# Exploring Periodic Computation and Invariant Manifolds within the Circular Restricted Three-body Problem

Joseph D. Carpinelli [*]

Periodic orbits within the Circular Restricted Three-body Problem (CR3BP) are more difficult to find than periodic orbits within the Restricted Two-body Problem (R2BP). While any elliptical or circular R2BP orbit is periodic, the periodic subset of CR3BP orbits is far more selective. Periodic CR3BP orbit-finding algorithms have been known for decades, but their implementations are not easily found in free and open-source software – the few implementations which do exist in free and open-source software don't hold generally. Instead, publicly accessible implementations may find periodic orbits for one particular CR3BP system (e.g. Sun-Earth), or may find periodic orbits which have non-zero z-amplitudes. General solutions for periodic CR3BP orbits are incredibly useful, as CR3BP analysis is can be used as an approximate starting-point during mission design. Invariant manifolds about periodic CR3BP orbits can provide low-cost interplanetary transfers when compared with traditional transfer designs. This paper introduces a periodic CR3BP orbit-finding implementation that ties together known methods and existing code implementations; as a result, this implementation works generally. The numerical Halo orbit solver is used to highlight several Halo orbit families in familiar CR3BP systems, and calculate invariant manifolds about Halo orbits. Interplanetary manifold-based transfer designs are explored, and challenges that inhibit methods for finding intersections are presented. All source code is available on GitHub as part of GeneralAstrodynamics.jl, an astrodynamics package written with the Julia programming language.

## Contents

## I. Nomenclature

| | | |
|---|---|---|
| $\vec{r}$ | = | Spacecraft position |
| $\vec{v}$ | = | Spacecraft velocity |
| $\Phi_m$ | = | Monodromy matrix |
| $\Phi$ | = | State transition matrix |
| R2BP | = | Restricted Two-body Problem |
| CR3BP | = | Circular Restricted Three-body Problem |
| $S_L \triangleq a$ | = | Distance between CR3BP bodies |
| $S_T$ | = | Time scale factor |
| $\mu^\star$ | = | Normalized CR3BP mass parameter |
| $\vec{r}^\star$ | = | Normalized spacecraft position |
| $\vec{v}^\star$ | = | Normalized spacecraft velocity |
| $\vec{a}^\star$ | = | Normalized spacecraft acceleration |
| $r_i$ | = | Nondimensional $x$ position of $i_{\text{th}}$ body |
| $H$ | = | CR3BP potential energy Hessian matrix |
| $M$ | = | Monodromy matrix |

## II. Introduction

As space exploration targets shift from Earth's moon to Mars and beyond, low-cost trajectory designs are increasingly important [1]. One such fam-

[*]M.S. Candidate, Department of Aerospace Engineering, University of Maryland, College Park

ily of low-cost trajectory designs utilizes invariant manifolds about Lagrange points [2]. Halo orbits can be estimated analytically with an expansion, as originally shown by Richardson [3] [4]. A known numerical algorithm can iterate on non-periodic initial Halo orbit conditions to produce numerically periodic Halo orbits [5]. When placed in series, these two algorithms provide a proverbial black box for astrodynamicists: given desired physical features (orbit amplitude, phase, etc.), the analytical algorithm can produce a Halo orbit estimate for the numerical algorithm to iterate on [2]. These algorithms were recently implemented with the Julia programming language [6] [7] [8].

This paper presents technical and implementation details, lessons learned, and periodic orbit and manifold results for common CR3BP systems in our solar system. First, an overview of the Circular Restricted Three-body Problem, and periodic CR3BP orbits, is provided. Methods for calculating periodic CR3BP orbits are briefly discussed, and a Julia implementation is introduced.

Next, a short catalog of Sun-Earth, Sun-Mars, and Sun-Jupiter Halo orbits is presented using the developed Halo solvers. This catalog will provide parameters for each Halo orbit, and initial conditions and orbital periods from which to propagate. These catalogues orbits may be a helpful reference for future engineers who wish to explore periodic orbits within CR3BP dynamics with applications in low-cost mission design.

Visualizations for stable and unstable manifolds near each chosen Halo orbit are shown. Points along the Halo orbits' stable manifolds are found, and transfers between LEO and these stable manifolds are defined. These manifolds are then used to generate interplanetary transfers. Methods and challenges relating to manifold-based transfer design are presented and discussed.

All code written for this project is included in the open-source Julia package titled GeneralAstrodynamics.jl [7]. A Pluto (Jupyter-like) notebook with code examples, and an informal project summary is provided [9] [6].

## III. CR3BP Overview

The Circular Restricted Three-body Problem is a simplified dynamical model of an infinitesimally small spacecraft traveling near two finite-mass celestial bodies. All masses are described as point masses. The system's barycenter is the center of mass of the two celestial bodies, and both celestial bodies are constrained to travel in a circle about an inertial frame placed at the system barycenter. CR3BP spacecraft dynamics are typically described in the Synodic frame, which is placed at the barycenter of the system, and rotates about its Z axis such that each celestial body is fixed on the Synodic X axis. CR3BP spacecraft dynamics are also typically described with normalized units, as shown in equations (1) (2) (3) (4). Normalized equations of motion for a spacecraft within the CR3BP are shown in equations (5) (6) (7) (8) (9). Note that $\mu_2 \leq \mu_1$ by definition.

Normalized Synodic CR3BP State

$$\mu^\star = \frac{\mu_2}{\mu_1 + \mu_2} \tag{1}$$

$$S_t = \sqrt{\frac{a^3}{\mu_1 + \mu_2}} \tag{2}$$

$$\overrightarrow{r}^\star \triangleq \begin{bmatrix} x^\star & y^\star & z^\star \end{bmatrix}^T = \frac{1}{S_L}\overrightarrow{r} \tag{3}$$

$$\overrightarrow{v}^\star \triangleq \begin{bmatrix} \dot{x}^\star & \dot{y}^\star & \dot{z}^\star \end{bmatrix}^T = \frac{1}{S_T}\overrightarrow{v} \tag{4}$$

Normalized Synodic CR3BP Dynamics

$$r_1 = \sqrt{(x^\star + \mu^\star)^2 + (y^\star)^2 + (z^\star)^2} \tag{5}$$

$$r_2 = \sqrt{(x^\star + \mu^\star - 1)^2 + (y^\star)^2 + (z^\star)^2} \tag{6}$$

$$\ddot{x}^\star = 2\dot{y}^\star + x^\star - \frac{(1-\mu^\star)(x^\star + \mu^\star)}{r_1^3} - \frac{\mu^\star(x^\star - 1 + \mu^\star)}{r_2^3} \tag{7}$$

$$\ddot{y}^\star = -2\dot{x}^\star + y^\star - \frac{(1-\mu^\star)y^\star}{r_1^3} - \frac{\mu^\star y^\star}{r_2^3} \tag{8}$$

$$\ddot{z}^\star = -\frac{(1-\mu^\star)z^\star}{r_1^3} - \frac{\mu^\star z^\star}{r_2^3} \tag{9}$$

$$\dot{\Phi} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} \\ H \triangleq \frac{\partial^2 U}{\partial \overrightarrow{r}^2} & \begin{bmatrix} 0 & 2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \Phi \tag{10}$$

$$\begin{aligned} H_{11} = 1 &- \frac{\mu}{r_2^3} - \frac{(1-\mu)}{r_1^3} + \frac{\frac{3}{4}\left(-2 + 2x^\star + 2\mu\right)^2 \mu}{r_2^5} \\ &- \frac{\frac{3}{2}\left(-x^\star - \mu\right)\left(2x^\star + 2\mu\right)\left(1-\mu\right)}{r_1^5} \end{aligned} \tag{11}$$

$$H_{12} = \frac{\frac{3}{2}y^\star \mu \left(-2 + 2x^\star + 2\mu\right)}{r_2^5} - \frac{3y^\star\left(-x^\star - \mu\right)\left(1-\mu\right)}{r_1^5} \tag{12}$$

$$H_{13} = \frac{\frac{3}{2}z^\star\mu\left(-2+2x^\star+2\mu\right)}{r_2^5} - \frac{3z^\star\left(-x^\star-\mu\right)\left(1-\mu\right)}{r_1^5} \tag{13}$$

