Regularizing Training of Physics Informed Neural Networks (PINNs) for Cislunar Orbit Determination via Transfer Learning

Gregory P. Badura,
Georgia Tech Research Institute
Miguel Velez-Reyes,
The University of Texas at El Paso

Brian Gunter,
Georgia Institute of Technology

Christopher R. Valenta,
Georgia Tech Research Institute

Koki Ho,
Georgia Institute of Technology

ABSTRACT

A recently introduced method for Orbit Determination (OD) is Physics Informed Neural Networks (PINNs). PINNs perform supervised prediction of cislunar satellite position while simultaneously respecting laws of orbital dynamics as described by nonlinear partial differential equations. The parameters of the PINN are trained to minimize a mean squared error loss that is a summation of (1) dynamics errors expressed under Circular Restricted 3 Body (CR3B) dynamics, and (2) measurement errors. The measurement error is a function of ancillary information (time and telescope location) and predicted satellite location, meaning that the true position of the observed satellite is not available to the PINN during the training phase. Due to lack of true state information, there can be no initial or bounding constraints placed on satellite position. Consequently, there is a high likelihood that a PINN gets trapped in a dynamically unstable gravitational region from which it cannot escape. We propose a repeatable method for training PINNs such that they avoid drifting into gravitationally unstable solutions: transfer learning. Transfer learning in this context means training the PINN to predict the position and velocity on an initialization trajectory prior to performing training on observed line of sight measurements. Transfer learning, in essence, primes the PINN parameters such that network biases and weights are not randomized at the beginning of training.

Our research shows that transfer learning results in more repeatable, faster, and more dynamically stable training of PINNs for OD. Qualitatively, we demonstrate the problem of poor parameter initialization of PINNs causing them to predict gravitationally unstable solutions and show how they are unable to escape due to explosion of the dynamical loss term. We also qualitatively show how transfer learning produces repeatable OD outcomes even when the initial state vector is randomized across the full span of a cislunar family's trajectory. Quantitatively, we demonstrate that transfer learning increases the percentage of runs for which acceptable OD trajectories are obtained. Our results show that transfer learning yields OD solutions that are within a 0.5 degree Field of View (FoV) of the true line of sight for the full trajectory at $\sim 4\times$ the rate of PINNs that have randomized initial weights (85% for transfer learning vs 19% for randomized weights). Furthermore, the mean position error of transfer learned solutions is $\sim$1E3 kms after training for 2000 epochs, as opposed to $\geq$1E4 kms for PINNs trained from a randomized initial state after $\sim$5000 epochs at the same learning rate.

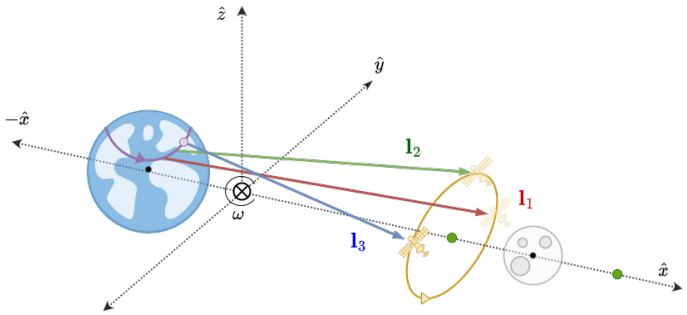


Fig. 1: Collecting line of sight data to perform Orbit Determination (OD) in a Circular Restricted Three Body (CR3B) frame: a telescope (purple) rotating with the surface of the Earth as the primary mass collects lines of sight ($\mathbf{l}_i$) of a satellite (orange) as it orbits about the L1 point.

1. INTRODUCTION

Determination of the position and velocity of a spacecraft via Orbit Determination (OD) is critical for tasks such as space traffic management and stationkeeping. Performing OD on near-earth objects is a well studied problem, with solutions varying from onboard Global Positioning System (GPS) data to ground-based observations [1]. Research has shown that Initial Orbit Determination (IOD) solutions such as Gauss' Method, Double-R Iteration, or Gooding's Method can be used to obtain satellite position and velocity from a small set of Electro-Optical (EO) line of sight measurements without *a priori* knowledge of the satellite's position using the collection scheme outlined in Figure 1 [1–4]. After performing IOD, Precise Orbit Determination (POD) is performed using sequential or batch estimation techniques to estimate the current state from the IOD state estimate [5]. Unfortunately, the methods of performing angles-only OD on near-earth objects do not extend to the cislunar domain [6, 7].

This is due to the fact that the techniques developed for near-Earth OD commonly rely on the two-body gravitational assumption for estimating satellite orbital parameters [8]. These traditional IOD methods solve for Keplerian orbital elements or the position and velocity that approximate the dynamical orbit of near-earth objects based on the underlying assumption that the trajectory is elliptical in nature [9]. These simplifying assumptions break for regions of cislunar space extending beyond Geostationary (GEO) orbit because the moon's gravitational effects become non-negligible, requiring three-body gravitational models to describe satellite trajectories [6, 10]. Under three-body dynamics orbits are not necessarily periodic, do not necessarily remain fixed to a plane, and are usually not elliptical [11].

The fundamental shift in assumptions that can be made requires novel techniques to be developed for cislunar OD. Consequently, optimizing observation networks [12–17] and techniques [7, 18, 19] for cislunar OD has become a critical focus in recent years along with the rise in Moon-bound traffic [20]. One solution that has recently been proposed is using Physics Informed Neural Networks (PINNs) to perform cislunar OD [21]. PINNs exploit the standard Machine Learning (ML) training technique known as automatic differentiation for the additional purpose of approximating partial derivatives of network outputs with respect to network inputs and model parameters [22]; this effectively allows PINNs to incorporate known differential equations of the physical problem under study into the training, reducing the resemblance of the problem to a "black-box" [23]. While PINNs have been shown to estimate high accuracy cislunar trajectories [19, 21], studies have not been performed on the stability and repeatability of these methods across the random initialized state from which they begin training.

Our results show that PINNs appear are plagued by similar convergence issues that affect traditional batch estimation algorithms. Spatial bounding constraints are known to affect the performance of traditional batch-estimation algorithms for cislunar IOD. As an example, Scorsoglio et al in a study on cislunar OD showed that if the initialization position or velocity error for batch estimation are greater than just 5%, that the algorithms can have $\geq$90% likelihood to not converge to the true trajectory [21]. In this study, we show that a lack of spatial constraints causes PINNs to have a similarly high potential to produce sub-optimal trajectory estimates. We demonstrate that this is due to the PINN becoming trapped across boundaries of high pseudopotential energy within the three-body system. We propose one potential solution for spatially constraining PINN trajectory estimates: transfer learning. Our results demonstrate

that transfer learning primes network biases and weights; this technique spatially constrains the PINN and allows it to converge on the true trajectory via gradient based optimization.

2. THEORY

2.1 Circular Restricted Three Body Problem (CR3BP) Dynamics

In this study, satellite motion under the gravitational influence of the Earth and Moon within the cislunar volume is modeled using Circular Restricted Three-Body Problem (CR3BP) dynamics. This topic is covered extensively in numerous publications such as [11] and [24]. It will therefore only be briefly described here for the sake of brevity.

Under CR3BP dynamics, the spacecraft's motion is due to gravitational influence of the Earth and Moon that are approximated as point masses [25]. In order to simplify equations, the spacecraft's state is expressed within a non-inertial (i.e. rotating) coordinate system that rotates with angular velocity of magnitude equal to the rate of rotation of the primary masses moving in circular orbits about their barycenter [24]. The time-scales, distances, and masses of the three bodies are adimensionalized to further simplify the equations of motion [24,25]. This results in the following equations for the non-maneuvering acceleration of the satellite within the Earth-Moon rotating frame as a function of it's non-inertial, adimensional position ($\mathbf{r} = [x, y, z]$) [24]:

$$\ddot{x} = -\frac{1-\mu}{{\rho_1}^3}(x+\mu) - \frac{\mu}{{\rho_2}^3}(x-1+\mu) + 2\dot{x} + \dot{x}\,, \tag{1}$$

$$\ddot{y} = -\frac{1-\mu}{{\rho_1}^3}y - \frac{\mu}{{\rho_2}^3}y - 2\dot{x} + \dot{y}\,, \tag{2}$$

$$\ddot{z} = -\frac{1-\mu}{{\rho_1}^3}z - \frac{\mu}{{\rho_2}^3}z\,, \tag{3}$$

where the mass ratio $\mu = m_2/(m_1+m_2) \approx 0.012277471$ is used to adimensionalize the *SI* masses of the Earth (m_1) and Moon (m_2); ρ_1 is the distance of the satellite from the Earth at fixed position ($\mathbf{R_1}$) such that $\rho_1 = |\mathbf{r} - \mathbf{R_1}|$; and ρ_2 is the distance of the satellite from the Moon at fixed position ($\mathbf{R_2}$) such that $\rho_2 = |\mathbf{r} - \mathbf{R_2}|$ [25].

The dimensional *SI* times are scaled by a characteristic time (t_{char}) that is defined by the equation: $t_{char} = \sqrt{\frac{|\mathbf{R}_{1_D} - \mathbf{R}_{2_D}|^3}{G(m_1+m_2)}}$, approximately equal to 4.348 days. [24, 25] In this equation, the Distance Unit (DU) is equal to the dimensional *SI* distance in between the primary masses, taking on a constant value of: $DU = |\mathbf{R}_{1_D} - \mathbf{R}_{2_D}| \approx$ 384,400 km. The adimensionalized times are used to integrate the adimensional acceleration of the satellite in Equations 1-3 using a Ordinary Differential Equation (ODE) solver that is described in previous studies [12] and [18].

2.2 Physics Informed Neural Networks (PINNs)

Physics Informed Neural Networks (PINNs) were recently introduced by Raissi et al [23] as a method for data-driven solution to and data-driven discovery of partial differential equations. The task of orbit determination falls under the former category, and we will therefore only focus on the solving properties of PINNs. Problems being solved under this data-driven solution framework take on the general form [23]:

$$f(t) = \mathbf{u}(t) + \mathscr{N}[\mathbf{u}(t)] = 0\ ,\ t \in [0, T] \tag{4}$$

where $\mathbf{u}(t)$ denotes the latent solution, and $\mathscr{N}[\cdot]$ is a nonlinear differential operator. The key concept of Raissi's approach is to utilize a deep neural network to predict the latent solution $\mathbf{u}(t)$ [23]. By predicting the latent solution using a neural network, the function $f(t)$ takes on the form of a *physics-informed neural network*. Under this framework, the weights and biases of the neural network are trained to minimize the following general Mean Squared Error (MSE) loss term that is a function of both physical constraints and measurement constraints:

$$L_{PINN} = MSE_u + \lambda\, MSE_f\ , \tag{5}$$

$$MSE_u = \frac{1}{N_u}\sum_{i_u=1}^{N_u}\left|\mathbf{u}(t_{i_u}) - \mathbf{u}_{i_u}\right|^2\ , \tag{6}$$

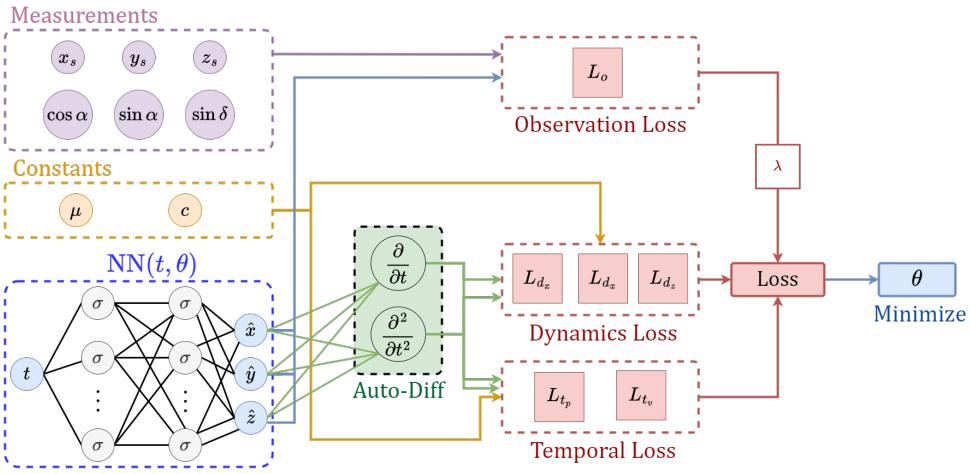


Fig. 2: The workflow used for physics informed machine learning orbit determination in this study.

$$MSE_f = \frac{1}{N_f} \sum_{i_f=1}^{N_f} \left| f(t_{i_f}) \right|^2, \tag{7}$$

where λ is a weighting factor, $[t_{i_u}, \mathbf{u}_{i_u}]_{i_u=1}^{N_u}$ denote the measured training data at discrete timesteps for the latent solution $\mathbf{u}(t)$, and $[t_{i_f}]_{i_f=1}^{N_f}$ denote timesteps where data is not observed but the PINN is required to satisfy physical constraints. In plain terms, the loss MSE_u fits the model to observed measurements while the loss MSE_f enforces the known constraints of the physical problem under study at timesteps in between collected measurements.

