ERDC/ITL TR-24-6

US Army Corps
of Engineers®
Engineer Research and
Development Center

Engineered Resilient Systems

Neural Ordinary Differential Equations for Rotorcraft Aerodynamics

Peter Rivera-Casillas and Ian Dettwiller

April 2024

Information Technology Laboratory

The US Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at https://erdclibrary.on.worldcat.org/discovery.

ERDC/ITL TR-24-6
April 2024

Neural Ordinary Differential Equations for Rotorcraft Aerodynamics

Peter Rivera-Casillas and Ian Dettwiller

US Army Engineer Research and Development Center (ERDC)
Information Technology Laboratory (ITL)
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final Technical Report (TR)

Prepared for Headquarters, US Army Corps of Engineers
Washington, DC 20314-1000

Under Program Element 0603465A, Project Number AL3, Task SAL301

Abstract

High-fidelity computational simulations of aerodynamics and structural dynamics on rotorcraft are essential for helicopter design, testing, and evaluation. These simulations usually entail a high computational cost even with modern high-performance computing resources. Reduced order models can significantly reduce the computational cost of simulating rotor revolutions. However, reduced order models are less accurate than traditional numerical modeling approaches, making them unsuitable for research and design purposes. This study explores the use of a new modified Neural Ordinary Differential Equation (NODE) approach as a machine learning alternative to reduced order models in rotorcraft applications—specifically to predict the pitching moment on a rotor blade section from an initial condition, mach number, chord velocity and normal velocity. The results indicate that NODEs cannot outperform traditional reduced order models, but in some cases they can outperform simple multilayer perceptron networks. Additionally, the mathematical structure provided by NODEs seems to favor time-dependent predictions. We demonstrate how this mathematical structure can be easily modified to tackle more complex problems. The work presented in this report is intended to establish an initial evaluation of the usability of the modified NODE approach for time-dependent modeling of complex dynamics over seen and unseen domains.

Contents

Figures and Tables

Figures

Tables

Preface

This study was conducted for the US Army Corps of Engineers under Program Element 0603465A, Project Number AL3, Task SAL301, "High Performance Computing for Rotorcraft Applications Advanced Technologies."

The work was performed by the Computational Analysis Branch of the Computational Science and Engineering Division (CSED), US Army Engineer Research and Development Center, Information Technology Laboratory (ERDC-ITL). At the time of publication, Mr. David Stuart was branch chief; Dr. Jeffrey L. Hensley was division chief; and Dr. Robert Wallace was the technical director. The deputy director of ERDC-ITL was Dr. Jackie S. Pettway, and the director was Dr. David A. Horner.

COL Christian Patterson was commander of ERDC, and Dr. David W. Pittman was the director.

This page intentionally left blank.

1 Introduction

1.1 Background

The design, testing, and evaluation of rotorcraft vehicles requires reliable and efficient tools to facilitate the analysis of fluid and structural dynamics. The High-Performance Computing Modernization Program's (HPCMP) Computational Research and Engineering Acquisition Tools, Air Vehicle Design (CREATE-AV) software, Helios, is a high-fidelity numerical modeling software that simulates these dynamics (Sankaran et al. 2010). High-fidelity numerical modeling is often prohibitively expensive for many applications. Aerodynamic studies of rotor blades can be accelerated by using an actuator line model (ALM) as an alternative to expensive overset, body-fitted configurations. The ALM method provides source terms for forces in the computational domain where the unsteady Reynolds-averaged Navier–Stokes (RANS) equations are solved. This approach is less computationally expensive than high-fidelity simulations but at the expense of reduced accuracy. Other efforts (Martinez-Gonzalez et al. 2022) explore traditional machine learning (ML) alternatives to generate surrogate models capable of reducing the computational cost of high-fidelity simulations while maintaining accuracy. Unfortunately, traditional ML models do not extrapolate well to complex cases and unseen domains.

The use of ML methods for physics-based modeling has become increasingly popular in recent years (Karniadakis et al. 2021; Rackauckas et al. 2020; Willard et al. 2020; Bhushan, Burgreen, Bowman, et al. 2020; Carleo et al. 2019; Wang et al. 2017). However, traditional ML approaches are usually not suitable for physics-based modeling due to their "black box" nature. They are referred to as "black box" methods because their complexity makes it difficult or impossible for humans to interpret the underlying components. In addition, traditional ML approaches are not suitable for physics-based applications because they do not satisfy physical conditions. As a result of this limitation, much effort has been dedicated to exploring physics-informed machine learning (PIML) methods that combine neural networks (NNs) with physics-based constraints (Dutta et al. 2022; Bhushan et al. 2021; Mao et al. 2020; Rivera-Casillas et al. 2020; Bhushan, Burgreen, Martinez, et al. 2020; Raissi et al. 2019; Raissi et al. 2018).

Most physical phenomena can be naturally represented by differential equations that model the changes in systems over time. The recently introduced Neural Ordinary Differential Equations (NODE) approach combines ordinary differential equation solvers with NNs to model physical phenomena (Chen et al. 2018). Previous studies (Dutta, Rivera-Casillas, Cecil, et al. 2021; Dutta, Rivera-Casillas, and Farthing 2021) have shown that NODE can effectively learn the underlying governing dynamics of a physical system from simulation data. In most cases, NNs are used to approximate a nonlinear function that maps input to output variables. However, NODE employs an NN to approximate the derivative of the system state and the final output is computed by a black box differential equation solver (Chen et al. 2018). NODE was originally formulated as a tool for solving ordinary differential equations (ODE). Recently it has been extended to solve partial differential equations (PDEs) (Sun et al. 2020; Dutta, Rivera-Casillas, Cecil, et al. 2021; Dutta, Rivera-Casillas, and Farthing 2021) and has been parameterized to various boundary conditions (Lee et al. 2021). These developments position NODE as a relevant tool to explore for highly accurate, efficient, physics-based simulation of rotorcraft. In this study, a new modified NODE approach is proposed to process additional input features relevant to pitching moment during the prediction of the time derivative.

1.2 Objective

Ongoing studies regarding the implementation of NN-based methods into computational numerical methods suggest that traditional, deep learning approaches provide good results for simple flight cases. Unfortunately, deep learning approaches do not extrapolate well to more complex cases and unseen domains (Boyer et al. 2021). This study seeks to quantify the numerical accuracy and computational efficiency of NODE compared to traditional deep learning approaches. Specifically, NODE is used to predict unseen, synthetic data representative of the pitching moment on a UH-60A rotor blade section during a Utility Tactical Transport Aircraft System (UTTAS) pull-up maneuver (Yamakawa et al. 1972). The approach is evaluated based on the accuracy of predicted seen and unseen data.

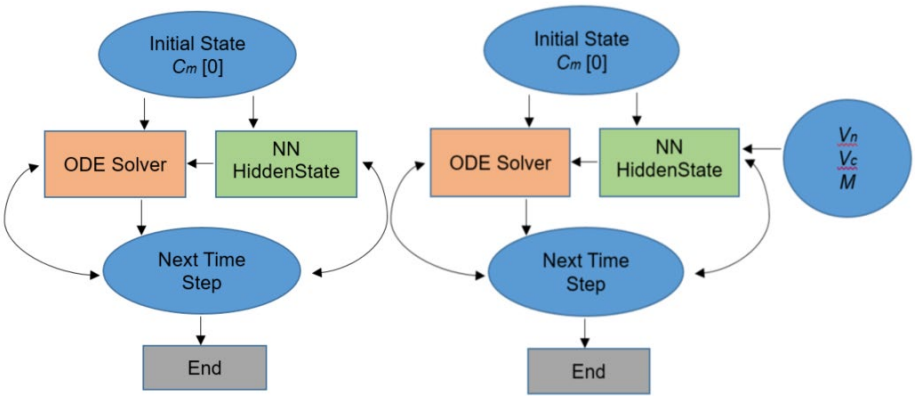
1.3 Approach

NODE is similar to residual networks, which are known to use a discrete sequence of hidden layers to represent a sequence of transformations to a hidden state. Residual networks are analogous to an Euler discretization of a continuous transformation. The difference between NODE and residual networks is that NODE uses an NN to parameterize the transformations to

the hidden state instead of a discrete sequence of hidden layers. As a result, NODE can be used for continuous time-series modeling. Figure 1*a* gives an overview of the steps followed during an iteration of the original NODE approach in the context of our work. First, the initial state is fed to the NN that outputs the derivative of the system state. The ODE solver uses the output of the NN and initial state to calculate the next time step. The new time step is fed to the NN to obtain a new derivative, and this information is sent to the ODE solver to calculate the next time step. This process is repeated until the final time step.

In this study, NODE is used to obtain time-dependent predictions of an artificial signal based on rotor blade pitching moments (C_m) when provided an initial value. The artificial signal generated with Helios represents pitching moment on a UH-60A rotor blade section during a UTTAS pull-up maneuver. The signal spans a total of 10 blade revolutions, where the first 6 revolutions are used for training and the remaining 4 are used for validation. Additional features like sectional airfoil normal velocity (V_n), chord velocity (V_c), and Mach number (M) are extracted from the simulation and used during training. NODE was designed to solve initial value problems defined by ODEs, but pitching moment is described by PDEs. To address the additional complexity of PDEs, the standard NODE approach has been modified in previous studies (Sun et al. 2020; Dutta, Rivera-Casillas, Cecil, et al. 2021; Dutta, Rivera-Casillas, and Farthing 2021). In this study, NODE is modified to process additional input features relevant to pitching moment during the prediction of the hidden state. Specifically, sectional airfoil normal velocity, chord velocity, and Mach number are introduced as input features to the NN. This modification, shown in Figure 1 (*right*), augments the hidden state prediction and allows NODE to handle the complex dynamics of the rotor blade pitching moment. For the remainder of the report, the approach depicted in Figure 1 (*left*) is referred to as the original NODE approach, and the approach depicted in Figure 1 (*right*) is referred to as the modified NODE approach.

