A COMPARATIVE STUDY OF NEURAL ODE AND UNIVERSAL ODE MODELS IN SOLVING CHANDRASEKHAR'S WHITE DWARF EQUATION

Raymundo Vazquez Martinez
National Autonomous University of Mexico
raymundo.vm@ciencias.unam.mx

Raj Abhijit Dandekar
Vizuara AI Labs
Massachusetts Institute of Technology (prior)
raj@vizuara.com

Rajat Dandekar
Vizuara AI Labs
Purdue University (prior)
rajatdandekar@vizuara.com

Sreedath Panat
Vizuara AI Labs
Massachusetts Institute of Technology (prior)
sreedath@vizuara.com

Abstract

In this study, we explore the application of two pillars of Scientific Machine Learning—Neural Ordinary Differential Equations (Neural ODEs) and Universal Differential Equations (UDEs)—to a cornerstone of astrophysical theory: the Chandrasekhar White Dwarf Equation (CWDE). The CWDE is fundamental for understanding the life cycle of a star and describes the relationship between the density of the white dwarf and its distance from the core. Despite the growing importance of SciML, the systematic exploration of these techniques in astrophysics, particularly in modeling complex ODEs like the CWDE, remains largely unexplored. In this study, we bridge that gap by demonstrating how Neural ODEs and UDEs can be employed for both accurate prediction and reliable long-term forecasting of the CWDE. Furthermore, we introduce the "forecasting breakdown point"—the time at which forecasting fails for both Neural ODEs and UDEs. Through rigorous hyperparameter optimization testing, we assess neural network architectures, activation functions, and optimizer configurations to determine the best performance. This study offers a new lens to understand the physics of white dwarfs and paves the way for future research on using SciML frameworks for forecasting tasks across a range of scientific domains.

1 Introduction

Scientific Machine Learning (Scientific ML) is a growing field with a wide range of applications in various fields such as epidemiology, gene expression, optics, circuit modeling, quantum circuits and fluid mechanics [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. This field of Scientific ML leverages the interpretability of scientific structures like ODEs/PDEs along with the expressivity of neural networks. Broadly, the rise of Scientific Machine Learning can be attributed to three popular methodologies:

- Neural Ordinary Differential Equations: The entire forward pass of an ODE/PDE is replaced with neural networks. We perform backpropagation through the neural network augmented ODE/PDE. In doing so, we find the optimal values of the neural network parameters. [15, 16, 17, 18]

Foundation Models for Science Workshop,38th Conference on Neural Information Processing Systems (NeurIPS 2024).

- Universal Differential Equations (UDEs): In contrast to Neural ODEs, only certain terms of the ODE/PDEs are replaced with neural networks. We then discover these terms by optimizing the neural network parameters. Universal Differential Equations can be used to correct existing underlying ODEs/PDEs as well as to discover new, missing physics. [19, 20, 21, 22]
- Physics Informed Neural Networks (PINNs): PINNs are predominantly used as an alternative to traditional ODE/PDE solvers to solve an entire ODE/PDE. We replace the function variable with a neural network and the loss function is determined by the ODE/PDE solution and the boundary conditions. When we minimize the loss function, we automatically find the optimum solution to the ODE/PDE. [23, 24, 25, 26]

Despite the advances of Scientific ML in various fields, there is a lack of applying Scientific ML methods in the field of astronomy. Although there are a few studies aimed at applying Neural ODEs to astronomy problems [27, 28, 29], there is no study investigating the application of Universal Differential Equations (UDEs) to astronomy or astrophysics problems.

In particular, the following questions are still unanswered:

- In the spirit of UDEs, can we replace certain terms of an astronomical ODE system with neural networks and recover them?
- How does the Neural ODE prediction compare with the UDE prediction?
- Can we do forecasting on the system of ODEs with Neural ODEs and UDEs?
- Are UDEs better at forecasting than Neural ODEs?

We aim to answer these questions by looking at a foundational ODE in astronomy and astrophysics: the Chandrasekhar White Dwarf Equation (CWDE) [30, 31]. The CWDE describes the relationship between the density of the white dwarf and it's distance from the center [30, 31]. This equation is fundamental for understanding the life cycle of a star. Using this equation, we can potentially predict when the star will collapse and transform into a supernova.

We use the advanced Scientific Machine Learning libraries provided by the Julia Programming Language [32, 33, 34, 35]. Through a robust hyperparameter optimization testing, we provide insights on the neural network architecture, activation functions and optimizers which provide the best results. We show that both Neural ODEs and UDEs can be used effectively for both prediction as well as forecasting of the Chandrasekhar's white dwarf ODE system. More importantly, we introduce the "forecasting breakdown point" - the time at which forecasting fails for the Neural ODE and UDE models. This provides an insight into the applicability of Scientific Machine Learning frameworks in forecasting tasks.

The paper is structured as follows. We start by presenting the methodology and detailed description for Neural ODEs and UDEs. Subsequently, we present the prediction and forecasting results for the Neural ODEs and UDEs. Finally, we conclude with a detailed discussion of our results, and the future scope of applying Scientific ML methods in astronomy and astrophysics.

2 Methodology

According to Chandrasekhar the equation that governs the structure of degenerate matter in gravitational equilibrium is given by the second-order ordinary differential equation [36]

$$\frac{1}{\eta^2}\frac{d}{d\eta}\left(\eta^2\frac{d\varphi}{d\eta}\right)+\left(\varphi^2-C\right)^{3/2}=0 \tag{1}$$

with initial conditions

$$\varphi(0)=1, \quad \varphi'(0)=0$$

This equations is one of Emden type, and therefore a solution exists in the neighborhood of $\eta = 0$ [37]. This equation exhibits the density φ of the white dwarf as a function of the dimensionless radius η. Particularly, the variables η and φ are expected to take real values due to their physical meaning. From this fact, we can entail more restrictions on the behavior of φ and η such as their bounds

$$1 \leq \varphi \leq \sqrt{C},$$

$$0 \leq \eta \leq \eta_\infty$$

Moreover, the density function is decreasing and tends to the lower bound $\sqrt{C}$, i.e.

$$\lim_{\eta \to \infty} \varphi(\eta) = \sqrt{C} \tag{2}$$

For the computational implementations, the constant C was set to 0.01. The ODE (1) was reformulated as a system of first order ODEs

$$\frac{d\varphi}{d\eta} = \theta \tag{3a}$$

$$\frac{d\theta}{d\eta} = -\frac{2}{\eta}\theta - (\varphi^2 - C)^{3/2} \tag{3b}$$

The finite-length η interval in which Chandrasekhar's white dwarf equation is solvable was obtained by implementing a numerical approach in the Julia programming language. For this C value, the set of valid values obtained for η was $D_f = \{\eta \in \mathbb{R} : 0.05 \leq \eta \leq 5.325\} = [0.05, 5.325]$. Subsequently, the domain D_f was discretized into 100 equally spaced η values. For these η points, the values for both φ and φ' were saved from the numerical calculation of the ODE, resulting in synthetic data characterizing the white dwarf for this fixed C. Additionally, noise was induced into the synthetic data with varying standard deviations, resulting in different training datasets. Specifically, the standard deviations for the added noise were 7% and 35% regarding the synthetic data. These datasets were labeled as moderate-noise data and high-noise data, respectively, while the synthetic data without any added noise was labeled as no-noise data. For the training routines different subsets of these datasets were used to test the forecasting capability of the neural network models. Particularly, the training routines were implemented with the entire, 90%, 80%, 40%, 20%, and 10% of the mentioned datasets.

2.1 Neural ODEs

Neural Ordinary Differential Equations (Neural ODEs) are a class of models that represent continuous-depth neural networks. Introduced by [15], Neural ODEs have opened up new possibilities in modelling continuous processes by using ordinary differential equations (ODEs) to define the evolution of hidden states in neural networks. Neural ODEs are a subset of the broader spectrum of Scientific Machine Learning and Physics Informed Machine Learning. The key idea behind Neural ODEs is to use a neural network to approximate the solution of an ODE, thereby allowing for flexible modelling of continuous-time dynamics [38, 39, 40, 41, 42, 43].

In a traditional neural network, hidden states are updated using discrete layers. In contrast, Neural ODEs use a continuous transformation defined by an ordinary differential equation:

$$\frac{dh}{dt} = f(h(t), t, \theta) \tag{4}$$

Where, $h(t)$ is the hidden state at time *t*, f is a neural network parameterized by θ and the hidden state evolves according to the function *f*.

In this work, the selection of the parameters of the Neural ODE model were obtained after a robust search over a range of possible values specific to the training data used.[1] Regarding the entire no-noise dataset, the selected hyperparameters for the Neural ODE model are shown in table 1.

[1]Review Appendix A for the specific hyperparameters used for each training dataset employed in this work.

Table 1: Neural ODE range of hyperparameters on training data (no-noise). Hyperparameters for the training routine with the entire available data.

Hyperparameter	Values	Search Range
t_{span}	(0.05, 5.325)	(0,0.5)-(0,10.0)
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	0.01,0.02,0.2,0.05,0.1,0.005, 0.006
Hidden units	160	15, 25, 50, 100, 160, 240
Number of Epochs	Adam: 80 & BFGS: 100	50-4000
Loss	4.11e-4	(0,0.2)

2.2 UDEs

UDEs (Universal Differential Equations) introduced by [19], combine traditional differential equations with machine learning models, such as neural networks, to create a more flexible and powerful tool for modelling complex systems. This approach integrates the robustness of classical differential equations with the adaptability of neural networks, allowing for more accurate and efficient modelling of systems with unknown or partially known dynamics. UDEs offer improved predictive power by combining data-driven approaches with physical laws [44, 45, 46, 47]. This is particularly useful in scenarios where purely data-driven models might overfit or fail to generalize. The physical laws embedded in UDEs constrain the learning process, ensuring that the model adheres to known scientific principles. Compared to purely data-driven models, UDEs often require fewer data points to achieve high accuracy. The known differential equations provide a strong prior that guides the learning process, reducing the amount of data needed for training. This efficiency makes UDEs suitable for applications with limited data availability. The UDE model for the Chandrasekhar's white dwarf equation defined in this work, employed the linear θ terms in (3) as the ground truth model or physical law, as shown in equation (5)

$$\frac{d\varphi}{d\eta} = \theta + NN_1(P, U) \tag{5a}$$

$$\frac{d\theta}{d\eta} = -\frac{2}{\eta}\theta + NN_2(P, U) \tag{5b}$$

Where P are the parameters of the Neural Network (NN) architecture, and $U = (\varphi(\eta), \varphi'(\eta))$ are the input parameters.
The performance of the trained UDE model can be observed further from the recovered interaction or missing term in the original ODE model (3), i.e $NN_1(P_{trained}, U)$ and $NN_2(P_{trained}, U)$

Hyperparameter tuning is a crucial aspect of the UDE model (and machine learning models in general). In this work, the model's parameters were selected after a robust search over a range of possible values specific to the dataset used for training. Regarding the entire no-noise dataset, the selected hyperparameters for the UDE model are shown in table 2.