The deep neural network ($\mathbf{u}$) that predicts output as a function of time (t) employs automatic differentiation in the backpropagation algorithm that is used to train the network's weights [26]. Automatic differentiation is used to train deep learning models by taking derivatives of the output with respect to the weights and biases of the deep learning model. These derivatives are then used by an optimizer to update the weights of the model, thereby enabling ML systems to improve at model prediction across training epochs.

Automatic differentiation exploits the fact that every computer calculation can be represented as a sequence of primitive arithmetic operations (multiplication. addition, subtraction, etc.) or functions (sin, cos, exp, etc.). By applying the chain rule backwards from the output layer up to the input layer, partial derivatives of any order can be retrieved automatically to machine precision [22, 23]. PINNs use these gradients to approximate differential equations ($\mathcal{N}[\cdot]$) with respect to time and, optionally, spatial variables [26]. Therefore, the exact same automatic differentiation techniques that are employed to train powerful neural networks are also used by PINNs to constrain model predictions to known physics by using the approximated partial differential equations in the loss term MSE_f [23].

2.3 Orbit Determination (OD) using PINNs

It has been recently shown that the solving power of PINNs can be applied to orbit determination problems [19, 21]. The motivation for applying PINNs to OD problems came about after it was shown that PINNs can potentially solve Ordinary Differential Equations (ODEs) and systems of ODEs in which the only input to the network is time and the boundary conditions are not defined [27]. The general framework for the orbit determination PINN that is used in this study is outlined in Figure 2.

The neural network's output is the position of the cislunar space object ($\mathbf{u} = \mathbf{r} = [x, y, z]$) and the neural network input is the normalized time of the observation (discussed in Section B). To clarify, no spatial information is input to the neural network such that the output is solely a function of input time. This means that boundary conditions are not placed on the position of the satellite at any time over the observed trajectory. The importance of this lack of spatial bounding will become clear when training issues of PINNs are discussed in Section 4 and will be revisited there.

The OD problem seeks to estimate the satellite's position within the CR3BP frame using a line of sight from the

telescope's position to the satellite. The line of sight vector is denoted by right ascension (α) and declination (δ) [21]:

$$\alpha = \arctan\left(\frac{y-y_s}{x-x_s}\right) \; ; \quad \delta = \arctan\left(\frac{z-z_s}{\sqrt{(x-x_s)^2+(y-y_s)^2+(z-z_s)^2}}\right) \tag{8}$$

where $\mathbf{r_s} = [x_s, y_s, z_s]$ denotes the known Earth-fixed location of the telescope. In order to avoid numerical problems related to angles and to avoid computing arcsin and arctan terms, we utilize the sines and cosines of the angles rather than the angles themselves [19]. Therefore, the measurement line of sight vector is taken to be $\mathbf{l} = [\cos\alpha, \sin\alpha, \sin\delta]$.

To reiterate, the position of the satellite is never directly observed; only the line of sight along which the satellite lies is observed. Due to this abstraction, there is no direct observability of the satellite's position over the time of observation. The loss function for the measurements, MSE_u, consequently takes on the form:

$$MSE_u = \frac{1}{N_u}\sum_{i_u=1}^{N_u}\left|los\left(\hat{\mathbf{r}}(t_{i_u}); \mathbf{r_{si_u}}\right) - \mathbf{l}_{i_u}\right|^2, \tag{9}$$

where los is the line of sight transformation function of Equation 8 that accepts the predicted satellite position by the deep learning model ($\hat{\mathbf{r}}$) and the known telescope location ($\mathbf{r_{si_u}}$) at timestep t_{i_u}, and $\mathbf{l}_{i_u}$ is the measured line of sight at timestep t_{i_u}. A diagram of this line of sight concept was outlined in Figure 1. Note that for this exploratory study, we assume that there is no noise in the line of sight measurements.

In order to express the dynamical loss, we utilize the CR3BP dynamics that were introduced in Equations 1-3 of Section 2.1. Specifically, we use automatic differentiation to approximate the velocity ($\partial\hat{\mathbf{r}}/\partial t$) and acceleration ($\partial^2\hat{\mathbf{r}}/\partial t^2$) from the satellite position that is predicted by the neural network of the PINN. This results in the following loss terms that must be minimized to satisfy the laws of motion within the three body frame:

$$L_{d_x} = \frac{1}{N_f}\sum_{i}^{N_t}\left(\frac{\partial^2\hat{x}}{\partial t^2} - 2\frac{\partial\hat{y}}{\partial t} + \hat{x} + \left(\frac{1-\mu}{{\rho_1}^3}\right)(\hat{x}+\mu) + \left(\frac{\mu}{{\rho_2}^3}\right)(\hat{x}-1+\mu)\right)^2 \tag{10}$$

$$L_{d_y} = \frac{1}{N_f}\sum_{i}^{N_f}\left(\frac{\partial^2\hat{y}}{\partial t^2} + 2\frac{\partial\hat{x}}{\partial t} - \hat{y} + \hat{y}\left(\frac{1-\mu}{{\rho_1}^3} + \frac{\mu}{{\rho_2}^3}\right)\right)^2 \tag{11}$$

$$L_{d_z} = \frac{1}{N_f}\sum_{i}^{N_f}\left(\frac{\partial^2\hat{z}}{\partial t^2} + \hat{z}\left(\frac{1-\mu}{{\rho_1}^3} + \frac{\mu}{{\rho_2}^3}\right)\right)^2 \tag{12}$$

In the process of hypertuning our PINN, it was found that incorporation of loss terms to enforce smoothness of the trajectory across adjacent timesteps resulted in better OD estimates. Therefore, our PINN training included additional loss terms that enforced temporal continuity in velocity (L_{t_v}) and position (L_{t_p}) along the predicted trajectory:

$$\begin{aligned} L_{t_p} = \frac{1}{N_f-1}\sum_{i}^{N_f-1} &\left(\hat{x}_{i-1} + \Delta t\left.\frac{\partial\hat{x}}{\partial t}\right|_{i-1} + \frac{1}{2}\Delta t^2\left.\frac{\partial^2\hat{x}}{\partial t^2}\right|_{i-1} - \hat{x}_i\right)^2 + \\ &\left(\hat{y}_{i-1} + \Delta t\left.\frac{\partial\hat{y}}{\partial t}\right|_{i-1} + \frac{1}{2}\Delta t^2\left.\frac{\partial^2\hat{y}}{\partial t^2}\right|_{i-1} - \hat{y}_i\right)^2 + \\ &\left(\hat{z}_{i-1} + \Delta t\left.\frac{\partial\hat{z}}{\partial t}\right|_{i-1} + \frac{1}{2}\Delta t^2\left.\frac{\partial^2\hat{z}}{\partial t^2}\right|_{i-1} - \hat{z}_i\right)^2 \end{aligned} \tag{13}$$

$$L_{t_v} = \frac{1}{N_f-1}\sum_{i}^{N_f-1}\left(\left.\frac{\partial\hat{x}}{\partial t}\right|_i + \Delta t\left.\frac{\partial^2\hat{x}}{\partial t^2}\right|_i - \left.\frac{\partial\hat{x}}{\partial t}\right|_{i+1}\right)^2 + \left(\left.\frac{\partial\hat{y}}{\partial t}\right|_i + \Delta t\left.\frac{\partial^2\hat{x}}{\partial t^2}\right|_i - \left.\frac{\partial\hat{y}}{\partial t}\right|_{i+1}\right)^2 + \left(\left.\frac{\partial\hat{z}}{\partial t}\right|_i + \Delta t\left.\frac{\partial^2\hat{z}}{\partial t^2}\right|_i - \left.\frac{\partial\hat{z}}{\partial t}\right|_{i+1}\right)^2 \tag{14}$$

The loss function that was utilized in order to enforce physical constraints on the PINN at the N_f timesteps where no measurement data was available is therefore given by:

$$MSE_f = L_{d_x} + L_{d_y} + L_{d_z} + L_{t_p} + L_{t_v} \tag{15}$$

We note that this *MSE* function's constituent loss terms are shown in the code samples from Appendix Section A. This section shows how the automatic differentiation process is performed in practice, and how first- and second-order partial derivatives are computed with respect to time in the Tensorflow library.

3. METHOD: TRAINING OF PINN FROM RANDOMIZED INITIAL STATE

While Section 2 described the theory behind training of a PINN for orbit determination, the practical details must be discussed. This study leverages Tensorflow as the ML language [26], and training details will therefore be presented in the context of this ML library. As was discussed, the power of PINNs derives from the underlying ML library's ability to perform automatic differentiation of the predicted position with respect to the input time variable. In order to differentiate automatically, TensorFlow keeps track of all operations that led to an output prediction over the "forward pass" that is used to generate an output prediction. During the "backward pass" to update the hidden weights (defined in the literature as reverse-mode differentiation or, more specficially, backpropagation [22]), these operations are traced in reverse in order to approximate the gradient of the output with respect to the neural network's input.

The TensorFlow ML platform provides a `tf.GradientTape` API to trace operations from the input layer through the output layer while making a prediction; this API ultimately enables automatic differentiation of the first and second order with respect to time for this study [26]. The API "records" operations that are executed within the scope of a `tf.GradientTape` block onto a "tape" object. TensorFlow then traces this "tape" backwards in order to compute the gradients of a computation of interest via reverse-mode differentiation. To perform high-order differentation, gradients of first-order gradients can also be recorded if they are nested within a `tf.GradientTape`'s scope. To shed light on this process, we outline the algorithm that is used in the training of a PINN for orbit determination in Algorithm 1. We also provide demonstrations of the calculation of the loss functions of Equation 15 using the automatic differentiation process via the code in Appendix Section A.