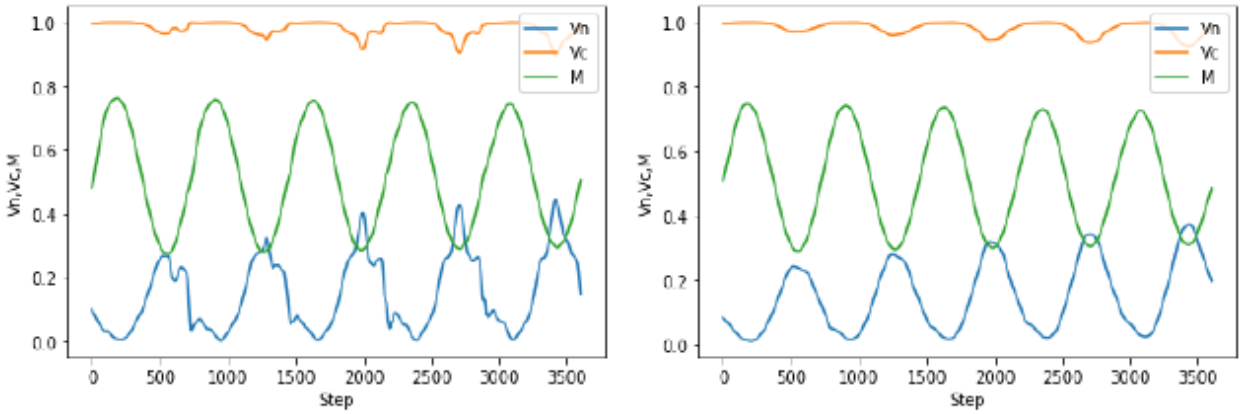
Figure 1. Graphical representation of the original (*left*) and modified (*right*) NODE approach.



A synthetic signal (S_d) is generated to help assess the applicability of the modified NODE approach for this type of signal complexity. The motivation behind the synthetic signal is not to emulate rotorcraft behavior; instead, it is to generate a realistic signal that only depends on the sectional airfoil normal velocity, chord velocity, and Mach number. If the modified NODE approach is unable to accurately predict the synthetic signal, it can be determined that the signal's complexity is too high or that the new approach does not improve the hidden state prediction. If the approach successfully predicts the synthetic signal but not the original, it can be determined that the constructed formulation (i.e., NODE NN, data content and formulation, or training methods) is not correctly designed for the underlying signal dynamics and complexity of the high-fidelity simulation.

Additionally, a Savitzky–Golay filter is used to smooth the input variables for the synthetic signal calculation. Smoothing the input variables removes noise from the synthetic signal to help simplify the problem space as part of the initial evaluation. The original sectional airfoil normal velocity, chord velocity, and Mach number inputs are shown in Figure 2*a* and the smoothed inputs in Figure 2*b*.

Figure 2. Modified and original input variables for the neural network (NN).



(a) Original input variables (b) Modified input variables

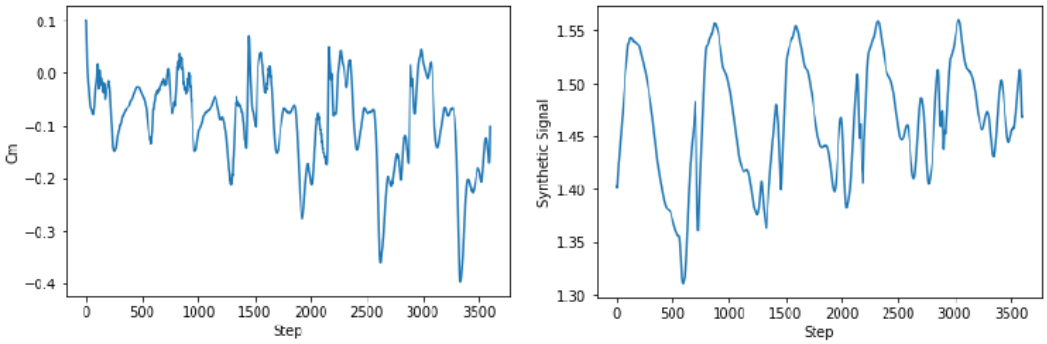
The synthetic signal is generated from the smoothed inputs using the following equation:

$$S_d = \sin(V_n) + \sin(V_c) + \sin(M)\,. \tag{1}$$

The pitching moment for one section of a rotor blade during a UH-60A UTTAS pull-up maneuver calculated using Helios is compared to the synthetic signal in Figure 3. The original signal in Figure 3*a* is noisier than the synthetic signal in Figure 3*b*, and 3*a* shows a decrease in minimum value whereas 3*b* shows an increase. Regardless, the synthetic signal retains a

periodic but changing structure with multiple peaks and troughs and a dependence on the sectional airfoil normal velocity, chord velocity, and Mach number inputs. These features allow for an initial evaluation of the modified NODE approach using simplified governing equations with relevance to the original pitching moment simulation.

Figure 3. Modified and original signals that serve as output labels during the training of the neural networks.



(a) Pitching Moment (b) Synthetic Signal

The TFDiffEq (https://github.com/titu1994/tfdiffeq) ODE solver library for the TensorFlow framework was used to implement the original and modified NODE approaches. Additionally, Keras Tuner (KT) (O'Malley et al. 2019) was used to perform an automated hyper-parameter search to fine tune the design. This was particularly challenging since implementing NODE using TensorFlow requires a custom training loop. This customization resulted in necessary modifications to some of the functions in the KT for compatibility. Furthermore, some parameters cannot be tuned with KT, necessitating a heuristic tuning approach for those parameters.

1.4 Scope

An extensive hyper-parameter search was performed to investigate the feasibility of implementing a modified NODE approach. Oftentimes a hyper-parameter search is required to obtain the correct neural network topology for certain type of datasets. The hyper-parameter search space for the KT is the following:

- Activation Function: Exponential linear unit (ELU), rectified linear unit (ReLU), hyperbolic tangent (tanh), and sigmoid
- Number of Layers: 1–6
- Number of Neurons: 16–256
- Optimizer: Adam, RMSprop, and stochastic gradient descent (SGD)
- Learning Rate: 1e-1 to 1e-5

The remaining parameters are tuned following a heuristic approach. These parameters are as follows:

- ODE solve: Euler and Runge-Kutta
- Batch: Full-batch and Mini-batch
- Search Algorithm: Bayesian and Random
- Scaling: (–1,1) and (0,1)

2 Results and Discussion

The results of the investigation are grouped into two sections. The first section includes results for the original and modified NODE approach using a Euler solver in full-batch mode, as well as mini-batch results for the modified approach. The second section includes the results for the modified approach using a Runge-Kutta (RK) solver in place of the Euler solver. High-order RK solvers generally produce better results and converge faster than Euler solvers, but the computational cost of RK solvers is higher. We perform a few tests with the RK solver to determine if there is a meaningful increase in accuracy.

2.1 Euler Solver

2.1.1 Full-Batch

2.1.1.1 Original NODE

Results for the original NODE approach using full-batch training and Euler ODE solver are shown first to establish a baseline performance. For this configuration, two scale ranges and two commonly used hyper-parameter search algorithms are investigated for each signal as shown in Table 1.

Table 1. Dataset, scaling, and search algorithm for the original NODE approach in full-batch mode with Euler solver.

ID	Dataset	Algorithm	Scaling
1	Synthetic	Bayesian	(−1,1)
2	Synthetic	Bayesian	(0,1)
3	Synthetic	Random	(−1,1)
4	Synthetic	Random	(0,1)
5	Original	Bayesian	(−1,1)
6	Original	Bayesian	(0,1)
7	Original	Random	(−1,1)
8	Original	Random	(0,1)

The KT was used to search for the best hyper-parameter configurations in each of the cases from Table 1. Each KT search was configured to run 200 trials, each one running for 1,000 epochs. The top five hyper-parameter configurations are shown in Table 2. The random search algorithm outperforms the Bayesian method, evident by the lower loss values. The Bayesian method seems to get stuck in a local minima of the hyper-parameter space, which could be caused by the structured and limited exploration of the hyper-parameter space (Barsce et al. 2017; Jasrasaria et al. 2018; Luong et

al. 2019). On the other hand, random search provides better hyper-parameter configurations, potentially because of the unstructured and wider exploration of the hyper-parameter space.

