A Reduced-complexity Trajectory Generation Algorithm for Three-body Regimes with Minimum Predefined Data

Brian Baker-McEvilly, Hansaka Aluvihare, Sirani M. Perera, and David Canales

***Abstract*—Computational limitations and big data analysis pose challenges in seeking efficient techniques to predict trajectories in three-body dynamics. Thus, a reduced-complexity classical algorithm is proposed utilizing predefined spacecraft's position and velocity data to achieve precise and accurate orbital trajectories of the spacecraft within three-body dynamics. The proposed algorithm seamlessly solves polynomial interpolation along with the boundary and interior conditions without the need for the spacecraft's acceleration data. Once the algorithm is derived, it will be tested across a diverse variety of periodic trajectories in the Earth-Moon system. Moreover, a comparative analysis is performed to evaluate the time complexity of the proposed algorithm compared with conventional orbit propagators. Finally, the proposed algorithm will be utilized and extended to learn and update distant retrograde orbits (DRO) while training a neural network with several initial conditions composing minimum predefined data. After the training is done, the neural network is used to accurately predict DRO trajectories for a given initial condition, demonstrating the exceptional accuracy and effectiveness of the proposed learning process.**

***Index Terms*—Cislunar, CR3BP, Orbital Trajectories, Sparse Matrices, Complexity and Performance of Algorithms, Minimum Predefined Data, Machine Learning, Neural Networks.**

NOMENCLATURE

U^*	Psuedo-potential
μ	CR3BP mass parameter
μ_{Earth}	Earth's gravitational parameter
$\tilde{L}$	Lower tridiagonal triangular matrix
$\underline{r}$	Cartesian position
A	Coefficient matrix
H	Polynomial equation
h	Polynomial coefficient
JC	Jacobi constant
m	Mass
t	Time
U	Upper triangular matrix

B. Baker-McEvilly is with the Department of Aerospace Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32703 USA e-mail: bakermcb@my.erau.edu

H. Aluvihare is with the Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32703 USA e-mail:aluvihah@my.erau.edu

S. M. Perera is with the Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32703 USA e-mail: pereras2@erau.edu (see https://faculty.erau.edu/Sirani.Perera).

D. Canales is with the Department of Aerospace Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32703 USA e-mail:canaled4@erau.edu (see https://staresearchgroup.com).

Sirani M. Perera's and Hansaka Aluvihare's work was partially funded by the National Science Foundation with Award Number 2229473.

I. INTRODUCTION

The Cislunar region (Fig. 1) is anticipated to see rapid growth over the next decade, with over thirty missions confirmed to travel into the region before 2030 with many more in planning [1], [2]. This is the result of the renewed interest in large Lunar programs, the commercial space industry breaking into Lunar space, and many new space-faring countries sending out Lunar missions. Large-scale programs, such as the United States' Artemis program, China's Chang'E program, and Russia's Luna program, all are multi-stage programs whose aim is to establish long-term Lunar bases on the Moon [3]. The commercial sector is increasingly drawn to lunar missions, spurred by initiatives like NASA's Commercial Lunar Payload Services (CLPS) which offers contracts to private companies for payload deliveries [4]. Independent ventures are also evident, exemplified by Hakuto-R's commercial lunar landing attempt in late 2023. Concurrently, new national players like India and South Korea are making strides in Cislunar space, with India's Chandrayaan-3 landing and South Korea's KPLO mission [5]. The growing engagement in Cislunar space underscores its strategic importance and commercial potential.

Complex motion in the Cislunar region, influenced by the Earth's and Moon's gravity, is often modeled using the circular restricted three-body problem (CR3BP) [6]. It is a highly nonlinear and sensitive system in which no closed-form solution has yet been derived. Thus, to design and analyze spacecraft trajectories in the CR3BP model, numerical methods are required. Differential corrections are often utilized, formulated as targeting schemes, seeking out trajectories with specific traits [7]–[9]. Novel computational techniques satisfy these desired traits with free variables and constraints implemented into differential correction techniques. In order to propagate orbits within nonlinear dynamical systems, well-applied numerical techniques including Gauss-Legendre, Dormand-Prince, Chebyshev-Picard [10], Gragg-Bulirsch-Stoer [11], and Adams-Bashforth [12] have been studied. Every method has advantages and disadvantages, but these numerical methods are usually computationally expensive. These typically tax the computational bandwidth of systems; bandwidth that could be allocated to other tasks. For instance, small spacecraft like KPLO and CubeSats, which have limited computational resources, greatly benefit from algorithms that optimize computational efficiency. Therefore, it is important to seek algorithms that reduce the complexity of the problem while still providing accurate trajectory solutions.

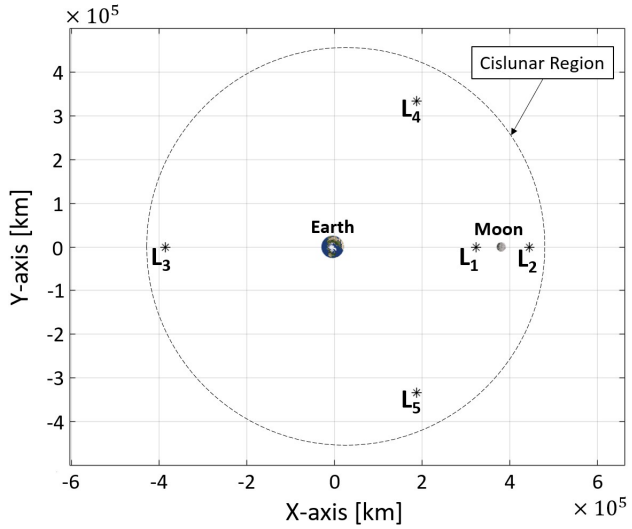


Fig. 1: Cislunar region of the Earth-Moon system depicted with the Earth radius scaled $\times 4$ and Moon radius scaled $\times 6$, and where $L_1, L_2, \cdots, L_5$ are Lagrange points.

Research in astrodynamics aims to progress the efficiency of trajectory propagation and generation through the use of polynomial interpolation and machine learning (ML) approaches. Studies show that polynomial interpolation effectively predicts positions between ephemerides for navigational satellites near Earth, achieving meter-level accuracy with 15-minute measurement intervals [13]. Karepova [14] completes an insightful summary on the formulation of polynomial interpolation for the purpose of trajectory generation and on the accuracy of different order solutions. Recent studies, such as those by Jacco Geul [15] and Parrish [16], explore polynomial interpolation in spacecraft trajectory calculations. Geul's work examines Hermite polynomial interpolation, Householder, and bisection methods in the perturbed two-body problem, finding that these methods approximate time similarly with increased sampling, although Hermite interpolation shows the least position error. Parrish's research explores polynomial interpolation to optimize low-thrust transfers within the CR3BP, integrating dynamical constraints for enhanced accuracy. The study identifies various low-thrust transfer families, from single to multi-revolution solutions, and notes decreased accuracy near the Moon, where the polynomial models struggle with sensitive dynamics. Developments regarding ML techniques for orbit propagation, models based on deep learning, neural network (NN), and support vector machines (SVM), are presented in the literature [17]–[22]. These implement SVM to improve satellite orbit prediction accuracy [18], [19]. Their results showed that SVM-based ML algorithms greatly increase the accuracy of orbit prediction, especially for resident space objects (RSO). It is important to note that the accuracy and precision of SVM-based methods strongly depend on the volume of available data [17]. To mitigate orbit propagation model flaws, an orbit propagation approach based on long short-term memory (LSTM) has been presented [17], [23]. Presently, the use of NN to solve the three-body problem has attracted a lot of attention [24]–[26]. In particular, [24] used an artificial neural network (ANN) to approximate solutions to the three-body problem over a fixed computation time interval, faster than Brutus integration. Furthermore, an alternate method is provided by Hamilton NN, which functions based on total energy conservation [25]. These networks tackle imprecise predictions arising from the conversion of time from continuous to discrete values, offering enhanced performance, particularly in systems with multiple degrees of freedom.

Considering the state-of-the-art trajectory generation techniques [13]–[21], the objective of this work is to explore a computationally efficient method for accurately determining orbit trajectories and to enable trajectory prediction using a low-computation NN. Typically, the problem of trajectory generation is approached as a system of differential equations with given initial conditions, where the dynamical system is numerically integrated with the initial values. The proposed trajectory generation approach is based on initial and boundary conditions to obtain the trajectory. Thus, a reduced low-complexity algorithm (RLCA) is proposed that utilizes position and velocity data at boundary and interior conditions to derive a piecewise continuous polynomial describing a trajectory within the specified boundary. The RLCA is used to find these polynomials based on the structure of the system and the decomposition of the coefficient matrix into tridiagonal and bidiagonal matrices. The results show that the algorithm reduces computation time over 70% for every demonstrated case, with a sacrifice in accuracy that varies based on the simulated orbits. Furthermore, the algorithm acts independently of a dynamical model, being applicable to the N-body problem. The algorithm is tested across numerous periodic trajectories in the CR3BP, analyzing the time and computation efficiency of the proposed method while comparing it with numerical integration techniques. Once the algorithm is validated, it will be used for an initial value problem using machine learning algorithms. Therefore, this manuscript includes the initial endeavors of using a NN to approximate spacecraft trajectories in the CR3BP framework, utilizing the RLCA to train, learn, and update DRO trajectories, and hence accurately predict trajectories with a low-complexity ML algorithm. The RLCA-based NN demonstrates faster learning and updates on the trained trajectories compared to the conventional numerical integration-based ML model. Finally, this research demonstrates that the RLCA-based ML model, when compared to the numerical integration-based ML model, exhibit lower error rates in generating future trajectories within an orbital period.

This paper provides a thorough explanation of the proposed RLCA, Earth-Moon dynamical model, computational analysis, and RLCA-based low-complexity ML algorithm to determine trajectories. The CR3BP and 2BP dynamical models are described in section II. Next, section III summarizes the author's previous work on a low-complexity algorithm, derives the new reduced low-complexity algorithm to determine trajectories, and provides the theory for a NN model trained by the RLCA to determine trajectories in the CR3BP. Then, section IV completes an analysis of the application of the RLCA across different orbital trajectories and the proposed RCLA-based low-complexity NN. Finally, section V concludes the paper.

II. Dynamical Models

A. The Two-Body Problem

In Cislunar space, spacecraft are subject to the influence of both the Earth and the Moon, requiring the incorporation of three-body dynamics. However, for spacecraft in close vicinity of a gravitational body, the influence of the third body may be neglected. In this instance, the two-body problem (2BP) is presented. The 2BP implements a primary body and a spacecraft of negligible mass. The dimensional position and velocity of the spacecraft are defined using $\underline{r} = [x, y, z]^{\mathrm{T}}$ and $\dot{\underline{r}} = [\dot{x}, \dot{y}, \dot{z}]^{\mathrm{T}}$ respectively, defined in a reference frame centered at the primary body, where the superscript T represents the transpose of a vector and dots denote the derivative with respect to time. The resulting motion of the spacecraft in the 2BP is represented by:

$$\ddot{\underline{r}} = -\frac{\mu_{Earth}}{||\underline{r}||^3}\underline{r}, \tag{1}$$

where μ_{Earth} is the gravitational parameter of the Earth and $||\underline{r}||$ is the magnitude of the spacecraft's position vector [27]. For trajectories close to either the Earth or the Moon, the two-body problem is sufficient. However, when trajectories are not close to the Earth or Moon, the influence of both bodies must be considered.

