(Preprint) AAS 25-867

CHARACTERIZATION OF NEURAL ORDINARY DIFFERENTIAL EQUATIONS FOR ASTRODYNAMICS APPLICATIONS

Sarah Wielgosz[*], Huan Xu[†], and John R. Martin[‡]

INTRODUCTION

Accurate dynamics models are critical for reliable mission design and analysis. While there exist high-fidelity models for perturbing forces such as solar radiation pressure, atmospheric drag, and non-uniform gravity fields, these models are seldom universally valid. There inevitably exists a gap between these analytic models and the dynamics experienced in practice. These may come from effects that are unknown a-priori or high-order forces simply beyond the reach of popular analytic models. While improvements to physics-based modeling and numerical methods can partially address these gaps, such approaches often become computationally expensive or require detailed knowledge of the system. Consequently, there is growing interest in augmenting existing models with data-driven techniques capable of capturing unknown or partially known dynamics in a more flexible and efficient manner. By improving our understanding of these forces and our consequent dynamic modeling, we can reduce the frequency and magnitude of corrective maneuvers, ultimately saving fuel, minimizing risk, and lowering cost.

Neural ordinary differential equations (neural ODEs) offer a powerful framework for learning models of complex dynamical systems in a data-driven fashion. Neural ODEs use a neural network to represent the dynamics of a system as a continuous-time differential equation, enabling the model to learn how the system evolves over time directly from data. This approach is particularly well-suited for representing highly nonlinear, time-dependent behaviors, making it a natural fit for challenging dynamics problems. Their compatibility with standard numerical integration tools further enhances their appeal. While neural ODEs have shown significant promise in broader machine learning and physics domains, their potential in astrodynamics remains largely unexplored, offering a valuable direction for research.

In this work, we aim to provide a preliminary overview of the utility and capabilities of neural ODEs applied to popular astrodynamics systems. In particular, our efforts focus on using Neural ODEs to model two canonical problems——the planar two-body problem (2BP) and the planar Circular Restricted Three-Body Problem (CR3BP). These benchmark systems allow us to validate the neural ODE's ability to recover known dynamics under idealized conditions by constructing models

[*]NSF Graduate Research Fellow, Department of Aerospace Engineering, University of Maryland, College Park, Martin Hall Bldg #088 College Park, MD 20742

[†]Associate Professor, Department of Aerospace Engineering, University of Maryland, College Park, Martin Hall Bldg #088 College Park, MD 20742

[‡]Assistant Professor, Department of Aerospace Engineering, University of Maryland, College Park, Martin Hall Bldg #088 College Park, MD 20742

of the form

$$\frac{d\mathbf{x}}{dt} = f_\theta(\mathbf{x}, t), \tag{1}$$

where f_θ represents the dynamical model (e.g., 2BP or CR3BP) parameterized by a neural network and trained using state data from families of sampled trajectories.

For this work, we prioritize the coarse characterization of neural ODEs ability to recover the underlying dynamics of the training data and their generalization capabilities when tested beyond the bounds of this data. In doing so, we identify important training considerations that enable neural ODEs to recover unknown dynamics with acceptable fidelity. This work highlights the strengths, limitations, and potential of neural ODEs as tools for augmenting traditional models, particularly in applications where effective control relies on accurate dynamical knowledge. By exploring model sensitivity to data and design choices, our objective is to establish practical guidelines for applying neural ODEs to real-world astrodynamics systems, ultimately paving the way for more accurate, fuel-efficient and autonomous space missions.

BACKGROUND

Historically, the modeling of spacecraft dynamics has been rooted in physics-based models. Classical approaches use Newtonian mechanics, Lagrangian or Hamiltonian formulations, and gravitational models to derive equations of motion. These models form the foundation of analytical orbit propagators and enable deterministic trajectory design and control.

When physical modeling becomes challenging due to system complexity or incomplete knowledge, engineers have turned to system identification techniques. Methods such as auto-regressive models,[1] linear state-space identification,[2] and Kalman filtering[3] enable estimation of dynamic behavior from observed data, particularly when system inputs and outputs are well-characterized.

For high-dimensional dynamical systems, modal decomposition methods offer a powerful means of extracting essential features and reducing complexity. Techniques such as Dynamic Mode Decomposition[4] and its theoretical foundation in Koopman operator theory[5] enable a data-driven analysis of system behavior by decomposing complex spatiotemporal evolution into a set of modes, each associated with a fixed oscillation frequency and growth/decay rate. These methods reveal spatial patterns that evolve in time according to simple rules, which can be particularly valuable for understanding and modeling complex phenomena. Unlike traditional model-based approaches, these decompositions require no prior knowledge of the underlying governing equations. Instead, they rely solely on observed data, enabling compact representations of system behavior even when a full physical model is intractable or unknown. This ability to linearize nonlinear dynamics in a higher-dimensional function space makes modal decomposition a cornerstone of modern system identification and control strategies in high-dimensional settings.

More recently, advances in symbolic modeling, such as Sparse Identification of Nonlinear Dynamics (SINDy)[6] and symbolic regression, have aimed to recover interpretable governing equations directly from data. In SINDy, one first assembles a library of candidate nonlinear functions of the system's state and then solves a sparse regression problem—often via sequential thresholded least-squares or LASSO—to identify only the few active terms that best reproduce the observed time derivatives. By enforcing sparsity, SINDy yields compact, interpretable models that capture the dominant dynamics without overfitting to spurious correlations. However, these methods often struggle in noisy or high-dimensional settings.

Modern developments in machine learning, particularly the advent of neural networks and deep learning, have introduced new avenues for modeling. Neural ODEs, Physics-Informed Neural Networks (PINNs), and latent dynamics models offer flexible frameworks for capturing nonlinear behavior, especially in partially known or data-rich regimes. These data-driven methods are now being explored as complements or alternatives to traditional dynamics models in guidance, navigation, and control (GNC) systems. A particularly promising alternative lies in the use of neural ODEs, which adopt a data-driven approach to learning complex dynamical systems. By modeling spacecraft dynamics through neural ODEs, more efficient and adaptive trajectory planning becomes feasible, especially in regimes where physical models are incomplete or uncertain.