The top two hyper-parameter configurations from the cases shown in Table 2 were selected to be trained for longer iterations (50,000 epochs). The large variation in the learning rate makes it hard to identify significant patterns. Therefore, the learning rates from the hyper-parameter search are not used. Instead, an initial learning rate of 1e-3 and a learning rate schedule are used during training. The learning rate schedule divides the learning rate in half every 5,000 epochs. Table 3 shows the different configurations and the mean squared error (MSE) they produced after training. For both signals the MSE values of the different configurations are within a very small range, in fact for the synthetic signal the results are basically the same. This is strange behavior for NNs with different hyper-parameter configurations. Hence, a visual assessment of the predicted signals is carried out for more insight.

Results for both the original and synthetic signal are shown below in Figures 4 and 5. The dashed black line divides the seen training data from the unseen data to show the extrapolation capabilities of NODE. As previously established, the original NODE approach cannot handle the complex equations that describe pitching moment, and it completely fails to capture the dynamics of the problem.

2.1.1.2 Modified NODE

Results for the modified NODE approach using full-batch training with an Euler solver are presented in this section and compared with the standard NODE results shown previously. Once again, an extensive hyper-parameter search was carried out following the same strategy used in the previous section. The dataset, scaling, and search algorithm are systematically changed as shown in Table 4.

The KT was configured in the same manner as the previous section, where it was set to run 200 trials, each one running for 1,000 epochs. The top five hyper-parameter configurations are shown in Table 5. Once again, the random search algorithm reaches lower loss values than the Bayesian method. This confirms that the Bayesian algorithm gets stuck in a local minima.

Table 2. Top five configurations for each of the hyper-parameters included in the search for the original NODE approach in full-batch mode with the Euler solver.

ID	Layers	Neurons	Activation Function	Optimizer	Learning Rate	Best Loss
1	2/2/2/2/2	112/112/96/112/112	relu/relu/relu/relu/relu	adam/adam/adam/adam/adam	0.0028/0.0028/0.0028/0.0028/0.0027	0.23891
2	5/5/5/5/5	256/256/256/256/256	tanh/tanh/tanh/tanh/tanh	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.0063/0.0095/0.0087/0.0063/0.0072	0.07483
3	4/5/6/4/3	160/208/32/112/128	relu/relu/relu/relu/relu	adam/adam/adam/adam/adam	0.0031/0.0083/0.00024/0.000211/0.0016	0.23690
4	4/4/4/4/4	192/32/192/128/64	relu/relu/relu/relu/relu	adam/adam/adam/adam/adam	0.0014/0.0045/0.00024/0.00047/0.00059	0.06048
5	2/4/4/4/3	312/256/80/256/176	relu/tanh/relu/tanh/tanh	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.0032/0.01/0.0001/0.0027/0.00023	0.02820
6	5/6/3/3/3	256/160/16/16/16	relu/tanh/tanh/tanh/tanh	adam/adam/rmsprop/rmsprop/rmsprop	0.00068/0.0016/0.00083/0.00082/0.00085	0.00713
7	6/5/6/5/4	16/240/192/112/80	relu/relu/relu/relu/relu	adam/rmsprop.rmsprop/rmsprop/rmsprop	0.00097/0.007/0.0016/0.004/0.0053	0.02402
8	6/3/6/6/2	176/144/192/224/160	relu/relu/relu/relu/relu	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.0019/0.0072/0.0026/0.001/0.0092	0.00711

Table 3. Top two configurations from the hyper-parameter search for the original NODE approach in full-batch mode with the Euler solver.

ID	Epochs	Dataset	Scaling	Layers	Neurons	Activation Function	Optimizer	MSE
9	50,000	Synthetic	(−1,1)	2	112	relu	adam	0.00318
10	50,000	Synthetic	(−1,1)	4	160	relu	adam	0.00318
11	50,000	Synthetic	(−1,1)	5	208	relu	adam	0.00318
12	50,000	Synthetic	(0,1)	5	256	tanh	rmsprop	0.00318
13	50,000	Synthetic	(0,1)	4	192	relu	adam	0.00318
14	50,000	Synthetic	(0,1)	4	32	relu	adam	0.00318
15	50,000	Original	(−1,1)	2	112	relu	rmsprop	0.00681
16	50,000	Original	(−1,1)	4	256	tanh	rmsprop	0.00681
17	50,000	Original	(−1,1)	6	16	relu	adam	0.00664
18	50,000	Original	(−1,1)	5	240	relu	rmsprop	0.00683
19	50,000	Original	(0,1)	5	256	relu	adam	0.00605
20	50,000	Original	(0,1)	6	160	tanh	adam	0.00666
21	50,000	Original	(0,1)	6	176	relu	rmsprop	0.00690
22	50,000	Original	(0,1)	3	144	relu	adam	0.00680

Table 4. Dataset, scaling, and search algorithm for the modified NODE approach in full-batch mode with the Euler solver.

ID	Dataset	Algorithm	Scaling
23	Synthetic	Bayesian	(−1,1)
24	Synthetic	Bayesian	(0,1)
25	Synthetic	Random	(−1,1)
26	Synthetic	Random	(0,1)
27	Original	Bayesian	(−1,1)
28	Original	Bayesian	(0,1)
29	Original	Random	(−1,1)
30	Original	Random	(0,1)

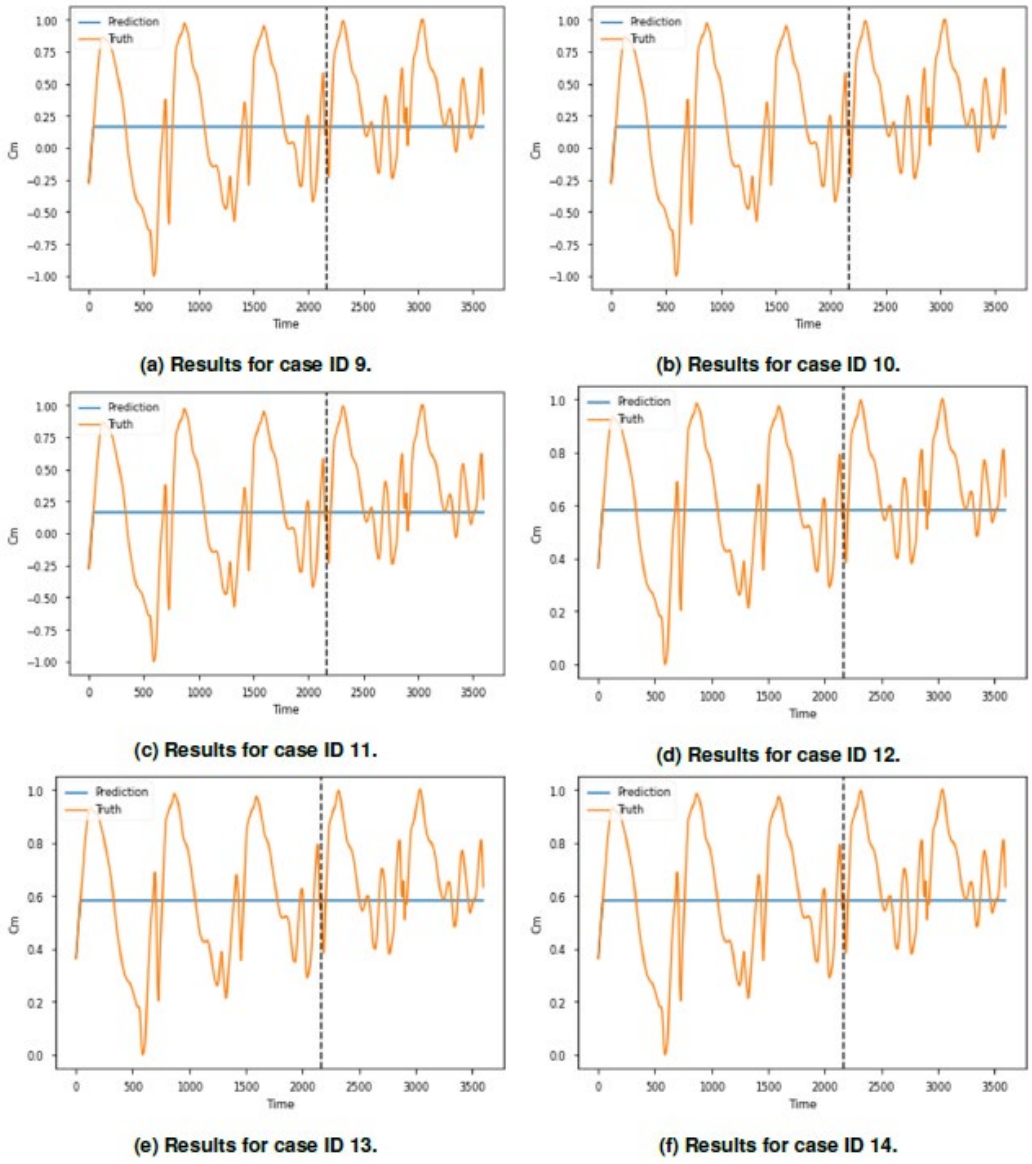
The top two hyper-parameter configurations from the cases in Table 5 were selected to be trained for longer iterations (50,000 epochs). Once again, the large variation in the learning rate makes it hard to identify significant patterns. Therefore, the same initial learning rate and learning rate schedule from the previous test are used for the remaining tests. Table 6 shows the different configurations and the MSE they produced after training. In this case we see noticeable differences between the different hyper-parameter configurations as opposed to the original NODE approach. The MSE values alone are not enough to determine if NODE accurately predicts pitching moment, therefore a visual inspection of the predicted signals is required.