$$H_{21} = \frac{\frac{3}{2}y^\star\mu\left(-2+2x^\star+2\mu\right)}{r_2^5} - \frac{3y^\star\left(-x^\star-\mu\right)\left(1-\mu\right)}{r_1^5} \tag{14}$$

$$H_{22} = 1-\frac{\mu}{r_2^3}-\frac{(1-\mu)}{r_1^3}+\frac{3\left(y^\star\right)^2\mu}{r_2^5}+\frac{3\left(y^\star\right)^2(1-\mu)}{r_1^5} \tag{15}$$

$$H_{23} = \frac{3y^\star z^\star\mu}{r_2^5} + \frac{3y^\star z^\star(1-\mu)}{r_1^5} \tag{16}$$

$$H_{31} = \frac{\frac{3}{2}z^\star\mu\left(-2+2x^\star+2\mu\right)}{r_2^5} - \frac{3z^\star\left(-x^\star-\mu\right)\left(1-\mu\right)}{r_1^5} \tag{17}$$

$$H_{32} = \frac{3y^\star z^\star\mu}{r_2^5} + \frac{3y^\star z^\star(1-\mu)}{r_1^5} \tag{18}$$

$$H_{33} = -\frac{\mu}{r_2^3}-\frac{(1-\mu)}{r_1^3}+\frac{3\left(z^\star\right)^2\mu}{r_2^5}+\frac{3\left(z^\star\right)^2(1-\mu)}{r_1^5} \tag{19}$$

## IV. Periodic CR3BP Orbits

Planar ($z \equiv 0$) periodic orbits about Lagrange point within CR3BP dynamics are known as Lypunov orbits, while periodic CR3BP orbits about with non-zero $Z$ components are known as Halo orbits [4]. Periodic CR3BP orbits are desirable for many reasons, including eclipse avoidance [10]. Of course, orbits within the CR3BP are not guaranteed to be periodic. An estimated analytical periodic orbit solution can be found with a third-order expansion, as originally shown by Richardson [3] [4] [2]. Howell developed an iterative algorithm find a numerically periodic CR3BP orbit [5] [4] [2]. The analytical and iterative periodic orbit-finding algorithms are described in more detail in the remainder of this section.

### A. Analytical Solution

Approximate analytical solutions exist for periodic orbits within the Circular Restricted Three-body Problem. CR3BP dynamics can be written as a polynomial expansion [4]. Richardson used a third-order expansion to develop an analytical solution for periodic CR3BP orbits about L1 or L2; the solution involves changing coordinates to be centered at the desired Lagrange point and normalized to the distance between the lagrange point and the less-massive celestial body [4] [3] [2]. This analytical solution is described in far more detail by Rund and Koon et al [2] [4]. Here, the reader should know two important points about this analytical method. First, periodic orbits can be completely described by their $Z$-axis amplitudes. This is a consequence of a $X$ and $Z$ amplitude constraint; given a Halo orbit's $X$ (or $Z$) amplitude, one has everything they need to calculate the orbit's analytical $Z$ (or $X$) amplitude. Second, analytical solution is only an approximation; propagating any analytical Halo orbit solution does not result in a numerically periodic orbit [4]. The latter point is shown by Fig.1.

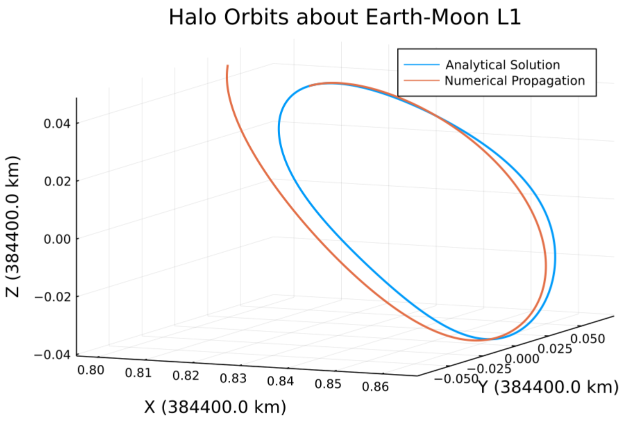

Fig. 1 Numerically propagated analytical solution

### B. Numerical Solution

As shown previously, the analytical Halo orbit approximation is not numerically periodic when provided as an initial condition to a numerical integrator. A differential correction scheme can be used to find a numerically periodic orbit within CR3BP dynamics, as described by Howell, summarized by Rund and Mireles, and implemented in private code written by Barbee, and with publicly available notes and code by Mireles [5] [2] [11] [12] [13]. There are several flavors of this Newton-iteration correction algorithm. Each flavor typically chooses two of the following initial condition components to iterate on: $x_0$, $z_0$, $\dot{y}_0$, $\frac{T_0}{2}$. Barbee's private Lyapunov orbit finding implementation chooses $\dot{y}_0$ and $\frac{T_0}{2}$ [11]. Mireles' publicly available implementation chooses three values: $z_0$, $\dot{y}_0$, and $\frac{T_0}{2}$ [13]. For this project, Mireles' approach was combined with Barbee's Lyapunov orbit computation aproach. More detail about these flavors, and the differential correction scheme more broadly, is provided in the remainder of this subsection.

Broadly, the differential correction scheme works as follows. Given an initial Halo orbit guess of the form provided by equations (20) and (21), and an initial Halo orbit period $T_0$, propagate the orbit for $\frac{T_0}{2}$, and record the final state transition matrix $\Phi$, Cartesian position $\overrightarrow{r}^\star$, velocity $\overrightarrow{v}^\star$, and acceleration $\overrightarrow{a}^\star$. Here, we check if the final $x$-axis and $z$-axis velocities are within some tolerance of zero; $10^{-8}$ is one reasonable tolerance to use. If the final $x$-axis and $z$-axis velocities are outside of your chosen tolerance, then a new Halo orbit initial condition must be found. There are several differential correction options for refining your Halo orbit initial conditions. After applying one such method, and generating new initial conditions, this process is repeated until some maximum number of iterations is reached, or final $x$-axis and $z$-axis velocities are successfully within some tolerance of zero.

$$\overrightarrow{r}^\star = \begin{bmatrix} x_0 & 0 & z_0 \end{bmatrix} \tag{20}$$

$$\overrightarrow{v}^\star = \begin{bmatrix} 0 & \dot{y}_0 & 0 \end{bmatrix} \tag{21}$$

Implementation used by Howell, Rund

Howell presents two options, as shown in equations (22) and (23) [5]. Equation (22) iterates on the initial $z_0$ and $\dot{y}_0$ conditions, and equation (23) iterates on the initial $x_0$ and $\dot{y}_0$ conditions. These corrections do not refine your $\frac{T_0}{2}$ guess, so they rely on your numerical integration stopping when the orbit's $y$-axis component enters within some tolerance of zero; this timepoint is the assumed half-period $\frac{T_0}{2}$ of the Halo orbit at this iteration. Investigations as part of this project found this approach problematic; approximate Halo orbit initial conditions produced by the previously discussed analytical algorithm may not cross the $X - Z$ plane in any reasonable time, and an arbitrary tolerance that doesn't produce "false positives" may be difficult to select. Still, others have implemented working versions of this particular algorithm. Rund summarized this correction scheme, and used equation (23) to calculate Halo orbit and invariant manifold results in a 2018 MS thesis [2].

