# Study: Interplanetary Superhighways

🎢 *Going to Jupiter with Julia, Halo orbits, and invariant manifolds.*

**May 2021**
**Joe Carpinelli**

Toggle Presentation

## 📚 Table of Contents



← →

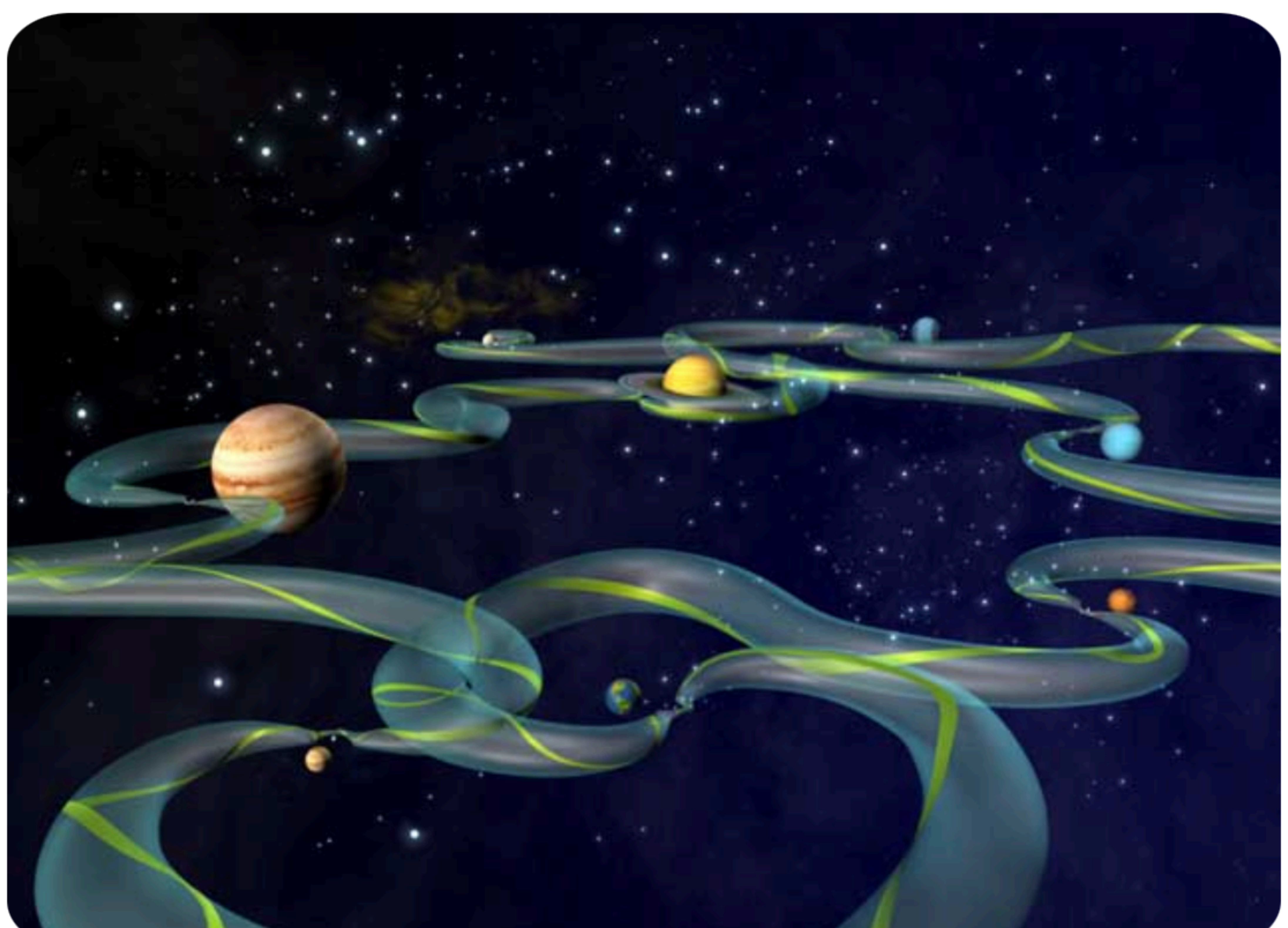
# Study Overview

## Mission Constraints

*Travel as cheap as possible.*

- Travel from Earth, to Jupiter
- Requires a periodic, or semi-periodic destination orbit
- Mission is robotic – duration has **no upper bound**
- We want to use as **little fuel as possible**

- **Transfers along invariant manifolds within CR3BP are great options for these constraints!**
- An artist's interpretation of invariant manifolds in space is featured on **NASA's website**, and is shown below