Table 2: UDE range of hyperparameters on training data (no-noise). Hyperparameters for the training routine with the entire available data.

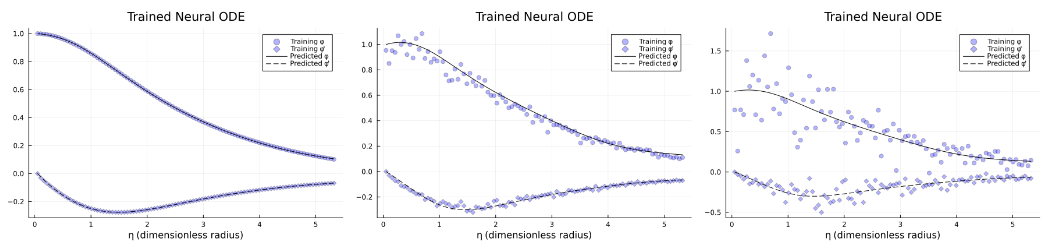
Hyperparameter	Values	Search Range
t_{span}	(0.05, 5.325)	(0,0.5)-(0,10.0)
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	0.01,0.2,0.001,0.1,0.006, 0.5
Hidden units	15	15,25,50,100
Number of Epochs	Adam: 300 & BFGS: 1000	50-4000
Loss	7.43e-8	(0,0.2)

3 Results

Six cases, corresponding to different percentages of the available datasets (no-noise, moderate-noise at 7% standard deviation, and high-noise at 35% standard deviation), were used for training the deep learning-based models. The results for the Neural ODE and UDE models trained on the full datasets and on 80% of the data available are presented in the main text, along with the breakpoints for the no-noise fraction. Additionally, summary tables are provided for the results across the three dataset fractions: no-noise, moderate-noise, and high-noise. A detailed view of the performance and breaking points of the Neural ODE and UDE models for all data sets is available in Appendix B.

3.1 Training in the full domain (100 η points)

First, the implementation of the Neural ODE and UDE models for the Chandrasekhar's White Dwarf equation (CWDE) were performed in the full domain. The three datasets were implemented: no-noise, moderate-noise and high-noise data. The results for the Neural ODE for these training sets are shown in figure 1:



(a) (b) (c)

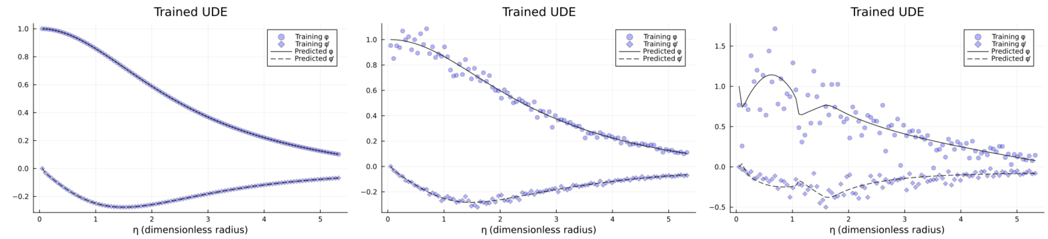
Figure 1: Comparison of the Neural ODE approximation for the Chandrasekhar's white dwarf model. The training of the Neural ODE was performed with varying noise added to the synthetic data in the full solution domain. These training datasets encompassed the values for φ and φ' at the 100 equally spaced η points with varied noise addition. Each figure shows the results for the different training sets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

We can observe from the graphs in Figure 1 that the Neural ODE learns the behavior of the Chandrasekhar's white dwarf equation for both φ and φ'. Even with the addition of moderate and high noise into the dataset, the Neural ODE is still capable of effectively learning the behaviour of the density and its derivative function. However, for the high-noise dataset, the Neural ODE misses the decreasing behaviour of the white dwarf's density function, and it predicts values larger than the initial condition $\varphi(0) = 1$. One distinctive aspect of Chandrasekhar's white Dwarf model is the convergence of the density function φ to the square root of the parameter C when η approaches the limit where it is defined (5.325 for our training dataset). This convergence is replicated by the Neural ODE approximation for these three datasets.

The UDE implementation for the Chandrasekhar's model is presented in figure 2. In this figure, we can observe that the trained UDE model approximates perfectly the training data for the synthetic set, even with the addition of moderate data (standard deviation of 7%), the UDE model can express precisely the behaviour of φ and φ'. For the addition of high-noise in the data (standard deviation of 35%), the UDE seems to overfit the training data, leading to a misinterpretation of the φ and φ' functions for $\eta \in (0, 1)$. In spite of this, the UDE recovers the converging nature of the density φ to the square root of C at the bounding η value.

The performance of the UDE can be observed further from the recovered interaction or missing term in the original ODE model. In this case, the missing term happens to be $-(\varphi^2 - C)^{3/2}$ and was recovered from UDE model after training. [2]

[2] The recovered term was specific of the dataset used to trained the model. It may not be exact to the analytical expression for the missing term, since the neural network component is learning it out of noisy or partial data. Visit Appendix B to review plots (Neural ODE and UDE approximation, forecasting, and missing term recovered plots) obtained for all datasets in this work.

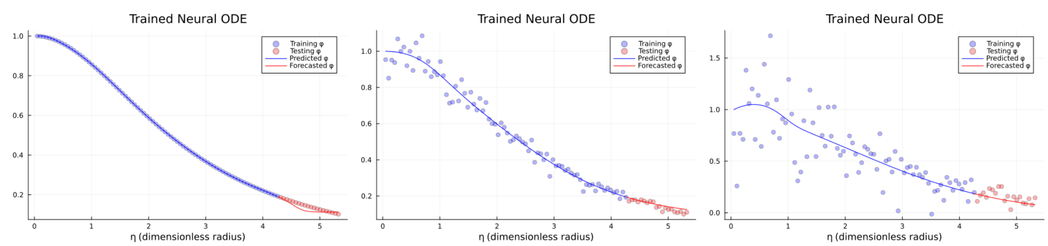


(a) (b) (c)

Figure 2: Comparison of the UDE approximation for the Chandrasekhar's white dwarf equation. The training of the UDE model was performed with varying noise added to the synthetic data in the full solution domain. These training datasets encompassed the values for φ and φ' of the 100 equally spaced η points with varied noise addition: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with standard deviation of 7%. (c) High-noise dataset with standard deviation of 35%.

3.2 Training with 80% of the full available data and forecasting

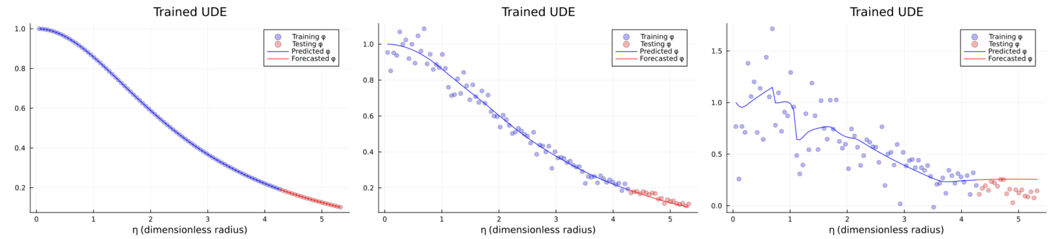
The Neural ODEs and UDEs were trained with smaller data subsets to further evaluate their forecasting capabilities. The previous subsets of no-noise, moderate-noise, and high-noise training datasets were trimmed, forming new training subsets including the φ and φ' values corresponding to the first 80 η points of the full domain. The results for the Neural ODE can be seen in the graphics in figure 3. The trained Neural ODE approximates the training φ data perfectly, but it is slightly off in forecasting the convergence of the density to $\sqrt{C}$. For the moderate-noise data, the Neural ODE successfully reproduces the behaviour of the φ function and its convergence to the square root of C. Finally, for the high-noise data, the Neural ODE manages to recover the shape of the φ function and its convergence out of this noisy training data.



(a) (b) (c)

Figure 3: Comparison of the Neural ODE approximation and forecasting for the Chandrasekhar's white dwarf model. The Neural ODE was trained with varying levels of noise added to the synthetic data. These training data subsets included the values for φ and φ' with different noise levels for the first 80 equally spaced η points of the solution domain. The forecasted φ corresponding to the remaining 20% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

In figure 4, we observe that the UDE accurately approximates the training data and forecasts the unseen data for the no-noise dataset. Similarly, the UDE model performs perfectly for the moderate-noise dataset. However, with the addition of high-noise to the dataset, there is a noticeable breakdown in UDE performance. The UDE tends to overfit abruptly the training data and fails forecasting the unseen values (testing data).

