mathematics

MDPI

Article

A Functional Interpolation Approach to Compute Periodic Orbits in the Circular-Restricted Three-Body Problem

Hunter Johnston [1,*], Martin W. Lo [2] and Daniele Mortari [1]

[1] Aerospace Engineering, Texas A&M University, College Station, TX 77843, USA; mortari@tamu.edu
[2] Mission Design and Navigation Section, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91125, USA; martin.w.lo@jpl.nasa.gov
* Correspondence: hunterjohnston@tamu.edu

Abstract: In this paper, we develop a method to solve for periodic orbits, i.e., Lyapunov and Halo orbits, using a functional interpolation scheme called the Theory of Functional Connections (TFC). Using this technique, a periodic constraint is analytically embedded into the TFC constrained expression. By doing this, the system of differential equations governing the three-body problem is transformed into an unconstrained optimization problem where simple numerical schemes can be used to find a solution, e.g., nonlinear least-squares is used. This allows for a simpler numerical implementation with comparable accuracy and speed to the traditional differential corrector method.

Keywords: functional interpolation; Theory of Functional Connections; ordinary differential equations; least-squares

Citation: Johnston, H.; Lo, M.W.; Mortari, D. A Functional Interpolation Approach to Compute Periodic Orbits in the Circular-Restricted Three-Body Problem. *Mathematics* **2021**, *9*, 1210. https://doi.org/10.3390/math9111210

Academic Editor: Ioannis K. Argyros

Received: 10 May 2021
Accepted: 25 May 2021
Published: 27 May 2021

1. Introduction

According to Poincaré, "periodic orbits" provide the only gateway into the otherwise impenetrable domain of nonlinear dynamics. With the advent of space exploration, periodic orbits have become an indispensable part of missions in space. The amazing fish-like Apollo orbit was the first three-body orbit used for space missions. The second three-body orbit used for space missions was the Halo orbit, discovered by Robert Farquhar in his Ph.D. thesis [1] under John Breakwell [2]. In 1978, Farquhar convinced NASA and led the International Sun-Earth Explorer 3 mission (ISEE3) to study the Sun from a Halo orbit around the Earth's L1 Lagrange point. Farquhar's original idea was to place a satellite in Halo orbit around the Lunar L2 for telecommunication support for the backside of the Moon. Today, this is indeed part of NASA's planned return of humans to the Moon in the next few years.

Computing Halo orbits, and finding nonlinear periodic orbits in general, has become an important part of modern mission design. This paper introduces a novel, simple, and efficient method for finding periodic orbits using the Theory of Functional Connections (TFC) [3,4]. First, we review perhaps the most popular standard method for computing periodic orbits: the differential correction method (also called shooting method), such as presented by Kathleen Howell [5]. One begins with an approximate solution obtained typically from normal form expansions. Using the variational equation, the guess solution is iteratively corrected for periodicity. Assuming that the initial guess is in a reasonable basis of attraction to a periodic orbit, the process converges to a periodic orbit. In Hamiltonian systems, periodic orbits typically occur in one-parameter families. Often, there are multiple families nearby. Hence, the convergence may not always lead to the desired orbit. Moreover, control over the specific features of the periodic orbit, such as its period or energy, requires additional work—for example, using continuation methods to reach the exact orbit desired. Using TFC, the functional interpolation theory, a simpler formulation and more efficient algorithm for finding periodic orbits is possible.

With this work, we removed the need for a shooting method by analytically embedding the boundary conditions in the solution space using TFC [3,4]. These expressions, which will be explicitly derived in the following sections, transform a differential equation subject to constraints into an unconstrained problem, reducing the whole solution function space to only the subspace satisfying the constraints. This method drastically simplifies the problem and allows for simple numerical implementation; in our case, we utilize a nonlinear least-squares technique.

2. Circular-Restricted Three-Body Problem

The circular-restricted three-body problem (CR3BP) is a dynamical model used to describe the motion of a particle $\boldsymbol{r} = \{x, y, z\}^{\intercal}$ of negligible mass under the influence of a primary body of mass m_1 and the secondary body of mass m_2. Furthermore, the orbits of m_1 and m_2 are subject to circular motion about the system's barycenter and lie in the x-y plane as depicted in Figure 1. Following this, the system can be non-dimensionalized by the following scaled units: unit mass is defined as $m_1 + m_2$; unit length is taken as the separation between m_1 and m_2; the unit time is chosen such that the orbital periods of m_1 and m_2 about the system's barycenter are 2π. By following these steps, the system can be reduced to a single parameter called the mass parameter, μ, where,

$$\mu = \frac{m_2}{m_1 + m_2} \tag{1}$$

From this, we define the terms μ_1 and μ_2 based on Equation (1) as

$$\mu_1 = 1 - \mu \qquad \text{and} \qquad \mu_2 = \mu.$$

Using this definition of the system, the equations of motion can be derived in the rotating frame, leading to the following system of equations:

$$\begin{aligned} \ddot{x} - 2\dot{y} &= \frac{\partial \Omega}{\partial x} \\ \ddot{y} + 2\dot{x} &= \frac{\partial \Omega}{\partial y} \\ \ddot{z} &= \frac{\partial \Omega}{\partial z} \end{aligned} \tag{2}$$

such that the shorthand dot notation is used to signify a derivative with respect to time (e.g., $\dot{x} := \dfrac{\mathrm{d}x}{\mathrm{d}t}$). Additionally, Ω is defined as

$$\Omega(x, y, z) := \frac{1}{2}(x^2 + y^2) + \frac{1 - \mu}{R_1} + \frac{\mu}{R_2} + \frac{1}{2}(1 - \mu)\mu \tag{3}$$

where $R_1 = \sqrt{(x + \mu)^2 + y^2 + z^2}$ and $R_2 = \sqrt{(x + \mu - 1)^2 + y^2 + z^2}$ are the distances to the primaries. Furthermore, the equations of motion are Hamiltonian and independent of time and, thanks to Noether's theorem, have an energy integral of motion $\mathcal{E}$, where, in the celestial mechanics community, the Jacobi constant is used, which is $C := -2\mathcal{E}$ and given as

$$C = 2\,\Omega - (\dot{x} + \dot{y} + \dot{z}) = (x^2 + y^2) + 2\frac{1 - \mu}{R_1} + 2\frac{\mu}{R_2} + (1 - \mu)\mu - (\dot{x} + \dot{y} + \dot{z}). \tag{4}$$

Moving forward, we solve the dynamics defined by the system of equations in Equation (2) such that the orbit is at a fixed energy level (or rather Jacobi constant) using Equation (4). For our implementation, it is useful to define the residuals of these equations:

$$0 = F_x := \ddot{x} - 2\dot{y} - \frac{\partial \Omega}{\partial x} \tag{5}$$

$$0 = F_y := \ddot{y} + 2\dot{x} - \frac{\partial \Omega}{\partial y} \tag{6}$$

$$0 = F_z := \ddot{z} - \frac{\partial \Omega}{\partial z} \tag{7}$$

$$0 = F_c := (x^2 + y^2) + 2\frac{1-\mu}{R_1} + 2\frac{\mu}{R_2} + (1-\mu)\mu - (\dot{x} + \dot{y} + \dot{z}) - C \tag{8}$$

Next, we generate analytical expressions for the states to guarantee a periodic orbit.