Originally introduced by Chen et al.,[7] neural ODEs have emerged as a powerful framework for modeling continuous-time dynamics using neural networks. Unlike traditional feedforward architectures, they define the hidden state evolution as a differential equation parameterized by a neural network, allowing for smooth interpolation over time and memory-efficient training. These models have shown success across various scientific and engineering domains, including fluid dynamics,[8] robotics,[9],[10] and spacecraft motion[11] where complex, nonlinear behavior is difficult to capture using classical parametric models. Their ability to learn governing equations directly from observational data makes them particularly well-suited for systems with unknown or partially known dynamics.

Several recent efforts have explored neural ODEs and other scientific machine learning algorithms to improve control performance. For example, Ueda and Owaga use neural ODEs to learn multi-objective control laws in settings with known dynamics. Similarly, Origer et al.[12] explore the application of neural ODEs for optimizing guidance and control networks to achieve a time-optimal interplaneteray transfer and mass-optimal landing on an asteroid under known dynamics. Meanwhile, others have employed PINNs to embed physical constraints into the learning process for spacecraft control laws.[13,14] These studies, while promising, remain focused on augmenting systems where at least partial models are known.

Another compelling use case for neural ODEs in astrodynamics is orbit propagation. In low Earth orbit (LEO), the growing presence of satellites and debris heightens collision risk. Subramanian et al.[15] leverage physics-informed neural ODEs and historical spacecraft data to accurately predict orbital trajectories and assess collision probability, enabling more timely and informed maneuver decisions. These physics-informed approaches embed known physical laws directly into the training process, enhancing model generalization and reliability.

Earlier attempts to infer unmodeled dynamics relied on symbolic regression methods, such as those proposed by Manzi et al.[16] However, symbolic regression tends to struggle in high-dimensional, noisy environments due to its reliance on interpretable but simplified representations.

A notable contribution in learning unmodeled dynamics comes from Murphy and Scheeres,[11] who combine neural ODEs with symbolic regression to augment spacecraft dynamic models from tracking data. In their framework, neural networks are first used to model discrepancies between observed spacecraft motion and predictions from known physics-based models. Symbolic regression is then applied to extract interpretable mathematical expressions that describe these residual dynamics, enabling both data-driven adaptation and physical insight. This hybrid approach not only improves prediction accuracy in the presence of unmodeled forces—such as irregular gravity fields or perturbations—but also facilitates integration with existing analytical models. Their work underscores the potential for neural differential equations to adaptively learn dynamic behaviors directly from measurements while maintaining interpretability and physical consistency.

In this paper, we aim to generalize these advances by providing a neural ODE framework which can be utilized to learn coarse representations of the full-system dynamics in lieu of prespecified base models. Our approach provides a preliminary characterization of neural ODE behavior across a range of scenarios. These efforts illustrate the growing role of neural ODEs in modern machine learning, offering improvements in orbit propagation with future implications on onboard autonomy.

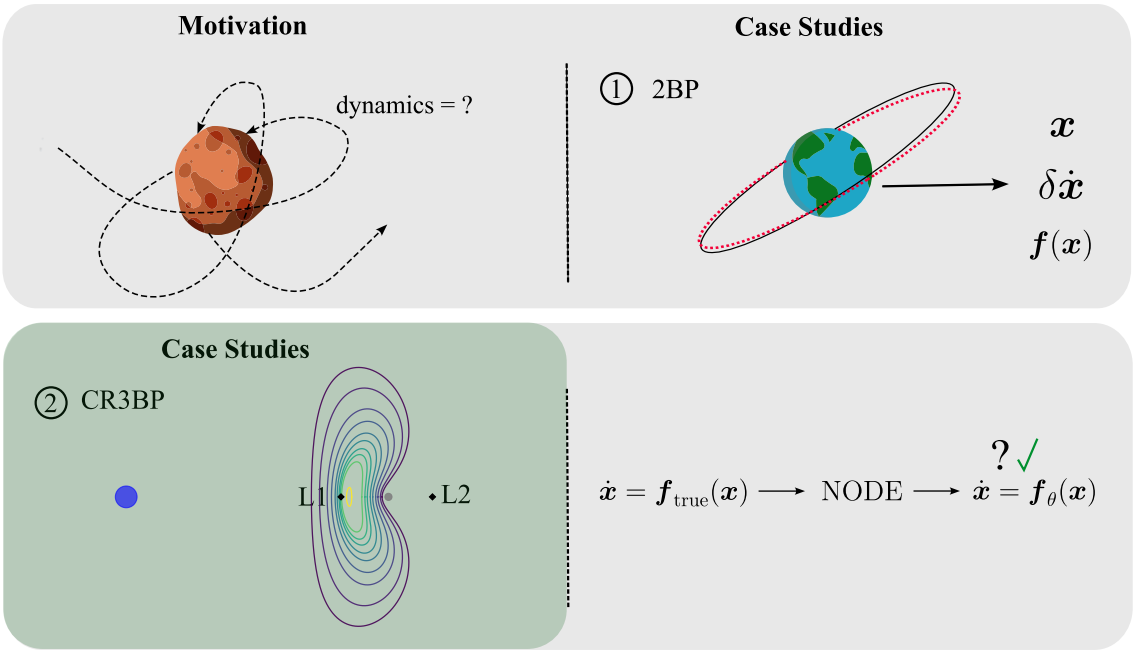


Figure 1: Training neural ODE to learn known and unknown dynamics.

METHODOLOGY

Neural ODEs

Neural ODEs provide a continuous-time framework in which the derivative of a hidden state is parameterized by a neural network. This structure replaces the layer-by-layer transformations in traditional deep neural networks with the solution of an initial value problem:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta), \tag{2}$$

where $\mathbf{h}(t)$ is the hidden state at time t, f is a neural network with parameters θ, and the dynamics are learned directly from data. Given an initial condition $\mathbf{h}(0)$, a black-box differential equation solver is used to compute the output $\mathbf{h}(T)$ at any time T.[7] This formulation yields a continuous-time model that is particularly well-suited for irregularly sampled time series, allowing predictions at arbitrary time points and removing the need for fixed-step discretization.

