Online Residual Dynamics Learning from Spacecraft Measurements

Giovanni Fereoli

***Abstract*—Employing Artificial Neural Networks (ANNs) to learn spacecraft dynamical disturbances enhances the precision and adaptability of the models on-board, thus granting the spacecraft greater autonomy. Conventional analytical models rooted in first-principles, often reliant on linearization and vulnerable to modeling errors, as well as neglecting complex disturbances due to limited knowledge, face inherent limitations. Looking ahead to future space missions involving celestial bodies such as asteroids, characterized by uncertain gravitational fields, or scenarios such as Entry, Descent, and Landing (EDL) with imprecise aerodynamic models, this paper advocates for a hybrid dynamical modeling approach. This strategy integrates first-principles models with an Online Supervised Training (OST) method to capture residual dynamics. Utilizing Extended Kalman Filter (EKF) estimates with Dynamical Model Compensation (DMC) generates features and labels for training, enabling the neural network to learn from spacecraft measurements and adapt to residual dynamics in real-time. The results of this study, bolstered by the differentiability of neural networks, promise enhanced performance in Guidance, Navigation, and Control (GNC) algorithms with unprecedented accuracy and robustness. Moreover, it expedites mission phases dedicated to refining dynamical models, thus aiding engineers and scientists in mission planning and execution.**

***Index Terms*—CR3BP, Kalman Filtering, Dynamical Model Compensation, Deep Learning, Dynamical Uncertainties, Disturbance Reconstruction.**

I. Introduction

SPACECRAFT guidance, navigation, and control are fundamental components in ensuring the success of space missions, necessitating precise dynamical models. Although traditional analytical models based on first principles perform well in certain scenarios, they frequently struggle to capture the complexities of intricate and non-linear dynamics, such as those found in Entry, Descent, and Landing (EDL) scenarios. Furthermore, they may struggle to account for uncertainties inherent in real-world space environments, such as navigating around unknown-shaped asteroids that affect the gravity field. This limitation has sparked an increasing interest in leveraging advanced learning techniques to improve the precision and effectiveness of dynamical modeling onboard aerospace systems.

Historically, researchers such as Leonard et al. [1] have employed Dynamical Model Compensation (DMC), leveraging exponentially time-correlated system noise processes like the first-order Gauss-Markov process to estimate and compensate for unmodeled accelerations, such as higher-order spherical harmonic gravity effects beyond J_3, within a filtering framework. Expanding on this notion, recent work by Bauersfeld et al. [2] introduced a hybrid approach integrating first-principles modeling with Artificial Neural Networks (ANNs) to improve guidance and control performance on high-performance maneuvering quadrotor platforms. A data-driven approach operating independently may encounter difficulties in reliably extrapolating to diverse flight scenarios. Consequently, the hybrid methodology emerges as a superior alternative, amalgamating and surpassing the capabilities of both first-principles theory and learned residual dynamics.

In spacecraft applications, gathering offline data for modeling purposes is often challenging due to its scarcity and complexity. Therefore, the development of an online framework capable of estimating and learning dynamical disturbance models in real-time from spacecraft measurements is highly desirable. In this regard, a synthesis of ideas from previous studies emerges. For example, Martin and Schaub [3] incorporated DMC into an Extended Kalman Filter (EKF) framework to provide features and labels for a Physics-Informed Neural Network (PINN) training, enhancing the performance of the navigation algorithm around asteroids. Similarly, Park and D'Amico [4] proposed an Online Supervised Training (OST) method within an Unscented Kalman Filter (UKF) to refine the optical navigation measurement models onboard during Rendezvous and Proximity Operations (RPO) using incoming flight images.

A key highlight of these approaches lies in their incremental online ANN learning, obviating the need for prior knowledge or pre-training. This characteristic significantly enhances the flexibility of the approach while reinforcing, rather than replacing, the Guidance, Navigation, and Control (GNC) system, thereby augmenting its robustness. Other approaches, like those discussed by Harl et al. [5], diverge from DMC by employing observer-based techniques and Lyapunov stability theorems to ensure estimation convergence and offer uncertainty outputs for specific states through ANNs, albeit with a different emphasis compared to acceleration modeling emphasized in this investigation. However, DMC retains its importance in this study due to its proven effectiveness in tackling similar challenges.

II. Theoretical Background

A. Dynamical Model

The Circular Restricted Three-Body Problem (CRTBP) models the trajectory of a spacecraft, considered to have negligible mass, under the gravitational influence of Earth and Moon, which have masses M_1 and M_2, respectively. These bodies orbit their common center of mass in circular paths. The

Graduate Research Assistant, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. AAS Member, AIAA Member.

motion is analyzed in a rotating reference frame centered at this barycenter with axes $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$, where $\hat{\mathbf{x}}$ points from Earth towards the Moon, $\hat{\mathbf{z}}$ aligns with the orbital angular momentum of the system, and $\hat{\mathbf{y}}$ forms a right-handed coordinate system (Figure 1).

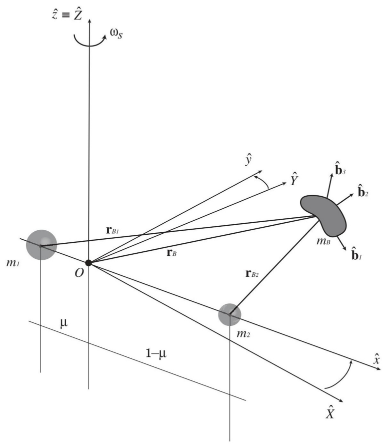

Fig. 1. Circular Restricted Three-Body Problem (CRTBP) synodic reference frame. Source: Colagrossi et al. [6].