Results for the synthetic signal are shown in Figure 6. The dashed black line divides the training data from the unseen data to show the extrapolation capabilities of NODE. The trained NODE models accurately reconstruct the training data in all cases, but provide mixed results during extrapolation. The best models can be easily picked out by a visual analysis. Models 33, 35, and 36 seem to perform better during extrapolation. These models accurately capture the main patterns in the signal, but fail to capture the finer details. Furthermore, there does not seem to be a defined hyper-parameter configuration that results in the best model. The only common hyper-parameter between models 33, 35, and 36 is a high number of layers (5–6). The scaling, number of neurons, activation function, and optimizer vary between the following ranges:

- Scaling: (−1,1) and (0,1)
- Number of Neurons: 16–144

- Activation function: ReLU and tanh
- Optimizer: Adam and RMSprop

Figure 4. Results from the best hyper-parameter configurations for the synthetic signal using the original NODE approach in full-batch mode with the Euler solver.



(a) Results for case ID 9.

(b) Results for case ID 10.

(c) Results for case ID 11.

(d) Results for case ID 12.

(e) Results for case ID 13.

(f) Results for case ID 14.

Figure 5. Results from the best hyper-parameter configurations for the original pitching moment signal using the original NODE approach in full-batch mode with the Euler solver.

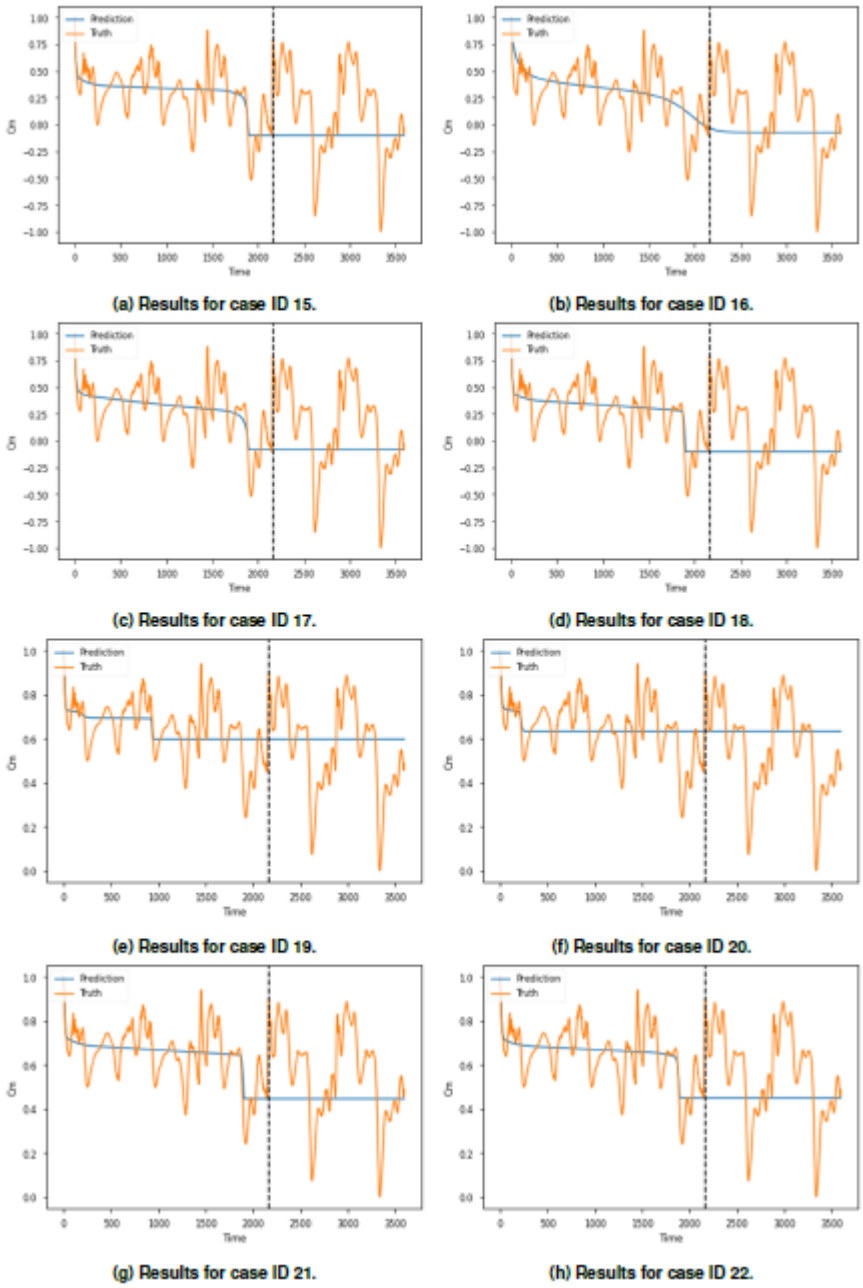


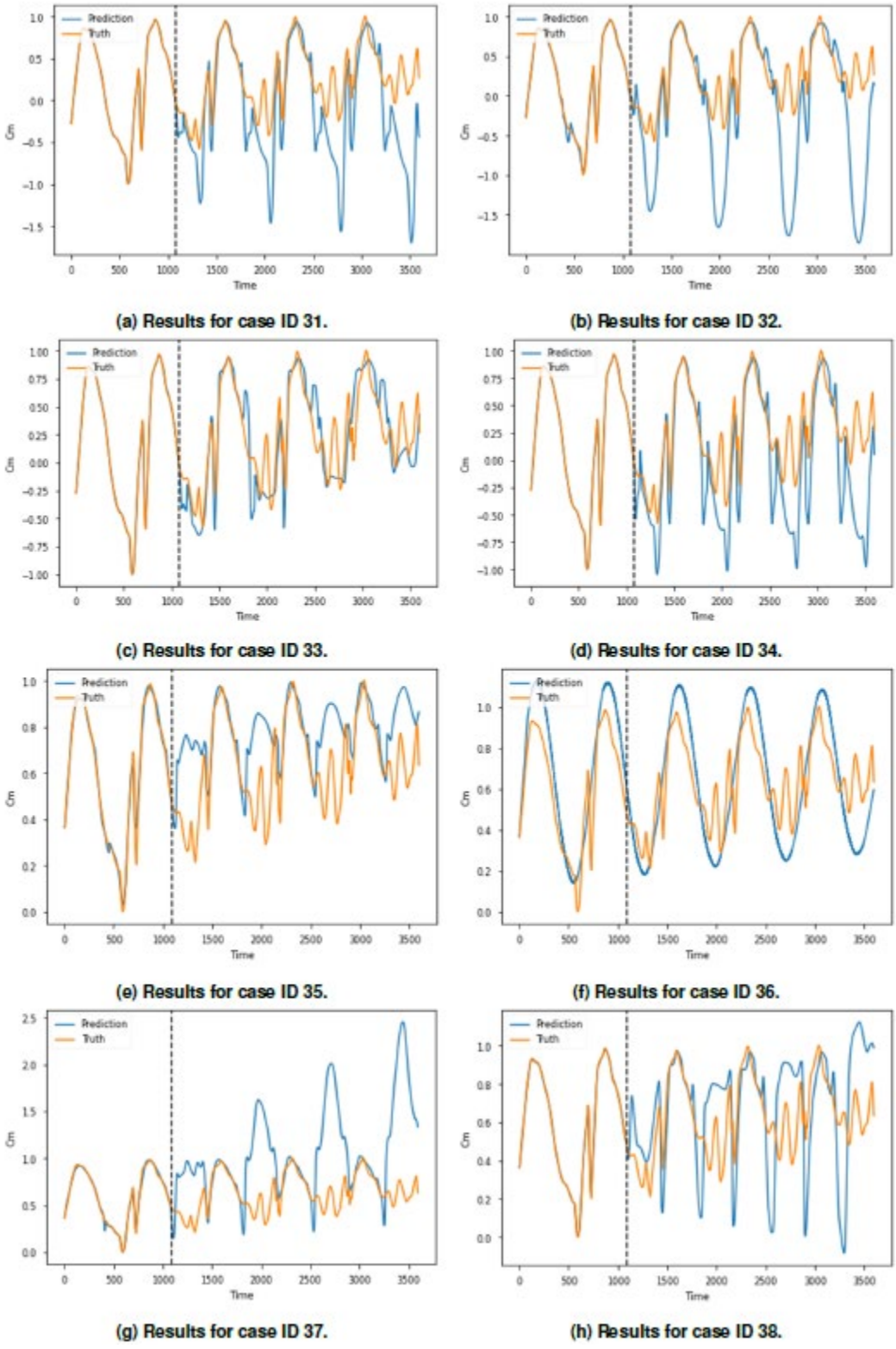
Table 5. Top five configurations for each of the hyper-parameters included in the search for the modified NODE approach in full-batch mode with the Euler solver.