Training with the Adjoint Method. A key innovation introduced in[7] is the use of the adjoint sensitivity method to enable efficient backpropagation through the ODE solver. Rather than storing the entire forward trajectory during training, which can be memory-intensive, the adjoint method solves a second differential equation backward in time to compute gradients with respect to the

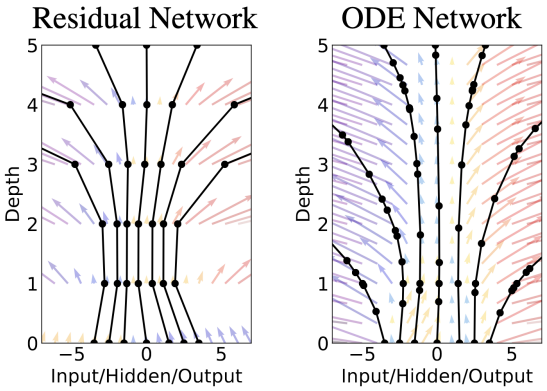


Figure 2: Finite transformations defined by residual networks (left) versus continuous transformations defined by neural ODEs (right).[7]

hidden states:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\intercal \frac{\partial f}{\partial \mathbf{h}}, \tag{3}$$

where $\mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{z}(t)}$ is the adjoint state. The gradients with respect to the model parameters can now be represented as an augmented ODE calculated as

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \mathbf{a}(t)^\intercal \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt \tag{4}$$

where the vector-Jacobian products are efficiently calcualted by automatic differentiation. This yields constant memory cost during training, making Neural ODEs scalable to long time horizons and deep continuous transformations. The solver is treated as a differentiable black-box, allowing seamless integration with existing deep learning pipelines.

In summary, neural ODEs are advantageous for learning dynamics because they provide a data-driven method of modeling continuous-time dynamics, they are memory efficient, These features make Neural ODEs a powerful tool for modeling the evolution of complex dynamical systems, including spacecraft trajectories under uncertain or perturbed conditions.

Comparison to Prior Dynamics Learning Methods. Neural ODEs distinguish themselves from classical approaches to learning dynamical systems, such as Dynamic Mode Decomposition (DMD),[4] which approximates the Koopman operator to model linear dynamics in a lifted feature space. While DMD and its variants offer interpretability and closed-form solutions under specific assumptions, they often require uniform sampling and may struggle with complex nonlinearities present in real-world data. In contrast, Neural ODEs are fully nonlinear, data-driven models that can flexibly adapt to arbitrary dynamics without restrictive assumptions on the system or measurement process.

Canonical Astrodynamic Systems

Two-body Problem We consider the relative form of the two-body problem (2BP) equation which describes the motion of a satellite orbiting a primary gravitating body. We assume that the satellite's mass is insignificant relative to that of the primary gravitating body, and its motion is only affected by gravity of the primary body. The two-body equations of motion are given as

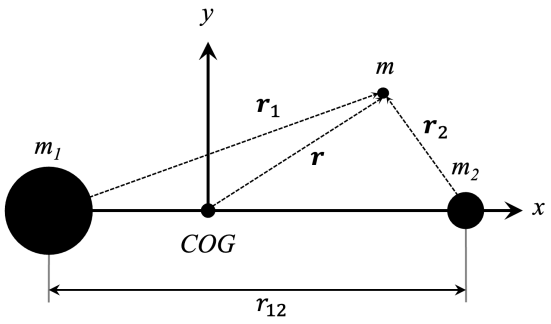


Figure 3: CR3BP in the rotating reference frame.

$$\ddot{\vec{r}} = -\frac{\mu}{r^2}\frac{\vec{r}}{r} \tag{5}$$

where μ is the gravitational parameter of the primary body and $\vec{r}$ is the position vector of the satellite in Cartesian coordinates.

To improve numerical performance, it is common to nondimensionalize the system by characteristic quantities. A characteristic length, l^*, is chosen to reflect the scale of the particular problem. For example, for an interplanetary orbit, it is common to set $l^* = 1$ AU. Characteristic time, t^*, is defined as

$$t^* = \sqrt{\frac{l^{*3}}{\mu}} \tag{6}$$

such that the non-dimensional gravitational parameter, μ^*, is equal to 1.

Circular Restricted Three-body Problem We consider the nondimensionalized form of the circular restricted three-body problem (CR3BP), which models the motion of a satellite of negligible mass under the gravitational influence of two bodies orbiting their barycenter in circular orbits. The primary body has mass m_1, and the secondary body has mass m_2 The dynamics are described in a rotating reference frame where the primaries remain fixed on the x^*-axis.

The nondimensional mass ratio is

$$\pi = \frac{m_2}{m_1 + m_2}, \tag{7}$$

and the nondimensional coordinates in the rotating frame are

$$\begin{aligned} x^* &= \frac{x}{r_{12}} \\ y^* &= \frac{y}{r_{12}} \\ z^* &= \frac{z}{r_{12}}. \end{aligned} \tag{8}$$

The nondimensional distances from the satellite to the primary and secondary bodies are found respectively as

$$\sigma = \sqrt{(x^* + \pi)^2 + y^{*2} + z^{*2}}$$
$$\psi = \sqrt{(x^* - 1 + \pi)^2 + y^{*2} + z^{*2}}, \tag{9}$$