Dimensionless variables are used, scaled by characteristic length (l^*), mass (m^*), and time (t^*) units, where l^* is the fixed distance between Earth and Moon, m^* is the combined mass of Earth and Moon, and t^* sets the non-dimensional orbital period at 2π. The spacecraft state in this frame is expressed as $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$, leading to the corresponding equations of motion:

$$\begin{aligned} \ddot{x} &= 2\dot{y} + x - (1-\mu)\frac{x+\mu}{r_1^3} - \mu\frac{x+\mu-1}{r_2^3} \\ \ddot{y} &= -2\dot{x} + y - (1-\mu)\frac{y}{r_1^3} - \mu\frac{y}{r_2^3} \\ \ddot{z} &= (1-\mu)\frac{z}{r_1^3} - \mu\frac{z}{r_2^3} \end{aligned} \tag{1}$$

$$\mathbf{r}_1 = [x+\mu,\ y,\ z] \tag{2}$$

$$\mathbf{r}_2 = [x+\mu-1,\ y,\ z] \tag{3}$$

The vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ represent the locations of the spacecraft P_3 relative to the primary bodies P_1 and P_2. The mass ratio $\mu = M_2/(M_2+M_2)$ governs the dynamics completely and, for the Earth-Moon system, μ is 0.012150584269542. The CRTBP State Transition Matrix (STM), denoted $\mathbf{\Phi}$, embodies the derivative of the state vector $\mathbf{x}$ with respect to its initial conditions $\mathbf{x}_0$, formulated as $\mathbf{\Phi}(t,\ t_0) = \partial\mathbf{x}(t)/\partial\mathbf{x}(t_0)$, and evolves according to $\dot{\mathbf{\Phi}}(t,\ t_0) = \mathbf{A}(t)\,\mathbf{\Phi}_i(t,\ t_0)$. Here, $\mathbf{A}(t) = \partial\mathbf{f}(\mathbf{x})/\partial\mathbf{x}(t)$ (Equation 4) signifies the Jacobian matrix of the CRTBP equations of motion, obtained using variational equations (Equations 5). The initial conditions of the STM are consistently established as $\mathbf{\Phi}(t_0,\ t_0) = \mathbf{I}_{6x6}$.

$$\mathbf{A}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\partial\dot{x}}{\partial x} & \frac{\partial\dot{x}}{\partial y} & \frac{\partial\dot{x}}{\partial z} & 0 & 2 & 0 \\ \frac{\partial\dot{y}}{\partial x} & \frac{\partial\dot{y}}{\partial y} & \frac{\partial\dot{y}}{\partial z} & -2 & 0 & 0 \\ \frac{\partial\dot{z}}{\partial x} & \frac{\partial\dot{z}}{\partial y} & \frac{\partial\dot{z}}{\partial z} & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$\begin{cases} \frac{\partial\dot{x}}{\partial x} = 1 - \frac{1-\mu}{r_1^3} - \frac{3(1-\mu)(x+\mu)^2}{r_1^5} - \frac{\mu}{r_2^3} + \frac{3\mu(x+\mu-1)^2}{r_2^5} \\ \frac{\partial\dot{x}}{\partial y} = \frac{\partial\dot{y}}{\partial x} = \frac{3(1-\mu)y(x+\mu)}{r_1^5} + \frac{3\mu y(x+\mu-1)}{r_2^5} \\ \frac{\partial\dot{x}}{\partial z} = \frac{\partial\dot{z}}{\partial x} = \frac{3(1-\mu)z(x+\mu)}{r_1^5} + \frac{3\mu z(x+\mu-1)}{r_2^5} \\ \frac{\partial\dot{y}}{\partial y} = 1 - \frac{1-\mu}{r_1^3} - \frac{3(1-\mu)y^2}{r_1^5} - \frac{\mu}{r_2^3} + \frac{3\mu y^2}{r_2^5} \\ \frac{\partial\dot{y}}{\partial z} = \frac{\partial\dot{z}}{\partial y} = \frac{(1-\mu)yz}{r_1^5} + \frac{3\mu yz}{r_2^5} \\ \frac{\partial\dot{z}}{\partial z} = -\frac{1-\mu}{r_1^3} - \frac{3(1-\mu)z^2}{r_1^5} - \frac{\mu}{r_2^3} + \frac{3\mu z^2}{r_2^5} \end{cases} \tag{5}$$

When considering the Earth-Moon system, two disturbances can be taken into account to improve the accuracy of the model without making it overly complex with ephemeris: Solar Radiation Pressure (SRP) and the gravitational effect of the Sun as a fourth body. To model the first, the spacecraft can be thought of as a perfect reflective sphere with uniform optical properties, and the SRP can be seen as a continuous force opposing the sunlight hitting it (Figure 2). This is known as the cannonball assumption [7], a first-order approximation used for the LAGEOS mission, and is implemented as follows:

$$\mathbf{a}_{SRP} = -C_r A\frac{P_{SRP}}{m}\left(\frac{1\ AU}{r_4}\right)^2\frac{\mathbf{r}_4}{r_4} \tag{6}$$

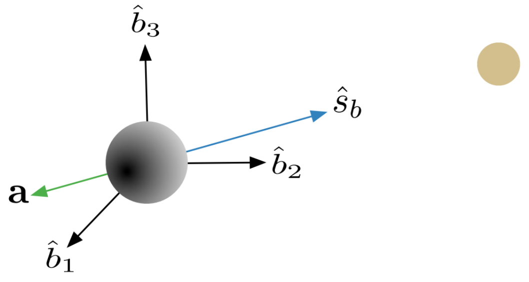

Fig. 2. Solar Radiation Pressure (SRP) disturbance modeling through the Cannonball assumptions. Source: Kenneally [7].

Where C_r is the reflectivity coefficient of the spacecraft, $P_{SRP} = 4.56 \cdot 10^{-6}\ Pa$ is the pressure of the photons that impinge on the cross section A of the spacecraft, and $\mathbf{r}_4$ (Equation 7) is the position with respect to the Sun. Instead, the gravitational effect of a fourth body, such as the Sun, can be computed considering the Bicircular Restricted Four-Body Problem (BRFBP) [8]. The Sun and the barycenter of the Earth-Moon system O move in a circular coplanar Keplerian orbit around their shared barycenter $\hat{O}$. This four-body problem is not consistent, meaning that the motion of Earth and Moon is not affected by the Sun's gravity. The model can be formulated as follows:

$$\mathbf{r}_4 = \left[x - r_s cos(\theta_s t),\ y - r_s sin(\theta_s t),\ z\right] \tag{7}$$

$$\Omega_{4B}(\mathbf{x},\ t) = \frac{m_s}{r_4} - \frac{m_s}{r_s^2}\left[x cos(\theta_s t) + y sin(\theta_s t)\right] \tag{8}$$

$$\mathbf{a}_{4B} = \mathbf{\nabla}\Omega_{4B}(\mathbf{x},\ t) \tag{9}$$

Where Ω_{4B} is the BCRFBP pseudo-potential in the Earth-Moon synodic frame, m_s is the mass of the Sun, w_s is the angular velocity and r_s is the radius of the circular orbit reflecting the Sun-O motion; thus, the current location of the Sun is $\left(r_s cos\left(\theta_s t\right),\ r_s sin\left(\theta_s t\right),\ 0\right)$. It is important to note that the Sun is not stationary in the chosen reference frame, resulting in a loss of autonomy. This is a great departure from the CRTBP as both the equilibrium points and the Jacobi integral vanish.