B. The Three-Body Problem

The CR3BP is a well-accepted method of modeling the Earth-Moon system [6], [27], [28]. The system is comprised of three bodies: a large and a small primary (Earth and Moon, respectively), and a spacecraft of negligible mass. The following derivation is completed in the non-dimensional, Earth-Moon rotating frame. The origin of the Earth-Moon rotating frame is located at the barycenter of the system, the $\hat{x}$-axis points towards the Moon, the $\hat{z}$-axis points out of the Earth-Moon orbital plane, and the $\hat{y}$-axis completes the right-handed system. The CR3BP is non-dimensional using characteristic quantities: characteristic length is defined as the distance between the two primaries, characteristic time is the period of the Earth-Moon system, and characteristic mass is the total mass of the Earth and Moon. It is assumed that the primaries orbit the system's barycenter in circular orbits. The mass parameter is found via $\mu = \frac{m_M}{m_E+m_M}$, where the m_E and m_M are the mass of the Earth and Moon respectively [27]. The states of the spacecraft are defined by the position $\underline{r} = [x, y, z]^{\mathrm{T}}$ and velocity $\dot{\underline{r}} = [\dot{x}, \dot{y}, \dot{z}]^{\mathrm{T}}$, where the superscript T represents the transpose of a vector and dots denote the derivative with respect to time. The CR3BP is governed by the following non-dimensional equations:

$$\ddot{x} = 2\dot{y} + \frac{\partial U^*}{\partial x}, \qquad \ddot{y} = -2\dot{x} + \frac{\partial U^*}{\partial y}, \qquad \ddot{z} = \frac{\partial U^*}{\partial z} \tag{2}$$

in which U^* is the pseudo-potential function of the system:

$$U^* = \frac{1-\mu}{||\underline{r}_{E-s/c}||} + \frac{\mu}{||\underline{r}_{M-s/c}||} + \frac{1}{2}(x^2 + y^2) \tag{3}$$

where $\underline{r}_{E-s/c}$ is the position vector from the Earth to the spacecraft, $\underline{r}_{M-s/c}$ is the position vector from the Moon to the spacecraft, and the double bars on each side of the vectors denotes magnitudes [1]. In the CR3BP, five points of equilibrium exist, referred to as libration point (L_1, $\cdots$, L_5). A schematic of the Earth-Moon CR3BP is shown in Fig. 2. Finally, a constant of integration directly related to the energy of a trajectory, the Jacobi constant (JC), exists in the CR3BP:

$$JC = 2U^* - (\dot{x}^2 + \dot{y}^2 + \dot{z}^2). \tag{4}$$

where $(2U^*)$ describes the spacecraft's potential energy, and the latter term, $(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)$ the spacecraft's kinetic energy. Such a parameter is often used to describe a specific orbit in its respective orbit family.

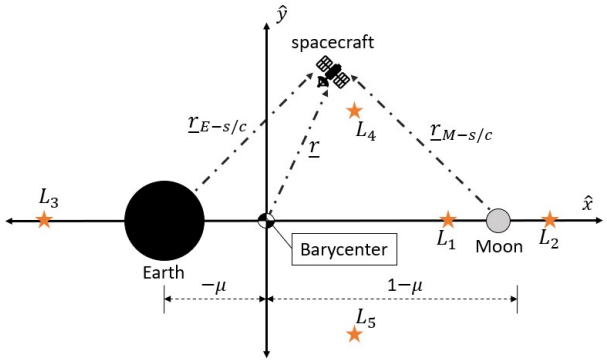


Fig. 2: CR3BP schematic in the Earth-Moon rotating frame, where $\underline{r}$ is the position vector of the spacecraft, $\underline{r}_{E-s/c}$ and $\underline{r}_{M-s/c}$ are the position vectors from the Earth and Moon to the spacecraft respectively, and μ is the mass parameter.

III. A Reduced Low-Complexity Algorithm (RLCA)

A. Previous work on a LCA [29]

A low-complexity algorithm (LCA) has been proposed by the authors to determine orbital trajectories utilizing polynomial interpolation with boundary conditions [29]. The algorithm takes into account the position, velocity, and acceleration of a spacecraft at two distinct times, acting as boundary conditions. The LCA then constructs a polynomial that interpolates the trajectory between these boundaries. The polynomial is the fifth order and the LCA has been derived through a sparse bidiagonal and upper triangular matrix factorization of a coefficient matrix corresponding to the system. This algorithm successfully reproduces entire orbits within the CR3BP, while reducing the computation time by more than 50% in comparison to ODE45. Overall, the LCA is capable of reproducing trajectories with sufficient measurements across Cislunar space for different types of orbits and families [29].

Implementation of the LCA requires extensive knowledge of a spacecraft state, in particular, additional information on the acceleration. Many spacecraft initial state determination methods (i.e. Gibbs Method, Gauss Method, Laplace Method) accurately determine the position and velocity states of a spacecraft from observations. Acceleration data is often calculated using knowledge of the dynamical system, or through additional observations. The previous LCA calculates the acceleration using the position and velocity inputted into a known dynamical system. This approach requires additional

knowledge on a reasonable assumption of the spacecraft's dynamical system. Furthermore, in nonlinear sensitive systems, inaccuracies in position and velocity measurements will significantly perturb the acceleration determined by the dynamical model, diminishing the accuracy of the predicted solution. Thus, it is necessary to develop an algorithm that applies to more realistic implications, operating on position and velocity information that is attainable through spacecraft observations. Learning from previous work on the LCA, a reduced low-complexity algorithm (RLCA) with minimum predefined data to determine orbital trajectories is introduced. While the LCA necessitates position, velocity, and acceleration information at two distinct boundaries, the new algorithm simplifies the process by only requiring position and velocity data at two boundaries and an interior point for constructing the approximate trajectory. Furthermore, the RLCA has been utilized to lay the groundwork for a low-complexity NN in predicting orbital trajectories.

B. Proposed RLCA

A reduced low-complexity algorithm is proposed, satisfying the position ($\underline{r}(t_i) = [x(t_i), y(t_i), z(t_i)]^{\mathrm{T}}$) and velocity ($\underline{\dot{r}}(t_i) = [\dot{x}(t_i), \dot{y}(t_i), \dot{z}(t_i)]^{\mathrm{T}}$) data of the spacecraft in space of $\mathbb{R}^3$ at known distinct times t_i, where $i = 0, 1, \cdots, n$ and $t_0 < t_1 < \cdots < t_n$. The trajectories of the spacecraft over $n+1$ position and velocity data points are described via piecewise functions defined on each interval $I_k = [t_k, t_{k+2}]$, where $k = 0, 2, \cdots, n-2$. Thus, the algorithm first starts with, the vectors $\underline{r}(t_i), \underline{\dot{r}}(t_i) \in \mathbb{R}^3$ in a dimensionless quantities s.t. $x(t), \dot{x}(t) \in \mathbb{R}$, and executes the algorithm in each dimension [29]–[31]. Following the fundamental theorem of Lagrange interpolation and the author's previous work [29], a fifth-degree polynomial is presented that fulfills only position and velocity data at the boundary, as well as interior conditions at each time interval. Thus, for a single dimension, the spacecraft's trajectory on the interval I_k is determined via

$$H_k(x(t)) = h_{0,k} + h_{1,k}t + h_{2,k}t^2 + h_{3,k}t^3 + h_{4,k}t^4 + h_{5,k}t^5 \tag{5}$$

and the corresponding velocity functions of the spacecraft on interval I_k is denoted via,

$$\dot{H}_k(x(t)) = h_{1,k} + 2h_{2,k}t + 3h_{3,k}t^2 + 4h_{4,k}t^3 + 5h_{5,k}t^4 \tag{6}$$

where $t_k \leq t \leq t_{k+2}$, and $h_{0,k}, h_{1,k}, \cdots, h_{5,k}$ are quantities dependent on the position and velocity of the spacecraft at the time interval I_k. To determine the coefficients of the polynomials or trajectories, the known vectors at the time t_k, t_{k+1} and t_{k+2} are utilized, serving as the boundary and interior conditions of the interval I_k. These are expressed as:

$$\begin{aligned} H_k(x(t_k)) = x(t_k), \quad \dot{H}_k(x(t_k)) = \dot{x}(t_k), \\ H_k(x(t_{k+1})) = x(t_{k+1}), \quad \dot{H}_k(x(t_{k+1})) = \dot{x}(t_{k+1}), \\ H_k(x(t_{k+2})) = x(t_{k+2}), \quad \dot{H}_k(x(t_{k+2})) = \dot{x}(t_{k+2}), \end{aligned} \tag{7}$$

where $x(t_k)$ and $\dot{x}(t_k)$ are position and velocity quantities, respectively. The set of equations corresponding to Eqs. (5-7) are rewritten as a matrix equation in the interval I_k s.t.

$$A_k \underline{h}_k = \underline{g}_k \tag{8}$$

where A_k is the coefficient matrix, $\underline{h}_k$ is the vector consisting of the coefficients to the polynomials, and $\underline{g}_k$ is the vector of boundary and interior conditions described by,

$$\begin{aligned} A_k &= \begin{bmatrix} 1 & t_k & t_k^2 & t_k^3 & t_k^4 & t_k^5 \\ 1 & t_{k+1} & t_{k+1}^2 & t_{k+1}^3 & t_{k+1}^4 & t_{k+1}^5 \\ 1 & t_{k+2} & t_{k+2}^2 & t_{k+2}^3 & t_{k+2}^4 & t_{k+2}^5 \\ 0 & 1 & 2t_k & 3t_k^2 & 4t_k^3 & 5t_k^4 \\ 0 & 1 & 2t_{k+1} & 3t_{k+1}^2 & 4t_{k+1}^3 & 5t_{k+1}^4 \\ 0 & 1 & 2t_{k+2} & 3t_{k+2}^2 & 4t_{k+2}^3 & 5t_{k+2}^4 \end{bmatrix}, \\ \underline{h}_k &= [h_{0,k}, h_{1,k}, h_{2,k}, h_{3,k}, h_{4,k}, h_{5,k}]^{\mathrm{T}}, \text{ and} \\ \underline{g}_k &= [x(t_k), x(t_{k+1}), x(t_{k+2}), \dot{x}(t_k), \dot{x}(t_{k+1}), \dot{x}(t_{k+2})]^{\mathrm{T}}. \end{aligned}$$

Note here that the coefficient matrix in Eq. (8) is different from that in [29], where the latter requires knowledge of position, velocity, and acceleration data and the former only requires position and velocity data. By solving the system of Eqs. (8), the polynomials that determine orbital trajectories based on the boundary and interior conditions are obtained. Fundamentally, the proposed approach differs from numerical methods in the manner of framing the trajectory generation problem. Traditional numerical methods typically employ an initial value with a set of differential equations, which is then numerically integrated. The proposed method approaches the solution through a polynomial interpolation problem using known conditions, circumnavigating the need for integration. Using the proposed approach offers different avenues of solving, many of which are computationally inexpensive relative to numerical integration. The proposed method will not replace traditional numerical methods, but offer alternative avenues for prediction in a low-complexity form.