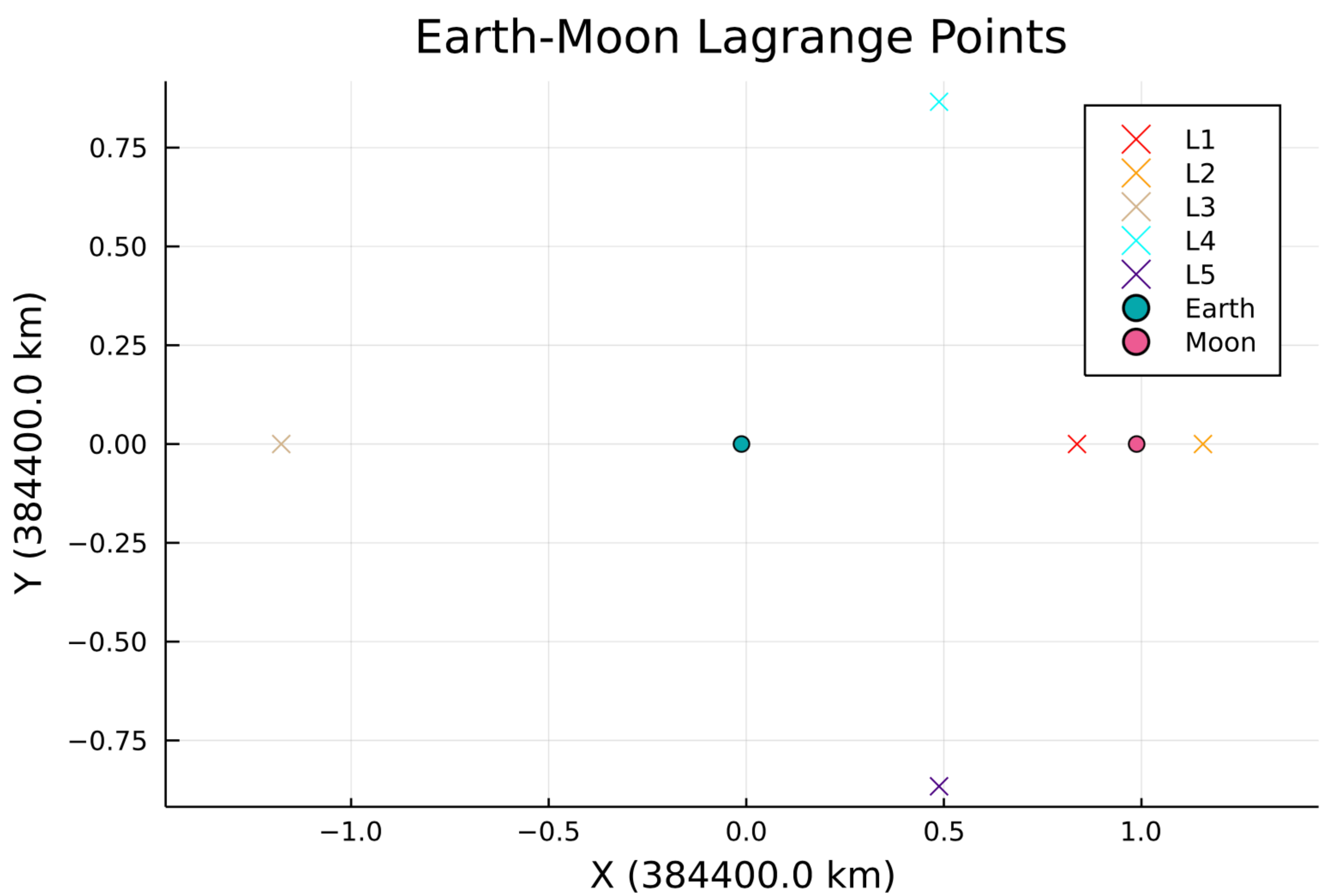
## 📚 Table of Contents

# Recall Lagrange Points

*The equilibrium points of CR3BP dynamics.*

- Like any equilibrium points, they can be **stable** or **unstable**
- We can verify the stability by looking at the **eigenvalues** of the linearized CR3BP dynamics **at** each Lagrange point