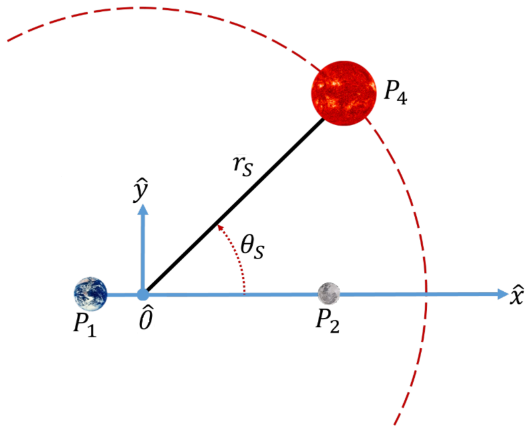

Fig. 3. Bicircular Restricted Four-Body Problem (BRFBP) coplanar motion of the Sun around $\hat{O}$ with respect to the Earth-Moon synodic frame. Source: Mccarthy and Howell [8].

B. Measurement Model

To properly establish an estimation problem, it's essential to implement a measurement model represented by $\mathbf{y} = \mathbf{h}\left(\mathbf{x}\right)$. In the context of spacecraft orbit determination, one of the most common measurements involves determining the distance between an Earth-based sensor and the satellite itself. Given the relative position vector $\delta\mathbf{r} = \mathbf{r} - \mathbf{r_s}$ between the spacecraft and the sensor, the ideal range, also known as geometric or instantaneous range, is computed as follows:

$$\rho^i = \|\delta\mathbf{r}\| \tag{10}$$

Noise, $\eta_\rho \sim \mathcal{N}\left(0, \sigma_\rho\right)$, is typically added to this ideal measurement as follows:

$$\rho = \rho^i + \eta_\rho \tag{11}$$

The partials of this noisy model are in relation to a spacecraft state $\mathbf{x} = \left[\mathbf{r}, \mathbf{v}\right]^T$ as follows:

$$\frac{\partial\rho}{\partial\mathbf{x}} = \left[\frac{\delta\mathbf{r}}{\rho^i},\ \mathbf{0}_{1x3}\right] \tag{12}$$

In some cases, the rate's time of change, also known as the range-rate, can be measured. Given the relative velocity vector $\delta\mathbf{v} = \mathbf{v} - \mathbf{v_s}$, the ideal range-rate is calculated as:

$$\dot{\rho}^i = \frac{\delta\mathbf{r}\cdot\delta\mathbf{v}}{\rho^i} \tag{13}$$

As $\eta_{\dot{\rho}} \sim \mathcal{N}\left(0, \sigma_{\dot{\rho}}\right)$, noise is added to this ideal measurement:

$$\dot{\rho} = \dot{\rho}^i + \eta_{\dot{\rho}} \tag{14}$$

The partials of this noisy model are in relation to a spacecraft state $\mathbf{x} = \left[\mathbf{r}, \mathbf{v}\right]^T$ as follows:

$$\frac{\partial\dot{\rho}}{\partial\mathbf{x}} = \left[\frac{\delta\mathbf{v}}{\rho^i} - \delta\mathbf{r}\frac{\dot{\rho}^i}{\left(\rho^i\right)^2},\ \frac{\delta\mathbf{r}}{\rho^i}\right] \tag{15}$$

Overall, for a ground station that can provide both types of measurements, a filter would obtain a measurement vector as $\mathbf{y}_k = \left[\rho_k,\ \dot{\rho}_k\right]$. and a measurement model Jacobian matrix, as shown below:

$$\mathbf{H}_k = \begin{bmatrix} \frac{\delta\mathbf{r}}{\rho^i} & \mathbf{0}_{1x3} \\ \frac{\delta\mathbf{v}}{\rho^i} - \frac{\delta\mathbf{r}\dot{\rho}^i}{(\rho^i)^2} & \frac{\delta\mathbf{r}}{\rho^i} \end{bmatrix} \tag{16}$$

In general, a filter with a state $\mathbf{x} \in \mathbb{R}^n$ and a measurement vector $\mathbf{y} \in \mathbb{R}^m$, the matrix is $\mathbf{H} \in \mathbb{R}^{mxn}$.

C. Extended Kalman Filtering

Kalman filters are recursive estimation algorithms employed to estimate spacecraft state and environmental parameters based on uncertain measurements [9]. The Extended Kalman Filter (EKF) represents a non-linear version of these sequential filters. Unlike the Linearized Kalman Filter (LKF), the EKF does not require linearization of dynamical and measurement models. The general framework for state and parameter estimation in an EKF is as follows:

1) Initialize the filter with the initial state estimate $\hat{\mathbf{x}}_0$, the initial state covariance $\mathbf{P}_0$, the process noise covariance matrix $\mathbf{Q}$, and the measurement noise covariance matrix $\mathbf{R}$;
2) Propagate from t_{k-1} to t_k by utilizing equations of motion (which incorporate the State Transition Matrix) with the initial conditions $\mathbf{x}(t_{k-1}) = \hat{\mathbf{x}}^+_{k-1}$ and $\mathbf{\Phi}(t_{k-1}, t_{k-1}) = \mathbf{I}$. This step yields $\mathbf{x}^*_k$ and $\mathbf{\Phi}(t_k, t_{k-1})$;
3) Perform the prediction step at time step t_k to calculate the a-priori state estimate $\hat{\mathbf{x}}^-_k$ and the a-priori state covariance $\mathbf{P}^-_k$ as follows:

$$\begin{cases} \hat{\mathbf{x}}^-_k = \mathbf{x}^*_k \\ \mathbf{P}^-_k = \mathbf{\Phi}\left(t_k,\ t_{k-1}\right)\mathbf{P}^+_{k-1}\mathbf{\Phi}\left(t_k,\ t_{k-1}\right)^T + \\ \quad \mathbf{Q}\left(t_k,\ t_{k-1}\right) \end{cases} \tag{17}$$

4) Using the knowledge of the measurement model, predict measurements and compute their partials in t_k using a-priori information as:

$$\begin{cases} \hat{\mathbf{y}}^-_k = \mathbf{h}\left(\hat{\mathbf{x}}^-_k,\ t_k\right) \\ \mathbf{H}_k = \frac{\partial\mathbf{h}\left(\mathbf{x},\ t\right)}{\partial\mathbf{x}}\Big|_{\hat{\mathbf{x}}^-_k,\ t_k} \end{cases} \tag{18}$$

5) Obtain the actual measurement $\mathbf{y}_k$, calculate the Kalman gain, and execute the correction step to obtain the a-posteriori state estimate and a-posteriori state covariance as:

$$\begin{cases} \mathbf{K}_k = \mathbf{P}^-_k\mathbf{H}^T_k\left(\mathbf{H}_k\mathbf{P}^-_k\mathbf{H}^T_k + \mathbf{R}\right)^{-1} \\ \hat{\mathbf{x}}^+_k = \hat{\mathbf{x}}^-_k + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}^-_k\right) \\ \mathbf{P}^+_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}^T_k\right)\mathbf{P}^-_k \end{cases} \tag{19}$$

6) Repeat for each incoming measurement.