$$\begin{gathered} F = \left\{ \begin{bmatrix} \Phi_{43} & \Phi_{45} \\ \Phi_{63} & \Phi_{65} \end{bmatrix} - \frac{1}{\dot{y}_f} \begin{bmatrix} \ddot{x}_f \\ \ddot{z}_f \end{bmatrix} \begin{bmatrix} \Phi_{23} & \Phi_{25} \end{bmatrix} \right\} \\ \begin{bmatrix} z_0 \\ \dot{y}_0 \end{bmatrix} \leftarrow \begin{bmatrix} z_0 \\ \dot{y}_0 \end{bmatrix} - F^{-1} \begin{bmatrix} \dot{x}_f \\ \dot{z}_f \end{bmatrix} \end{gathered} \tag{22}$$

$$\begin{gathered} F = \left\{ \begin{bmatrix} \Phi_{41} & \Phi_{45} \\ \Phi_{61} & \Phi_{65} \end{bmatrix} - \frac{1}{\dot{y}_f} \begin{bmatrix} \ddot{x}_f \\ \ddot{z}_f \end{bmatrix} \begin{bmatrix} \Phi_{21} & \Phi_{25} \end{bmatrix} \right\} \\ \begin{bmatrix} x_0 \\ \dot{y}_0 \end{bmatrix} \leftarrow \begin{bmatrix} x_0 \\ \dot{y}_0 \end{bmatrix} - F^{-1} \begin{bmatrix} \dot{x}_f \\ \dot{z}_f \end{bmatrix} \end{gathered} \tag{23}$$

Implementation used by Barbee

Barbee's Lyapunov orbit-finding MATLAB implementation uses a slightly different approach. Rather than refine $x_0$ and $\dot{y}_0$, his implementation iterates on $\dot{y}_0$ and $\frac{T_0}{2}$ [11]. This approach works for Lyapnuov orbits: periodic orbits with $z \equiv 0$. The iteration of $\frac{T_0}{2}$ removes the reliance on accurate numerical integration terminal when $y$ reaches some tolerance of the $x{-}z$ plane; the integration can simply terminate when the current half-period guess, $\frac{T_0}{2}$, is reached. Investigations as part of this project found that correcting $\frac{T_0}{2}$, as opposed to relying on an integration termination condition to refine $\frac{T_0}{2}$, produces a far more robust orbit-finding implementation. Barbee's chosen initial condition refining approach is shown in equation (24).

$$\begin{gathered} F = \begin{bmatrix} \Phi_{25} & \dot{y}_f \\ \Phi_{45} & \ddot{x}_f \end{bmatrix} \\ \begin{bmatrix} \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} \leftarrow \begin{bmatrix} \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} - F^{-1} \begin{bmatrix} y_f \\ \dot{x}_f \end{bmatrix} \end{gathered} \tag{24}$$

At this point, we've discussed three flavors of the periodic-orbit-finding differential correction algorithm. The first two implementations, as originally presented by Howell in [5], discussed do not refine the half-period guess for the periodic orbit; this results in a potential implementation weakness when used in tandem with the analytical solution, as previously discussed. The third implementation, as provided by Barbee in [11], removes this weakness by correcting the half-period initial guess with each iteration. Still, Barbee's implementation does not refine three-dimensional (Halo) orbit guesses. Combining the best aspects of each flavor would allow for Lyapunov and Halo orbit computation, without relying on numerical integration termination conditions. Mireles provides one such implementation in publicly available notes and code [12] [13].

Implementation used by Mireles

Mireles' implementation iterates on three initial conditions: $z_0$, $\dot{y}_0$, and $\frac{T_0}{2}$ [12] [13]. This approach to initial condition correction is shown in equation (25). Investigations as part of this project found that the approach described by equation (25) is robust when applied to Halo orbit guesses provided by the previously discussed analytical solution. The implementation of this approach developed for this project failed to converge on numerically periodic Lyapunov (two-dimensional) orbits. This is likely a result of the $z_0$ iteration used. Lyapunov orbits have $z \equiv 0$ by definition, so an initial condition with $z_0 = 0$ would not produce any correction. In other words, this scheme relies on using $z_0$ corrections to converge on a numerically periodic orbit. If $z \equiv 0$, then we've lost a degree of freedom for this correction algorithm. Next, one final implementation is provided that corrects this issue.

$$F = \begin{bmatrix} \Phi_{43} & \Phi_{45} & \ddot{x}_f \\ \Phi_{63} & \Phi_{65} & \ddot{z}_f \\ \Phi_{23} & \Phi_{25} & \dot{y}_f \end{bmatrix}$$
$$\begin{bmatrix} z_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} \leftarrow \begin{bmatrix} z_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} - F^{-1} \begin{bmatrix} \dot{x}_f \\ \dot{z}_f \\ y_f \end{bmatrix} \tag{25}$$

General Implementation

Recall that Howell presented two iteration options: one to refine $x_0$, and another to refine $z_0$. We can use this approach in combination with Mireles' implementation to produce a general periodic orbit solver that works for both Lyapunov and Halo orbits. This approach is also discussed in Mireles' course notes [12]. The general orbit solver implementation can use an if, else block to check whether the initial condition is that of an approximate Lyapunov orbit ($z_0 = 0$), or a Halo orbit ($z_0 \neq 0$). If the initial condition is an approximate Lyapunov orbit, then equation (26) should be used. Otherwise, investigations as part of this project found equation (27) produced robust results.

$$F = \begin{bmatrix} \Phi_{41} & \Phi_{45} & \ddot{x}_f \\ \Phi_{61} & \Phi_{65} & \ddot{z}_f \\ \Phi_{21} & \Phi_{25} & \dot{y}_f \end{bmatrix}$$
$$\begin{bmatrix} x_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} \leftarrow \begin{bmatrix} x_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} - F^{-1} \begin{bmatrix} \dot{x}_f \\ \dot{z}_f \\ y_f \end{bmatrix} \tag{26}$$

$$F = \begin{bmatrix} \Phi_{43} & \Phi_{45} & \ddot{x}_f \\ \Phi_{63} & \Phi_{65} & \ddot{z}_f \\ \Phi_{23} & \Phi_{25} & \dot{y}_f \end{bmatrix}$$
$$\begin{bmatrix} z_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} \leftarrow \begin{bmatrix} z_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} - F^{-1} \begin{bmatrix} \dot{x}_f \\ \dot{z}_f \\ y_f \end{bmatrix} \tag{27}$$

## C. Halo Orbit Results

This final implementation (the combination of equations (26) and (27)) is robust for Lyapunov and Halo orbits about collinear Lagrange points. This implementation was impemented within GeneralAstrodynamics.jl, and was used with the previously discussed analytical algorithm to generate periodic orbits for the remainder of this project [7]. The analytical algorithm takes physical orbit attributes as inputs: $z$-axis amplitude, phase angle, Lagrange point ($L1$ or $L2$), and hemisphere (northern, or southern) [2]. The approximate analytical solution generated with Richardson's analytical algorithm can then be used as an initial condition for the numerical algorithm [2] [3]. When placed in series in this way, the analytical and numerical orbit-finding algorithms discussed so far form a kind of "black box"; the user specifies desired Halo orbit attributes, and the algorithms provide a numerically periodic Halo orbit initial condition. Fig.2 shows one result of the analytical and numerical algorithms working in tandem; the numerical algorithm found a periodic orbit nearby the approximate analytical solution.

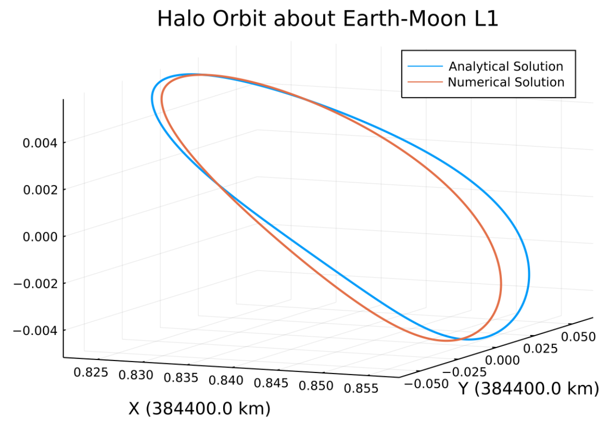

Fig. 2 Analytical and Iterative Numerical solutions

While periodic orbits within CR3BP dynamics are catalogued in literature, they can be difficult to find. A catalog of over 130,000 periodic orbits within our solar system has been posted to GitHub as a collection of CSV files [6]. The logged files are not optimized for disk space; they are instead optimized for clarity. Each row within the logged files provides normalized mass parameter, Lagrange point to orbit, Jacobi constant, normalized $z$-axis amplitude, orbital period, and normalized, Synodic Cartesian states. Periodic orbits were catalogued for Sun-Venus, Sun-Earth, Earth-Moon, Sun-Mars, Sun-Jupiter, and Sun-Saturn CR3BP systems. A fraction of the computed Halo orbits are provided in the Appendix, Table1.

