conformity score for each calibration point. To create the scores, for each tuple (t_i, N_i) of the calibration points, we evaluate the absolute error between each true N_i, and the corresponding predicted $\hat{N}_i$. With miscoverage value $\alpha = 0.1$ (and corresponding confidence level 90%), we calculate the $\hat{q}_\alpha = \lceil (1-\alpha)(n_c+1) \rceil / n_c$th quantile of the calibration non-conformity scores. For a new prediction $\hat{y}_{test}$, we can use the interval $\hat{y}_{test} \pm \hat{q}_\alpha$ as the 90% confidence interval. Any interval generated this way via split conformal prediction is proven to have valid coverage as per [8].

We repeat this procedure for both noiseless and noisy (noise level 0.08) data. In order to generate noisy data, we add Gaussian noise proportionally to the mean of the data.

Conformalizing a PDE with a discontinuous solution (Buckley-Leverett Equation)

We conformalize the forward problem as before for the Buckley-Leverett equation. A PINN is used to solve eq. 4.2 with 25 noiseless training points for $t = 1$. The resulting solution matches the training data, but may not adequately represent the solution near discontinuities at $x = -2.5, 3.0$.

4.3.2 ODE parameter estimate (inverse problem)

In this section, we conformalize the inverse problem. Suppose that we are tasked with providing a valid interval over the DE parameter $\hat{\beta}_{test}$ for a new dataset D_{test}. In order to conformalize the parameter estimate (which does not have a ground truth label), we need to compare the PINN's estimate $\hat{\beta}_{test}$ to a held-out calibration set of parameter estimates. However, we do not have this set and we must generate it. For this reason, we will assume that we are given a distribution over possible datasets D_i. In the simplest form of this assumption, which we explore here, the given distribution is a distribution over possible β, which then induces a distribution over possible datasets D_i. Then, the C-PINN can be used to construct a valid interval over the estimate $\hat{\beta}_{test}$ for D_{test} by generating additional datasets with their true parameters β, and inferring $\hat{\beta}$. We outline the procedure to do this, and show that the method has valid coverage. We note that the method may be extended in several different ways to predicting a vector of parameters, which is an interesting avenue for future work.

In order to conformalize a new $\hat{\beta}_{test}$ estimate, we generate holdout pairs $\{\beta_j, D_j, \hat{\beta}_j\}$ (sampled true β_j, generated datasets D_j using β_j, inferred $\hat{\beta}_j$ of D_j), shown in Fig 4.4(a). We sample β 1000 times from the given distribution, which we assume is uniform in $[0, 0.5)$. We then solve Eq 4.1 via an ODE solver for these values of β, generating 10 points $\{t_i, N_i\}$ per dataset. We obtain a set of 1000 tuples $\{\beta_j, D_j = \{t_i, N_i\}_j\}$. Then, for each D_j, we employ a standard continuous PINN to estimate the parameter β_j in Eq 4.1. This yields one $\hat{\beta}$ estimate per dataset. Since each PINN can only infer β after being trained on a specific dataset, we must use a separately initialized PINN to find β for each dataset. We explore alternative model options in the Section 4.5.

We perform split conformal prediction using the previously generated holdout set. For a new dataset D_{test}, we first use a PINN to estimate $\hat{\beta}_{test}$ for the dataset D_{test}. Then, we sample 800 points from the holdout set $\{\beta_j, \hat{\beta}_j\}$ to form the calibration set, and construct an 80% confidence interval as per Section 4.3.1. The interval given by the C-PINN will have valid coverage as long as β_{test} in fact comes from the given distribution (Uniform$[0, 0.5)$ in this case).

4.3.3 Model architecture and training

The PINN was implemented in PyTorch ([123]) similarly to https://github.com/jayroxis/PINNs. The NN that was used to model the surrogate solution contains one input, 2 hidden layers with 10 hidden units each, and one output. The activation function used between layers is

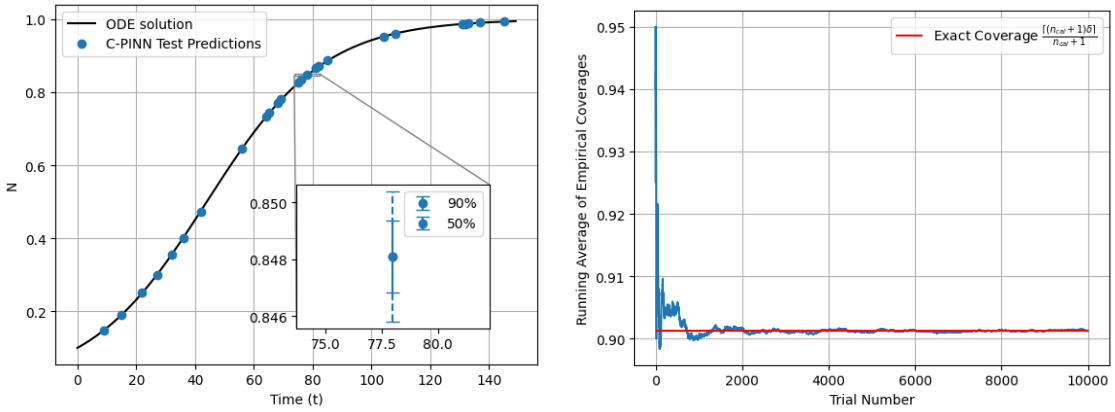
tanh. In order to train the PINN, the Adam ([87]) optimizer (with default parameters and linear annealing learning rate scheduling) was run for 100 epochs, and then the L-BFGS optimizer was run until convergence.

4.4 Results

In this section, we detail the results (and verify the guarantees) of conformalizing the forward problem (Section 4.4.1) and the inverse problem (Section 4.4.2).

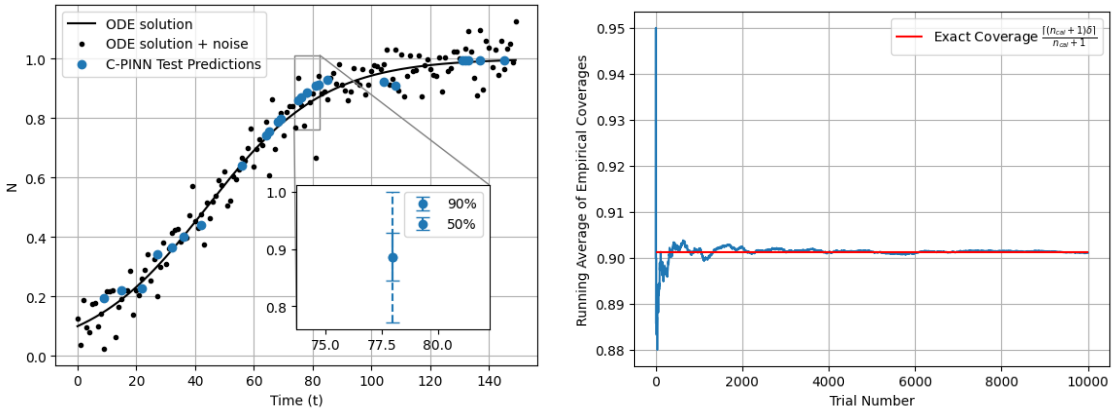
4.4.1 Forward problem

For the logistic growth ODE, we plot the coverage intervals for both noisy (Fig 4.2(a)) and noiseless data (Fig 4.1(a)). The procedure yields labels that are centered on the point estimate, but the 50% and 90% confidence intervals are much wider for the noisy solution. The predicted 90% and 50% confidence intervals are smaller for the noiseless data since the PINN is better able to predict N_{test} on an unseen test point t_{test}.



(a) (b)

Figure 4.1: (a) The performance of a single PINN on non-noisy data. (b) Using the PINN and noiseless data from Fig 4.1(a), coverage estimated through repeated construction of an interval, for different calibration and validation sets. The running average coverage is computed over 10000 trials.



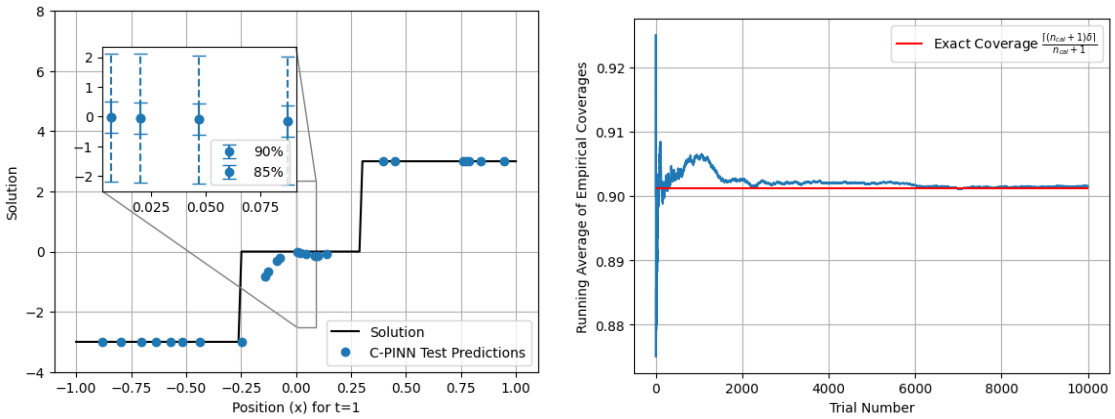
(a) (b)

Figure 4.2: (a) The performance of a single PINN on noisy data (noise level 0.08). (b) Using the PINN and noisy data from Fig 4.2(a), coverage estimated through repeated construction of an interval and validation, for different calibration and validation sets.

Additionally, we empirically show that the 90% interval has valid coverage in the noisy and noiseless regimes (Figs 4.1(b), 4.2(b)). To do this, we perform split conformal prediction over many different splits of the validation and calibration points. This procedure is outlined in [8]. After constructing the interval using the calibration set, we check how many of the validation points in fact fall in the constructed interval. [8] also shows that for n_c calibration points and miscoverage level α, the coverage of the split conformal interval is exactly $\lceil (1-\alpha)(n_c+1) \rceil / (n_c+1)$. Figures 4.1(b), 4.2(b) show that the mean coverage converges to exactly this coverage over 10000 random splits (trials), which implies that the constructed intervals are valid.

For the Buckley-Leverett equation, Fig 4.3(a) shows the true and predicted solution, along with the 90% and 85% confidence intervals. However, since the model only performs poorly at discontinuities, adaptive intervals would be more informative. The coverage is still as expected despite the PINN's poor performance in recovering the true solution (Fig 4.3(b)).

Hence, conformal prediction can be used to quantify the uncertainty and fit of a PINN solution even when there is only a single dataset available (with significant noise). Furthermore, the method yields valid coverage, which we have verified rigorously. Hence, this method can be used to quantify the performance of the PINN on unseen test data, which is a good measure of the generalization of the solution.



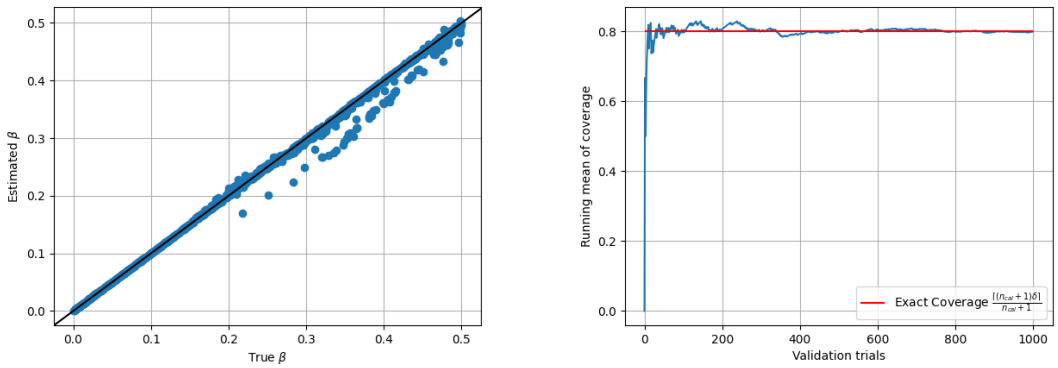
(a) (b)

Figure 4.3: (a) The performance of a single PINN in approximating the Buckley-Leverett equation (eq. 4.2). (b) Using the PINN and data from Fig 2a, coverage estimated through repeated construction of an interval and validation, for different calibration and validation sets.

4.4.2 Inverse problem

In Fig 4.4(a), we can see that the PINN is quite adept at recovering the true β from the dataset from only 10 noiseless datapoints. The mean interval coverage converges to the expected coverage (Fig 4.4(b)) over many validation points (β_{test}, D_{test}). To compute the coverage, we repeatedly sample a new $\{\beta_{test}, D_{test}\}$ with $\beta \sim \text{Uniform}[0, 0.5)$ and compute the proportion of the time that β_{test} falls in the constructed interval. We note that for a smaller holdout set, different confidence level, and for a different distribution of β, the coverage would still be valid due to exchangeability, as discussed in [8]. We repeat this experiment with noisy data (Fig 4.5(a)) and obtain similarly valid coverage, since each dataset D_i is still sampled i.i.d. We note that for noisy data, the PINN is less accurate in inferring β for higher values, and as such would benefit from adaptive conformal prediction. We obtain valid coverage upon repeatedly splitting the holdout set into validation and calibration points for both noiseless and noisy data, as per Section 4.3.1 (Figure 4.6 and 4.7).

When conformalizing the inverse problem, in addition to testing our method on repeated i.i.d. validation samples as described in Section 4.3.2, we also show valid coverage using the averaging method described in Section 4.3.1.



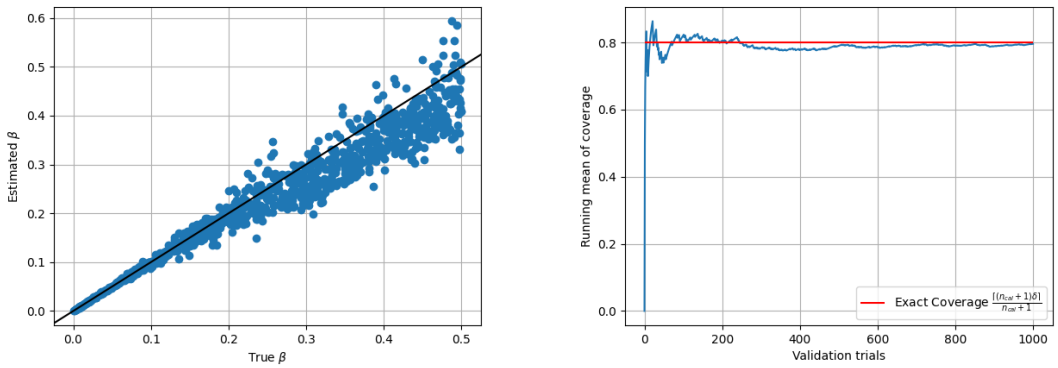
(a) (b)

Figure 4.4: (a) Plot of the inferred $\hat{\beta}$ vs. true β for 1000 datasets generated from sampled values of β (noiseless data) (b) Estimated coverage using the holdout set of $\{\beta, \hat{\beta}\}$ from Fig 4.4(a).

4.5 Conclusion and Discussion

In this work, we conformalize PINNs in two ways: first, we conformalize the surrogate solution of the logistic growth ODE and the Buckley-Leverett PDE (Eqs. 4.1 and 4.2, forward problem) using a holdout calibration set and split conformal prediction as per [122]. In order to validate the method, we employ an averaging approach to compute the coverage of the intervals over many calibration/validation splits. Even when the data is significantly noisy, we show that the empirical coverage converges to the theoretical coverage. Secondly, we conformalize one inferred parameter (Eq 4.1, inverse problem). We use the parameter estimates from many PINNs to conformalize the parameter estimate for a new dataset. The resulting interval provides valid coverage for a new, unseen value of β. In order to reduce the computational cost of this method, we note that it is possible to use UPINNs as introduced by [128] to infer β for several datasets simultaneously.

We note several limitations and extensions of the introduced method. Firstly, both forward problem and inverse problem conformalization for PINNs would benefit greatly from adaptive intervals, which were mentioned previously. Adaptive conformal prediction is a method that uses estimates of feature-dependent variance (e.g. estimates of the variance of the PINN solution at different timepoints) as the basis for constructing adaptive intervals with coverage guarantees. Hence, instead of creating an interval that is the same width



(a) (b)

Figure 4.5: (a) Plot of the inferred vs. true β for 1000 datasets generated from sampled values of β (data noise level is 0.03) (b) Estimated coverage using the holdout set of $\{\beta, \hat{\beta}\}$ from Fig 4a.

for all inputs of the model, its length would differ based on the local variance estimate. Additionally, for the inverse problem, the conformalization procedure that we introduce depends on an assumption on the data distribution and the parameter. This is a significant limitation of the method—if this distribution is chosen incorrectly then the intervals will have the expected coverage only under those specific assumptions. Future work might look into how to relax this assumption, or how to best integrate real-world knowledge and assumptions into the conformalization procedure. Finally, it is worth noting that approximate Bayesian computation (ABC), which already benefits significantly from ML based summary statistics [1], provides a natural framework for generating data and inferring parameters. Integrating ABC with conformal prediction has also been recently explored [15]. Another possibility for an extension to this work is to conformalize UPINNs [128] using split conformal prediction. Recent works such as Conformalized DeepONets [115] have been integrating conformal prediction with other physics-informed ML approaches, which shows that this area is quickly gaining traction.

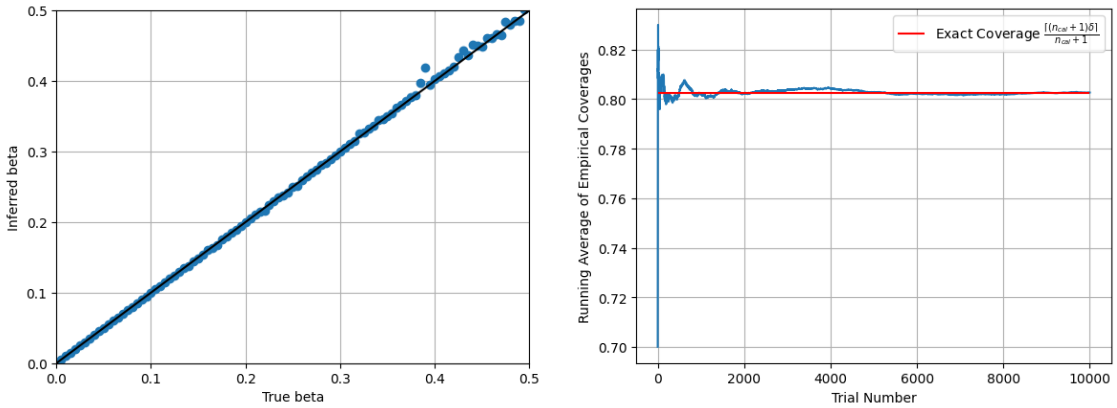


Figure 4.6: (a) Plot of the inferred vs. true β for 100 different values of β, equispaced from 0 to 0.5. The data is noiseless. (b) Using the β estimates from noiseless data, coverage estimated through repeated construction of an interval, for different calibration and validation sets. The running average coverage is computed over 10000 trials.

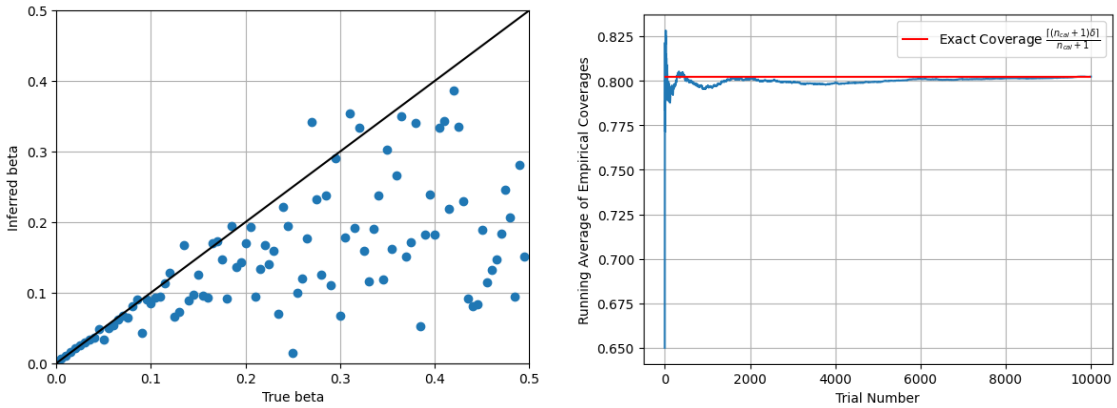


Figure 4.7: (a) Plot of the inferred vs. true β for 100 different values of β, equispaced from 0 to 0.5. The data is noisy with noise level 0.08. (b) Using the β estimates and noisy data, coverage estimated through repeated construction of an interval, for different calibration and validation sets. The running average coverage is computed over 10000 trials.

Chapter 5

Enhancing Symbolic Regression with Dimensional Analysis

Chapter Summary

In engineering and applied mathematics, developing accurate mathematical models to predict and understand real-world phenomena is of utmost importance. Symbolic regression is a useful ML-based tool to fit models but it can be computationally expensive. We present a new method for enhancing symbolic regression for DEs via dimensional analysis, specifically the Buckingham Π theorem and Ipsen's method. Since symbolic regression often suffers from high computational costs and overfitting, non-dimensionalizing datasets reduces the number of input variables, simplifies the search space, and ensures that derived equations are physically meaningful. As a first step, we combine dimensional analysis with the PySR symbolic regression algorithm to show that dimensional analysis improves the accuracy of recovering algebraic equations. The results demonstrate that transforming data into a dimensionless form significantly improves the training and test error of the symbolic expressions found. Then, as our main contribution, we perform nondimensionalization guided by Ipsen's method. We then integrate the nondimensionalized equation with UPINNs and symbolic regression, for the recovery of the unknown term when a DE is only partly known. We find that symbolic regression is able to better recover the unknown term after nondimensionalizing the data. These findings suggest that integrating dimensional analysis with symbolic regression can significantly lower computational costs and increase accuracy, providing a robust framework for automated discovery of governing equations in complex systems when data is limited.

5.1 Introduction

In many engineering disciplines, such as aerodynamics, heat transfer, and materials science, developing accurate, interpretable mathematical models is crucial for control and optimization. Engineers often rely on a combination of empirical data and physical laws to build such models [20, 74, 175]. However, deriving governing equations from raw experimental data can be difficult, especially when the underlying relationships are nonlinear or partially unknown. Symbolic regression has gained popularity as a method for discovering mathematical models directly from data [101, 151, 158, 159]. Unlike traditional regression, which assumes a specific model form, symbolic regression searches for the best-fitting closed-form parsimonious equation for the dataset without predefined assumptions. Symbolic regression also has the advantage of being more interpretable than model-fitting via NNs [56]. The flexibility of symbolic regression algorithms makes them powerful for uncovering hidden relationships in complex datasets. However, the method's exhaustive search over possible equations can be computationally expensive, leading to slow performance and the risk of overfitting. Symbolic regression is generally an NP-hard problem [164].

Dimensional analysis, a widely used technique in mathematics and engineering [58], simplifies algebraic or DEs by reducing the number of variables involved. This allows the true underlying equations and hidden parameters to be recovered with higher accuracy and with less computational overhead. It also ensures that the recovered equation adheres to the physical laws. Dimensional analysis can be performed using different methods e.g. manual non-dimensionalization, Buckingham Π theorem [148], and Ipsen's method [40]. Manual non-dimensionalization involves scaling each dimensional variable by a hand-chosen value to remove its units, then substituting the non-dimensional variables for the dimensional ones. This removes the units but may not reduce the number of free parameters. The Buckingham Π theorem introduces an automated procedure to minimize the possible number of dimensionless constants (π terms) that the equation can be expressed with. Although there are many choices of sets of π terms, it is difficult to directly control the final equation, although it will have the minimal number of free variables. Ipsen's method is very similar to the Buckingham Π theorem, but enables straightforward user selection of the dimensionless constants. This method is useful when one is interested in reducing (or maximizing) the number of free variables that appear in a specific term in the equation. Additionally, within mathematical modelling for engineering applications, it is common to reduce a problem to a similar but simpler formulation [58, 106, 144].