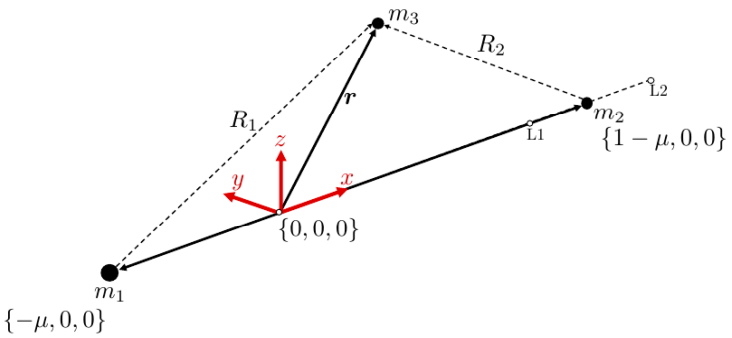


Figure 1. Geometry of the circular-restricted three-body problem where the secondary body, m_2, is in a circular orbit around the primary body, m_1. The third-body, whose mass is $m_3 \ll m_2 < m_1$, is negligible and at a distance R_1 from m_1, R_2 from m_2, and $\boldsymbol{r}$ from the system's barycenter (the system's center of mass).

However, before moving forward, it is necessary to summarize the major steps of the differential corrector method [5], since this technique is used as a baseline to compare results with the proposed method. In general, the differential correction method can be classified as a shooting method since it relies on transforming the boundary-value problem into an initial-value problem where the initial conditions are iteratively adjusted. First, unlike the the proposed technique, the differential correction method can take advantage of the symmetry of the orbits about the x-z plane, which leads to the following conditions:

$$X(0) = \begin{Bmatrix} x_0, & 0, & z_0, & 0, & \dot{y}_0, & 0 \end{Bmatrix}^{\mathrm{T}} \tag{9}$$

and

$$X(T/2) = \begin{Bmatrix} x, & 0, & z, & 0, & \dot{y}, & 0 \end{Bmatrix}^{\mathrm{T}} \tag{10}$$

where Equations (9) and (10) represent the initial state and the state at half of the period, T, where the orbit crosses the x-z plane (the sign of the y variable). Therefore, the equations of motion only need to be propagated to this sign change. At this point, if the values of $|\dot{x}|$ and $|\dot{z}|$ are zeros within an acceptable tolerance (Ref. [5] uses 10^{-8}), then the orbit is considered periodic. If they are not, the initial conditions are updated by determining

$$\delta X(0) = \begin{Bmatrix} \delta x_0, & 0, & \delta z_0, & 0, & \delta \dot{y}_0, & 0 \end{Bmatrix}^{\mathrm{T}} \tag{11}$$

and adding the result to Equation (9). The values in Equation (11) are determined by the state transitions matrix calculated in the propagation step and the fact that the only changes to state should be the variables $\dot{x}$ and $\dot{z}$, which should be equal to the negative of their values at $T/2$, i.e., $\delta\dot{x}$ should be $-\dot{x}$ (see Refs. [2,5] for the detailed equations).

After updating the initial conditions, the equations of motion are propagated again and the process continues until the periodic conditions are met.

3. Solving the Problem Using the Theory of Functional Connections Framework

The numerical technique to solve differential equations based on the Theory of Functional Connections [4] has produced machine error accuracy within milliseconds for both linear and nonlinear [6] differential equations for a wide array of constraint cases. In fact, several works have already adopted the TFC framework to solve mathematical problems including hybrid systems [7], optimal control problems [8,9], quadratic and nonlinear programming [10], and other applications [11,12].

The foundation of this approach is the ability to derive a class of functionals, called *constrained expressions*, which have the characteristic of always analytically satisfying assigned linear constraints. Using these constrained expressions, the differential equation can be transformed into an algebraic expression that can ultimately be solved via a variety of optimization techniques. In general, the TFC methodology provides *functional interpolation* by embedding a set of k linear constraints. Let $g(t)$ be a general function in our unconstrained function space, G. We define next the subspace, $\Gamma \subset G$, of constrained functions, $x(t, g(t))$, subject to these k linear constraints written as the following functional:

$$x(t, g(t)) = g(t) + \sum_{j=1}^{k} \phi_j(t)\, \rho_j(t, g(t)), \tag{12}$$

where $g(t)$ is a free function, and $\phi_j(t)$ are *switching functions*—by definition, a function is equal to 1 when evaluated at the constraint it is referencing, and equal to 0 when evaluated at all other constraints. In our case, the switching functions are composed of a set of mutually linearly independent functions called support functions, $s_k(t)$, with unknown coefficients α_{ij}, such that $\phi_j(t) = s_i(t)\, \alpha_{ij}$; however, the switching functions can be very general depending on the nature of the original function space, G.

Furthermore, $\rho_j(t, g(t))$ are called *projection functionals*, since they are derived by setting the constraint function equal to zero and replacing $x(t)$ with $g(t)$. The following step-by-step procedure can be used to derive a constrained expression from Equation (12):

1. Choose k linearly independent support functions, $s_k(t)$.
2. Write each switching function as a linear combination of the support functions with k unknown coefficients.
3. Based on the switching function definition, write a system of equations to solve for the unknown α_{ij} coefficients.

3.1. Derivation of Constrained Expression

First, since, in our problem, the orbital period is unknown, let us normalize the time domain from $t \in [0, T]$, where $t_0 = 0, t_f = T$, to $\tau \in [-1, +1]$, where $\tau_0 = -1, \tau_f = +1$. The simplest mapping is a linear one as follows:

$$\tau(t) = \tau_0 + \frac{\tau_f - \tau_0}{t_f - t_0}(t - t_0) \quad \longleftrightarrow \quad t = t_0 + \frac{t_f - t_0}{\tau_f - \tau_0}(\tau - \tau_0). \tag{13}$$

where we defined the following coefficient term of the derivative of τ with respect to t. (Note: here, we define the coefficient term as the square of b in order to ensure that the ratio is always positive, i.e., the problem domain is always consistent. This is important in numerical implementation when solving for the value of b.)

$$b^2 := \frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{\tau_f - \tau_0}{t_f - t_0}, \tag{14}$$

Now, we write the constrained expression in terms of this new variable τ. Let us define the position vector of the spacecraft as $\boldsymbol{r}(\tau) = \{r_x(\tau), r_y(\tau), r_z(\tau)\}^{\mathrm{T}} = \{x(\tau), y(\tau), z(\tau)\}^{\mathrm{T}}$.

Since the problem presented in this paper is to find periodic orbits in the CR3BP, we can use the TFC expression to satisfy the following constraints:

$$r_i(\tau_0) = r_i(\tau_f) = \alpha_i \quad \text{and} \quad r_i'(\tau_0) = r_i'(\tau_f) = \frac{\beta_i}{b^2}$$

or, in other words, the trajectory must return to the initial state at some period. Since the constraints for each component of the vector $r_i(\tau)$ are of the same form, Equation (12) can be easily adapted to the vector form:

$$r_i(\tau, g_i(\tau)) = g_i(\tau) + \sum_{j=1}^{k} \phi_j(\tau)\, \rho_j(t, g_i(\tau)), \tag{15}$$

Following the process detailed in the prior section, one must first choose the support functions $s_k(\tau)$. For these specific constraints, the support functions are defined in the same manner as Ref. [9], where $s_k(\tau) = \tau^{(k-1)}$. Now, the definition of the switching functions is used to produce a set of equations. For example, the first switching function has the four following equations:

$$\phi_1(\tau_0) = 1, \quad \phi_1(\tau_f) = 0, \quad \phi_1'(\tau_0) = 0, \quad \text{and} \quad \phi_1'(\tau_f) = 0,$$