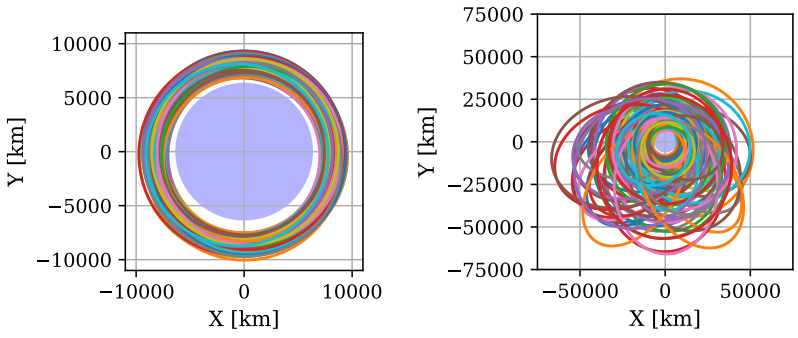
and then finally the nondimensional equations of motion are given as

$$\begin{aligned}
\ddot{x}^* - 2\dot{y}^* - x^* &= -\frac{1-\pi}{\sigma^3}(x^* + \pi) - \frac{\pi}{\psi^3}(x^* - 1 + \pi) \\
\ddot{y}^* + 2\dot{x}^* - y^* &= -\frac{1-\pi}{\sigma^3} y^* - \frac{\pi}{\psi^3} y^* \\
\ddot{z}^* &= -\frac{1-\pi}{\sigma^3} z^* - \frac{\pi}{\psi^3} z^*.
\end{aligned} \tag{10}$$

PROBLEM STATEMENT

We conduct a series of case studies across increasingly complex dynamical regimes to evaluate the neural ODE's ability to model relevant spacecraft dynamics. The first study explores the recovery of planar 2BP dynamics around Earth, and the second explores the planar dynamics around Lagrange points in the Earth-Moon three-body system. These studies train the neural ODEs on datasets that cover increasingly large regions of the state space before then testing their generalization capabilities to previously unseen dynamical regimes.

Datasets In training the 2BP, we define the simple and complex datasets which will be referenced herein. The simple dataset refers to one whose bounds on orbital elements are close to a circular, equatorial, low Earth orbit with limited orientation changes. Conversely, the complex dataset spans a variety of regimes around Earth, from low Earth orbit to geostationary altitudes, circular to highly eccentric, and a full range of orbital orientations. The simple and complex training datasets for the 2BP can be seen in Figures 4a and 4b, respectively. In both cases, initial conditions are randomly chosen from a uniform distribution of specified orbital element ranges as detailed in Table 1.



(a) Simple training dataset (b) Complex training dataset

Figure 4: Two-body problem training data in the simple and complex case.

Table 1: Bounds of orbital elements for simple versus complex 2BP datasets.

Orbital Element	Simple	Complex
Semi-major axis	$[R_E + 1000,\ R_E + 3000]$ km	$[R_E + 500,\ R_E + 20000]$ km
Eccentricity	$[0,\ 0.1]$	$[0,\ 1.0]$
Argument of periapsis	$[0,\ 180]^\circ$	$[0,\ 180]^\circ$
Initial true anomaly	$[0,\ 360]^\circ$	$[0,\ 360]^\circ$
Inclination	0°	0°

To generate training data for the CR3BP, we again build out increasingly complex datasets. To compute periodic planar orbits around all of the Lagrange points we utilize numerical shooting methods developed by Jain.[17] In order of increasing complexity, our training datasets include a single Lyapunov orbit around Lagrange point 1, a family of Lyapunov orbits around Lagrange point 1 found via varying the Jacobi constant, families of Lyapunov orbits around the collinear Lagrange points 1 through 3, and finally families of planar orbits around all Lagrange points (Lyapunov orbits around collinear Lagrange points 1 through 3, short-period orbits around triangular Lagrange points 4 and 5). The training dataset for all planar families is shown in Figure 5. The datasets descriptions and their associated names used herein are summarized in Table 3.

Table 2: Overview of planar CR3BP training datasets and their reference Lagrange points.

Dataset Name	Reference Lagrange Point(s)	Num Orbits per Lagrange Point
L1 Orbit	L1	1
L1 Family	L1	50
Co-Linear Families	L1, L2, L3	50
All Families	L1, L2, L3, L4, L5	50

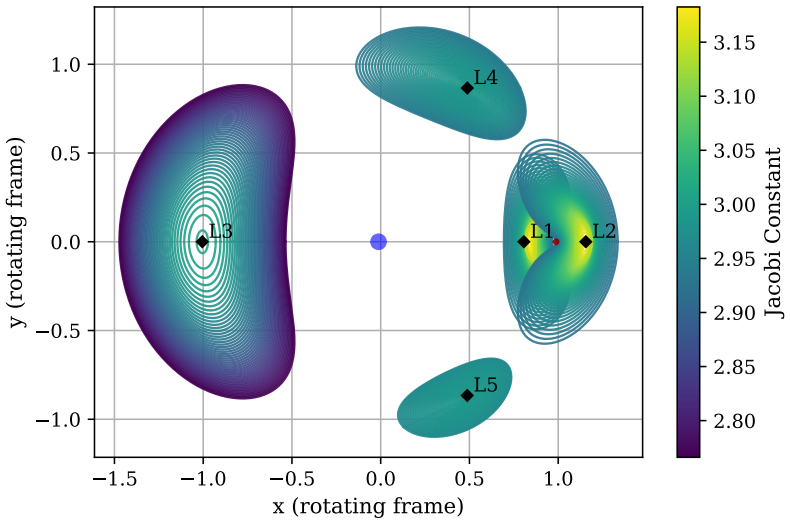


Figure 5: Training dataset for planar orbits families at all Lagrange points.

Table 3: Overview of planar CR3BP training datasets and their reference Lagrange points.

Dataset Name	Reference Lagrange Point(s)	Num Orbits per Lagrange Point
L1 (LO1)	L1	1
L1 Family (LF1)	L1	50
Co-Linear Families (CLF)	L1, L2, L3	50
All Families (ALF)	L1, L2, L3, L4, L5	50

Training Details

A neural network is randomly initialized. This network parameterizes the continuous-time dynamics of the hidden states, making the neural ODE framework well-suited for scientific machine learning problems involving systems governed by physical laws. Given a set of initial orbital conditions, the network is used to propagate the state forward in time via integration of the learned dynamics. A scalar loss defined as the percent error between the predicted and true states is computed. Gradients of this loss with respect to the network parameters are then calculated using the adjoint method[7] and used to update the model. Notably, the default training process for neural ODEs is not naturally stable, particularly when integrated over long time intervals. Strategies to encourage model convergence are discussed below.