The explicit calculations of coefficients are achieved by taking the explicit inverse (brute force approach) of A_k and multiplying it by the vector $\underline{g}_k$. Therefore, to obtain trajectories from t_0 to t_n, the process must be repeated for each subsequent interval I_k. This brute force method requires the arithmetic complexity of $\mathcal{O}(n^3)$ operations, as there are $6\left\lfloor \frac{n}{2} \right\rfloor$ equations, where $\left\lfloor \frac{n}{2} \right\rfloor$ is the greatest integer less than or equal to $\frac{n}{2}$, resulting $\mathcal{O}(n^3)$ operations. However, the explicit inverse of a dense matrix is rarely computed [32]. To reduce the arithmetic complexity, the A_k in Eqs. (8) is decomposed into a product of highly sparse matrices and an upper triangular matrix. These non-singular sparse matrices are converted into lower-tridiagonal (almost bidiagonal) matrices, denoted by $\tilde{L}$. The system is then solved with the tridiagonal matrix-vector product of $(\prod_{r=1}^{5} \tilde{L}_{r,k})\underline{g}_k = \tilde{L}_{5,k} \cdots \tilde{L}_{1,k}\underline{g}_k$ followed by backward substitution into the matrix U. Resulting in the reduced computational burden of solving the system from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ via

$$U_k \underline{h}_k = \underline{\tilde{g}}_k, \quad \text{where} \quad \underline{\tilde{g}}_k = \left(\prod_{r=1}^{5} \tilde{L}_{r,k} \right) \underline{g}_k \tag{9}$$

where $\underline{\tilde{g}}_k$ is the transformed boundary and interior condition vector. The matrices $\tilde{L}_{r,k} \in \mathbb{R}^{6\times 6}$, $r = 1, 2, .., 5$ and U_k are given via

$$\tilde{L}_{1,k} = \begin{bmatrix} 1 & & & & & \\ -c_k & c_k & & & & \\ -d_k & & d_k & & & \\ & & & 1 & & \\ & & & -c_k & c_k & \\ & & & -d_k & & d_k \end{bmatrix},$$

$$\tilde{L}_{2,k} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & -e_k & e_k & & & \\ & c_k & & -c_k & & \\ & & & & 1 & \\ & & & & -e_k & e_k \end{bmatrix},$$

$$\tilde{L}_{3,k} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & d_k & -d_k & & \\ & & & -2c_k & c_k & \\ & & & & & 1 \end{bmatrix},$$

$$\tilde{L}_{4,k} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & e_k & -e_k & \\ & & & -3b_k & & b_k \end{bmatrix},$$

$$\tilde{L}_{5,k} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & -a_k & a_k \end{bmatrix},$$

$$U_k = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ & 1 & f_{1,k} & f_{2,k} & f_{3,k} & f_{4,k} \\ & & 1 & i_{1,k} & i_{2,k} & i_{3,k} \\ & & & 1 & j_{1,k} & j_{2,k} \\ & & & & 1 & l_k \\ & & & & & 1 \end{bmatrix} \quad (10)$$

where empty spaces represent zero elements and the entries of the matrices are given by:

$$\begin{aligned}
c_k &= \frac{1}{t_{k+1}-t_k}, \\
d_k &= \frac{1}{t_{k+2}-t_k}, \\
e_k &= \frac{1}{t_{k+2}-t_{k+1}}, \\
a_k &= \frac{2d_k - e_k}{e_k d_k}, \\
b_k &= \frac{1}{2d_k - e_k}, \\
f_{1,k} &= t_{k+1} + t_k, \\
f_{2,k} &= t_{k+1}^2 + t_{k+1}t_k + t_k^2, \\
f_{3,k} &= (t_{k+1} + t_k)\left(t_{k+1}^2 + t_k^2\right), \\
f_{4,k} &= t_{k+1}^4 + t_{k+1}^3 t_k + t_{k+1}^2 t_k^2 + t_{k+1} t_k^3 + t_k^4, \\
i_{1,k} &= t_{k+2} + t_{k+1} + t_k, \\
i_{2,k} &= t_{k+2}^2 + t_{k+2}t_{k+1} + t_{k+1}^2 + t_k\left(t_{k+2} + t_{k+1}\right) + t_k^2, \\
i_{3,k} &= \left(t_{k+2} + t_{k+1}\right)\left(t_{k+2}^2 + t_{k+1}^2\right) + t_k(t_{k+2}^2 \\
&\quad + t_{k+2}t_{k+1} + t_{k+1}^2) + t_k^2\left(t_{k+2} + t_{k+1}\right) + t_k^3, \\
j_{1,k} &= t_{k+2} + t_{k+1} + 2t_k, \\
j_{2,k} &= t_{k+2}^2 + 2t_{k+2}t_k + 3t_k^2 + t_{k+1}\left(t_{k+2} + 2t_k\right) + t_{k+1}^2 \\
l_k &= t_{k+2} + 2t_{k+1} + 2t_k. \qquad (11)
\end{aligned}$$

Note here that t_k, t_{k+1}, and t_{k+2} are distinct time values, so division by zeros in calculating a_k, b_k, c_k, d_k and e_k are not encountered. The pseudocode of the proposed RLCA for a closed-loop trajectory is denoted through Algorithm 1.

Algorithm 1 RLCA pseudocode for closed-loop trajectories

```
Collect n known state measurements
Break n measurements to k = n − 2 intervals, each containing an interior condition and two boundary conditions shared with neighboring intervals
for k intervals do
    for Each dimension of the conditions do
        Pre-compute L̃_{·,k} and U_k matrix entries
        Store entries into L̃_{·,k} and U_k matrices
        Compute tridiagonal matrix-vector product, i.e., g̃_k
        Solve for h_k using backward substitution
    end for
end for
```

Unlike the well-known LU decomposition of a matrix for solving a system of equations using full lower and upper triangular matrices, the derivation obtained is based on sparse lower tridiagonal matrices and subsequent backward substitution, thereby reducing the complexity. For all time intervals I_k, the computation of the lower tridiagonal and upper triangular matrix entries costs only $\mathcal{O}(n)$ operations as it is an update of the predefined data. After computing the matrix entries, the tridiagonal matrix-vector product $\left(\prod_{r=1}^{5} \tilde{L}_{r,k}\right) \underline{g}_k$ to compute $\underline{\tilde{g}}_k$, costs $\mathcal{O}(n)$ operations due to the sparse tridiagonal nature of each $\tilde{L}_{r,k}$ matrices. Finally, the backwards substitution in Eq. (9), and hence computing the coefficients

$h_{0,k}, h_{1,k}, \cdots, h_{5,k}$, yield the cost of $\mathcal{O}(n^2)$ operations. Thus, the overall complexity of RLCA costs $\mathcal{O}(n^2)$ as opposed to the explicit brute-force method with the complexity of $\mathcal{O}(n^3)$.

By interpolating a trajectory in this manner, the dynamical system of the spacecraft is independent of the RLCA. The dynamics of the spacecraft are intrinsically considered within the boundary and interior conditions provided to the algorithm. This scalability enables the algorithm to accommodate an N-body system or any other perturbed model that the spacecraft operates within. This includes trajectories that are subjected to low thrust or perturbation forces, as those forces naturally evolve the position and velocity of the spacecraft. The position and velocity then influences the RLCA through the boundary and interior conditions. In the instance of uncertainties in the boundary and interior conditions that perturb the measured state of a spacecraft from its true state, the algorithm may behave differently. These uncertainties may arise from imperfect measurements of a spacecraft's position and velocity, causing a drift in the prediction. This effect is also present in previous work on the LCA [29], but is further exasperated by the inclusion of acceleration. Initial investigation demonstrates the RLCA is stable when subjected to uncertainties, but future work will explore how the RLCA-based piecewise-defined functions will be perturbed based on the uncertainties. Typically, the knowledge of dynamics aids in the accuracy of a trajectory generation method, placing the proposed RLCA at a disadvantage to numerical techniques that specifically incorporate dynamical models in calculations. The absence of a dynamical model causes the algorithm to become highly dependent on the boundary and interior conditions. If conditions are spread over long periods of time during which system dynamics change quickly, the algorithm's accuracy may decrease, necessitating the use of other techniques to increase fidelity. Thus, the effectiveness is practically limited by the availability of frequent spacecraft trajectory data. In a scenario in which adequate and numerous measurements are taken, the algorithm improves in accuracy and the complexity reduction may be fully taken advantage of. For example, the RLCA is a strong live target tracking algorithm in which a target may be locked onto and constantly observed. Live tracking with the RLCA produces a trajectory at sufficient accuracy through the availability of continuous tracking updates and at a low computational cost, freeing up computational resources for other onboard operations.

An inherent benefit of the RLCA is the expression of a trajectory in the form of a continuous polynomial. Numerical integration techniques require an iteration process to generate trajectories from its initial condition across a desired time interval. Conversely, the RLCA may be utilized to obtain the position of the spacecraft at any time without the need to rebuild the trajectory up to that point. Furthermore, to present an entire trajectory using numerical integration, the trajectory at every time step across an interval must be stored, which may become cumbersome. The RLCA requires only eighteen coefficients (six coefficients in each dimension) to express a trajectory, further highlighting its computational resource efficiency. Overall, the RLCA offers a low-complexity, i.e., computationally efficient, and alternate method of trajectory generation at potentially lower accuracy. The RLCA's ability to predict orbital trajectories at lower complexity enables it to be used for swift training of machine learning algorithms.

C. Implementation of RLCA into a Neural Network

The primary objective of this section is to learn, update, and predict orbital trajectories in the three-body regime using a NN and the proposed RLCA. Predicting position and velocity typically involves solving a system of differential equations using numerical integration to determine the motion of a dynamical system. Alternatively, the proposed RLCA generates trajectories faster by solving an interpolation problem using boundary and initial conditions. Here, a novel algorithm transferring RLCA into a NN to predict trajectories even when minimal data is proposed. The result is an algorithm that may be utilized for applications involving initial value problems to determine trajectories. The RLCA is first used to learn and train a NN for trajectory generation. Once the NN learns these trajectories, it is utilized to predict future trajectories within a three standard deviation (3σ) margin from the mean, at any given initial condition. As this work is a preliminary view of the RLCA integrated into an ML model, the NN is trained using a DRO. Other orbits will be studied in future work.