**System:** Earth-Moon



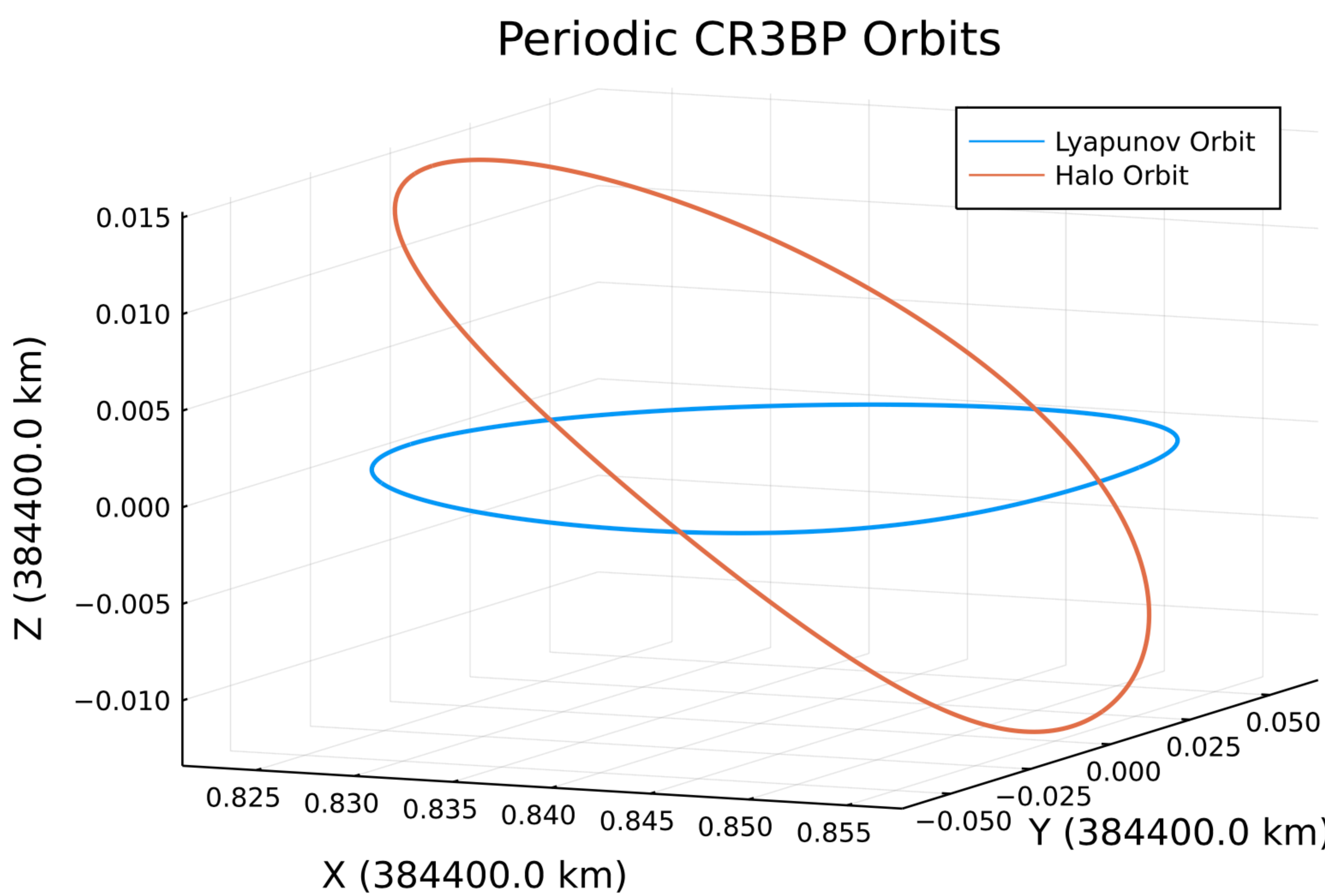

## 📚 Table of Contents

# Periodic Orbits within CR3BP

*Classifications don't really matter for this study!*

## Classifications

- **Lyapunov Orbits** are periodic with $z \equiv 0$
- **Halo Orbits** are periodic with $z \not\equiv 0$




## Table of Contents

# Analytical Halo Approximation

*As described in detail by Koon et al.*

Richardson approximated CR3BP dynamics with a **third-order expansion**:

$$\ddot{x} - 2\dot{y} - (1 + 2c_2)x = \frac{3}{2}c_3(2x^2 - y^2 - z^2) + 2c_4x(2x^2 - 3y^2 - 3z^2) + O(4)$$
$$\ddot{y} + 2\dot{x} + (c_2 - 1)y = -3c_3xy - \frac{3}{2}c_4y(4x^2 - y^2 - z^2) + O(4)$$
$$\ddot{z} + c_2z = -3c_3xz - \frac{3}{2}c_4z(4x^2 - y^2 - z^2) + O(4)$$

We can use the following facts to find a unique, periodic solution to this third-order expansion:

1. Nonlinearities affect the **frequency** of the linearization (Lindstedt-Poincaré Method)
2. The amplitudes of the linearization solution produce **equal** eigen-frequencies
3. The $X$, $Y$, and $Z$ amplitudes are **related** and **unique**