D. Dynamical Model Compensation

Besides the influence of non-linearities, errors in the dynamical model can also induce divergence in the estimation process. To address these errors during filtering, one prevalent technique is State Noise Compensation (SNC), which entails approximating the error through process noise. A more sophisticated method, an extension of SNC, involves augmenting the state vector to estimate process noise parameters. This advanced model is known as Dynamic Model Compensation (DMC) [9].

DMC assumes that process noise is correlated in time and that this correlation can be exploited to reconstruct unmodeled dynamics. By assuming that there is an underlying first-order linear stochastic differential equation governing the evolution of the process noise, the noise can be treated as part of the dynamical system and estimated as part of the state. This is often done by assuming that the process noise evolves according to a first-order Gauss-Markov process (more precisely known as an Omstein-Uhlenbeck process). A Gauss-Markov process is one that obeys a Gaussian probability law and displays the Markov property. The Markov property means that the probability density function at t_k given its past history at $t_{k-1}, t_{k-2}, \ldots$ is equal to its probability density function at t_k given its value at t_{k-1}. A Gauss-Markov process obeys a differential equation (often referred to as a Langevin equation) of the form:

$$\dot{\mathbf{w}} = -\mathbf{B}\mathbf{w} \tag{20}$$

Typically, the damping coefficient matrix $\mathbf{B}$ is diagonal and defined as $B_{ii} = 1/\tau_i$ through the correlation time τ_i. Using this approximation, the unknown perturbations, $\mathbf{w}$, can be augmented to the state vector and estimated directly to approximate any unmodeled accelerations during the orbit determination procedure. The augmented state vector of the spacecraft during estimation is $\mathbf{x} = [\mathbf{r},\ \mathbf{v},\ \mathbf{w}]^T$ and $\dot{\mathbf{x}} = [\mathbf{v},\ \dot{\mathbf{v}} + \mathbf{w},\ \dot{\mathbf{w}}]^T$. In the orbit determination process, therefore, the dynamic and measurement model Jacobians should be updated as follows: $\mathbf{A}' = [\mathbf{A},\ \mathbf{D};\ \mathbf{0}_{3\times6},\ -\mathbf{B}]$, $\mathbf{D} = [\mathbf{0}_{3\times3};\ \mathbf{I}_{3\times3}]$, and $\mathbf{H}' = [\mathbf{H},\ \mathbf{0}_{m\times3}]$. In the DMC model, the discrete process noise covariance can be obtained from the continuous process noise covariance as follows:

$$\mathbf{Q}\left(t_k,\ t_{k-1}\right) = \mathbf{\Gamma}\left(t_k,\ t_{k-1}\right)\mathbf{Q}\mathbf{\Gamma}\left(t_k,\ t_{k-1}\right)^T \tag{21}$$

$$\mathbf{\Gamma}\left(t_k,\ t_{k-1}\right) = \begin{bmatrix} \frac{\Delta t^2}{2}\mathbf{B}^{-1} - \Delta t\mathbf{B}^{-2} + \mathbf{B}^{-3}\left(\mathbf{I}_{3x3} - e^{-\mathbf{B}\Delta t}\right) \\ \Delta t\mathbf{B}^{-1} - \mathbf{B}^{-2}\left(\mathbf{I}_{3x3} - e^{-\mathbf{B}\Delta t}\right) \\ \mathbf{B}^{-1}\left(\mathbf{I}_{3x3} - e^{-\mathbf{B}\Delta t}\right) \end{bmatrix} \tag{22}$$

E. Function Approximators

Artificial Neural Networks (ANNs) are frequently used to approximate nonlinear functions. These networks consist of interconnected units that emulate some aspects of human neurons, the fundamental elements of the nervous system. ANNs can be categorized into three main types: feedforward, convolutional, and recurrent networks. Figure 4 illustrates a typical feedforward ANN, featuring an output layer comprising two units, an input layer comprising four units, and two hidden layers (excluding input and output layers). Each connection is associated with a weight and a bias, both real-valued, similar to the strength of synaptic connections in biological neural networks.

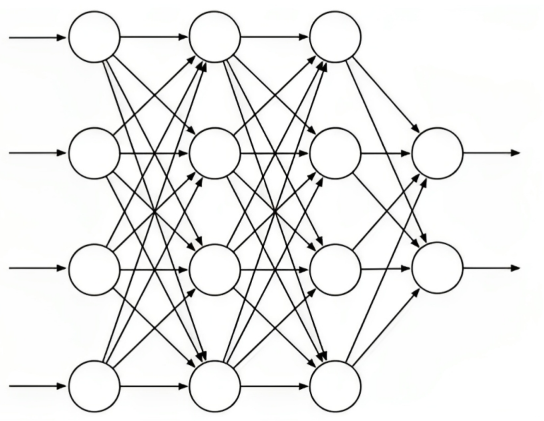
Fig. 4. A Feedforward Artificial Neural Network (Feedforward ANN) with four input units, two output units, and two hidden layers. Source: Sutton and Barto [10].

Artificial Neural Networks (ANNs) are crafted to approximate a specific function f by assigning an output value to an input value via a mapping $y = N(x, \boldsymbol{\theta})$. This process involves learning the values of the parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, known as weights, to achieve the most precise approximation. ANNs typically employ Stochastic Gradient Descent (SGD) methods for learning. In the common supervised learning scenario, the objective function is often the Mean Squared Error (MSE) computed over a set of labeled training examples. To optimize performance, it is essential to assess how changes in the weights of each connection affect overall network performance. This is typically accomplished through Backpropagation (BP), a method used to compute the gradient of the network.