Symbolic regression algorithms have been built using various approaches such as genetic programming, sparse regression, NNs and language models [11, 41, 85, 159], with some works [101, 151, 158] integrating dimensional analysis into their workflow. However, none of

these approaches integrate dimensional analysis in a partial physical knowledge setting, and systematically show the effect of nondimensionalization on a symbolic regression algorithm such as PySR [41]. As a preliminary proof of concept, we test that dimensional analysis in fact improves the performance of symbolic regression. Then, we propose a novel method for increasing efficiency and accuracy of symbolic regression under partial-knowledge by first non-dimensionalizing the data. We stay in a regime where some terms in a DE are known, and some need to be learned, which is common in many engineering and mathematical modelling applications. Ipsen's method of nondimensionalization allows for the fine control of the variables that appear in different components of the DE. Then, we apply UPINNs [92] to learn individual unknown terms in an equation. This approach allows the user to reduce the number of input variables to the unknown components while ensuring the dimensional consistency of expressions. Due to nondimensionalization, our strategy also potentially avoids calculations involving numbers of very high or low orders of magnitude. The true underlying expression can be found by undoing the nondimensionalization.

By integrating dimensional analysis with any symbolic regression workflow, we can significantly enhance the applicability of data-driven discovery in engineering. The symbolic search space becomes significantly smaller because candidate expressions only need to be constructed from dimensionless groups rather than the original variables, and dimensionally inconsistent expressions are automatically eliminated from consideration. In turn, the reduced search space decreases the risk of overfitting and spurious symbolic relationships. Our contributions in this Chapter are:

- Demonstration of how the Buckingham Π theorem can be combined with any symbolic regression algorithm. Specifically, when paired with PySR [41] symbolic regression, this combination recovers the correct expressions with greater accuracy.
- Introduction of a novel pipeline that integrates dimensional analysis with UPINNs [92] and symbolic regression. This pipeline can be used in many fields of mathematical modelling where some physics is known and some needs to be learned, such as engineering, materials science, and computational biology.

We note that unlike statistical compression methods such as PCA or latent-variable embeddings, dimensional analysis preserves dimensional consistency, physical interpretability, and invariant relationships between quantities. In our framework, the objective is not to simply reduce the dimensionality of the dynamical system, but rather to reduce the number of independent physically admissible inputs entering the unknown term. This constrains the symbolic regression search space in a physically meaningful manner while allowing the recovered expressions to be mapped back to the original governing equations.

5.2 Background

In this section, we describe the problem of symbolic regression and previous work in the area (Section 5.2.1), the Buckingham Π method of nondimensionalization (Section 5.2.2), and Ipsen's method of nondimensionalization (Section 5.2.3), including how it differs from the Buckingham Π method and its advantages. For an overview of UPINNs, please refer to Chapter 2, Section 2.3 for a detailed review, or Section 3.3.1 for a brief overview of how UPINNs can be applied to ODEs.

5.2.1 Symbolic Regression

Symbolic regression is a powerful technique for discovering mathematical equations and DEs directly from data, integrating various methods to optimize search efficiency and ensure physical coherence of the derived models. Large classifications of symbolic regression techniques include: genetic programming; NN and regression approaches; large language models; reinforcement learning.

Traditional approaches to symbolic regression include genetic programming, brute force [89], Monte Carlo tree search [150], and optimization with hand-crafted features [26]. In genetic programming, equations are considered strings of information much like a gene, and new equations for a particular dataset are generated by "crossing over" the best-fit equations. Notable works that use this method are [11, 110, 137]. However, this method suffers from several drawbacks. Firstly, it is fairly slow compared to other approaches [85]. Secondly, it is difficult to incorporate prior knowledge into the generation. The EPDE framework [105] is an evolutionary algorithm that operates without a predefined library of functions, utilizing basic terms to discover both differential and algebraic equations. It employs a multi-objective optimization approach to select the most effective equations and includes a custom solver that handles noisy data and supports the visualization of the discovery process. SINDy [26] identifies dynamical systems by finding sparse representations of governing equations from data. When integrated with dimensional analysis, SINDy ensures that the equations are dimensionally consistent, which is crucial for maintaining physical accuracy in complex systems.

Approaches based on NNs, such as reinforcement learning and language models, have also been widely used. Large language model-based approaches include works such as [85, 159]. SymbolicGPT uses a decoder-only approach to generate a skeleton first, and then populate constants with real values. The work by [85] follows a similar approach, but generates

equations with constants embedded, skipping the skeleton generation. Although language-model-based generation of equations for symbolic regression has been successful, it can be difficult to directly control or constrain these methods to generate expressions with consistent units, or enforce other constraints. Finally, reinforcement learning based approaches have been widely used by works such as [95, 124, 151] due to their flexibility in choosing the reward function and ability to enforce both hard constraints on the search space, and soft constraints that are formulated via the reward function. [124] train a recurrent NN which creates the expression tree, and outperforms Eureqa [144], one of the gold-standard methods that use this approach.

A number of works integrate dimensional analysis or nondimensionalization within the symbolic regression algorithm [12, 101, 151, 158]. This has been done mainly to ensure that the output of the symbolic regression is physically consistent, and to reduce the number of independent variables in the problem. However, this integration in past works has been limited and is not always tailored to the problem at hand. AI Feynman [158] does incorporate dimensional analysis, but the dimensions for variables are fixed and analyzed automatically, not allowing user input or partial knowledge. Φ-SO [151] incorporates dimensional analysis but does not non-dimensionalize datasets before fitting them, which can be beneficial for both resetting the scale of the data, and reducing the number of variables. SciMED [101] integrates dimensional analysis with symbolic regression and applies it to both algebraic and DEs, but its application is restricted to one-side DEs. No symbolic regression method, to our knowledge, has incorporated nondimensionalization in a partial-knowledge scenario using UDEs [138] or UPINNs [92].

The integration of PINNs with dimensional analysis has been explored to some extent in previous works [136, 178] but has not been applied in a partial knowledge setting such as UPINNs [92]. NDAWL-PINNs use nondimensionalization primarily in the loss function [136]. The work by [178] applies dimensional analysis to a physics problem but does not enforce the DE constraints via a PINNs style loss function. A recent work [34] shows that nondimensionalization improves the efficiency of parameter recovery for PINNs and for SINDy symbolic regression [26]. There has been no work, to our knowledge, integrating UDEs [138] with the Buckingham Π theorem.

5.2.2 Buckingham π Theorem

The Buckingham Π theorem is a fundamental tool in dimensional analysis, used to derive dimensionless quantities when the governing equation is not explicitly known [14, 27]. This theorem reduces the number of parameters in the original equation from n to $n - k$, where

n is the number of variables and k is the number of fundamental dimensions (such as length, mass, time, etc.).

As an example, consider the equation for the potential energy difference due to gravity:

$$U = Gm_1m_2\left(\frac{1}{r_2} - \frac{1}{r_1}\right),$$

where U is the potential energy difference between two points, with dimensions $[ML^2T^{-2}]$, G is the gravitational constant with dimensions $[M^{-1}L^3T^{-2}]$, m_1 and m_2 are the masses involved (dimensions $[M]$ each), and r_1 and r_2 are the distances from each mass to a point of interest (with dimensions $[L]$ each). Using the Buckingham π theorem, the number of variables in the equation can be reduced from six to three. The dimensional matrix $\mathcal{D}$ representing the physical dimensions of each variable in the gravitational potential energy equation is structured as follows:

$$\mathcal{D} = \begin{array}{c} \\ \mathcal{M} \\ \mathcal{L} \\ \mathcal{T} \end{array} \begin{array}{c} \begin{array}{cccccc} U & G & m_1 & m_2 & r_1 & r_2 \end{array} \\ \begin{pmatrix} 1 & -1 & 1 & 1 & 0 & 0 \\ 2 & 3 & 0 & 0 & 1 & 1 \\ -2 & -2 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

Three dimensionless variables (π groups) can be formed as follows:

$$\pi_1 = \frac{U\,r_2}{Gm_1^2} = \frac{[ML^2T^{-2}] \times [L]}{[M^{-1}L^3T^{-2}] \times [M]^2} = 1,$$

$$\pi_2 = \frac{m_2}{m_1} = \frac{[M]}{[M]} = 1,$$

$$\pi_3 = \frac{r_1}{r_2} = \frac{[L]}{[L]} = 1.$$

This choice of π groups is just one possibility, and other combinations could be considered. One common method of deriving these dimensionless π groups is the method of repeating variables [51].

To derive a functional relationship $\pi_1 = f(\pi_2, \pi_3)$, we substitute for m_2 and r_1 in terms of π_2 and π_3 respectively, into the equation for U:

$$U = Gm_1m_2\frac{r_1 - r_2}{r_1r_2} = Gm_1m_2\frac{(r_2\pi_3) - r_2}{(r_2\pi_3)r_2} = Gm_1m_2\frac{r_2(\pi_3 - 1)}{r_2^2\pi_3}\ .$$

Simplifying further with $m_2 = m_1\pi_2$:

$$U = Gm_1(m_1\pi_2)\frac{r_2(\pi_3 - 1)}{r_2^2\pi_3} = Gm_1^2\pi_2\frac{\pi_3 - 1}{r_2\pi_3} \ .$$

Substituting this expression for U into π_1:

$$\pi_1 = \frac{Ur_2}{Gm_1^2} = \frac{Gm_1^2\pi_2\frac{\pi_3-1}{r_2\pi_3}r_2}{Gm_1^2} = \frac{\pi_2(\pi_3 - 1)}{\pi_3} \ .$$

Thus, the relationship $\pi_1 = f(\pi_2, \pi_3)$ is established as:

$$\pi_1 = \frac{\pi_2(\pi_3 - 1)}{\pi_3} \ .$$

5.2.3 Ipsen's Method of Dimensional Analysis

Ipsen's method for non-dimensionalizing equations is an alternative method to Buckingham Π which is used to nondimensionalize an equation [59]. Ipsen's method also yields a set of π terms, just like the Buckingham Π theorem, but this method provides flexibility to choose the dimensional terms precisely, which can be used to minimize the number input variables for specific (unknown) expressions. The original form of the equation can subsequently be easily recovered from the dimensionless terms. Given the dependency of Ipsen's method on specific equations, we detail the application of Ipsen's method to two separate DEs in Section 5.3, and here we provide a general procedure. Another name for this method is the "step-by-step" method of nondimensionalization [40].

Suppose that you have an equation

$$F(x_1, x_2 \ldots, x_k) = 0 \ ,$$

where x_i are variables present in the equation, with specific known dimensions. Ipsen's method of nondimensionalization is an iterative method where at each step, one can choose which dimension to eliminate. The output of the method is a set of dimensionless variables. Suppose that there are three variables, and $x_1 = [M], x_2 = [T], x_3 = [M^2T]$, and that

$$F(x_1, x_2, x_3) = 0 \ .$$

One may let $x_3' = x_3/x_1^2$ and $x_1' = x_1/x_1$ which would now have dimensions of $[T]$ and 1 respectively. Then, one can substitute x_1', x_3' into the equation to yield

$$F(1, x_2, x_3/x_1^2) \ .$$

Similarly, now dividing the second and third term by x_2 yields

$$F(1, 1, x_3/x_2x_1^2) \ .$$

which shows us that the expression F can be nondimensionalized with exactly one term $\pi = x_3/x_2x_1^2$.

5.3 Methods

We first illustrate how we integrated symbolic regression with the Buckingham Π theorem to derive algebraic equations (Section 5.3.1). We use this preliminary investigation to check if nondimensionalization in fact improves the accuracy of PySR [41], a symbolic regression algorithm. Subsequently, to learn unknown components of DEs, we employed a combination of PySR, dimensional analysis, and UPINNs (Section 5.3.2).

5.3.1 Full Discovery of Algebraic Equations

We utilized the open-source model PySR [41] for symbolic regression to predict symbolic expressions of data from 7 different algebraic equations (Tables 5.1, 5.2), each varying in complexity and variable count. We do this both for the traditional formulation of the algebraic equation, and after we nondimensionalize the data with the Buckingham Π theorem. We include Coulomb's law, which is a special case because it reduces to exactly one π term. We do not need to run symbolic regression on an equation which reduces to exactly one π term – instead, the value of the π term can be computed for each row of the data separately. In the case of noiseless data, the value of the π term will be the same for all observations of the original variables. However, when there is noise, these values can be aggregated into a single value e.g. by taking the mean. Since the value of the π term is a constant, the equation can be recovered exactly from this value by undoing the nondimensionalization.

PySR uses an evolutionary algorithm to construct and change expression trees that represent the full expression. Settings such as the functions used (e.g. $\sin, \cos, \log, \exp$), the maximum size of the expression, and the number of algorithm iterations are tunable. We set them to a maximum equation size of 20, number of iterations to 40, and the unary operators to: $\cos(x), \sin(x), \exp(x), \log(x), \sqrt{x}, 1/x$, and the binary operators of addition and multiplication. The settings were chosen based on a brief tutorial on PySR's GitHub page [41], but we note that there could be significant sensitivity to these settings.

First, we assessed the performance of PySR by evaluating its runtime and MSE on a train/test datasets of 10 datapoints each. Both training data and test data were generated for each equation (10 datapoints each), and only the training data was used to fit the PySR model. The test data was used for evaluation only. For each equation, data was generated with all independent variables being sampled uniformly between 0 and 1, except for the exponential decay example. In this case, the data ranges were set to uniform between 1×10^{-23} and 5×10^{-23} for the mass, and the Boltzmann constant $k_B = 1.38 \times 10^{-23}$. The reason for this is to keep the data within a realistic range. The gravitational constant g was set to 9.81 $\mathrm{m/s^2}$ in all cases. The gravitational constant $G = 6.67 \times 10^{-11}$ $\mathrm{m^3 kg^{-1} s^{-2}}$. All other variables were sampled uniformly at random within the previously specified ranges.

Following this, we nondimensionalized the training data using the Buckingham Π theorem and applied symbolic regression to the resulting dimensionless quantities, employing the same evaluation protocols. The objective of this comparative analysis was to determine the extent to which dimensional analysis enhances the performance of symbolic regression, specifically in terms of the accuracy of the equation fits. We note that while PySR was our chosen tool for symbolic regression, alternative algorithms could also be applied in this framework. Additionally, we note that while in our results, the nondimensionalized form of the equations is often not the most common form used by scientists (due to the process being handled automatically, and the specific set of dimensionless π terms being chosen at random), the nondimensionalized equation can be easily converted to the original equation through direct substitution.

In terms of code contribution, we created an automated nondimensionalization framework based on Buckingham Py [51], which is a python-based library to implement automated derivation of dimensionless π terms based on the Buckingham Π theorem. The input is the original equation, and based on the dimensionless terms found by Buckingham Py, we are able to automatically nondimensionalize both an expression and its corresponding data. This was used to automatically process all of the algebraic equations.

5.3.2 Discovery of Differential Equation Hidden Terms

For two distinct types of DEs, first-order and second-order, we explored the effectiveness of UPINNs [92] combined with dimensional analysis and PySR [41] in recovering the true equation under partial knowledge. First, we employed Ipsen's method to find the nondimensionalization that minimizes the number of input variables in the unknown term. Then, we trained a UPINN to recover the term under that partial nondimensionalization. Although Ipsen's method is not strictly necessary in this step, it offers some insight into the set of π terms that minimizes the number of inputs to the unknown term. Finally, the form of the unknown term recovered by UPINNs can be passed to a symbolic regression algorithm such as AI Feynman [158] or PySR [41]. Here, we use PySR as the symbolic regression algorithm. This nondimensionalization is designed to simplify the symbolic regression process, enabling the method to more effectively identify the true unknown term from its UPINN approximation.

We now outline the DE setup and the steps that we perform, in detail, for both the logistic growth DE and the rotating bead DE. Suppose we have a DE which has a known part P and an unknown part Q:

$$0 = P(a_1, a_2, \ldots, a_k) + Q(b_1, b_2, \ldots, b_l) \ ,$$

where a_i and b_j are variables appearing in the known and unknown terms respectively. We assume that either P or Q (or both) contain the spatial and/or temporal derivatives. These variables are assumed to have dimensions, and we assume that the dimensions are known, although some variables may be dimensionless. Additionally, we assume that we know which variables appear in P and Q.

The steps that we perform for each DE are as follows:

1. Note all the variables and their dimensions in the equation.
2. Partially nondimensionalize the equation with guidance from Ipsen's method, and retrieve the set of π terms. How to do this is shown in detail for each equation that we consider, but generally is a process dependent on the specific equation.
3. Transform existing data to the nondimensional form (or, in the case of synthetic data, generate already dimensionless data), using the dimensionless π terms from Step (2).
4. Recover the unknown term $Q(\pi_1, \pi_2, \ldots, \pi_k)$ using UPINNs, as a function of only the π terms that appear in it. This will be known in advance from the nondimensionalization.

5. Apply symbolic regression to find a closed form for $Q(\pi_1, \pi_2, \ldots, \pi_k)$, or the unknown term as a function of the dimensionless variables.

6. Optionally, the recovered equation in terms of dimensionless variables can be converted back to an equation with the original variables.

Aside from following these steps, we also compare this method to not performing the nondimensionalization, and inferring the unknown term directly from the data. In this case, here are the steps:

1. Recover the unknown term $Q(b_1, b_2, \ldots, b_l)$ using UPINNs as a function of the original variables that appear in Q.

2. Apply symbolic regression to find a closed form for $Q(b_1, b_2, \ldots, b_l)$, or the unknown term as a function of the dimensionless variables.

3. Optionally, the recovered equation in terms of dimensionless variables can be converted back to an equation with the original variables.

The UPINN implementation follows https://github.com/jayroxis/PINNs and is in PyTorch [123]. The surrogate solution and unknown term was each represented by a multilayer perceptron with hidden layer sizes: $(20, 128, 128, 128, 128, 128, 128, 128)$, and tanh activation. We perform 1000 iterations of Adam and then several thousand of iterations of Limited Memory BFGS, until convergence. The PySR settings were the same as those used in the previous section.

We detail the application of our method to the extended logistic growth model (Section 5.3.2), a first-order DE, and the rotating bead model (Section 5.3.2), a second-order DE.

Extended Logistic Growth Model

To demonstrate the effectiveness of our method on a DE, we start with a variant of the logistic growth model, commonly used in computational biology [3]:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{k}\right) + \frac{AN^2}{B^2 + N^2}\ , \tag{5.1}$$

where $N(t)$ are the number of cells over time, t is time, r, k, A, B being constant values. The dimensional matrix is:

$$D = \begin{array}{c} \\ M \\ T \end{array} \begin{array}{c} \begin{array}{cccccc} N & r & k & A & B & t \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \end{bmatrix} \end{array},$$

where M is the dimension mass, and T is the dimension time. Here, M represents an abstract population-count dimension rather than physical mass. This notation is introduced solely to facilitate dimensional analysis while preserving consistency among the quantities N, K, and B. For this problem, we are interested in recovering the true form of the unknown term $Q(A, N, B) = AN^2/(B^2 + N^2)$. The rest of the DE is assumed to be known. The inputs to the function Q are also assumed to be known.

Using the Buckingham Π theorem to nondimensionalize this equation provides us with seven unique sets of π values. However, it is initially unclear which set would allow the subsequent symbolic regression step to proceed with the highest accuracy in recovering the true unknown term. A π set for this equation which minimizes the number of parameters and variables in the unknown term is the following:

$$\beta = \tfrac{Br}{A} \quad \epsilon = \tfrac{K}{B} \quad \tau = \tfrac{At}{B} \quad \alpha = \tfrac{N}{B} \ .$$

By algebraic manipulation, we can rewrite the original equation in terms of the non-dimensional parameters.

$$\frac{d\alpha}{d\tau} = \beta\alpha\left(1 - \frac{\alpha}{\epsilon}\right) + \frac{\alpha^2}{\alpha^2 + 1} \ . \tag{5.2}$$

When Q is unknown, this equation takes the form of:

$$\frac{d\alpha}{d\tau} = \beta\alpha\left(1 - \frac{\alpha}{\epsilon}\right) + Q^*(\alpha) \ , \tag{5.3}$$

where Q^* is the transformed unknown term.

By contrast, Ipsen's method offers a preliminary insight into the number of parameters each non-dimensional function will have, without the need to know what the unknown terms are. We express the ODE as the sum of P and Q, with the parameters shown in Equation (3):

$$0 = P(r, N, k, t) + Q(A, N, B) \ , \tag{5.4}$$

where $P = rN\,(1 - N/k) - dN/dt$ and Q is the unknown component $AN^2/(B^2 + N^2)$. The goal is to minimize the number of parameters in the unknown function Q. The reason this is useful is because the fewer parameters Q has, the simpler the function is to learn for UPINNs, and the simpler the subsequent equation discovery task is for symbolic regression. Ipsen's method facilitates this by manipulating the number of parameters in each function without altering the overall behavior of the system, using dimensions as a scaling factor.

To illustrate, we show how to apply Ipsen's method in this example.

First, we eliminate B (with dimension $[M]$):

$$0 = P\left(r, \frac{N}{B}, \frac{K}{B}, t\right) + Q\left(\frac{A}{B}, \frac{N}{B}, \frac{B}{B}\right) \ . \tag{5.5}$$

Next, we eliminate A/B (with dimension $[T]$):

$$0 = P\left(\frac{rB}{A}, \frac{N}{B}, \frac{K}{B}, \frac{tA}{B}\right) + Q\left(\frac{AB}{AB}, \frac{N}{B}, \frac{B}{B}\right) \ . \tag{5.6}$$

The method allows us to eliminate terms or variables (B or A/B) from the unknown component by dividing all other variables by each term appearing in Q. This decreases the number of terms in Q at each step. These steps are repeated until the remaining inputs to the functions P and Q are dimensionless. We note that here we can obtain the same set of dimensionless parameters as we obtained earlier with Buckingham Π.

Hence, the set of π terms that minimizes the number of inputs to Q is the above set,

$$\beta = \tfrac{Br}{A} \quad \epsilon = \tfrac{K}{B} \quad \tau = \tfrac{At}{B} \quad \alpha = \tfrac{N}{B} \ .$$

We can nondimensionalize the partially known equation as follows:

1. Nondimensionalize time and N: $\tau = At/B$, $\alpha = N/B$. So $dN/dt = d\alpha/d\tau \cdot A$. Substituting gives:
$$A\frac{d\alpha}{d\tau} = rB\alpha\,(1 - B\alpha/K) + Q(A, N, B) \ .$$
2. Divide both sides by A, which appears in Q and does not introduce new variables into it:
$$0 = -\frac{d\alpha}{d\tau} - \frac{rB\alpha}{A}\,(1 - B\alpha/K) + Q(A, N, B)/A \ .$$

3. The only π term that can appear in $Q^* = Q(A, N, B)/A$ must be $\alpha = N/B$. Hence, the other dimensionless terms in the equation are

$$\gamma = rB/A, \epsilon = K/B$$

and they are in the known term only.

We note that Ipsen's method is useful as a way to verify the process of the nondimensionalization, but it is not strictly necessary in order to perform the nondimensionalization.

Next, we need to create a dataset of tuples $\{A_i, N_i, B_i, \hat{Q}_i(A_i, N_i, B_i)\}$, where $\hat{Q}(A_i, N_i, B_i)$ is the unknown term inferred by the UPINN. In order to do this, we generate 10 solutions (from the original equation) with different parameters, and we set the parameters to be randomly sampled from a uniform distribution. Here are the ranges of each parameter: $r : (0.2, 1.0)$; $K : (1, 10)$; $A : (10, 20)$; $B : (10, 20)$. The initial condition is set to be $N_0 = 0.1$, with time ranging from 0 to 20. We generate 100 datapoints for each solution, and there is no added noise.

For each of these 10 solutions, we can construct a dataset $\{N_i, t_i\}$ with specific parameters r, K, A, B which can be used to infer $\hat{Q}(N_i)$ using the UPINN. Concatenating each dataset and UPINN solution together, we can create the dataset $D_{dim} = \{A_i, N_i, B_i, \hat{Q}_i(A_i, N_i, B_i)\}$ on which PySR can be run, where $\hat{Q}$ is treated as a function of A, N, B. Without nondimensionalization, we attempt to recover Q as a function of A, N, B.