These equations are expanded in terms of the support functions:

$$\begin{aligned}
\phi_1(\tau_0) &= (1)\cdot\alpha_{11} + (\tau_0)\cdot\alpha_{21} + (\tau_0^2)\ \cdot\alpha_{31} + (\tau_0^3)\ \cdot\alpha_{41} = 1\\
\phi_1(\tau_f) &= (1)\cdot\alpha_{11} + (\tau_f)\cdot\alpha_{21} + (\tau_f^2)\ \cdot\alpha_{31} + (\tau_f^3)\ \cdot\alpha_{41} = 0\\
\phi_1'(\tau_0) &= (0)\cdot\alpha_{11} + (1)\ \cdot\alpha_{21} + (2\tau_0)\cdot\alpha_{31} + (3\tau_0^2)\cdot\alpha_{41} = 0\\
\phi_1'(\tau_f) &= (0)\cdot\alpha_{11} + (1)\ \cdot\alpha_{21} + (2\tau_f)\cdot\alpha_{31} + (3\tau_f^2)\cdot\alpha_{41} = 0
\end{aligned}$$

which can be written in matrix form as

$$\begin{bmatrix} 1 & \tau_0 & \tau_0^2 & \tau_0^3 \\ 1 & \tau_f & \tau_f^2 & \tau_f^3 \\ 0 & 1 & 2\tau_0 & 3\tau_0^2 \\ 0 & 1 & 2\tau_f & 3\tau_f^2 \end{bmatrix} \begin{Bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \\ \alpha_{41} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}.$$

The same is done for the other three switching functions to produce a set of equations that can be solved by matrix inversion, where $\tau_0 = -1$ and $\tau_f = 1$.

$$\begin{bmatrix} 1 & \tau_0 & \tau_0^2 & \tau_0^3 \\ 1 & \tau_f & \tau_f^2 & \tau_f^3 \\ 0 & 1 & 2\tau_0 & 3\tau_0^2 \\ 0 & 1 & 2\tau_f & 3\tau_f^2 \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

that is,

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} = \begin{bmatrix} 1 & \tau_0 & \tau_0^2 & \tau_0^3 \\ 1 & \tau_f & \tau_f^2 & \tau_f^3 \\ 0 & 1 & 2\tau_0 & 3\tau_0^2 \\ 0 & 1 & 2\tau_f & 3\tau_f^2 \end{bmatrix}^{-1} = \frac{1}{4}\begin{bmatrix} 2 & 2 & 1 & -1 \\ -3 & 3 & -1 & -1 \\ 0 & 0 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}.$$

This leads to the switching functions,

$$\phi_1(\tau) = \frac{1}{4}\left(2 - 3\tau + \tau^3\right) \qquad \phi_2(\tau) = \frac{1}{4}\left(2 + 3\tau - \tau^3\right)$$
$$\phi_3(\tau) = \frac{1}{4}\left(1 - \tau - \tau^2 + \tau^3\right) \qquad \phi_4(\tau) = \frac{1}{4}\left(-1 - \tau + \tau^2 + \tau^3\right).$$

By their definition, the projection functionals are

$$\rho_1 = \alpha_i - g_i(\tau_0) \qquad \rho_2 = \alpha_i - g_i(\tau_f)$$
$$\rho_3 = \frac{\beta_i}{b^2} - g_i'(\tau_0) \qquad \rho_4 = \frac{\beta_i}{b^2} - g_i'(\tau_f)$$

Collecting these terms in the form of Equation (15), the constrained expression and its derivatives can be expressed as

$$r_i(\tau, g_i(\tau)) = g_i(\tau) + \phi_1(\tau)\left(\alpha_i - g(\tau_0)\right) + \phi_2(\tau)\left(\alpha_i - g(\tau_f)\right) + \phi_3(\tau)\left(\frac{\beta_i}{b^2} - g'(\tau_0)\right) + \phi_4(\tau)\left(\frac{\beta_i}{b^2} - g'(\tau_f)\right) \tag{16}$$

In order to use this expression in our differential equation, we need to utilize the mapping parameter b from Equation (14), where

$$\underbrace{\frac{\mathrm{d}r_i}{\mathrm{d}t}}_{\dot{r}_i} = b^2 \underbrace{\frac{\mathrm{d}r_i}{\mathrm{d}\tau}}_{r_i'}.$$

3.2. Definition of the Free Function $g_j(\tau)$

For this application, let the free function be expressed as a linear combination of basis functions multiplied with unknown coefficients according to

$$g_i(\tau) = \sum_{j=1}^{m} a_{ij}\, T_j(\tau) = \boldsymbol{\xi}_i^{\mathrm{T}}\, \boldsymbol{h}(\tau) \tag{17}$$

where $\boldsymbol{h} \in \mathbb{R}^m$ is a vector of the basis functions $T_j(\tau)$ and $\boldsymbol{\xi}_i \in \mathbb{R}^m$ is the unknown coefficient vector associated with the ith vector component. Like the original work on the solution of ordinary differential equations with TFC [6], we utilize Chebyshev orthogonal polynomials in the paper. The orthogonal polynomial representations are generally preferred due to their approximating and convergence properties. For example, Chebyshev polynomials generate a function that minimizes the maximum error in its application, and, therefore, they are well suited to approximating other functions [13,14].

However, due to the structure of the constrained expression, the free functions can be defined by a variety of function approximation methods, including other orthogonal polynomial sets and techniques in the field of machine learning. In fact, the application to neural networks (NN) was studied in Ref. [11], and promising results have been obtained by using the Extreme Learning Machine (ELM) [15], which is a single-layer feed-forward NN. The interested reader is directed to Ref. [16] for an in-depth look at the ELM implementation with TFC and Ref. [17] for applications of this technique to problems in spaceflight.

3.3. Discretization of the Domain

In order to solve problems numerically, the problem domain (and therefore the basis function domain) must be discretized. Since this article uses Chebyshev orthogonal polynomials, an optimal discretization scheme is the Chebyshev–Gauss–Lobatto nodes [18,19] shown in Equation (18). For $N + 1$ points, the discrete points are calculated as

$$\tau_k = -\cos\left(\frac{k\pi}{N}\right) \quad \text{for} \quad k = 0, 1, 2, \cdots, N. \tag{18}$$

Compared with the uniform distribution, the collocation point distribution results in a much slower increase in the condition number of the matrix to be inverted in the least-squares as the number of basis functions, m_2, increases. The collocation points can be realized in the problem domain through the relationship provided in Equation (13).

3.4. Solution of the Resulting Algebraic Equation

By defining $g_i(t)$ according to Equation (17), our definition of Equations (2) and (3) is converted into four algebraic equations:

$$\tilde{F}_i(\tau, \Xi) = 0 \quad \text{for} \quad i = x, y, z, c \tag{19}$$

defined from Equation (5) through Equation (8). Note that the tilde symbol ($\sim$) is used to signify that the differential Equation (5) through Equation (8) have been transformed into a new set of equations with embedded constraints. Additionally, Ξ represents the unknown parameters of this transformed set of equations, and these are defined in the following equations.