There are three primary defined objects related to the machine learning components of the PINN in Algorithm 1. First, we define the neural network (`net`) that is taken to be a Multi-Layer Perceptron (MLP) for this study is a function of the number of layers (`layers`), the activation function (`activation`), and the number of units per layer (`units`). Second, we define the optimizer (`optim`) is a function not only of the type of optimizer (`type`) but also of the learning rate (`learning_rate`). Third, we define the number of epochs over which the system will be trained (`N_e`).

In terms of measurement information, the observed electro-optical observations and associated metadata are assumed known prior to beginning training: the timesteps of observation ($\mathbf{t_{obs}}$) taken to be of length N_u, the telescope position at those timesteps ($\mathbf{r_s}$) of size $N_u \times 3$, and the line of sight observed from the telescope to the satellite at the timesteps ($\mathbf{l_{obs}}$) of size $N_u \times 3$. The timesteps for which dynamics constraints will be imposed ($\mathbf{t_{dyn}}$) are taken to be of length $N_f = kN_u$ where k is an integer greater than one such that physics constraints are enforced at a higher cadence than the observation data.

Finally, there are three constants that are defined and fed to the training loop: the mass ratio of the Earth Moon rotating system (μ = `mu` = 0.01215058) [24], the weighting of the dynamics loss (λ), and a time normalization factor that must be applied to normalize the timesteps (`c`) that is elaborated on in Appendix Section B.

After all of these values are prepared, the training loop is executed in a very straightforward manner. For each epoch, the loss function operations are recorded by the `tf.GradientTape` object within it's scope (Line 10). The gradients of the total loss (Line 14) with respect to the neural network's training weights and biases are then recorded (Line 15). The optimizer then applies these gradients to update the network's weights using the optimizer (Line 16).

While additional ML training concepts such as patience and early stopping can be added, this was the basic loop carried out in this paper. We note that the reported results in Section 7 for an isolated PINN training run are derived from the trajectory prediction with the lowest overall training *MSE* across epochs, as defined by Equation 5.

The values used in this study for training of the PINN from a randomized initial state are outlined in Table 1. The values of N_f, N_u, `layers` were chosen based on previous studies on optimal values for PINN-based orbit determination by

Algorithm 1 Training a Physics Informed Neural Network (PINN) for Orbit Determination without transfer learning.

Defined
  Neural Network: `net(layers, activation, units)`
  Optimizer: `optim(learning_rate, type)`
  Number of Epochs: `N_e`
  Observations: t_{obs}, r_s, l_{obs},
  Dynamics Timesteps: t_{dyn}
  Constants : `mu`, `c`, λ
Training Loop
for epochs = 1 : `N_e` **do**
  with `tf.GradientTape()` recording the gradients as variable `tape`:
    Compute dynamics loss `L_d` = `dynamics_loss`(t_{dyn}, net, mu, c) using code in Section A
    Compute temporal continuity loss `L_t` = `temporal_loss`(t_{dyn}, net, c) using code in Section A
    Compute observational loss `L_u` = `observation_loss`(t_{obs}, net, r_s, l_{obs})
    Compute the total weighted loss `L_total` = `L_u` + λ (`L_t` + `L_d`)
    Compute gradients of `L_total` with respect to `net`'s trainable weights using `tape.gradient()`
    Apply gradients to `net` to update weights using the optimizer's routine `optim.apply_gradients()`
end for

Table 1: Parameters used to trained the PINN from a randomized initial state using Algorithm 1 in this study.

	Physics			**Machine Learning (ML)**					
Variable	N_u	N_f	λ	`learning_rate`	`type`	`units`	`activation`	`layers`	`N_e`
Value	200	$2*N_u = 400$	1E4	0.03	Adam [28]	64	`tanh` [29]	1	5000

Scorsoglio et al [19]. A brief hypertuning across learning rate and weighting factor (λ) was then performed on a subset of 10 cislunar trajectories. The other parameters were held fixed and were not tuned as hypertuning ML parameters was not the focus of the study, but rather studying the stability of PINNs for orbit determination.

As can be seen in Table 1, there are a broad number of both physics based and ML based parameters that must be tuned in order to obtain an optimal result for PINN-derived orbit determination. Optimally hypertuning across all of these parameters is highly impractical to perform operationally due to the time involved and, as we will show in the next Section, the potential to get trapped in sub-optimal OD solutions within a single randomized PINN OD run.

4. CHALLENGE: TRAINING INSTABILITY OF PINNS IN A RANDOMIZED INITIAL STATE

In Section 3, we outlined an approach for training PINNs for orbit determination. There are two complications with using this training approach to obtain operationally useful trajectories. Our experiments show that these two limitations can lead PINNs to predict sub-optimal trajectory estimates. In this Section, we isolate two training examples that demonstrate these pitfalls and the ways in which they can prevent successful retrieval of the true trajectory.

4.1 Issue #1: Overfitting of Dynamics Loss

The first pitfall that can occur when training is that the PINN can overfit to the dynamics loss term. Recall that the MSE loss function of Equation 5 requires definition of a constant (λ) to account for the relative balancing of the observation loss (MSE_u) and dynamics loss (MSE_f). An improperly tuned value of λ can lead the network to seek out trajectories with lower potential energy to the detriment of obtaining a trajectory that explains the line of sight measurements.

To demonstrate this, we trained a PINN using the approach of Section 3 to predict the trajectory of a satellite orbiting the Earth-Moon L1 equilibrium point in a Vertical orbit characterized by a period (P) of 6.298847 adimensional Time Units (TUs) and a stability index of 5.230350. The initial state vectors used to generate the orbit were obtained from the JPL Three-Body Periodic Orbit Catalog [30] and fed to a first-order Ordinary Differential Equation (ODE) solver for integrating satellite motion under CR3BP equations of motion [13]. We assumed that the satellite was observed continuously from Atlanta, GA over a time window equaling $0.5P$. The true trajectory of this satellite along the adimensional XYZ axes of the Earth-Moon rotating system are shown in the top row of Figure 3.

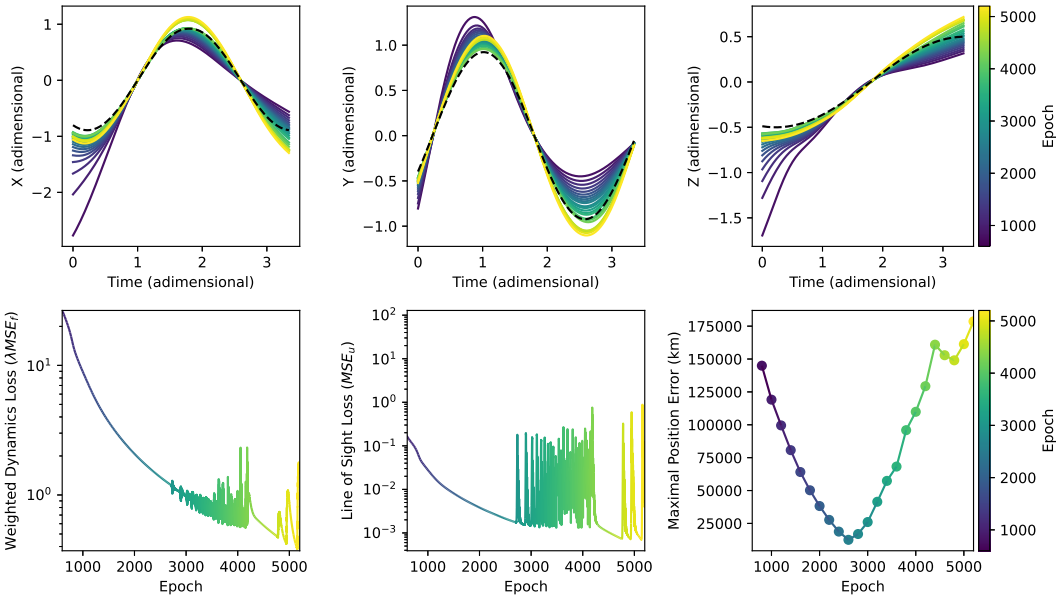


Fig. 3: PINN results using a randomized initial state on a cislunar satellite orbiting the Earth-Moon L1 equilibrium point in a Vertical orbit with P =6.298847 adimensional Time Units (TUs) and a stability index of 5.230350. The top row shows the adimensional trajectory estimates (colors) and the true trajectory (dashed black line). The bottom row shows the dynamics loss MSE_f (left) the observational loss MSE_u (middle) and the estimated positional error (right).

A PINN was allowed to train on the observational data for `N_e` = 5250 epochs using the parameters from Table 1 and randomly initialized weights and biases. Trajectory estimates at different epoch iterations are shown via a colormap in the top row of Figure 3. The scaled dynamics loss ($\lambda * MSE_f$) and the observational line of sight loss (MSE_u) were tracked and plotted in the left and middle figures of the bottom row, respectively. Additionally, the maximum position error along the estimated trajectory was plotted as a function of epoch in the righthand of the bottom row of Figure 3.

From the trajectory estimates, it can be inferred that training proceeds in a stable manner for the initial $\sim$ 2500 epochs; the observational loss and the dynamics loss both decrease in tandem suggesting appropriate choice of scale factor λ. After this period, however, there is a short duration of training stability with spiking of both the dynamics and line of sight loss terms. It can be seen that this instability ends near epoch 4200, after which the values of MSE_u and MSE_f are actually lower than at the start of the training instability.

Analysis of the error in estimated satellite position (lower right of Figure 3) reveals that overfitting of the network occurred as soon as the loss spiking began (epoch 2500). However, the truth trajectory position would not be available when operationally deploying the PINN; only the values of MSE_u and MSE_f would be available. While it could be argued that spiking of the MSE errors indicated overfitting, it would not be directly evident if the magnitude of the spiking was lower or the period of spiking was shorter.

An additional challenge with diagnosing overfitting to the dynamical loss in this scenario is that the dynamics (MSE_f) and line of sight (MSE_u) errors are *smaller* in magnitude after the training instability period ends. Analysis of the predicted trajectory in the top row of Figure 3 reveals that the predicted trajectory after `N_e` epochs is similar in shape to the true Vertical orbit, but that it extends farther into the XY direction of the adimensional Earth-Moon reference frame. This PINN solution is a plausible trajectory for explaining the observed line of sight, as evidenced by the relatively lower line of sight loss (bottom middle plot Figure 3) after epoch 4200. This example highlights that preventing overfitting of PINNs for orbit determination may not be as simple as employing techniques such as early stopping or patience if the PINN can predict spatially incorrect, but observationally plausible trajectories.

4.2 Issue #2: Trapping of PINN Trajectory Estimate by Steep Pseudopotential Boundaries

The second pitfall that can occur when training a PINN from a randomized initial state is that the PINN can become trapped into predicting a physically implausible trajectory estimate. As was discussed in Section 2.3, no spatial posi-

tion information is available to constrain the PINN for initial orbit determination. Boundary conditions are therefore not enforced on the satellite's position at any point over the observed trajectory. This lack of spatial bounding can lead the network's trajectory estimate to become trapped on one side of a high gravitational potential boundary from which it cannot escape without the dynamics loss (MSE_f) exploding in magnitude.

To demonstrate this, we trained a PINN using the approach of Section 3 to predict the trajectory of a satellite orbiting the Earth-Moon L1 equilibrium point in a Lyapunov orbit characterized by period P = 6.511973 TUs and a stability index of 53.675366. We once again assumed that the satellite was observed continuously from Atlanta, GA over a duration of $\sim 0.5P$. The true trajectory of this satellite along the adimensional XYZ axes of the Earth-Moon rotating system are shown in the top row of Figure 4. The PINN was allowed to train on the observational data for `N_e` = 8250 epochs using the parameters from Table 1 and randomly initialized weights and biases.