Before delving into the intricacies of neuron structure and training algorithms, it is crucial to understand why ANNs possess the capability to accurately approximate any non-linear and complex function. This capability is facilitated by activation functions, as illustrated by Theorem 1.

Theorem 1 (Universal Approximation Theorem)**.** *Let $\phi : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, continuous function (i.e., the activation function). Let I_m denote the hypercube unit m dimensional* $[0,\ 1]$*. The space of continuous functions with real value in I_m is denoted by $C\left(I_m\right)$. Then, given any $\varepsilon > 0$ and any function $f \in C\left(I_m\right)$, there exist an integer N, real constants $v_i,\ b_i \in \mathbb{R}$ and real vectors $\boldsymbol{\theta}_i \in \mathbb{R}^p$ for $i = 1, ..., N$ such that we may define:*

$$F\left(\mathbf{x}\right) = \sum_{i=1}^{N} v_i \varphi\left(\boldsymbol{\theta}_i^T \mathbf{x} + \mathbf{b}\right) \tag{23}$$

As an approximate realization of the function f

$$\left|F\left(\mathbf{x}\right) - f\left(\mathbf{x}\right)\right| < \varepsilon \qquad \forall \mathbf{x} \in I_m \tag{24}$$

Among the ANNs types mentioned earlier, the most prevalent deep model is the feedforward ANN, commonly referred to as a Multi-Layer Perceptron (MLP). This type of network facilitates the flow of information from the input layer through the hidden layers to the output layer without any feedback, resulting in an acyclic graph. The fundamental component of an MLP is the neuron. As depicted in Figure 5, the induced local field v_j of the neuron j, which serves as the input to the activation function $\varphi_j(\cdot)$, can be formulated as:

$$v_j = \sum_{i \in C_i} \theta_{ij} x_j + b_j \tag{25}$$

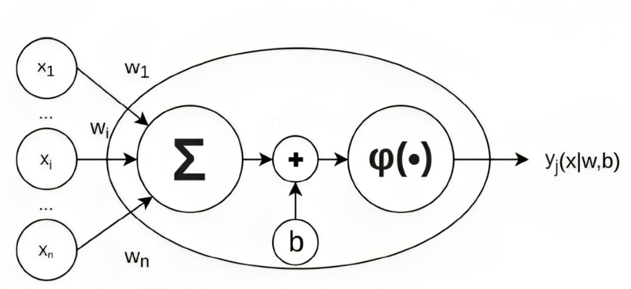

Fig. 5. Elementary architecture of a Multi-Layer Perceptron (MLP). Source: Silvestrini et al. [11].

The neurons in the set C_i are linked to layer j, and b_j represents the bias term. The output of a neuron y_j is determined by the activation function (which performs the non-linear transformation) applied to the local field v_j:

$$y_j = \varphi_j\left(v_j\right) \tag{26}$$

Each training algorithm for ANNs revolves around an optimization task. Typically, determining the weights and biases of an ANN entails pinpointing the most favorable set of parameters that minimize a designated loss function:

$$\left(\boldsymbol{\theta}^*,\ \mathbf{b}^*\right) = \operatorname{argmin} J\left(\boldsymbol{\theta},\ \mathbf{b}\right) \tag{27}$$

One of the most commonly used loss functions in deep learning, particularly for regression problems, is the Mean Squared Error (MSE). Quantifies the average squared difference between the predicted values and the actual target values. The formula for MSE is given by:

$$J_{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{28}$$

Here, y_i represents the actual target value for the i-th sample, $\hat{y}_i$ represents the predicted value for the i-th sample, and n is the total number of samples in the dataset. A lower MSE indicates better performance, with zero being the ideal value when the predictions match perfectly the actual targets. Usually, obtaining a closed-form solution for this problem is impractical. Hence, iterative methods are often employed, leveraging the derivative of the objective function to converge to the optimal value. The BP algorithm specifically pertains to calculating the gradient, while another algorithm, such as SGD, performs the actual learning process using this gradient. The BP algorithm uses two passes through a network: forward and backward. The forward pass evaluates network output and neuron activation functions, while the backward pass calculates local gradients. In gradient-based approaches for a conventional MLP, adjustments to neuron weights, denoted θ_{ij}, are guided by partial derivatives, as determined by the chain rule:

$$\frac{\partial J}{\partial \theta_{ij}} = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial v_j} \frac{\partial v_j}{\partial \theta_{ij}} \tag{29}$$

The neuron weights are modified by $\Delta\theta_{ij}$ through a step in the weight space guided by the gradient composed of these partial derivatives. Typically, this falls into the family of optimization algorithms known as Gradient Descent (GD) [12], where the parameters $\boldsymbol{\theta}$ of an objective function J are adjusted opposite to the gradient of the objective function $\boldsymbol{\nabla} J\left(\boldsymbol{\theta}\right)$ with a step size α called the learning rate. This iterative process continues until the gradient approaches zero, eventually leading to a local optimal solution based on the batch of data used to estimate the gradient. The standard update equation is as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \widehat{\boldsymbol{\nabla} J\left(\boldsymbol{\theta}_k\right)} \tag{30}$$

Methods that utilize the complete training set are labeled as Batch Gradient methods, whereas those employing a subset of training examples are typically referred to as Stochastic Gradient methods. Among the vital optimization algorithms is Adam, short for the Adaptive Moment Estimation method [13], which integrates the concepts of momentum and adaptive learning rates.

III. Problem Formulation

To assess the effectiveness of the proposed online residual dynamics learning approach, a straightforward scenario is being developed. Consider a spacecraft navigating in orbit around the Moon. The on-board filter equations of motion solely consider the gravitational accelerations of the Earth and Moon (hence, the Circular Restricted Three-Body Problem). However, the true trajectory is propagated by incorporating the gravitational acceleration of the Sun and the effects of Solar Radiation Pressure (thus, the Bicircular Restricted Four-Body Problem with SRP). This discrepancy between the reduced filter equations of motion and the actual dynamics results in a model mismatch. To address this, the residual dynamics is estimated using Dynamical Model Compensation (DMC) within an Extended Kalman Filter (EKF). In this scenario, noisy range and range-rate measurements are acquired for the spacecraft navigation algorithm from an inertially-fixed observer located at the origin of the rotating synodic frame.