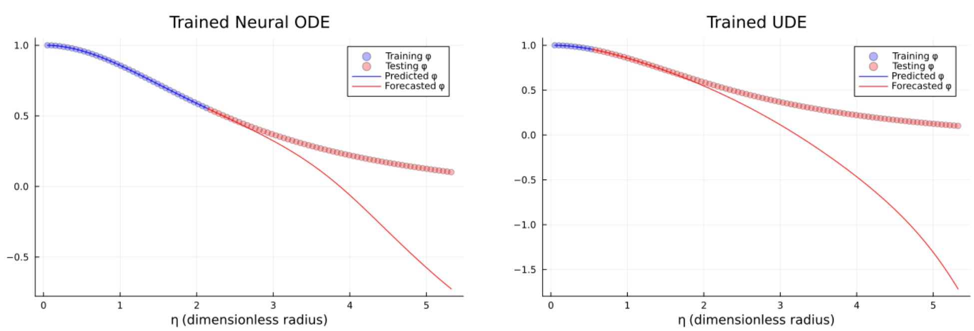


(a) (b) (c)

Figure 4: Comparison of the UDE approximation and forecasting for the Chandrasekhar's white dwarf model. The UDE was trained with varying levels of noise added to the synthetic data. These training data subsets included the values for φ and φ' with different noise levels for the first 80 equally spaced η points of the solution domain. The forecasted φ corresponding to the remaining 20% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

3.3 Breaking points

The forecasting performance of the models (Neural ODEs and UDEs) were explored further when trained with less data identifying their breaking points for the no-noise datasets. The Neural ODE failed forecasting the unseen data when trained with 40% of the entire no-noise data, while the UDE foracasted well with as little as 20% of the no-noise data available. However, the UDE collapsed when trained with 10% of the data. It is important to point out that both models failed in forecasting when trained with noisy (moderate and high) data percentages before the 40% threshold. The breaking points for no-noise datasets can be observed in figure 5, while the models' performance and breaking point plots for the three datasets fractions are available in Appendix B.



(a) (b)

Figure 5: Comparison of the models' breaking point, corresponding to the training data percentage at which the model fails predicting unseen data. a) Neural ODE breaking point occurred when training with (40%) of the available no-noise data. b) UDE breaking point occurred when training with (10%) of the available no-noise data.

3.4 Tables summarizing results

Table 3: Summary of performance for the neural network-based models for the CWDE employed in this work. These results correspond to the null-noise fraction of the datasets (synthetic data).

Method	Neural ODE	UDE
Training loss for the full dataset	4.11e-4	7.43e-8
Forecasting breakdown data subset	40% of the data	10% of the data
Forecasting breakdown η point	2.13	0.53
Training loss at breakdown point	7.09e-5	2.77e-11
Minimum training loss obtained	1.35e-5	5.29e-13

Table 4: Summary of performance for the neural network-based models for the CWDE employed in this work. These results correspond to the moderate-noise (7% standard deviation) fraction of the datasets.

Method	Neural ODE	UDE
Training loss for the full dataset	0.19	0.19
Forecasting breakdown data subset	40% of the data	40% of the data
Forecasting breakdown η point	2.13	2.13
Training loss at breakdown point	0.12	0.15
Minimum training loss obtained	0.03	0.02

Table 5: Summary of performance for the neural network-based models for the CWDE employed in this work. These results correspond to the high-noise (35% standard deviation) fraction of the datasets.

Method	Neural ODE	UDE
Training loss for the full dataset	4.65	3.88
Forecasting breakdown data subset	40% of the data	40% of the data
Forecasting breakdown η point	2.13	2.13
Training loss at breakdown point	4.39	3.96
Minimum training loss obtained	1.04	0.20

4 Conclusion

We successfully approximated the underlying data for Chandrasekhar's white dwarf equation (CWDE) with a fixed parameter C using a trained Neural Ordinary Differential Equation (Neural ODE) model with both noiseless and noisy data. A comprehensive study was conducted to identify favorable hyperparameters and neural network architectures. Ultimately, the combination of ADAM and BFGS optimizers, the tanh or RBF kernel activation function, and a streamlined neural network architecture synergistically contributed to a significant improvement of the models' performance (loss reduction). For all datasets (no-noise, moderate-noise, and high-noise), the Neural ODE model effectively approximated the training data. In terms of forecasting, the model performed well in predicting unseen data when trained with at least 80% of the available data. However, the Neural ODE model breaks down and fails to predict the unseen data when trained with less than or equal to 40% of the available datasets, indicating that while Neural ODEs offer easier modeling without relying on physical knowledge, they require a substantial amount of data to maintain forecasting reliability.

UDEs demonstrated superior performance in data-scarce situations, successfully forecasting for all testing values of the dimensionless radius η even when trained with just 20% of the noiseless available data. This capability is particularly valuable in astrophysics and astronomy where large datasets can be challenging or expensive to obtain. It also offers an advantage due to the efficiency in computational resources. Furthermore, astrophysical data often contain noise from different sources such as instrumental errors, atmospheric disturbances, cosmic rays, and background light. While most of this noise is impossible to avoid, UDE models can be employed in future investigations to model such noise, encoding it using their neural network component. This capability could lead to the identification of noise sources and uncover unknown interactions within the astrophysical system.

The UDE model's ability to recover missing interactions or contributions from the training data in this project highlights its potential in the data-driven discovery of missing physics. This capability is crucial in astrophysics and astronomy, where the exact form of governing equations might not be fully known due to incomplete theories or observational limitations. In this regard, UDEs also provide a valuable tool for theoretical advancements, refining physical laws, and testing hypotheses, therefore offering a new way to investigate the physics of the cosmos. Beyond their modeling capabilities, The UDE demonstrated strong forecasting power for the CWDE. This suggests that UDEs can enhance the accuracy of predictive models in astrophysics research by leveraging both physical laws (in the form of differential equations) and data-driven corrections (via neural networks). This hybrid approach could improve forecasts of astrophysical events and behaviors, such as stellar evolution, black hole dynamics, or the behavior of neutron stars and white dwarfs. Although the UDE model failed when noise was added in the data-scarce scenario (40% of the available datasets), its performance with noise-free data suggests that it can be highly effective in controlled experimental settings or simulations, even with low data availability.

Applications beyond white dwarfs can be explored, including modeling cosmic ray propagation, understanding galactic dynamics, and simulating accretion processes around black holes. Their versatility makes them a powerful tool for exploring various unsolved problems in astrophysics, offering computational efficiency and effectiveness when data is limited. Overall, the efficiency of UDEs in learning from data and refining model parameters suggests their broad applicability across various scientific and engineering domains, especially those where acquiring data or processing is computationally expensive.

In conclusion, while both Neural ODEs and UDEs effectively capture and predict complex dynamics over shorter intervals, their accuracy and reliability can decline over extended time spans. This underscores the need for continuous model validation and potential adjustments or enhancements to improve long-term forecasting capabilities. The Neural ODE approach offers a black-box-like solution for forecasting and phenomenological modeling when no ground truth is known, but this advantage is contrasted by its need for larger data availability. The UDE model overcomes the problem of extensive data requirement but necessitates a well-known physical model to leverage its learning power. Looking ahead, as scientific machine learning methods are further investigated, a significant focus needs to be placed on forecasting. Most studies in the literature are aimed towards predictions. While the predictive power of SciML methods has been reliably demonstrated, as shown in this study, there are still uncertainties about reliable long-term forecasting. In future work, we will modify the employed SciML models to ensure better forecasting performance, and apply symbolic regression to the recovered terms to discover the symbolic formulations of the terms recovered.

Acknowledgments and Disclosure of Funding

This research was conducted without financial support from any external funding agency, commercial entity, or non-profit organization. The authors received no financial contributions or grants for this study, and all research activities were self-funded. Three authors lead the private research center, Vizuara AI Labs, and this work was conducted in collaboration with them under their affiliation with the center. However, Vizuara AI Labs did not provide funding or exert any influence on this work in any manner.

References

[1] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[2] Raj Dandekar, Chris Rackauckas, and George Barbastathis. A machine learning-aided global diagnostic and comparative tool to assess effect of quarantine control in covid-19 spread. *Patterns*, 1(9), 2020.

[3] Raj Dandekar, Shane G Henderson, Marijn Jansen, Sarat Moka, Yoni Nazarathy, Christopher Rackauckas, Peter G Taylor, and Aapeli Vuorinen. Safe blues: A method for estimation and control in the fight against covid-19. *medRxiv*, pages 2020–05, 2020.

[4] Raj Abhijit Dandekar. *A new way to do epidemic modeling*. PhD thesis, Massachusetts Institute of Technology, 2022.

[5] Weiqi Ji, Franz Richter, Michael J Gollner, and Sili Deng. Autonomous kinetic modeling of biomass pyrolysis using chemical reaction neural networks. *Combustion and Flame*, 240:111992, 2022.

[6] Alexander Bills, Shashank Sripad, William L Fredericks, Matthew Guttenberg, Devin Charles, Evan Frank, and Venkatasubramanian Viswanathan. Universal battery performance and degradation model for electric aircraft. *arXiv preprint arXiv:2008.01527*, 2020.

[7] Zhilu Lai, Charilaos Mylonas, Satish Nagarajaiah, and Eleni Chatzi. Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508:116196, 2021.

[8] Emily Nieves, Raj Dandekar, and Chris Rackauckas. Uncertainty quantified discovery of chemical reaction systems via bayesian scientific machine learning. *Frontiers in Systems Biology*, 4:1338518, 2024.

[9] Allen M Wang, Darren T Garnier, and Cristina Rea. Hybridizing physics and neural odes for predicting plasma inductance dynamics in tokamak fusion reactors. *arXiv preprint arXiv:2310.20079*, 2023.

[10] Ali Ramadhan. *Data-driven ocean modeling using neural diferential equations*. PhD thesis, Massachusetts Institute of Technology, 2024.

[11] Chris Hill Rackauckas, Jean-Michel Campin, and Raffaele Ferrari. Capturing missing physics in climate model.

[12] Pushan Sharma, Wai Tong Chung, Bassem Akoush, and Matthias Ihme. A review of physics-informed machine learning in fluid mechanics. *Energies*, 16(5):2343, 2023.

[13] Pushan Sharma, Wai Tong Chung, Bassem Akoush, and Matthias Ihme. A review of physics-informed machine learning in fluid mechanics. *Energies*, 16(5):2343, 2023.

[14] Doaa Aboelyazeed, Chonggang Xu, Forrest M Hoffman, Jiangtao Liu, Alex W Jones, Chris Rackauckas, Kathryn Lawson, and Chaopeng Shen. A differentiable, physics-informed ecosystem modeling and learning framework for large-scale inverse problems: Demonstration with photosynthesis simulations. *Biogeosciences*, 20(13):2671–2692, 2023.