Next, we evaluate the three differential equations and one algebraic equation given by Equation (19) at the discretization points to construct a loss vector of the residuals of these equations.

$$\mathbb{L}_i(\Xi) = \left\{\tilde{F}_i(\tau_0, \Xi), \quad \ldots, \quad \tilde{F}_i(\tau_k, \Xi), \quad \ldots, \quad \tilde{F}_i(\tau_f, \Xi)\right\}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$$

with the total loss vector of

$$\mathbb{L}(\Xi) = \left\{\mathbb{L}_x^{\mathrm{T}}(\Xi), \quad \mathbb{L}_y^{\mathrm{T}}(\Xi), \quad \mathbb{L}_z^{\mathrm{T}}(\Xi), \quad \mathbb{L}_c^{\mathrm{T}}(\Xi)\right\}^{\mathrm{T}} = \mathbf{0}_{4N\times 1}^{\mathrm{T}} \tag{20}$$

where the unknown vector, of size composed of $3m + 7$ terms, is defined as

$$\Xi = \left\{\boldsymbol{\xi}_x^{\mathrm{T}}, \quad \boldsymbol{\xi}_y^{\mathrm{T}}, \quad \boldsymbol{\xi}_z^{\mathrm{T}}, \quad \boldsymbol{\alpha}^{\mathrm{T}}, \quad \boldsymbol{\beta}^{\mathrm{T}}, \quad b\right\}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$$

Since the governing dynamics of the circular-restricted three-body problem are nonlinear, the unknown parameters occur nonlinearly in the loss vector, Equation (20). Therefore, a nonlinear least-squares technique was utilized using Equation (21) to update the parameters:

$$\Xi_{k+1} = \Xi_k + \Delta\Xi_k \tag{21}$$

where Equation (22) is the classical least-squares equation using the Moore–Penrose inverse of the Jacobian of Equation (20):

$$\Delta\Xi_k = -\left(\mathbb{J}^{\mathrm{T}}(\Xi_k)\,\mathbb{J}(\Xi_k)\right)^{-1}\mathbb{J}^{\mathrm{T}}(\Xi_k)\,\mathbb{L}(\Xi_k), \tag{22}$$

This nonlinear least-squares method, known as the Gauss–Newton algorithm, is a modification of Newton's method for finding a minimum of a function—in our case, minimizing the sum of the squared residual values in Equation (20).

The iteration continues until one of the following conditions is met:

$$\max|\mathbb{L}(\Xi_k)| < \varepsilon \quad \max|\Delta\Xi_k| < \varepsilon,$$

based on the user defined tolerance, ε, or the iteration exceeding some user-defined maximum.

4. Numerical Test

In the numerical implementation, all Jacobians were calculated using the JAX package [20,21] utilizing automatic differentiation [22]. In the nonlinear least-squares steps,

NumPy's `pinv()` was used to calculate the Moore–Penrose pseudo-inverse through singular-value decomposition (SVD). Additionally, a just-in-time compiler was used to optimize the code. All timing was handled by the `process_time()` from the Python package `time`. For all problems, we considered the Earth–Moon system with the parameters given in Table 1. Additionally, for the TFC implementation, the parameters used are summarized in Table 2.

Table 1. Earth–Moon system parameters.

Variable	Value
Earth mass m_1 (kg)	5.9724×10^{24}
Moon mass m_2 (kg)	7.3460×10^{22}

Table 2. TFC algorithm parameters.

Variable	Value
N [number of points]	140 [Lyapunov] or 200 [Halo]
m [basis terms]	130 [Lyapunov] or 190 [Halo]
ε [tolerance]	2.22×10^{-16}
Maximum iterations	20
Least-squares method	`numpy.pinv()`

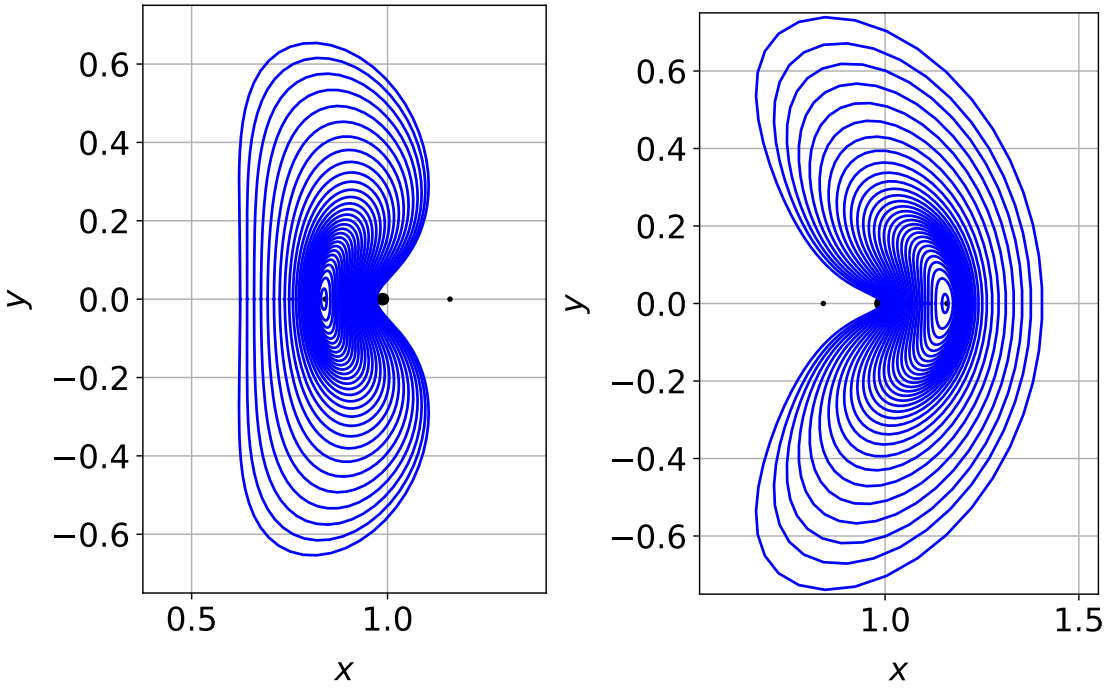
4.1. Initialization

For all numerical tests, the unknown vector must be initialized. First, the terms $\boldsymbol{\xi}_x$, $\boldsymbol{\xi}_y$, and $\boldsymbol{\xi}_z$ were all initialized by a null vector, which ultimately represents the simplest interpolating expression for the state variables. This initialization represents the worst-case scenario when there is no estimation of the trajectory, but just fitting the constraints. Next, the other unknown values of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and b (which are associated with the position, velocity, and the period of the orbit) were initialized using Richardson's third-order analytical method for Halo-type periodic motion [23]. This initialization was used to find the first orbit of the specified Jacobi constants. For the following orbits, the desired Jacobi constant was incrementally increased, and the converged values from the prior Jacobi constant level were used to initialize each following step.

The same process was utilized for the differential corrector method [5], which was implemented to compare with TFC. In the differential corrector inner-loop, the desired Jacobi constant was obtained by an iterative least-squares approach to update the initial guess.

4.2. Lyapunov Orbits

First, the TFC method was used to explore the computation of Lyapunov orbits, which lie in the x-y plane, the orbital plane of the two primaries. For our test, the Lyapunov orbits were computed over a range of Jacobi constants, starting close to the equilibrium point's specific energy levels and terminating at a Jacobi constant of 2.92. The associated trajectories for the orbits around L1 and L2 are provided in Figure 2a,b.



(**a**) Lyapunov orbits around L1 Lagrange point (**b**) Lyapunov orbits around L2 Lagrange point

Figure 2. Trajectories of Lyapunov orbits for the Earth–Moon system ranging from energies close to the L1/L2 equilibrium points to 2.92.