The actual initial state, initial state covariance, process noise matrix, measurement noise matrix, and the damping coefficient matrix $\mathbf{B}$ associated with the Langevin equation are detailed in Table I. The initial estimated state simply represents a state randomly selected from the true initial state, taking into account the provided state covariance distribution. Although this scenario may not precisely mirror real-world conditions, it provides a valuable opportunity to assess the initial efficacy of the Artificial Neural Network (ANN) in reconstructing residual dynamics within the orbit determination pipeline.

TABLE I
INITIAL STATE, INITIAL STATE COVARIANCE, PROCESS NOISE MATRIX, MEASUREMENT NOISE MATRIX, AND DAMPING COEFFICIENT MATRIX

State	Value	Unit
$\mathbf{r}_0$	$[1, 0, 0]$	l^*
$\mathbf{v}_0$	$[0, 0.5, 0]$	l^*/t^*
$\mathbf{w}_0$	$[0, 0, 0]$	l^*/t^{*2}
$\mathbf{P}_{\mathbf{r}_0}$	diag($[10^{-12}, 10^{-12}, 0]$)	$(l^*)^2$
$\mathbf{P}_{\mathbf{v}_0}$	diag($[10^{-8}, 10^{-8}, 0]$)	$(l^*/t^*)^2$
$\mathbf{P}_{\mathbf{w}_0}$	diag($[10^{-2}, 10^{-2}, 0]$)	$(l^*/t^{*2})^2$
$\mathbf{Q}$	diag($[10^{-2}, 10^{-2}, 10^{-2}]$)	$(l^*/t^{*2})^2$
$\mathbf{R}$	diag($[10^{-16}, 10^{-10}]$)	$[(l^*)^2, (l^*/t^*)^2]$
$\mathbf{B}$	diag($[1, 1, 1]$)	$1/t^*$

Throughout the flight and the estimation process handled by the EKF, the estimated state of the spacecraft $\hat{\mathbf{x}}$ and the estimated residual dynamics $\hat{\mathbf{w}}$ obtained through the first-order DMC act as input characteristics and output labels, respectively, to train the ANN in a supervised learning manner. The neural network reconstructs the dynamical disturbance within the estimation pipeline in a black-box fashion, leveraging DMC to generate training data to update the ANNs online during flight.

Once a sufficient number of observations are collected, the neural network undergoes updates using traditional stochastic gradient descent for a specified number of epochs. Consequently, the reconstruction of residual dynamics becomes embedded entirely in the weights and biases of the network. The ANN predicts the acceleration resulting from disturbance effects as its output, using the state of the spacecraft as input after training. This is denoted as $\mathbf{a}_{res} = \mathbf{f}\left(\mathbf{x},\ \boldsymbol{\theta}^*,\ \mathbf{b}^*\right)$. The training methodology for the ANN within the EKF involves selecting appropriate hyperparameters and model architecture. Three critical hyperparameters require careful attention. First, the learning rate governs the magnitude of gradient descent steps during network training. Although smaller values improve stability, they may also decelerate learning. Secondly, the number of epochs determines how many times the dataset is iterated during training. Insufficient epochs may lead to underfitting, while excessive epochs can result in overfitting. Additionally, batch size, which refers to the number of samples used in an SGD step, is a crucial consideration. Large batches provide more accurate estimates of the cost landscape, but may increase the risk of converging into local minimums. In contrast, small batch sizes produce noisier steps, potentially slowing down learning but offering a better chance of escaping local minimums. To ensure the validity of the results, it is crucial to thoroughly validate the training process. This involves partitioning the measurement batch into a 80% training dataset and a 20% validation dataset. This partitioning facilitates cross-validation and helps identify any signs of overfitting post-training.

In terms of deep learning models, alternative studies such as those of Bauersfeld et al. [2] advocate the utilization of Temporal-Convolutional (TCN) encoders. Similarly, Silvestrini and Lavagna [14] propose the use of Long Short-Term Memory (LSTM) networks, while Martin and Schaub [3] explore the efficacy of Physics-Informed Neural Networks (PINNs). Recent advances in deep learning, as highlighted by Chen et al. [15], suggest the potential application of Neural Ordinary Differential Equations (Neural ODEs), which could be a novel avenue to explore in aerospace contexts. Future investigations may involve analyzing and benchmarking these models to determine their effectiveness in relevant applications. However, given the preliminary nature of this analysis, only feedforward networks with a few small hidden layers are considered to yield initial results. The selected ANN architecture comprises six input nodes (to accommodate $\mathbf{x}$) and three output nodes (to generate $\mathbf{a}_{res}$). Each layer, excluding the output layer, utilizes a ReLU activation function, while the output layer employs a linear activation function. As outlined later, the resulting neural network is quite compact, featuring a total parameter count of 433 and an estimated memory usage of 1732 bytes.

The exhaustive selection of hyperparameters and specifics regarding the deep learning model is described in Table II. Although no dedicated hyperparameter search is performed in this study, it signifies a promising avenue for future investigation, particularly in balancing function approximation accuracy with spacecraft on-board computational efficiency. The empirical validation of hyperparameters revolves around minimizing prediction errors in the unseen validation set.

TABLE II
HYPERPARAMETERS

Hyperparameters	Values
Hidden Layers	3
Width Hidden Layers	10
Activation Function	ReLU
Epochs	200,000
Learning Rate	1e-4
Batch Size	10,000
Loss Function	MSE
Optimizer	Adam

It is important to emphasize that this analysis is preliminary; the ultimate goal is to establish a robust and efficient method for training the neural network seamlessly during flight and integrating the learned residual dynamics into the GNC algorithms. This involves leveraging acceleration for improved propagation and utilizing Jacobians for navigation filters and guidance targeting schemes. However, in this initial analysis, a dataset has been accumulated and collectively provided to the training method, deviating somewhat from the strict definition of ”online” learning and without demonstrating the exploitation of the learned dynamics. However, efforts have been made to align with this concept as closely as possible to yield preliminary results and pave the way for further enhancements. For example, a very small batch size has been chosen for measurements compared to similar works within a supervised learning framework (that is, thousands instead of millions) [2]. Future endeavors will focus on more effectively integrating the learning and GNC algorithms.