A collection of periodic orbits can be computed with slightly varying $z$-axis amplitudes; this can be referred to as a family of Halo orbits [2]. Example families of periodic orbits within the Earth-Moon, Sun-Mars, and Sun-Jupiter systems are presented in Fig.3, Fig.4, and Fig.5. Note that both Fig.4 and Fig.5 look very similar because all orbits are plotted with nondimensional units.

In summary, several Halo orbit solvers have been pieced together to form a robust, open-source Julia implementation. This implementation produced all results presented so far, and was also used to explore invariant manifolds within CR3BP dynamics. The remainder of this paper summarizes the existance of invariant manifolds, computational methods, applications in mission design, and methods and associated challenges in computing manifold-based interplanetary transfers.

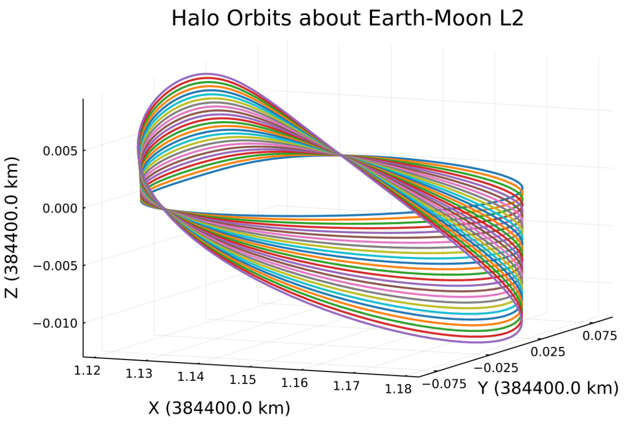

Fig. 3 Famly of Halo orbits about Earth-Moon L2

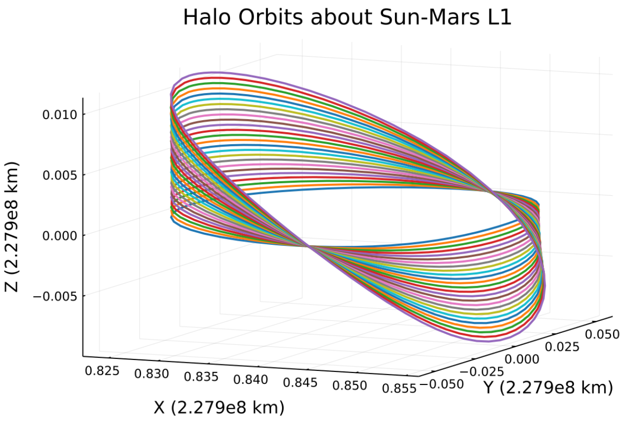

Fig. 4 Famly of Halo orbits about Sun-Mars L1

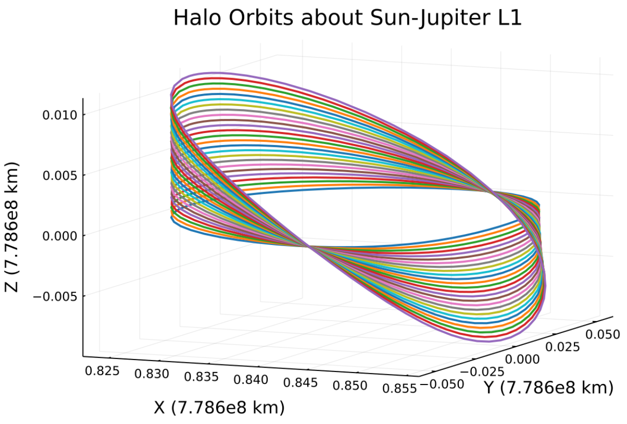

Fig. 5 Famly of Halo orbits about Sun-Jupiter L1

## V. Invariant Manifolds

Before discussing invariant manifolds, a brief discussion of general nonlinear dynamics may be helpful.

### A. Nonlinear Dynamics Review

Lagrange points are the equilibrium points of the nonlinear CR3BP dynamics. The phase space around these Lagrange points will have stability characteristics, e.g. stable focus, unstable focus, and saddle point behavior. As discussed in the preceding sections, the phase space around Lagrange points can also feature periodic orbits, e.g. Lyapunov and Halo orbits. Once again, similar to any nonlinear dynamical system, the Jacobian of a CR3BP state vector is the local linearization of the CR3BP dynamics evaluated at that particular state vector. We can use the local linearization to characterize the stability properties of the dynamics at that point in the phase space, and points along periodic orbits are no exception. States along periodic orbits can benefit from some reduced computational expense by using the Monodromy matrix, $\boldsymbol{M}$ [2]. We can find $\boldsymbol{M}$ by propagating a periodic orbit's initial conditions for one period, with the dynamics of the state transition matrix included. The state transition matrix should be initialized to a $6 \times 6$ identity matrix at the start of the numerical integration. The final state transition matrix in this propagation is the Monodromy matrix, and we can use this matrix to evaluate local linear modes at each point along the periodic orbit, as opposed to computing the state transition matrix at each point we wish to evaluate.

### B. Calculating Invariant Manifolds

Invariant manifolds within CR3BP dynamics are tubes of trajectories that converge to, or diverge from, periodic orbits or Lagrange points [2] [14]. The term manifold comes from their three-dimensional, surface-like shapes. The term invariant applies because, once placed in a manifold, a spacecraft governed by CR3BP dynamics will stay in that manifold for all time (without external forces applied).

For this project, invariant manifolds about Halo orbits were explored. A spacecraft on a periodic orbit can be perturbed onto a trajectory within a manifold by perturbing the state vector in the direction of the Mondomy matrix's stable or unstable eigenvectors. Each Monodromy matrix should have two real eigenvalues, and two pairs of complex eigenvalues. The smaller real eigenvalue should be less than one, and the larger real eigenvalue should be greater than one [15]. As an additional check, the pair of real eigenvalues should also be multiplicative inverses of one another [12]. The eigenvector of the Monodromy matrix that is associated with the larger real eigenvalue is the unstable eigenvector $V^U$, and the eigenvector associated with the smaller real eigenvalue is the stable eigenvector $V^S$ [2]. Equations (28), (29), (30), (31), and (32) describe the perturbation required to move from a periodic orbit onto the orbit's stable manifold $X^S$, or unstable manifold $X^U$; all are pulled from Rund's thesis, and were also presented in a previous paper [2] [6]. As described in upcoming sections, I believe there is a mistake in the presented invariant manifold plots. I computed these plots by perturbing the spacecraft in the direction of the stable and unstable eigenvectors of the Monodromy matrix directly, instead of changing basis by left-multiplying the eigenvector by the Jacobian. I will look into this more, but did not have time to correct this mistake before this paper's submission.