The trajectory predictions by the PINN shown in the top row of Figure 4 reveal that PINN initially predicts that the satellite is at an extreme distance of nearly $7\times$ the Earth-to-Moon distance along the $Z = 0$ plane. As the training progresses from epoch 1000 to epoch 4000, the PINN network progressively learns that the predicted trajectory should orbit about the L1 equilibrium point, as indicated by the trajectory predictions approaching the Moon and the line of sight loss (MSE_u) decreasing with increasing epoch.

Upon approaching the Moon's center of mass at ($X = 1-\mu$, $Y = 0$, $Z = 0$), the dynamics loss (MSE_f) begins to sharply spike. This can be explained by the steep change in pseudopotential (U) magnitude that comes from moving the satellite's trajectory closer to the Moon's center of mass. This pseudopotential is described by Equation 16 and plotted as a contour map in Figure 5 [24]:

$$U = (x^2+y^2)/2 + (1-\mu)/\rho_1 + \mu/\rho_2 \tag{16}$$

where ρ_1 and ρ_2 respectively denote the distance of a point in the Earth-Moon rotating frame from the primary (i.e. Earth) and secondary (i.e. Moon) bodies under CR3B dynamics [24].

Ultimately, the explosion of the dynamics loss MSE_f prevents the optimizer from moving the predicted satellite trajectory towards the the L1 point on the near side of the Moon; the predicted trajectory remains trapped on the far side of the Moon near the L2 point. The observational loss (MSE_u) and dynamical loss (MSE_f) levels after epoch 5500 indicate that the PINN has settled for a sub-optimal trajectory prediction and that the optimizer is unable to reach the optimal solution via gradient-based optimization with the current learning rate.

This PINN training example highlights that improper initialization of the PINN weights can cause the PINN's trajectory prediction to become trapped on the wrong side of high psuedopotential energy boundaries. An alternative representation of Figure 4 to prove this point is shown in Figure 5. In this Figure, the natural logarithm of the psuedopotential (U) is plotted as a blue contour map, with the true trajectory of the L1 Lyapunov satellite shown as a dashed black line. The predicted trajectories at select epochs are shown via a plasma colormap. It can be seen that the PINN's trajectory predictions become stalled on the ridge of the steep pseudopotential gradient on the far side of the Moon around epoch 4000.

5. SOLUTION: TRANSFER LEARNING

Section 4 shows that there are two primary challenges with training of PINNs from a randomized initial state. To overcome these challenges, we essentially want to prevent the model from under-fitting to the observation loss (MSE_u) and also prevent the model's trajectory predictions from becoming trapped on the wrong side of high psuedopotential boundaries. In order to correct these issues we propose a simple and intuitive method for priming the weights of the PINN before Algorithm 1 begins: transfer learning the ML model on a cislunar trajectory.

5.1 Algorithm

The motivation for this transfer learning method comes from recent research on classifying cislunar families using light curves. Researchers have shown that ML systems can be trained to retrieve the orbital family [9,18] and potentially the stability index [18] from electro-optical observations of cislunar satellites. From the classified cislunar family, a time series of positions ($\mathbf{r_{init}}$) and velocities ($\mathbf{v_{init}}$) can be generated by using a randomly phased initial state vector. In our proposed transfer learning approach, the initialization trajectory is then used to train the network to predict position and velocity according to Algorithm 2. As can be observed, the training loop is remarkably straightforward in nature.

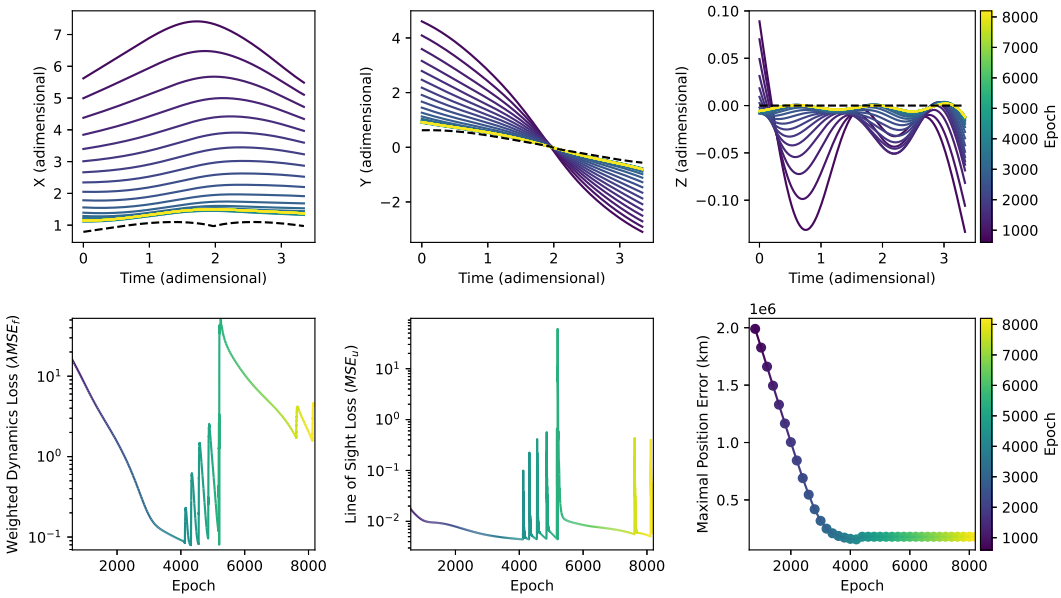


Fig. 4: PINN results using a randomized initial state on a cislunar satellite in a Lyapunov L1 orbit with P =6.511973 adimensional Time Units (TUs) and a stability index of 53.675366. The top row shows the adimensional trajectory estimates (colors) and the true trajectory (dashed black line). The bottom row shows the dynamics loss MSE_f (left) the observational loss MSE_u (middle), and the estimated positional error (right).

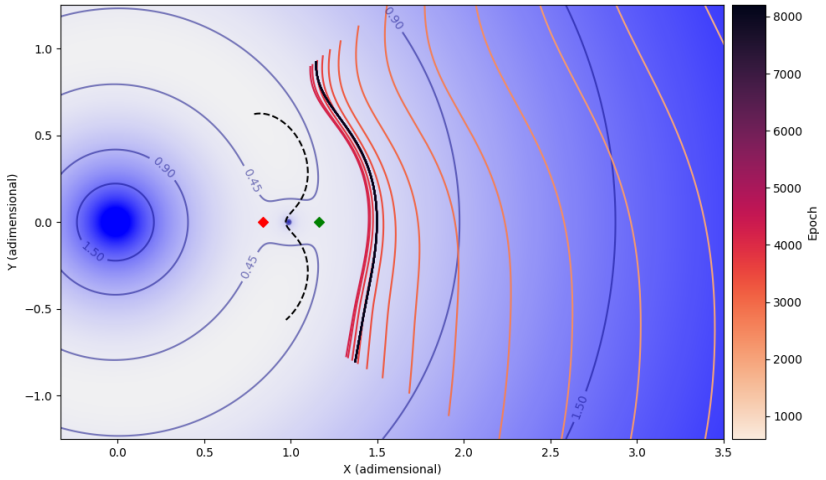


Fig. 5: Alternative form of the PINN predictions from Figure 4. The natural logarithm of the psuedopotential (U) is plotted as a blue contour map. The L1 and L2 points are plotted as red and green diamonds, respectively. The true Lyapunov L1 orbit is plotted via a black dashed line. The predicted trajectories by the PINN across training epoch are shown via the colormapped lines.

Algorithm 2 Transfer learning a Physics Informed Neural Network (PINN) to prime weights and biases.

Defined
  Neural Network: `net(layers, activation, units)`
  Optimizer for Transfer Learning: `optim_tl(learning_rate_tl, type_tl)`
  Number of Transfer Learning Epochs: `N_e_tl`
  Initialization Trajectory: $\mathtt{t_{init}}$, $\mathtt{p_{init}}$, $\mathtt{v_{init}}$,
  Constant : `c`
Training Loop
for epochs = 1 : `N_e_tl` **do**
  with `tf.GradientTape()` recording the gradients as variable `tape`:
    Compute transfer loss using code in Section A: `L = tl_loss(` $\mathtt{t_{init}}$, `net,` $\mathtt{p_{init}}$, $\mathtt{v_{init}}$, `c)`
    Compute gradients of `L` with respect to `net`'s trainable weights using `tape.gradient()`
    Apply gradients to `net` to update weights using the optimizer's routine `optim_tl.apply_gradients()`
end for

Table 2: Parameters used to transfer learn the PINN according to Algorithm 2.

Variable	`learning_rate_tl`	`type_tl`	`N_e_tl`
Value	0.03	Adam	150

The same network (`net`) that was used in Algorithm 1 is used in this precursor transfer learning stage. A separate optimizer (`optim_tl`) is used to train the network during this transfer learning stage. A differing, smaller number of training epochs (`N_e_tl`) is used in the transfer learning stage with the goal of mitigating overfitting. These parameters for this study are outlined in Table 2.

The new inputs to the transfer learning algorithm are the initialization timesteps ($\mathtt{t_{init}}$), the initialization positions along the trajectory ($\mathtt{p_{init}}$), and the initialization velocities along the trajectory ($\mathtt{v_{init}}$). Note that, in reality, the initialization timesteps are the same as the observation timesteps ($\mathtt{t_{obs}}$) given that we are attempting to fit to a trajectory that occurs over the same timescale as the observed lines of sight ($\mathtt{l_{obs}}$).

This initialization trajectory is used as input to a transfer learning loss function (`tl_loss`) that trains the neural network (`net`) to minimize the following transfer learning MSE loss (MSE_{tl}) in the transfer learning loss function (`tl_loss(·)`):

$$MSE_{tl} = \frac{1}{N_u}\sum_{i_u=1}^{N_u}\left(\left|\hat{\mathbf{r}}(t_{i_u}) - \mathbf{r}_{\mathbf{init}i_u}\right|^2 + \left|\frac{\partial\hat{\mathbf{r}}(\mathbf{t_{i_u}})}{\partial t} - \mathbf{v}_{\mathbf{init}i_u}\right|^2\right), \tag{17}$$

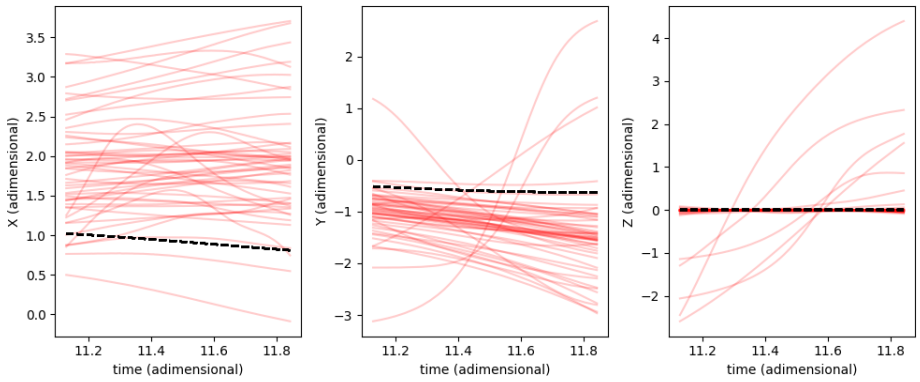
where we are computing the norms of the neural network predicted trajectory ($\hat{\mathbf{r}}$) and its derivatives ($\partial\hat{\mathbf{r}}/\partial t$) to the initialization positions ($\mathbf{r_{init}}$) and velocities ($\mathbf{v_{init}}$), respectively. This loss is computed over the same N_u timesteps (t_{i_u}) of our observed line of sight measurements.