IV. NUMERICAL RESULTS

The spacecraft's initial state and all EKF hyperparameters are detailed in Table I. Solar Radiation Pressure (SRP) acceleration is modeled for a spacecraft with specific characteristics:

a reflectivity coefficient of $C_r = 0.1$, a cross-section of $A = 1\ m^2$, and a mass of $m = 500\ kg$. The trajectory of the spacecraft is propagated, and thus the filtering process is underway, with a flight time of $ToF = 0.5\ t^* = 2.17\ days$ and a time step of $\delta t = 5 \cdot 10^{-5}\ t^* = 18.76\ s$, resulting in the generation of the required batch of $10,000$ data points (that is, filter estimates) within the specified interval for the next training phase. At each time step, the spacecraft receives range and range-rate measurements to trigger the orbit determination pipeline. The motion from the provided initial state follows an elliptical trajectory around the Moon, completing numerous revolutions. The equations governing this motion are propagated using the `scipy.integrate` library in Python, employing the non-stiff Adam predictor-corrector method known as `LSODA`. Relative and absolute tolerances of $2.22 \cdot 10^{-14}$ are utilized. The code used in this study is accessible on a dedicated GitHub repository[1]. The results showing both the true trajectory and the filtered trajectory are provided in Figure 6.

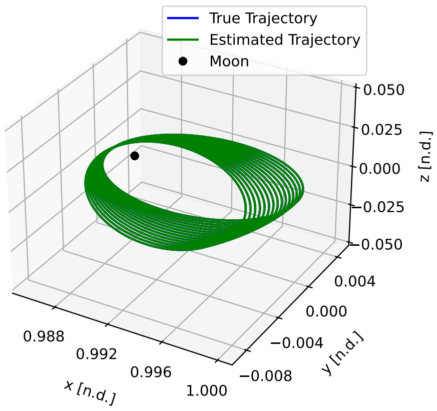

Fig. 6. True and estimated trajectory from spacecraft measurements.

The filter effectively estimates the spacecraft's state, as indicated by the close alignment of the true and estimated trajectories shown in the figure. This alignment is further illustrated by the logarithmic scale of position and velocity errors $\boldsymbol{\varepsilon}_\mathbf{x}$ between the true and estimated values, as depicted in Figure 7. As the trajectory lies in the xy-plane, the z-coordinate is omitted. These observations confirm the successful convergence and accuracy of the filter implementation within this scenario. As outlined in the Problem Formulation section, the onboard filter equations of motion only consider the CRTBP, whereas the true trajectory is propagated using the BCRFBP with SRP effects. This discrepancy in dynamics between the considered and true models is addressed by compensating for the mismatch and estimating the residual dynamics using DMC within the EKF. The results are depicted in Figure 8, which illustrates, on a logarithmic scale, the error $\boldsymbol{\varepsilon}_\mathbf{w}$ between the residual accelerations estimated by the filter and the true values. Once again, the filter demonstrates accurate convergence, with all errors falling below the 3σ threshold and consistently remaining around the fourth decimal place.

[1]https://github.com/giovannifereoli/Online-Residual-Dynamics-Learning-from-Spacecraft-Measurements

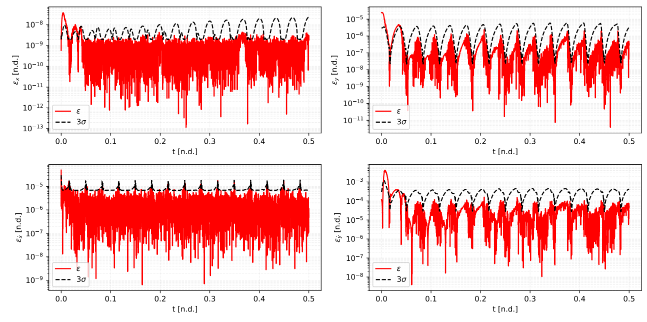

Fig. 7. Error between true and estimated spacecraft state.

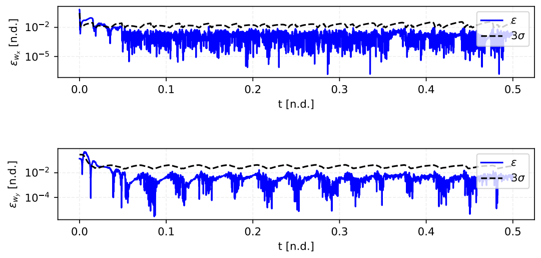

Fig. 8. Error between true and estimated spacecraft residual dynamics.

Throughout the estimation process, features and labels are generated to train the deep learning model: the estimated spacecraft state, denoted $\hat{\mathbf{x}}$, serves as input, while the estimated spacecraft dynamical disturbances, denoted $\hat{\mathbf{w}}$, serve as target output. Once the spacecraft filter has generated the specified batch size[2] outlined in Table II, the model undergoes gradient descent for the specified number of epochs, also detailed in the same table. The progression of the MSE loss function throughout the training epochs is depicted in Figure 9, illustrating the correctness of the model learning process. The MSE loss consistently and monotonically decreases its value until it reaches a plateau, a trend that will be further highlighted in the subsequent validation phase.

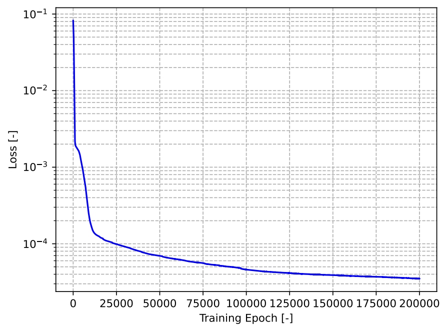

Fig. 9. Evolution of the loss function across training epochs.

Empirical selection is used to determine hyperparameters. Less successful models were associated with high learning rates, long training duration (epochs), and small batch sizes. These

[2]The filter swiftly reaches convergence in both spacecraft state and disturbance estimates, typically within 0.05 t^*. This rapid convergence justifies the utilization of the entire set of estimates for training purposes.

results were somewhat anticipated. Furthermore, sensitivity to DMC parameters, such as τ, is observed during the training process. Typically, higher correlation times values contributed to more effective acceleration measurements and promoted more stable training.

Validation during training is crucial for monitoring the training process and identifying potential overfitting. It also assesses the neural network's capability to effectively model the residual dynamics. Since the filter successfully converges to estimate the disturbance acceleration, accurate fitting of filter estimates implies precise reconstruction of the disturbance by the model. Therefore, during training, the validation dataset is evaluated by comparing the model's predictions with the target labels, instead of solely relying on the loss function, to ensure comprehensive evaluation of overfitting and modeling effectiveness. The Prediction Relative Percent Error is defined as follows:

$$\varepsilon_{rel,\%} = 100 \cdot \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|} \tag{31}$$

The validation process is depicted in Figure 10, further confirming the success of the training process in avoiding overfitting and effectively learning from the dataset. Furthermore, the relative prediction percent error converges to approximately 3. 4% as training progresses. Interestingly, even within the first quarter of the total training epochs, starting with initial random ANN parameters, there is a significant enhancement, reducing the error from over 300% to around 5%. Since the EKF converged to the true value and the model converged to the estimated, it can be inferred that the model correctly converged to true disturbances. In the future, it would be interesting to conduct a more in-depth evaluation of the error between the ANN model and the truth, as well as feeding the learned residual dynamics back to the filter to observe real-time improvement in its estimates.