$$V_i^S = \Phi(t_0 + t_i, t_0)V^S \tag{28}$$

$$V_i^U = \Phi(t_0 + t_i, t_0)V^U \tag{29}$$

$$X_i^S = X_i \pm \epsilon \frac{V_i^S}{|V_i^S|} \tag{30}$$

$$X_i^U = X_i \pm \epsilon \frac{V_i^U}{|V_i^U|} \tag{31}$$

$$M = \Phi(t_0 + T, t_0) \tag{32}$$

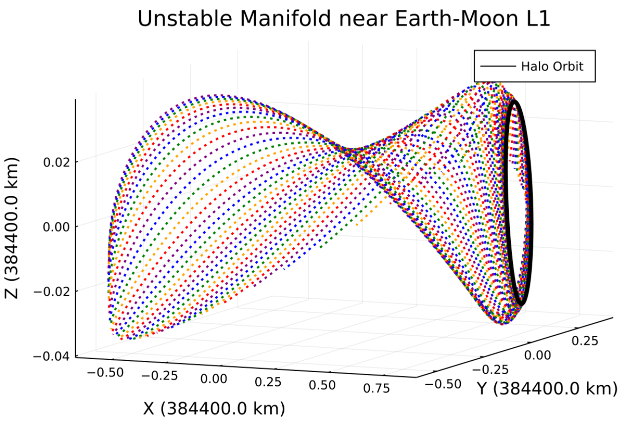

Fig. 6 Unstable Manifold near Earth-Moon L1

## VI. Manifold-based Transfers

### A. Manifolds for Interplanetary Transfers

One attempted manifold-based interplanetary transfer design follows. Given a destination Sun-Jupiter L1 Halo orbit, one wishes to travel cheaply from Earth to the destination orbit. Invariant manifolds can theoretically be used to reduce the $\Delta v$ cost for an interplanetary transfer. For this theoretical mission, the spacecraft can launch directly into a stable manifold of a Sun-Earth L2 Halo orbit. Once the spacecraft is on some trajectory within the stable manifold, CR3BP dynamics will eventually cause the spacecraft to converge on our chosen Halo orbit. The L2 lagrange point is selected because it is located near the "mouth" of the zero-velocity curves for nearby orbits; in other words, selecting an orbit at random within Sun-Earth L2's unstable manifold have a higher probability of that orbit escaping the confines of the zero velocity curves, and diverging into interplanetary space. This is a useful property for the next mission step: perturb the spacecraft off of the Sun-Earth L2 Halo orbit, and onto the orbit's unstable manifold. Next, we need to find some intersection between the Sun-Earth Halo's unstable manifold, and our destination Halo's stable manifold; the difference in velocity at this intersection can be approximated as an impulsive maneuver. Once the spacecraft is on some trajectory within our destination Halo's stable manifold, it will eventually converge to our destination Halo orbit.

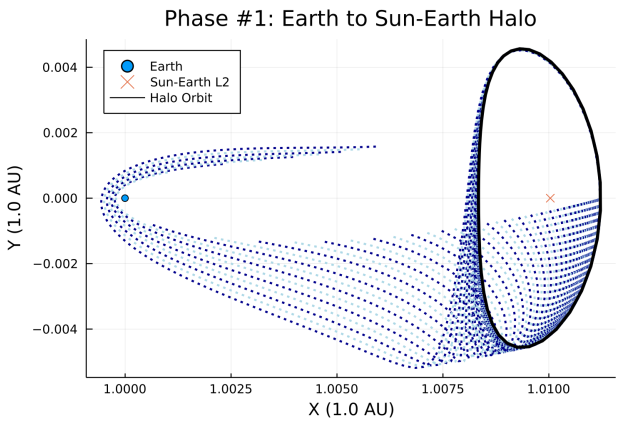

Fig. 7 Launch from Earth to Stable Manifold

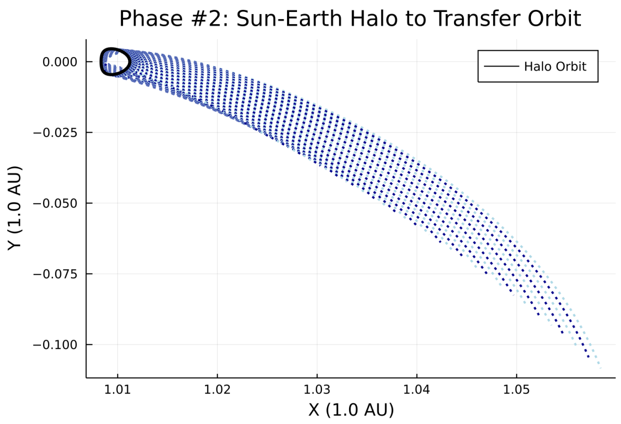

Fig. 8 Perturb Spacecraft onto Sun-Earth Unstable Manifold

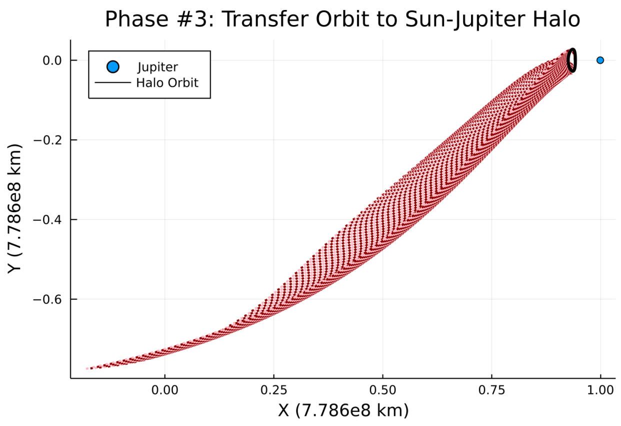

Fig. 9 Maneuver Spacecraft onto Sun-Jupiter Stable Manifold

## B. Implementation Challenges

While manifold-based transfer designs are simple in concept, there are several pitfalls that may cause project delays. This subsection will outline several pitfalls, and associated lessons learned.

### Implementation Mistakes

An implementation mistake that affected the results in this paper was discovered last-minute. Instead of following the manifold perturbation equations presented in this paper, this project's source code was written to instead perturb the spacecraft directly in the direction of the Monodromy matrix's stable and unstable eigenvectors. All manifold plots here are presented with this mistake. After briefly correcting this, all manifold plots differed greatly. More investigation will be necessary to correct this issue. I plan to do so this summer, as part of GeneralAstrodynamics.jl development efforts.

### Algorithmic Complexity

The naive solution for finding the closest distance between two manifolds is $O(n^2)$, which is far too algorithmically complex for larger manifolds, which can contain millions of Cartesian states. The naive implementation never returned, and quick calculations showed that the implementation would never return any values in a reasonable amount of time. After posting this issue to a Julia Discourse forum, I was pointed towards a Julia package called NearestNeighbors.jl [16] [17]. This package uses an algorithm that find find the closest pair of points in two large sets far faster than the naive solution. This algorithm is apparently common in computational chemistry [16]. NearestNeighbors.jl found the closest pair of Cartesian states between the two manifolds in seconds.

### Manifold Intersection

After algorithmic complexity in finding the closest pair of points was addressed, a new problem emerged; manifolds are not guaranteed to intersect. Often, the closest pair of Cartesian states that could be found were still hundreds of meters apart. This may be a consequence of the Implementation Mistakes discussed previously, but others have found similar results. Topputo et al found that manifolds do not intersect among our solar system's inner planets [14]. Rund did not rely on manifold intersection for low-cost transfer design; Rund's thesis presented manifold-based transfers to lower the cost of impulsive maneuvers to place the spacecraft on a hyperbolic, Heliocentric trajectory [2]. It's possible that manifold intersections are incredibly difficult to find for CR3BP systems that are so far apart within the Heliocentric inertial frame. Topputo et al note that transfers along manifold intersections may be best suited for specific mission conditions, such as travel between moons of one large planet – Koon's "Petit Grand Tour" of Jupiter's moons comes to mind [14] [4].

## VII. Conclusions and Forward Work

This project implemented added robustness and bug fixes to previously implented analytical and numerical periodic orbit computation algorithms within CR3BP dynamics. In a previous project, analytical and numerical algorithms were written, but produced incorrect results [6]. Now, both analytical and numerical Lyapunov and Halo orbit algorithms appear to be implemented correctly in GeneralAstrodynamics.jl, an open source Julia package with the aim of providing (at the author's highest aspirations) another Poliastro-like tool for the scientific computing and astrodynamics communities [7].

Pre-existing numerical Lyapunov and Halo orbit solver implementations were explained, and combined to form one general numerical periodic orbit solver, as described in Mireles' notes, and partially available in Mireles' publicly available code [12] [13]. This function was used to catalogue over $130,000$ periodic orbits within CR3BP systems in our solar system [6]. Several families of Halo orbits were presented, and the mathematics behind Halo orbit computation, and CR3BP dynamics, was reviewed.

The concept and computation of invariant manifolds within CR3BP dynamics was discussed. A visualization of an unstable manifold (barring implementation issues previously addressed) was presented, and a generic mission design which uses invariant manifolds to travel from Earth to Sun-Jupiter L1 was discussed.