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

[16] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

[17] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *Advances in Neural Information Processing Systems*, 33:3952–3963, 2020.

[18] Hanshu Yan, Jiawei Du, Vincent YF Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. *arXiv preprint arXiv:1910.05513*, 2019.

[19] 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.

[20] Jordi Bolibar, Facundo Sapienza, Fabien Maussion, Redouane Lguensat, Bert Wouters, and Fernando Pérez. Universal differential equations for glacier ice flow modelling. *Geoscientific Model Development Discussions*, 2023:1–26, 2023.

[21] Takeshi Teshima, Koichi Tojo, Masahiro Ikeda, Isao Ishikawa, and Kenta Oono. Universal approximation property of neural ordinary differential equations. *arXiv preprint arXiv:2012.02414*, 2020.

[22] Olivier Bournez and Amaury Pouly. A universal ordinary differential equation. *Logical Methods in Computer Science*, 16, 2020.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Raghav Gupta, PK Srijith, and Shantanu Desai. Galaxy morphology classification using neural ordinary differential equations. *Astronomy and Computing*, 38:100543, 2022.

[28] Lorenzo Branca and Andrea Pallottini. Neural networks: solving the chemistry of the interstellar medium. *Monthly Notices of the Royal Astronomical Society*, 518(4):5718–5733, 2023.

[29] Sebastien Origer and Dario Izzo. Closing the gap: Optimizing guidance and control networks through neural odes. *arXiv preprint arXiv:2404.16908*, 2024.

[30] Subrahmanyan Chandrasekhar and Subrahmanyan Chandrasekhar. *An introduction to the study of stellar structure*, volume 2. Courier Corporation, 1957.

[31] S Chandrasekhar. The highly collapsed configurations of a stellar mass. *Neutron Stars, Black Holes and Binary X-Ray Sources*, 48:259, 1975.

[32] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[33] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[34] Kaifeng Gao, Gang Mei, Francesco Piccialli, Salvatore Cuomo, Jingzhi Tu, and Zenan Huo. Julia language in machine learning: Algorithms, applications, and open issues. *Computer Science Review*, 37:100254, 2020.

[35] Chris Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit. Diffeqflux. jl-a julia library for neural differential equations. *arXiv preprint arXiv:1902.02376*, 2019.

[36] Subrahmanyan Chandrasekhar and Subrahmanyan Chandrasekhar. *An introduction to the study of stellar structure*, volume 2. Courier Corporation, 1957.

[37] Harold Thayer Davis. *Introduction to nonlinear differential and integral equations*. Dover Publications, INC., New york, 1960.

[38] Kookjin Lee and Eric J Parish. Parameterized neural ordinary differential equations: Applications to computational physics problems. *Proceedings of the Royal Society A*, 477(2253):20210162, 2021.

[39] Farshud Sorourifar, You Peng, Ivan Castillo, Linh Bui, Juan Venegas, and Joel A Paulson. Physics-enhanced neural ordinary differential equations: Application to industrial chemical reaction systems. *Industrial & Engineering Chemistry Research*, 62(38):15563–15577, 2023.

[40] Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9), 2021.

[41] Mostafa Kiani Shahvandi, Matthias Schartner, and Benedikt Soja. Neural ode differential learning and its application in polar motion prediction. *Journal of Geophysical Research: Solid Earth*, 127(11):e2022JB024775, 2022.

[42] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. *arXiv preprint arXiv:2002.02798*, 2, 2020.

[43] Gavin D Portwood, Peetak P Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T Nadiga, Juan A Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, Andreas Dengel, et al. Turbulence forecasting via neural ode. *arXiv preprint arXiv:1911.05180*, 2019.

[44] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[45] Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B Shah, and Will Tebbutt. A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*, 2019.

[46] Chaopeng Shen, Alison P Appling, Pierre Gentine, Toshiyuki Bandai, Hoshin Gupta, Alexandre Tartakovsky, Marco Baity-Jesi, Fabrizio Fenicia, Daniel Kifer, Li Li, et al. Differentiable modelling to unify machine learning and physical models for geosciences. *Nature Reviews Earth & Environment*, 4(8):552–567, 2023.

[47] Chris Rackauckas, Yingbo Ma, Andreas Noack, Vaibhav Dixit, Patrick Kofod Mogensen, Simon Byrne, Shubham Maddhashiya, José Bayoán Santiago Calderón, Joakim Nyberg, Jogarao VS Gobburu, et al. Accelerated predictive healthcare analytics with pumas, a high performance pharmaceutical modeling and simulation platform. *BioRxiv*, pages 2020–11, 2020.

A Hyperparameters employed for the training datasets

A.1 Case 1: Training with 100% of the available data.

Neural ODEs

Table 6: Neural ODE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	4.11e-4	(0, 0.2)

Table 7: Neural ODE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	0.19	(0,0.2)

Table 8: Neural ODE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	4.65	(0,5.0)

UDEs

Table 9: UDE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	7.43e-8	(0,0.2)

Table 10: UDE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	0.19	(0,0.2)

Table 11: UDE Range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.006	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	3.88	(0,5.0)

A.2 Case 2: Training with 90% of the available data and forecasting.

Neural ODEs

Table 12: Neural ODE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	1.68e-4	(0,0.2)

Table 13: Neural ODE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	0.19	(0,0.2)

Table 14: Neural ODE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	4.58	(0,5.0)

UDEs

Table 15: UDE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	4.06e-8	(0,0.2)

Table 16: UDE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	0.13	(0,0.2)

Table 17: UDE Range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1100	$50 - 4000$
Loss	4.49	(0,5.0)

A.3 Case 3: Training with 80% of the available data and forecasting.

Neural ODEs

Table 18: Neural ODE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 150	$50 - 4000$
Loss	1.82e-4	(0,0.2)

Table 19: Neural ODE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 100	$50 - 4000$
Loss	0.18	(0,0.2)

Table 20: Neural ODE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 80 & BFGS: 150	$50 - 4000$
Loss	4.54	(0,5.0)

UDEs

Table 21: UDE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	6.85e-9	(0,0.2)

Table 22: UDE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1300	$50 - 4000$
Loss	0.18	(0,0.2)

Table 23: UDE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1100	$50 - 4000$
Loss	3.85	(0,4.0)

A.4 Case 4: Training with 40% of the available data and forecasting.

Neural ODEs

Table 24: Neural ODE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.02 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 150	$50 - 4000$
Loss	7.09e-5	(0,0.2)

Table 25: Neural ODE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.05 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 300	$50 - 4000$
Loss	0.12	(0,0.2)

Table 26: Neural ODE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 300	$50 - 4000$
Loss	4.39	(0, 5.0)

UDEs

Table 27: UDE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	3.49e-10	(0,0.2)

Table 28: UDE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1500	$50 - 4000$
Loss	0.15	(0.0.2)

Table 29: UDE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.1	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 200	$50 - 4000$
Loss	3.96	(0,5.0)

A.5 Case 5: Training with 20% of the available data and forecasting.

Neural ODEs

Table 30: Neural ODE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.02 & BFGS: 0.005	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 125	$50 - 4000$
Loss	6.75e-5	(0,0.2)

Table 31: Neural ODE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.05 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 100	$50 - 4000$
Loss	0.09	(0,0.2)

Table 32: Neural ODE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.001	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 100 & BFGS: 100	$50 - 4000$
Loss	1.46	(0,5.0)

UDEs

Table 33: UDE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	5.29e-13	(0,0.2)

Table 34: UDE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.001	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1500	$50 - 4000$
Loss	0.09	(0,0.2)

Table 35: UDE Range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1100	$50 - 4000$
Loss	2.15	(0,5.0)

A.6 Case 6: Training with 10% of the available data and forecasting.

Neural ODEs

Table 36: Neural ODE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.02 & BFGS: 0.005	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 125	$50 - 4000$
Loss	1.35e-5	(0,0.2)

Table 37: Neural ODE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.05 & BFGS: 0.01	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 150 & BFGS: 100	$50 - 4000$
Loss	0.03	(0,0.2)

Table 38: Neural ODE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	tanh	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.1 & BFGS: 0.001	$0.01, 0.02, 0.2, 0.05, 0.1, 0.005, 0.006$
Hidden units	160	$15, 25, 50, 100, 160, 240$
Number of Epochs	Adam: 100 & BFGS: 100	$50 - 4000$
Loss	1.04	(0,5.0)

UDEs

Table 39: UDE range of hyperparameters on training data (no-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1000	$50 - 4000$
Loss	2.77e-11	(0,0.2)

Table 40: UDE range of hyperparameters on training data (moderate-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.001	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 1500	$50 - 4000$
Loss	0.02	(0,0.2)

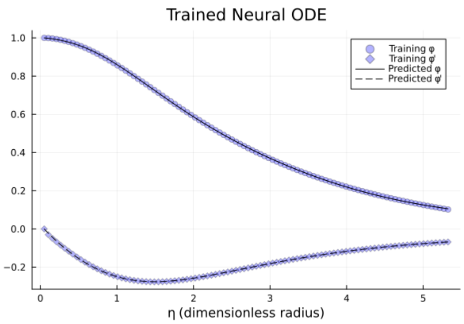
Table 41: UDE range of hyperparameters on training data (high-noise)

Hyperparameter	Values	Search Range
t_{span}	$(0.05, 5.325)$	$(0, 0.5) - (0, 10.0)$
Activation Function	RBF kernel	ReLU, tanh, sigmoid, RBF kernel
Optimization Solver	Adam & BFGS	Adam, RAdam, BFGS
Learning Rate	Adam: 0.2 & BFGS: 0.01	$0.01, 0.2, 0.001, 0.1, 0.006, 0.5$
Hidden units	15	$15, 25, 50, 100$
Number of Epochs	Adam: 300 & BFGS: 200	$50 - 4000$
Loss	0.20	(0,4.0)