For the validation of our method, we nondimensionalize each of the ODE solutions with the above set of dimensionless terms to yield a dataset $\{\alpha_i, \tau_i\}$ with parameters γ, ϵ as defined above, where $\{\alpha_i, \tau_i\}$ from each solution can be given to UPINNs to find $\hat{Q}^*(\alpha_i)$. We create a final dataset $D_{nondim} = \{\alpha_i, \hat{Q}^*(\alpha_i)\}$ by concatenating the UPINN unknown terms from each ODE solution, and recover $\hat{Q}^*(\alpha)$ as a function of α only. The results for this evaluation are detailed in Section 5.4.

Rotating Bead Model

Now, we introduce a second-order DE which governs the displacement of a bead on a rotating hoop:

$$0 = -mr\frac{d^2\theta}{dt^2} - b\frac{d\theta}{dt} - mg\sin\theta + mr\omega^2\sin\theta\cos\theta \ . \tag{5.7}$$

We define the known term $P = -mr\frac{d^2\theta}{dt^2} - b\frac{d\theta}{dt}$ and the unknown term $Q = mr\omega^2 \sin\theta\cos\theta - mg\sin\theta$. The dimensions of the variables here are:

$$D = \begin{array}{c} \\ M \\ L \\ T \end{array} \begin{array}{c} \begin{array}{ccccccccc} m & r & t & \theta & \dot{\theta} & \ddot{\theta} & g & \omega & b \end{array} \\ \left[\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & -2 & -2 & -1 & -1 \end{array}\right] \end{array},$$

$$0 = P(m, r, b, t) + Q(m, g, r, \omega) \ . \tag{5.8}$$

Eliminating m (dimension $[M]$):

$$0 = P\left(\frac{m}{m}, r, \frac{b}{m}, t\right) + Q\left(\frac{m}{m}, g, r, \omega\right) \ . \tag{5.9}$$

Eliminating ω (dimension $[T]$):

$$0 = P\left(\frac{m}{m}, r, \frac{b}{m\omega}, t\omega\right) + Q\left(\frac{m}{m}, \frac{g}{\omega^2}, r, \frac{\omega}{\omega}\right) \ . \tag{5.10}$$

Eliminating r (dimension $[L]$):

$$0 = P\left(\frac{m}{m}, \frac{r}{r}, \frac{b}{m\omega r}, t\omega\right) + Q\left(\frac{m}{m}, \frac{g}{\omega^2 r}, \frac{r}{r}, \frac{\omega}{\omega}\right) \ . \tag{5.11}$$

We have concluded that the set of dimensionless parameters that reduces the number of parameters in Q is:

$$\varepsilon = \frac{g}{\omega^2 r}; \quad \tau = t\omega; \quad \gamma = \frac{b}{m\omega r} \ .$$

We can use this set of dimensionless parameters to nondimensionalize the equation. This set of parameters yields the following dimensionless form for the true underlying equation:

$$0 = -\frac{d^2\theta}{d\tau^2} - \gamma\frac{d\theta}{d\tau} - \epsilon\sin\theta + \sin\theta\cos\theta \ . \tag{5.12}$$

In order to nondimensionalize the equation when it has an unknown term, we can follow the following procedure:

1. Nondimensionalize time and/or the solution variable θ. Here we use $\tau = t\omega$. Here θ is dimensionless so we don't need to nondimensionalize it. Substituting in dimensionless time yields:

$$0 = -mr\omega^2 \frac{d^2\theta}{d\tau^2} - b\omega \frac{d\theta}{d\tau} + Q(\theta, m, g, r, \omega) \ .$$

2. To reduce the number of variables in the known term and remove dimensions from the entire equation, divide through by a dimensional scaling factor that makes all terms in the equation dimensionless. Here, $mr\omega^2$ is a good choice because these are all variables that appear in Q and hence we won't introduce any new variables into Q.

$$0 = -\frac{d^2\theta}{d\tau^2} - \frac{b}{mr\omega}\frac{d\theta}{d\tau} + Q(\theta, m, g, r, \omega)/(mr\omega^2) \ .$$

3. Q can be nondimensionalized with $5 - 3 = 2$ terms:

$$\theta, \epsilon = \frac{g}{\omega^2 r} \ .$$

4. Read off (or calculate, if π_2 appears in the known term) the corresponding dimensionless quantities in the known term. Here, the remaining dimensionless quantity is

$$\gamma = \frac{b}{mr\omega} \ .$$

Hence, Ipsen's method can be used to identify which exact sets of dimensionless variables will minimize the number of inputs to the unknown term, but the above procedure will allow us to actually perform the nondimensionalization.

For this DE, we proceed similarly to the logistic growth equation. To generate data, the original equation, with dimensions, is solved for different values of the parameters. We sample the parameters from a uniform distribution with the following ranges: $r : (1, 100), m : (1, 3), b : (1, 3)$. We use the fixed parameters $g = 9.8, \omega = 1.0$ and fixed initial conditions $(\theta, \dot{\theta} = (3.14, -0.1))$, with time ranging from 0 to 20. Then, the DE is solved (100 evaluations per solution) for each of the 10 sets of parameters through time using `scipy.odeint` [165].

To construct D_{dim}, for each ODE solve, we employ UPINNs to recover $\hat{Q}(\theta)$ from observations $\{\theta_i, t\}$ with specific parameters m, r, ω, g, b. Then, these estimates can be concatenated into a final dataset $D_{dim} = \{m_i, r_i, \theta_i, \hat{Q}_i(m_i, r_i, \theta_i)\}$. We do not need to

include variables such as b, ω, g because they are either fixed or do not appear in Q. Then, we run PySR to recover a symbolic expression for $\hat{Q}(m, r, \theta)$.

In order to construct D_{nondim}, we first nondimensionalize each ODE solve to yield a dataset $\{\theta_i, \tau_i\}$ for each set of parameters γ, ϵ. Then, we use UPINNs to recover $\hat{Q}^*(\theta)$ for each ODE solve. Concatenating the results together, we create $D_{dim} = \{(\epsilon_i, \theta_i, \hat{Q}^*(\theta_i, \epsilon_i))\}$. Now, we can use PySR to recover Q^* as a function of θ, ϵ. We expect that symbolic regression would be able to recover the function $Q(\theta, \epsilon) = -\epsilon \sin\theta + \sin\theta\cos\theta$.

5.4 Results

We detail the results of combining symbolic regression with the Buckingham Π theorem to discover algebraic equations (Section 5.4.1). We also show the results of learning unknown terms in two separate DEs–the extended logistic growth model, and the rotating bead model–using UPINNs combined with partial nondimensionalization (Section 5.4.2).

5.4.1 PySR with Buckingham π theorem

In our experiment, we utilized 7 distinct algebraic equations to compare the solution time and error for finding both the original and dimensionally reduced equations. Each data entry in Tables 5.1, 5.2 (split for readability) contains the values of the known parameters, which are generated randomly, and the corresponding value of the target expression. Figure 5.1 demonstrates the performance of the PySR model for each equation before and after dimensional analysis in terms of MSE on both the training set and the test set.

From Tables 5.1, 5.2, Fig. 5.1, we can see that nondimensionalizing the equation yields better recovery of the true equation and better test MSE. For all equations, nondimensionalizing the data results in PySR being able to recover the true nondimensionalized equation exactly (disregarding small constants on the order of 10^{-7} and smaller) every time, except for exponential decay. For these equations, both training and test error is at most 10^{-10}. In the case of exponential decay, the discovered equation still generalizes well to unseen test data, with a train and test MSE of 10^{-6}. By contrast, the expressions found by PySR for the original equation forms are most of the time incorrect. The correct expression was recovered for only 1 of 7 equations, that of the gravitational force. The train and test MSE sufficiently small (less than 10^{-10}) only for the gravitational force equation (original form). For the remainder of the equations in original form, the recovered expression does not generalize well to test data. Additionally, due to a

consequence of the Buckingham Π theorem, for Coulomb's law (last row), the nondimensionalization reduces the number of variables to a single π term, which allows, in the case of noiseless data, exact recovery of the equation. The inference time does not significantly differ between the original and dimensionless versions of the same equation for all equations except the first two; it is possible that for larger datasets or for more complex equations, nondimensionalization would significantly decrease the time taken to converge. However, the decrease in computation time upon nondimensionalization for *Free Fall* and *Terminal Velocity* indicates that the computational load decreases.

5.4.2 Symbolic Regression with Ipsen's method and UPINNs

We tested our method on first order and second order DEs, i.e. the extended logistic growth equation and the rotating bead equation of motion, comparing the results of discovered expression through UPINNs and PySR to the true equation.

For the extended logistic growth equation, we test our workflow on nondimensionalized and original data as per the methods. We plot the surrogate solution and the unknown term as recovered by the UPINN in Figure 5.2. We can see that for the dimensionless and original data, the UPINN is very good at recovering the unknown term in both scenarios. Table 5.3 shows the MSE for all the different sampled parameters, which is generally below 10^{-4} for both dimensionless and original equations. Table 5.4 and Table 5.5 show the recovered equations for the original data and the nondimensionalized data respectively. We note that for the original data, none of the proposed candidate expressions are correct, or have a loss less than 0.82. For the nondimensionalized data, PySR is able to recover the true expression exactly (row 6). To investigate whether a larger amount of data could help the unknown term be recovered more precisely for the original data, we show the results of repeating the experiment with 50 ODE solves (50 random sets of parameters) in Table 5.6. Although the lowest loss is now 0.71, the true equation is still not recovered.

For the rotating bead model, we obtain observations of the unknown term using UPINNs and then we give it as input to PySR and recover the true unknown term, in both the nondimensionalized and original equation. Table 5.7 show the parameters of each ODE solve and the MSE in recovering the unknown term for the original and dimensionless equation. The MSE tends to be quite large (>10) for the original equation. Figure 5.3 shows the recovered unknown term for a single ODE solve both before and after nondimensionalization. We note that the nondimensionalization we chose earlier for this problem does not change the scale of the data at all. Despite this, the unknown term changed, and the solution is much more difficult for the UPINN to recover from the

Equation	Original Equation	PySR-Deduced Equation	Train MSE	Test MSE	Runtime (s)
Free Fall Traditional	$S = s_1 + vt + gt^2/2$	$S = 0.83 \cdot s_1\sqrt{\exp(v)} - t(4.32t - 0.20)$	2.925×10^{-4}	0.01336	14.61
Free Fall Dimensionless	$\pi_1 = \pi_2 + \pi_3/2 + 1$	$\pi_1 = \pi_2 + (1.65 \times 10^{-8})\pi_2(\pi_2 - 0.612)\cos(\pi_3 + 0.762) + 0.50\pi_3 + 1.00$	1.01×10^{-13}	3.99×10^{-13}	5.36
Terminal Velocity Traditional	$V = \sqrt{2mg/(rCA)}$	$V = (-2A - 11.52m - 1.25)(0.309C + 0.309r\sin(A) - 1.0)/(C + r\sin(A))$	0.473	424.5	14.38
Terminal Velocity Dimensionless	$\pi_1 = 2\pi_2/\pi_3$	$\pi_1 = \pi_2(1.41 \times 10^{-7}\pi_3 + 2.0)/\pi_3$	2.64×10^{-16}	4.39×10^{-16}	5.16
Darcy-Weisbach Traditional	$P = f \cdot v^2/(2dg)$	$P = (4.896f(1.477g - 1) + g(d - 3.123))(\log(\cos(v)) + 0.00808)/g$	2.64×10^{-3}	10.56	5.35
Darcy-Weisbach Dimensionless	$\pi_1 = \pi_3/(2\pi_2)$	$\pi_1 = \pi_3(1.06 \times 10^{-7}\pi_3^2\cos(\pi_2) + 0.500)/\pi_2$	1.50×10^{-14}	2.64×10^{-14}	5.65
Poiseuille Pressure Drop Traditional	$P = 128\mu LQ/(\pi D^4)$	$P = (\mu\exp(L + 8.043\mu + 8.043\cos(D + 7.709)) + 7.391)/\mu$	292.7	1.17×10^{32}	5.20
Poiseuille Pressure Drop Dimensionless	$\pi_1 = 0.0245\pi_2^4$	$\pi_1 = 0.02454\pi_2(\pi_2 - 2.22 \times 10^{-8})^2(\pi_2 + 1.66 \times 10^{-7})$	2.38×10^{-14}	1.69×10^{-12}	5.81

Table 5.1: Analysis of PySR-discovered algebraic equations with traditional and dimensionless forms (first 4 of 7 equations).

original data. This shows that nondimensionalization, aside from potentially changing the scale of the data, could significantly change the difficulty of recovering the unknown term.

Equation	Original Equation	PySR-Deduced Equation	Train MSE	Test MSE	Runtime (s)
Exponential Decay Traditional	$n = n_0 \exp(-mgx/(k_B T))$	$n = 4.97 \times 10^{-6} \exp(1/\sin(x)) + 9.42 \times 10^{-6} \sin(0.0233/m) + 3.81 \times 10^{-6}$	8.15×10^{-11}	8.64×10^{27}	4.92
Exponential Decay Dimensionless	$\pi_1 = 1/\log(\pi_2/\pi_3)$	$\pi_1 = 0.205(\sqrt{\pi_3}(1.575\pi_2 + 1)/\pi_2)^{1/4} + 0.0262$	1.74×10^{-6}	1.83×10^{-6}	5.45
Gravitational Force Traditional	$F = -Gm_1m_2/r^2$	$F = -6.669 \times 10^{-11} \cdot m_1m_2/r^2$	1.42×10^{-27}	1.99×10^{-23}	5.07
Gravitational Force Dimensionless	$\pi_1 = -\pi_2$	$\pi_1 = -1.0\pi_2$	2.10×10^{-32}	3.21×10^{-32}	4.70
Coulomb's Law Traditional	$F = q_1q_2/(4\pi\epsilon_0 r^2)$	$F = \log(\epsilon_0 + 0.209 \exp(1/(r \cdot (5.81 \cdot \epsilon_0 + 0.125))))$	0.0216	0.1107	14.17
Coulomb's Law Dimensionless	$\pi_1 = 1/4\pi$	$\pi_1 = 1/4\pi$	0	0	–

Table 5.2: Analysis of PySR-discovered algebraic equations with traditional and dimensionless forms (last 3 of 7 equations).

For this reason, PySR does not perform well in recovering the unknown term from the UPINN output. Table 5.8 shows the result of PySR attempting to recover the true equation from the original data, and Table 5.9 shows the same for the dimensionless equation. PySR is able to recover the dimensionless equation, but not the original one. In fact, there are no equations that fit the unknown term well for the original equation.

r	K	A	B	γ	ϵ	MSE (orig.)	MSE (nondim.)
0.5502	8.0682	11.9152	16.2211	0.7490	2.0105	8.38×10^{-6}	6.54×10^{-7}
0.4212	8.2168	17.7998	12.7259	0.3011	1.5488	1.51×10^{-5}	7.51×10^{-8}
0.4863	5.5090	19.5814	18.7593	0.4658	3.4052	1.99×10^{-5}	1.16×10^{-7}
0.4962	6.0508	16.8346	17.1270	0.5048	2.8306	1.86×10^{-6}	2.92×10^{-7}
0.8183	8.9438	15.0308	10.1377	0.5519	1.1335	2.71×10^{-2}	2.19×10^{-5}
0.2603	4.3194	13.6489	16.1540	0.3081	3.7398	2.52×10^{-8}	1.00×10^{-7}
0.5178	8.0986	19.3314	16.5138	0.4423	2.0391	4.11×10^{-6}	1.64×10^{-7}
0.8953	4.9256	13.1684	15.6810	1.0661	3.1836	6.19×10^{-7}	6.69×10^{-8}
0.7634	7.3412	18.0215	11.4377	0.4845	1.5580	2.71×10^{-6}	6.99×10^{-7}
0.5537	9.1838	12.1879	19.2487	0.8745	2.0959	7.73×10^{-7}	6.75×10^{-6}

Table 5.3: Parameter values and MSE for original and nondimensionalized fits of the unknown term for the logistic growth equation.

Complexity	Loss	Symbolic Expression
1	18.845097	4.08144
2	12.891532	$\sqrt{N}$
3	2.7578475	$N \cdot 0.554923$
4	2.753279	$\exp(N \cdot 0.1348798)$
5	1.7926946	$(A \cdot 0.03401371) \cdot N$
6	1.0458996	$N \cdot \mathrm{inv}(B \cdot 0.13509956)$
7	0.99565214	$N \cdot (\cos(\sqrt{B}) \cdot -0.7250628)$
8	0.87281895	$\sin(\sin(\log(B) \cdot 0.9430373)) \cdot N$
10	0.81654054	$(\sin(\sin(0.9430345 \cdot \log(B))) \cdot N) + (-0.23878135)$

Table 5.4: Symbolic expressions returned by PySR for the logistic growth DE (original equation, data from 10 ODE solves).

Complexity	Loss	Symbolic Expression
1	0.064947285	0.25141922
2	0.057834398	$\sin(\alpha)$
3	0.00093266973	$\alpha \cdot 0.45889705$
5	0.0006026333	$(\alpha - 0.05566975) \cdot 0.48454857$
6	0.00053291634	$\sin((\alpha \cdot 0.53107023) - 0.03838172)$
7	**3.0821457e-6**	$\boldsymbol{\alpha \cdot \text{inv}(\text{inv}(\alpha) + \alpha)}$
9	3.0816245e-6	$\alpha \cdot \text{inv}(\alpha + (\text{inv}(\alpha) \cdot 1.0001483))$

Table 5.5: Symbolic expressions returned by PySR for the extended logistic growth model (nondimensionalized, data from 10 ODE solves). *inv* represents the function $1/x$. The true discovered equation is bolded, and is algebraically equivalent to the true unknown term, $\alpha^2/(1+\alpha^2)$.

Complexity	Loss	Symbolic Expression
1	15.0113325	3.293288
2	14.870333	$\sqrt{A}$
3	2.1417365	$N \cdot 0.52549046$
5	1.6920228	$N \cdot (N \cdot 0.03511109)$
6	1.3674117	$N \cdot (\text{inv}(B) \cdot 6.700879)$
7	1.1217595	$(N \cdot 0.017628053) \cdot (A + N)$
8	0.79894626	$N \cdot (\text{inv}(B) \cdot (A \cdot 0.4347375))$
9	0.7115485	$N \cdot (A \cdot ((B \cdot -0.0022081388) + 0.06363801))$

Table 5.6: Symbolic expressions returned by PySR for the logistic growth DE (original equation, data from 50 ODE solves).

b	r	m	ϵ	MSE (orig.)	MSE (nondim.)
2.5707	19.9604	1.8755	0.4910	1.30×10^2	2.04×10^{-4}
2.6037	78.2176	1.5529	0.1253	1.45×10^3	1.63×10^{-3}
2.0020	95.8558	1.7156	0.1022	3.24×10^3	3.90×10^{-3}
2.1224	68.6628	1.7405	0.1427	1.04×10^3	7.95×10^{-2}
2.7653	50.8052	2.5457	0.1929	1.56×10^3	7.65×10^{-3}
1.7376	37.1237	1.1508	0.2640	1.37×10^2	9.01×10^{-4}
2.5775	93.3809	1.7944	0.1049	2.85×10^3	3.59×10^{-4}
1.8723	32.3668	2.7383	0.3028	7.31×10^2	5.61×10^{-4}
2.4092	80.4126	2.4085	0.1219	3.51×10^3	7.87×10^{-2}
2.8186	22.6604	1.8843	0.4325	1.51×10^2	2.89×10^{-5}

Table 5.7: Parameter values and MSE for original and nondimensionalized fits for the rotating bead equation. Note that $g = 9.8$ and $w = 1.0$ are fixed across all runs and omitted from the table.

Complexity	Loss	Symbolic Expression
1	548.1215	3.4967551
2	545.7659	$\log(r)$
3	324.75864	$\theta \cdot -9.199068$
4	204.39363	$\exp(\theta \cdot -1.6174623)$
6	179.17783	$m \cdot \exp(\theta \cdot -1.3801409)$
7	146.54767	$((\theta + -3.6895895) \cdot \theta) \cdot 3.4955583$
8	144.21558	$\exp((\theta \cdot -0.78455627) + 2.3025424) + (-12.418046)$
9	136.51599	$(\theta \cdot ((\theta + -6.8664646) + \theta)) \cdot m$

Table 5.8: Symbolic expressions returned by PySR for the rotating bead DE (original equation, data from 10 ODE solves).

Complexity	Loss	Symbolic Expression
1	0.16081181	-0.030887194
3	0.0690577	$\theta \cdot -0.16079584$
4	0.068979	$\sin(\theta \cdot -0.16550536)$
5	0.055191655	$\sin(\theta) \cdot \cos(\theta)$
6	0.052178796	$\sin(\theta \cdot -2.0650928) \cdot -0.46723858$
7	0.025603876	$(\cos(\theta) - 0.24893238) \cdot \sin(\theta)$
8	0.022500763	$\sin(\theta) \cdot \sin(\cos(\theta) - 0.31793493)$
9	**0.017333511**	$\boldsymbol{\sin(\theta) \cdot (\cos(\theta) + (\epsilon \cdot -1.0024753))}$
10	0.014994156	$((\epsilon \cdot -1.1765212) + \cos(\theta)) \cdot \sin(\sin(\theta))$

Table 5.9: Symbolic expressions returned by PySR for the rotating bead problem (nondimensionalized, 10 ODE solves). The correct expression of the candidate expressions is bolded.

5.5 Conclusion and Discussion

Symbolic regression can be an invaluable tool in engineering and applied mathematics, allowing practitioners to use ML to automatically infer the equations that relate variables within a dataset. To enhance the accuracy and decrease the runtime of symbolic regression, we demonstrate the integration of dimensional analysis with the PySR symbolic regression pipeline. However, the approach can be used on any symbolic regression algorithm. This integration also allows one to discover equations in cases of partial knowledge, via UPINNs. The framework combines symbolic regression with dimensional analysis, utilizing both the Buckingham Π theorem and Ipsen's method. This approach effectively reduces the complexity of the search space and ensures physical consistency in the discovered equations. Our findings indicate that non-dimensionalizing datasets prior to symbolic regression significantly enhances both accuracy and computational efficiency.

For algebraic equations, the application of the Buckingham Π theorem notably decreased the number of independent variables, thus streamlining the symbolic regression process. This dimensionality reduction facilitated faster convergence, reduced the risk of overfitting, and improved accuracy, even in scenarios involving sparse datasets. We successfully recovered the symbolic expressions for several physics-based equations, demonstrating that this combined methodology can efficiently discover physically meaningful equations in complex systems with minimal computational resources.

In the context of DEs, we expanded this framework by integrating dimensional analysis and UPINNs under a partial knowledge scenario. Ipsen's method systematically reduced the number of unknown parameters by transforming the equations into their dimensionless forms and offered guidance on how to best nondimensionalize the partially known equation. The synergy between dimensional analysis and symbolic regression enhanced the capability of UPINNs to efficiently uncover unknown terms in DEs, leading to faster convergence and a diminished reliance on extensive datasets. This is particularly beneficial in situations where data acquisition is limited or expensive.

Additionally, while other works, such as those by Bakarji et al. [12], apply symbolic regression and the Buckingham Π theorem to scenarios like the rotating bead problem, our method introduces key distinctions. Bakarji et al. focus on identifying the most *physically meaningful* dimensionless terms that best match the data given an optimal relationship between the terms. In contrast, our approach utilizes the Buckingham Π theorem to uncover an unknown term in a partially known model, treating all nondimensionalizations as equivalent and favoring those that minimize the number of π groups involved in the unknown term. This simplification aids UPINNs in learning fewer variable interactions, especially under conditions of sparse data availability. Initially applying UPINNs allows for the learning of potentially complex unknown terms with minimal inputs, followed by a powerful and versatile symbolic regression technique, such as PySR, to precisely identify the true unknown term without optimizing for the "most physically meaningful" dimensionless π groups. Moreover, PySR could be substituted with other equally versatile symbolic regression algorithms like SymbolicGPT [159] or AI Feynman [158].