ID	Layers	Neurons	Activation Function	Optimizer	Learning Rate	Best Loss
23	2/3/2/3/3	192/16/208/16/16	tanh/tanh/tanh/tanh/tanh	rmsprop/rmsprop/adam/rmsprop/rmsprop	0.01/0.01/0.002277/0.01/0.01	0.00566
24	3/3/3/3/3	16/16/16/16/16	tanh/tanh/tanh/tanh/tanh	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.01/0.01/0.01/0.01/0.01	0.00264
25	5/5/5/5/5	240/176/240/112/160	relu/relu/relu/relu/relu	adam/rmsprop/adam/rmsprop/adam	0.0034/0.0012/0.0007/0.00097/0.0045	0.00152
26	5/4/6/4/5	192/176/224/240/208	relu/relu/relu/relu/relu	adam/rmsprop/adam/adam/adam	0.00098/0.0015/0.00071/0.0047/0.0083	0.00100
27	6/3/3/3/5	256/256/256/256/256	relu/tanh/relu/tanh/tanh	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.01/0.0067/0.066/0.0066/0.006	0.00270
289	6/6/6/6/6	16/16/16/16/16	tanh/tanh/tanh/tanh/tanh	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.0038/0.0038/0.038/0.0038/0.0038	0.00170
29	4/5/6/6/5	240/192/160/224/208	relu/relu/relu/relu/relu	adam/adam/adam/adam/adam	0.0047/0.00098/0.0011/0.00071/0.0083	0.00071
30	5/6/6/6/5	240/144/80/48/192	relu/relu/relu/relu/relu	adam/adam/rmsprop/adam/rmsprop	0.0040/0.0038/0.0028/0.000083/0.003	0.00064

Table 6. Top two configurations from the hyper-parameter search for the modified NODE approach in full-batch mode with the Euler solver.

ID	Epochs	Dataset	Scaling	Layers	Neurons	Activation Function	Optimizer	MSE
31	50,000	Synthetic	(−1,1)	5	144	relu	rmsprop	0.00355
32	50,000	Synthetic	(−1,1)	5	160	relu	adam	0.00670
33	50,000	Synthetic	(−1,1)	5	144	relu	adam	0.00062
34	50,000	Synthetic	(−1,1)	5	224	relu	rmsprop	0.00204
35	50,000	Synthetic	(0,1)	6	16	relu	adam	0.00223
36	50,000	Synthetic	(0,1)	6	128	tanh	rmsprop	0.00196
37	50,000	Synthetic	(0,1)	5	176	relu	adam	0.01974
38	50,000	Synthetic	(0,1)	5	208	relu	adam	0.00340
39	50,000	Original	(−1,1)	5	256	tanh	rmsprop	0.00728
40	50,000	Original	(−1,1)	6	160	relu	adam	0.00520
41	50,000	Original	(−1,1)	5	240	relu	adam	0.00696
42	50,000	Original	(0,1)	6	192	relu	rmsprop	0.11871
43	50,000	Original	(0,1)	6	240	relu	adam	0.00622
44	50,000	Original	(0,1)	4	192	relu	adam	0.00650

Figure 6. Results from the best hyper-parameter configurations for the synthetic signal in full-batch mode using the Euler solver.



(a) Results for case ID 31.

(b) Results for case ID 32.

(c) Results for case ID 33.

(d) Results for case ID 34.

(e) Results for case ID 35.

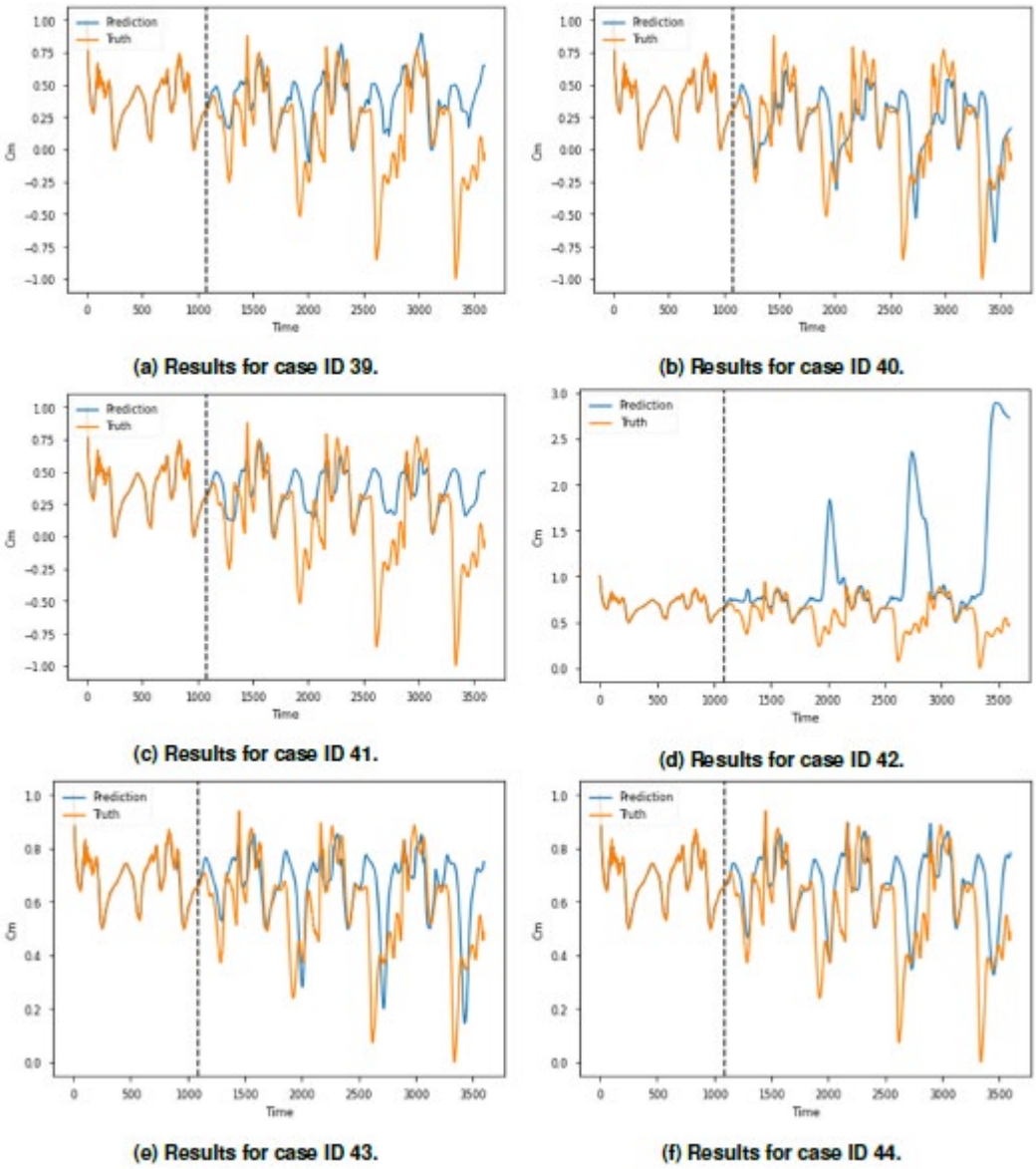
(f) Results for case ID 36.

(g) Results for case ID 37.

(h) Results for case ID 38.

Results for the original pitching moment signal are shown in Figure 7. As seen with the synthetic signal data, the trained NODE models accurately reconstruct the training data in all cases but provide mixed results during extrapolation.

Figure 7. Results from the best hyper-parameter configurations for the original pitching moment signal in full-batch mode using the Euler solver.



The best models picked out by a visual analysis of the resulting signals are 40, 43, and 44. The extrapolated results are not as accurate as those for the synthetic signal (although most of the signal patterns are captured in the first 1,000 unseen time steps). It is worth noting that these patterns are also visible in the training domain, and it is possible that the models

are simply repeating these patterns. Although this is a possibility, the model is also capturing the critical decaying behavior that is not visible in the training domain.

For this case there seems to be a distinct set of hyper-parameter configurations that result in the best model. The common hyper-parameters between models 40, 43, and 44 are a high number of layers (4–6), high number of neurons (160–240), ReLU activation function, and the Adam optimizer. The scaling varies between (–1,1) and (0,1).

2.1.2 Mini-Batch

The next tests were carried out using mini-batches to train and Euler as the ODE solver. Mini-batching for NODE is not as straightforward as for standard NN. In previous studies the authors suggest concatenating the states of each batch element and then performing the ODE solver evaluations (Chen et al. 2018). However, we are interested in understanding if we can train NODE models capable of providing results for different initial conditions. In our approach, each batch receives a different initial condition, and the resulting states are fed to the ODE solver; the weights are optimized based on the final output.

Once again, an extensive hyper-parameter search was carried out following the same strategy used in the first tests. The dataset, scaling, and search algorithm are systematically changed as shown in Table 7. A fixed batch size of 32 is used for all cases.

The KT was limited to 200 trials, with each of those trials running for 1,000 epochs. The top five hyper-parameter configurations are shown in Table 8. In this case the learning rate was fixed to 0.001. Similar to the previous tests, the Bayesian search seems to get stuck in a local minima while the Random search reaches lower loss values. Table 9 chosen configurations and the MSE they produced after training.

Results for the synthetic signal are shown in Figure 8. In this case the trained NODE models are not able to accurately reconstruct the training data, but they perform better during extrapolation in comparison to the full-batch models. Models 54, 55, 56, and 57 are selected as the best models by visual analysis. The models accurately capture the main patterns in the signal but are not able to capture the finer details.

In this case there are some dominant hyper-parameters, but there is no dominant configuration among these parameters. For example, three of the best models have (−1,1) scaling, but the rest of the hyper-parameters are mixed. A lower number of layers (2–4) seems to be more appropriate for this case.