A NN is introduced, integrating RLCA into a ML model, for trajectory generation that learns, generates, and updates position ($\underline{r} = [x, y, z]^{\mathrm{T}}$) and velocity ($\underline{\dot{r}} = [\dot{x}, \dot{y}, \dot{z}]^{\mathrm{T}}$) at initial time t_i to the next time instance t_{i+1}. Given the non-linear nature of the CR3BP, to accurately advance the trajectory in time, non-linear transfer functions are implemented through the NN. The NN architecture is structured with fully connected nodes in each layer, ensuring that all nodes in a layer are connected to those in the subsequent layer. The input and output layers of the NN posses six nodes each, corresponding to three position and velocity states. Three hidden layers are utilized, each consisting of 30, 25, and 25 nodes respectively, that capture the non-linear mapping between input and output of the NN. In the first two hidden layers, two logsig (Sigmoid) activation functions are incorporated. In the final hidden layer, one radbas (Radial basis) activation function is incorporated.

In this paper, two ML models are trained and their learning convergence is compared. These two NN are trained using the Levenberg-Marquardt algorithm, a default optimization algorithm in MATLAB for shallow neural networks. In the Levenberg-Marquardt optimization algorithm, at each iteration, updates to the weights are computed using the Jacobian matrix, the residuals, and the damping factor [33]. Employing this adaptive learning strategy aids the model in navigating towards the global minimum, preventing confinement to local minima. The first model is based on ODE45 integrated into a ML model (Model 1) and the second model is based on the proposed RLCA integrated into a ML model (Model 2). The two ML models are trained using the following datasets:

1) Dataset 1 - Position and velocity data are obtained directly by leveraging the numerical propagation of the CR3BP for 50 trajectories using ODE45. The initial position and velocity of these trajectories are obtained randomly from a Gaussian distribution whose mean is

the true position and velocity values of the trajectory that is being predicted. The ODE45 integrator then uses these initial conditions to produce the trajectories that are utilized for training.

2) Dataset 2 - Position and velocity data are obtained by leveraging the RLCA for 50 trajectories. The initial position and velocity of these trajectories are obtained randomly from the same Gaussian distribution as before, whose mean is the true position and velocity values of the trajectory that is being predicted. Then for each orbit, 10 boundary and interior conditions (six RLCA arcs), each containing position and velocity, are provided to the RLCA. The RLCA then uses these boundary conditions to produce the trajectories that are utilized for training.

In the datasets provided above, the training process is initiated using position and velocity data only, without incorporating acceleration information. By relying solely on position and velocity data, the algorithm is able to accurately predict DRO trajectories, even with minimal predefined data. The compelling results are detailed in section IV-B, demonstrating that the proposed network successfully predicts DRO trajectories with remarkable accuracy, as depicted in Figure 10 and Figure 11. Notably, the mean squared error (MSE) of 25.457 km^2 was achieved at 50 epochs, further assuring the performance of the network in predicting DRO trajectories. In the learning process, the position and velocity data are extracted at time t_i (i.e. $[\underline{r}(t_i), \dot{\underline{r}}(t_i)]$) and set as the input of the network to learn the output position and velocity vectors at time t_{i+1} (i.e. $[\underline{r}(t_{i+1}), \dot{\underline{r}}(t_{i+1})]$), to learn the non-linear mapping between $[\underline{r}(t_i), \dot{\underline{r}}(t_i)]$ and $[\underline{r}(t_{i+1}), \dot{\underline{r}}(t_{i+1})]$. The MSE function serves as the loss function, guiding the updates of the NN weights with the Levenberg-Marquardt optimization algorithm via:

$$MSE = \frac{1}{6N}\sum_{j=0}^{N} ||\hat{\underline{r}}_j(t_{i+1}) - \underline{r}_j(t_{i+1})||^2 + ||\dot{\hat{\underline{r}}}_j(t_{i+1}) - \dot{\underline{r}}_j(t_{i+1})||^2,$$

where N is the number of samples in the dataset, $\hat{\underline{r}}_j(t_{i+1})$ and $\hat{\dot{\underline{r}}}_j(t_{i+1})$ are the predicted position and velocity respectively of the j-th sample, $\dot{\underline{r}}_j(t_{i+1})$ and $\hat{\dot{\underline{r}}}_j(t_{i+1})$ are the true position and velocity respectively for the j-th sample, and double bars denotes magnitude of a vector. The root MSE (RMSE) is defined:

$$RMSE = \sqrt{MSE}. \tag{12}$$

When comparing the testing performance of both ML models in the numerical results section IV-B, RMSE is used. Through the integration of the RLCA into a NN, the ML algorithm is now capable of solving the initial value problem, as opposed to the initial and boundary value problem in which it is formulated.

IV. Numerical Results for the RLCA and RLCA Neural Network

To analyze the accuracy of the RLCA in recreating trajectories, a high tolerance (10^{-13}), fifth-order Runge-Kutta integrator (ODE45) is used to create the reference periodic trajectory that the RLCA is interpolating. This method is selected to produce the reference trajectory as it is an accurate and well-accepted method of propagating motion in the CR3BP. The proposed RLCA neural network is also compared with a ML model based on a high tolerance ODE45 integrator. Numerical integrators are typically more precise [34] than the RLCA, prompting an accuracy analysis focused exclusively on the RLCA. Otherwise, the accuracy analysis on the numerical integrator will be on a more precise scale and not offer much insight into the RLCA itself. The RLCA simulates boundary and interior conditions (position and velocity) as if they are measurements from a spacecraft's reference trajectory. In practical applications, real data may be inserted. An assortment of periodic trajectories are analyzed using the RLCA. The trajectories are selected such that, through each example, the geometry and dynamics generally become more challenging as to fully flesh out the capabilities of the RLCA. The initial conditions of these trajectories are obtained through differential corrections and continuation schemes in the CR3BP that search for families of periodic trajectories [6]. For precise identification of an orbit in its respective family, the Jacobi constant for each orbit is identified. It is assumed the measurements of the reference trajectory contain no noise, the reference trajectory follows the dynamics associated with the CR3BP, and a measurement of the spacecraft's state is always available to the RLCA. For this initial analysis, a trajectory for a single orbital period is analyzed. This trajectory is broken into five continuous arcs, equally spaced in time. The RLCA is executed for these five arcs separately, pulling in the boundary and interior condition of the arc. In this analysis, the boundaries are shared between two arcs, and the interior condition is taken at the midpoint of a respective arc. Following this process, the five arcs require ten measurements to be predicted using the RLCA, as the trajectory is continuous and a closed loop.

A. Numerical Results for the Orbital Trajectories using RLCA

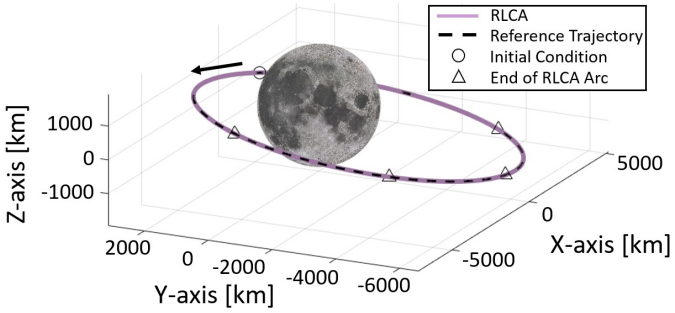
To begin the analysis, a simple low-lunar 2BP elliptical orbit is studied to acquire a gauge of the algorithm's performance in a simpler dynamical system than the CR3BP. This example also highlights the algorithm's capabilities to be applied across dynamical models. Next, three CR3BP trajectories are selected to test the RLCA: an L_2 Lyapunov orbit, an L_4 axial orbit, and a near-rectilinear halo orbit (NRHO). Note that each of the simulated orbits are sampled from a respective orbit family in the CR3BP [6]. The L_2 Lyapunov is selected to demonstrate the algorithm's ability to reproduce a simple planar trajectory in the CR3BP, while the L_4 axial orbit is chosen to test the algorithm's capabilities to generate a unique 3D trajectory. Finally, the NRHO is selected due to its relevance to Cislunar space as well as its extreme dynamical environment around perilune. The NRHO is a pivotal trajectory in Cislunar space

as it is the current location of the Capstone orbiter and is the selected orbit for the long-term Lunar station, Gateway [35].

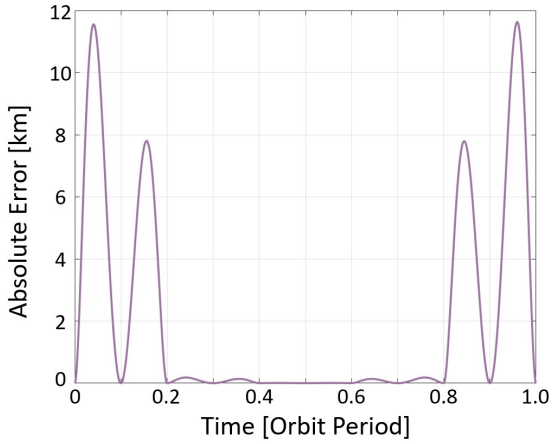
The algorithm requires that first, the position and velocity measurements of a trajectory are acquired. For an initial analysis, a trajectory is broken into five distinct arcs that form the complete periodic trajectory. Recall, that each arc requires three measurements to be defined by RLCA. Therefore, for five arcs whose boundaries are continuous, a total of ten unique measurements are needed. These ten measurements are spaced equally through time on a tested trajectory. The proposed algorithm is executed in dimensionless form and repeats the process for each dimension at time intervals I_k to produce a complete 3D trajectory. With the polynomials of an arc solved for, the trajectory of the arc is computed and compared to the reference trajectory. Lastly, the computation time of Runge-Kutta methods, Adams-Bashforth method, and the proposed algorithm is then compared.

First, the low-lunar 2BP elliptical orbit (LLO) is simulated as it offers a simple geometry and dynamical model for an initial test. The resulting orbit is shown in Fig. 3a, with the associated error between ODE45 and the algorithm represented in Fig. 3b. In Fig. 3a, and subsequent similar figures, the triangles indicate an RLCA arc boundary, or how the trajectory is broken into the five distinct arcs. Furthermore, the circle marks the orbit's starting point and the division between initial and final RLCA arcs, the solid colored line depicts the RLCA propagated trajectory, the black dotted line represents the ODE45 propagated trajectory, and the solid black arrow signifies the orbit's direction. In this example trajectory, the algorithm is capable of reproducing the trajectory to a decent extent as the ODE45 and RLCA paths are difficult to distinguish from one another in Fig. 3a. Thus, for a refined comparison between the reference and RLCA trajectory, Fig. 3b is introduced. In Fig. 3b, the error is defined as the magnitude of the difference between the ODE45 and RLCA propagated trajectory at a given instance in time. Through this comparison, it is seen that the maximum error demonstrated is about 12 km. Notably, the RLCA sections that are near perilune possess higher errors. However, the error being quite below half a kilometer for significant portions of the orbit shows promising results for the accuracy of the algorithm.