It can be argued that our method is successful at improving the efficiency of symbolic regression primarily because the number of variables is reduced. As such, it would be worthwhile to compare our method to a method that randomly reduces the number of variables in the equation by arbitrarily recombining them into new ones, or simply by removing some of the existing variables. This can be explored in detail in future work. We note that arbitrarily reducing the number of variables in the equation (or recombining them) will not necessarily constrain the symbolic regression search space in a way that yields dimensionally consistent formulas and simplifies the search problem. For example, removing a variable that is present in the true formula could make it difficult to recover the original formula, since some information is lost. Recombining the variables in a way that loses information (e.g. in discovering the true relationship $y = x_1 x_2$, if $x_1 = -x_2$ then creating a new variable $x_3 = x_1 + x_2$ will lose information) would make the search for the true equation difficult. Finally, as symbolic regression searches for the true underlying expression between the transformed variables, there is no guarantee that the resulting equation found would be dimensionally consistent. By contrast, the Buckingham Π

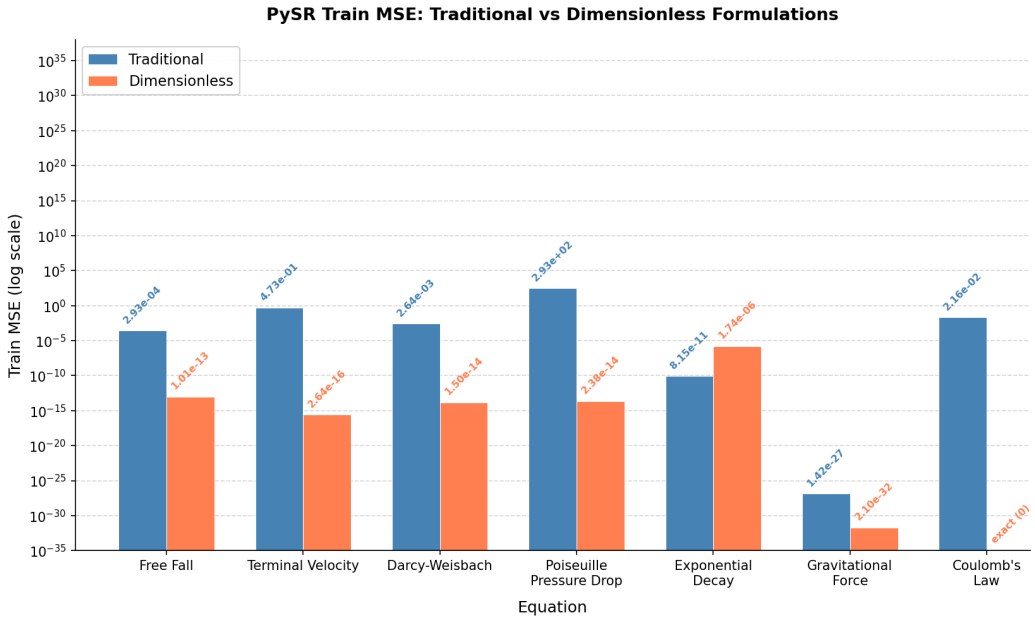
theorem (and Ipsen's method) provides a way to reduce the number of variables, preserve the true relationship between the original variables, and ensure dimensional consistency of recovered expressions.

On the role of π groups and their uniqueness: while the groups themselves are not unique and here we have focused on employing the set of π groups that minimizes the number of inputs to the hidden term, future work can more rigorously investigate the influence of the choice of π groups on UPINN and symbolic regression accuracy. We expect that sets of π groups that rescale the data and hidden term in a way that brings these components closer to $O(1)$ would yield better performance than other sets. However, further experiments would be necessary to test this hypothesis.

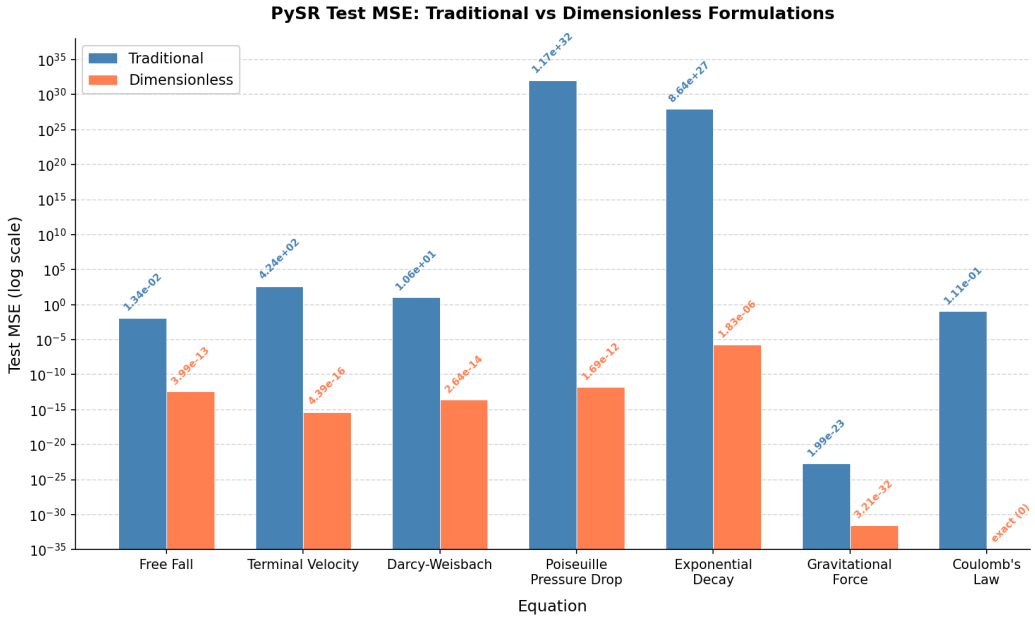
Our results confirm that the combination of symbolic regression and dimensional analysis offers a robust methodology for automatically discovering governing equations in various domains, including physics. By leveraging the inherent scaling properties of physical laws, we can significantly simplify symbolic regression tasks, resulting in more interpretable and accurate models. In disciplines such as engineering and computational biology, the task of fitting a model to data can be greatly simplified and made more robust using an integrated framework such as the one we propose.

Chapter Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), under the Discovery Grants Program, Grant No. RGPIN-2021-03472.

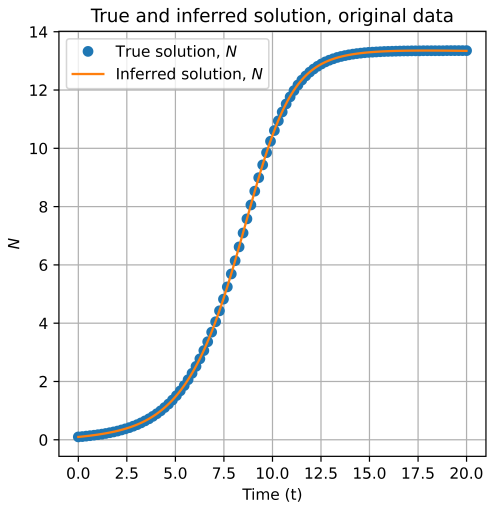


(a)

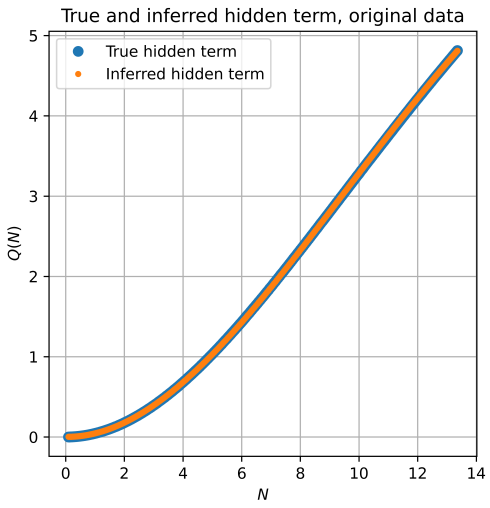


(b)

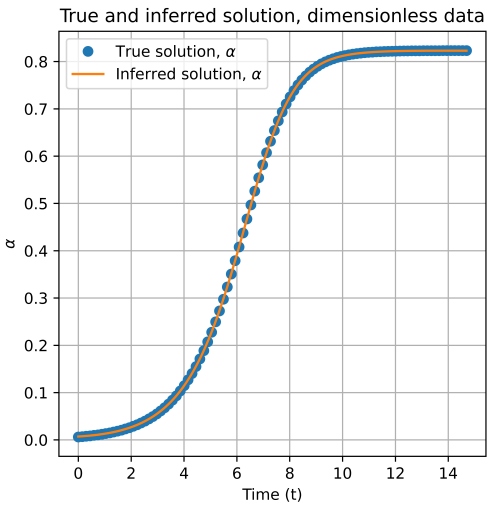
Figure 5.1: Comparison of train and test MSE for each algebraic equation.



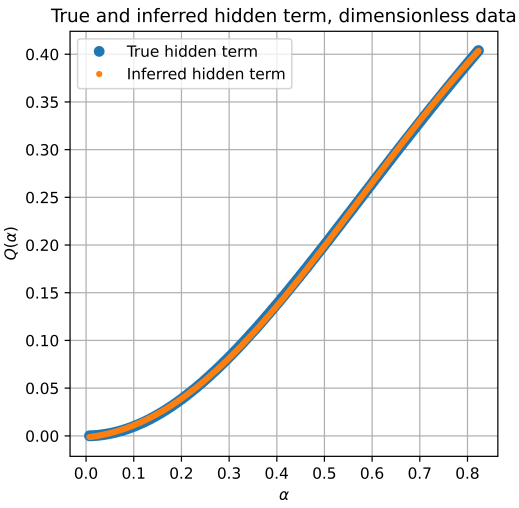
(a)



(b)



(c)



(d)

Figure 5.2: Solution and unknown term fit for the logistic equation, for one set of parameters. UPINNs do very well in recovering the unknown term in both cases of the original and the dimensionless equation. (a) and (b) are the original equation, whereas (c) and (d) are the fits for the dimensionless equation.

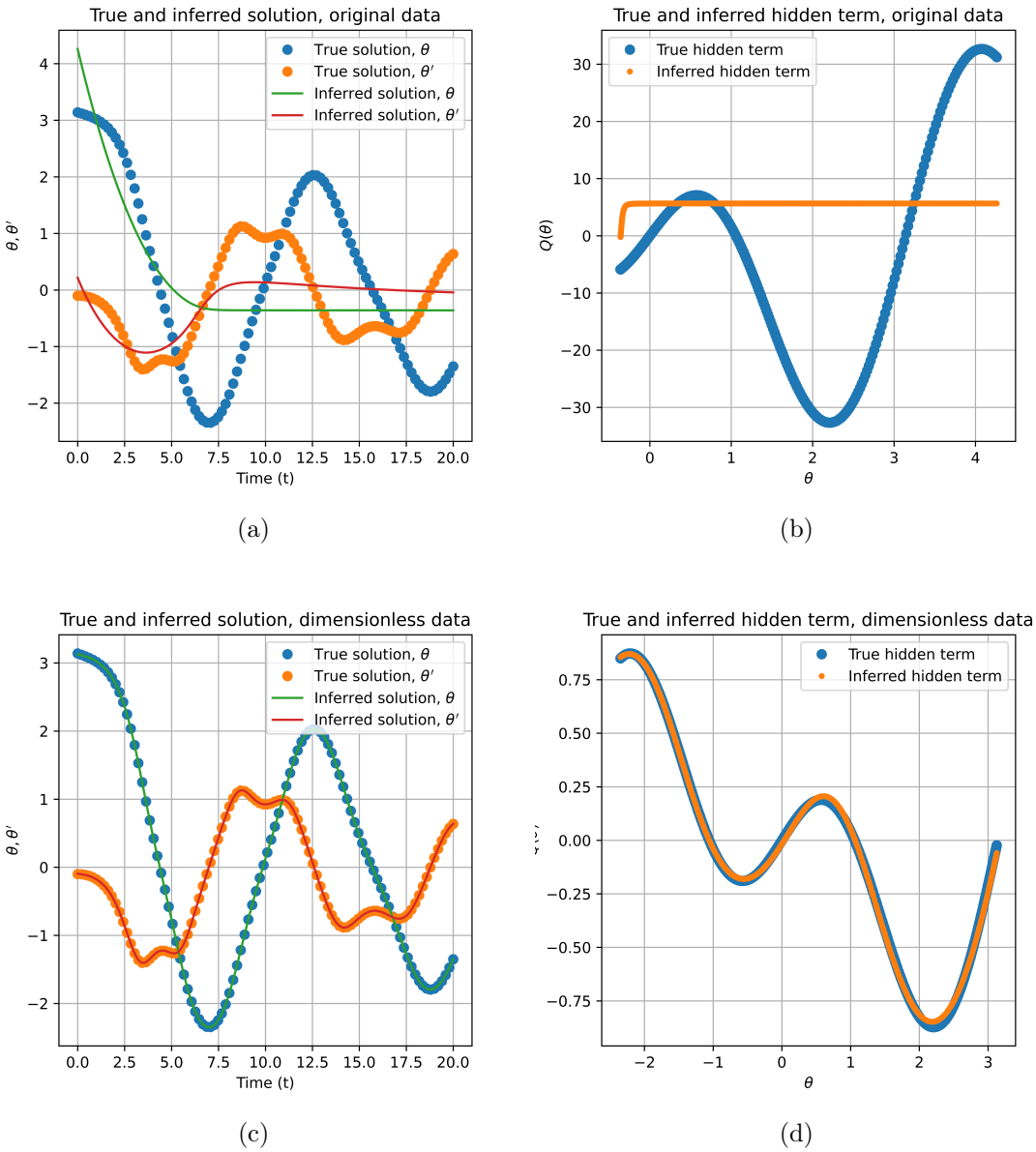


Figure 5.3: The recovered UPINN fit for both the original and dimensionless rotating bead DEs. (a) and (b) are the original equation, whereas (c) and (d) are the fits for the dimensionless equation.

Chapter 6

Catalyst GFlowNet for Electrocatalyst Design

Chapter Summary

Efficient and inexpensive energy storage is essential for accelerating the adoption of renewable energy and ensuring a stable supply, despite fluctuations in sources such as wind and solar. Electrocatalysts play a key role in hydrogen energy storage (HES), allowing the energy to be stored as hydrogen. However, the development of affordable and high-performance catalysts for this process remains a significant challenge. We introduce Catalyst GFlowNet, a generative model that leverages ML-based predictors of formation and adsorption energy to design crystal surfaces that act as efficient catalysts. We demonstrate the performance of the model through a proof-of-concept application to the hydrogen evolution reaction (HER) and the oxygen evolution reaction (OER), key reactions in HES, for which we successfully identified platinum (for HER) and ruthenium oxide (for OER) as the most efficient known catalysts. In future work, we aim to open the search space to discover new materials, in particular for OER, where current optimal catalysts are expensive metal oxides. This generative modeling framework offers a promising pathway for accelerating the search for novel and efficient catalysts.

6.1 Introduction

As society moves away from fossil fuels and towards renewable energy sources, technological advances such as renewable energy storage and efficient energy production become key drivers in this transition [61, 185]. Since peak energy usage hours differ from energy production hours [126], known as the "duck" curve, energy storage can ensure that cities obtain the energy they need at the right time. However, methods suitable for large-scale storage, such as hydrogen energy storage (HES) [9], require effective catalyst materials both for energy storage (via electrolysis) and for energy retrieval (via fuel cells) [120, 185]. Most catalysts that are currently relied upon to accelerate both energy storage and retrieval are expensive to manufacture [97]. For this reason, the search for efficient, cheap, and durable catalysts is vital to increasing the adoption of renewable energy.

ML, growing in its applications and reaching state-of-the-art performance in image generation [43], language modelling [162], and scientific discovery [168], has been applied to predict catalyst properties and generate new catalysts [35, 53, 143, 156]. Graph neural networks such as FAENet [53, 64, 145] are able to predict adsorption energies, which are a key predictor of catalytic efficiency [109, 117]. Methods such as Crystal-GFN [2] are capable of generating crystal structures within specified symmetry constraints, which meet certain desirable properties. These ML methods can be broadly divided into two categories, predictive models (used for catalyst evaluation) [52, 185], and generative models (used for catalyst design) [113, 143].

Here, we introduce Catalyst GFlowNet (Figure 6.1), a novel catalyst discovery method building on Crystal-GFN [2], designed to generate a diverse set of efficient catalyst materials. Generative flow networks (GFlowNets or GFNs) are generative models that are able to sample sequentially constructed objects from a desired distribution. The ability to sample proportionally to a reward function allows our framework to propose a variety of catalyst materials rather than a single optimal material. By contrast, a method such as reinforcement learning, which learns a policy to maximize a reward, would be unlikely to sample a wide diversity of materials. Within catalyst discovery, sampling a diversity of materials is key, as the reward function may not capture all the relevant predictors of whether a material will be a good catalyst. For example, the reward function may capture a target adsorption energy for the catalyst, but not its stability, which is an important trait (high stability) if the catalyst is to be used in real-world applications. Given a diversity of samples, subsequent real-world experiments have a greater chance of discovering a truly high-quality catalyst material when they are able to test multiple proposed materials.

Our approach for catalyst design builds on crystal generation by constructing a periodic crystal and then cutting a surface that acts as the catalyst. Additionally, we integrate

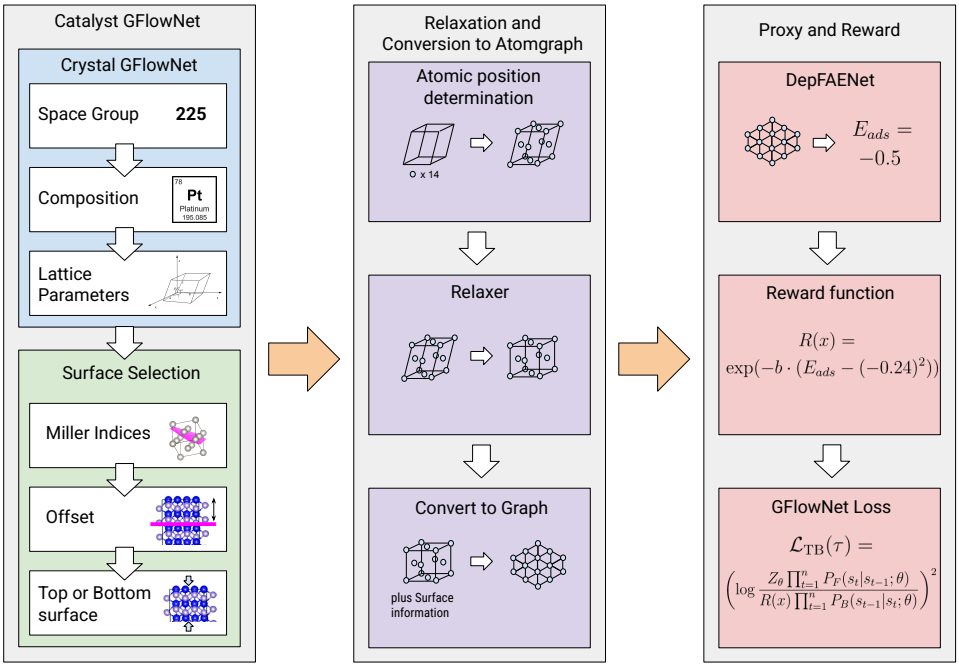


Figure 6.1: Overview of the Catalyst GFlowNet framework. The leftmost section, Catalyst GFlowNet, samples the catalyst surface. The middle section determines atom positions, relaxes the structure, and converts to a graph. The rightmost section obtains the adsorption energy from a predictive model and the reward function to train the GFlowNet.

ML-based structure relaxations into our framework to ensure that the final samples are relaxed (stable). As a signal on the quality of the catalyst material, we use a Graph Neural Network (GNN) based on FAENet [53] to estimate the adsorption energy for a given adsorbate molecule in a reaction of interest. A tailored reward function ensures that we generate realistic crystals that can then serve as the basis for efficient catalysts. In our case studies, we verify that Catalyst GFlowNet is able to rediscover the best-known catalysts, demonstrating its promising value for catalyst design.

6.2 Background

In this section, we perform an overview of catalysts for hydrogen energy storage (Section 6.2.1), describe GFlowNets in detail (Section 6.2.2), and we perform a survey of related ML work that tackles the catalyst discovery problem (Section 6.2.3).

6.2.1 Catalysts for hydrogen energy storage

Hydrogen Energy Storage (HES) is a scalable, sustainable option for storing excess energy for later use. Although HES has a relatively low efficiency compared to alternatives such as pumped storage hydroelectricity and battery storage [23, 120, 185], it is more scalable, transportable and stable than the alternatives (e.g. batteries). Green hydrogen production, which is a key component of HES, via water electrolysis (water-splitting) can be thought of as two half-cell reactions: the hydrogen evolution reaction (HER) and the oxygen evolution reaction (OER). Both HER and OER are electrochemical reactions which occur in the catalyzer at different electrodes. While HER is a one-step reaction, OER consists of four intermediate reactions which, under an applied voltage, produces oxygen gas, protons and electrons under acidic conditions. HER, in this setting, brings electrons and protons together to produce hydrogen gas. The reaction for water-splitting is

$$\mathrm{H_2O\,(liquid)} + 2.46\,\mathrm{V} \longrightarrow \mathrm{H_2\,(gas)} + \frac{1}{2}\,\mathrm{O_2(gas)}$$

consisting of half-reactions

$$\mathrm{H_2O\,(liquid)} \longrightarrow \frac{1}{2}\,\mathrm{O_2(gas)} + 2\,\mathrm{H^+\,(aqueous)} + 2\,\mathrm{e^-} \qquad \text{Anode (OER)} \tag{6.1}$$

$$2\,\mathrm{H^+(aqueous)} + 2\,\mathrm{e^-} \longrightarrow \mathrm{H_2(gas)} \qquad \text{Cathode (HER)} \tag{6.2}$$

As renewable energy is used for water electrolysis, the hydrogen gas can be collected and stored in tanks, or other storage containers. When the energy is in demand again, fuel cells can extract the energy from the stored hydrogen at about 60% efficiency [57]. An efficient catalyst, used in the fuel cell or in the electrocatalyzer, can significantly speed up the reaction rate and improve the efficiency of the process [185]. The best known catalysts for water-splitting are very expensive and rare: platinum (Pt) for HER, and iridium (Ir) for OER [152]. Although catalysts can have different form factors such as powders or nanostructures [177], all catalyst forms interact with the reaction and adsorbates through the catalyst surface. Hence, here model catalysts as crystal surfaces to which adsorbates (molecules in the reaction) can bind, and we do not account for the form factor. In this way, catalysts can speed up the rate of the reaction.

6.2.2 Generative Flow Networks

General approach

A GFlowNet is an inference method introduced by [18] that allows one to sequentially construct objects with probability proportional to a non-negative reward function. They have been employed in tasks such as molecule generation, crystal generation, and for causal discovery [2, 10, 91]. A desirable capability of GFlowNets is their ability to generate a diversity of objects, several of which could have a high reward (relative to other possible candidates in the space). This greatly aids inverse design and downstream experimental testing, as explained in the previous sections. Furthermore, GFlowNets perform particularly well in cases where the search space is quite large, and outperform methods such as MCMC and reinforcement learning in terms of mode mixing [19].

The main computational framework of GFlowNets can be thought of as a directed graph $G = (\mathcal{S}, \mathcal{A})$ where $\mathcal{S}$ are a set of states. Some of these states are terminal states, corresponding to fully defined catalyst surface. Terminal states have a nonnegative reward $R(s)$, while all other states in general do not provide a reward. Actions $a \in \mathcal{A}$ can be applied to transition from state $u \in \mathcal{S}$ to $v \in \mathcal{S}$. To give a concrete example, a Catalyst GFlowNet state could potentially be any of the following: an empty object (source state), a partially constructed crystal with a specific symmetry but no atoms within it (intermediate state), or a fully constructed catalyst surface (terminal state). Within this framework, the GFN learns the *edge flows* between states, $F : \mathcal{A} \to \mathbb{R}_{\geq 0}$. If action $a \in \mathcal{A}$ transitions from state u to state v, then the edge flow on action a (in intuitive terms) corresponds to the unnormalized probability of transitioning to v given state u. The flow on each action can be converted to a GFlowNet policy, which samples the transition $u \to v$ with probability $F(u \to v)/\sum_{v'} F(u \to v')$. It has been shown in [19] that if the total inflow into each state v equal the total outflow from v (and the outflows from terminal states x are set to be $R(x)$), then the GFlowNet samples terminal states (or completed objects) proportionally to the reward function. This is called the linear flow matching constraint, which can be implemented with the flow matching loss.