Table 7. Dataset, scaling, and search algorithm for the modified NODE approach in mini-batch mode using the Euler solver.

ID	Dataset	Algorithm	Scaling
45	Synthetic	Bayesian	(−1,1)
46	Synthetic	Bayesian	(0,1)
47	Synthetic	Random	(−1,1)
48	Synthetic	Random	(0,1)
49	Original	Bayesian	(−1,1)
50	Original	Bayesian	(0,1)
51	Original	Random	(−1,1)
52	Original	Random	(0,1)

The number of neurons seems to be within the same range as the full-batch test (16-160). The activation function varies between ReLU and tanh. The dominant optimizer seems to be RMSprop, but one of the configurations uses Adam.

Results for the original pitching moment signal are shown in Figure 9. The models are not able to capture the pitching moment dynamics. In other words, the NN is unable learn the hidden state for different initial conditions. This is likely due to the complexity of the signal or missing information about the physical processes. These results are not acceptable for design purposes and there is no point in outlining the best models or dominant hyper-parameter configurations for this test.

Table 8. Top five configurations for each of the hyper-parameters included in the search for the modified NODE approach in mini-batch mode using the Euler solver.

ID	Layers	Neurons	Activation Function	Optimizer	Best Loss
45	5/2/3/3/4	176/144/256/64/16	tanh/relu/relu/relu/tanh	adam/adam/rmsprop/adam/adam	0.05847
46	4/3/4/3/3	16/80/16/144/144	relu/tanh/tanh/tanh/relu	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.01584
47	4/6/3/4/3	160/112/192/240/176	tanh/relu/relu/relu/relu	rmsprop/adam/rmsprop/adam/rmsprop	0.05744
48	4/6/6/2/2	208/160/144/80/160	relu/relu/relu/relu/relu	rmsprop/rmsprop/adam/rmsprop/rmsprop	0.01409
49	4/4/4/4/4	64/64/64/64/64	relu/relu/relu/relu/relu	rmsprop/rmsprop/rmsprop/rmsprop/rmsprop	0.02857
50	4/4/4/4/4	96/96/96/96/96	relu/relu/relu/relu/relu	adam/adam/adam/adam/adam	0.00722
51	5/3/5/4/3	192/208/96/112/240	relu/relu/relu/relu/tanh	rmsprop/adam/adam/adam/rmsprop	0.02881
52	3/4/4/5/6	160/240/112/224/240	relu/relu/relu/relu/relu	adam/adam/adam/adam/rmsprop	0.00734

Table 9. Chosen configurations for the modified NODE approach in mini-batch mode using the Euler solver.

ID	Epochs	Dataset	Scaling	Layers	Neurons	Activation Function	Optimizer	MSE
53	50,000	Synthetic	(−1,1)	5	176	tanh	adam	0.00117
54	50,000	Synthetic	(−1,1)	2	144	relu	adam	0.00061
55	50,000	Synthetic	(−1,1)	4	16	relu	rmsprop	0.00077
56	50,000	Synthetic	(−1,1)	3	80	tanh	rmsprop	0.00049
57	50,000	Synthetic	(0,1)	4	160	tanh	rmsprop	0.00558
58	50,000	Synthetic	(0,1)	6	112	relu	adam	0.00301
59	50,000	Synthetic	(0,1)	4	208	relu	rmsprop	0.00781
60	50,000	Synthetic	(0,1)	6	160	relu	rmsprop	0.00153
61	50,000	Original	(−1,1)	4	64	relu	rmsprop	0.00381
62	50,000	Original	(−1,1)	4	96	relu	adam	0.00478
63	50,000	Original	(0,1)	5	192	relu	rmsprop	0.00433
64	50,000	Original	(0,1)	3	208	relu	adam	0.00557
65	50,000	Original	(0,1)	3	160	relu	adam	0.00459
66	50,000	Original	(0,1)	4	240	relu	adam	0.00768

Figure 8. Results from the best hyper-parameter configurations for the synthetic signal in mini-batch mode using the Euler solver.

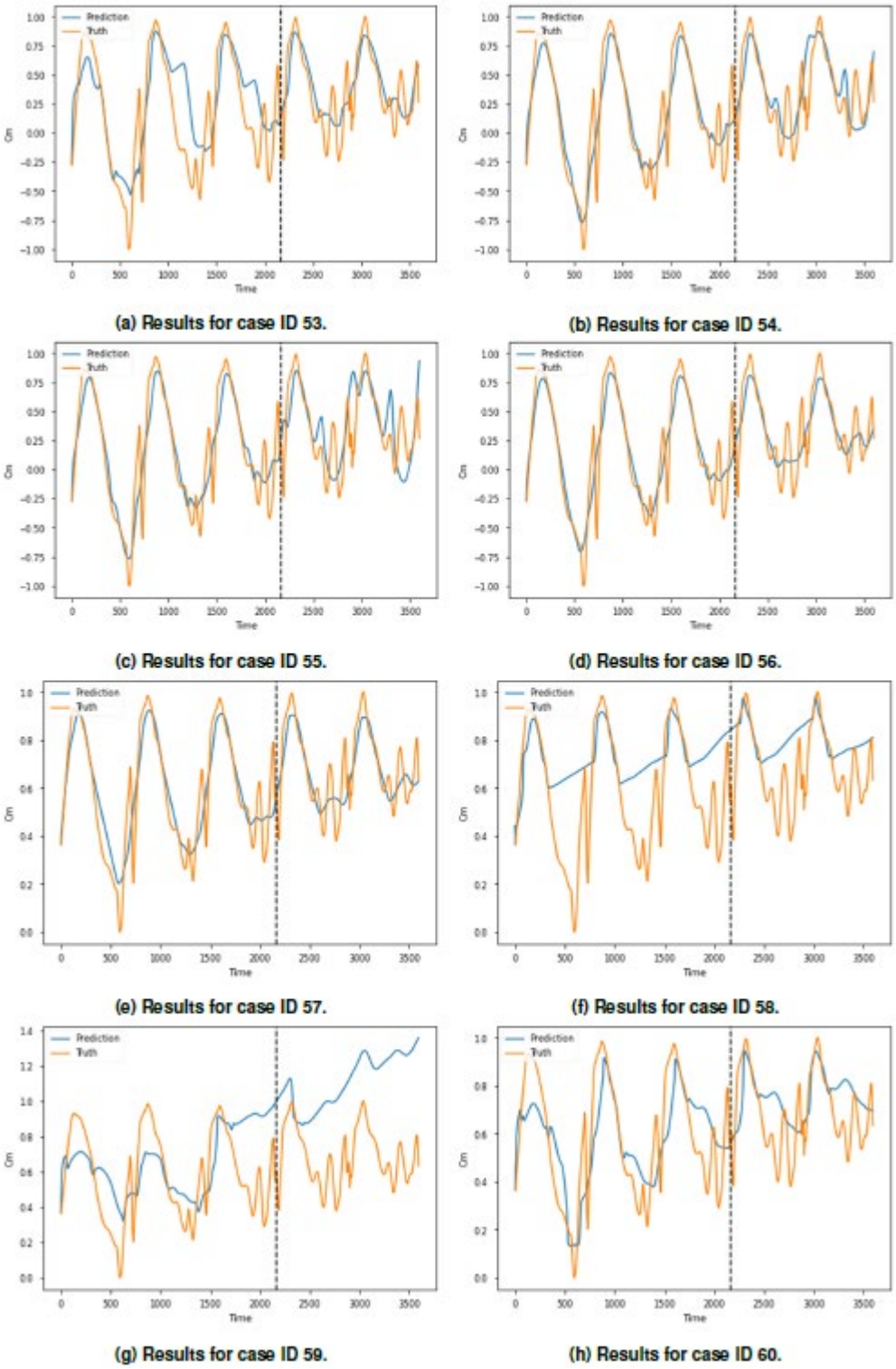
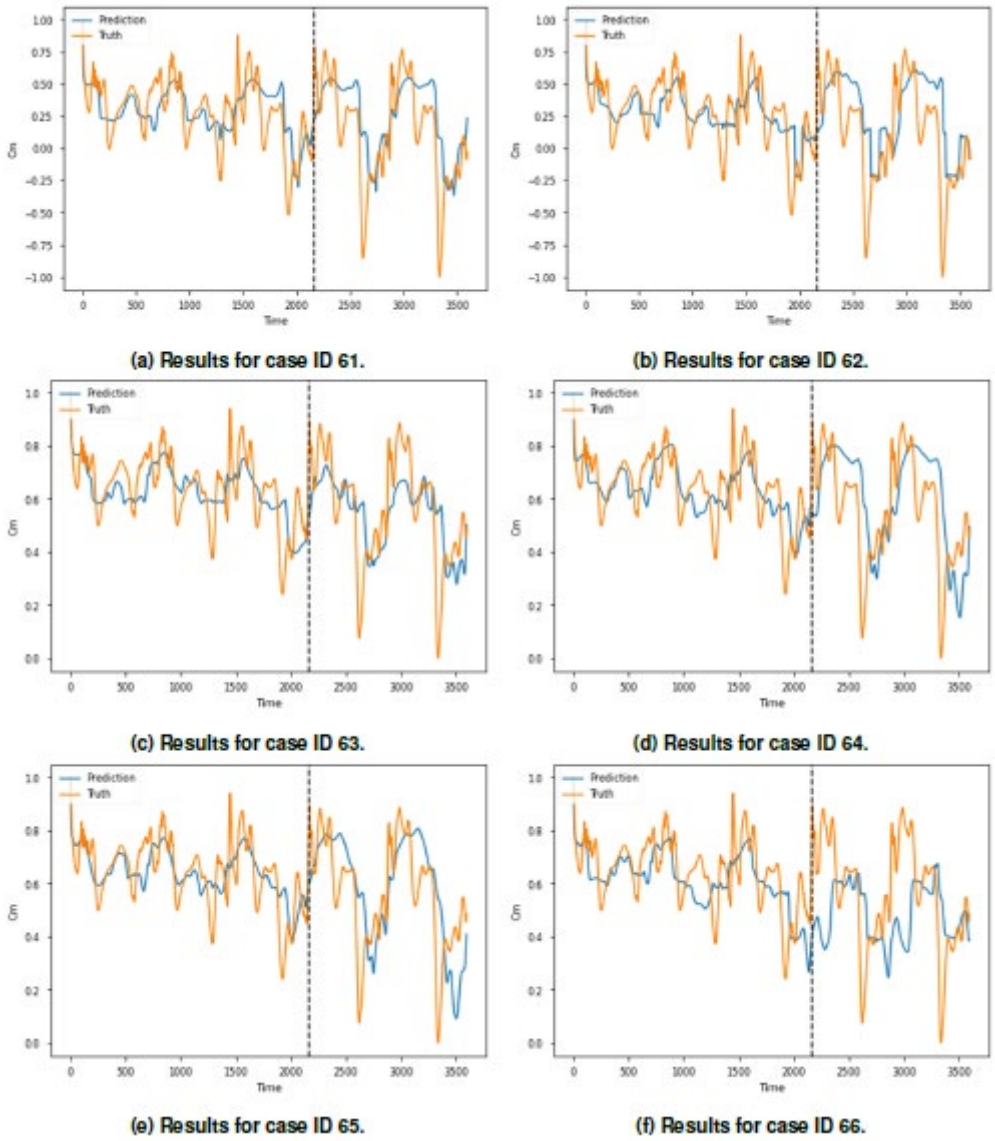