For the solution of Lyapunov orbits, it was important to transform the time for orbits of lower Jacobi constants, i.e., orbits closer to the secondary body, m_2. This was done by the time transformation from Ref. [5], where $t \to \zeta$, such that

$$\frac{\mathrm{d}t}{\mathrm{d}\zeta} = R_2. \tag{23}$$

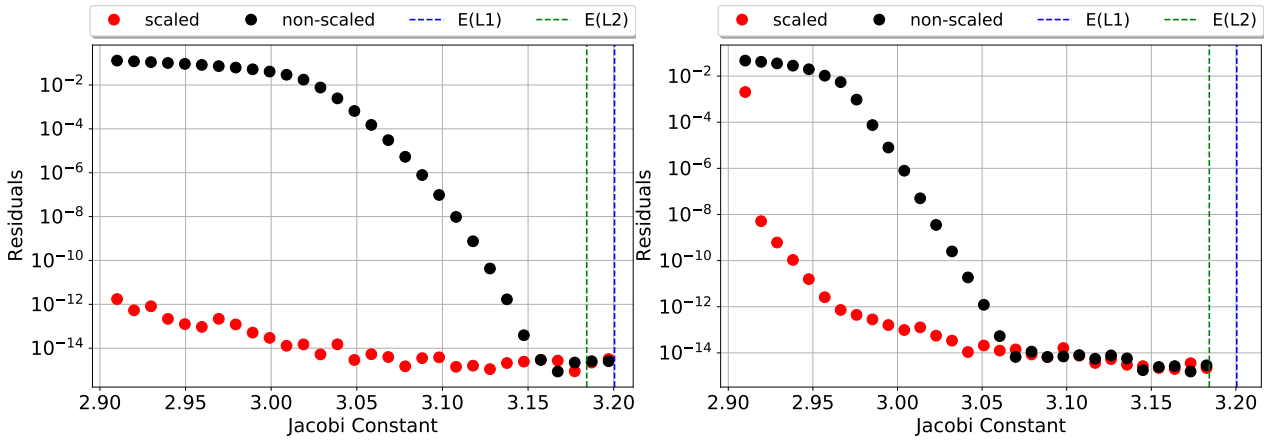
This produces the following chain of transformations $t \to \zeta \to \tau$, or rather $[0, T] \to [0, \zeta_f] \to [-1, +1]$, which produces the final transformation:

$$\dot{r}_i(t) = \frac{\mathrm{d}r_i}{\mathrm{d}t} = \frac{\mathrm{d}r_i}{\mathrm{d}\tau} \cdot \frac{\mathrm{d}\tau}{\mathrm{d}\zeta} \cdot \frac{\mathrm{d}\zeta}{\mathrm{d}t} = \frac{b^2}{R_2} r_i'(\tau)$$

To stress the importance of this scaling, Figure 3 shows both the scaled and non-scaled TFC solution for Lyapunov orbits around L1 (Figure 3a) and L2 (Figure 3b).

It can be seen that this scaling has a drastic effect on the accuracy of the converged solution. For orbits around L1, without scaling, significant accuracy loss is observed starting at a Jacobi constant of around 3.15. For orbits around L2, this accuracy reduction starts at a Jacobi constant of around 3.06, but is just as drastic.

Additionally, a comparison with the differential corrector method is provided in terms of speed and accuracy. Figure 4 compares the residuals for both methods, where it can be seen that the TFC approach is around two orders of magnitude more accurate than the differential corrector at higher Jacobi constants. Furthermore, the computation of the TFC solution is slightly faster, a little over 0.25 s in the extreme case, as displayed in Figure 5.



(**a**) Solution for L1 Lagrange point (**b**) Solution for L2 Lagrange point

Figure 3. Accuracy comparison of the scaled vs. non-scaled solution of Lyapunov orbits. The scaled solution utilizes the time transformation given by Equation (23) and clearly produces more accurate results as the Jacobi constant decreases. The scaled formulation was used in all of the following results for Lyapunov orbits.

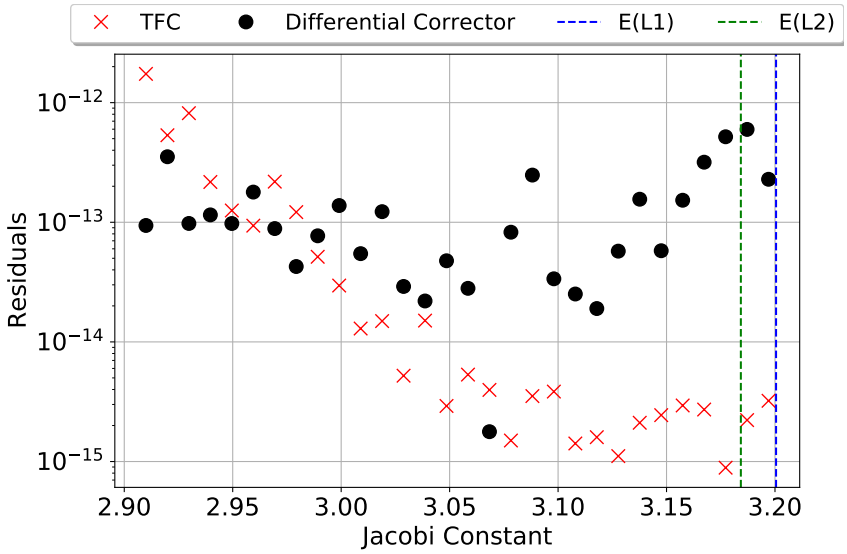


Figure 4. Maximum residuals of the loss vector for the TFC method solving for the trajectories plotted in Figure 2a compared to that of the differential corrector. The lines of E(L1) and E(L2) represent the energy of the L1 and L2 Lagrange points, respectively. The TFC approach has a slight accuracy advantage (an order of magnitude) as compared to the differential corrector method at higher Jacobi constants.

To further understand the computation time associated with the TFC approach, consider the computational breakdown given in Figure 6. Additionally, the statistics associated with this breakdown, in terms of mean and standard deviation, are given in Table 3. In the TFC method, the three major computations in each nonlinear least-squares iteration are evaluating the loss vector, the Jacobian, and the least-squares. Figure 6 shows that around 77% of the total computation time is associated with the least-squares portion. This should come as no surprise since the Jacobian is a 4 N $\times$ 2 m matrix—560 $\times$ 260 for this problem.

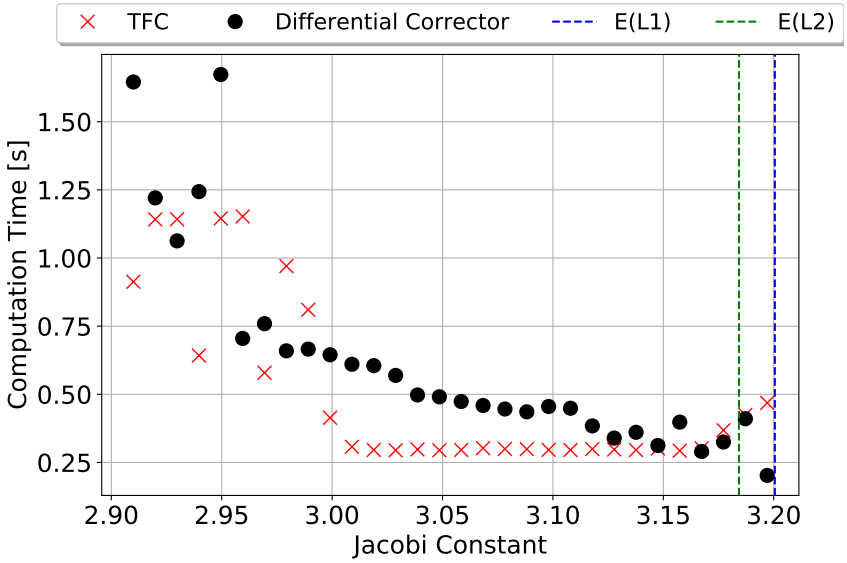


Figure 5. Computational time of the the TFC method for the trajectories plotted in Figure 2a compared to that of the differential corrector. The TFC method holds a slight speed gain over the differential corrector.

While the least-squares, i.e., matrix inversion, is the main bottleneck with the computational efficiency, a marginal speed gain could be achieved by removing the reliance on automatic differential through JAX [20,21] for the computation of the loss vector and Jacobian. Admittedly, these two could be "hard-coded" since the partial derivatives populating the Jacobian can be analytically derived. However, the speed gain associated with this is unknown and should be the focus of future implementations and numerical tests.