Moving onto the CR3BP, the L_2 Lyapunov orbit is simulated first as it offers planar motion with a unique geometry. The resulting orbit is shown in Fig. 4a, with the associated error

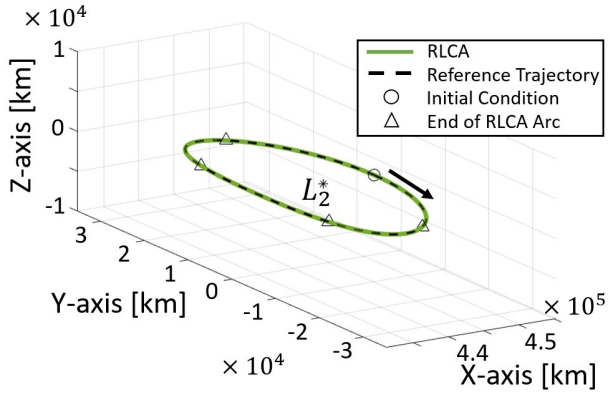


(a) ODE45 and RLCA trajectory for a Lunar orbit (LLO).

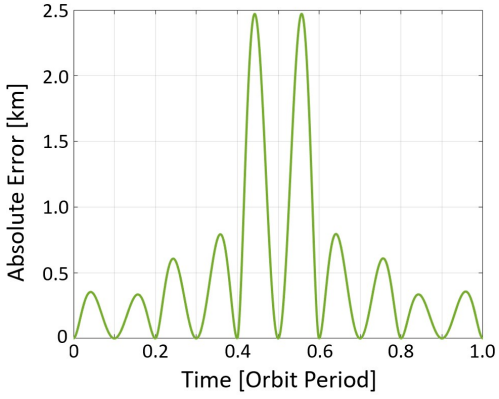


(b) Error between ODE45 and RLCA throughout 1 period.

Fig. 3: Accuracy analysis of RLCA on a sample Lunar elliptical orbit with a period of 8.8 hours.



(a) ODE45 and RLCA trajectory of an L_2 Lyapunov orbit.



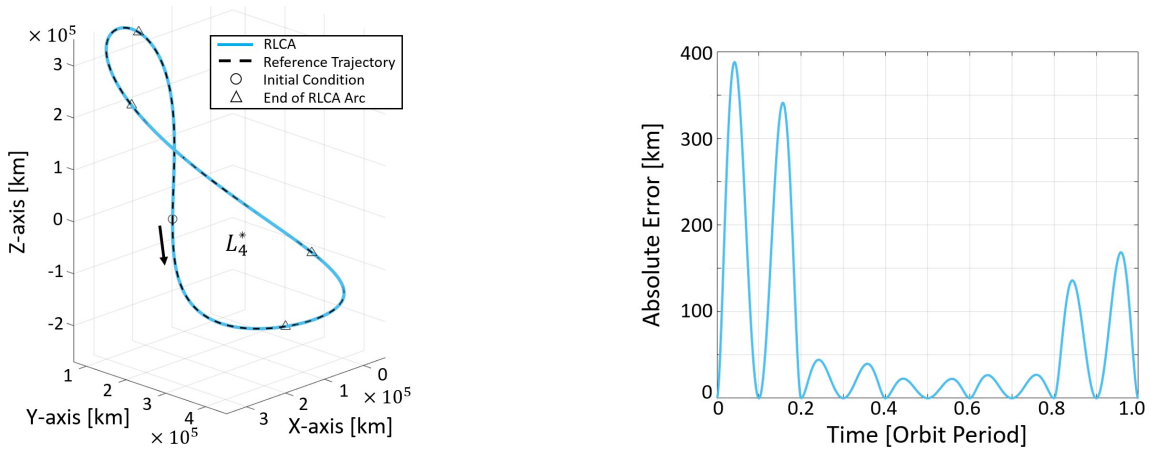
(b) Error between ODE45 and RLCA throughout 1 period.

Fig. 4: Accuracy analysis of RLCA on a sample L_2 Lyapunov orbit with a period of 14.7 days (JC = 3.1622).

between ODE45 and the algorithm represented in Fig. 4b. This analysis further supports the accuracy of the RLCA to reproduce the reference trajectory, as the maximum error is about 2.5 km. This is a reasonable result for an orbit that spans around $40,000$ km in the y-direction and $20,000$ km in the x-direction. In Fig. 4b, a key property of the RLCA algorithm is demonstrated: the utilized measurements are always satisfied by the polynomial, denoted by the instances in which the error between RLCA and ODE45 is null. This causes slow deviations in the RLCA from the reference with time before converging back to the next measurement since the following condition must be satisfied. The magnitude of deviation from the reference depends on the corresponding portion of the trajectory and is investigated later. The symmetry in the error plot stems from the L_2 Lyapunov orbit's symmetry, symmetrical initial conditions, and evenly spaced time measurements. The analysis highlights the RLCA's accuracy for the L_2 Lyapunov orbit, though it represents just a minor planar orbit example. This example also confirms an important trait of the proposed algorithm in that it may operate across dynamical models, as shown by successful trajectory reproduction in the 2BP and CR3BP without changing the algorithm itself.

Next, the L_4 axial orbit is simulated to test the algorithm's ability to recreate a large complex orbit with significant out-of-plane motion. Using the described simulation process, the orbit is split into five distinct arcs and the RLCA is simulated. The resulting orbit, as well as the reference trajectory propagated using ODE45, is presented in Fig. 5a. Figure 5b represents the error between these two trajectories. In comparison to the reference trajectory, the RLCA does possess areas of higher error; in particular, the first arc of the trajectory spanning the first fifth of the orbit. However, when this large error is placed in perspective compared to the vast size of the L_4 axial orbit, the error becomes less significant. The L_4 axial orbit as a whole spans about $500,000$ km in the z-direction, with the first arc spanning $200,000$ km on its own. All measurements are spaced equally in time across the period of a respective orbit, but since the period of the L_4 axial orbit is much larger than the previous L_2 Lyapunov example, the time between measurements is also higher. This increase in time between measurements is another factor weighing into the higher error results, allowing a larger time span for the RLCA to deviate before needing to converge back to satisfy the measurement. Additionally, the axial orbit is traversing a significant portion of space, requiring instances of rapid change. The RLCA interpolates from existing measurements based on boundary and interior conditions to predict trajectories, but its accuracy declines in rapidly changing orbit segments without fresh data to update the polynomials. Consequently, the unique motion of the L_4 axial orbit impacts the algorithm's precision without adequate measurements. Regardless, for the remaining section of the trajectory, the error is reasonable and shows potential in the RLCA's ability to reproduce complex trajectories.

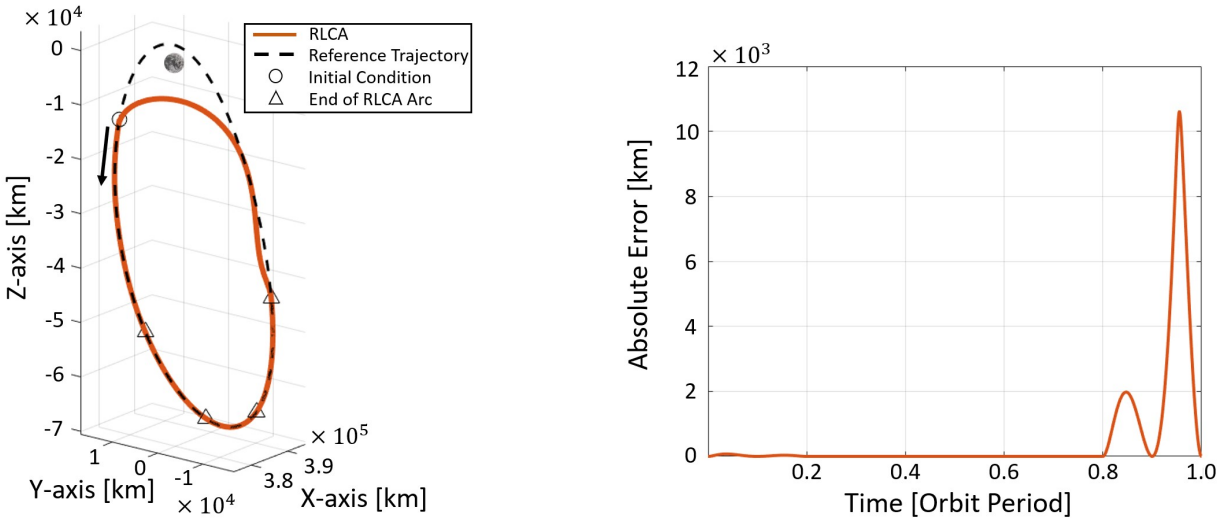
Finally, the NRHO is simulated using the same method as the previous trajectories, yet it presents a significant challenge to the RLCA. For the first time out of the simulated trajectories, the RLCA is clearly visible compared to the ODE45 trajectory (Fig. 6a). The first four arcs of the trajectory are accurate, but the arcs close to perilune struggle significantly to match the ODE45 results as the error stretches above $10,000$ km (Fig. 6b). The significant error is a direct result of the NRHO trajectory lacking measurements for the RLCA during points of rapid change, in particular the NRHO's perilune. The NRHO's perilune is a dynamically extreme portion of the orbit as the trajectory undergoes rapid acceleration passing close to the Moon. In this simulation, the perilune is measured right before and after the passage of the Moon. These measurements are points on the orbit that are approaching the region of rapid change near perilune. Thus, the interpolated RLCA trajectory does not accurately capture the perilune of the NRHO, as the measurements are not adequate to portray the behavior of the entire portion of the orbit. The resulting error quickly deviates as the mildly changing RLCA trajectory poorly presents the extreme reference trajectory. This behavior corresponds to the L_4 axial trajectory in that the sections of an orbit undergoing rapid change cause significant deviation of the RLCA from the reference. A remedy to the significant error is elaborated upon in further sections.



(a) ODE45 and RLCA trajectory for an L_4 axial orbit.

(b) Error between ODE45 and RLCA throughout 1 period.

Fig. 5: Accuracy analysis of RLCA on a sample L_4 axial orbit with a period of 27.3 days (JC = 2.0941).



(a) ODE45 and RLCA trajectory for a NRHO.

(b) Error between ODE45 and RLCA throughout 1 period.

Fig. 6: Accuracy analysis of RLCA on a sample NRHO with a period of 6.6 days (JC = 3.0455).