Activation Function The activation function used for training both the 2BP and CR3BP models is the hyperbolic tangent activation function, $\tanh$. It is well suited for neural ODE models since it is smooth, bounded, and infinitely differentiable. Infinite differentiability is critical in training neural ODEs because in the adjoint sensitivity method, gradients are computed by integrating the augmented ODE backward in time. Since $\tanh$ is smooth and infinitely differentiable, adjoint dynamics are consequently smooth and continuous resulting in a numerically stable backward integration.

Feature Layers Feature layers for the 2BP and CR3BP are chosen to provide the network with strategic features which improve learning efficiency and generalization. In our neural ODE model for the 2BP, the feature layer is composed of the reciprocal distance $1/r$, the unit direction vector $[s, t, u]$, and the Cartesian velocity vector $[v_x, v_y, v_z]$ based on the dynamics of the 2BP described in Equation 5. Notably the true vector field for the 2BP does not rely on the velocity vector; however, we find the the inclusion of these states in the feature layer afford the network greater stability during training.

For the CR3BP, we adopt a feature layer similar to the 2BP case, but adapted to the rotating frame. The feature layer is composed again of the reciprocal distance, unit direction, and velocity vector (all with respect to the rotating frame), and we additionally include $1/r_1$ and $1/r_2$, the inverse distances to the primary and secondary bodies, and the Jacobi constant, JC which encodes the conserved energy level of each periodic orbit of interest.

Output Layers In a similar manner, neural ODE outputs can be suffixed with a final transformation layer that can be used to transform numerically favorable outputs into the physically meaningful dynamics. This is accomplished by designing a neural ODE with four outputs that correspond to (a_r, a_s, a_t, a_u), where a_r is the acceleration magnitude, and $a_i \forall i \in \{s, t, u\}$ are the unit magnitudes in the x, y, and z directions respectively. These values are converted to a_x, a_y, a_z via multiplication with a_r, and then appended with the velocity vector of the initial state to form the final dynamics

output — i.e.

$$\mathbf{X} \rightarrow f_\theta(t) \rightarrow \dot{\mathbf{z}} \rightarrow g(\dot{\mathbf{z}}, \mathbf{X}) \rightarrow \dot{\mathbf{X}} \tag{11}$$

where $\dot{\mathbf{z}} = [a_r, a_s, a_t, a_u]$ and $g(\dot{\mathbf{z}}, \mathbf{X}) = [v_x, v_y, v_z, a_r \cdot a_s, a_r \cdot a_t, a_r \cdot a_u]$.

Training Curriculum When training neural ODEs, a length strategy is often implemented. This curriculum learning approach involves training the model on only a subset of the full time series for each training example. In doing so, model efficiency is improved by training on a limited data subset before increasing longer sequences of training data.

However, we directly observed through model development and training that if length strategies become too long, the model may suffer from vanishing gradients. As integration time increases, the augmented ODE described in Equation 4 is dominated by a decaying integrand, resulting in vanishing gradients. Consequently, the model cannot make useful updates and it fails to train sufficiently. To mitigate this behavior, we segmented each trajectory to decrease the time horizon of the subdivided trajectories. In doing so, integration time was limited such that the gradients were not dominated by decaying terms and the gradient remained stable.

Evaluation Metrics Model success is evaluated based on its accuracy in reproducing instantaneous dynamics. Therefore we assess the performance of the neural ODE framework via the mean percent error of the acceleration residual L2 norm. This reflects the fidelity of the learned dynamical model and ultimately indicates its utility for accurate state propagation over time.

RESULTS & DISCUSSION

Two-Body Problem

Dynamics of the 2BP are learned from training on a single simple orbit, the simple dataset, and the complex dataset. Every orbit trajectory was segmented into 20 intervals equal in eccentric anomaly. Each segment then represented a unique trajectory passed to the neural ODE model. The model parameters are summarized in Table 4.

Table 4: Neural ODE parameters for learning 2BP dynamics.

Parameter	Value
Input Features	$[1/r,\ s,\ t,\ u,\ v_x,\ v_y,\ v_z]$
Network Width	64
Network Depth	2
Batch Size	256
Loss Function	State percent error
Activation Function	$\tanh$
ODE Solver	Tsitouras 5/4 Runge–Kutta
ODE Solver Absolute Error	1×10^{-8}
ODE Solver Relative Error	1×10^{-6}
Optimizer	ADAM
Learning Rate Strategy	0.001
Steps Strategy	[500, 500, 500, 500, 500, 500, 500, 500, 500, 5000]
Length Strategy	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

Train \ Test	Single	Simple	Complex
Single	5.9173293	5.748998	11.683002
Simple	2.6530995	2.393616	9.857474
Complex	0.50465333	0.48402855	0.61609894

Table 5: Mean Acceleration Error for Different Datasets

To test the generalization ability of each of the trained model, we observe the cross-domain acceleration error in Table 5. A limited number of the associated propagated trajectories are visualized in Figure 6. As the complexity of the training dataset increases, the cross-domain error decreases. Conversely, the model trained on a single orbit or a simple dataset has difficulty generalizing to more complex domains. Both of these results are intuitive – as the model is exposed to increased phase space, its ability to generalize improves. We additionally observe that as complexity of the dataset increases and the model is exposed to increased phase space, the ability of the model to learn the dynamics of its own training dataset improves as exemplified by the errors on the diagonal in Table 5.