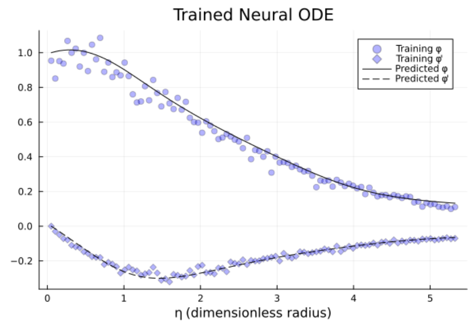
B Approximation and forecasting performance for all training datasets

B.1 Case 1: Training in the full domain (100 η points)

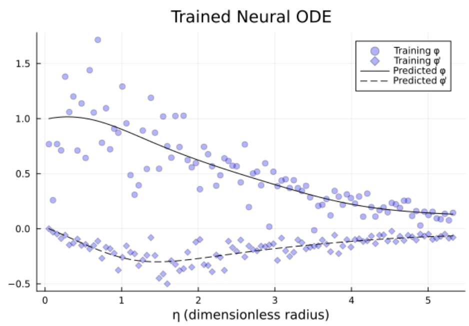
Neural ODE



(a)



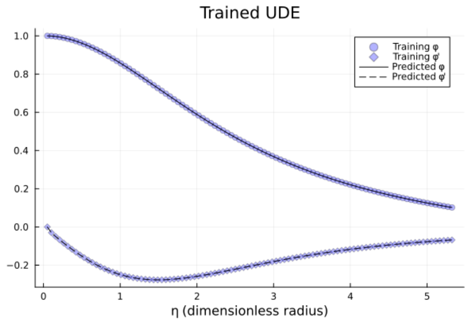
(b)



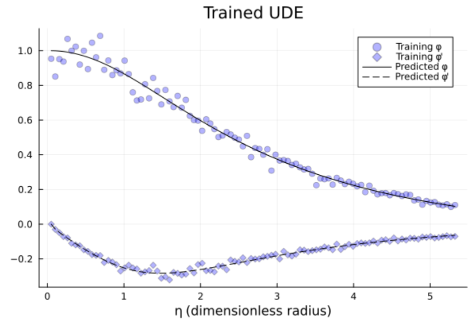
(c)

Figure 6: Comparison of the Neural ODE approximation for the Chandrasekhar's white dwarf model. The training of the Neural ODE was performed with varying noise added to the synthetic data in the full solution domain. These training datasets encompassed the values for φ and φ' at the 100 equally spaced η points with varied noise addition. Each figure shows the results for the different training sets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

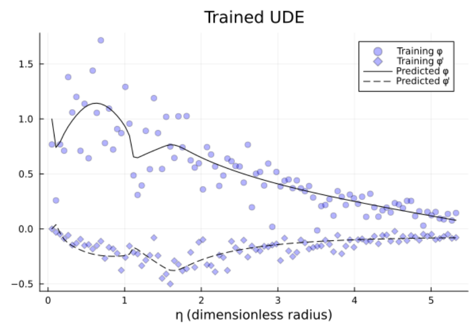
UDE



(a)



(b)



(c)

Figure 7: Comparison of the UDE approximation for the Chandrasekhar's white dwarf equation. The training of the UDE model was performed with varyng noise added to the synthetic data in the full solution domain. These training datasets encompassed the values for φ and φ' of the 100 equally spaced η points with varied noise addition: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with standard deviation of 7%. (c) High-noise dataset with standard deviation of 35%.

Missing term recovered

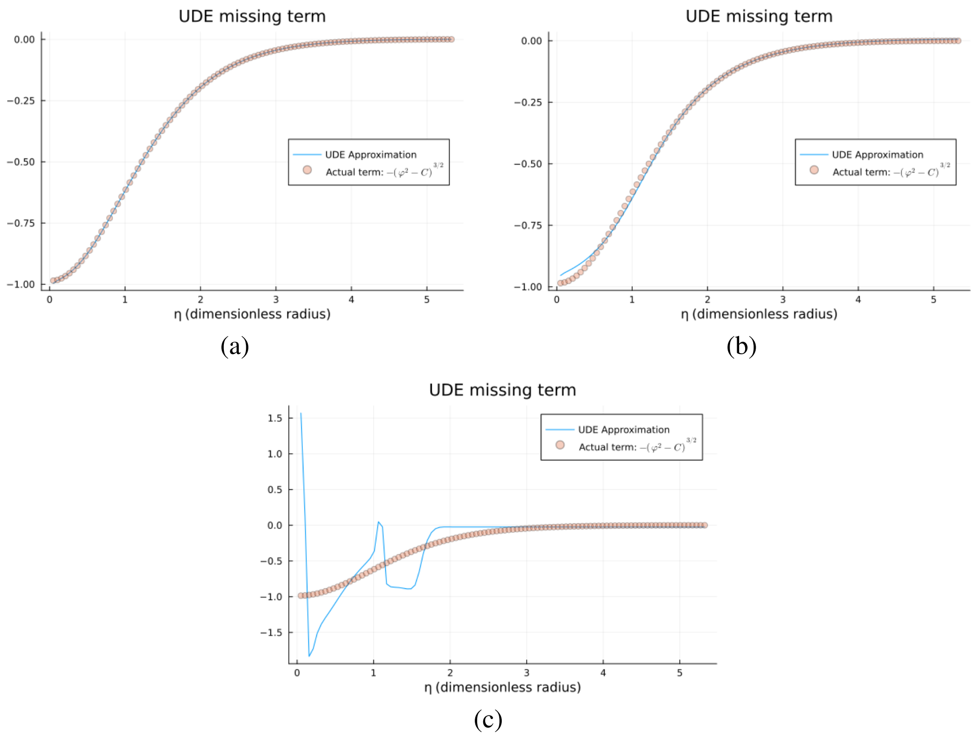
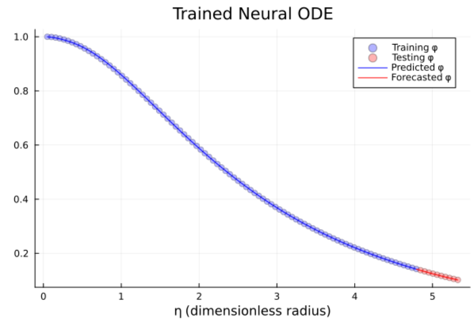


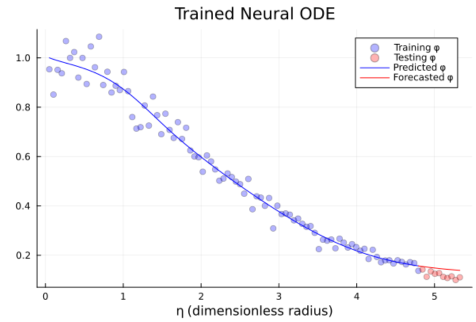
Figure 8: Comparison of the approximated missing term in the Chandrasekhar's white dwarf UDE model for the different training datasets: (a) No-noise dataset (synthetic data) set encompassing the numerically obtained values for φ and φ' within the solution domain (0, η_∞). (b) Moderate-noise dataset with standard deviation of 7% added directly to the synthetic data. (c) High-noise dataset with standard deviation of 35% added directly to the synthetic data.

B.2 Case 2: Training with 90% of the full available data and forecasting

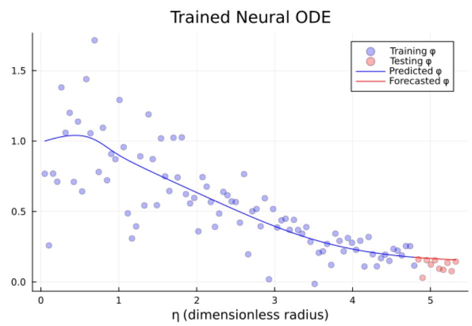
Neural ODE



(a)



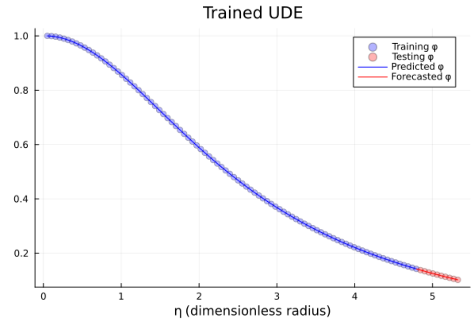
(b)



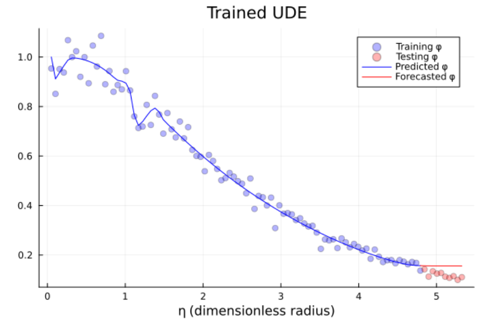
(c)

Figure 9: Comparison of the Neural ODE approximation and forecasting for the Chandrasekhar's white dwarf model. The training of the Neural ODE was performed with varying noise added to the synthetic data. These training data subsets encompassed the values for φ and φ' with varied noise levels added to the first 90 equally spaced η points of the solution domain. The forecasted φ corresponding to the remaining 10% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

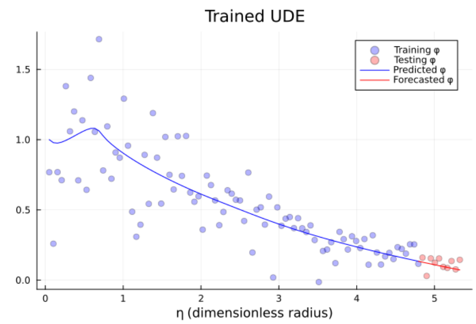
UDE



(a)



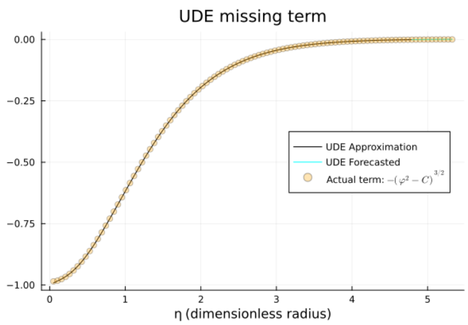
(b)