After training the network via Algorithm 2, training proceeds according to the same loop as the original PINN training method in Algorithm 1. However, the network weights begin the training loop in a non-randomized state that predicts positions along the initialization trajectory. This can be thought of as a way of passively enforcing spatial constraints on the trajectory estimates, given that none are imposed when training via the approach of Algorithm 1.

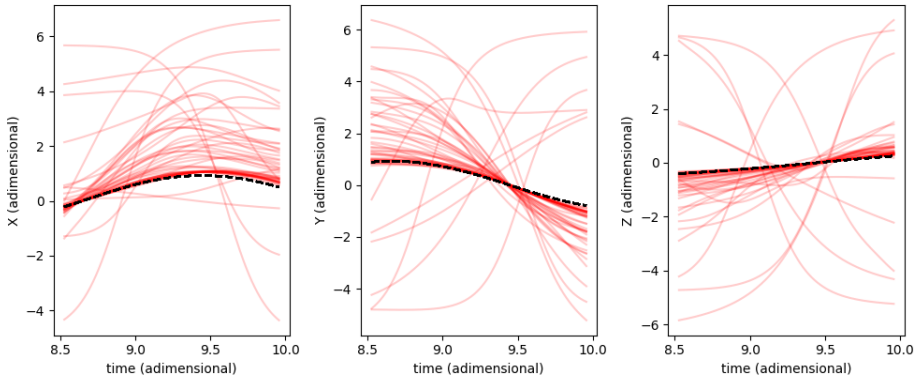
5.2 Qualitative Analysis of Transfer Learning

We believe that transfer learning approach provides a means for overcoming the two primary challenges that were addressed in the preceding Section. First, priming the network to predict a trajectory that explains the line of sight observations increases the potential for retrieving a trajectory estimate that explains the measurements. Second, initializing the network adjacent to the true equilibrium point potentially prevents it from becoming trapped on the wrong side of pseudopotential boundaries.

To qualitatively demonstrate the potential of transfer learning, we present example cases of training a PINN to predict the trajectory of satellites in two separate orbits: (a) a Lyapunov orbit about the L1 equilibrium point characterized by a period of P = 6.512 TUs and a stability index (v) of 53.675; (b) a Vertical orbit about the L1 equilibrium point



(a)



(b)

Fig. 6: PINN training results using a randomized initial state: (a) Lyapunov L1 orbit with P =6.512 adimensional Time Units (TUs) and a stability index (ν) of 53.675, (b) Vertical L1 orbit with P = 6.298 TU and ν = 9.090. The true trajectory is shown via a dashed black line and the results of 25 runs are shown in red lines.

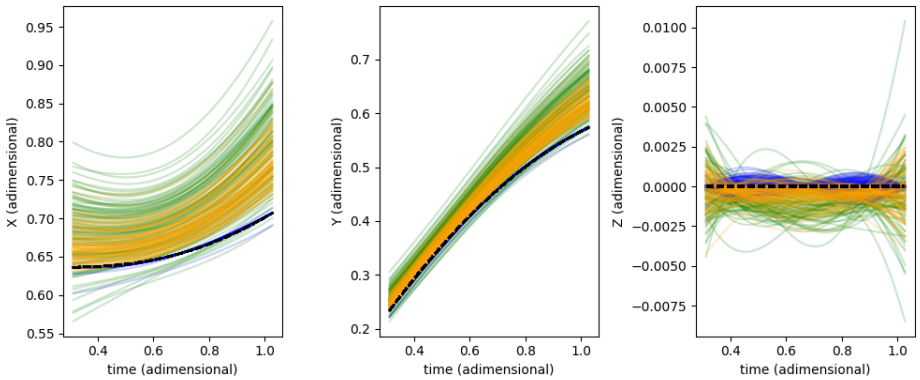
characterized by P = 6.298 TU and ν = 9.090.

In the first case, we performed 25 training runs of a PINN on a slice of the trajectory of duration $0.1P$ for 3000 epochs from a randomized initial state. Resulting trajectory estimates are shown via the red trajectory plots in Figure 6. It can be seen that the spread of trajectory predictions is quite broad, indicating that the PINN is likely encountering similar training issues to those described in Section 4. However, the PINN appears to have a higher success rate for the Vertical L1 orbit, suggesting that measuring lines of sight off the Earth-Moon (i.e. $Z = 0$) plane can increase the performance of PINNs in general.

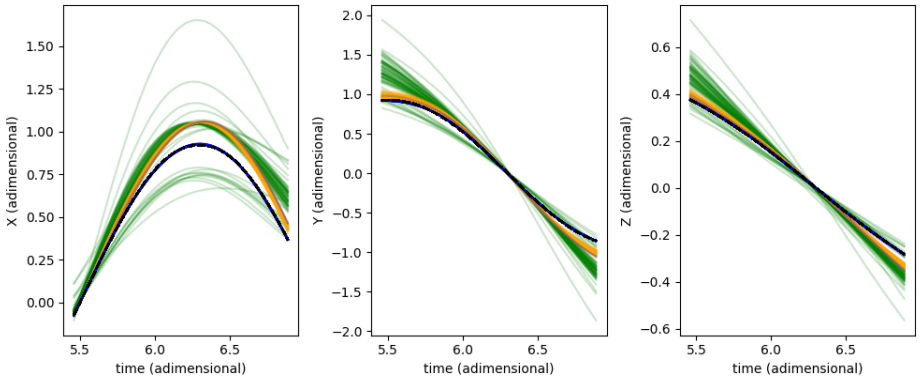
We then performed 25 training runs of a PINN on a slice of the trajectory of duration $0.1P$ for 3000 epochs on three different initialization cases for transfer learning.

In the first case, shown in blue, transfer learning was performed using initializaton trajectories from the same cislunar family and stability index as the truth trajectory. Note that these trajectories were randomly phased within the cislunar family's trajectory for each run, meaning that the initialization positions and velocities could be substantially different from the true positions and velocities. Despite this, the PINN is able to consistently settle on nearly the exact trajectory as the truth trajectory. This ability to overcome an incorrect initial state estimate suggests that priming a PINN's parameters to predict a trajectory with a similar pseudopotential to the truth can yield consistent and accurate results.

In the second case, shown in orange, we assumed that the correct cislunar family was predicted by the ML system but that the stability index was incorrectly predicted. From these results, we can see that the shape profiles of predicted trajectories nearly align with the true trajectory. These predicted trajectories would ultimately still be operationally useful for re-acquisition of a lost track due to their close proximity to the true trajectory.



(a)



(b)

Fig. 7: PINN training results using transfer learning for initialization: (a) Lyapunov L1 orbit with P =6.512 adimensional Time Units (TUs) and a stability index (ν) of 53.675, (b) Vertical L1 orbit with P = 6.298 TU and ν = 9.090. The true trajectory is shown via a dashed black line and the results of 25 runs for transfer learning on (1) the correct cislunar family and correct stability, (2) the correct cislunar family but incorrect stability, and (3) the incorrect cislunar family are shown in (1) blue, (2) orange, and (3) green, respectively.

In the final case, we simulated that the incorrect cislunar family was predicted by the ML classifier system, but that it was able to correctly predict that the satellite was orbiting the L1 equilibrium point. These results are shown in green in Figure 7. From these results, we can see that the spread of the predicted trajectories is broader, but that the trajectories often nearly match the true trajectory. This provides qualitative evidence that transfer learning can be useful even if it is only used to constrain the initial pseudopotential region that is predicted by the network.

We stress that the results produced in Figure 7 do not require knowledge of the true initial position and velocity. The initial position and velocity slices were randomized for each run, but the transfer learned PINN was able to yield consistent results across multiple optimizations. This provides evidence that simply initializing the trajectory estimates adjacent to the equilibrium point of the satellite's orbit can yield the needed spatial bounding that was outlined in Section 4. In the next section, we outline an experimental setup that was used to test this hypothesis.

6. EXPERIMENTAL SETUP

In order to test the hypothesis that transfer learned PINNs can yield higher accuracy and more repeatable orbit determination results than randomly initialized PINNs, we set up an experiment to test performance across cislunar orbits and initialization methods. We tested four different initialization cases, and eight different cislunar families for which orbit determination was being performed.

In the first initialization case, denoted as "Random Weights" in our Results, the weights and biases of the neural

Table 3: The transfer learning experimental cases. A truth trajectory was generated and a PINN was trained to perform transfer learning OD for three cases: (1) the family and stability index (v) was known, (2) the family was known but v was not, and (3) the family was not known. Units of period (P) are in TU and units of stability (v) are unitless.

		Initialization Trajectory		
Case	**Truth Trajectory**	**Same Family, Same Stability (v)**	**Same Family, Different Stability (v)**	**Different Family**
1	Axial L1 P=4.052 v=248.05	Axial L1 P=4.052 v=248.05	Axial L1 P=4.030 v=237.52	Lyap L1 P=7.102 v=62.53
2	DPO P=3.615 v=1.00	DPO P=3.615 v=1.00	DPO P=5.030 v=1.00	N Halo L1 P=2.416 v=1.42
3	Lyap L1 P=6.629 v=53.86	Lyap L1 P=6.629 v=53.86	Lyap L1 P=7.102 v=62.53	LPO P=3.501 v=42.30
4	Lyap L1 P=7.239 v=70.19	Lyap L1 P=7.239 v=70.19	Lyap L1 P=7.191 v=66.96	Axial L1 P=3.955 v=202.66
5	Lyap L2 P=7.133 v=50.60	Lyap L2 P=7.133 v=50.60	Lyap L2 P=7.445 v=53.52	DPO P=5.03 v=1.00
6	N Halo L1 P=2.071 v=2.31	N Halo L1 P=2.071 v=2.31	N Halo L1 P=1.857 v=2.79	Lyap L1 P=7.007 v=59.27
7	N Halo L2 P=1.788 v=1.69	N Halo L2 P=1.788 v=1.69	N Halo L2 P=2.416 v=1.42	Axial L2 P=4.34 v=140.27
8	Vertical L1 P=6.298 v=9.09	Vertical L1 P=6.298 v=9.09	Vertical L1 P=6.297 v=12.19	Lyap L1 P=7.102 v=62.53

network were randomized before training the neural network to perform orbit determination according to the approach outlined in Section 3. In this case, there was no transfer learning performed on the neural network to prime the weights, and the parameters in Table 1 were used to train the model.

In other three initialization cases, transfer learning on an initialization trajectory was performed using the approach outlined in Section 5.1 and the parameters outlined in Table 2. After performing transfer learning to prime the neural network parameters, training proceeded according to the approach of Section 3 using the parameters of Table 1.

There were three different methods for choosing an initialization trajectory. In the first method, denoted as "Same Family, Same Stability" in our results, we assumed that the orbital family, period, and stability were correctly predicted by an ML classification algorithm. In the second method, denoted as "Same Family, Different Stability" in our results, we assumed that the family was correctly predicted but that the orbital stability and period were incorrectly predicted. In the third method, denoted as "Different Family", we assumed that the family was incorrectly predicted by the ML classification system.

There were eight total cislunar orbits for which truth trajectories were drawn and then trained on according to one of these four neural network parameter initialization methods. The truth cislunar orbits are shown in Table 3, where it can be seen that we chose a spread of in-plane vs. out-of-plane families and also chose equilibrium points covering the region surrounding the Moon. Table 3 shows the initialization orbit that was used to generate randomly drawn initialization trajectories for the three transfer learning methods.