**The analytical algorithm can return the orbital period, positions, and velocities.**

```
julia> μ = 0.012150584269940356
julia> analyticalhalo(μ; Az = 0.03, L = 1, steps = 1)

r = [0.8231434287553542 0.0 0.03372857723603456]
v = [0.0 0.14183210515633737 0.0]
T = 2.9019861371476376
```

## 📚 Table of Contents

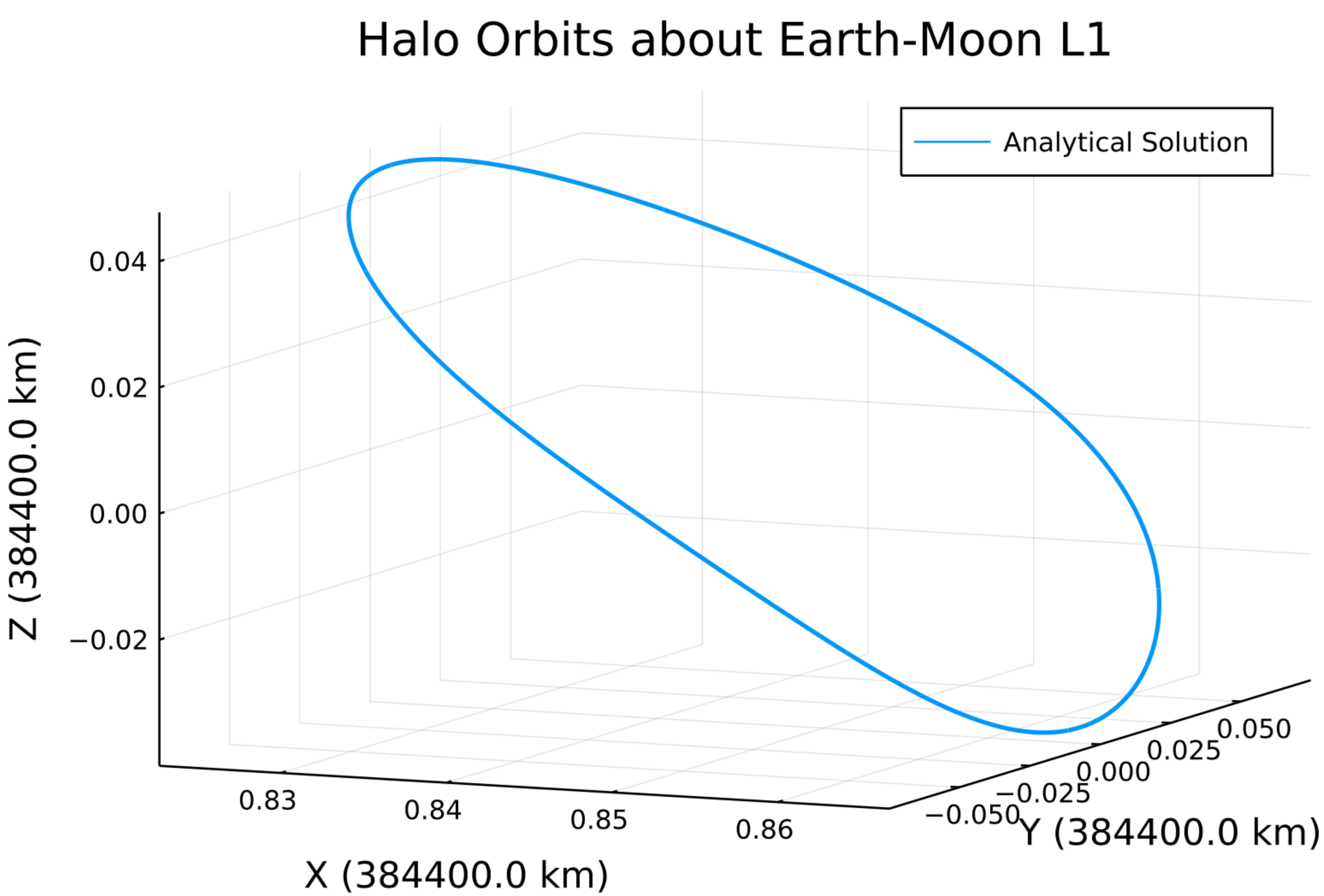
# Propagating Analytical Solutions

*It doesn't work!*

**Note**

Remember, we used a **third-order expansion** of the CR3BP dynamics. As a result, our analytical halo solutions are **approximations**!