The results in Table 5 and Figure 6 can be intuitively interpreted by examining the difference in the acceleration residual vectors of the simple and complex models applied to the same testing dataset. In Figure 7, we apply the simple and complex models to a highly eccentric orbit with an eccentricity of 0.8326 and a semi-major axis of 38,268.78 km and observe the absolute acceleration residuals (scaled for visibility). We observed larger acceleration residuals when applying the simple model compared to the complex model because the simple model is being applied to unseen phase space. The simple model underpredicts accelerations as it reaches apoapsis and the absolute error increases as it approaches phase spaces further from its training domain. The complex model, however, estimates the dynamics very closely with only a slight overprediction of the accelerations.

The success of the neural ODE model in learning two-body dynamics exemplifies the potential of neural ODE models to be used in astrodynamic applications. While the dynamics in this case are simple, training the model provided valuable insight into improving neural ODE efficacy. Most importantly, without orbit segmentation we observed vanishing gradients due to long integration times and the model could not learn two-body dynamics to any acceptable accuracy. By segmenting datasets, we decreased the time horizon and achieved successful model training.

Circular Restricted Three-Body Problem

Dynamics of the planar orbits about the Lagrange points in the CR3BP are learned from training on increasingly complex datasets as described in Table 3. Compared to the 2BP, CR3BP dynamics pose a significantly greater challenge for neural ODE models due to their increased complexity. The results presented here are preliminary and represent a lower bound on model performance; efforts to improve learning complex dynamics are ongoing. Nonetheless, these initial results provide valuable insight into the training process. They highlight key considerations for developing more effective neural ODE models, while demonstrating promising results for future model improvement.

When training the CR3BP, each trajectory containing 1000 evenly spaced timesteps was segmented into trajectories of length 18 timesteps. Like in the 2BP model, each segment was treated as a unique trajectory when passed to the neural ODE model. The CR3BP model parameters are

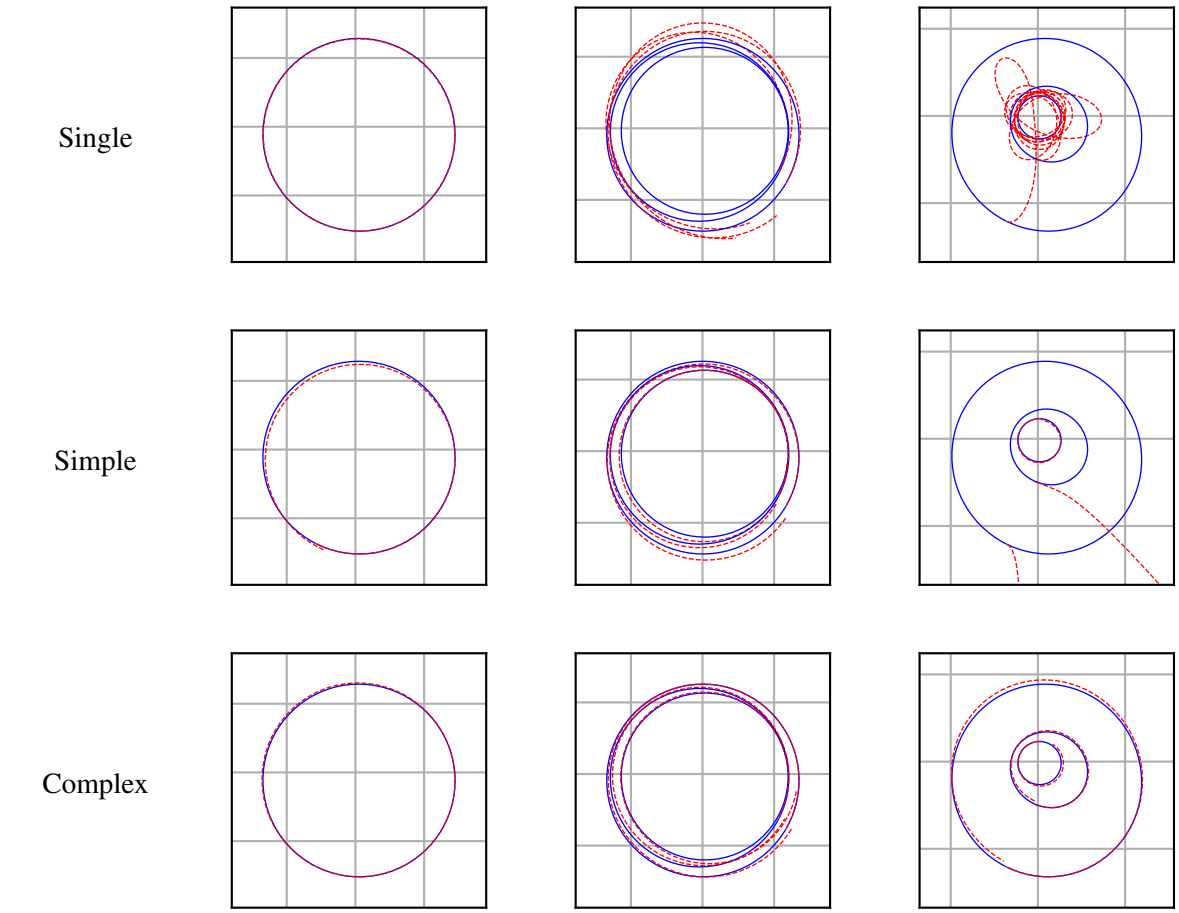


Figure 6: Integrated Orbits for Different Datasets

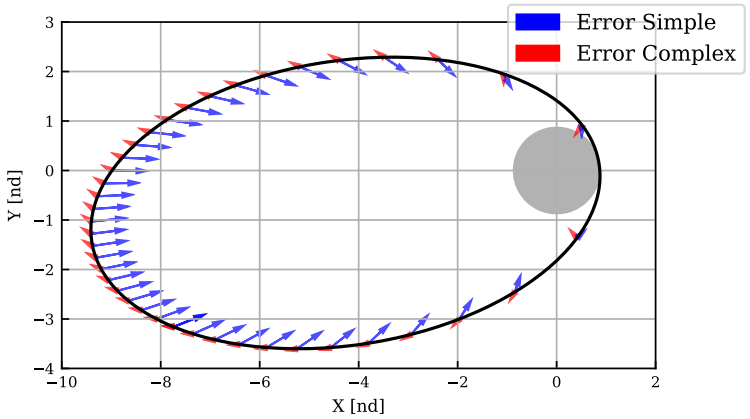


Figure 7: Resultant acceleration error vectors from training on simple dataset (blue) vs complex dataset (red). The test orbit has an eccentricity of 0.8326 and a semi-major axis of 38,258.78 km. Residual errors are shown in non-dimensionalized frame.

summarized in Table 6.