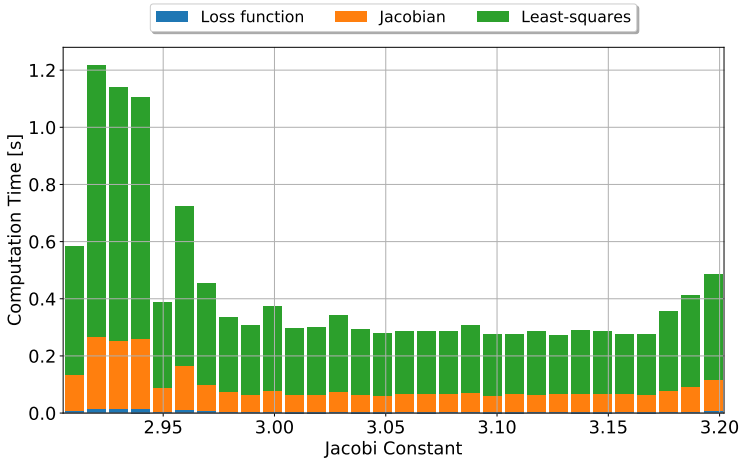


Figure 6. Breakdown of computation time for the trajectories solved in Figure 2a. Each bar represents the total computation time (all nonlinear least-squares iterations). On average, the evaluation of the loss vector (Equation (20)), the Jacobian, and the nonlinear least-squares represent 1.3%, 21.4%, and 77.3% of the computation time, respectively. A further breakdown of the computation time statistics is provided in Table 3.

Table 3. Statistics associated with the computational breakdown given in Figure 6.

Variable	Average (ms)	Standard Deviation (ms)
Loss vector	5.69	3.46
Jacobian	91.1	55.8
Least-squares	329.8	203.8

Similar to the L1 Lagrange point test, Figure 2a displays the computed trajectories around the L2 Lagrange point. Additionally, as in Figures 4 and 5, the accuracy and computation time for these tests are provided in Figures 7 and 8. In Figure 7, the TFC method is more accurate, albeit only slightly. At a Jacobi constant level of around 3.00 and above, the differential corrector method does not converge, as shown by the jump in accuracy. At this Jacobi constant level, the TFC method's accuracy starts to decrease before failing to converge at the Jacobi constant value of 2.92. This drop-off in the accuracy of the TFC method can easily be seen by analyzing Figure 8, where the red box highlights where the TFC method's max iteration limit is reached. In other words, at the Jacobi constant levels lower than 3.00, the representation of the orbit through the free function composed of Chebyshev polynomials starts to fail.

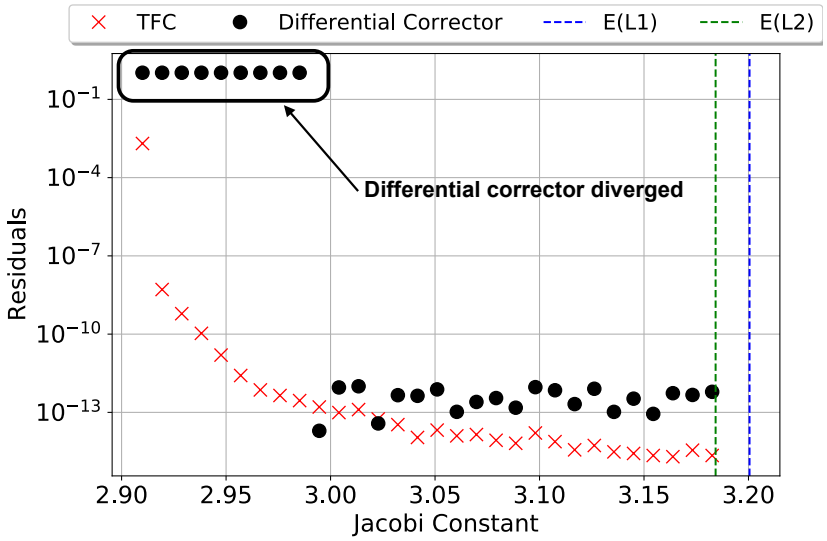


Figure 7. Maximum residuals of the loss vector for the TFC method solving for the trajectories plotted in Figure 2b as compared to the differential corrector. For the trajectories around L2, the differential corrector diverged around a Jacobi constant level of 3.00, while the TFC method was able to solve the problem with diminishing accuracy. The black box highlights the diverged cases.

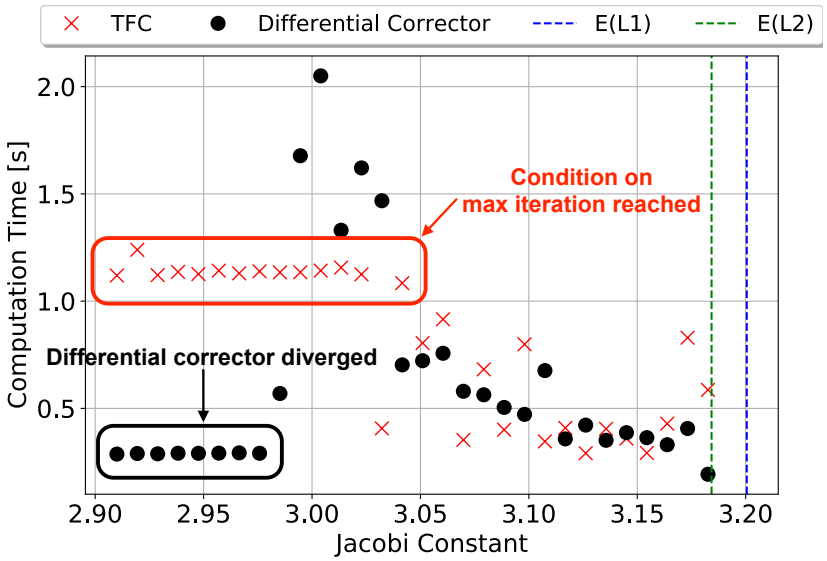


Figure 8. Computational time of the TFC method for the trajectories plotted in Figure 2b compared to that of the differential corrector. Again, the black box highlights where the differential corrector diverged. Additionally, the red box shows where the TFC method reached its maximum allowed iterations of 20. These cases are correlated to the reduction in accuracy seen in Figure 7.

In both Lyapunov orbits around L1 and L2, the reduction in accuracy can be attributed to the definition of the free function, $g(x)$, not fully capturing the solution space of the trajectory. As the Jacobi constant decreases, the complexity of the trajectory's shape increases and therefore an increasing number of basis functions is needed. However, contrary to this, increasing the basis terms to more than 130 does not provide a more accurate solution in the numerical implementation. Therefore, when solving lower Jacobi constant level Lyapunov orbits, a different definition of the free function must be used; future effort should focus on exploring other basis types.

4.3. Halo Orbits

Next, the proposed technique was utilized to compute Halo orbits around L1 and L2. These orbits differ from the Lyapunov orbit because they are *not* restricted to the *x*-*y* plane and are therefore three-dimensional. In fact, this family of orbits is a bifurcation of the Lyapunov orbits computed in the previous section and is characterized by "northern" and "southern" bifurcations. However, using the TFC method to compute these Halo orbits, the only difference is in the initialization of the $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and b parameters. Additionally, for Halo orbits, the time transformation introduced in Equation (23) was not used because it produced unfavorable convergence properties, i.e., the method would simply converge to the Lyapunov family of orbits. First, we look at the computation of the "northern" family of Halo orbits around L1 and L2, as plotted in Figure 9.