Although the flow matching loss is sufficient to train the GFlowNet, in our implementation we employ the equivalent trajectory balance loss [107]. Trajectory balance loss [107] accelerates training convergence compared to other objectives such as flow matching and the detailed balance loss. First, let a complete trajectory $\tau = (s_0 \to s_1 \to \ldots \to s_n = x)$ be the sequence of states taken to reach a terminating state x. Define $Z = F(s_0)$, or the total flow coming out of the source state. Suppose that the edge flows are Markovian as per [107]. Then, the edge flows induce a conditional probability distribution $P_F(\cdot|s_t)$ of reaching a

nonterminal state s_t's child (outgoing) states, and let $P_B(\cdot|s_t)$ be the corresponding probability distribution induced on a state's parent (incoming) states. These are the forward and backward policies respectively. Then the trajectory balance loss enforces

$$Z \prod_{t=1}^{n} P_F(s_t|s_{t-1}) = R(x) \prod_{t=1}^{n} P_B(s_{t-1}|s_t) \ . \tag{6.3}$$

Hence, the loss function that enforces this constraint on a particular trajectory τ can be formulated as

$$\mathcal{L}_{TB}(\tau) = \left(\log \frac{Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1})}{R(x) \prod_{t=1}^{n} P_B(s_{t-1}|s_t)} \right)^2 \ . \tag{6.4}$$

We use this version in Catalyst GFlowNet, adapted for batching. Z_θ is set as a learnable parameter, and the distributions P_F (forward policy) and P_B (backward policy) are represented with multi-layer perceptrons implemented in PyTorch [123].

Materials discovery considerations

A key component of GFlowNets is the proxy model, which is how the GFlowNet obtains information on the reward landscape, and hence what "desirable", high-reward samples are. Proxy models in our context are predictive models (possibly ML models) that allow one to evaluate the reward of a particular sampled object. In most cases, the proxy model is an approximation of another function, the oracle, which represents the true (e.g. experimental) evaluation of possible catalysts. Informing the GFN reward with the oracle directly (or performing repeated real-world experiments) is often computationally infeasible. For this reason, a ML model, the proxy, is used to evaluate the object. In the case of molecule or crystal discovery, it may be that the sampled molecule or crystal does not exist in any database, and as such, relevant properties of the crystal must be computed in real time instead. Here, we note that for certain properties such as the formation energy of a crystal, physics simulations such as density functional theory (DFT) are traditionally performed. However, these simulations are computationally expensive and not scalable to high batch sizes. Hence, a ML model that is able to replace the physics simulation and compute the properties of interest is valuable; for crystals, there are many works that develop ML-based approaches to compute properties such as formation energy and adsorption energy [37, 53, 145, 146].

Crystal GFlowNet is a GFlowNet designed to discover new materials by sequentially sampling crystal structures [2]. This work uses the GFlowNet paradigm to construct and

sample crystal structures that have a high reward. The authors test the method on a reward based upon formation energy, where a high reward corresponds to a low formation energy, and a low reward corresponds to a high formation energy. This allows the framework to be used to sample thermodynamically stable materials with desirable properties. As mentioned earlier, the reward function may use any measurable (or predicted, via a proxy model) characteristic of the material. Sequential crystal construction proceeds in three steps. First, the space group (symmetry operations) of the crystal are selected. Then, the composition (elements and their ratios), and finally, the lattice parameters, which determine the shape of the final lattice. Atom positions are not sampled by the model.

6.2.3 Machine learning for catalyst discovery

Since traditional catalyst design methods are time-consuming and involve trial and error experiments within a very large search space [71], scientists are turning to ML approaches to more efficiently explore the space of good catalysts. Recent work has leveraged both predictive models (which evaluate the performance of catalysts based on their measurable attributes) and generative models (which aim to construct catalyst representations with desirable properties) for catalyst discovery.

One major thread of predictive ML for catalyst design are predictive models to replace density functional theory (DFT) calculations of adsorption energies. The Open Catalyst Dataset 2020 [35] and 2022 [156] are datasets of 1.2 million and 62 thousand relaxations respectively. The datasets span a wide variety of materials, surfaces and adsorbates, with the inclusion of oxides in OC22. ML models such as FAENet, Schnet, PaiNN, and Graphormer [53] are neural networks (e.g. graph neural networks, transformers) that can be trained to perform the OC20 Initial Structure to Relaxed Energy (IS2RE) task, which involves predicting the relaxed energy of an adsorbate-catalyst system from their initial atomic positions. ML models have also been used to predict formation energies, perform relaxations, and predict forces [37]. Predictive ML models have also been used to predict the thermodynamic stability of catalysts [186]. All of these predictive models have the potential to be included as proxy models within generative approaches.

Recent generative approaches to catalyst discovery include reinforcement learning [93], large language models [94], variational autoencoders [143], and generative adversarial networks [76] which aim to generate catalysts *de novo*, by meeting criteria such as target adsorption energies and encoding this in the reward or loss. AdsorbRL [93] employs Deep Q-learning to find compositions of catalysts that have either minimal or maximal

adsorption energy. They only model composition, omitting space groups, crystal symmetries, surface selection, and atom positions. [143] addresses the problem by employing a VAE to generate catalysts by sampling from the latent space. We note that reliance on the latent space makes controllable generation and systematic exploration of possible catalysts difficult. [76] takes the approach of GANs to generate catalysts for the ammonia formation reaction. However, this work does not take catalyst stability into account during generation. Tools based on large language models such as [94, 118] have so far been mainly applied to help screen known catalysts and perform a literature search.

6.3 Methodology

In this section, we introduce the general architecture and workflow of the Catalyst GFlowNet (Section 6.3.1) and the setup for the hydrogen evolution reaction (HER) (Section 6.3.2) and oxygen evolution reaction (OER) case studies (Section 6.3.3).

6.3.1 Architecture

The Catalyst GFlowNet builds on the Crystal-GFN framework. The overall framework, depicted in Figure 6.1, consists of three sequential steps: sample generation, sample preparation, and reward computation. After the reward is computed, a GFlowNet objective, such as the trajectory balance loss [107], can be used to train the GFlowNet to sample catalyst representations proportionally to the reward function.

In the first phase, the catalyst surface is sequentially constructed by the GFlowNet. Here, we use Crystal-GFN to build catalyst surfaces given a crystal structure. In particular, this involves first selecting the space group (integer between 1 and 230, possibly decomposed into lattice system and point group); following this, selecting the composition (e.g. $A_xB_yC_z$); and finally, selecting the lattice parameters (lattice vectors a, b, c and angles between them α, β, γ). Conditioned on this crystal representation, the Catalyst GFlowNet selects the Miller indices [167] (three integers ranging from -2 to 2) representing the surface of the periodic crystal object that would act as the catalyst. For example, the Miller indices (100) cut a plane that is parallel to the Y-Z plane created by the lattice vectors[1]. Following this, the offset for the surface plane is selected (continuous value between 0 and 1), which represents the location of the cut in the direction perpendicular to the Miller plane. Finally,

[1]Miller indices hkl denote the 3D plane (and those parallel to it) that intersects the x-axis at $1/h$, the y-axis at $1/k$, the z-axis at $1/l$.

the network selects whether the top or bottom surface (represented by a boolean) is used. For a single-element material, such as platinum, the offset and top/bottom surface boolean do not change the resulting surface. However, for more complex materials, such as oxides, the offset and the surface boolean can lead to surfaces with widely different properties.

In the second phase (middle part of Figure 6.1), first we obtain atomic positions symmetrically compatible with the crystal representation by using templates. We create a lookup table which maps each space group in the search space to an existing stable structure in the Materials Project database [79]. When a particular space group is selected, we use the Materials Project structure to set the atomic positions. In particular, we rescale the structure based on the GFN-sampled lattice parameters, and replace the species in the template structure with those sampled by the GFN. The order in which the elements were sampled by the GFN determines which elements will go in which set of atomic positions. If for a given space group, the sampled composition does not match the template (e.g. if the number of elements or number of atoms does not match), then the sample is marked as invalid and the GFN obtains a reward of zero for that sample. The full template lookup table can be seen in 6.1. In a larger search space, using templates would not be possible–there could be hundreds or thousands of possible stable structures within the same space group. We are able utilize templates in our case studies because we have a limited search space, and we intend to filter out certain unstable space group-composition combinations. After the structure is fully determined, M3GNet [37] is used to relax the atoms and the unit cell without explicitly preserving symmetries. Then, a converter module creates a surface object using the final structure and sampled Miller indices, top surface information, and offset. Finally, this surface object is converted into a graph, where each node represents an atom. The adsorbate is its own graph, disconnected from the surface [31].

In the final phase (right-most side of Figure 6.1), the atom graph is passed to the proxy model, DepFAENet in our case, which is a GNN based on FAENet [53], equivariant to Euclidean symmetries, trained to predict adsorption energies for different adsorbates. DepFAENet differs from the original FAENet implementation in that the surface and adsorbate are disconnected graphs [31]. The model predicts the adsorption energy directly without situating the adsorbate on the surface. Then, the predicted adsorption energy is passed to the reward function (described in detail in Section 6.3.2). The rewards obtained for the constructed samples are then used to compute the trajectory balance loss to train the GFlowNet.

Table 6.1: Templates used to set the atomic positions for HER and OER. The materials can be looked up on Materials Project based on their ID.

Experiment	Space Group	Composition Form	Material	Materials Project ID
HER	225	A	Pt	mp-126
HER	229	A	Pt	mp-2646979
OER	136	AO_2	IrO_2	mp-2723
OER	221	ABO_3	$SrTiO_3$	mp-5229

6.3.2 Case study: Hydrogen Evolution Reaction

For this case study, our aim is to generate stable catalysts for the hydrogen evolution reaction (HER) that have the smallest possible *overpotential*. In catalysis, the overpotential (η) is the amount of electric potential that needs to be applied on top of the theoretical reaction potential; it is a measure of the efficiency of a chemical reaction and its catalyst. For HER, in acidic conditions, it can be predicted directly from the adsorption free energy of hydrogen [117] (which is what our GNN proxy is trained to predict) plus a small correction. Here, to obtain proposed catalysts (x) with low overpotentials, we use the following reward function:

$$R_{E_{ads}}(x) = \exp(-b\eta^2) \quad \text{where} \quad \eta = E_H(x) + E_{corr} \ . \tag{6.5}$$

We set $b = 100$ and the Gibbs correction $E_{corr} = -0.24$. Although the reward only takes the overpotential (η) into account, structure relaxation for each sample (middle component of Figure 6.1) ensures that we generate reasonably stable structures. Additionally, at sampling time, structures that have high formation energy (= energy above hull, since we only have single-element structures) among structures with the same composition are discarded.

The adsorption energy is predicted by a proxy model called DepFAENet, which is a graph neural network (GNN) based closely on FAENet [53]. Both models employ frame averaging in order to ensure that adsorption energy predictions are invariant to rotations, translations and reflections in 3D. DepFAENet is trained to predict the adsorption energy of various adsorbates, including *H, on the Initial Structure to Relaxed Energy task of the OC20 [35] dataset (1.2 million DFT relaxations). Hence, the input to the model is an adsorbate and a structure graph, and the output is a real value representing the adsorption

energy of the adsorbate to the structure. DepFAENet differs from the original FAENet [53] implementation in that it is *disconnected*. This means that the graph containing the adsorbate and surface contains a disconnected graph object for each of the adsorbate and surface. Previous implementations relied on situating the adsorbate and catalyst relatively close in space, and predicting the adsorption energy from this point (Initial Structure to Relaxed Energy task from [35]). In contrast, a disconnected FAENet is simpler to train, because it does not take binding sites into account. Instead, it is learning general trends about a surface and adsorbate. We assume that the effect of the site is small compared to the termination, surface, coverage and material, following [36]. Training and implementation details can be found on FAENet's public repository on GitHub. The mean absolute error of the model on the validation set is 0.77 eV.

In order to incentivize the GFlowNet to generate a higher proportion of stable structures, we additionally test the following reward function:

$$R_{comp}(x) = R_{E_{ads}}(x) \cdot R_{forces}(x) \cdot R_{E_{hull}}(x) \ , \tag{6.6}$$

$$R_{forces}(x) = \mathbb{1}_{\Delta E<t} \ , \tag{6.7}$$

$$R_{E_{hull}}(x) = \exp(-E_{hull}(x)^2/2\sigma^2) \ , \tag{6.8}$$

where R_{forces} represents a reward on the final forces on the structure, and $R_{E_{hull}}$ represents a reward on the structure's energy above hull.

R_{forces} is set to be 1 if the forces on the structure are sufficiently low, and 0 otherwise. This incentivizes the GFN to create structures that after relaxation have low forces on the cell and the atoms. We compute a proxy for this quantity by computing the energy difference between the last two relaxation steps of the structure. We set the threshold t to 0.001 eV, which is a standard measure of convergence for DFT.

$R_{E_{hull}}$ is a reward on the estimated energy above hull, which is a measure of the thermodynamic stability of the structure. The *energy above hull* is a measure of the stability, computed in eV. If a structure has an energy above hull of 0, then it is stable. Above zero but less than a small threshold (such as 0.2 eV) is metastable. The reason for introducing energy above hull is the need for a measure of the composition-dependent global stability of a structure based on all the other possible forms those elements could take. Although low formation energy indicates a more stable structure, formation energy alone cannot be used as a reliable signal on stability in a large search space, because it can only be meaningfully compared between structures with the same composition. Additionally, while the forces on a particular structure can be zero, this only guarantees

that this structure is in a *local* energy minimum, but not necessarily in a *global* energy minimum. Hence, there could be a different structure for that composition with low forces and a lower formation energy, which would mean it is more stable. In that case, this second structure would be more stable than the first. Energy above hull as a reward adds in a second measure of stability based on formation energy. The reward for a particular structure is a Gaussian centered at zero as a function of the estimated energy energy above hull. We set $\sigma = 0.2$ eV. We note that for structures in the HER search space, formation energy and energy above hull are the same because they are all single-element structures, but this is not true in general.

In order to compute the energy above hull we use PyMatGen [119]. For a new GFN-sampled relaxed structure x, we first estimate its formation energy E_f with M3GNet. Then, we add (x,E_f) to the phase diagram. For each k-dimensional composition space (e.g. {Sr, Cr, O}, $k = 3$), this utility constructs a convex hull of the formation energies as a function of the composition. Each sample is placed in the composition space based on the ratios of its elements (e.g. $SrCrO_3$ is placed at {Sr: 20%, Cr: 20%, O: 60%}). The sample (x,E_f) is added to the hull if there is no lower-energy entry for that particular element ratio, and if there is no lower-energy decomposition of existing entries (e.g. if 50% Sr and 50% O has a lower formation energy than SrO, then SrO will not be added to the hull). After updating the convex hull, the energy above hull for x is 0 if it it was added to the hull, and more generally is the vertical distance to the hull within the composition space. We construct the hull dynamically during training as the GFN discovers new structures. For larger composition spaces, constructing a hull based purely on exploration could lead to missing large sections of the phase space, so for future work we consider precomputing a convex hull based on the structures in Materials Project [79] and updating it dynamically during training.

As a proof of concept, we test our workflow on a highly restricted search space, where atomic positions are fully determined given the space group and the number of atoms. In this search space, we have an expectation of what materials should be sampled the most. With other methods, such as those based on large language models [113], imposing constraints on crystal symmetry, space group, and number of atoms would be much more difficult, if not impossible. The search space that we consider matches closely the catalysts described in Nørskov et al. [117]. The full description of the experimental setting can be found in Table 6.2. We consider 12 possible elements, including platinum, the best known HER catalyst [117]. We consider the space of single-element structures only, as per Nørskov et al. [117]. Taking the most common form of each structure as found in the Materials Project [79], we note the space group of each structure and restrict the search space to those space groups only. These are space groups $Fm\bar{3}m$ (225) and $Im\bar{3}m$ (229).

Table 6.2 details the search space for the Hydrogen evolution reaction case study (Section 6.3.2). We note that in general, the Catalyst GFlowNet's constraints and search space can be set to be much larger than that used in the HER case study. In future work, we will consider a larger search space up to 80-100 atoms, more elements and space group options, while enforcing a neutral charge constraint.

Table 6.2: Hydrogen evolution reaction catalyst search space.

Setting	Value	Units
Possible elements	Pt, Ag, Au, Pd, Ir, Ni, W, Co, Cu, Mo, Rh, Nb	
Min. # of different elements in unit cell	1	
Max. # of different elements in unit cell	1	
Min. atoms in unit cell	2	
Max. atoms in unit cell	4	
Min. atoms per element	2	
Max. atoms per element	4	
Enforce neutral charge?	No	
Possible space groups	225, 229	
Min. lattice parameter length	2	Angstroms
Max. lattice parameter length	6	Angstroms
Min. angle between lattice vectors	60	Degrees
Max. angle between lattice vectors	140	Degrees

6.3.3 Case study: Oxygen Evolution Reaction

The oxygen evolution reaction (OER) occurs in the electrolyzer alongside HER. We aim to search a subset of the space of elements considered in [109] and recover the best known catalysts, RuO_2 and IrO_2. There are currently no known inexpensive, effective catalysts for OER. The search space is detailed in Table 6.3. We search the space of AO_2 and ABO_3 type structures from [109], which have space groups 136 and 221 respectively. In future work, we may open the space to all structures from [109].

There are three intermediates involved in the OER reaction that are produced in the following order: *OH, *O, *OOH. Each of their adsorption energies corresponds to a step in the reaction. Because of the existence of a relationship (referred to as "scaling relationship" [109]) between the adsorption energy of *OH and *OOH that applies across a

Table 6.3: Oxygen evolution reaction catalyst search space.

Setting	Value	Units
Possible elements	Sr, La, Cu, Mo, Ir, Ru, Ti, Rh, Nb, Re, V, Mn, Cr, Fe, Co, Ni, O	
Min. # of different elements in unit cell	2	
Max. # of different elements in unit cell	3	
Min. atoms in unit cell	1	
Max. atoms in unit cell	6	
Min. atoms per element	1	
Max. atoms per element	6	
Enforce neutral charge?	Yes	
Possible space groups	136, 221	
Min. lattice parameter length	2	Angstroms
Max. lattice parameter length	6	Angstroms
Min. angle between lattice vectors	60	Degrees
Max. angle between lattice vectors	140	Degrees

wide range of materials, it is reasonable to assume that the step that determines the rate of the reaction (and equivalently, the overpotential) is the one that happens when *OH is converted into *O. In this approximation, the overpotential is minimized when $E_{\mathrm{O}} - E_{\mathrm{OH}} = 1.6 \text{ eV} + E_{corr}$, where E_{O} is the predicted adsorption energy of *O and E_{OH} is the predicted adsorption energy of *OH. $E_{corr} = 0.251$ eV [68] is a Gibbs free energy correction based on values from [109]. Hence, we choose the adsorption energy reward for OER to be:

$$R_{E_{ads}}(x) = \exp(-b\eta^2) \quad \text{where} \quad \eta = (E_{\mathrm{O}} - E_{\mathrm{OH}}) - (1.6 \text{ eV} + E_{corr}) \ , \tag{6.9}$$

where $b = 4$. Due to findings by [36] on the uncertainty of oxide DFT calculations in OC22, we choose a wider Gaussian for the adsorption energy reward than the one selected for HER. We set the composite reward to be exactly the same as that of HER:

$$R_{comp}(x) = R_{E_{ads}}(x) \cdot R_{forces}(x) \cdot R_{E_{hull}}(x) \ , \tag{6.10}$$

$$R_{forces}(x) = \mathbb{1}_{\Delta E < t} \ , \tag{6.11}$$

$$R_{E_{hull}}(x) = \exp(-E_{hull}(x)^2 / 2\sigma^2) \ , \tag{6.12}$$

where the convex hull is constructed only from samples from within the OER search space. We set $t = 0.001$ as before, and $\sigma = 0.2$.

In order to predict the adsorption energy of *O, *OH, we finetuned the DepFAENet adsorption energy proxy model from the HER case study on the OC22 [156] dataset. This was necessary because the OC20 dataset, despite being very large, does not contain oxides. The new OC22 dataset of 62k DFT relaxations was introduced to fill this gap. Further training details can be found on FAENet's public repository on GitHub. The mean absolute error on the validation set is 1.29 eV. This is a fairly large error, most likely due to generalization error from DepFAENet, and the large (0.5eV) uncertainty of OC22 DFT calculations [36].

In order to sample oxides only (compounds with oxygen), we must restrict the action space. This can be done by enforcing the neutral charge constraint, which would ensure that all compositions sampled by the GFN are "charge-neutral". The total charge of a structure here is defined as the sum of estimates of each element's *oxidation state*. For example, if O takes oxidation state -2 and Ru takes $+4$ then RuO_2 is charge-neutral, but RuO_3 is not. However, elements can take different oxidation states in different compounds. We implement the charge neutrality check by assigning each element to have a restricted set of oxidation states that it can take. Within our subset of elements and their corresponding oxidation states, charge-neutral structures are of the following forms: AO_2, ABO_3, A_2O_2. Table 6.4 shows the oxidation states of each element that enforce this. In particular, La and Sr can form ABO_3 compounds with the remaining elements, but only elements with oxidation state $+4$ can form AO_2 type compounds. A_2O_2 type compounds with space group 136 are not stable, and we filter these out via templates (Section 6.3.1): any structure with space group 136 must have a composition in AO_2 format. If A_2O_2 is sampled, the sample is marked as invalid and gets a reward of zero.

In order to restrict the action space to meet the neutral charge constraint, we perform a depth-first search on each selected composition action (before it is taken) to ensure that after taking it, we are still able to reach a neutral charge. This ensures that all sampled compositions have a neutral charge.

Table 6.4: Oxidation states of elements in OER.

Composition	Possible Oxidation States
La, Sr	$+2$
O	-2
Cu, Mo, Ir, Ru, Ti, Rh, Nb, Re, V, Mn, Cr, Fe, Co, Ni	$+4$

6.4 Results

In this section, we detail our results. First, we investigate the effect of relaxation on lattice parameters for platinum (Section 6.4.1), and note that structures with different initial lattice parameters relax to the same lattice corresponding to the bottom of the energy basin. In Section 6.4.2, we show that the GFN recovers the best known HER catalyst, platinum, using the pipeline. By incorporating the composite reward (eq. 6.6), however, we can improve the likelihood of generating stable samples (Section 6.4.3). In Section 6.4.4, we show that we can recover ruthenium oxide and iridium oxide as the best known catalysts for OER, by using the composite reward. Finally, in Section 6.4.5 we perform an ablation study to show how the composition distribution changes for various components of the composite reward.

6.4.1 Effect of relaxation on lattice parameters

First, we briefly investigate the effect of ML-based relaxation on lattice parameters for Platinum with space group 225 (cubic lattice), and note that lattice parameters tend to converge to the same value. After the GFlowNet generates samples, we relax them using M3GNet PES [37], as detailed in Section 6.3.1. Figure 6.2 demonstrates the change in lattice parameters of the cubic platinum lattice before and after relaxation, for the HER case study with only $R_{E_{ads}}(x)$. We note that the lattice parameters relax to the minimum of the total energy of the system. This indicates that given a structure with a specific arrangement of atoms (here, Pt with space group 225), incorporating ML-based relaxation is sufficient to generate relaxed versions of this structure.