Implementation complications around manifold-based transfers were presented, and potential fixes were addressed. One large implementation mistake was discussed – manifold perturbation calculations are incorrectly using the eigenvectors of the Monodromy matrix without left-multiplying the Jacobian at each time step. Algorithmic complexity can be an issue when finding intersections between manifolds, but the so-called "Nearest Neighbors" algorithm implemented in NearestNeighbors.jl allows for finding the minimum intersection between manifolds in seconds [17]. Finally, manifold intersections have been found to be rare in many conditions, as shown by other astrodynamicists [14], and this project's results.

Going forward, manifold perturbation calculations will be fixed and verified. If Sun-Earth L2 and Sun-Jupiter L1 Halo orbit manifold intersections are found, then a specific mission design will be outlined using only invariant manifolds. If no intersections are found, then manifolds will be used to reduce the costs of hyperbolic interplanetary transfers, as shown by Rund [2]. Development for GeneralAstrodynamics.jl will continue. Desired future features include robust coordinate frame definitions and transformations, realistic time-handling, catalogued physical constants for solar system bodies, hooks into existing ephemeris and other astrodynamics tools, and more.

## VIII. References

[1] NASA, NASA's Lunar Exploration Program Overview, 2020.

[2] Rund, M. S., "Interplanetary Transfer Trajectories Using the Invariant Manifolds of Halo Orbits," , 2018.

[3] Richardson, D., "Analytical construction of periodic orbits about the collinear points of the Sun-Earth system." asdy, 1980, p. 127.

[4] Koon, W. S., Lo, M. W., Marsden, J. E., and Ross, S. D., "Dynamical systems, the three-body problem and space mission design," Free online Copy: Marsden Books, 2008.

[5] Howell, K. C., "Three-dimensional, periodic,'halo'orbits," Celestial mechanics, Vol. 32, No. 1, 1984, pp. 53–71.

[6] Carpinelli, J., "Halo Orbit Explorations," https://github.com/cadojo/Halo-Orbit-Explorations, 2020-2021.

[7] Carpinelli, J., "GeneralAstrodynamics.jl," https://github.com/cadojo/GeneralAstrodynamics.jl, 2020.

[8] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B., "Julia: A fresh approach to numerical computing," SIAM review, Vol. 59, No. 1, 2017, pp. 65–98. URL https://doi.org/10.1137/141000671.

[9] van der Plas, F., "Pluto.jl," https://github.com/fonsp/Pluto.jl, 2020.

[10] Williams, J., Lee, D. E., Whitley, R. J., Bokelmann, K. A., Davis, D. C., and Berry, C. F., "Targeting cislunar near rectilinear halo orbits for human space exploration," 2017.

[11] Barbee, B., "target_libration_orbits_SE.m," Private Code, Unknown Publication Date.

[12] Mireles, J., "Celestial Mechanics Notes Set 5: Symmetric Periodic Orbits of the Circular Restricted Three Body Problem and their Stable and Unstable Manifolds," http://cosweb1.fau.edu/ jmireles-james/hw5Notes.pdf, 2006.

[13] Mireles, J., "EarthSunHaloOrbit_NewtonMethod.m," http://cosweb1.fau.edu/ jmireles-james/MatLabCode/EarthSunHaloOrbit_NewtonMethod.m, 2006.

[14] Topputo, F., Vasile, M., and Bernelli-Zazzera, F., "Low energy interplanetary transfers exploiting invariant manifolds of the restricted three-body problem," The Journal of the Astronautical Sciences, Vol. 53, No. 4, 2005, pp. 353–372.

[15] Gómez, G., Dynamics and Mission Design Near Libration Points: Fundamentals-The Case of Collinear Libration Points, Vol. 1, World Scientific, 2001.

[16] et al, J. C., "Closest distance between two Euclidian grids," https://discourse.julialang.org/t/closest-distance-between-two-euclidian-grids/61218, 2021.

[17] Carlsson, K., "NearestNeighbors.jl," https://github.com/KristofferC/NearestNeighbors.jl, 2021.

[18] Vallado, D. A., Fundamentals of astrodynamics and applications, Vol. 12, Springer Science & Business Media, 2001.

[19] Lara, M., Russell, R., and Villac, B., "Classification of the distant stability regions at Europa," Journal of Guidance, Control, and Dynamics, Vol. 30, No. 2, 2007, pp. 409–418.

[20] Zimovan-Spreen, E. M., Howell, K. C., and Davis, D. C., "Near rectilinear halo orbits and nearby higher-period dynamical structures: orbital stability and resonance properties," Celestial Mechanics and Dynamical Astronomy, Vol. 132, No. 5, 2020, pp. 1–25.

[21] Gómez, G., Koon, W. S., Lo, M., Marsden, J. E., Masdemont, J., and Ross, S. D., "Connecting orbits and invariant manifolds in the spatial restricted three-body problem," Nonlinearity, Vol. 17, No. 5, 2004, p. 1571.

[22] Yingjing, Q., Xiaodong, Y., Wuxing, J., and Zhang, W., "An improved numerical method for constructing Halo/Lissajous orbits in a full solar system model," Chinese Journal of Aeronautics, Vol. 31, No. 6, 2018, pp. 1362–1374.

[23] Gómez, G., Jorba, A., Masdemont, J. J., and Simo, C., Dynamics And Mission Design Near Libration Points, Vol Iii: Advanced Methods For Collinear Points, Vol. 4, World Scientific, 2001.

## IX. Appendix

A brief collection of Halo orbits, the analytical Halo orbit solver implementation, and the numerical Halo orbit solver implementation are provided here for convenience. The full results are available online, and are far more valuable than the fraction of the results presented in the subsections below.