The motivation for the creation of the proposed RLCA is to reduce the computational time required for methods for trajectory determination. The arithmetic complexity of the RCLA is $\mathcal{O}(n^2)$, which is typically proportional to the computational time of the algorithm. To get a full gauge of the time complexity of the algorithm, an analysis measuring the computational time is completed. The computational time of an algorithm depends deeply on the computational system itself and the methodology of code writing. Therefore, steps are taken to ensure a fair comparison of computation time between numerical integrators and the RLCA. First, the only numerical integrators used are built-in MATLAB functions such that they consistently follow the same standards set by MATLAB [36]. This fact limits possible variations in computation time from different methodologies of code writing as they are held to the same standard. Next, the number of time steps in which the numerical integrator iterates through directly increases or decreases the computation time. Thus the numerical integrators are allowed to shift step size using MATLAB's built in adaptive step size control. This will allow the numerical integrators to determine the number of iterations through a consistent process without an artificially inflated or deflated number of steps. As a final precaution, simulations are conducted on identical systems and averaged over 1,000 iterations of each algorithm to minimize the impact of potential outlier computation times.

The computation time of four algorithms will be compared: the RLCA, Adams-Bashfourth (ODE113), fourth-order Runge-Kutta (ODE45), and a seventh-order Runge-Kutta (ODE78). The time computation of the RLCA is computed for the analysis seen above. Specifically, the time it takes the algorithm to output the polynomial coefficients needed to describe a trajectory that has been broken into five arcs. For numerical integrators, the chosen tolerance significantly impacts their convergence time. In models like the 2BP, a default tolerance of 10^{-8} generally suffices. However, for the more numerically sensitive CR3BP, a tighter tolerance of 10^{-13} is used. Thus, this analysis incorporates an example with the fourth-order Runge-Kutta numerical integrator at the stricter tolerance for CR3BP periodic orbits. The computation time analysis, averaged over 1,000 runs, is provided in Fig. 7. Figure 7 shows that the algorithm is consistently faster than the analyzed numerical integration techniques.

In this simulation, the RLCA performs 94.8% faster in LLO case against ODE78, 80.1% faster in L_1 Lyapunov case against ODE78, 87.1% faster in L_4 Axial case against ODE78, and 92.0% faster in NRHO case against the lower tolerance ODE45. Further summary on the improvement in computation time the RLCA provides over the numerical methods is shown in Table I. The RLCA's efficiency derives from its ability to quickly generate a complete trajectory polynomial, accessible at any desired time step, using just eighteen coefficients for a three-dimensional trajectory from three measurements. This contrasts with numerical methods that require iterative integration to a set tolerance and step-by-step calculations for

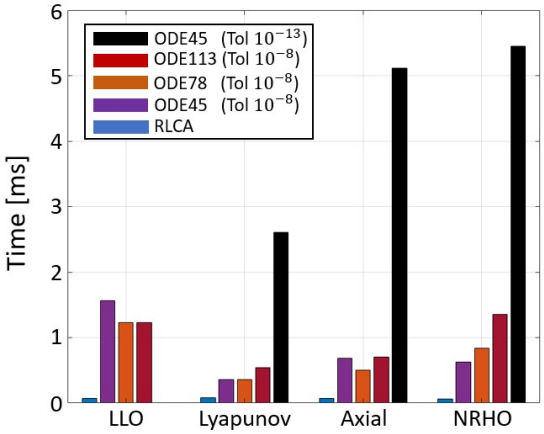


Fig. 7: Computational time of numerical integration methods and the RLCA averaged over 1000 runs (System: Intel i7-12700H 2.3 GHz, 16.0 GB memory).

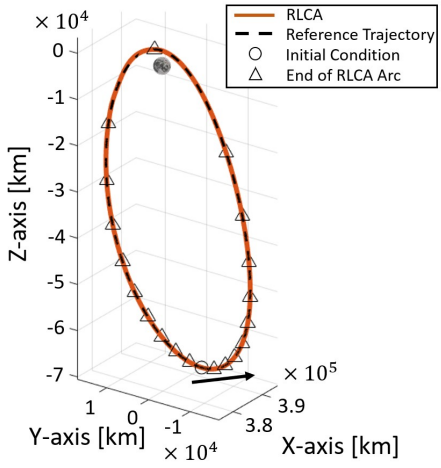
TABLE I: Comparison of the time complexities of the proposed RLCA and other trajectory generation algorithms.

Orbit	RLCA Time [ms]	% Decrease in RLCA Computation Time over Respective Numerical Method			
		ODE45 (Tol 10^{-8})	ODE78 (Tol 10^{-8})	ODE113 (Tol 10^{-8})	ODE45 (Tol 10^{-13})
LLO	0.06256	96.1 %	94.8 %	94.9 %	-
L_1 Lyapunov	0.06050	82.3 %	80.1 %	87.1 %	97.7 %
L_4 Axial	0.06001	91.7 %	87.1 %	91.6 %	99.0 %
NRHO	0.06005	92.0 %	92.4 %	95.6 %	99.0 %

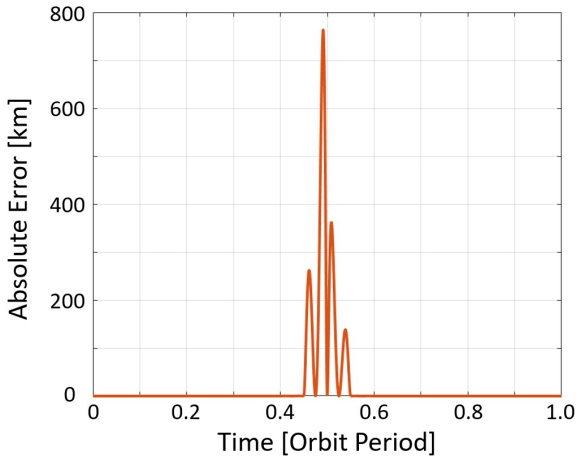
specific times. However, while the RLCA improves speed, it sometimes sacrifices accuracy, as demonstrated in trajectory recreations. Finally, it is important to note that the scaling of this analysis is at milliseconds. Although this appears as a minuscule amount of time, the speed improvements relative to the scale of the problem is significant. The scale of numerical time saved will further alter depending on the system or computational resources available, but the relationship showing improved computational capabilities will persist in a similar simulation.

To reduce error in the NRHO, it is crucial to optimize the RLCA's dependency on accurate boundary and interior conditions. The RLCA's accuracy hinges on the frequency and placement of measurements; while specific locations are often limited by factors like observer location and visibility [37], increasing the measurement frequency is more practical. More frequent trajectory segments for interpolation enhance the RLCA's precision, especially effective in rapidly changing orbits, as demonstrated by improved results at the NRHO's perilune through denser sampling. To simulate this, the NRHO is now broken into 20 arcs spaced apart equally in time (Fig. 8a). Along with the increase in the total number of arcs, the initial condition of the orbit is shifted in order to adequately sample the NRHO's perilune. Previously this section was described by a single arc and is now described by three arcs. In the 20 arc simulation, the perilune of the NRHO occurs around the middle of the orbital period and demonstrates a maximum error of below 800 km (Fig. 8b). An 800 km error is a significant improvement from the single arc perilune simulation, where the error is found to be $10,000$ km. The remainder of the 20 arc simulation is found to have an error of less than 10 km. Focusing on the perilune of the NRHO, the addition of two new arcs requires only 4 more measurements and drops the error by thousands of kilometers. Further improvement of sampling frequency and location in the perilune of the NRHO may further bring down the error across perilune, but this analysis demonstrates the significant reduction in error with only a few additional measurements

This NRHO case is further explored by analyzing the relationship between number of RLCA arcs in a trajectory and the computation time and accuracy. This is completed by breaking the NRHO into a set number of arcs, and calculating the average computation time of the RLCA over 1,000 iterations, in the same manner as before. Furthermore, the average maximum absolute error found across the trajectory over 1,000 iterations is also analyzed. This process is then repeated for a range of RLCA arcs the NRHO is broken down into. The resulting average computation time and average maximum absolute error for each simulation is shown in Fig. 9. Generally, the algorithm takes a longer time for increasing the number of arcs, as the algorithm runs a greater number of times as a whole. The analysis highlights the trade off between accuracy and computation time in the RLCA. As the RLCA is given more measurements to break a trajectory down, it will become more accurate, but it must generate more polynomials to completely describe the trajectory and thus takes longer to run. Through denser sampling of the NRHO across perilune, a more accurate representation of the orbit is produced by the RLCA. Consequently, it is generally concluded that denser sampling in areas of rapid change yields more precise results.



(a) ODE45 and RLCA trajectory for an NRHO using 20 RLCA arcs.



(b) Error between ODE45 and RLCA trajectory throughout 1 period using 20 RLCA arcs.

Fig. 8: Accuracy analysis of RLCA on a sample NRHO broken into 20 arcs (Period = 6.6 days).

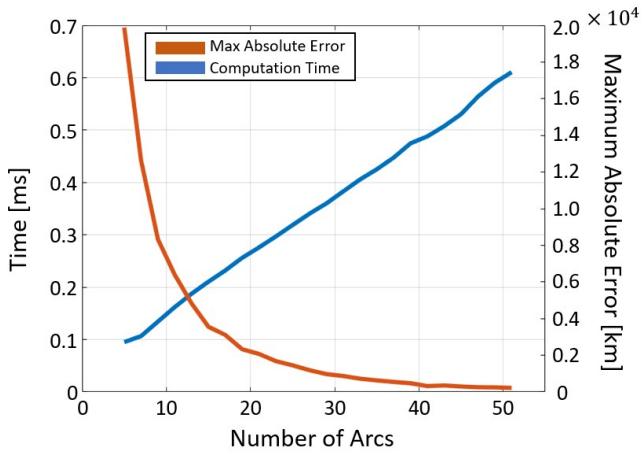


Fig. 9: Average computation time and maximum absolute error of the RLCA to reproduce an NRHO, averaged over $1,000$ runs, dependent on number of arcs (System: Intel i7-12700H 2.3 GHz, 16.0 GB memory).

General conclusions on the RLCA's limitations may be drawn from the analysis of the NRHO case. For example, the RLCA struggls to accurately predict regions undergoing rapid dynamical change. This outlines a limitation of the RLCA in its ability to sufficiently predict dynamically sensitive regions when the region is not properly sampled. Therefore when attempting to predict these regions, extra attention to the sampling of boundary and interior conditions is required. Another limitation that presents itself is the increase in computation time that arises when a trajectory is discretized into a large number of arcs. Since the algorithm runs for each arc, continuously increasing the number of arcs will linearly increase computation time, eventually causing the RLCA to take longer to run than a numerical method. However, this is unlikely to occur as it requires an unnecessary number of RLCA arcs for a given trajectory. Overall, the RLCA's performance faces limitations in its accuracy of predicting trajectories in extremely sensitive dynamical regions and limitations in its computation time for trajectories that are overly discretized. These short-comings present themselves in specific cases, and proper sampling of boundary and interior conditions shall help remedy such limitations.