In total, there were 75 training runs performed for each of the 4 initialization methods across each of the 8 cislunar truth families. Additionally, we varied the duration of the orbit that was observed across three time windows values as a function of the period of the truth orbit: [0.1P, 0.2P, 0.4P]. Essentially, this means that orbit determination was performed on observed line of sight measurements spanning 10%, 20%, and 40% of the truth orbit's period, respectively. In total, (8 orbits $\times$ 4 initialization cases $\times$ 75 randomized runs $\times$ 3 temporal windows) = 7200 PINN training runs were performed across this series of experiments. The results of these training runs are quantitatively analyzed in the next Section in terms of the overall accuracy and repeatability of each of the tested neural network parameter initialization methods.

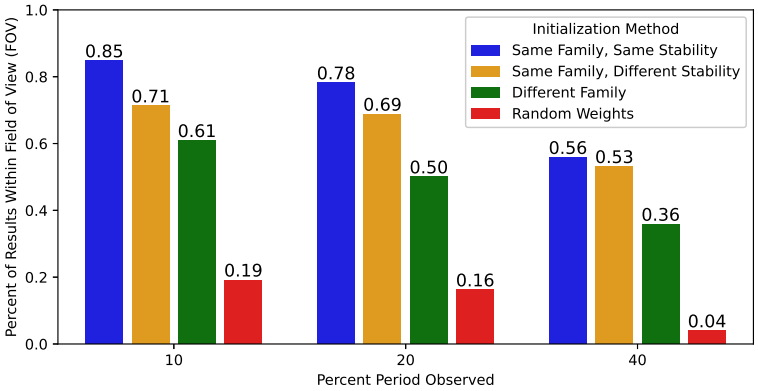


Fig. 8: The percentage of PINN line of sight estimates that remained within 0.5° of the true line of sight for the entire trajectory plotted as a function of percentage of the orbit's period observed and the initialization method for the PINN.

7. RESULTS

7.1 Line of Sight Performance of PINNs

When performing orbit determination the position of the spacecraft is unknown. For the case of Initial Orbit Determination (IOD) in particular, *a priori* knowledge of the spacecraft state is unavailable and must be retrieved in order to perform Precise Orbit Determination (POD) using follow on measurements [1, 3, 31]. Because the true position is unknown when performing orbit determination, positional accuracy cannot be used to evaluate the quality of an orbit determination estimate operationally. The ability of the PINN's estimated trajectory to model the measured line of sight measurements is therefore a useful operational metric. Research has shown that obtaining low line of sight errors in an IOD solution can lead to better performance for POD solutions [1, 32]. Furthermore, if the angular error of a trajectory estimate is lower than the Field of View (FOV) of a telescope, then there is a greater likelihood of recovering the object at a later time [19]. We therefore evaluated the line of sight errors of the PINN solutions across the percentage of the orbit's period that was observed and the initialization method employed.

In the first analysis, we studied the percentage of PINN solutions that remained within 0.5° of the true trajectory for the entire observed line of sight. This modeled a scenario of a telescope with a 0.5° FOV maintaining custody of cislunar objects. This metric is plotted in Figure 8. This plot shows that priming the PINN's parameters via transfer learning using an initialization trajectory of the same stability and period significantly increases the proportion of time that the retrieved trajectory explains the line of sight.

Interestingly, the success rates across all initialization methods are inversely proportional to the duration of the trajectory that was observed. When short arcs ($0.1P$) were used to train the PINN, transfer learning on trajectories with the same period and stability yielded a $\sim 4\times$ improvement over random initialization (0.85 vs. 0.19). When longer arcs ($0.4P$) were used to train the PINNs, the improvement grew to $\sim 14\times$ but the overall success rates of both methods dropped (0.56 vs. 0.04). **This suggests that the PINNs trained in this study performed better for shorter arcs, regardless of the initialization method used to prime the neural network parameters.**

Another interesting point drawn from Figure 8 is that using any initialization method, even if the cislunar family is unknown, increases the proportion of time that the PINN solution converges to a trajectory explaining the line of sight measurement by a factor of $3\times$ to $9\times$ as the length of the observed arc increases. **This suggests that PINN transfer learning can be used to explore the trajectory solutions that would be obtained by different cislunar families simply by priming the PINN parameters; transfer learning therefore potentially has an exploratory application as well as training regularization capabilities.**

We note that these results held as a general rule but did vary across the eight different cases considered in this study. To illustrate this, we show the MSE_u results for Case #2 (Distant Prograde Orbit (DPO), $P = 3.615$ TU, v=1.00) and Case #7 (Northern Halo L2, $P = 1.788$ TU, $v = 1.69$) in Figures 9 and 10, respectively. These box-and-whisker plots show the Interquartile Range (IQR) in the colored segment, with whiskers extending to $1.5\times$IQR and outliers shown

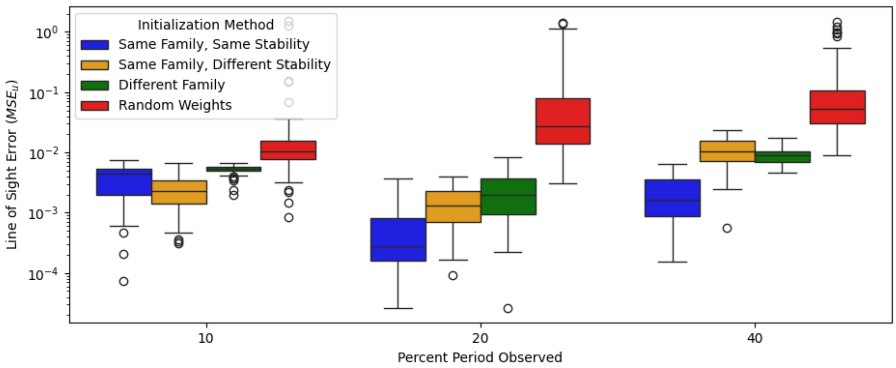


Fig. 9: Box-and-whisker plot of the final MSE_u values for Case #2 of a Distant Prograde Orbit (DPO), characterized by $P = 3.615$ TU and v=1.00. Whiskers extend to 1.5× the Interquartile Range (IQR).

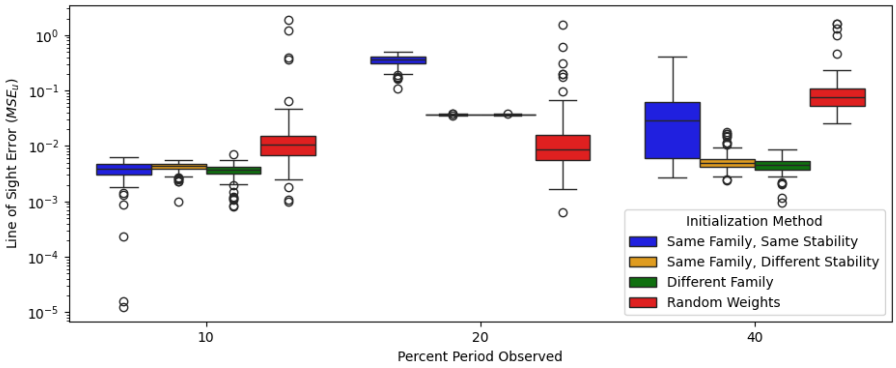


Fig. 10: Box-and-whisker plot of the final MSE_u values for Case #7 of a Northen Halo L2 orbit characterized by $P = 1.788$ TU and $v = 1.69$. Whiskers extend to 1.5× the Interquartile Range (IQR).

via unfilled circles.

For the DPO orbit of Case #2, the trends across initialization method and percent of the period observed follow the same general trends that are illustrated across all cases in Figure 8. However, the MSE_u results for Case #7 do not follow this trend across the percentage of the period observed. For example, when $0.2P$ is used to train the PINN, the random initialization method significantly outperforms all other initialization methods. This highlights that even when PINNs are regularized via transfer learning, there is still the potential to drift into sub-optimal trajectory solutions. **This suggests that there is a need for error monitoring to prevent both overfitting and the potential for drifting into sub-optimal trajectory solutions.**

7.2 Accuracy of PINN Estimated Trajectories

Ultimately, the most important metric for orbit determination is the ability to retrieve the true position of the satellite. While the true position is unknown when performing IOD operationally, it is known for the case of our simulated data. We therefore assessed the position accuracy of the retrieved solutions at the end of training using a MSE metric.

The results are shown in Figure 11 across the initialization method and percentage of the object's period observed. From this result, we can infer that initializing the PINN using a trajectory of the same period and stability that generated the line of sight measurements yields the best overall performance. When performing initialization using the "Same Family, Same Stability" method, the 25^{th} percentile of MSE errors are at error levels of $\sim$ 10 km, $\sim$ 100 km, and $\sim$ 1000 km across the respective observed arcs of $0.1P$, $0.2P$, and $0.4P$. For the "Random Weights" initialization, the 25^{th} percentile error levels jump to $\geq 1 \times 10^5$ km across all observed arc lengths. The alternative initialization

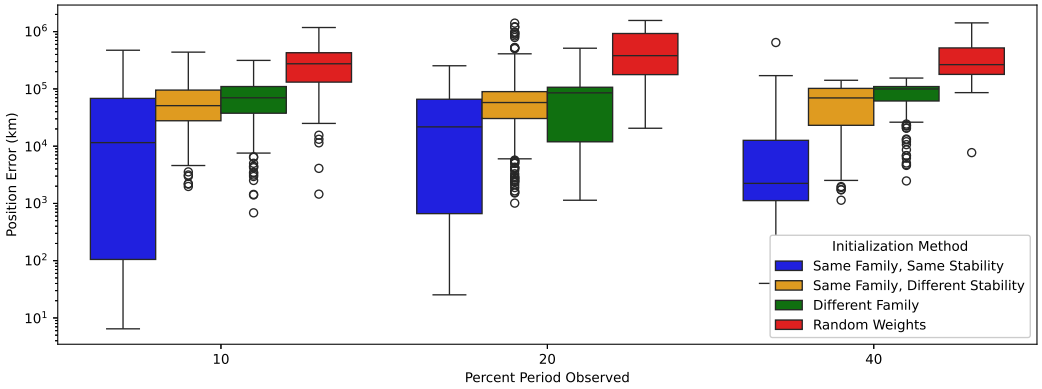


Fig. 11: Mean Squared Error (MSE) of the position at the end of training of the PINN plotted across the initialization method. Whiskers extend to 1.5× the Interquartile Range (IQR).

methods ("Same Family, Different Stability" and "Different Family") fall in between these two extremes, suggesting that initializing the PINN position estimates in a similar psuedopotential region to the true trajectory can regularize the PINN training process.

We believe that the results of Figure 11 indicate that overfitting and/or trapping of the PINN solutions in incorrect psuedopotential regions has occurred of the manner that was discussed in Section 4. We make the important caveat that the results do not suggest that random initialization of trajectory weights can never work. It is possible that a combination of ML techniques such as patience monitoring, hypertuning of the PINN machine learning parameters on a case by case basis, and spatial bounding can yield high accuracy position estimates. However, employing these sort of techniques was not the focus of this study. Rather, the focus was to highlight training issues that can arise with training PINNs without any spatial regularization. Understanding these training pitfalls is key to making PINN solutions repeatable and understanding when ML-derived orbit determination solutions should not be trusted.