**Include Propagation:** ☐




## Table of Contents

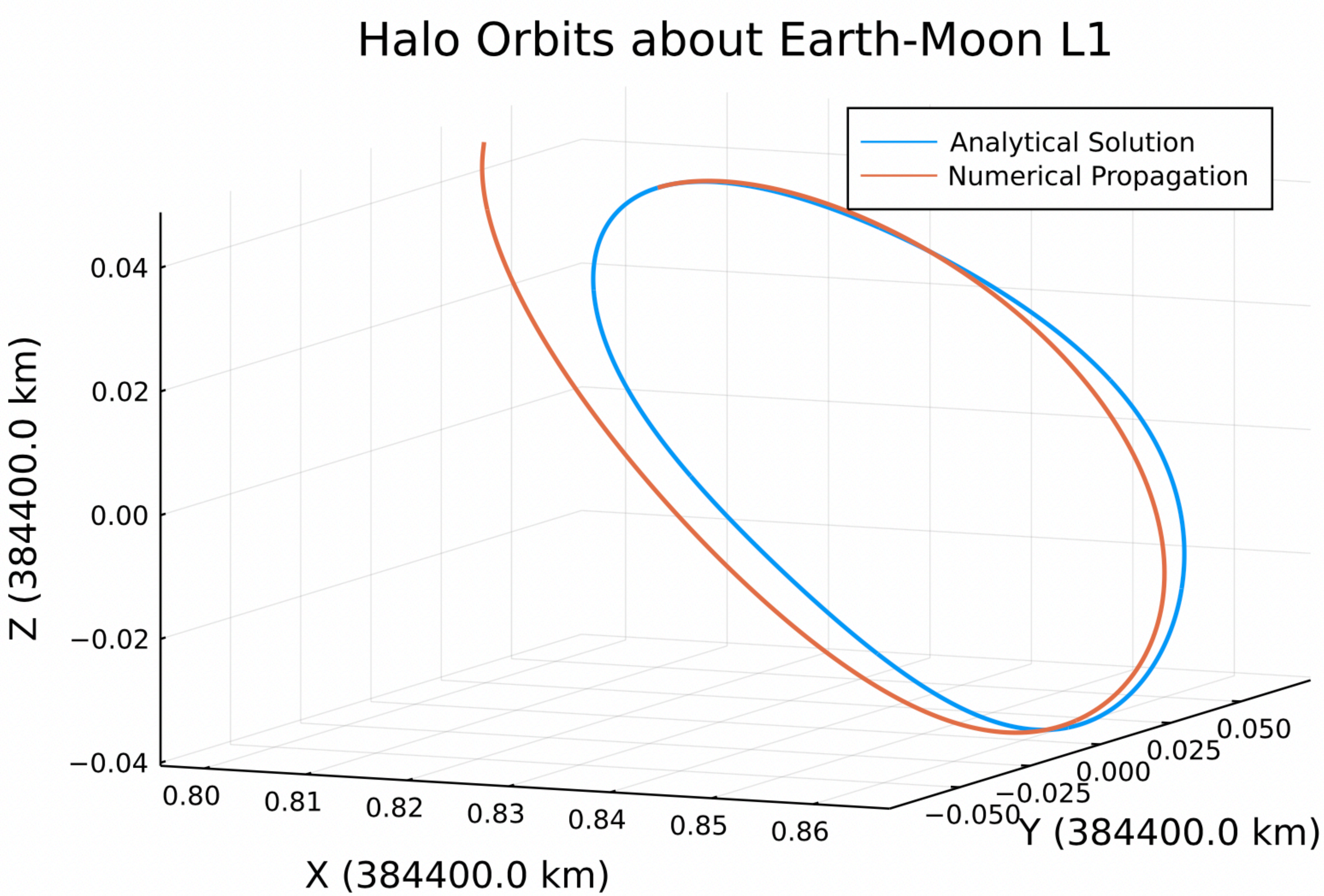
# Propagating Analytical Solutions

*It doesn't work!*

**Note**

Remember, we used a **third-order expansion** of the CR3BP dynamics. As a result, our analytical halo solutions are **approximations**!

**Include Propagation:** ☑




## Table of Contents

# Numerical Halo Solver

*AKA a differential corrector!*

Recall the **differential correction algorithm** introduced in ENAE601.

1. Start with a Halo guess of the form $r_0 = \begin{bmatrix} x_0 & 0 & z_0 \end{bmatrix},\ v_0 = \begin{bmatrix} 0 & \dot{y}_0 & 0 \end{bmatrix},\ T_0$
2. Propagate the solution for **half a period** (until $y$ crosses the $x - z$ plane again)
3. Use a **differential correction**$^\star$ to form a new Halo guess
4. Repeat until your half-period $\dot{x}_0$ and $\dot{z}_0$ values are within some acceptable tolerance of $0$

Literature shows **two**$^\star$ acceptable differential correction calculations, with $\Phi$ representing the Jacobian of the Cartesian state at the half-period.