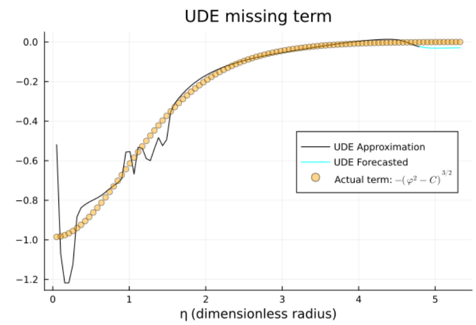
(c)

Figure 10: Comparison of the UDE approximation and forecasting for the Chandrasekhar's white dwarf model. The training of the UDE was performed with varying noise added to the synthetic data. These training data subsets encompassed the values for φ and φ' with varied noise addition for the first 90 equally spaced η points of the solution domain. The forecasted φ corresponding to the remaining 10% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

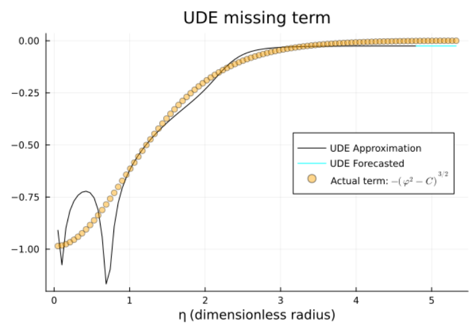
Missing term recovered



(a)



(b)



(c)

Figure 11: Comparison of the approximated missing term in the Chandrasekhar's white dwarf UDE model trained with 90% of the full available datasets: (a) No-noise data (synthetic data). (b) Moderate-noise dataset with a standard deviation of 7% added directly to the synthetic data. (c) High-noise dataset with a standard deviation of 35% added directly to the synthetic data.

B.3 Case 3: Training with 80% of the full available data and forecasting

Neural ODE

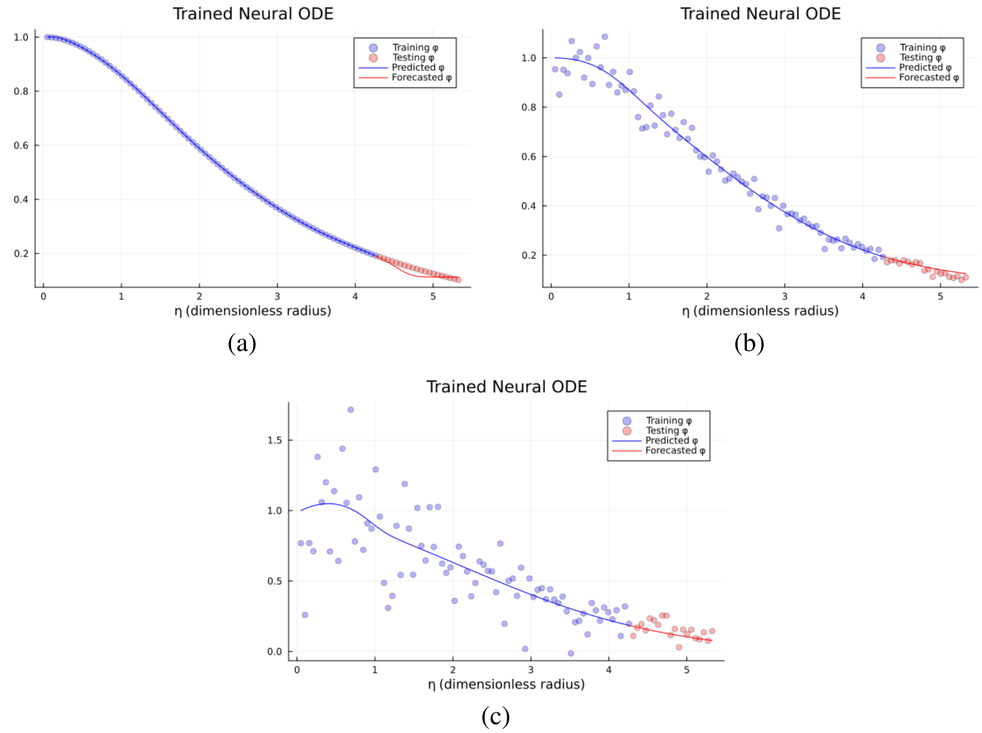
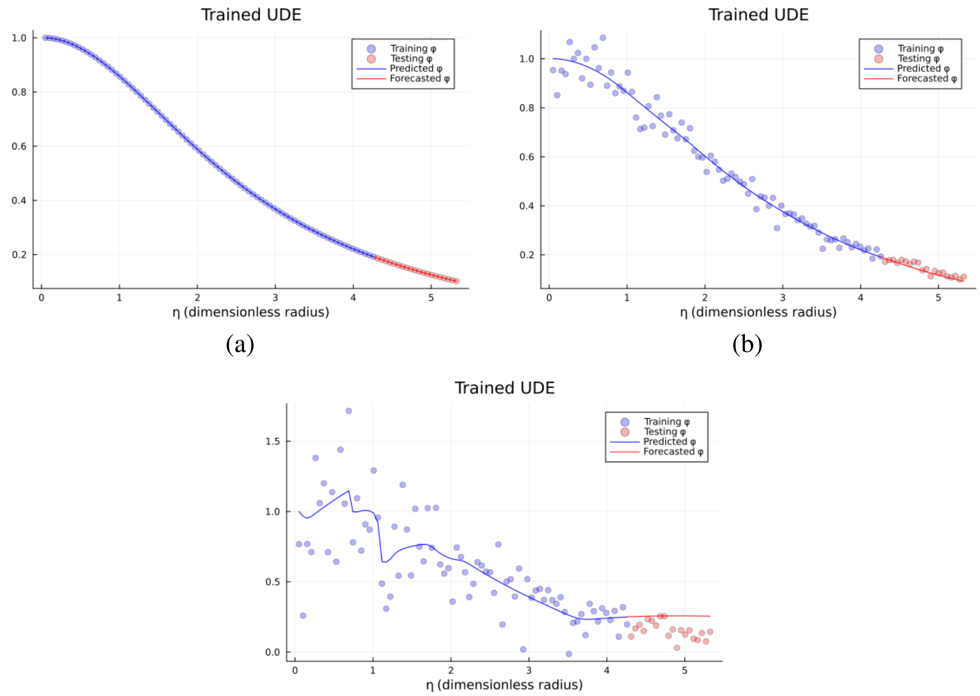


Figure 12: Comparison of the Neural ODE approximation and forecasting for the Chandrasekhar's white dwarf model. The Neural ODE was trained with varying levels of noise added to the synthetic data. These training data subsets included the values for φ and φ' with different noise levels for the first 80 equally spaced η points of the solution domain. The forecasted φ corresponding to the remaining 20% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

UDE



(a) (b)

(c)

Figure 13: Comparison of the UDE approximation and forecasting for the Chandrasekhar's white dwarf model. The UDE was trained with varying levels of noise added to the synthetic data. These training data subsets included the values for φ and φ' different noise levels for the first 80 equally spaced η points of the solution domain. The forecasted φ corresponding to the remaining 20% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

Missing recovered term

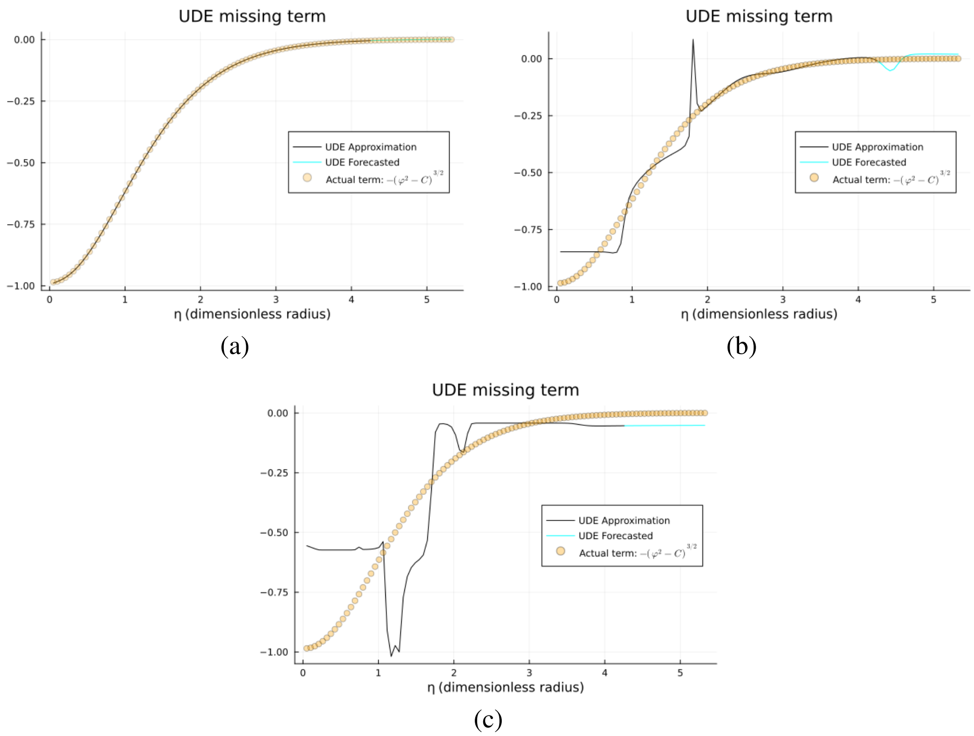
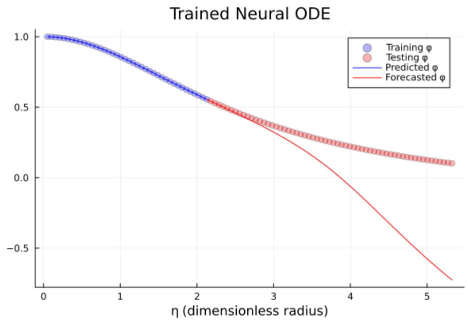


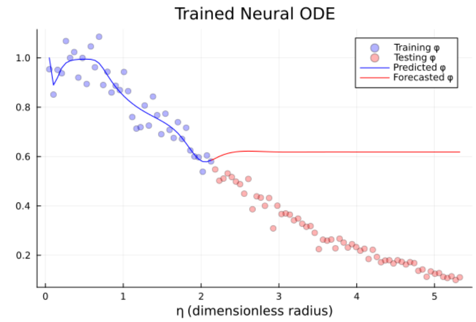
Figure 14: Comparison of the approximated missing term in the Chandrasekhar's white dwarf UDE model trained with 80% of the full available datasets: (a) No-noise data (synthetic data). (b) Moderate-noise dataset with a standard deviation of 7% added directly to the synthetic data. (c) High-noise dataset with a standard deviation of 35% added directly to the synthetic data.