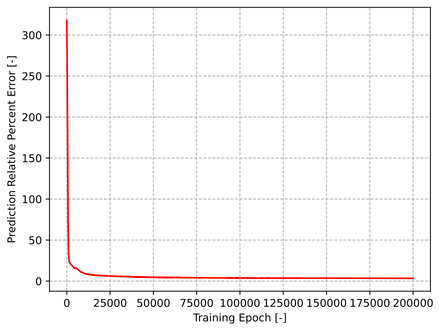

Fig. 10. Evolution of the Prediction Relative Percent Error over training epochs, assessed on the validation dataset.

V. Conclusion

This study introduces an innovative method for spacecraft disturbance reconstruction by combining first-principles modeling with a residual term learned through an Artificial Neural Network (ANN), using data provided by the on-board Extended Kalman Filter (EKF) estimates. The neural network undergoes Online Supervised Training (OST), utilizing the spacecraft state and residual dynamics estimates obtained from the Dynamical Model Compensation (DMC) method implemented in the filter as training data. The aim of the method is to mitigate the impact of modeling errors, external disturbances, and non-linearities. Preliminary numerical validation and performance evaluation focus on a spacecraft absolute navigation scenario within cislunar space. Simulation results demonstrate the effectiveness of the proposed solution in reconstructing spacecraft dynamics, particularly in handling Sun's fourth-body and Solar Radiation Pressure disturbances. The experiment highlights the efficacy of DMC with OST; DMC accurately estimates the modeling mismatch between true dynamics and on-board first-principles dynamics, while the ANNs fully learn the estimated residual dynamics, showing a relative error of up to 4% in the validation dataset. This advancement has significant potential to accelerate the development and improve the robustness of advanced Guidance, Navigation, and Control (GNC) strategies for spacecraft. Future efforts should aim to better accommodate the limited computational resources onboard the spacecraft and integrate the learning methods more effectively within the GNC algorithms of the spacecraft, benefiting from ANN differentiability.

VI. Future Work

Initially, it is crucial to thoroughly evaluate the effectiveness of the model for on-board spacecraft tasks. This assessment should focus on reducing the accumulated trajectory propagation error across multiple orbits and enhancing the performance and robustness of the GNC algorithms. Leveraging the learned dynamics acceleration and derivatives through deep learning models' auto-differentiation capabilities can facilitate these improvements. Additionally, seamless integration of the developed algorithm into the GNC framework is vital to achieve true real-time operation. Exploring various design choices, such as the frequency of updating the ANN model for effective learning, assessing the computational feasibility of on-board implementation, and analyzing generalization capabilities beyond the training data, is essential to extend and advance this work.

To directly enhance this work, exploring the application of DMC with second-order dynamics shows promise in improving the accuracy of the estimation process and obtaining a higher-quality data set for training. Investigating more realistic scenarios will be crucial in refining the system's performance across diverse and challenging conditions. Additionally, experimenting with various deep learning models will allow the identification of the optimal architecture. Exploring the efficacy of transfer learning concepts to pre-train and initialize the deep learning model before flight, potentially enhancing its performance during flight, could be an additional valuable avenue of investigation.

VII. Contributions and Release

The sole contributor to this work is the author. All algorithms developed are implemented by the author, using Python libraries such as `NumPy`, `SciPy`, `Matplotlib`, and `PyTorch`. The author grants permission for this report to be publicly posted.

References

[1] Jason M Leonard, Felipe G Nievinski, and George H Born. Aas 11-502 gravity error compensation using second-order gauss-markov processes, 2013.

[2] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza. Neurobem: Hybrid aerodynamic quadrotor model. *ArXiv*, 6 2021. doi: 10.15607/RSS.2021.XVII.042. URL http://arxiv.org/abs/2106.08015.

[3] John R. Martin and Hanspeter Schaub. Preliminary analysis of small-body gravity field estimation using physics-informed neural networks and kalman filters. In *Proceedings of the 73th International Astronautical Congress*. International Astronautical Federation, 2022.

[4] Tae Ha Park and Simone D'Amico. Online supervised training of spaceborne vision during proximity operations using adaptive kalman filtering. *ArXiv*, 9 2023. URL http://arxiv.org/abs/2309.11645.

[5] Nathan Harl, Karthikeyan Rajagopal, and S. N. Balakrishnan. Neural network based modified state observer for orbit uncertainty estimation. *Journal of Guidance, Control, and Dynamics*, 36(4):1194–1209, 2013.

[6] Andrea Colagrossi, Michelle Lavagna, James Douglas Biggs, and Pierangelo Masarati. *Absolute and Relative 6DOF Dynamics, Guidance and Control for Large Space Structures in Cislunar Environment*. PhD thesis, Politecnico di Milano, 2019.

[7] Patrick W. Kenneally. High geometric fidelity solar radiation pressure modeling via graphics processing unit. Master's thesis, University of Colorado Boulder, 2016.

[8] Brian P. Mccarthy and Kathleen C. Howell. Quasi-periodic orbits in the sun-earth-moon bicircular rrestricted four-body problem. In *31st AAS/AIAA Space Flight Mechanics Meeting*, 2021.

[9] B.D. Tapley, Bob Schutz, and George Born. Fundamentals of orbit determination, 2004.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction Second edition, in progress*. MIT Press, 2019.

[11] Stefano Silvestrini, Lorenzo Pasqualetto Cassinis, Robert Hinz, David Gonzalez-Arjona, Massimo Tipaldi, Pierluigi Visconti, Filippo Corradino, Vincenzo Pesce, and Andrea Colagrossi. *Modern Spacecraft GNC*, pages 819–981. Elsevier, 2023.

[12] Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, 9 2016. URL https://arxiv.org/abs/1609.04747.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations, San Diego*, 12 2015. URL https://arxiv.org/abs/1412.6980.

[14] Stefano Silvestrini and Michèle Lavagna. Deep learning and artificial neural networks for spacecraft dynamics, navigation and control, 10 2022. ISSN 2504446X.

[15] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.