Figure 9. Results from the best hyper-parameter configurations for the original pitching moment signal in mini-batch mode using the Euler solver.



2.2 Runge-Kutta Solver

In this section results for full-batch training and a fourth-order Runge-Kutta (RK4) ODE solver are presented. Given the nature of the RK4 solver, an additional step was added to the training routine. This step performs a linear interpolation of sectional airfoil normal velocity, chord velocity, and Mach number and the interpolated values are fed to the NN during the intermediate steps taken by the solver. The RK4 solver is significantly more expensive than Euler and therefore an extensive hyper-parameter search is not performed. Instead, some of the best configurations from the previous tests are selected and a NODE model is trained for 20,000 epochs. We

train for less epochs than the previous case due to the computational expense. Table 10 shows the chosen configurations and the MSE they produced after training.

Table 10. Chosen configurations for the full-batch and fourth-order Runke-Kutta solver.

ID	Epochs	Dataset	Scaling	Layers	Neurons	Activation Function	Optimizer	MSE
67	20,000	Synthetic	(−1,1)	5	144	relu	adam	0.00278
68	20,000	Synthetic	(0,1)	5	208	relu	adam	0.00260
29	20,000	Original	(−1,1)	6	160	relu	adam	0.00827
70	20,000	Original	(0,1)	6	240	relu	adam	0.00132

Results for both signals are shown below. Figures 10*a* and 10*b* show results for the synthetic signal, while Figures 10*c* and 10*d* show results for the original pitching moment. Mixed results are obtained for both cases. An extensive search has to be conducted before establishing the best configurations for the RK4 solver. Nevertheless, the results show that the scaling has a significant effect: for both signals the (0,1) scaling outperforms the (−1,1) scaling. A higher number of neurons per layer seems to be more appropriate. Nothing can be said about the number of layers because only a high number of layers was tested. The same activation function and optimizer was used for all cases. Overall, RK4 solver seems to perform similar to the Euler solver with the exception of model 70. The results from model 70 shown in Figure 10*d* hint that higher order solvers improve the model's performance. However, RK4 takes approximately 10 times longer to train, and for this reason it is a significantly more challenging-to-use alternative.

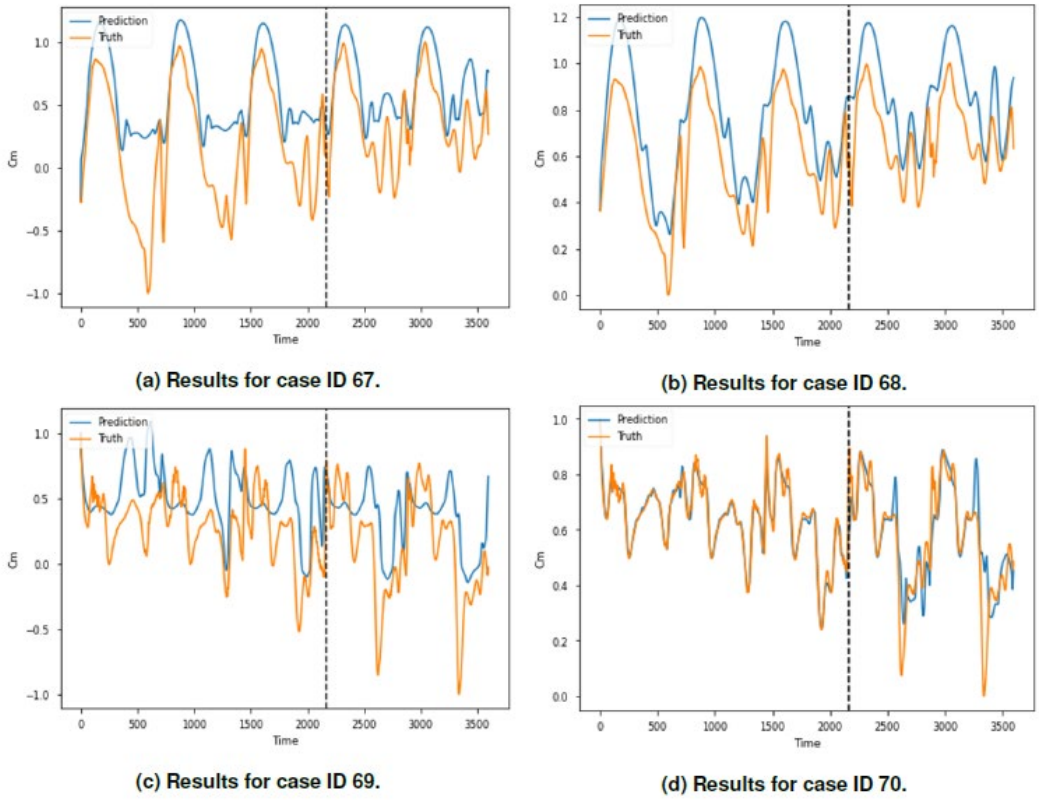
2.3 Summary of Results

In all cases when comparing the Bayesian method to the random search algorithm, the Bayesian method appeared to become stuck in local minima. The random search algorithm was able to consistently return better parameters, possibly due to the unstructured, wider exploration of the hyperparameter space. The performance of the Bayesian method could inversely be limited by the structured and limited exploration of the hyper-parameter space.

The original NODE approach completely fails to capture the dynamics of even the synthetic signal. The modified approach using full-batch training with a Euler solver was shown to accurately reconstruct the training data for both the synthetic and original pitching moment signals. Several of the

tuned models were also able to adequately predict unseen data for an appreciable number of time steps, capturing much of the critical dynamics of the signals.

Figure 10. Results from the best hyper-parameter configurations for both signals in full-batch mode using the fourth-order Runge-Kutta solver. Plots (*a*) and (*b*) show results for the synthetic signal, while (*c*) and (*d*) show results for the original pitching moment.



Swapping to a mini-batch training scheme for the modified NODE approach resulted in improved extrapolation for the synthetic signal, although it performed more poorly for training data and was completely unable to capture the dynamics of the original pitching moment signal. The introduction of the Runge-Kutta solver in place of the Euler solver showed promise, but the increased computational cost of the solver prevented the hyper-parameter search required to tune the method.

3 Conclusion

Similar NN configurations seem to follow different optimization paths and produce notably different results. This behavior suggests that there is a possibility of finding better models with more extensive hyper-parameter searches. In the mini-batch section, NODE models are trained to learn different initial conditions, and while it fails for the original pitching moment signal, it performs well for the synthetic signal. Additionally, the independent variables provided to the NN during hidden state predictions, as part of our modified approach, seem to provide crucial information that helps guide the prediction. This is observed through the significant improvements in model performance for both the synthetic and high-fidelity pitching moment signals.

Overall the NODE approach does not provide the efficiency and accuracy required to replace the ALM, but in some cases it can outperform simple multilayer perceptron networks that usually provide noisy predictions outside of the training domain. The mathematical structure of NODE seems to favor time-dependent predictions and extrapolation to unseen domains. The modified NODE approach is shown to offer significant improvement over the original method. The predictive capabilities, especially those demonstrated on the synthetic signal, indicate an improved capacity to model complex, time-dependent dynamics. However, significant additional work is required to reliably capture the complexities of the high-fidelity pitching moment. Specifically, more work is necessary to explore a wider hyper-parameter space, as well as to accelerate training routines and implementing higher order differential equation solvers that will likely lead to better results.