Table 6: Neural ODE parameters for learning 3BP dynamics.

Parameter	Value
Input Features	$[1/r,\ s,\ t,\ u,\ v_x,\ v_y,\ v_z, 1/r_1, 1/r_2, JC]$
Network Width	64
Network Depth	4
Batch Size	64
Loss Function	State percent error
Activation Function	$\tanh$
ODE Solver	Tsitouras 5/4 Runge–Kutta
ODE Solver Absolute Error	1×10^{-8}
ODE Solver Relative Error	1×10^{-6}
Optimizer	ADAM
Learning Rate Strategy	0.005
Steps Strategy	[500, 500, 500, 500, 500, 500, 500, 500, 500, 5000]
Length Strategy	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

We observe the cross-domain acceleration error across increasing complex training and testing datasets. The errors are quantified in Table 7, and a selection of propagated orbits are visualized in Figure 8. Due to the increased complexity of the problem, we see decreased accuracy as compared to the two-body model, as expected. However, current errors from the coarse model are encouraging and exemplify sensible trends – training on increasingly complex datasets improves the model's ability to estimate the dynamics, while models trained on limited datasets suffer in their ability to generalize. When observing the error of models tested on datasets laying within training data bounds, acceleration errors are sufficiently low enough that we are confident that further model optimization will yield improved and reliable results.

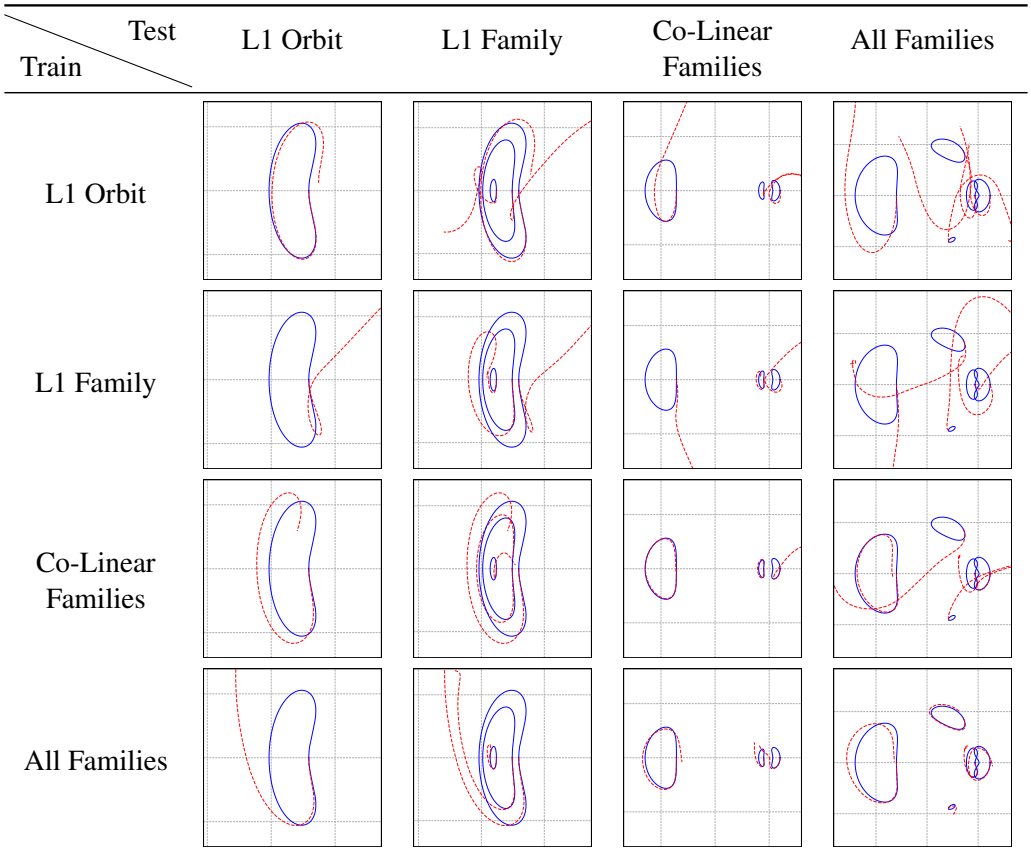


Figure 8: Integrated Orbits for Different Datasets

Train \ Test	L1 Orbit	L1 Family	Co-Linear Families	All Families
L1 Orbit	3.4156537	43.83782	65.76061	277.37698
L1 Family	7.428957	8.860915	62.909714	351.03394
Co-Linear Families	6.4854164	7.704996	6.9688735	431.7849
All Families	7.5511627	7.855743	8.237325	14.695192

Table 7: Mean Acceleration Error for Different Datasets

To further observe the predicted dynamics of the neural ODE, we examine the acceleration residuals for the model trained on one orbit around Lagrange point 1, referred to as model LO1, versus the model trained on families of orbits around all Lagrange points, referred to as model ALF. These residuals are shown in Figure 9. Model LO1 has higher errors than the model ALF as expected due to limited exposure to the relevant phase space. Additionally, acceleration errors appear unbiased for model ALF while they are radially biased for model LO1. This is a result of the larger phase space exposure for model ALF. The lack of bias in error residuals indicates that model ALF is generalizing well to other datasets, which is supported by the errors presented in Table 7.