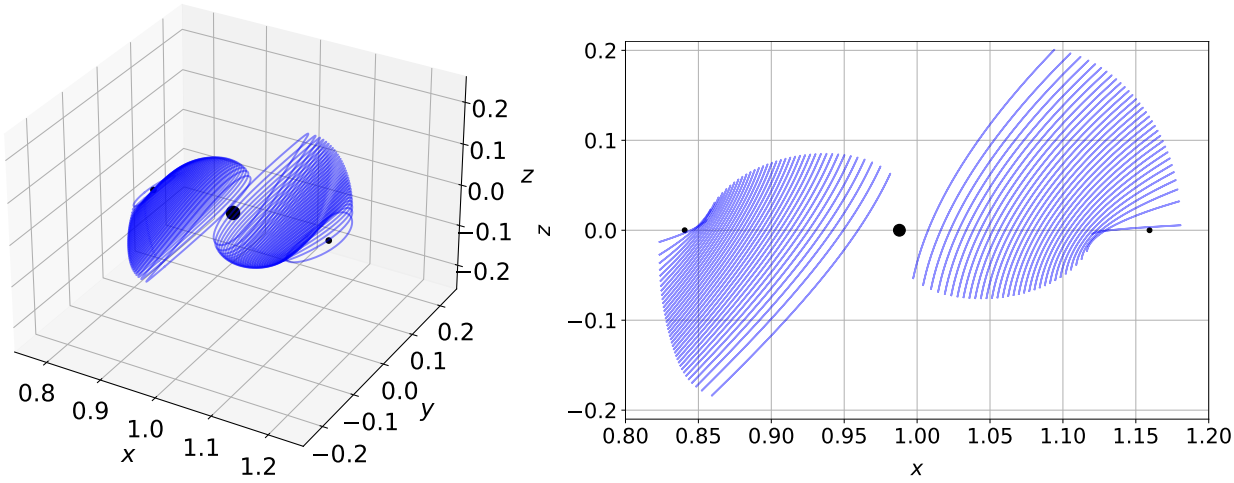


Figure 9. Halo orbits of the "northern" bifurcation around both L1 and L2 Lagrange points.

In these plots, we can see that, around the L1 equilibrium point, the method converged to Lyapunov orbits for higher Jacobi constants. However, as the Jacobi constant decreases below 3.025, the method does not converge to a periodic orbit, as shown by the increase in residuals around the Jacobi constant value of 3.025. At the other equilibrium point, L2, the method converges for all Jacobi constant values; however, at values below 3.025, the solution jumps to a circular orbit around the Moon. Therefore, these orbits were not plotted. This "jumping" from the Halo orbit bifurcation to the circle orbit bifurcation is the major drawback with the algorithm in its current state; however, it is noted that the differential corrector approach also suffers from this. The reason that the TFC method exhibits this behavior is that the numerical solution, i.e. the nonlinear least-squares optimization step, has no information on the desired bifurcation. Therefore, the solution becomes closely linked to (1) the initialization of the unknown parameters and (2) the definition of the free function $g(x)$. In other words, the initialized parameters will converge to the closest local minimum.

Like the Lyapunov orbit tests, the loss vector's maximum residual was recorded and is plotted in Figure 10. For almost all converged solutions, the residuals were on the order

of $\mathcal{O}(10^{-14})$; however, around a Jacobi constant level of 3.025, the accuracy decreases for orbits around L1. The convergence for Halo orbits took longer, with some cases taking 8 s, while, on average, the solution time was around 2 s, as shown in Figure 11. Similar to the "northern" Halo orbits plotted in Figure 9, the Halo orbits of the "southern" bifurcation were computed with similar findings and, therefore, omitted from this paper for brevity.

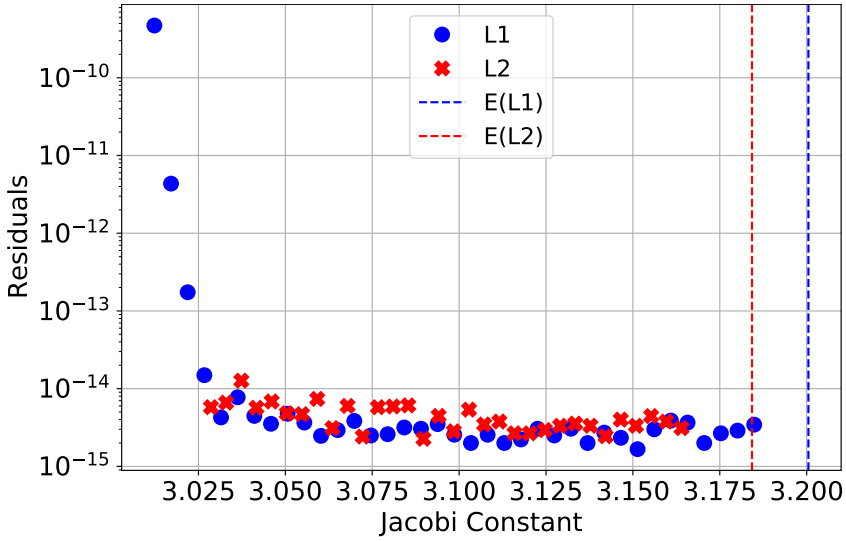


Figure 10. Maximum residuals of the loss vector for the TFC method solving for the trajectories plotted in Figure 9. For almost all cases, the solution accuracy is on the order of $\mathcal{O}(10^{-14})$. However, around a Jacobi constant level of 3.025, the accuracy decreases for orbits around L1. The solutions for orbits around L2 lower than 3.025 are not plotted because, while they converged to a valid period orbit with high accuracy, it was not a Halo-type orbit.

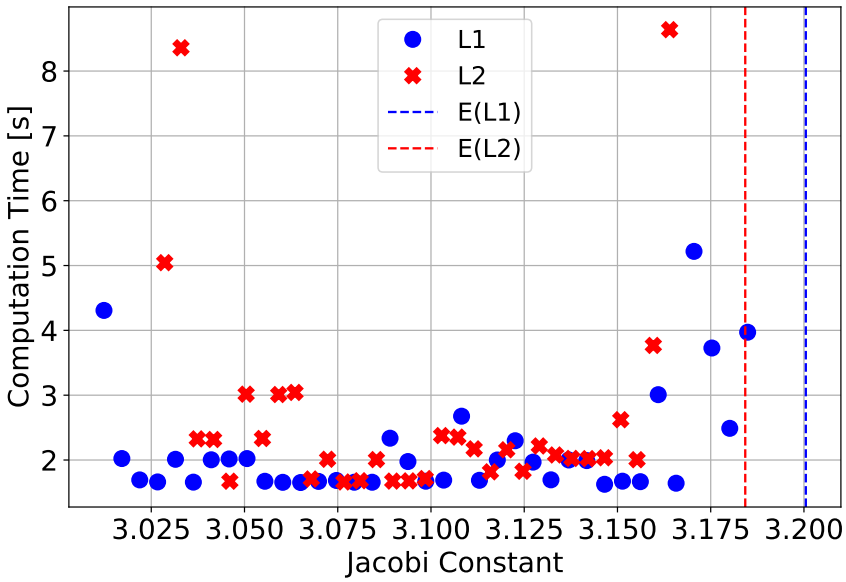


Figure 11. Computational time of the TFC method for the solution of "northern" Halo orbits around L1 and L2 plotted in Figure 9. At first glance, it can easily be seen that the computation of these orbits took around twice as long to compute as the Lyapunov orbits. One cause of the increased computation time is that the system of equations increased since more points and basis functions were need in the computation of these orbits.