We also evaluated whether there were any identifiable trends in MSE position accuracy across the cislunar families. The results of this exploration are shown in Figure 12. We find that there are several trends across cislunar family that shed light on potential PINN performance in operational contexts:

- The Distant Prograde Orbit (DPO) family's positional accuracy increases as the length of the observed arc increases, suggesting that increased observation time is required to obtain high accuracy PINN position estimates.
- The Northern Halo L2, Northen Halo L1, and Vertical L1 families often have the highest overall position uncertainties, likely due to the extreme distances of these objects from Earth and their trajectories entering gravitationally unstable out-of-plane regions that can cause instability of the dynamics loss function [24]. Adding more diverse observation geometry can likely improve these PINN results rather than simply using a single Earth-based telescope as was done in this study.
- The in-plane Lypunov L1 and Lyapunov L2 families generally have higher positional accuracy for shorter arcs. This suggests that feeding PINN's shorter arcs can increase the success rate of PINN convergence when the observed trajectory is within the XY plane of the Earth-Moon rotating frame.

To clarify these trends, we show the scatter plot of positional error broken into in-plane and out-of-plane components. In this context, the in-plane component denotes the error along the XY axes of the Earth-Moon rotating frame that contains the point masses of the Earth and Moon. The out-of-plane component is the Z axis of the Earth-Moon frame that is generally more gravitationally unstable due to it not containing the Earth and Moon. It can be seen from Figure 13 that "in-plane" families (i.e. Lyapunov, DPO) generally have higher out-of-plane error due to the lack of diversity of the line of sight measurements along this plane. The "out-of-plane" families (i.e. Halo, vertical) have lower overall in-plane error but have an overall error that is approximately proportional to the family's distance from the Earth-based telescope. We also see that the "Random Weights" initialization method rarely exceeds an overall

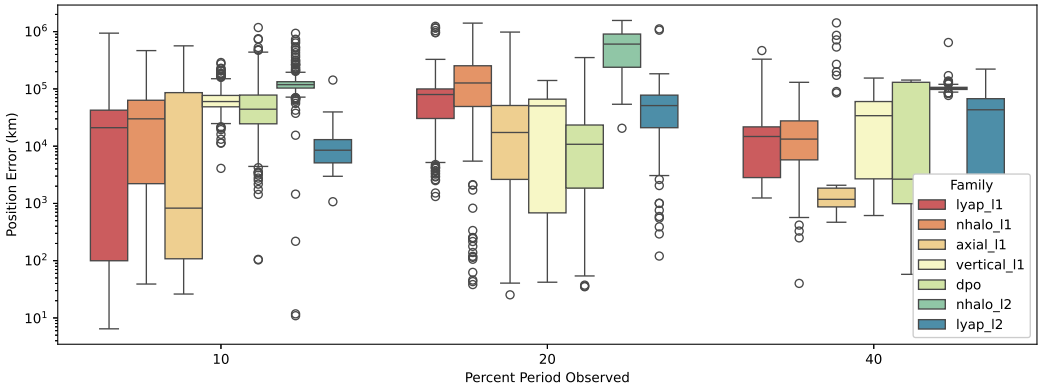


Fig. 12: Mean Squared Error (MSE) of the position at the end of training of the PINN plotted across orbital family. Whiskers extend to 1.5× the Interquartile Range (IQR).

positional accuracy of 10,000 km; this clarifies the critical needs for both spatial constraints when training PINNs and monitoring trajectory solutions across epoch to ensure that overfitting to the dynamics loss function does not occur.

8. DISCUSSION

In this Section, we highlight key insights from our experiments in using PINNs for orbit determination. Our goal with this study was to assess the stability and repeatability of physics informed machine learning methods for the task of cislunar orbit determination. In the process of carrying out these experiments, several useful lessons for operational deployment of PINNs for orbit determination emerged.

8.1 Sensitivity of PINNs to Initial Position and Velocity Error

Traditional OD methods that rely on line of sight observations of cislunar objects can fail to converge if the initial position and velocity error is too high. For example, Scorsoglio et al showed that if the initial state error for batch estimation was greater than 5%, then there is a $\geq$90% likelihood to not converge to an acceptable trajectory [21].

Our study shows that PINNs can overcome a poor initial state error if the trajectory is initialized in a close proximity pseudopotential region to the true trajectory. By performing transfer learning using a random position and velocity series from the same orbital family as the true trajectory, it was shown in Figure 8 that there was a $\geq 53\%$ likelihood to converge to a trajectory solution that explained the line of sight measurements. Furthermore, if the initialization trajectory was of the same family and stability as the true trajectory, then the mean positional error was shown to be $< 10,000$ km across all observational arcs in Figure 11.

Our results therefore suggest that while traditional IOD methods are sensitive to the initial position and velocity estimate, PINNs are sensitive primarily to the initial pseudopotential error. In other words, simply initializing the PINN's state estimate about the true equilibrium point can regularize the PINN and provide a higher likelihood to converge on the true trajectory solution.

8.2 Potential for PINNs to Overfit to Dynamics Error

Our study revealed that PINNs can overfit to the dynamics loss to the detriment of obtaining a trajectory solution that explains the line of sight measurements. This occurs if the relative weighting factor (λ) is not optimally tuned to properly weight the observation loss (MSE_u) and dynamics loss (MSE_f) in the training loss of Equation 5.

A clear example of this overfitting effect was seen for our training example in Figure 3 as discussed in Section 4.1. In this example, the PINN's positional error achieves a minimum error of $< 10,000$ km at epoch of approximately 2500. However, the model was allowed to continue training as the total MSE loss (Equation 5) was still decreasing. The model continues to train and ultimately achieves a lower overall dynamics and line of sight loss than at the epoch where the minimum position error was achieved.

Because true position knowledge is not available to the operator when performing IOD, the dynamics error (MSE_f) and the line of sight error (MSE_u) are the only available metrics for evaluating when overfitting occurs. Both our

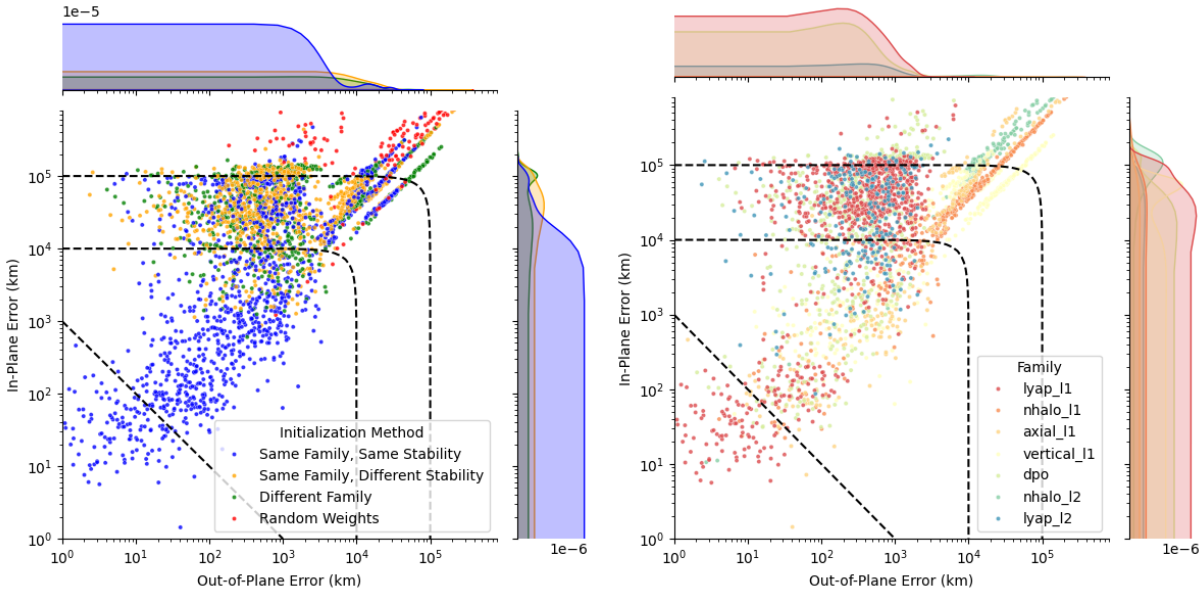


Fig. 13: Positional error along the out-of-plane direction (x-axis) and the in-plane direction (y-axis) of the Earth-Moon rotating frame. The left plot shows the initialization method and the right plot shows the orbital family.

isolated example in Figure 3 and the full set of PINN training results in Figure 11 suggest that PINNs can settle in sub-optimal orbital trajectories if the lowest overall *MSE* is used to determine the "optimal" trajectory from a PINN training run. This result shows that new techniques must be developed to monitor overfitting of PINNs.

Our results show that transfer learning can prevent a PINN from overfitting. As shown in Figures 11 and 13, the average position errors obtained using transfer learned networks are on average orders of magnitude lower than estimates obtained by networks trained from a random initial state. This suggests that initializing the PINN to predict a trajectory that is in close proximity to the true trajectory can allow the PINN's gradient optimization to progress towards the true trajectory solution rather than drifting into dynamically unstable solutions.

We add the important caveat that even when using transfer learning to spatially constrain the PINN trajectory estimate, the results in Figures 11 and 13 show that the PINN solution can produce sub-optimal trajectory estimates. This suggests that operators should always evaluate the solutions to ensure that they make logical sense and are in line with historical observations of the cislunar asset.

9. CONCLUSION

We explored the training stability of Physics Informed Neural Networks (PINNs) for cislunar orbit determination. It was found that PINNs that are initialized in a random state have a high probability to drift into sub-optimal trajectory solutions. We showed that transfer learning can be used to spatially regularize the PINN training process, allowing the PINN to produce trajectories with average positional errors that are two orders of magnitude lower than when training from a random initial state.

This research has numerous applications for cislunar orbit determination. We believe that the research has exploratory applications; the transfer learning concept can allow operators to explore trajectory solutions that produced a sequence of line measurements for a specific cislunar family. Transfer learning also opens up the door to utilizing novel light curve classification methods in orbit determination pipelines. For example, recent research has shown that neural networks can predict cislunar family from visual magnitude time series with high accuracy [9, 18]. If merged with a transfer learning PINN system, it is possible that fully Machine Learning based orbit determination systems can be realized.

10. ACKNOWLEDGEMENTS

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-23-1-0603. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

Initial work on the concepts and software used for simulations of cislunar trajectories was funded under Georgia Tech Research Institute (GTRI) Independent Research and Development (IRAD) funding.