$$\begin{bmatrix} z_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} \Longleftarrow \begin{bmatrix} x_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} - \operatorname{inv}\left( \begin{bmatrix} \Phi_{1,4} & \Phi_{5,4} & \ddot{x} \\ \Phi_{1,6} & \Phi_{5,6} & \ddot{z} \\ \Phi_{1,2} & \Phi_{5,2} & \dot{y} \end{bmatrix} \right) \begin{bmatrix} \dot{x} \\ \dot{y} \\ y \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} x_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} \Longleftarrow \begin{bmatrix} z_0 \\ \dot{y}_0 \\ \frac{T_0}{2} \end{bmatrix} - \operatorname{inv}\left( \begin{bmatrix} \Phi_{3,4} & \Phi_{5,4} & \ddot{x} \\ \Phi_{3,6} & \Phi_{5,6} & \ddot{z} \\ \Phi_{3,2} & \Phi_{5,2} & \dot{y} \end{bmatrix} \right) \begin{bmatrix} \dot{x} \\ \dot{y} \\ y \end{bmatrix} \tag{2}$$

**Tip**

You can't just *choose* whatever corrector you want! If you're solving for a Lyapunov orbit $(z \equiv 0)$ then the $3 \times 3$ matrix will be singular (second row) if you choose equation $(1)$. In practive, I've learned to use a switch statement – if your desired $z$-axis ampltitude is $0$, then use $(2)$, and otherwise use $(1)$.

## 📚 Table of Contents

# Propagating Numerical Solutions

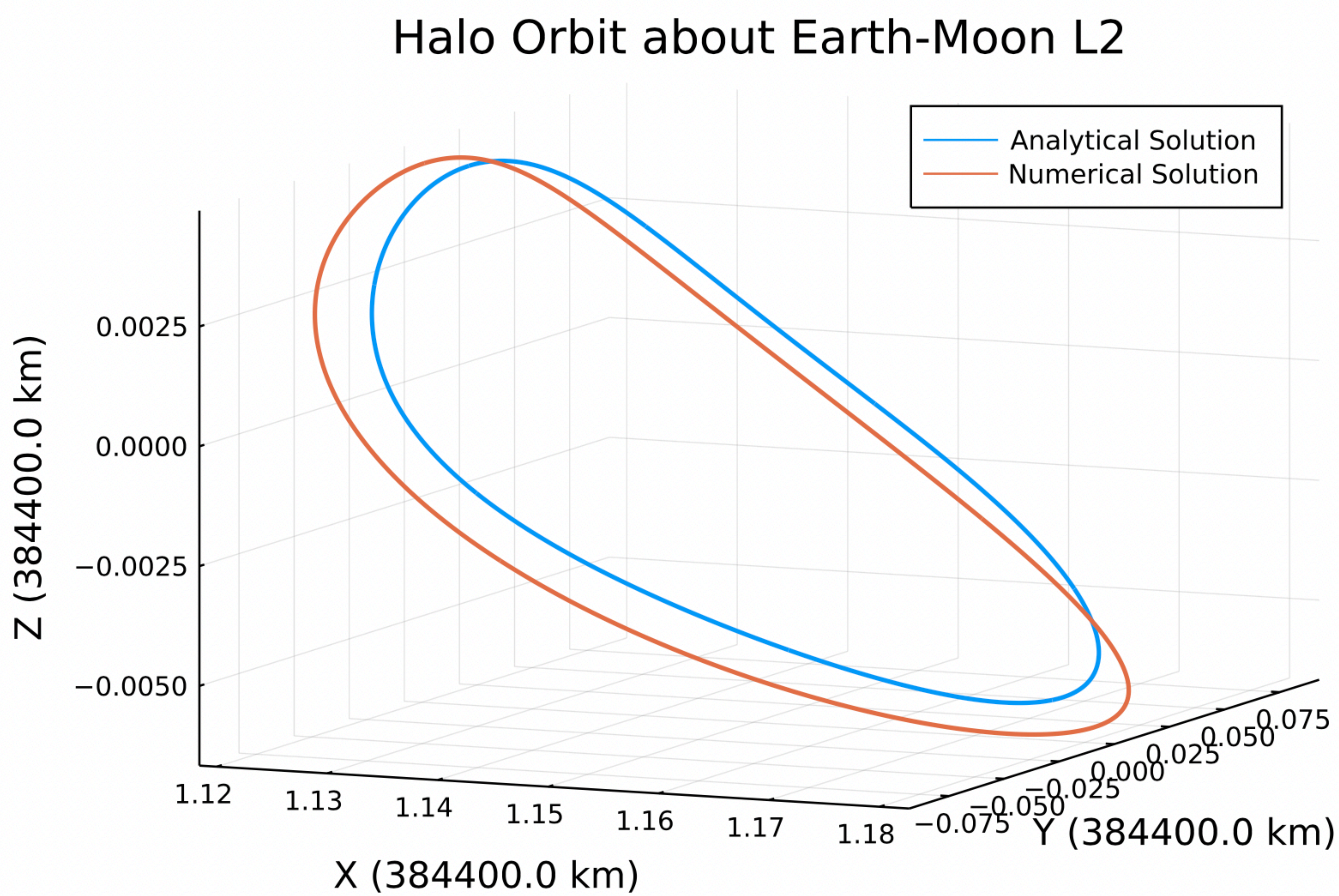
*This ~does~ work!*

We can **combine** the working analytical and numerical solvers to form one ergonomic halo solver!