Project Links

- GeneralAstrodynamics on GitHub
- Analytical Halo Solver Implementation
- Numerical Halo Solver Implementation
- This project, hosted on GitHub
- Catalogued Halo Orbits

```
"""
Returns an analytical solution for a Halo orbit about `L`.

__Arguments:__
- `μ`: Non-dimensional mass parameter for the CR3BP system.
- `Az`: Desired non-dimensional Z-amplitude for Halo orbit.
- `ϕ`: Desired Halo orbit phase.
- `steps`: Number of non-dimensional timepoints in returned state.
- `L`: Lagrange point to orbit (L1 or L2).
- `hemisphere`: Specifies northern or southern Halo orbit.

__Outputs:__
- Synodic position vector `r::Array{<:AbstractFloat}`
- Synodic velocity vector `v::Array{<:Abstractfloat}`.
- Halo orbit period `T`.
- Throws `ArgumentError` if L is not `1` or `2`.

__References:__
- [Rund, 2018](https://digitalcommons.calpoly.edu/theses/1853/).
"""
function analyticalhalo(μ; Az=0.00, ϕ=0.0, steps=1,
                        L=1, hemisphere=:northern)

    if L == 1
        point = first(lagrange(μ, 1))
        γ = abs(one(μ) - μ - point)
        n = collect(1:4) .* one(μ)
        c = @. (μ + (-one(1))^n * (one(μ)-μ)γ^(n+1)) / (γ^3 * (one(μ) - γ^(n+1)))
    elseif L == 2
        point = first(lagrange(μ, 2))
        γ = abs(point - one(μ) + μ)
        n = collect(1:4) .* one(μ)
        c = @. ((-one(μ))^n * μ + (-one(μ))^n * (one(μ)-μ)γ^(n+1)) / (γ^3 * (one(μ) + γ^(n+1)))
    else
        throw(ArgumentError("Only Halo orbits about L1 or L2 are supported."))
    end

    ωₚ  = √((2 - c[2] + √((9c[2]^2 - 8c[2])))/2)
    k   = (ωₚ^2 + 1 + 2c[2]) / (2ωₚ)

    d₁  = (3ωₚ^2 / k) * (k*(6ωₚ^2 - 1) - 2ωₚ)
    d₂  = (8ωₚ^2 / k) * (k*(11ωₚ^2 - 1) - 2ωₚ)
    a₂₁ = (3c[3] * (k^2 - 2)) / (4(1 + 2c[2]))
    a₂₂ = (3c[3]) / (4(1 + 2c[2]))
    a₂₃ = (-3c[3]ωₚ / (4k*d₁)) * (3k^3 * ωₚ - 6k*(k-ωₚ) + 4)
    a₂₄ = (-3c[3]ωₚ / (4k*d₁)) * (2 + 3k*ωₚ)
    b₂₁ = (-3c[3]ωₚ / (2d₁)) * (3k*ωₚ - 4)
    b₂₂ = -3c[3]*ωₚ / d₁
    d₂₁ = -c[3] / (2ωₚ^2)
    a₃₁ = (-9ωₚ / (4d₂)) * (4c[3] * (k*a₂₃-b₂₁) + k*c[4]*(4+k^2)) +
          ((9ωₚ^2 + 1 - c[2]) / (2d₂)) * (3c[3]*(2a₂₃-k*b₂₁) + c[4]*(2+3k^2))
    a₃₂ = (-9ωₚ / (4d₂)) * (4c[3] * (3k*a₂₄-b₂₂) + k*c[4]) -
          (3 / (2d₂)) * (9ωₚ^2 + 1 - c[2]) * (c[3]*(k*b₂₂+d₂₁-2a₂₄) - c[4])
    b₃₁ = (3 / (8d₂)) * 8ωₚ * (3c[3] * (k*b₂₁ - 2a₂₃) - c[4]*(2+3k^2)) +
            (3/(8d₂)) * (9ωₚ^2 + 1 + 2c[2]) * (4c[3]*(k*a₂₃-b₂₁) + k*c[4]*(4+k^2))
    b₃₂ = (9ωₚ/d₂)*(c[3]*(k*b₂₂+d₂₁-2a₂₄)-c[4]) +
            (3(9ωₚ^2 + 1 + 2c[2]) / (8d₂) * (4c[3]*(k*a₂₄-b₂₂)+k*c[4]))
    d₃₁ = (3 / (64ωₚ^2)) * (4c[3]*a₂₄ + c[4])
    d₃₂ = (3 / (64 + ωₚ^2)) * (4c[3]*(a₂₃ - d₂₁) + c[4]*(4+k^2))

    s₁  = (1 / (2ωₚ*(ωₚ*(1+k^2) - 2k))) *
            (3c[3]/2 * (2a₂₁*(k^2 - 2) - a₂₃*(k^2 + 2) - 2k*b₂₁) -
            (3c[4]/8) * (3k^4 - 8k^2 + 8))
    s₂  = (1 / (2ωₚ*(ωₚ*(1+k^2) - 2k))) *
            (3c[3]/2 * (2a₂₂*(k^2-2) + a₂₄*(k^2 + 2) + 2k*b₂₂ + 5d₂₁) +
            (3c[4]/8) * (12 - k^2))
    l₁  = (-3c[3] / 2) * (2a₂₁ + a₂₃ + 5d₂₁) - (3c[4]/8)*(12 - k^2) + 2ωₚ^2 * s₁
    l₂  = (3c[3]/2) * (a₂₄ - 2a₂₂) + (9c[4]/8) + 2ωₚ^2 * s₂
    Δ   = ωₚ^2 - c[2]

    Aᵧ  = Az / γ
    Aₓ  = √((-l₂*Aᵧ^2 - Δ) / l₁)

    ν   = 1 + s₁*Aₓ^2 + s₂*Aᵧ^2
    T   = 2π / (ωₚ*ν)
    τ   = ν .* (steps > 1 ? range(0, stop=T, length=steps) : range(0, stop=T, length=1000))

    if hemisphere == :northern
        m = 1.0
    elseif hemisphere == :southern
        m = 3.0
    else
        throw(ArgumentError("`hemisphere` must be `:northern` or `:southern`."))
    end

    δₘ = 2 - m
    τ₁ = @. ωₚ*τ + ϕ

    x = @. γ * (a₂₁*Aₓ^2 + a₂₂*Aᵧ^2 - Aₓ*cos(τ₁) + (a₂₃*Aₓ^2 -
                    a₂₄*Aᵧ^2)*cos(2τ₁) + (a₃₁*Aₓ^3 - a₃₂*Aₓ*Aᵧ^2)*cos(3τ₁)) + 1 - μ - (L == 1 ? γ : -γ)
    y = @. γ * (k*Aₓ*sin(τ₁) + (b₂₁*Aₓ^2 - b₂₂*Aᵧ^2)*sin(2τ₁) +
                    (b₃₁*Aₓ^3 - b₃₂*Aₓ*Aᵧ^2)*sin(3τ₁))
    z = @. γ * (δₘ*Aᵧ*cos(τ₁) + δₘ*d₂₁*Aₓ*Aᵧ*(cos(2τ₁)-3) +
                    δₘ*(d₃₂*Aᵧ*Aₓ^2 - d₃₁*Aᵧ^3)*cos(3τ₁))

    ẋ = @. γ * (ωₚ*ν*Aₓ*sin(τ₁) - 2ωₚ*ν*(a₂₃*Aₓ^2 - a₂₄*Aᵧ^2)*sin(2τ₁) -
                    3ωₚ*ν*(a₃₁*Aₓ^3 - a₃₂*Aₓ*Aᵧ^2)*sin(3τ₁))
    ẏ = @. γ * (ωₚ*ν*k*Aₓ*cos(τ₁) + 2ωₚ*ν*(b₂₁*Aₓ^2 - b₂₂*Aᵧ^2)*cos(2τ₁) +
                    3ωₚ*ν*(b₃₁*Aₓ^3 - b₃₂*Aₓ*Aᵧ^2)*cos(3τ₁))
    ż = @. γ * (-ωₚ*ν*δₘ*Aᵧ*sin(τ₁) - 2ωₚ*ν*δₘ*d₂₁*Aₓ*Aᵧ*sin(2τ₁) -
                    3ωₚ*ν*δₘ*(d₃₂*Aᵧ*Aₓ^2 - d₃₁*Aᵧ^2)*sin(3τ₁))

    return hcat(x, y, z)[1:steps, :], hcat(ẋ, ẏ, ż)[1:steps, :], T

end
```