6.4.2 Hydrogen Evolution Reaction: Adsorption Energy Reward

In this section, we analyze the catalysts sampled by the Catalyst GFlowNet after being trained, following the experimental setup for the hydrogen evolution reaction (HER). We sampled 1000 structures and performed a M3GNet-based relaxation on the crystal lattice. Then, we filter out all structures that relax to a high formation energy. Specifically, we compute the minimal sampled formation energy per composition, and keep only those relaxed structures that have a formation energy within 0.05 eV of the minimum energy. We do this because in some cases the relaxation strays too far from the true basin given the starting lattice parameters, and as such would not be a stable structure. Then, for each composition, we keep only the space group that has the lowest minimum energy, since the

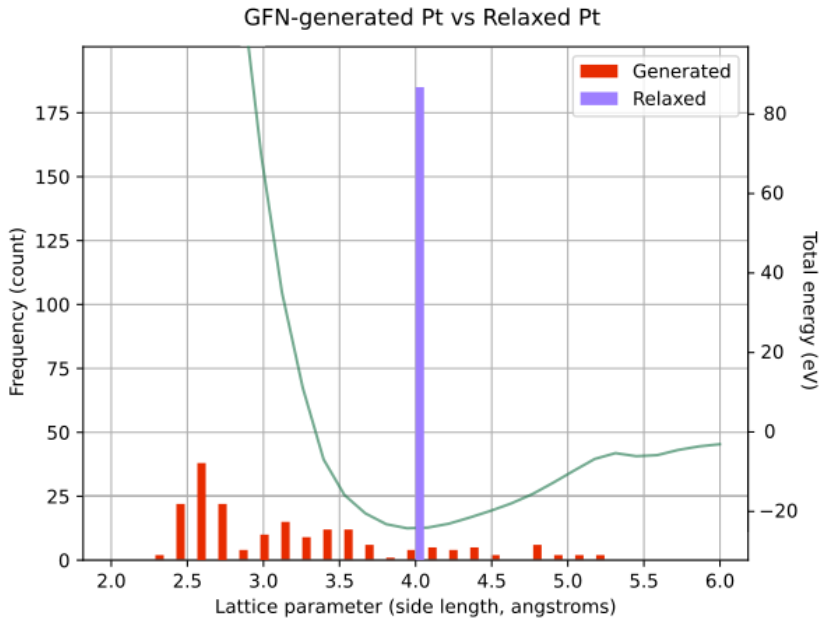


Figure 6.2: For the HER case study, the platinum samples generated by the GFlowNet relax to the minimum energy every time. The green line represents the total energy of the system.

one with the higher energy is estimated to be unstable. Finally, we cut the surface according to the parameters chosen by the Catalyst GFlowNet to construct the final catalyst surface.

The filtered set of relaxed structures contains 425 structures. Figure 6.3 details the percentage of samples of each of the structures in [117]. As expected, the most frequently sampled structures are those that have a low overpotential, as predicted by the proxy model and DFT values by Nørskov et al. [117] (see Figure 6.3, left). Platinum (Pt) is known to be the best catalyst for HER, but rhodium (Rh) also performs well, as evidenced by experimental, DFT, and proxy values. We find in fact that rhodium and platinum are the most sampled structures. All generated structures except palladium with space group 229 are found in Nørskov et al. [117] (and are experimentally observed [79]). Palladium with space group 225, the one that is experimentally observed, was sampled, but filtered out because its M3GNet-predicted formation energy was slightly higher than its 229 counterpart. This is a limitation of the proxy, and it occurs because the actual formation energies are close for the two structures. Overall, this experiment validates that the

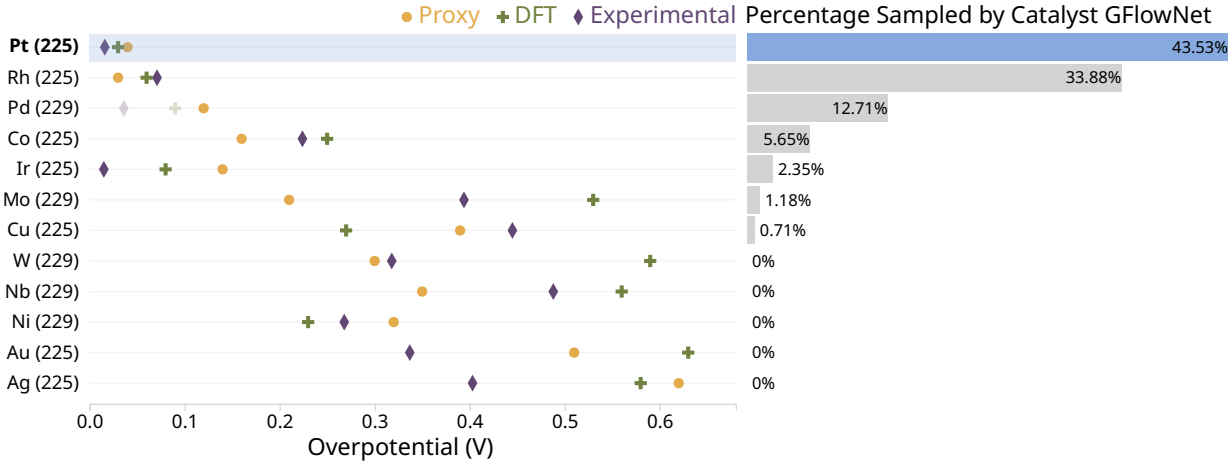


Figure 6.3: Proportion of sampled structures (right) for the hydrogen evolution reaction case study. Low overpotential (left) is a predictor for efficient catalysts and aligns with the higher sampling rates. The values are detailed in Table 6.5 in the appendix. Pd(229) does not have experimental or DFT overpotentials, corresponding values for Pd(225) are displayed with shaded colors for reference.

Catalyst GFlowNet framework can independently rediscover the best known HER catalysts, and sample them proportionally to their performance.

6.4.3 Hydrogen Evolution Reaction: Composite Reward

Next, we perform a GFlowNet run with the composite reward (eq. 6.6), and compare to a run with $R_{E_{ads}}$. This reward, $R_{comp}(x)$, is a product of a reward on the adsorption energy (evaluated in Section 6.4.2), energy above hull, and a reward on the total forces on the structure. This reward is described in detail in Section 6.3.2. We filter the 1000 samples in the same way as described in the previous section, by only keeping the samples that correspond to the space group per formula that has the lowest formation energy. In Table 6.5, we reproduce the results from the previous section, and compare to samples from the composite reward. The distributions of the sampled compositions are very similar for both runs, with the top sampled compositions being 43% of platinum for $R_{E_{ads}}$ and 40% of platinum for R_{comp}. However, with the composite reward, there are now 729 valid structures from the sampled 1000 compared to using the adsorption energy reward only, which only has 425 structures left after filtering. This means that the additional rewards on the forces and energy above hull enables us to sample stable structures with a greater

probability. Additionally, in Table 6.5, we add the experimental and DFT values for the sampled structures. When experimental measurements were unavailable, the exchange current density and overpotential was inferred from a linear relationship between overpotential and log of exchange current density from [157] and [45]. The linear fit can be seen in Figure 6.4.

Table 6.5: Proportions of sampled structures for the case study: hydrogen evolution reaction.

		Overpotential (eV)			Samples, $R_{E_{ads}}(x)$		Samples, $R_{comp}(x)$	
Composition	Space Group	Exp.[2]	DFT	Proxy Model	Count	Percentage	Count	Percentage
Pt	225	0.016	0.03	0.04	185	**43.53**	292	**40.05**
Rh	225	0.071*	0.06	0.03	144	33.88	228	31.27
Pd	229	0.036*	0.09*	0.12	54	12.71	110	15.10
Co	225	0.224*	0.25	0.16	24	5.65	42	5.76
Ir	225	0.015	0.08	0.14	10	2.35	10	1.37
Mo	229	0.394*	0.53	0.21	5	1.18	5	0.69
Cu	225	0.445	0.27	0.39	3	0.71	42	5.76
W	229	0.318*	0.59	0.30	0	0.00	0	0.00
Nb	229	0.488*	0.56	0.35	0	0.00	0	0.00
Ni	229	0.268	0.23	0.32	0	0.00	0	0.00
Au	225	0.337	0.63	0.51	0	0.00	0	0.00
Ag	225	0.403	0.58	0.62	0	0.00	0	0.00
Sum					425	100.00	729	100.00

6.4.4 Oxygen Evolution Reaction: Composite Reward

In this section, we analyze the samples from a GFlowNet run with reward $R_{comp}(x)$ (eq. 6.10) as applied to OER, as detailed in Section 6.3.3. In particular, we compare the sampled composition distribution, inferred space groups, adsorption energy, formation energy, forces, and energy above hull for both experiments – R_{comp} and $R_{uniform}$. We find that in all metrics, the composite reward yields more stable structures, with an adsorption energy closer to the target than the uniform reward. We sample 1000 structures each from a GFlowNet trained on the OER composite reward, and a GFlowNet trained using a uniform reward, where every sample gets a reward of 1.0. We call this reward $R_{uniform}$.

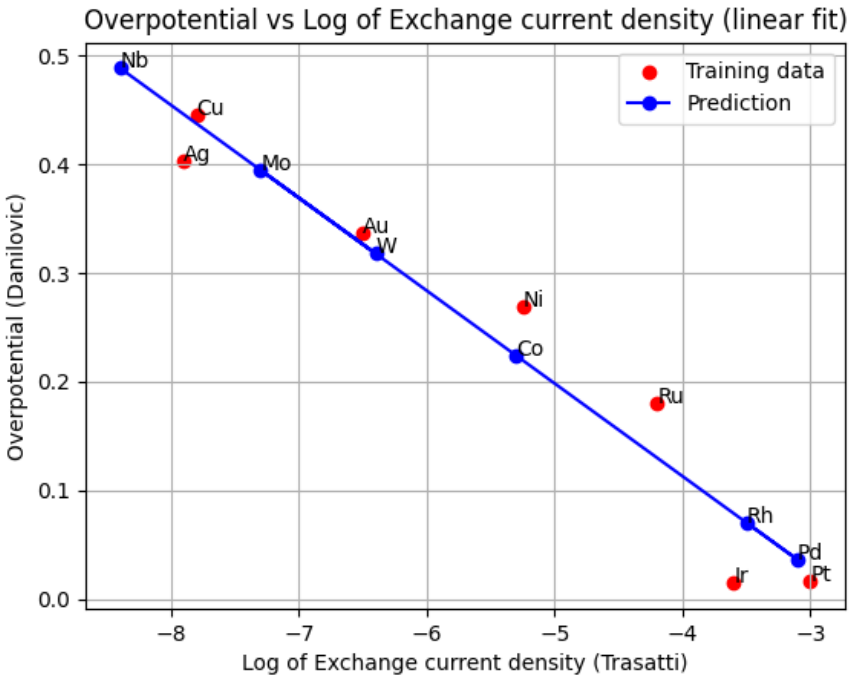


Figure 6.4: A linear fit that maps log of exchange current density to experimental overpotential using data from [157] and [45]. These values are used column 3 (from the left) of Table 6.5.

For both runs after sampling, we relax the samples for 40 steps. We do not perform any space-group or formation energy-based filtering in this section, because there are no compositions in the set of sampled structures that take two or more different space groups.

Table 6.6 shows the distribution of inferred space groups for $R_{uniform}$ and R_{comp}. In order to infer the space groups of the relaxed samples, we used PyMatGen's [119] `SpaceGroupAnalyzer` utility. Based on the symmetry of the crystal, this utility infers the space group of the crystal within some tolerance. In particular, the utility returns the space group with the highest number of valid symmetry operations. Since the relaxation does not preserve symmetries, the space group may change after relaxation from the GFN-sampled space group. For R_{comp} samples, the space group distribution was overwhelmingly 221 and 136, with only 5 samples changing space groups from the GFN-sampled space group. By contrast, for $R_{uniform}$ samples, only 65% of samples stayed in their intended space groups, and the rest changed to various lower-symmetry space groups (including space group 1,

which has no symmetry). This means that relaxation significantly changed the symmetry of the crystals, which implies the stable structure for that composition and set of lattice parameters is not in the sampled space group. Hence, there are no stable structures nearby with that symmetry. However, we note that structures that changed space group may in fact be quite close to matching the sampled space group.

Although 1000 samples were sampled from both models, the uniform reward yielded 148 invalid samples (and 852 valid samples). These are sampled where the composition did not match any possible template. Under the composite OER reward, these samples would obtain a reward of zero. Since there are no invalid samples from the composite reward, the zero-reward was successful in preventing the GFN from sampling invalid compositions.

Table 6.6: Inferred space groups for the OER experiment with R_{comp} reward.

	Samples, $R_{\text{comp}}(x)$		Samples, $R_{\text{uniform}}(x)$	
Space Group	Count	Percentage (of 1000)	Count	Percentage (of 852)
136	726	72.6	276	32.4
221	269	26.9	287	33.7
1	0	0.0	206	24.2
2	0	0.0	1	0.1
3	0	0.0	1	0.1
6	2	0.2	14	1.6
7	0	0.0	3	0.4
8	0	0.0	15	1.8
31	0	0.0	5	0.6
35	0	0.0	2	0.2
38	1	0.1	15	1.8
65	0	0.0	7	0.8
99	0	0.0	6	0.7
102	1	0.1	0	0.0
113	1	0.1	6	0.7
160	0	0.0	8	0.9

Tables 6.7 and 6.8 show the distribution of the compositions for the composite OER reward and uniform reward side-by-side. The most sampled compositions are RuO_2, MnO_2, IrO_2, and it is known that RuO_2 and IrO_2 are the best known catalysts for OER. By

contrast, the uniform reward samples all compositions fairly uniformly, as required. MnO_2, the second most-sampled composition for R_{comp}, is not the best known catalyst for OER [109], but it has been recently investigated as a potential alternative for RuO_2 [72].

Table 6.7: Proportions of sampled structures for the case study: oxygen evolution reaction. For the samples from the uniform reward run, we only list the compositions that were also sampled by the composite-reward run. Additionally, we only list those structures that were sampled at least 3 times by the composite-reward run.

		Samples, $R_{comp}(x)$		Samples, $R_{uniform}(x)$	
Composition	Space Group	Count	Percentage (of 1000)	Count	Percentage (of 759)
RuO_2	136	196	**19.6**	21	2.46
MnO_2	136	140	14.0	25	2.93
IrO_2	136	138	**13.8**	25	2.93
$LaRhO_3$	221	109	10.9	16	1.87
$SrRhO_3$	221	92	9.2	15	1.76
VO_2	136	75	7.5	24	2.81
RhO_2	136	69	6.9	21	2.46
ReO_2	136	51	5.1	29	3.40
FeO_2	136	32	3.2	17	1.99
CoO_2	136	24	2.4	22	2.58
$LaCrO_3$	221	16	1.6	17	1.99
$SrCrO_3$	221	15	1.5	8	0.93
$LaCuO_3$	221	5	0.5	25	2.93
$LaNiO_3$	221	4	0.4	18	2.11
CrO_2	136	4	0.4	16	1.87
$SrNiO_3$	221	4	0.4	10	1.17
$SrCuO_3$	221	3	0.3	16	1.87
$SrFeO_3$	221	3	0.3	18	2.11

Figure 6.5 shows the adsorption energy of the two sets of samples. It can be seen that the adsorption energy of the samples are closer to the target for R_{comp} than for $R_{uniform}$. In fact, the number of samples that are within 0.5 eV (the estimated uncertainty of the OC22 DFT dataset, as reported by [36]) of the target energy for OER are 366/852 (43%) of the uniform dataset, and 628/1000 (63%) for the composite reward. It is also worth noting that both distributions of samples are centered around the target energy. This is not surprising,

given that investigations in [36] showed that the structures in OC22 also have this property.

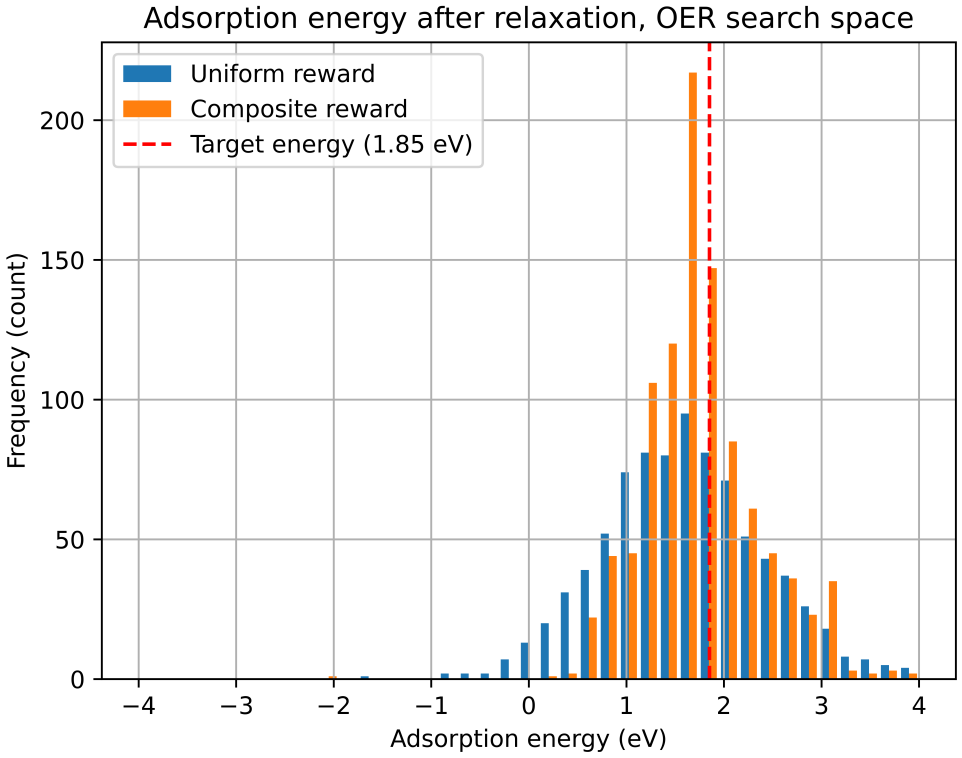


Figure 6.5: Adsorption energy of samples from R_{comp} and $R_{uniform}$, within the OER search space.

Figure 6.6 shows the distribution of formation energies predicted from both $R_{uniform}$ and R_{comp}. We can see that the uniform reward yields a wider distribution of formation energies, while the composite reward yields a more peaky distribution. This is consistent with the composite reward concentrating the sampling to specific compositions. Additionally, we can see that the uniform reward yields a few samples that have a formation energy above zero, which are by definition not thermodynamically stable.

Figure 6.7 shows the distribution of the energies above hull for each sample. The hull was created by creating a single convex hull from both sets of samples. Then, the energy above hull was computed with PyMatGen's `PhaseDiagram` utility. Although formation energies cannot be compared directly, energy above hull should be as close as possible to zero. Here, we note that the composite reward has 947/1000 samples that have E_{hull} below 0.1 eV,

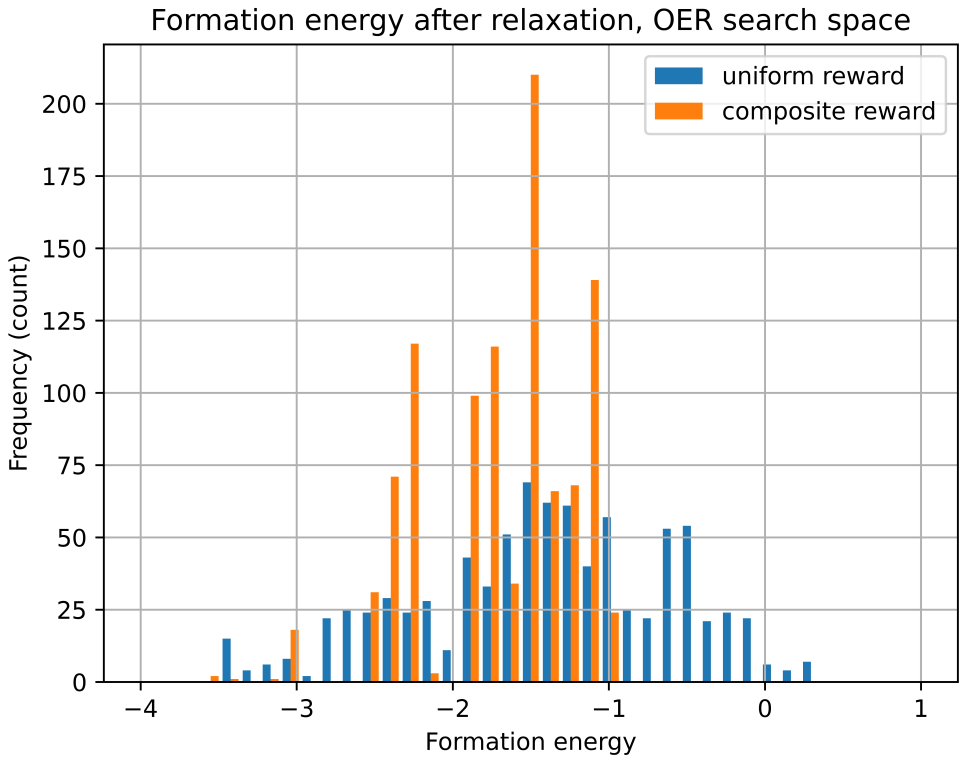


Figure 6.6: Formation energy of samples from R_{comp} and $R_{uniform}$, within the OER search space.

whereas for the uniform reward, 232/852. This shows that the composite reward is very good at generating stable structures.

Finally, in Figure 6.8 we have plotted the histogram of the force measure (difference between the last two energies of the relaxation). We see that 91% of samples from R_{comp} have low forces (below 0.001) on the atoms. 51% of samples from $R_{uniform}$ have this property. Hence the composite reward is effective at creating structures with low forces on the atoms.

In summary, this comparison between a run with the uniform reward and the composite OER reward shows that in the OER setting, the different components of the reward work together to generate a diversity of structures that are stable, have an adsorption energy close to the target, and keep their symmetry upon relaxation.

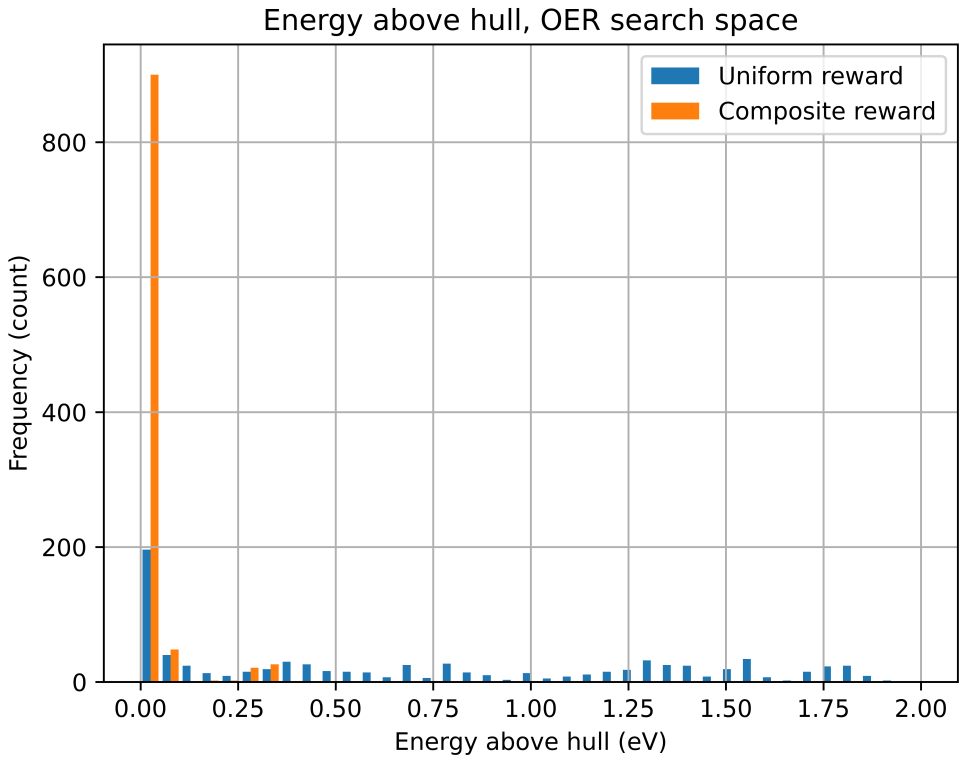


Figure 6.7: Energy above hull of samples from R_{comp} and $R_{uniform}$, within the OER search space.