B. Numerical Results of the RLCA Neural Network

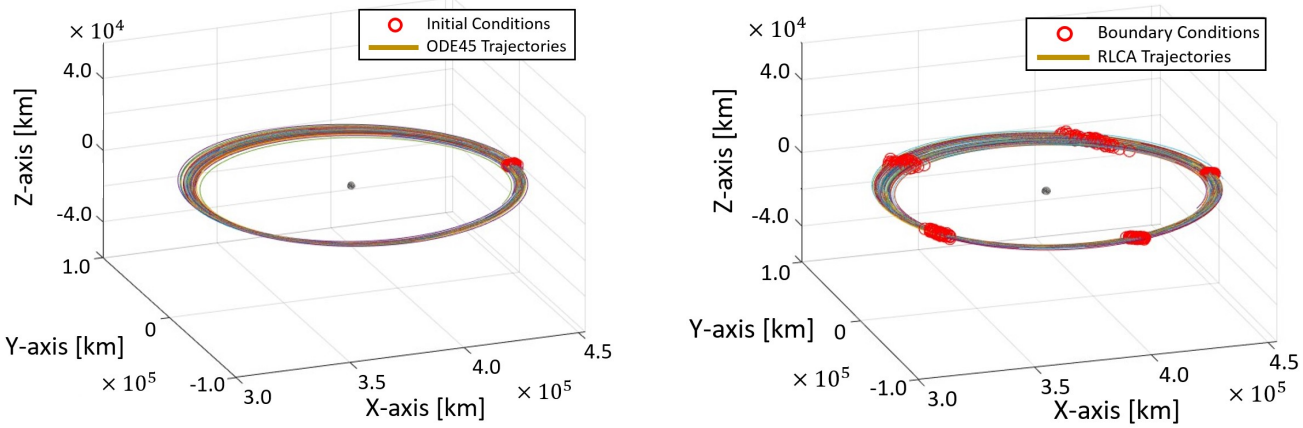
Now that the explicit use of the RLCA to predict trajectories is investigated, the RLCA is embedded into an ML model to generate DRO trajectories. The planar DRO family covers extensive areas of Cislunar space, offers favorable stability properties, and features geometrically simple motion. These traits make it an ideal candidate family to initially test on. The JC, initial condition ($[\underline{r}_0, \underline{\dot{r}}_0]$), and the standard deviation of the Gaussian distribution for position and velocity are provided:

$$\begin{aligned} JC &= 2.9339, \\ [\underline{r}_0,\, \underline{\dot{r}}_0] &= [1.17,\ 0,\ 0,\ 0,\ -0.489780292125578,\ 0], \\ \sigma &= [2.6042,\ 2.6042,\ 0,\ 0,\ 1.3021,\ 0] \cdot 10^{-3} \end{aligned} \quad (13)$$

where all presented values are non-dimensional. This initial condition for a DRO trajectory is found via differential correction methods in the CR3BP [6]. Utilizing the conditions stated in Eq. 13, the two datasets for training Model 1 and Model 2 are generated following the methods described in section III-C. The generated datasets, shown in Fig. 10 and denoted as Dataset 1 and Dataset 2, comprise $16,000$ location coordinates to represent 50 trajectories (each with 320-time steps), spanning the 13.91 day period of the reference DRO.

A split for each dataset is completed, allocating 70% for training, 15% for testing, and 15% for validation. The two models are trained using the datasets provided, an initial damping factor of 0.001 in the Levenberg-Marquardt optimization algorithm, and the NN architecture is described in section III-C. The weights are initialized between all layers using the Nguyen-Widrow layer initialization function in MATLAB. The MSE of the NN predictions on the training set, validation set, and testing set for both ML models are illustrated in Fig. 11. Specifically, the first model reaches an MSE of 28.555 km^2 at 200 epochs (iterations), while the second model reaches an MSE of 25.457 km^2 at 50 epochs (iterations). The learned and updated results indicate that the ML model trained on the dataset derived from the RLCA converges faster than that of the ODE45. Furthermore, the MSE convergence of the ML model is stable, exhibiting a consistent decrease without fluctuations. This difference can be attributed to the inherent strength of neural networks, which lies in their ability to effectively learn, update, and represent highly non-linear relationships of CR3BP [38]. This analysis further demonstrates the computational efficiency of the RLCA as seen by the fast convergence of the RLCA NN to learn and update trajectories.

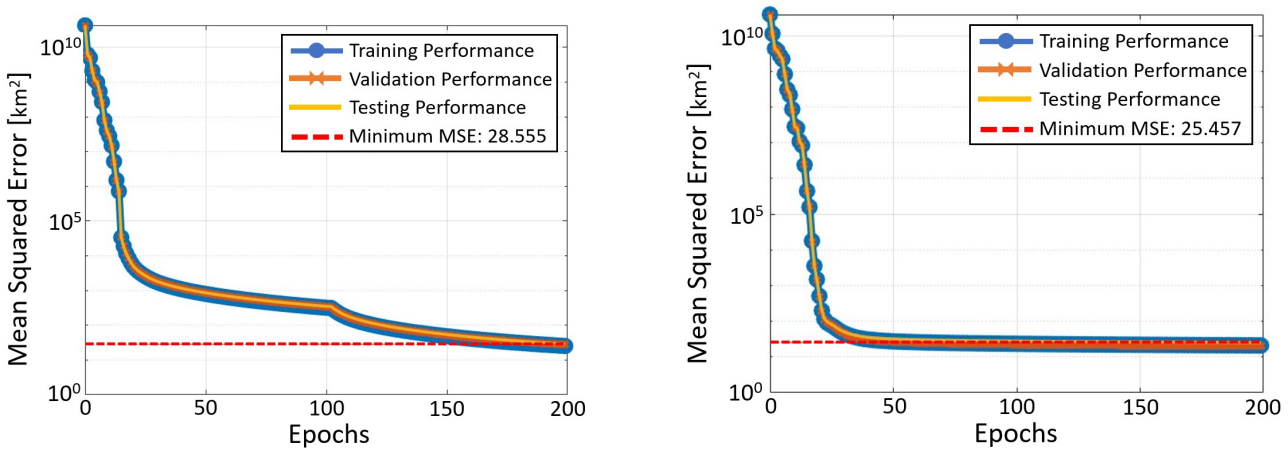
During the testing phase, i.e., after the neural network is trained, a new DRO trajectory is generated starting from an arbitrary initial condition that is close to the trained data. This random initial condition is generated from within the same Gaussian distributions that are used for generating the Datasets. In DRO trajectory prediction, the output of the current state is subsequently fed as the input to predict the next state trajectories, i.e. moving from t_i to t_{i+1}. The NN's predicted results indicate slight deviations in the DRO trajectory prediction compared to the ODE45 embedded ML model. To verify the long-term deviations, the three DRO trajectories are propagated throughout 10 orbital periods, i.e., 139.1 days. The trajectories are plotted (Fig. 12) while considering the RMSE, which is calculated as the error between the RLCA-based and the ODE45-based ML models. This analysis is presented to provide a clearer understanding of the error of both NN models. Note here that ODE45 gives an approximate solution for orbital propagation that accumulates error over time, thus using it for long periods as a reference for an ML model causes errors and deviations. Therefore, during the testing phase, the performance of each model is assessed based on the RMSE of position data. As shown in Fig. 12, on average, the prediction error of NN trained on the RLCA dataset is close to the prediction error of NN trained on the ODE45 dataset up to two orbital periods. In the first two orbital periods, the two models are similar to one another. Then, the RLCA ML Model possesses a higher error than the ODE45 ML Model.



(a) Dataset 1 (High tolerance, ODE45 propagator).

(b) Dataset 2 (RLCA).

Fig. 10: Dataset 1 and Dataset 2 are used for the learning and updating of ML Model 1 and Model 2 respectively, following the process described in section III-C and using the conditions in Eq. 13.

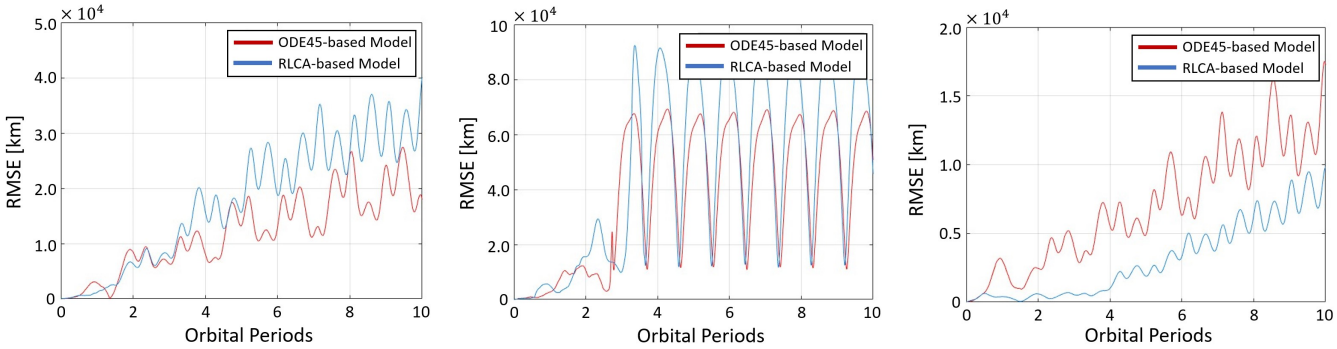


(a) Model 1 learning and updating process for the DROs based on Dataset 1 (embedding ODE45 into ML model).

(b) Model 2 learning and updating process for the DROs based on Dataset 2 (embedding RLCA into ML model).

Fig. 11: Training and testing performance of DRO trajectories for the described Models.

Even small disturbances to a spacecraft's orbital path may have long-term effects, such as after 4 months. Furthermore, it is observed that while predicting the position and velocity for more than one orbital period using only one initial condition, the error of the NN of RLCA becomes significantly higher than the NN of ODE45. More importantly, ODE45 gives an approximate solution for orbital propagation that diminishes with time, and using it as a reference for an ML model shows deviations in error at these later points in time. Therefore, after several orbital period propagations, the NN of ODE45 gives a lower error compared to the NN of RLCA. In Fig. 12, the true reference DRO, as well as two random DROs with perturbed initial conditions, are propagated significantly past a single orbital period using both ML Models, and the propagation error is presented. *Notably, if the spacecraft is propagated precisely on the actual DRO, it perfectly aligns with the RLCA dataset, yielding an error comparable to the trained ODE45 NN but with faster training convergence.*. This suggests that the proposed NN method is more suitable for generating trajectories near true DRO throughout two orbital periods. In future work, ML models will be proposed to generate positions and velocities with low RMSE per multiple periods. Furthermore, in future development, it is considered that these periodic trajectories are part of a continuous family in the CR3BP that extend from one another. A small perturbation in initial conditions may place these trajectories in a slightly different orbit that the NN must account for. Due to the nonlinear behavior of the CR3BP, this slight perturbation may result in a significantly different orbit, or a very similar one. The DRO selected for this analysis is extremely similar to its nearby counterparts, and thus this scenario is not a huge concern in this analysis, but for other trajectories, this may not be the case and must be considered. The difference between the ODE45-based NN and the predicted RLCA NN is not quite significant compared to the $100,000$ km the DRO stretches. This shows the high accuracy of the DRO trajectory prediction through an ML model except at 20% of the DRO period.