$$\text{Desired Attributes} \rightarrow \boxed{\text{Analytical}} \rightarrow \boxed{\text{Numerical}} \rightarrow \text{Halo Orbit}$$

## 🥁 Halo Orbit Attributes

**System:** Earth-Moon **Lagrange Point:** 2 **Z-axis Amplitude:** 0.005




## 📚 Table of Contents

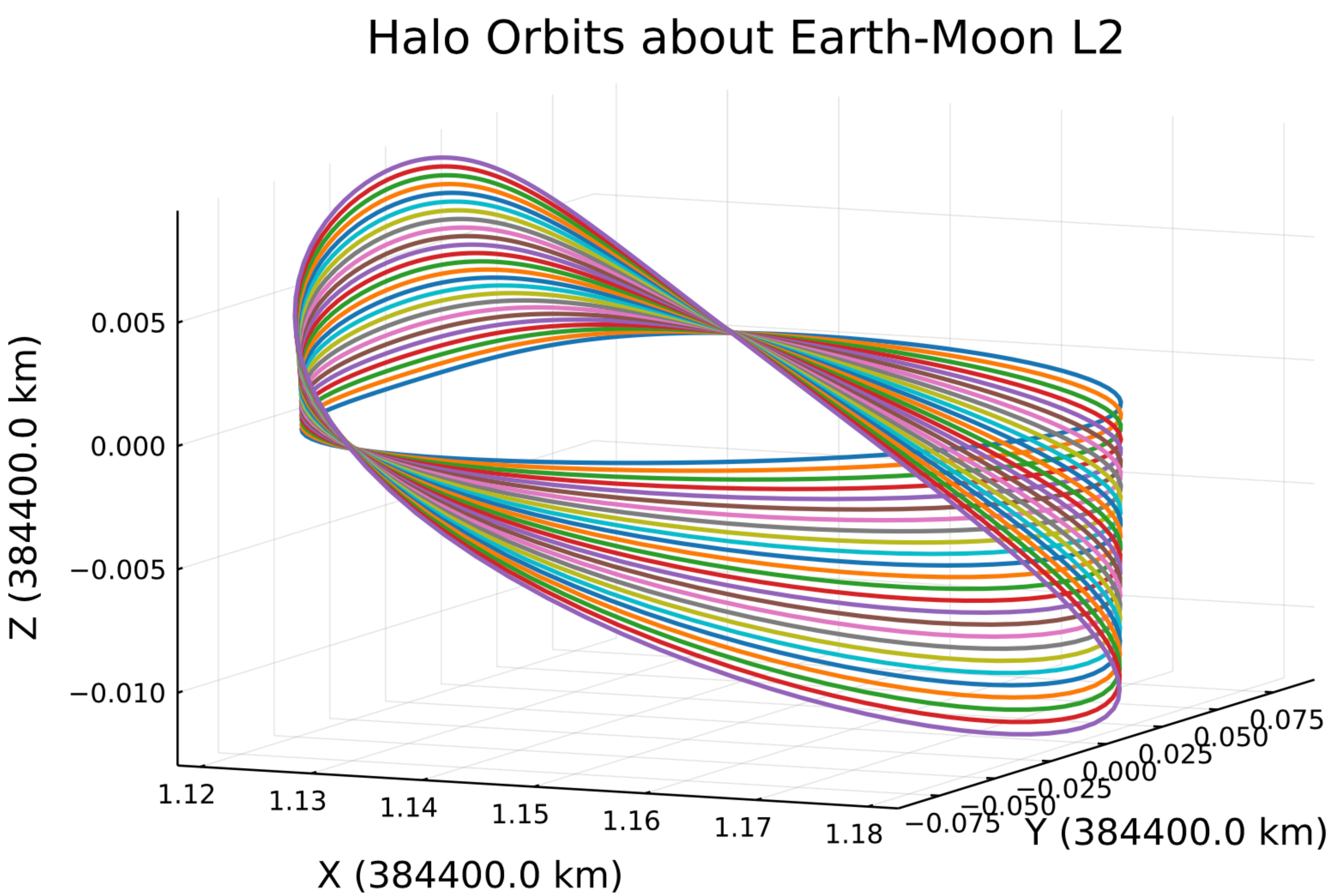
# Halo Orbit Families

*Variations in Z-axis amplitude.*

- A Halo orbit *family* is a collection of Halos with varying Z-axis amplitudes
- A collection of over $130,000$ Halo orbits in our solar system is available on GitHub **here**!