6.4.5 Ablation study

Finally, we perform an ablation study on the reward components with the HER search space (Table 6.2). In order to investigate the effect of the stability rewards (E_{hull} and forces) on the composition distribution, we ran three different experiments. In the first experiment, we run the HER experiment with the forces reward only. In the second experiment, we run the HER experiment with the E_{hull} reward. In the third experiment, we use the product of these. More concretely, the rewards for the three experiments are:

$$
\begin{aligned}
R_1(x) &= R_{forces}(x) \ , \\
R_2(x) &= R_{hull}(x) \ , \\
R_3(x) &= R_{forces}(x) \cdot R_{hull}(x) \ ,
\end{aligned}
$$

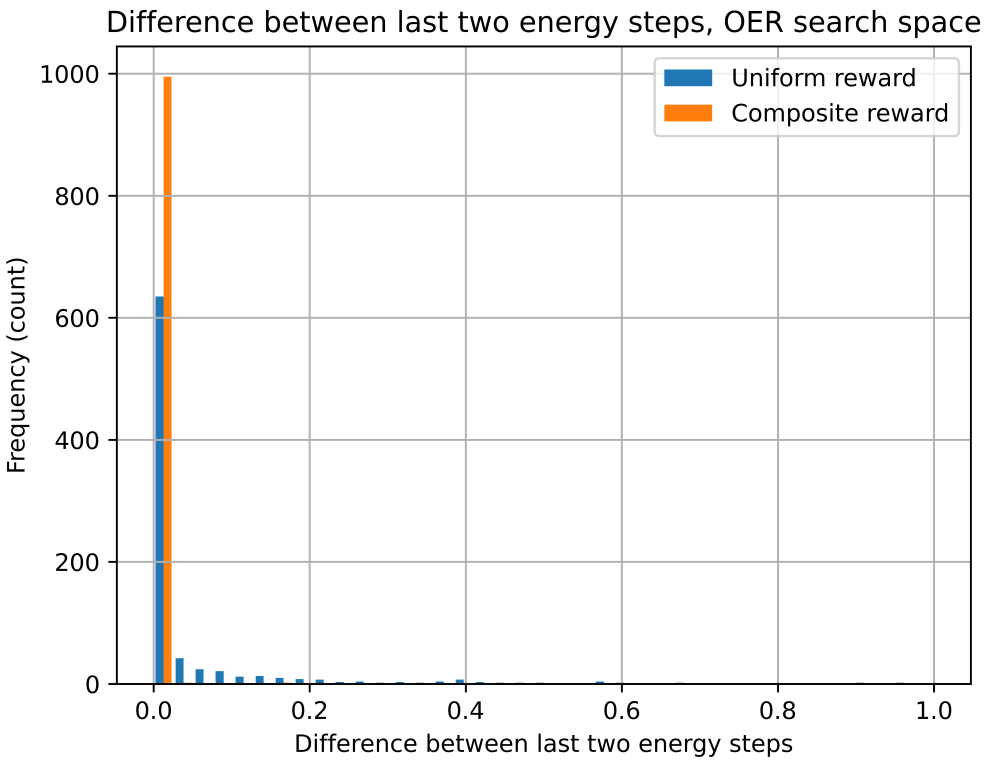


Figure 6.8: Force estimate of samples from R_{comp} and $R_{uniform}$, within the OER search space.

where the rewards are defined as eq. 6.6. The composition distribution of samples from these rewards is shown in Table 6.9 (and Figure 6.9), with the two most sampled compositions of each experiment bolded. It is noteworthy that for R_1 and R_2 (and hence by consequence for R_3 as well), the distribution of the compositions is far from uniform. In the context of R_1, it means that certain compositions have more actions that lead to relaxed structures than others. For example, cobalt with space group 229 may have many more lattice parameter combinations that relax to a structure with low forces. This kind of difference between structures would significantly impact the composition distribution. This phenomenon can be seen with R_2 as well, and since $R_3 = R_1R_2$ then it is unsurprising that this occurs with R_3 as well. However, we note that although platinum (225) and (229) are not highly sampled for R_3, they still make up a large proportion of samples for R_{comp}. Hence, if the adsorption energy reward is sufficiently narrow, this effect of non-uniform

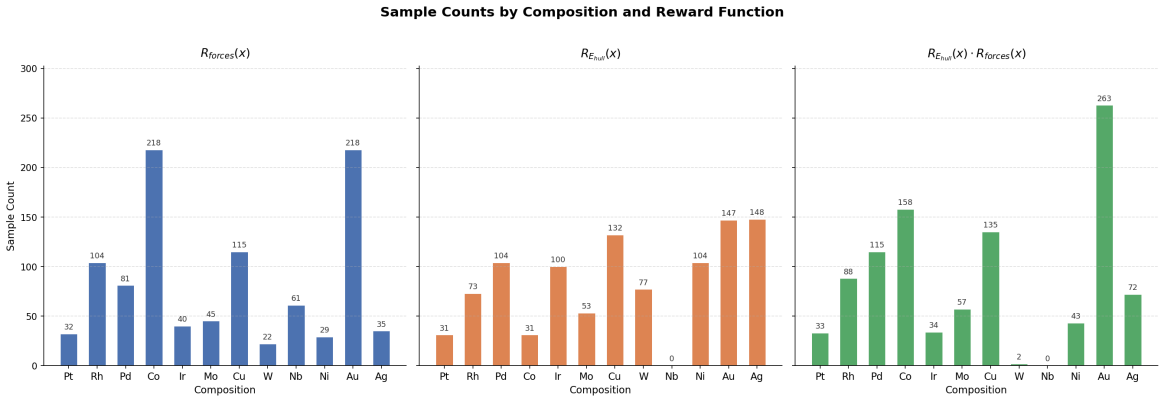


Figure 6.9: Ablation study with the HER search space.

composition sampling under $R_{E_{hull}} \cdot R_{forces}$ is less pronounced.

This kind of unbalance in the compositions in $R_{1,2,3}$ is in fact not desirable, because our task is to sample compositions based on whether they are suitable catalysts, not whether they easily relax to stable structures. A more desirable distribution of compositions would be uniform over all single-element compositions, and within the samples for each composition, preference towards the more stable space group in that composition. However, this would require either conditional sampling, or reweighting at sampling time. We leave this investigation for future work.

6.5 Future Work

Firstly, the ablation study in Section 6.4.5 naturally suggests several additional changes to the pipeline. Ideally, we would require that the distribution of the compositions was approximately equal for all of $R_{1,2,3}$ in Section 6.4.5. However, as shown by the sample distribution, this naturally does not occur. Hence, there are several paths forward: (1) one could train partially (or fully) without relaxation and (2) one could reweight the samples from R_{comp} samples based on the composition distribution of samples from R_3. Option (1) makes the reward and distribution that the GFN learns more difficult. This is because now the GFN has to generate structures with exactly relaxed lattice parameters. Additionally, since we build the energy hull during training, without relaxation it could take a long time for the hull to converge to a realistic representation of the search space. A compromise here could involve performing relaxation the first k steps of training, in order to build up the hull, but training without relaxation after this. Further investigation would show if the

continuous component of the GFN which samples the lattice parameters (and which will sample atomic positions in the future) can create fully relaxed structures reliably without relaxation. The second strategy (2) of reweighting the policy is likely to be successful with the current architecture. However, it does require two runs of the GFN: one with a reward on E_{hull} and forces, and one with the full composite reward. There may be a dynamic reweighting scheme which allows correcting for the composition imbalance within a single run.

Additionally, certain improvements can be made to the architecture which render the pipeline more realistic. One such change could be to improve the charge check. Currently the hard constraints are based on straightforward assignments of oxidation states. However, some elements have a higher preference for specific oxidation numbers; other oxidation numbers are relatively rare for that element. These details could be brought into consideration by either an additional reward, or an additional inductive bias on the actions. Another consideration for improving the pipeline is to select the offset based on the lowest energy surface for that set of Miller indices. Selecting the offset can make a significant difference in the adsorption energy of the structure, but stability (both forces and E_{hull} is assessed only for the repeating 3D crystal structure, not the final surface). Hence, integrating some kind of surface energy predictor and minimizing per structure could be a potential extension of the work.

6.6 Conclusion

In this chapter, we propose a novel catalyst discovery framework that generates complete and relaxed crystal structure descriptions (including atomic positions) for the purpose of finding an inexpensive, efficient catalyst for renewable energy storage. We show that the method can find the best currently known catalyst for the hydrogen evolution reaction (i.e., platinum). We also test the framework on the oxygen evolution reaction, for which there is currently no efficient, inexpensive catalyst. In particular, we search the space of oxides detailed in Man et al. [109] and aim to recover the best known catalysts (ruthenium oxide, iridium oxide) in that search space. Furthermore, we show that our method samples diverse structures, increasing the likelihood of finding suitable catalysts that withstand real-world testing. This framework is the prototype of a novel method to generate *de novo* stable catalysts for a variety of materials science tasks; integration with real-world experiment and active learning approaches would have an impact in the search to reduce the cost of renewable energy storage. In future work, we aim to add functionality to the Catalyst GFlowNet in order to sample atomic positions. We foresee the extension of this

work by embedding the GFlowNet in a multi-fidelity active learning loop [73].

Table 6.8: Proportions of sampled structures for the case study: oxygen evolution reaction (continued). For completeness, here we list the remainder of the sampled structures, by either the uniform run and the composite-reward run.

		Samples, $R_{comp}(x)$		Samples, $R_{uniform}(x)$	
Composition	Space Group	Count	Percentage	Count	Percentage
$LaNbO_3$	221	2	0.2	29	3.40
$LaRuO_3$	221	2	0.2	12	1.40
$LaReO_3$	221	2	0.2	20	2.34
$SrTiO_3$	221	2	0.2	20	2.34
$SrReO_3$	221	2	0.2	18	2.1
$SrRuO_3$	221	2	0.2	15	1.76
$SrCoO_3$	221	2	0.2	18	2.11
$LaVO_3$	221	1	0.1	21	2.46
$SrIrO_3$	221	1	0.1	18	2.11
MoO_2	136	1	0.1	20	2.35
$LaMoO_3$	221	1	0.1	19	2.23
TiO_2	136	1	0.1	25	2.93
$SrMnO_3$	221	1	0.1	21	2.46
$SrNbO_3$	221	0	0	18	2.11
$SrVO_3$	221	0	0	18	2.11
$SrMoO_3$	221	0	0	28	3.28
$SrRhO_3$	221	0	0	15	1.76
$LaIrO_3$	221	0	0	13	1.52
$LaMnO_3$	221	0	0	23	2.69
$LaFeO_3$	221	0	0	28	3.28
$LaTiO_3$	221	0	0	24	2.81
$LaCoO_3$	221	0	0	11	1.29
NbO_2	136	0	0	39	4.57
NiO_2	136	0	0	27	3.16
Sum		1000	100	852	100%

Table 6.9: Ablation study with the HER search space.

	Sample Counts		
Composition	$R_{forces}(x)$	$R_{E_{hull}}(x)$	$R_{E_{hull}}(x) \cdot R_{forces}(x)$
Pt (225)	7	28	11
Pt (229)	25	3	22
Rh (225)	44	73	52
Rh (229)	60	0	36
Pd (225)	46	39	28
Pd (229)	35	65	87
Co (225)	47	30	31
Co (229)	**171**	1	**127**
Ir (225)	15	**100**	17
Ir (229)	25	0	17
Mo (225)	12	0	1
Mo (229)	33	53	56
Cu (225)	42	59	20
Cu (229)	73	73	115
W (225)	8	0	1
W (229)	14	77	1
Nb (225)	15	0	0
Nb (229)	46	0	0
Ni (225)	7	91	4
Ni (229)	22	13	39
Au (225)	101	**97**	114
Au (229)	**117**	50	**149**
Ag (225)	8	90	27
Ag (229)	27	58	45
Sum	1000	1000	1000

Chapter 7

Conclusion

In this thesis, we made three main contributions to physics-informed ML for scientific discovery, which leverages prior physics information to improve the performance of ML models, applied to scientific problems. In Chapters 2, 3, and 4, we made contributions to PINNs.

In Chapter 2, we extended PINNs to learn unknown terms in differential equations. We call this method UPINNs. They can be used to learn terms in differential equations under partial knowledge. We compare them to UDEs and find that they perform better than UDEs in recovering the unknown term under both sparse and noisy data. Additionally, we recover the true form of the unknown term in the Lotka-Volterra system of equations by running AI Feynman symbolic regression on the output of UPINNs. We can recover the true term with greater accuracy when we use UPINNs as opposed to using UDEs. Additionally, we test the UPINN method on the 1D Burger's PDE, and on a system of ODEs representing the mechanism of apoptosis. We obtain good identification of the unknown term even with sparse observations of the variables. In future work, this method could be more tightly integrated with symbolic regression. In particular, a symbolic regression algorithm could offer additional structure or regularization for discovery of the unknown term in a PINN framework under partial knowledge. Work such as [39], which integrated PINNs with SINDy [26] has already taken steps in this direction. Additionally, UPINNs still struggle with recovering the solution and hidden term when it is at a large or small scale, or recovering oscillatory behaviour. UPINNs can be integrated with recent PINN improvements [116, 171, 184] in order to improve the recovery of the hidden term.

In Chapter 3, we applied UPINNs, which were introduced in the previous chapter, to identify three different types of chemotherapy drug actions. In quantitative systems

pharmacology (QSP), drug actions are often assumed to take one of several forms. In this work, we learn the drug action from data. We generate synthetic data with all three types of drug actions, and successfully use UPINNs to recover the drug action even when data is noisy. Additionally, we show how to apply UPINNs to recover multiple sets of parameters in a differential equation simultaneously. Finally, we apply the method to experimental data (*in vitro*) of cancer cells growing and dying due to applied chemotherapeutic (doxorubicin) from [111]. After validating the method on synthetic data from the QSP model in [111], we recover representations of the drug action for experimental data, and we find that the mathematical model presented in [111] is not expressive enough to capture the recovered drug action. In particular, the learned drug action shows a drug action that is varying with time. Further investigation could show the reasons for this time-variation, and could lead to the introduction of new models for chemotherapy pharmacokinetics. In future work, UPINNs could be used to recover unknown terms in other biomedical applications. Since the introduction of UPINNs, they have been applied to model other biomedical systems such interactions between glucose and lactate [161]. Additionally, studies on the identifiability such as those performed on UDEs [100] could be performed on UPINNs in the context of biomedical applications (or other applications with sparse and noisy data).

In Chapter 4, we introduce C-PINNs, or Conformalized PINNs. Here, we integrate conformal prediction, which is an uncertainty quantification framework, with physics-informed neural networks. Conformal prediction, introduced by [166], allows one to create confidence intervals with provable guarantees on any predictive model via the use of a holdout calibration dataset. We conformalize both the forward problem (solving a DE) in a PINN context, and the inverse problem (parameter fitting). We empirically show that constructed intervals have the expected coverage for both the forward and inverse problem. We validate the method on both sparse and noisy data, as well as the Buckley-Leverett equation, which has a discontinuous solution. In future work, we can extend the method to output adaptive conformal intervals, which could adapt the confidence intervals based on estimates of the model's uncertainty for different inputs. Additionally, there is the possibility of combining UPINNs or UDEs with conformal prediction to make their predictions better suited to high-stakes prediction tasks.

In Chapter 5, we make a contribution to symbolic regression by introducing a method of leveraging dimensional analysis in order to improve symbolic regression performance and generalization. We first investigate the effect of nondimensionalization on the symbolic discovery of algebraic equations. In this case, reducing the number of variables to the minimum needed to fully define the equation, as per the Buckingham Π theorem, greatly improves the performance and generalization of the PySR symbolic regression algorithm [42]. Then, we employ a partial nondimensionalization scheme, guided by Ipsen's

method of nondimensionalization, which allows fine control of the dimensionless variables that appear in both the known and unknown terms in the equation. We apply the method to two differential equations, the rotating bead model and the extended logistic growth equation, where part of the DE is known and part of it is unknown. We find that with a dataset consisting of only 10 solutions, a combination of Universal PINNs and PySR is able to fully recover the true dimensionless form of the equation. In future work, a systematic noise study can be conducted to thoroughly assess the effect of noise on this method. Additionally, an algorithm could be constructed to choose the nondimensionalization automatically rather than manually. Finally, several limitations of the work suggest open research directions: for this work (and many others such as [151, 158]), the dimensional matrix is fully assumed to be known. Additionally, the presence of "lurking" variables that are in the true equation but unknown to the symbolic regression may confound the nondimensionalization. Scaling factors (that have dimensions) that cannot be measured directly but appear in the true equation may further make the nondimensionalization difficult to perform. Finally, we make the assumption that all inputs to the known and unknown terms are known. The analysis may significantly change when these assumptions are loosened.

In Chapter 6, we make a contribution to materials discovery by building a GFlowNet pipeline to search the space of possible catalyst materials to improve the efficiency of hydrogen energy storage systems. We call this pipeline Catalyst GFlowNet. In this pipeline, we integrate action constraints, specialized stability and adsorption energy rewards, and ML-based structure relaxation into a GFlowNet, which is an inference machine that samples catalyst structures proportionally to a reward function. The final goal is to generate a diversity of catalyst surfaces which are both thermodynamically stable and yield a theoretically high reaction rate (equivalent to a low overpotential). We validate the method on the two half-reactions that occur within a water electrolyzer, hydrogen evolution reaction, and oxygen evolution reaction. For each of these reactions, we are able to rediscover the best known catalyst (platinum for HER, ruthenium oxide for OER). We show that the generated samples with the proposed reward are more stable and have lower overpotential than samples selected uniformly across the search space. Future work will expand the search space to discover new catalysts, taking care to ensure that proposed catalysts are synthesizable and testable within an experimental framework. Several additional proxies such as Pourbaix stability could significantly improve the quality of the sampled catalysts. Additionally, we aim to integrate an active learning loop and density functional theory simulations into the sampling.

More broadly, GFlowNets and UPINNs could be combined into a framework for small molecule discovery. A UDE or UPINN could serve as the proxy model, and the GFN could

sample small molecules proportionally to a reward that represents the performance of a drug. Works such as [160] have already integrated reinforcement learning with QSP modelling in order to inform *de novo* drug discovery. The first step would be to combine a QSP model into a GFN framework and validate small molecules in that way, but there could be an opportunity to learn part of the QSP model during training.

References

[1] Atiyeh Ahmadi, Lena Podina, Sebastian Höpfl, and Brian Ingalls. Machine learned summary statistics for bayesian inference of systems biology models parameters: opportunities and challenges. *Current Opinion in Systems Biology*, page 100560, 2025.

[2] Mila AI4Science, Alex Hernandez-Garcia, Alexandre Duval, Alexandra Volokhova, Yoshua Bengio, Divya Sharma, Pierre Luc Carrier, Yasmine Benabed, Michał Koziarski, and Victor Schmidt. Crystal-GFN: Sampling crystals with desirable properties and constraints. *arXiv preprint arXiv:2310.04925*, 2023.

[3] Md Ramjan Ali, Santanu Raut, Susmita Sarkar, and Uttam Ghosh. Unraveling the combined actions of a holling type iii predator–prey model incorporating allee response and memory effects. *Computational and Mathematical Methods*, 3(2):e1130, 2021.

[4] Shaikhah Alkhadhr, Xilun Liu, and Mohamed Almekkawy. Modeling of the forward wave propagation using physics-informed neural networks. In *2021 IEEE International Ultrasonics Symposium (IUS)*, pages 1–4. IEEE, 2021.

[5] RJ Allen, Theodore R Rieger, and Cynthia J Musante. Efficient generation and selection of virtual populations in quantitative systems pharmacology models. *CPT: pharmacometrics & systems pharmacology*, 5(3):140–146, 2016.

[6] William F Ames. *Numerical methods for partial differential equations*. Academic press, 2014.

[7] Ioannis P Androulakis. Towards a comprehensive assessment of qsp models: what would it take? *Journal of Pharmacokinetics and Pharmacodynamics*, 51(5):521–531, 2024.

[8] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[9] A.Z. Arsad, M.A. Hannan, Ali Q. Al-Shetwi, M. Mansur, K.M. Muttaqi, Z.Y. Dong, and F. Blaabjerg. Hydrogen energy storage integrated hybrid renewable energy systems: A review analysis for future research directions. *International Journal of Hydrogen Energy*, 47(39):17285–17312, 2022.

[10] Lazar Atanackovic, Alexander Tong, Bo Wang, Leo J Lee, Yoshua Bengio, and Jason S Hartford. Dyngfn: Towards bayesian inference of gene regulatory networks with gflownets. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 74410–74428. Curran Associates, Inc., 2023.

[11] Douglas Adriano Augusto and Helio JC Barbosa. Symbolic regression via genetic programming. In *Proceedings. Vol. 1. Sixth Brazilian symposium on neural networks*, pages 173–178. IEEE, 2000.

[12] Joseph Bakarji, Jared Callaham, Steven L Brunton, and J Nathan Kutz. Dimensionally consistent learning with buckingham pi. *Nature Computational Science*, 2(12):834–844, 2022.

[13] Chayan Banerjee, Kien Nguyen, Clinton Fookes, and Maziar Raissi. A survey on physics informed reinforcement learning: Review and open problems. *Expert Systems with Applications*, 287:128166, 2025.

[14] G. Bar-Meir. *Basics of Fluid Mechanics*. Potto Proj. Publ., 0.6.2 edition, Apr 2022. doi:10.5281/zenodo.5521908.

[15] Meili Baragatti, Céline Casenave, Bertrand Cloez, David Métivier, and Isabelle Sanchez. Approximate bayesian computation with deep learning and conformal prediction. *Journal of Computational and Graphical Statistics*, 35(1):443–460, 2026.

[16] Britta Basse, Bruce C Baguley, Elaine S Marshall, Wayne R Joseph, Bruce van Brunt, Graeme Wake, and David JN Wall. Modelling cell death in human tumour cell lines exposed to the anticancer drug paclitaxel. *Journal of Mathematical Biology*, 49:329–357, 2004.

[17] Zdenek Belohlav, Petr Zamostny, Petr Kluson, and Jiri Volf. Application of random-search algorithm for regression analysis of catalytic hydrogenations. *The Canadian Journal of Chemical Engineering*, 75(4):735–742, 1997.

[18] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[19] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[20] Karianne J Bergen, Paul A Johnson, Maarten V de Hoop, and Gregory C Beroza. Machine learning for data-driven discovery in solid earth geoscience. *Science*, 363(6433):eaau0323, 2019.

[21] Alan A Berryman. The orgins and evolution of predator-prey theory. *Ecology*, 73(5):1530–1535, 1992.

[22] Bogdan Bochenek and Zbigniew Ustrnul. Machine learning in weather prediction and climate analyses—applications and perspectives. *Atmosphere*, 13(2):180, 2022.

[23] Shannon W. Boettcher. Introduction to green hydrogen. *Chemical Reviews*, 124(23):13095–13098, 2024. PMID: 39659176.

[24] Tedra Bolger, Brydon Eastman, Madeleine Hill, and Gail Wolkowicz. A predator-prey model in the chemostat with holling type ii response function. *Mathematics in Applied Sciences and Engineering*, 1(4):333–354, 2020.

[25] Larisse Bolton, Alain HJJ Cloot, Schalk W Schoombie, and Jacobus P Slabbert. A proposed fractional-order gompertz model and its application to tumour growth data. *Mathematical medicine and biology: a journal of the IMA*, 32(2):187–209, 2015.

[26] 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.

[27] Edgar Buckingham. On physically similar systems; illustrations of the use of dimensional equations. *Physical review*, 4(4):345, 1914.

[28] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[29] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[30] Diogo M Camacho, Katherine M Collins, Rani K Powers, James C Costello, and James J Collins. Next-generation machine learning for biological networks. *Cell*, 173(7):1581–1592, 2018.

[31] Alvaro Carbonero, Alexandre Duval, Victor Schmidt, Santiago Miret, Alex Hernandez-Garcia, Yoshua Bengio, and David Rolnick. On the importance of catalyst-adsorbate 3d interactions for relaxed energy predictions. *arXiv preprint arXiv:2310.06682*, 2023.

[32] Arijit Chakraborty, Abhishek Sivaram, and Venkat Venkatasubramanian. Ai-darwin: A first principles-based model discovery engine using machine learning. *Computers & Chemical Engineering*, 154:107470, 2021.

[33] Souvik Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *CoRR*, abs/2005.10614, 2020.

[34] Abhishek Chandra, Joseph Bakarji, and Daniel M Tartakovsky. Role of physics in physics-informed machine learning. *Journal of Machine Learning for Modeling and Computing*, 5(1), 2024.

[35] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 11(10):6059–6072, 2021.

[36] Shahana Chatterjee, Alexander Davis, Lena Podina, Divya Sharma, Yoshua Bengio, Alexandre Duval, Oleksandr Voznyy, Alex Hernández-Garcia, David Rolnick, and Félix Therrien. Adsorption energies are necessary but not sufficient to identify good catalysts. *arXiv preprint arXiv:2512.05938*, 2025.

[37] Chi Chen and Shyue Ping Ong. A universal graph deep learning interatomic potential for the periodic table. *Nature Computational Science*, 2(11):718–728, 2022.

[38] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018.

[39] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.