Appendices

A. CODE FOR COMPUTING LOSS FUNCTIONS

```
def dynamics_loss(t, net, mu, c):
    """
    Args:
        t (tf.tensor): Normalized observation times
        net (tf.keras.Sequential): Neural network that accepts normalized time and predicts position
        mu (tf.tensor) : The spatial adimensionalization for CR3BPP dynamics.
        c (tf.tensor) : The time normalization factor for the observation window.

    Returns:
        loss (tf.tensor): Scalar loss function

    """
    one_minus_mu =tf.constant(1.0 -mu, dtype=tf.float32)

    # Use tensorflow GradientTape to record gradients for automatic differentiation
    with tf.GradientTape() as t2:
        t2.watch(t)
        with tf.GradientTape() as t1:
            t1.watch(t)
            r = net(t)
        dr_dt =tf.squeeze(t1.batch_jacobian(r, t))
    d2r_dt2 =tf.squeeze(t2.batch_jacobian(dr_dt, t))

    # Scale the velocity and acceleration to account for time normalization
    dr_dt =(c) *dr_dt
    d2r_dt2 =(c*c) *d2r_dt2

    # Dynamics loss across three spatial dimensions
    rho_1 =tf.sqrt((r[:,0] +mu)**2 +(r[:,1])**2 +(r[:,2])**2)
    rho_2 =tf.sqrt((r[:,0] -(one_minus_mu))**2 +(r[:,1])**2 +(r[:,2])**2)
    f_x =d2r_dt2[:,0] -2*dr_dt[:,1] -r[:,0] +(one_minus_mu/rho_1**3) *(r[:,0] +mu) +(mu/rho_2**3) *(r[:,0] -
                                                                    (one_minus_mu))
    f_y =d2r_dt2[:,1] +2*dr_dt[:,0] -r[:,1] +r[:,1]*((one_minus_mu/rho_1**3) +(mu/rho_2**3))
    f_z =d2r_dt2[:,2] +r[:,2]*((one_minus_mu/rho_1**3) +(mu /rho_2**3)))

    return tf.reduce_mean(f_x**2 +f_y**2 +f_z**2)
```

```
def temporal_loss(t, net, c):
    """
    Args:
        t (tf.tensor): Normalized observation times
        net (tf.keras.Sequential): Neural network that accepts normalized time and predicts position
        c (tf.tensor) : The time normalization factor for the observation window.

    Returns:
        loss (tf.tensor): Scalar loss function

    """
    # Use tensorflow GradientTape to record gradients for automatic differentiation
    with tf.GradientTape() as t2:
        t2.watch(t)
        with tf.GradientTape() as t1:
            t1.watch(t)
            r = net(t)
        dr_dt =tf.squeeze(t1.batch_jacobian(r, t))
    d2r_dt2 =tf.squeeze(t2.batch_jacobian(dr_dt, t))

    # Scale the velocity and acceleration to account for time normalization
    dr_dt =(c) *dr_dt
```

```
    d2r_dt2 =(c*c) *d2r_dt2

    # Undo normalization of time series
    dt = (t[1:] -t[:-1]) /c

    # Temporal loss accounting for position consistency
    loss_pos =tf.reduce_mean((r[:-1, 0] +dr_dt[:-1, 0] *dt +0.5 *d2r_dt2[:-1, 0] *dt**2 -r[1:,0])**2 +\
                             (r[:-1, 1] +dr_dt[:-1, 1] *dt +0.5 *d2r_dt2[:-1, 1] *dt**2 -r[1:,1])**2 +\
                             (r[:-1, 2] +dr_dt[:-1, 2] *dt +0.5 *d2r_dt2[:-1, 2] *dt**2 -r[1:,2])**2
                            )

    # Temporal loss accounting for velocity consistency
    loss_vel =tf.reduce_mean((dr_dt[:-1, 0] +d2r_dt2[:-1, 0] *dt -dr_dt[1:,0])**2 +\
                             (dr_dt[:-1, 1] +d2r_dt2[:-1, 1] *dt -dr_dt[1:,1])**2 +\
                             (dr_dt[:-1, 2] +d2r_dt2[:-1, 2] *dt -dr_dt[1:,2])**2
                            )

    return loss_pos +loss_vel
```

```
def tl_loss(t, net, p_init, v_init, c):
    """
    Loss function to train a network to fit the position and velocity of an initialization trajectory

    Args:
           t (tf.tensor): Normalized observation times
           net (tf.keras.Sequential): Neural network with input of normalized time and output of position
           p_init (tf.tensor): The positions of the satellite used to initialize the neural network
           v_init (tf.tensor): The velocities of the satellite used to initialize the neural network
           c (tf.tensor) : The time normalization factor for the observation window.

    Returns:
        loss (tf.tensor): Scalar loss function
    """
    with tf.GradientTape() as tape:
        tape.watch(t)
        position_pred =net(t)
    dr_dt =tf.squeeze(tape.batch_jacobian(position_pred, t))

    velocity_pred =c *dr_dt

    loss =tf.keras.losses.MeanSquaredError()(position_pred, p_init) +\
          tf.keras.losses.MeanSquaredError()(velocity_pred, v_init)

    return loss
```

B. TIME NORMALIZATION AND EFFECTS ON PARTIAL DERIVATIVES

Machine learning systems train more effectively when the input is normalized to be within the range of -1 to +1. In this study, the canonical time of the observation (t) is the only input to the network, and therefore must be normalized. In order to achieve this, we follow the approach of Scorsoglio et al to map the time sequence into the desired range [21]. We first define a normalization term, c that is a function of the initial time of the observed trajectory (t_0) and the final time of the observed trajectory (t_f):

$$c = 2/(t_f - t_0) \tag{18}$$

Using this term, we map time into a normalized time variable (t^*) and back according to the following transformations:

$$t^* = c(t - t_0) - 1\,; \quad t = (t^* + 1)/c + t_0 \tag{19}$$

Therefore, the normalized time (t^*) is input to the network and all partial derivatives computed by the automatic differentiation procedure are with respect to this variable. Because we wish to take derivatives with respect to canonical time, we must account for this scaling when computing dynamics loss terms. To understand this, consider the first and

second order partial derivatives of an arbitrary function (f) with respect to t^*:

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial t^*}\frac{\partial t^*}{\partial t} f = c\frac{\partial f}{\partial t^*}, \quad \frac{\partial^2 f}{\partial t^2} = \frac{\partial^2}{\partial (t^*)^2}\frac{\partial^2 t^*}{\partial t^2} f = c^2\frac{\partial^2 f}{\partial (t^*)^2} \tag{20}$$

From this, we see that the first and second order partial derivatives that are computed by the automatic differentiation procedure must be scaled by the factors c and c^2, respectively, in order to provide appropriately scaled velocity and accelerations [21]. An example of this in practice is shown for the code samples in Appendix A.

REFERENCES

[1] Andrew Vernon Schaeperkoetter. *A comprehensive comparison between angles-only initial orbit determination techniques*. PhD thesis, Texas A & M University, 2012.

[2] Reza Raymond Karimi and Daniele Mortari. Initial orbit determination using multiple observations. *Celestial Mechanics and Dynamical Astronomy*, 109:167–180, 2011.

[3] Kyle J DeMars, Moriba K Jah, and Paul W Schumacher. Initial orbit determination using short-arc angle and angle rate data. *IEEE Transactions on Aerospace and Electronic Systems*, 48(3):2628–2637, 2012.

[4] John A Christian and Courtney L Hollenberg. Initial orbit determination from three velocity vectors. *Journal of Guidance, Control, and Dynamics*, 42(4):894–899, 2019.

[5] Bob Schutz, Byron Tapley, and George H Born. *Statistical orbit determination*. Elsevier, 2004.

[6] Samuel Wishnek, Marcus J Holzinger, and Patrick Handley. Robust cislunar initial orbit determination. In *AMOS Conf. Proc*, 2021.

[7] Michael R Thompson, Nathan P Ré, Cameron Meek, and Bradley Cheetham. Cislunar orbit determination and tracking via simulated space-based measurements. 2021.

[8] David A Vallado. *Fundamentals of astrodynamics and applications*, volume 12. Springer Science & Business Media, 2001.

[9] Greg Martin, Charles J Wetterer, Jenna Lau, Jeremy Case, Nathan Toner, C Channing Chow, and Phan Dao. Cislunar periodic orbit family classification from astrometric and photometric observations using machine learning. In *2020 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2020.

[10] David Zuehlke, Taylor Yow, Daniel Posada, Joseph Nicolich, Christopher W Hays, Aryslan Malik, and Troy Henderson. Initial orbit determination for the cr3bp using particle swarm optimization. *arXiv preprint arXiv:2207.13175*, 2022.

[11] MJ Holzinger, CC Chow, and P Garretson. A primer on cislunar space. Technical report, Air Force Research Laboratory, 2021.

[12] Lois Visonneau, Yuri Shimane, and Koki Ho. Optimizing multi-spacecraft cislunar space domain awareness systems via hidden-genes genetic algorithm. *The Journal of the Astronautical Sciences*, 70(4):22, 2023.

[13] Gregory P Badura, Matthew Gilmartin, Yuri Shimane, Stef Crum, Lois Visonneau, Christopher R Valenta, Michael Steffens, Selcuk Cimtalay, Francis Humphrey, Mariel Borowitz, et al. Optimizing distributed space-based networks for cislunar space domain awareness in the context of operational cost metrics. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, page 70, 2023.

[14] Darin C Koblick and Joseph S Choi. Cislunar orbit determination benefits of moon-based sensors. In *The 23rd Advanced Maui Optical and Space Surveillance Technologies (AMOS) Conference*, 2022.

[15] Carolin Frueh, Kathleen Howell, Kyle DeMars, Surabhi Bhadauria, and M Gupta. Cislunar space traffic management: Surveillance through earth-moon resonance orbits. In *8th European Conference on Space Debris*, volume 8, 2021.

[16] Mark Bolden, Timothy Craychee, and Erin Griggs. An evaluation of observing constellation orbit stability, low signal-to-noise, and the too-short-arc challenges in the cislunar domain. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2020.

[17] Matthew L Gilmartin, Stef Crum, Jason Hodkin, Gregory Badura, Alaric Gregoire, Yuri Shimane, Lois Visonneau, Michael J Steffens, Selcuk Cimtalay, Francis Humphrey, et al. Optimization of lunar-based radar networks via a multi-disciplinary analysis and optimization (mdao) approach. In *AIAA SCITECH 2024 Forum*, page 1063, 2024.

[18] Gregory P. Badura, Dan DeBlasio, Aryzbe Najera, Ana C. Chavez-Lopez, Miguel Velez-Reyes, Nathan Un, Yuri Shimane, and Koki Ho. Identifying cislunar orbital families via machine learning on light curves. *The Journal of the Astronautical Sciences*, 2024 (In Review).

[19] Andrea Scorsoglio, Andrea D'Ambrosio, Luca Ghilardi, Roberto Furfaro, and Vishnu Reddy. Physics-informed orbit determination for cislunar space applications. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, page 1, 2023.

[20] Steve Creech, John Guidi, and Darcy Elburn. Artemis: an overview of nasa's activities to return humans to the moon. In *2022 ieee aerospace conference (aero)*, pages 1–7. IEEE, 2022.

[21] Andrea Scorsoglio, Luca Ghilardi, and Roberto Furfaro. A physic-informed neural network approach to orbit determination. *The Journal of the Astronautical Sciences*, 70(4):1–30, 2023.

[22] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[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] Erin E Fowler and Derek A Paley. Observability metrics for space-based cislunar domain awareness. *The Journal of the Astronautical Sciences*, 70(2):10, 2023.

[25] Wang Sang Koon, Martin W Lo, Jerrold E Marsden, and Shane D Ross. Dynamical systems, the three-body problem and space mission design. In *Equadiff 99: (In 2 Volumes)*, pages 1167–1181. World Scientific, 2000.

[26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[27] Enrico Schiassi, Roberto Furfaro, Carl Leake, Mario De Florio, Hunter Johnston, and Daniele Mortari. Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations. *Neurocomputing*, 457:334–356, 2021.

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

[29] Barry L Kalman and Stan C Kwasny. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581. IEEE, 1992.

[30] Mar Vaquero and Juan Senent. Poincaré: A multi-body, multi-system trajectory design tool. *7th International Conference on Astrodynamics Tools and Techniques*, 2018.

[31] Roberto Armellin, Pierluigi Di Lizia, and Renato Zanetti. Dealing with uncertainties in angles-only initial orbit determination. *Celestial mechanics and dynamical astronomy*, 125:435–450, 2016.

[32] Jean-Sébastien Ardaens and Gabriella Gaias. Angles-only relative orbit determination in low earth orbit. *Advances in Space Research*, 61(11):2740–2760, 2018.