Numerical Halo Implementation

```julia
"""
Returns a numerical solution for a Halo orbit about `L`.

__Arguments:__
- `μ`: Non-dimensional mass parameter for the CR3BP system.
- `Az`: Desired non-dimensional Z-amplitude for Halo orbit.
- `ϕ`: Desired Halo orbit phase.
- `L`: Lagrange point to orbit (L1 or L2).
- `hemisphere`: Specifies northern or southern Halo orbit.

__Outputs:__
- Tuple of initial states: `(r, v)` where `r::Vector{<:AbstractFloat}`, `v::Vector{<:Abstractfloat}`.
- Throws `ArgumentError` if L is not `:L1` or `:L2`

__References:__

The iterative scheme was pulled from directly from literature
and sample code, including Rund 2018,
and Dr. Mireles' lecture notes and EarthSunHaloOrbit_NewtonMewhod.m
file available on their website.
Specifically, the half-period iterative scheme (the `F` matrix
in the source code, and corresponding "next guess" computation)
was ported __directly__ from Dr. Mireles' public code and notes, which are
available online.
- [Dr. Mireles Notes](http://cosweb1.fau.edu/~jmirelesjames/hw5Notes.pdf)
- [Dr. Mireles Code](http://cosweb1.fau.edu/~jmirelesjames/notes.html)
- [Rund, 2018](https://digitalcommons.calpoly.edu/theses/1853/).
"""
function halo(μ; Az=0.0, L=1, hemisphere=:northern,
                tolerance=1e-8, max_iter=20,
                reltol=1e-14, abstol=1e-14,
                nan_on_fail = true, disable_warnings = false)

    r₀, v₀, T = analyticalhalo(μ; Az=Az, ϕ=0.0, L=L, hemisphere=hemisphere)
    r₀ = r₀[1,:]
    v₀ = v₀[1,:]
    τ  = T/2

    Φ  = Matrix{promote_type(eltype(r₀), eltype(v₀), typeof(τ))}(undef, 6, 6)

    for i ∈ 1:max_iter

        problem = ODEProblem(
            CR3BPSTMTic!,
            ComponentVector(vcat(r₀, v₀, [row for row ∈ eachrow(I(6))]...),
                            Axis(r=1:3, v=4:6, Φ₁=7:12, Φ₂=13:18,
                                    Φ₃=19:24, Φ₄=25:30, Φ₅=31:36, Φ₆=37:42)),
            (0.0, τ),
            (μ = μ,)
        )

        retcode, rₛ, vₛ, Φ = let
            sols  = solve(problem; reltol=reltol, abstol=abstol)
```

```
                final = sols.u[end]
                sols.retcode, final.r, final.v,  transpose(hcat(final.Φ₁, final.Φ₂, final.Φ₃, final.Φ₄, final.Φ₅
            end

            ∂vₛ = accel(rₛ, vₛ, μ)

            # All code in this `if, else, end` block is ported from
            # Dr. Mireles' MATLAB code, which is available on his
            # website: http://cosweb1.fau.edu/~jmirelesjames/notes.html.
            # Lecture notes, which describe this algorithm further,
            # are available for reference as well:
            # http://cosweb1.fau.edu/~jmirelesjames/hw5Notes.pdf
            if Az ≉ 0
                F = @SMatrix [
                    Φ[4,1] Φ[4,5] ∂vₛ[1];
                    Φ[6,1] Φ[6,5] ∂vₛ[3];
                    Φ[2,1] Φ[2,5]  vₛ[2]
                ]

                xᵧ = SVector(r₀[1], v₀[2], τ) - inv(F) * SVector(vₛ[1], vₛ[3], rₛ[2])

                r₀[1] = xᵧ[1]
                v₀[2] = xᵧ[2]
                τ     = xᵧ[3]
            else
                F = @SMatrix [
                    Φ[4,3] Φ[4,5] ∂vₛ[1];
                    Φ[6,3] Φ[6,5] ∂vₛ[3];
                    Φ[2,3] Φ[2,5]  vₛ[2]
                ]

                xᵧ = SVector(r₀[3], v₀[2], τ) - inv(F) * SVector(vₛ[1], vₛ[3], rₛ[2])

                r₀[3] = xᵧ[1]
                v₀[2] = xᵧ[2]
                τ     = xᵧ[3]
            end

            if abs(vₛ[1]) ≤ tolerance && abs(vₛ[3]) ≤ tolerance
                break;
            elseif τ > 5 * one(τ)
                disable_warnings || @warn "Unreasonably large halo period, $τ, ending iterations."
                !nan_on_fail || return [NaN, NaN, NaN], [NaN, NaN, NaN], NaN
                break
            elseif i == max_iter
                disable_warnings || @warn "Desired tolerance was not reached, and iterations have hit the maximu
                !nan_on_fail || return [NaN, NaN, NaN], [NaN, NaN, NaN], NaN
                break
            elseif retcode != :Success
                !disable_warnings || @warn "Integrator returned $(string(retcode))."
                !nan_on_fail || return [NaN, NaN, NaN], [NaN, NaN, NaN], NaN
                break
            end
        end
```

```
    return r₀, v₀, 2τ
end
```

| Mass Parameter | Lagrange Point | $Z$-Amplitude | Period | $x$ | $z$ | $\dot{y}$ |
|---|---|---|---|---|---|---|
| 0.012150584269940356 | 1.0 | 0.0 | 2.7536820171259744 | 0.8222791805122408 | 0.0 | 0.13799313179964737 |
| 0.012150584269940356 | 1.0 | 0.002 | 2.7430279744649004 | 0.8233905115990996 | 0.0022207698036084363 | 0.1264086161524851 |
| 0.012150584269940356 | 1.0 | 0.004 | 2.743129618348479 | 0.8233893741253737 | 0.004442307958743803 | 0.12665484442364439 |
| 0.012150584269940356 | 1.0 | 0.006 | 2.743298907640046 | 0.8233876253798795 | 0.006665380456556792 | 0.12706382260243482 |
| 0.012150584269940356 | 1.0 | 0.008 | 2.7435356656350174 | 0.8233854825357569 | 0.008890748615484967 | 0.12763347860600016 |
| 0.012150584269940356 | 1.0 | 0.01 | 2.7438396430341294 | 0.8233832430275673 | 0.011119166862915583 | 0.12836097250130557 |
| 0.012150584269940356 | 2.0 | 0.001999 | 3.4154785217654346 | 1.1203619239893596 | 0.001835091590818184 | 0.17611109647933998 |
| $3.003480593992993e-6$ | 1.0 | 0.0 | 3.057037166436106 | 0.9889069589528534 | 0.0 | 0.008529372360506582 |
| $3.003480593992993e-6$ | 1.0 | 0.002 | 3.0562630985504198 | 0.9889296115452058 | 0.0022759531712711633 | 0.009571654363317172 |
| $3.003480593992993e-6$ | 1.0 | 0.004 | 3.0408810610908192 | 0.9891686188174361 | 0.0046921863531775585 | 0.011428450586881073 |
| $3.003480593992993e-6$ | 1.0 | 0.006 | 2.9968780486251165 | 0.9897509664037121 | 0.007350439066516196 | 0.013571383652713521 |
| $3.003480593992993e-6$ | 1.0 | 0.008 | 2.8360875768267277 | 0.9909674701532162 | 0.010323704902503393 | 0.015198885580121493 |
| $3.003480593992993e-6$ | 2.0 | 0.001798 | 3.09794993304811 | 1.0080662252502852 | 0.001672550237255738 | 0.010798428273484711 |
| $3.003480593992993e-6$ | 2.0 | 0.003798 | 3.0755344619414036 | 1.0068608443606484 | 0.0035047324922114834 | 0.014513397367974044 |
| $3.2271548760451657e-7$ | 1.0 | 0.0 | 3.067790305289647 | 0.9947153604918691 | 0.0 | 0.004052940170205446 |
| $3.2271548760451657e-7$ | 1.0 | 0.0001 | 3.0713331541226148 | 0.9946990969840135 | 0.00011259909689815706 | 0.00420141728779284 |
| $3.2271548760451657e-7$ | 1.0 | 0.0002 | 3.071197879663506 | 0.9946998314896827 | 0.0002252842583036334 | 0.004214196175484041 |
| $3.2271548760451657e-7$ | 1.0 | 0.0003 | 3.0709713120763475 | 0.9947010768650656 | 0.00033814100956578 | 0.004235364341723893 |
| $3.2271548760451657e-7$ | 1.0 | 0.0004 | 3.0706517674163125 | 0.9947028642521663 | 0.00045125381930966155 | 0.004264731532729223 |
| $3.2271548760451657e-7$ | 1.0 | 0.0005 | 3.070236850446078 | 0.9947052359932343 | 0.0005647056232279373 | 0.004302040405811711 |
| $3.2271548760451657e-7$ | 1.0 | 0.0006 | 3.069723416205373 | 0.9947082444991275 | 0.0006785774056064239 | 0.004346974627389325 |
| 0.0009536838895767626 | 1.0 | 0.0 | 2.9370190457587504 | 0.9253021269565835 | 0.0 | 0.0585266341496578 |
| 0.0009536838895767626 | 1.0 | 0.002 | 2.9354795531765805 | 0.9253921878089174 | 0.0022382733013625957 | 0.05785899301066716 |
| 0.0009536838895767626 | 1.0 | 0.004 | 2.9354974958876 | 0.9254047001045744 | 0.004480217902893781 | 0.05829041425456174 |
| 0.0009536838895767626 | 1.0 | 0.006 | 2.9355234346153742 | 0.9254269085233069 | 0.006729454676907429 | 0.05899947503986548 |
| 0.0009536838895767626 | 1.0 | 0.008 | 2.935551270469769 | 0.9254607390135051 | 0.008989508064503429 | 0.059972051406433184 |
| 0.0009536838895767626 | 1.0 | 0.01 | 2.935572154787768 | 0.9255086965138954 | 0.011263768116134604 | 0.06118987977165465 |
| 0.0009536838895767626 | 2.0 | 0.001999 | 3.223538695196383 | 1.0560878711527533 | 0.001854944179107667 | 0.07004521113569465 |

Table 1 Periodic Orbits within the Earth-Moon, Sun-Earth, Sun-Mars, and Sun-Jupiter Systems