5. Conclusions

In this paper, we implemented a functional interpolation method, based on the Theory of Functional Connections (TFC), to computed periodic orbits, i.e., Lyapunov and Halo

orbits, in the circular-restricted three-body problem. The TFC approach allowed for the analytical embedding of the periodic constraint, such that the orbit's initial and final position and velocity repeated after some period T. This process reduces the search space of the numerical algorithm to only those trajectories satisfying the periodic constraint.

In general, the method produced comparable results to the differential corrector method; however, the ease of implementation of the TFC method provides one benefit. For example, when using the TFC method, the user only needs to code the loss vector, Equation (20), and the parameter update is handled through automatic differentiation. For readers interested in this implementation, the authors direct them to the TFC GitHub (https://github.com/leakec/tfc, accessed on 15 February 2021) [24], where both codes for the Lyapunov (https://github.com/leakec/tfc/tree/main/examples/Hunter_Johnston_Dissertation/Chapter_4/Example_4_8, accessed on 15 February 2021) and Halo (https://github.com/leakec/tfc/tree/main/examples/Hunter_Johnston_Dissertation/Chapter_4/Example_4_9, accessed on 15 February 2021) orbit cases are provided. In addition to the code used to produce the results of this paper, this toolbox provides examples to the solution of a wide variety of ODEs and PDEs.

In all, this work was the first time that the TFC method was utilized to enforce the constraints of periodic orbits. Future work in this area should focus on two major aspects in order to increase speed, accuracy, and robustness of the technique, which are summarized below:

- **Definition of the free function.** In this study, Chebyshev orthogonal polynomials were used to define the free function; however, a large number of basis terms were needed to accurately describe the trajectories for both Lyapunov and Halo orbits. One idea to remedy this is to use a hybrid basis composed of terms to capture the periodic and non-periodic portions separately.
- **Faster least-squares implementation.** As seen in Figure 6, the major bottleneck with computation time lies in the least-squares portion, which is an order of magnitude slower than the evaluation of the loss vector and Jacobian combined. The authors acknowledge that a more efficient algorithm than NumPy's `pinv()` could be implemented.

Author Contributions: Conceptualization, H.J. and M.W.L.; Formal analysis, H.J.; Methodology, H.J.; Software, H.J.; Supervision, M.W.L. and D.M.; Validation, H.J.; Writing—original draft, H.J. and M.W.L.; Writing—review and editing, H.J., M.W.L. and D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was carried out in part at the Texas A&M University, supported by a NASA Space Technology Research Fellowship, Johnston (NSTRF 2019) Grant#: 80NSSC19K1149. This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code presented in this paper is available at: https://github.com/leakec/tfc/, accessed on 15 February 2021.

Acknowledgments: Many thanks to JPL and Section 392 for hosting the first author's summer internship in 2020.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CR3BP	circular-restricted three-body problem
ELM	Extreme Learning Machine
ISEE3	International Sun-Earth Explorer 3
JPL	Jet Propulsion Laboratory
NASA	National Aeronautics and Space Administration
NN	neural networks
NSTRF	NASA Space Technology Research Fellowship
ODE	ordinary differential equation
PDE	partial differential equation
SVD	singular-value decomposition
TFC	Theory of Functional Connections

References

1. Farquhar, R.W. The Control and Use of Libration-Point Satellites. Ph.D. Thesis, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA, 1968.
2. Breakwell, J.V.; Brown, J.V. The 'Halo'family of 3-Dimensional Periodic Orbits in the Earth-Moon Restricted 3-Body Problem. *Celest. Mech.* **1979**, *20*, 389–404. [CrossRef]
3. Mortari, D. The Theory of Connections: Connecting Points. *Mathematics* **2017**, *5*, 57. [CrossRef]
4. Leake, C.; Johnston, H.; Mortari, D. The Multivariate Theory of Functional Connections: Theory, Proofs, and Application in Partial Differential Equations. *Mathematics* **2020**, *8*, 1303. [CrossRef]
5. Howell, K. Three-dimensional, Periodic, Halo Orbits. *Celest. Mech.* **1984**, *32*, 53–71. [CrossRef]
6. Mortari, D.; Johnston, H.; Smith, L. High Accuracy Least-squares Solutions of Nonlinear Differential Equations. *J. Comput. Appl. Math.* **2019**, 293–307. [CrossRef] [PubMed]
7. Johnston, H.; Mortari, D. Least-squares Solutions of Boundary-value Problems in Hybrid Systems. *J. Comput. Appl. Math.* **2021**, *393*, 113524. [CrossRef]
8. Furfaro, R.; Mortari, D. Least-squares Solution of a Class of Optimal Space Guidance Problems via Theory of Connections. *Acta Astronaut.* **2020**, *168*, 92–103. [CrossRef]
9. Johnston, H.; Schiassi, E.; Furfaro, R.; Mortari, D. Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies via Theory of Functional Connections. *J. Astronaut. Sci.* **2020**, *67*, 1521–1552. [CrossRef] [PubMed]
10. Mortari, D.; Mai, T.; Efendiev, Y. Theory of Functional Connections Applied to Nonlinear Programming under Equality Constraints. In Proceedings of the 2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, USA, 11–15 August 2019; Paper AAS 19-675.
11. Leake, C.; Mortari, D. Deep Theory of Functional Connections: A New Method for Estimating the Solutions of Partial Differential Equations. *Mach. Learn. Knowl. Extr.* **2020**, *2*, 37–55. [CrossRef] [PubMed]
12. Leake, C.; Johnston, H.; Smith, L.; Mortari, D. Analytically Embedding Differential Equation Constraints into Least Squares Support Vector Machines Using the Theory of Functional Connections. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 1058–1083. [CrossRef] [PubMed]
13. Gil, A.; Segura, J.; Temme, N. *Numerical Methods for Special Functions*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2007; pp. 51–80. [CrossRef]
14. Lanczos, C. *Applied Analysis*; Dover Publications, Inc.: New York, NY, USA, 1957; p. 453.
15. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme Learning Machine: Theory and Applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
16. Schiassi, E.; Leake, C.; De Florio, M.; Johnston, H.; Furfaro, R.; Mortari, D. Extreme Theory of Functional Connections: A Physics-Informed Neural Network Method for Solving Parametric Differential Equations. *arXiv* **2020**, arXiv:2005.10632.
17. Schiassi, E.; D'Ambrosio, A.; Johnston, H.; de Florio, M.; Furfaro, R.; Curti, F.; Mortari, D. Physics-Informed Extreme Theory of Functional Connections Applied to Optimal Orbit Transfer. In Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, USA, 9–13 August 2020; AAS 20-524.
18. Lanczos, C. Applied Analysis. In *Progress in Industrial Mathematics at ECMI 2008*; Dover Publications, Inc.: New York, NY, USA, 1957; p. 504.
19. Wright, K. Chebyshev Collocation Methods for Ordinary Differential Equations. *Comput. J.* **1964**, *6*, 358–365. [CrossRef]
20. Frostig, R.; Johnson, M.; Leary, C. Compiling Machine Learning Programs via High-level Tracing. In Proceedings of the SysML Conference, Stanford, CA, USA, 15–16 February 2018.
21. Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M.J.; Leary, C.; Maclaurin, D.; Wanderman-Milne, S. JAX: Composable Transformations of Python+NumPy Programs. 2018. Available online: http://github.com/google/jax (accessed on 5 February 2021).

22. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic Differentiation in Machine Learning: A Survey. *J. Mach. Learn. Res.* **2018**, *18*, 1–43.
23. Richardson, D.L. Analytic Construction of Periodic Orbits about the Collinear Points. *Celest. Mech.* **1980**, *22*, 241–253. [CrossRef]
24. Leake, C.; Johnston, H. TFC: A Functional Interpolation Framework. 2020. Available online: https://github.com/leakec/tfc (accessed on 15 February 2021).