B.4 Case 4: Training with 40% of the full available data and forecasting

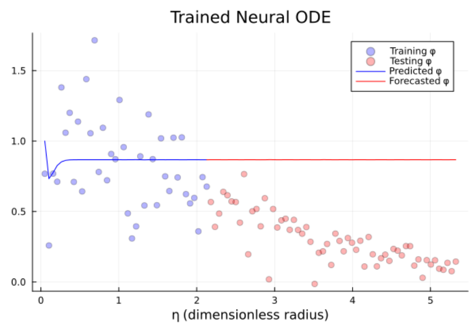
Neural ODE



(a)



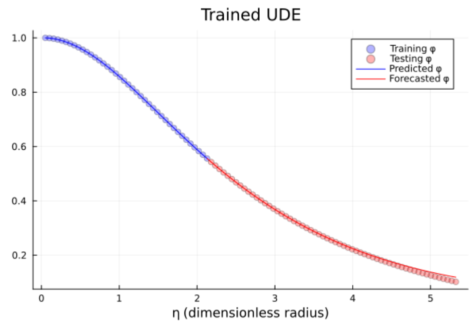
(b)



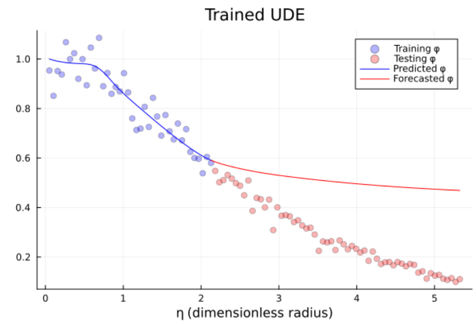
(c)

Figure 15: Comparison of the Neural ODE approximation and forecasting for the Chandrasekhar's white dwarf model. The Neural ODE was trained with varying levels of noise added to the synthetic data. These training data subsets included the values for φ and φ' with different noise levels for the first 40 equally spaced η points of the solution domain. The forecasted φ values corresponding to the remaining 60% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

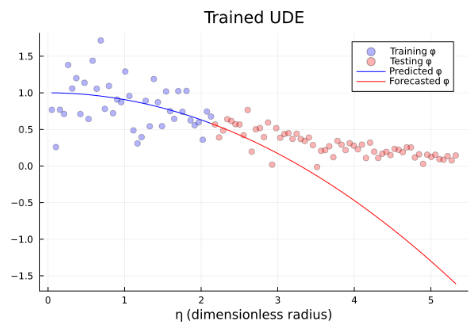
UDE



(a)



(b)



(c)

Figure 16: Comparison of the UDE approximation and forecasting for the Chandrasekhar's white dwarf model. The UDE was trained with varying levels of noise added to the synthetic data. These training datasets included the values for φ and φ' with different noise levels for the first 40 equally spaced η points of the solution domain. The forecasted φ values corresponding to the remaining 60% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

Missing term recovered

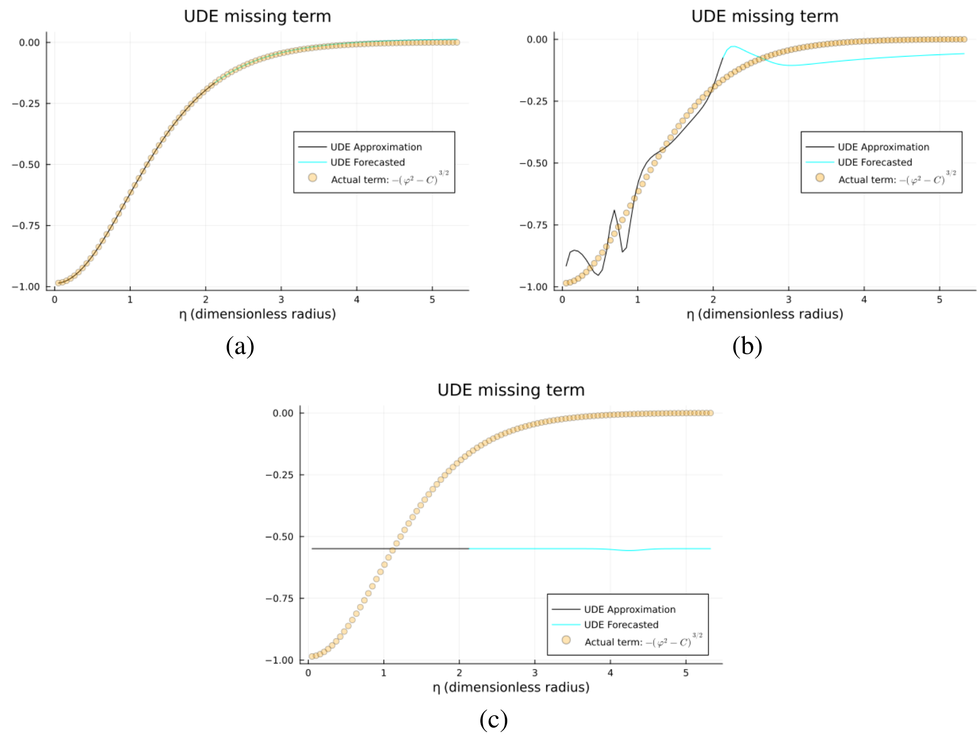
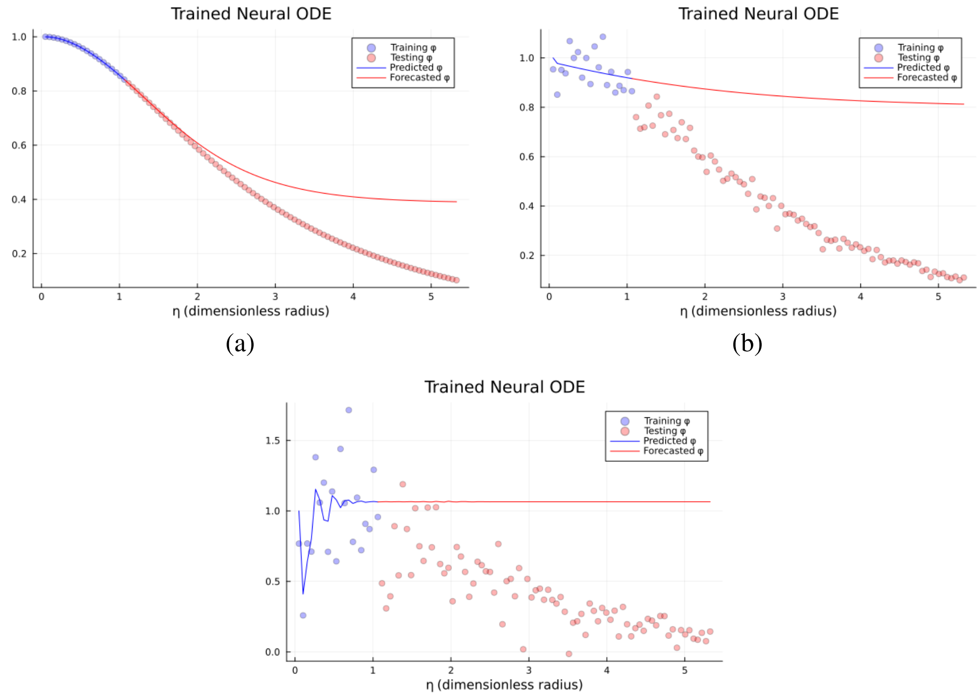


Figure 17: Comparison of the approximated missing term in the Chandrasekhar's white dwarf UDE model trained with 40% of the full available datasets: (a) No-noise data (synthetic data). (b) Moderate-noise dataset with a standard deviation of 7% added directly to the synthetic data. (c) High-noise dataset with a standard deviation of 35% added directly to the synthetic data.

B.5 Case 5: Training with 20% of the full available data and forecasting

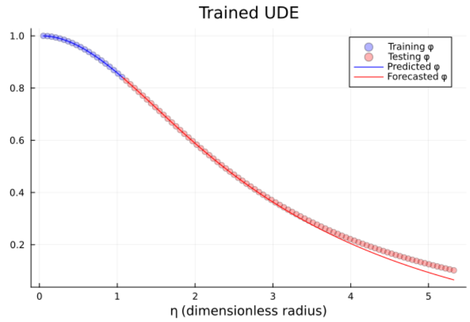
Neural ODE



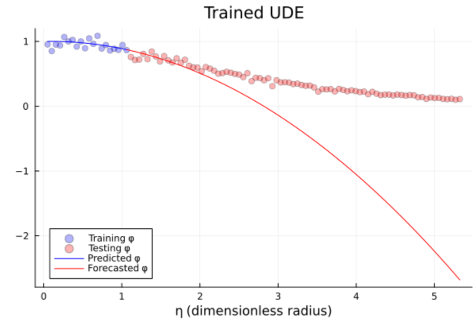
(a) (b) (c)

Figure 18: Comparison of the Neural ODE approximation and forecasting for the Chandrasekhar's white dwarf model. The Neural ODE was trained with varying levels of noise added to the synthetic data. These training datasets included theφ and φ' values with different noise levels for the first 20 equally spaced η points of the solution domain. The forecasted φ values for the remaining 80% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

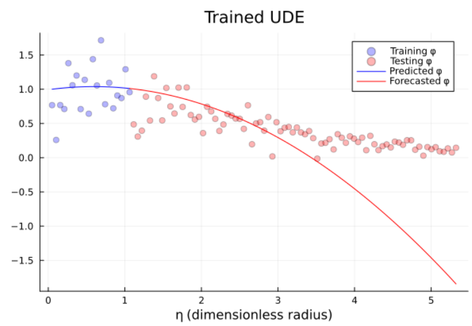
UDE



(a)