Bibliography

Barsce, Juan Cruz, Jorge A. Palombarini, and Ernesto C. Martınez. 2017. "Towards Autonomous Reinforcement Learning: Automatic Setting of Hyperparameters Using Bayesian optimization." In 2017 XLIII Latin American Computer Conference (CLEI), 1–9. IEEE.

Bhushan, S., Greg W Burgreen, D. Martinez, and Wes Brewer. 2020. "Machine Learning for Turbulence Modeling and Predictions." In *Fluids Engineering Division Summer Meeting*, vol. 83730, V003T05A008. American Society of Mechanical Engineers.

Bhushan, Shanti, Greg W. Burgreen, Joshua L. Bowman, Ian D. Dettwiller, and Wesley Brewer. 2020. "Predictions of Steady and Unsteady Flows Using Machine-Learned Surrogate Models." In *2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)*, 80–87. IEEE.

Bhushan, Shanti, Greg W. Burgreen, Wesley Brewer, and Ian D. Dettwiller. 2021. "Development and Validation of a Machine Learned Turbulence Model." *Energies* 14 (5): 1465.

Boyer, Mathew, and Wes Brewer. 2021. *Physics-Informed Machine Learning Support for Rotorcraft Download Prediction*. Technical report. Department of Defense High Performance Computing Modernization Program User Productivity Enhancement and Training.

Carleo, Giuseppe, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, et al. 2019. "Machine Learning and the Physical Sciences." *Reviews of Modern Physics* 91 (4): 045002.

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. "Neural Ordinary Differential Equations." *Advances in Neural Information Processing Systems* 31.

Dutta, Sourav, Peter Rivera-Casillas, Orie Cecil, Matthew Farthing, Emma Perracchione, and Mario Putti. 2021. "Data-Driven Reduced Order Modeling of Environmental Hydrodynamics Using Deep Autoencoders and Neural ODEs." In *Proceedings of the IXth International Conference on Computational Methods for Coupled Problems in Science and Engineering (COUPLED PROBLEMS 2021)*, 2021, pp. 1–16. arXiv preprint arXiv:2107.02784.

Dutta, Sourav, Peter Rivera-Casillas, and Matthew W Farthing. 2021. "Neural Ordinary Differential Equations for Data-Driven Reduced Order Modeling of Environmental Hydrodynamics." In *Proceedings of the AAAI 2021 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physical Sciences, Virtual Meeting*, 22–24 March 2021; CEUR-WS: Stanford, CA. arXiv preprint arXiv:2104.13962.

Dutta, Sourav, Peter Rivera-Casillas, Brent Styles, and Matthew W Farthing. 2022. "Reduced Order Modeling Using Advection-Aware Autoencoders." *Mathematical and Computational Applications* 27 (3): 34.

Jasrasaria, Dipti, and Edward O Pyzer-Knapp. 2018. "Dynamic Control of Explore/Exploit Trade-off in Bayesian Optimization." In *Science and Information Conference*, 1–15. New York: Springer.

Karniadakis, George Em, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. "Physics-Informed Machine Learning." *Nature Reviews Physics* 3 (6): 422–40.

Lee, Kookjin, and Eric J. Parish. 2021. "Parameterized Neural Ordinary Differential Equations: Applications to Computational Physics Problems." *Proceedings of the Royal Society A* 477 (2253): 20210162.

Luong, Phuc, Sunil Gupta, Dang Nguyen, Santu Rana, and Svetha Venkatesh. 2019. "Bayesian Optimization with Discrete Variables." In *Australasian Joint Conference on Artificial Intelligence*, 473–84. New York: Springer.

Mao, Zhiping, Ameya D. Jagtap, and George Em Karniadakis. 2020. "Physics-Informed Neural Networks for High-Speed Flows." *Computer Methods in Applied Mechanics and Engineering* 360:112789.

Martinez-Gonzalez, Daniel A., Dylan Jude, and Andrew M. Wissink. 2022. "ROAM-ML: A Reduced Order Aerodynamic Module Augmented with Neural Network Digital Surrogates." In *AIAA SCITECH 2022 Forum*, 1248.

O'Malley, Tom, Elie Bursztein, James Long, Francois Chollet, Haifeng Jin, Luca Invernizzi, et al. 2019. Keras Tuner. https://github.com/keras-team/keras-tuner.

Rackauckas, Christopher, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, et al. 2020. "Universal Differential Equations for Scientific Machine Learning." arXiv preprint arXiv:2001.04385.

Raissi, Maziar, and George Em Karniadakis. 2018. "Hidden Physics Models: Machine Learning of Nonlinear Partial Differential Equations." *Journal of Computational Physics* 357: 125–41.

Raissi, Maziar, Paris Perdikaris, and George E Karniadakis. 2019. "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.

Rivera-Casillas, Peter Gabriel Rivera, Matthew Farthing, Daniel Martinez-Gonzalez, and Wesley Brewer. 2020. "A Survey on Physics-Informed Neural Networks for Shallow Water Problems." In *AGU Fall Meeting* 2020. AGU.

Sankaran, Venkateswaran, Jayanarayanan Sitaraman, Andrew Wissink, Anubhav Datta, Buvana Jayaraman, Mark Potsdam, Dimitri Mavriplis, et al. 2010. "Application of the Helios Computational Platform to Rotorcraft Flowfields." AIAA paper 1230: 2010.

Sun, Yifan, Linan Zhang, and Hayden Schaeffer. 2020. "Neupde: Neural Network Based Ordinary and Partial Differential Equations for Modeling Time-Dependent Data." In *Mathematical and Scientific Machine Learning*, 352–372. PMLR.

Wang, Jian-Xun, Jin-Long Wu, and Heng Xiao. 2017. "Physics-Informed Machine Learning Approach for Reconstructing Reynolds Stress Modeling Discrepancies Based on DNS Data." *Physical Review Fluids* 2 (3): 034603.

Willard, Jared, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2020. "Integrating Physics-Based Modeling with Machine Learning: A survey." arXiv preprint arXiv:2003.04919 1 (1): 1–34.

Yamakawa, George M., Donald G. Broadhurst, and John R. Smith. 1972. *Utility Tactical Transport Aircraft System (UTTAS) Maneuver Criteria*. Technical report. Army Aviation Systems Test Activity, Edwards AFB, Edwards, CA.

Abbreviations

ALM	Actuator line model
CREATE-AV	Computational Research and Engineering Acquisition Tools, Air Vehicle Design
ELU	Exponential linear unit
HPCMP	High-Performance Computing Modernization Program
KT	Keras Tuner
ML	Machine learning
MSE	Mean squared error
NN	Neural network
NODE	Neural Ordinary Differential Equations
ODE	Ordinary differential equation
PDE	Partial differential equation
PIML	Physics-informed machine learning
RANS	Reynolds-averaged Navier–Stokes
ReLU	Rectified linear unit
RK	Runge-Kutta
RK4	Fourth-order Runke-Kutta
SGD	Stochastic gradient descent
Tanh	Hyperbolic tangent
UTTAS	Utility Tactical Transport Aircraft System

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
April 2024	Final Technical Report (TR)	FY21	FY22

4. TITLE AND SUBTITLE

Neural Ordinary Differential Equations for Rotorcraft Aerodynamics

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT
		0603465A
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
AL3	SAL301	

6. AUTHOR(S)

Peter Rivera-Casillas and Ian Dettwiller

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

US Army Engineer Research and Development Center (ERDC)
Information Technology Laboratory (ITL)
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC/ITL TR-24-6

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Headquarters, US Army Corps of Engineers Washington, DC 20314-1000	HQUSACE	

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A. Approved for public release: distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

High-fidelity computational simulations of aerodynamics and structural dynamics on rotorcraft are essential for helicopter design, testing, and evaluation. These simulations usually entail a high computational cost even with modern high-performance computing resources. Reduced order models can significantly reduce the computational cost of simulating rotor revolutions. However, reduced order models are less accurate than traditional numerical modeling approaches, making them unsuitable for research and design purposes. This study explores the use of a new modified Neural Ordinary Differential Equation (NODE) approach as a machine learning alternative to reduced order models in rotorcraft applications—specifically to predict the pitching moment on a rotor blade section from an initial condition, mach number, chord velocity and normal velocity. The results indicate that NODEs cannot outperform traditional reduced order models, but in some cases they can outperform simple multilayer perceptron networks. Additionally, the mathematical structure provided by NODEs seems to favor time-dependent predictions. We demonstrate how this mathematical structure can be easily modified to tackle more complex problems. The work presented in this report is intended to establish an initial evaluation of the usability of the modified NODE approach for time-dependent modeling of complex dynamics over seen and unseen domains.

15. SUBJECT TERMS

Helicopters--Design; High performance computing; Numerical analysis; Rotors (Helicopters)--Aerodynamics--Computer simulation; Rotors (Helicopters)--Structural dynamics--Computer simulation; Systems engineering

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	36

19a. NAME OF RESPONSIBLE PERSON	19b. TELEPHONE NUMBER (include area code)