[40] J.P. Clark. An Alternate Means to Form Non-Dimensional Products in Dimensional Analysis. In *ASME Turbo Expo: Power Land Sea Air*, volume 86021, page V005T08A003. ASME, 2022.

[41] M. Cranmer. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl, May 2023. arXiv:2305.01582 [astro-ph, physics:physics].

[42] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *NeurIPS 2020*, 2020.

[43] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.

[44] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[45] N Danilovic, Ram Subbaraman, D Strmcnik, Kee-Chul Chang, AP Paulikas, VR Stamenkovic, and Nenad M Markovic. Enhancing the alkaline hydrogen evolution reaction activity through the bifunctionality of ni (oh) 2/metal catalysts. *Angewandte Chemie International Edition*, 51(50):12495–12498, 2012.

[46] Stéphane d'Ascoli, Sören Becker, Alexander Mathis, Philippe Schwaller, and Niki Kilbertus. Odeformer: Symbolic regression of dynamical systems with transformers. *arXiv preprint arXiv:2310.05573*, 2023.

[47] Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. *arXiv preprint arXiv:2402.10309*, 2024.

[48] Ivan Depina, Saket Jain, Sigurdur Mar Valsson, and Hrvoje Gotovac. Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 16(1):21–36, 2022.

[49] Abdallah Derbalah, Hesham Al-Sallami, Chihiro Hasegawa, Abhishek Gulati, and Stephen B Duffull. A framework for simplification of quantitative systems pharmacology models in clinical pharmacology. *British Journal of Clinical Pharmacology*, 88(4):1430–1440, 2022.

[50] Dániel András Drexler, Tamás Ferenci, András Füredi, Gergely Szakács, and Levente Kovács. Experimental data-driven tumor modeling for chemotherapy. *IFAC-PapersOnLine*, 53(2):16245–16250, 2020.

[51] Pankaj Dumka, Rishika Chauhan, Ayush Singh, Gaurav Singh, and Dhananjay Mishra. Implementation of buckingham's pi theorem using python. *Advances in Engineering Software*, 173:103232, 2022.

[52] Alexandre Duval, Simon V Mathis, Chaitanya K Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D Malliaros, Taco Cohen, Pietro Lio, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems. *arXiv preprint arXiv:2312.07511*, 2023.

[53] Alexandre Agm Duval, Victor Schmidt, Alex Hernández-García, Santiago Miret, Fragkiskos D. Malliaros, Yoshua Bengio, and David Rolnick. FAENet: Frame averaging equivariant GNN for materials modeling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 9013–9033. PMLR, 23–29 Jul 2023.

[54] Brydon Eastman, Michelle Przedborski, and Mohammad Kohandel. Reinforcement learning derived chemotherapeutic schedules for robust patient-specific therapy. *Scientific Reports*, 11(1):17882, 2021.

[55] Federica Eduati, Patricia Jaaks, Jessica Wappler, Thorsten Cramer, Christoph A Merten, Mathew J Garnett, and Julio Saez-Rodriguez. Patient-specific logic models of signaling pathways from screenings on cancer biopsies to prioritize personalized combination therapies. *Molecular systems biology*, 16(2):e8664, 2020.

[56] Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6):741–760, 2021.

[57] Lixin Fan, Zhengkai Tu, and Siew Hwa Chan. Recent development of hydrogen and fuel cell technologies: A review. *Energy Reports*, 7:8421–8446, 2021.

[58] Vito Ferro. Assessing flow resistance law in vegetated channels by dimensional analysis and self-similarity. *Flow Measurement and Instrumentation*, 69:101610, 2019.

[59] H. Xue F.M. White. *Fluid Mechanics, 9th Ed.* McGraw-Hill, 2021.

[60] Edwin Fong and Chris C Holmes. Conformal bayesian computation. *Advances in Neural Information Processing Systems*, 34:18268–18279, 2021.

[61] Sergio Freeman and Ertan Agar. The impact of energy storage on the reliability of wind and solar power in new england. *Heliyon*, 10(6), 2024.

[62] Timo Freiesleben and Thomas Grote. Beyond generalization: a theory of robustness in machine learning. *Synthese*, 202(4):109, 2023.

[63] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[64] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34:6790–6802, 2021.

[65] Philip Gerlee. The model muddle: in search of tumor growth laws. *Cancer research*, 73(8):2407–2411, 2013.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[67] Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174, 2024.

[68] G. T. Kasun Kalhara Gunasooriya and Jens K. Nørskov. Analysis of Acid-Stable and Active Oxides for the Oxygen Evolution Reaction. *ACS Energy Letters*, 5(12):3778–3787, December 2020.

[69] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

[70] Gary W Harrison. Global stability of predator-prey interactions. *Journal of Mathematical Biology*, 8(2):159–171, 1979.

[71] Geoffroy Hautier, Anubhav Jain, and Shyue Ping Ong. From the computer to the laboratory: materials discovery and design using first-principles calculations. *Journal of Materials Science*, 47:7317–7340, 2012.

[72] Yunlong He, Zhenye Kang, Jing Li, Yawei Li, and Xinlong Tian. Recent progress of manganese dioxide based electrocatalysts for the oxygen evolution reaction. *Industrial Chemistry & Materials*, 1(3):312–331, 2023.

[73] Alex Hernández-García, Nikita Saxena, Moksh Jain, Cheng-Hao Liu, and Yoshua Bengio. Multi-fidelity active learning with GFlownets. *Transactions on Machine Learning Research*, 2024. Expert Certification.

[74] William W Hsieh. *Machine learning methods in the environmental sciences: Neural networks and kernels*. Cambridge university press, 2009.

[75] Jiaqi Hua, Yingguang Li, Changqing Liu, Peng Wan, and Xu Liu. Physics-informed neural networks with weighted losses by uncertainty evaluation for accurate and stable prediction of manufacturing systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[76] Atsushi Ishikawa. Heterogeneous catalyst design by generative adversarial network and first-principles based microkinetics. *Scientific Reports*, 12(1):11657, 2022.

[77] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.

[78] Ameya D Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, 2022.

[79] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand

Ceder, and Kristin a. Persson. The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[80] Angela M Jarrett, Alay Shah, Meghan J Bloom, Matthew T McKenna, David A Hormuth, Thomas E Yankeelov, and Anna G Sorace. Experimentally-driven mathematical modeling to improve combination targeted and cytotoxic therapy for her2+ breast cancer. *Scientific reports*, 9(1):12830, 2019.

[81] Jouhyun Jeon, Satra Nim, Joan Teyra, Alessandro Datti, Jeffrey L Wrana, Sachdev S Sidhu, Jason Moffat, and Philip M Kim. A systematic approach to identify novel cancer drug targets using machine learning, inhibitor design and high-throughput screening. *Genome medicine*, 6(7):57, 2014.

[82] Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36):8098–8106, 2021.

[83] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[84] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[85] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *arXiv preprint arXiv:2204.10532*, 2022.

[86] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

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

[88] M Kohandel, S Sivaloganathan, and A Oza. Mathematical modeling of ovarian cancer treatments: sequencing of surgery and chemotherapy. *Journal of theoretical biology*, 242(1):62–68, 2006.

[89] Michael F Korns. A baseline symbolic regression algorithm. *Genetic Programming Theory and Practice X*, pages 117–137, 2013.

[90] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[91] MichałKoziarski, Andrei Rekesh, Dmytro Shevchuk, Almer van der Sloot, Piotr Gaiński, Yoshua Bengio, Cheng-Hao Liu, Mike Tyers, and Robert A. Batey. Rgfn: Synthesizable molecular generation using gflownets. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 46908–46955. Curran Associates, Inc., 2024.

[92] M. Kohandel L. Podina, B. Eastman. Universal Physics-Informed Neural Networks: Symbolic Differential Operator Discovery with Sparse Data. In *Proc. 40th Int. Conf. Mach. Learn.*, volume 202, pages 27948–27956. PMLR, Jul 2023.

[93] Romain Lacombe, Lucas Hendren, and Khalid El-Awady. Adsorbrl: Deep multi-objective reinforcement learning for inverse catalysts design, 2023.

[94] Nung Siong Lai, Yi Shen Tew, Xialin Zhong, Jun Yin, Jiali Li, Binhang Yan, and Xiaonan Wang. Artificial intelligence (ai) workflow for catalyst design and optimization. *Industrial & Engineering Chemistry Research*, 62(43):17835–17848, 2023.

[95] Wenqiang Li, Weijun Li, Lina Yu, Min Wu, Linjun Sun, Jingyi Liu, Yanjie Li, Shu Wei, Yusong Deng, and Meilan Hao. A neural-guided dynamic symbolic network for exploring mathematical expressions from data, 2024.

[96] Ying Li and Fangjun Mei. Deep learning-based method coupled with small sample learning for solving partial differential equations. *Multimedia Tools and Applications*, 80(11):17391–17413, 2021.

[97] Taeho Lim and Soo-Kil Kim. Non-precious hydrogen evolution reaction catalysts: Stepping forward to practical polymer electrolyte membrane-based zero-gap water electrolyzers. *Chemical Engineering Journal*, 433:133681, 2022.

[98] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[99] 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.

[100] Torkel E Loman and Ruth E Baker. Functional and parametric identifiability for universal differential equations applied to chemical reaction networks. *arXiv preprint arXiv:2510.14140*, 2025.

[101] T. Lazebnik L.S. Keren, A. Liberzon. A Computational Framework for Physics-Informed Symbolic Regression with Straightforward Integration of Domain Knowledge. *Sci. Rep.*, 13(1):1249, 2023.

[102] James Lu, Brendan Bender, Jin Y Jin, and Yuanfang Guan. Deep learning prediction of patient response time course from early data via neural-pharmacokinetic/pharmacodynamic modelling. *Nature machine intelligence*, 3(8):696–704, 2021.

[103] James Lu, Kaiwen Deng, Xinyuan Zhang, Gengbo Liu, and Yuanfang Guan. Neural-ode for pharmacokinetics modeling and its advantage to alternative machine learning models in predicting new dosing regimens. *Iscience*, 24(7):102804, 2021.

[104] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[105] A.V. Kalyuzhnaya M. Maslyaev, A. Hvatov. Partial Differential Equations Discovery with EPDE Framework: Application for Real and Synthetic Data. *J. Comput. Sci.*, page 101345, 2021.

[106] AM Măgurean. Three-dimensional numerical model for heat transfer dynamic analysis for the building-soil interaction by applying the similarity theory. *Thermal Science and Engineering Progress*, 43:102013, 2023.

[107] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

[108] Polina Mamoshina, Marina Volosnikova, Ivan V Ozerov, Evgeny Putin, Ekaterina Skibina, Franco Cortese, and Alex Zhavoronkov. Machine learning on human muscle transcriptomic data for biomarker discovery and tissue-specific drug target identification. *Frontiers in genetics*, 9:378508, 2018.

[109] Isabela C. Man, Hai-Yan Su, Federico Calle-Vallejo, Heine A. Hansen, José I. Martínez, Nilay G. Inoglu, John Kitchin, Thomas F. Jaramillo, Jens K. Nørskov, and Jan Rossmeisl. Universality in oxygen evolution electrocatalysis on oxide surfaces. *ChemCatChem*, 3(7):1159–1165, 2011.

[110] Ben McKay, Mark J Willis, and Geoffrey W Barton. Using a tree structured genetic algorithm to perform symbolic regression. In *First international conference on genetic algorithms in engineering systems: innovations and applications*, pages 487–492. IET, 1995.

[111] Matthew T McKenna, Jared A Weis, Stephanie L Barnes, Darren R Tyson, Michael I Miga, Vito Quaranta, and Thomas E Yankeelov. A predictive mathematical modeling approach for the study of doxorubicin treatment in triple negative breast cancer. *Scientific reports*, 7(1):5725, 2017.

[112] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401:109020, 2020.

[113] Dong Hyeon Mok and Seoin Back. Generative pretrained transformer for heterogeneous catalysts. *Journal of the American Chemical Society*, 146(49):33712–33722, 2024. PMID: 39576215.

[114] Christian Moya and Guang Lin. Dae-pinn: A physics-informed neural network model for simulating differential-algebraic equations with application to power networks. *arXiv preprint arXiv:2109.04304*, 2021.

[115] Christian Moya, Amirhossein Mollaali, Zecheng Zhang, Lu Lu, and Guang Lin. Conformalized-deeponet: A distribution-free framework for uncertainty quantification in deep operator networks. *Physica D: Nonlinear Phenomena*, 471:134418, 2025.

[116] Pouyan Nasiri and Roozbeh Dargazany. Reduced-pinn: An integration-based physics-informed neural networks for stiff odes. *arXiv preprint arXiv:2208.12045*, 2022.

[117] J. K. Nørskov, T. Bligaard, A. Logadottir, J. R. Kitchin, J. G. Chen, S. Pandelov, and U. Stimming. Trends in the exchange current for hydrogen evolution. *Journal of The Electrochemical Society*, 152(3):J23, jan 2005.

[118] Janghoon Ock, Chakradhar Guntuboina, and Amir Barati Farimani. Catalyst energy prediction with catberta: Unveiling feature exploration strategies through large language models. *ACS Catalysis*, 13(24):16032–16044, 2023.

[119] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L Chevrier, Kristin A Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

[120] Ahmed I Osman, Neha Mehta, Ahmed M Elgarahy, Mahmoud Hefny, Amer Al-Hinai, Ala'a H Al-Muhtaseb, and David W Rooney. Hydrogen production, storage, utilisation and environmental impacts: a review. *Environmental Chemistry Letters*, pages 1–36, 2022.

[121] John Carl Panetta and K Renee Fister. Optimal control applied to competing chemotherapeutic cell-kill strategies. *SIAM Journal on Applied Mathematics*, 63(6):1954–1971, 2003.

[122] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 345–356. Springer, 2002.

[123] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[124] Bent Petersen, Morten Pedersen, and Lars Kai Hansen. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *Available on Papers with Code*, 2021.

[125] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[126] Giovani Manuel Pitra and KS Sastry Musti. Duck curve with renewable energies and storage technologies. In *2021 13th International conference on computational intelligence and communication networks (CICN)*, pages 66–71. IEEE, 2021.

[127] Lena Podina, Diba Darooneh, Joshveer Grewal, and Mohammad Kohandel. Enhancing symbolic regression and universal physics-informed neural networks with dimensional analysis. *arXiv preprint arXiv:2411.15919*, 2024.

[128] Lena Podina, Brydon Eastman, and Mohammad Kohandel. Universal physics-informed neural networks: symbolic differential operator discovery with sparse data. In *International Conference on Machine Learning*, pages 27948–27956. PMLR, 2023.

[129] Lena Podina, Ali Ghodsi, and Mohammad Kohandel. Learning chemotherapy drug action via universal physics-informed neural networks. *Pharmaceutical Research*, 42(4):593–612, 2025.

[130] Lena Podina, Christina Humer, Alexandre Duval, Victor Schmidt, Ali Ramlaoui, Shahana Chatterjee, Yoshua Bengio, Alex Hernandez-Garcia, David Rolnick, and Félix Therrien. Catalyst gflownet for electrocatalyst design: A hydrogen evolution reaction case study. *arXiv preprint arXiv:2510.02142*, 2025.

[131] Lena Podina, Mahdi Torabi Rad, and Mohammad Kohandel. Conformalized physics-informed neural networks. *arXiv preprint arXiv:2405.08111*, 2024.

[132] Tomaso Poggio, Qianli Liao, Brando Miranda, Andrzej Banburski, Xavier Boix, and Jack Hidary. Theory iiib: Generalization in deep networks. *arXiv preprint arXiv:1806.11379*, 2018.

[133] Sebastian Polak, Zofia Tylutki, Mark Holbrook, and Barbara Wiśniowska. Better prediction of the local concentration–effect relationship: the role of physiologically based pharmacokinetics and quantitative systems pharmacology and toxicology in the evolution of model-informed drug discovery and development. *Drug discovery today*, 24(7):1344–1354, 2019.

[134] Michelle Przedborski, Munisha Smalley, Saravanan Thiyagarajan, Aaron Goldman, and Mohammad Kohandel. Systems biology informed neural networks (sbinn) predict response and novel combinations for pd-1 checkpoint blockade. *Communications Biology*, 4(1):877, 2021.

[135] Bhanwar Lal Puniya, Rada Amin, Bailee Lichter, Robert Moore, Alex Ciurej, Sydney J Bennett, Ab Rauf Shah, Matteo Barberis, and Tomáš Helikar. Integrative computational approach identifies drug targets in cd4+ t-cell-mediated immune disorders. *NPJ systems biology and applications*, 7(1):4, 2021.

[136] Xin Qi, Dawei Zhang, Fan Wang, Wuxia Bi, and Mingda Lu. Ndawl-pinn: a new non-dimensionalization and multi-task learning approach for efficient training of physics-informed neural networks to solve the shallow water equations. *Engineering Applications of Computational Fluid Mechanics*, 19(1):2535015, 2025.

[137] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. Prediction of dynamical systems by symbolic regression. *Physical Review E*, 94(1), July 2016.

[138] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[139] Rahul Rahaman et al. Uncertainty quantification and deep ensembles. *Advances in neural information processing systems*, 34:20063–20075, 2021.

[140] Maziar Raissi, Paris Perdikaris, and George 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.

[141] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[142] Sameera Ramasinghe, Lachlan E MacDonald, and Simon Lucey. On the frequency-bias of coordinate-mlps. *Advances in Neural Information Processing Systems*, 35:796–809, 2022.

[143] Oliver Schilter, Alain Vaucher, Philippe Schwaller, and Teodoro Laino. Designing catalysts with deep generative models and computational data. a case study for suzuki cross coupling reactions. *Digital discovery*, 2(3):728–735, 2023.

[144] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[145] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and

R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[146] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.

[147] Jorge Sola and Joaquin Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science*, 44(3):1464–1468, 1997.

[148] A.A. Sonin. The Physical Basis of Dimensional Analysis. *Department of Mechanical Engineering, MIT*, page 1–36, 2001.

[149] Peter K Sorger, Sandra RB Allerheiligen, Darrell R Abernethy, Russ B Altman, Kim LR Brouwer, Andrea Califano, David Z D'Argenio, Ravi Iyengar, William J Jusko, Richard Lalonde, et al. Quantitative and systems pharmacology in the post-genomic era: new approaches to discovering drugs and understanding therapeutic mechanisms. In *An NIH white paper by the QSP workshop group*, volume 48, pages 1–47. NIH Bethesda Bethesda, MD, 2011.

[150] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. *arXiv preprint arXiv:2205.13134*, 2022.

[151] Wassim Tenachi, Rodrigo Ibata, and Foivos I Diakogiannis. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, 2023.

[152] Jahanvi Thakur, Peeyush Phogat, et al. Catalyst design for efficient water splitting: A comprehensive review of challenges and opportunities. *Fuel*, 392:134954, 2025.

[153] Dirk Thierens. Scalability problems of simple genetic algorithms. *Evolutionary computation*, 7(4):331–352, 1999.

[154] Juan Diego Toscano, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. From pinns to pikans: Recent advances in physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1(1):15, 2025.

[155] Tiffany A Traina, Ute Dugan, Brian Higgins, Kenneth Kolinsky, Maria Theodoulou, Clifford A Hudis, and Larry Norton. Optimizing chemotherapy dose and schedule by norton-simon mathematical modeling. *Breast disease*, 31(1):7–18, 2010.

[156] Richard Tran, Janice Lan, Muhammed Shuaibi, Brandon M Wood, Siddharth Goyal, Abhishek Das, Javier Heras-Domingo, Adeesh Kolluru, Ammar Rizvi, Nima Shoghi, et al. The open catalyst 2022 (oc22) dataset and challenges for oxide electrocatalysts. *ACS Catalysis*, 13(5):3066–3084, 2023.

[157] Sergio Trasatti. Work function, electronegativity, and electrochemical behaviour of metals: Iii. electrolytic hydrogen evolution in acid solutions. *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry*, 39(1):163–184, 1972.

[158] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[159] Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. Symbolicgpt: A generative transformer model for symbolic regression. *arXiv preprint arXiv:2106.14131*, 2021.

[160] Helle W van den Maagdenberg, Jikke de Mol van Otterloo, JG Coen van Hasselt, Piet H Van Der Graaf, and Gerard JP van Westen. Integrating pharmacokinetics and quantitative systems pharmacology approaches in generative drug design. *Journal of Chemical Information and Modeling*, 65(10):4783–4796, 2025.

[161] Shadi Vandvajdi, Yuannong Mao, Mahla Poudineh, and Mohammad Kohandel. Investigating glucose-lactate metabolism in glioblastoma multiforme via universal physics-informed neural networks. *Mathematical Biosciences and Engineering*, 22(9):2486–2505, 2025.

[162] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[163] Francesco Verdoja and Ville Kyrki. Notes on the behavior of mc dropout. *arXiv preprint arXiv:2008.02627*, 2020.

[164] Marco Virgolin and Solon P Pissis. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*, 2022.

[165] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[166] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[167] Klaus Wandelt. *Encyclopedia of interfacial chemistry: surface science and electrochemistry*. Elsevier, 2018.

[168] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[169] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[170] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[171] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.

[172] Yaning Wang, Hao Zhu, Rajanikanth Madabushi, Qi Liu, Shiew-Mei Huang, and Issam Zineh. Model-informed drug development: current us regulatory practice and future considerations. *Clinical Pharmacology & Therapeutics*, 105(4):899–911, 2019.

[173] Keng Boon Wee and Baltazar D Aguda. Akt versus p53 in a network of oncogenes and tumor suppressor genes regulating cell survival and death. *Biophysical journal*, 91(3):857–865, 2006.

[174] Jeffrey West and Paul K Newton. Cellular interactions constrain tumor growth. *Proceedings of the National Academy of Sciences*, 116(6):1918–1923, 2019.

[175] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, 55(4):1–37, 2022.

[176] Chi Heem Wong, Kien Wei Siah, and Andrew W Lo. Estimation of clinical trial success rates and related parameters. *Biostatistics*, 20(2):273–286, 2019.

[177] Hongyuan Yang, Matthias Driess, and Prashanth W Menezes. Self-supported electrocatalysts for practical water electrolysis. *Advanced energy materials*, 11(39):2102074, 2021.

[178] Kuang Yang, Haifan Liao, Bo Xu, Qiuxiang Chen, Zhenghui Hou, and Haijun Wang. Data-driven dryout prediction in helical-coiled once-through steam generator: A physics-informed approach leveraging the buckingham pi theorem. *Energy*, 294:130822, 2024.

[179] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[180] 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):e1007575, 2020.

[181] Lei Yuan, Yi-Qing Ni, Xiang-Yun Deng, and Shuo Hao. A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462:111260, 2022.

[182] Bastian Zapf, Johannes Haubner, Miroslav Kuchta, Geir Ringstad, Per Kristian Eide, and Kent-Andre Mardal. Investigating molecular transport in the human brain from mri with physics-informed neural networks. *Scientific Reports*, 12(1):15475, 2022.

[183] Tongli Zhang, Ioannis P Androulakis, Peter Bonate, Limei Cheng, Tomáš Helikar, Jaimit Parikh, Christopher Rackauckas, Kalyanasundaram Subramanian, and Carolyn R Cho. Two heads are better than one: current landscape of integrating qsp and machine learning: an isop qsp sig white paper by the working group on the integration of quantitative systems pharmacology and machine learning. *Journal of Pharmacokinetics and Pharmacodynamics*, 49(1):5–18, 2022.

[184] Anni Zhao, Davood Fattahi, and Xiao Hu. Physics-informed neural networks for physiological signal processing and modeling: a narrative review. *Physiological measurement*, 46(7):07TR02, 2025.

[185] C Lawrence Zitnick, Lowik Chanussot, Abhishek Das, Siddharth Goyal, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Thibaut Lavril, Aini Palizhati, Morgane

Riviere, et al. An introduction to electrocatalyst design using machine learning for renewable energy storage. *arXiv preprint arXiv:2010.09435*, 2020.

[186] Hao Zou, Haochen Zhao, Mingming Lu, Jiong Wang, Zeyu Deng, and Jianxin Wang. Predicting thermodynamic stability of inorganic compounds using ensemble machine learning based on electron configuration. *Nature Communications*, 16(1):203, 2025.

[187] Zongren Zou, Xuhui Meng, and George Em Karniadakis. Correcting model misspecification in physics-informed neural networks (pinns). *Journal of Computational Physics*, 505:112918, 2024.