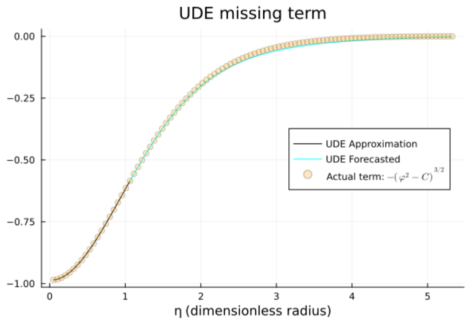
(b)



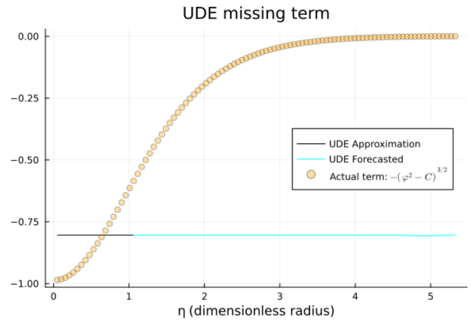
(c)

Figure 19: Comparison of the UDE approximation and forecasting for the Chandrasekhar's white dwarf model.The UDE was trained with varying levels of noise added to the synthetic data. These training datasets included the φ and φ' values with different noise levels for the first 20 equally spaced η points of the solution domain. The forecasted φ values corresponding to the remaining 80% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

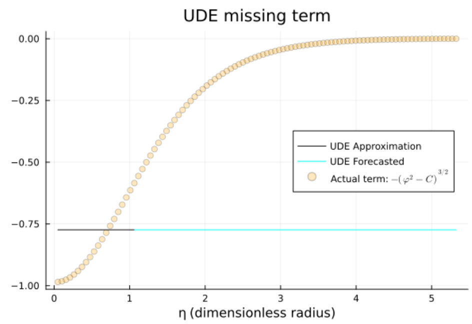
Missing term recovered



(a)



(b)

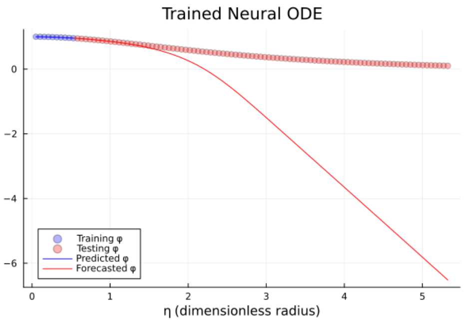


(c)

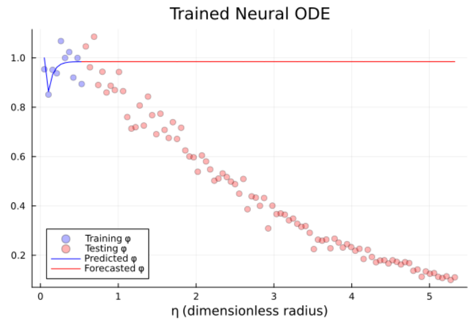
Figure 20: Comparison of the approximated missing term in the Chandrasekhar's white dwarf UDE model trained with 20% of the full available datasets: (a) No-noise data (synthetic data). (b) Moderate-noise dataset with a standard deviation of 7% added directly to the synthetic data. (c) High-noise dataset with a standard deviation of 35% added directly to the synthetic data.

B.6 Case 6: Training with 10% percent of the full available data and forecasting

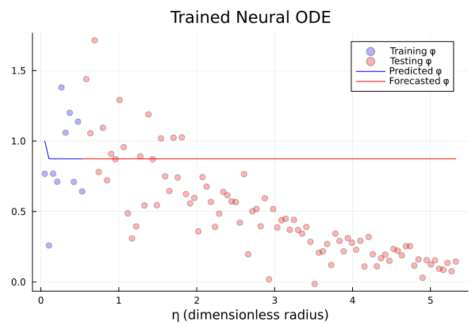
Neural ODE



(a)



(b)



(c)

Figure 21: Comparison of the Neural ODE approximation and forecasting for the Chandrasekhar's white dwarf model. The Neural ODE was trained with varying levels of noise added to the synthetic data. These training datasets included the φ and φ' values with different noise levels for the first 10 equally spaced η points of the solution domain. The forecasted φ values corresponding to the remaining 90% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

UDE

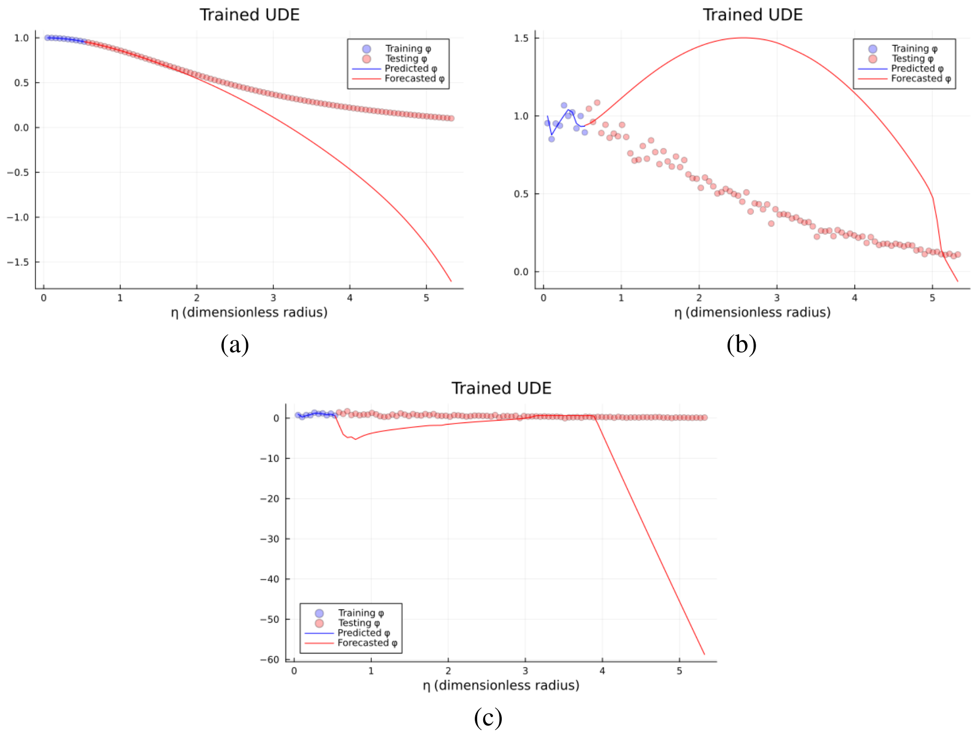


Figure 22: Comparison of the UDE approximation and forecasting for the Chandrasekhar's white dwarf model. The UDE was trained with varying levels of noise added to the synthetic data. These training datasets included the φ and φ' values with different noise levels for the first 10 equally spaced η points of the solution domain. The forecasted φ values corresponding to the remaining 90% of the η points are shown against the testing data. Each figure shows the results for the different datasets: (a) No-noise data (synthetic data) obtained numerically from the white dwarf ordinary differential equation (1). (b) Moderate-noise dataset with a standard deviation of 7%. (c) High-noise dataset with a standard deviation of 35%.

Missing term

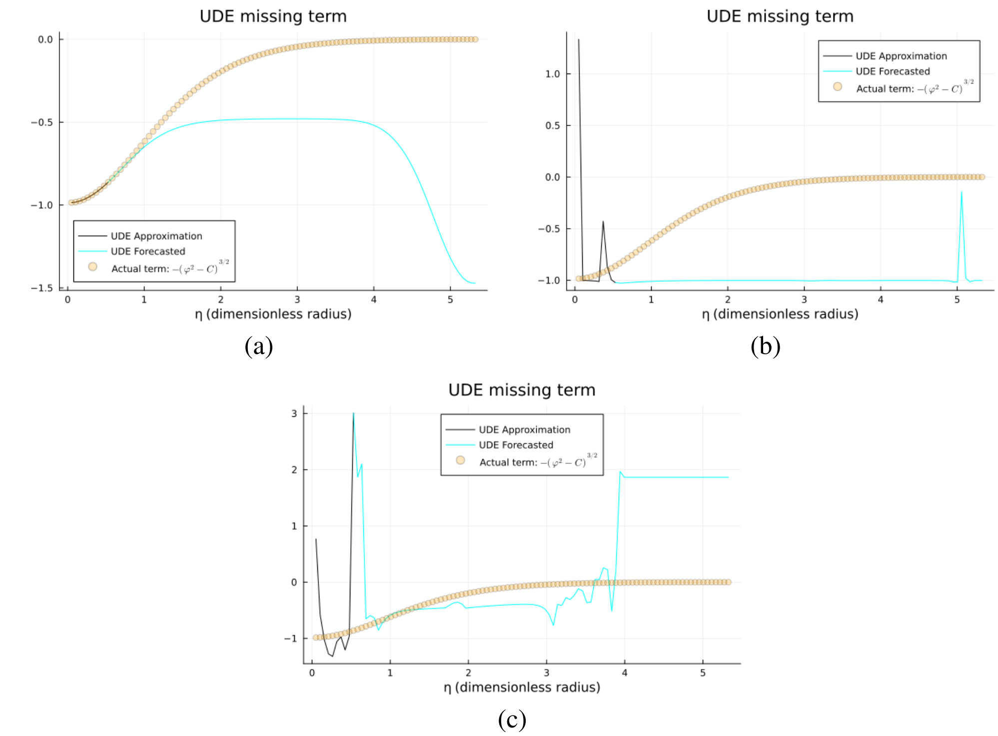


Figure 23: Comparison of the approximated missing term in the Chandrasekhar's white dwarf UDE model trained with 10% of the full available datasets: (a) No-noise data (synthetic data). (b) Moderate-noise dataset with a standard deviation of 7% added directly to the synthetic data. (c) High-noise dataset with a standard deviation of 35% added directly to the synthetic data.