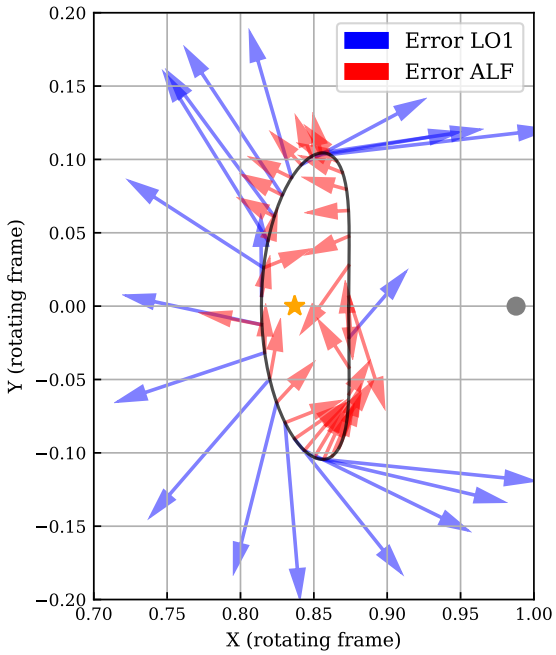


Figure 9: Resultant acceleration error vectors from training on a single orbit around Lagrange point 1 (blue) vs training on families of orbits at all Lagrange points (red). The model trained on a more complex dataset has higher success at predicting model dynamics.

Overall, the results demonstrate that neural ODEs are capable of learning meaningful representations of complex astrodynamical systems when trained with care. While performance on the CR3BP remains less accurate than in the 2BP case, the observed trends are encouraging. Models trained on more diverse data generalize more effectively, and acceleration errors remain moderate when testing models on orbits within their phase space. These findings affirm that neural ODEs can capture complex nonlinear gravitational dynamics and highlight the importance of training strategy, segmentation, and feature design. As model development continues, we anticipate that more systematic tuning and curriculum design will further enhance generalization and fidelity.

CONCLUSION

This work demonstrates the potential of neural ODEs as a flexible, data-driven framework for modeling complex astrodynamical systems. We exmplify this by successfully modeling the planar 2BP dynamics and coarsely modeling the planar CR3BP dynamics. Our results show that neural ODEs are capable of learning meaningful representations of underlying dynamics over long trajec-

tories and in large phase spaces, particularly when trained on sufficiently diverse datasets and with careful attention to model design and training strategy.

For the 2BP, neural ODEs achieve high accuracy and generalize well when exposed to a broad range of orbital regimes. In the more challenging CR3BP case, neural ODEs are able to capture key features of the dynamics, but performance is currently less accurate than in the 2BP scenario. These results are preliminary and should be interpreted as a lower bound on the achievable performance; further improvements are expected with continued development.

It is critical to note that training neural ODEs on highly diverse systems requires segmentation of trajectories even when underlying dynamics are simple, like in the case of the highly eccentric 2BP. Issues such as vanishing gradients, sensitivity to data scaling, training curriculum, model architecture, and input and output features are active areas of continued investigation. Nonetheless, the trends observed in this study are encouraging: as training data complexity increases, so does the model's ability to generalize and accurately reproduce the underlying dynamics.

Neural ODEs represent a promising tool for augmenting traditional astrodynamics models, particularly in regimes where unknown or partially known forces play a significant role. While the results presented here are a work in progress, they lay the groundwork for future research aimed at improving model fidelity, robustness, and applicability to real-world mission scenarios. With continued advances in training strategies and model architectures, neural ODEs have the potential to become a valuable component of next-generation space mission design and analysis.

REFERENCES

[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[2] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[4] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, Vol. 656, 2010, pp. 5–28.

[5] S. E. Otto and C. W. Rowley, "Koopman operators for estimation and control of dynamical systems," *Annual Review of Control, Robotics, and Autonomous Systems*, Vol. 4, No. 1, 2021, pp. 59–87.

[6] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, Vol. 113, No. 15, 2016, pp. 3932–3937.

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

[8] C. J. Rojas, A. Dengel, and M. D. Ribeiro, "Reduced-order model for fluid flows via neural ordinary differential equations," *arXiv preprint arXiv:2102.02248*, 2021.

[9] Z. Meleshkova, S. E. Ivanov, and L. Ivanova, "Application of Neural ODE with embedded hybrid method for robotic manipulator control," *Procedia Computer Science*, Vol. 193, 2021, pp. 314–324.

[10] M. Kasaei, K. K. Babarahmati, Z. Li, and M. Khadem, "A data-efficient neural ODE framework for optimal control of soft manipulators," *The Conference on Robot Learning 2023*, PMLR, 2023, pp. 2700–2713.

[11] J. Murphy and D. J. Scheeres, "SALAMANDER: Simulating and Leveraging Autonomous Model Augmentation Using Neural Differential Equations and (Symbolic) Regression," *AIAA SCITECH 2022 Forum*, 2022, p. 1763.

[12] S. Origer and D. Izzo, "Closing the gap: Optimizing guidance and control networks through neural odes," *arXiv preprint arXiv:2404.16908*, 2024.

[13] J. Varey, J. D. Ruprecht, M. Tierney, and R. Sullenberger, "Physics-Informed Neural Networks for Satellite State Estimation," *2024 IEEE Aerospace Conference*, IEEE, 2024, pp. 1–8.

[14] T. Goldman and K. Cowan, "An unsupervised physics-informed neural network for finding optimal low-thrust transfer trajectories with the direct method," *Association for Asian Studies Annual Conference 2024*, 2024, pp. AAS–24.

[15] S. Subramanian, R. Ramnani, S. Sengupta, and S. Yadav, "Orbit propagation from historical data using physics-informed neural odes," *2023 International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2023, pp. 1643–1648.

[16] M. Manzi and M. Vasile, "Discovering unmodeled components in astrodynamics with symbolic regression," *2020 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2020, pp. 1–7.

[17] D. Jain, "Astrodynamics_Research," `https://github.com/DhruvJ22/Astrodynamics_Research`, 2024. Accessed: 2025-08-01.