(a) A DRO with random perturbed initial conditions, falling in the standards of Eq. 13 and close to mean (true) DRO.

(b) A second DRO with random perturbed initial conditions, falling in the standards of Eq. 13 and close to mean (true) DRO.

(c) The mean (true) DRO.

Fig. 12: The RMSE variation of NN predictions across extended periods, for randomly perturbed DROs and the mean DRO.

V. Conclusion

Trajectory design and prediction in the complex three-body dynamics of the Earth-Moon system requires computationally inexpensive methods. Numerical integration techniques like Gauss-Legendre, Dormand-Prince, and Adams-Bashforth algorithms are commonly used to solve trajectory generation as an initial value problem by integrating differential equations that describe system dynamics. However, these methods tend to be computationally expensive. In this work, the RLCA approaches the trajectory generation problem as a boundary value problem, in which a polynomial describing spacecraft trajectory is determined using boundary and interior conditions. Using the boundary and interior conditions, the algorithm formulates a unique system where the coefficient matrix is decomposed into tridiagonal matrices and an upper triangular matrix. By doing so, polynomial trajectories are computed with $\mathcal{O}(n^2)$ arithmetic complexity, offering an improvement over the brute force calculation with $\mathcal{O}(n^3)$ complexity. The RLCA further improves upon the authors' previous work [29] by only requiring position and velocity at the boundary and interior conditions as opposed to needing position, velocity, and acceleration [29]. By removing the need for acceleration, the algorithm is capable of operating on conditions that are often available through observation techniques (i.e., position and velocity). The RLCA is demonstrated across an assortment of orbital trajectories. In the CR3BP, the algorithm is used to recreate an L_2 Lyapunov orbit, L_4 axial orbit, and NRHO by breaking each orbit into five continuous arcs. In addition to the CR3BP simulations, a low-Lunar elliptical orbit in the 2BP is presented to show the algorithm's capability to scale to any dynamical framework. The RLCA recreated the CR3BP trajectories with reasonable accuracy, except in the case of the NRHO in which more measurements were required to reach an acceptable accuracy. The NRHO case highlighted the fact that poor measurements around rapidly changing portions of trajectories diminish the algorithm's ability to produce accurate results and by increasing measurements in these regions, accuracy is improved. In comparison to the numerical methods shown, the RLCA showed significant improvement in computation time, beating out the numerical methods in the five arc simulation by at least 70%. Overall, the computational efficiency and accuracy of the algorithm demonstrate the potential of the algorithm to be used as an alternative low-complexity tool to supplement computationally expensive numerical techniques in certain scenarios.

In traditional trajectory generation algorithms, solving the set of differential equations with a specific initial condition puts the RLCA at a disadvantage in its current form. Thus, the RLCA is embedded into an ML model and predicted DRO trajectories at given initial conditions. Given the non-linear nature of the CR3BP, the NN architecture is designed with six nodes in the input and output layers and three hidden layers. These hidden layers operate with different activation functions and quantities of nodes. This NN architecture is trained and tested on a DRO. Two NN are trained, one with DRO trajectories using ODE45 and another with the RLCA. The training of these NN found the RLCA Model converged faster and smoother with less MSE than the ODE45 model. Furthermore, the RLCA NN predicted the DRO with significantly less error than the ODE45-based NN in a single orbital period. Alternatively, outside an orbital period, the RLCA-based NN shows a higher error compared with ODE45-based NN. Note here that ODE45 gives an approximate solution that deviates over significant periods for orbital propagation, and using it as a reference for an ML model shows deviations in error. Using the exact initial conditions, the RLCA-based NN accurately predicts future DRO trajectories better than an ODE-based NN, resulting in less RMSE. Future work aims to enhance the RLCA-based NN across multiple periods and orbital trajectories.

References

[1] B. Baker-McEvilly, S. Doroba, A. Gilliam, F. Criscola, D. Canales, C. Frueh, and T. Henderson, "A review on hot-spot areas within the cislunar region and upon the moon surface, and methods to gather passive information from these regions," in *AAS/AIAA 33rd Space Flight Mechanics Meeting*, 2023.

[2] B. Baker-McEvilly, S. Bhadauria, D. Canales, and C. Frueh, "A comprehensive review on cislunar expansion and space domain awareness," *Progress in Aerospace Sciences*, vol. 147, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042124000459

[3] K. Johnson, "Fly me to the moon, worldwide cislunar and lunar missions," *CSIS Aerospace Security Project*, 2022.

[4] P. Niles, "Summary of the contracted deliveries of nasa payloads to the moon via commercial lunar payload services," *53rd Lunar and Planetary Science Conference*, 2022.

[5] Y. Song, Y. Kim, J. Bae, J. Park, S. Hong, D. Lee, and D. Kim, "Overview of the flight dynamics subsystem for korea pathfinder lunar orbiter mission," in *Aerospace*, vol. 8, 2021.

[6] D. Grebow, "Generating periodic orbits in the circular restricted three-body proglem with application to lunar south pole coverage," Master's thesis, Purdue University, 2006.

[7] H. B. Keller, *Numerical Solution of Two Point Boundary Value Problems*. Philadelphia: SIAM, 1976.

[8] S. M. Roberts and J. S. Shipman, "Continuation in shooting methods for two-point boundary value problems," *Journal of Mathematical Analysis and Applications*, vol. 18, no. 1, pp. 45–58, 1967.

[9] T. Pavlak and K. Howell, "Strategy for optimal, long-term stationkeeping of libration point orbits in the Earth-Moon system," *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2012.

[10] J. M. Aristoff, J. T. Horwood, and A. B. Poore, "Orbit and uncertainty propagation: a comparison of Gauss-Legendre-, Dormand-Prince-, and Chebyshev-Picard-based approaches," *Celestial Mechanics and Dynamical Astronomy*, vol. 118, no. 1, pp. 13–28, Jan. 2014.

[11] P. Deuflhard, "Order and stepsize control in extrapolation methods," *Numerische Mathematik*, vol. 41, no. 3, pp. 399–422, 1983. [Online]. Available: https://doi.org/10.1007/BF01418332

[12] E. Suli and D. F. Mayers, *An Introduction to Numerical Analysis*. Cambridge: Cambridge University Press, 2003.

[13] M. Horemuz and J. Andersson, "Polynomial interpolation of gps satellite coordinates," in *GPS Solutions*, vol. 10, 2006, pp. 67–72.

[14] E. Karepova and V. Petrakova, "The comparison of several approaches to the interpolation of a trajectory of a navigation satellite," *IOP Conference Series: Materials Science and Engineering*, vol. 537, 06 2019.

[15] J. Geul, E. Mooij, and R. Noomen, "Regularised methods for high-efficiency propagation," in *AAS Astrodynamics Specialist Conference*, 2015.

[16] N. Parrish, J. Parker, S. Hughes, and J. Heiligers, "Low-thrust transfers from distant retrograde orbits to l2 halo orbits in the earth-moon system," in *6th International Conference on Astrodynamics Tools and Techniques*, 03 2016.

[17] A. Osama, M. Raafat, A. Darwish, S. Abdelghafar, and A. E. Hassanien, "Satellite orbit prediction based on recurrent neural network using two line elements," in *2022 5th International Conference on Computing and Informatics (ICCI)*, 2022, pp. 298–302.

[18] H. Peng and X. Bai, "Machine learning approach to improve satellite orbit prediction accuracy using publicly available data," *The Journal of the astronautical sciences*, vol. 67, no. 2, pp. 762–793, 2020.

[19] ——, "Improving orbit prediction accuracy through supervised machine learning," *Advances in Space Research*, vol. 61, no. 10, pp. 2628–2646, 2018.

[20] I. Pérez, J. F. San-Juan, M. San-Martín, L. M. López-Ochoa *et al.*, "Application of computational intelligence in order to develop hybrid orbit propagation methods," *Mathematical Problems in Engineering*, 2013.

[21] H. Shou, "Orbit propagation and determination of low earth orbit satellites," *International Journal of Antennas and Propagation*, 2014.

[22] H. Peng and X. Bai, "Comparative evaluation of three machine learning algorithms on improving orbit prediction accuracy," *Astrodynamics*, vol. 3, pp. 325–343, 2019.

[23] X. Zhou, T. Qin, M. Ji, and D. Qiao, "A LSTM Assisted Orbit Determination Algorithm for Spacecraft Executing Continuous Maneuver," *Acta Astronautica*, vol. 204, pp. 568–582, 2023.

[24] P. G. Breen, C. N. Foley, T. Boekholt, and S. P. Zwart, "Newton versus the machine: solving the chaotic three-body problem using deep neural networks," *Monthly Notices of the Royal Astronomical Society*, vol. 494, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1093/mnras/staa713

[25] L. Mi, "Solving the three-body problem using numerical simulations and neural networks," in *2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, 2021, pp. 338–341.

[26] Y. Neiman and L. Sugaipova, "GNSS orbit approximation by means of artificial neural networks." *Geodesy and Cartography*, vol. 85, no. 5, pp. 11–23, 2024.

[27] D. Canales, "Transfer design methodology between neighborhoods of planetary moons in the circular restricted three-body problem," *Purdue University*, Dec. 2021.

[28] M. Gupta, "Finding order in chaos: Resonant orbits and poincare sections," *Purdue University*, May 2020.

[29] D. Canales, S. M. Perera, A. Kurttisi, and B. Baker-McEvilly, "A low-complexity algorithm to determine trajectories within the circular restricted three-body problem," *Journal of the Astronautical Sciences*, vol. 70, no. 46, 2023.

[30] H. Poincaré, *Les méthodes nouvelles de la mécanique céleste*. Paris, France: Gauthier-Villars et Fils, 1892.

[31] V. Szebehely, *The General and Restricted Problems of Three Bodies*. Vienna: Springer, 1974.

[32] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Philadelphia, USA: SIAM, 1996.

[33] A. Ranganathan, "The levenberg-marquardt algorithm," *Tutoral on LM algorithm*, vol. 11, no. 1, pp. 101–110, 2004.

[34] A. Atallah, R. Woollands, T. Elgohary, and J. Junkins, "Accuracy and efficiency comparison of six numerical integrators for propagating perturbed orbits," *Journal of the Astronautical Sciences*, vol. 67, 2020.

[35] T. Gardner, B. Cheetham, A. Forsman, C. Meek, E. Kayser, J. Parker, and M. Thompson, "Capstone: A cubesat pathfinder for lunar gateway ecosystem," *35th Annual Small Satellite Conference*, 2021.

[36] L. F. Shampine and M. W. Reichelt, "The matlab ode suite," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997.

[37] C. Frueh, K. Howell, K. J. DeMars, and S. Bhadauria, "Cislunar space situational awareness," in *AAS/AIAA 31st Space Flight Mechanics Meeting*, 2021.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.