## Halo Family Attributes

**System:** Earth-Moon **Lagrange Point:** 2

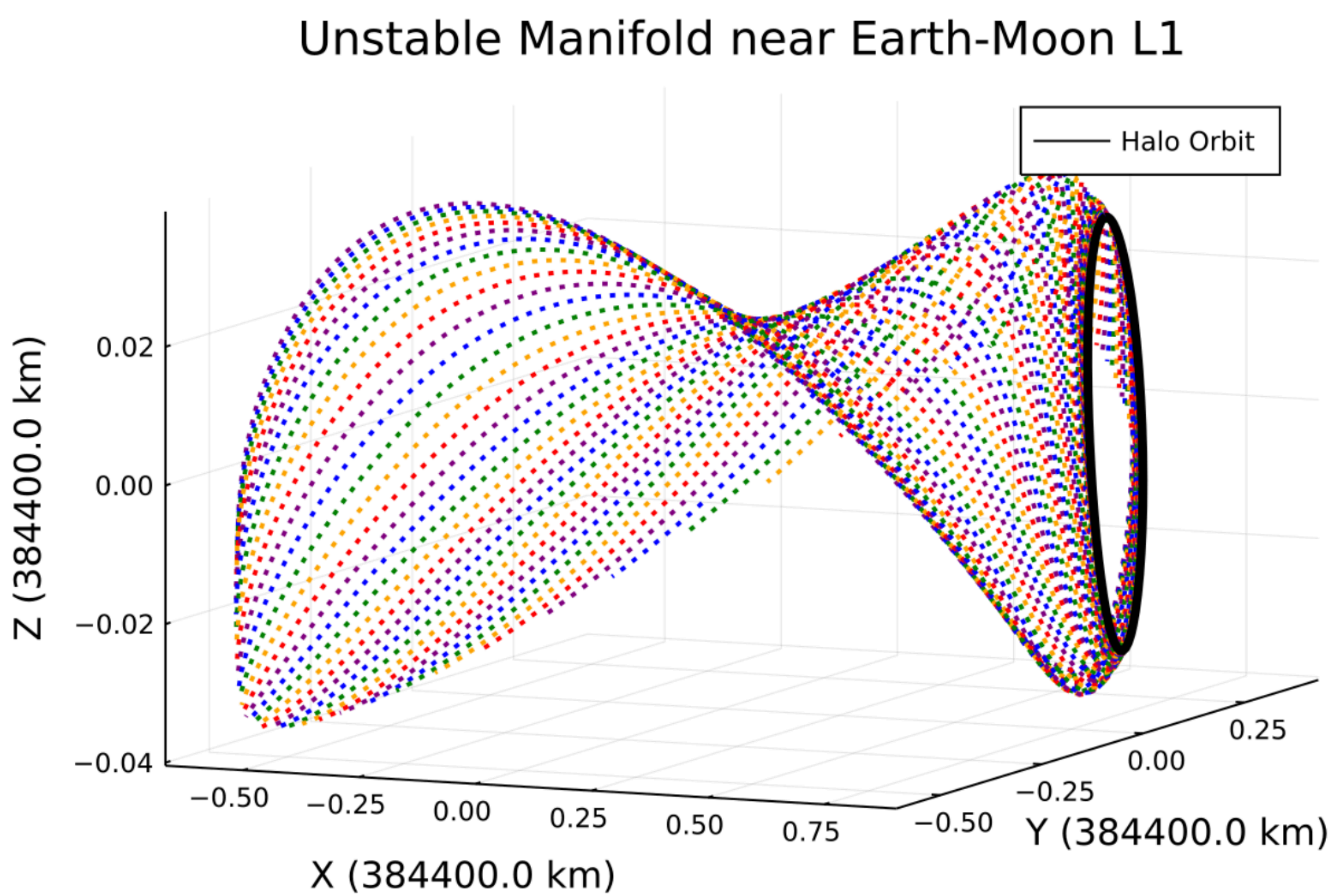



### 📚 Table of Contents

# Manifolds Overview

*Superhighways in ~space~.*

- Manifolds are **collections of trajectories** that **converge to** or **diverge from** a periodic orbit or Lagrange point
- They are visualized by plotting trajectories *near* a Lagrange point, or a periodic CR3BP orbit



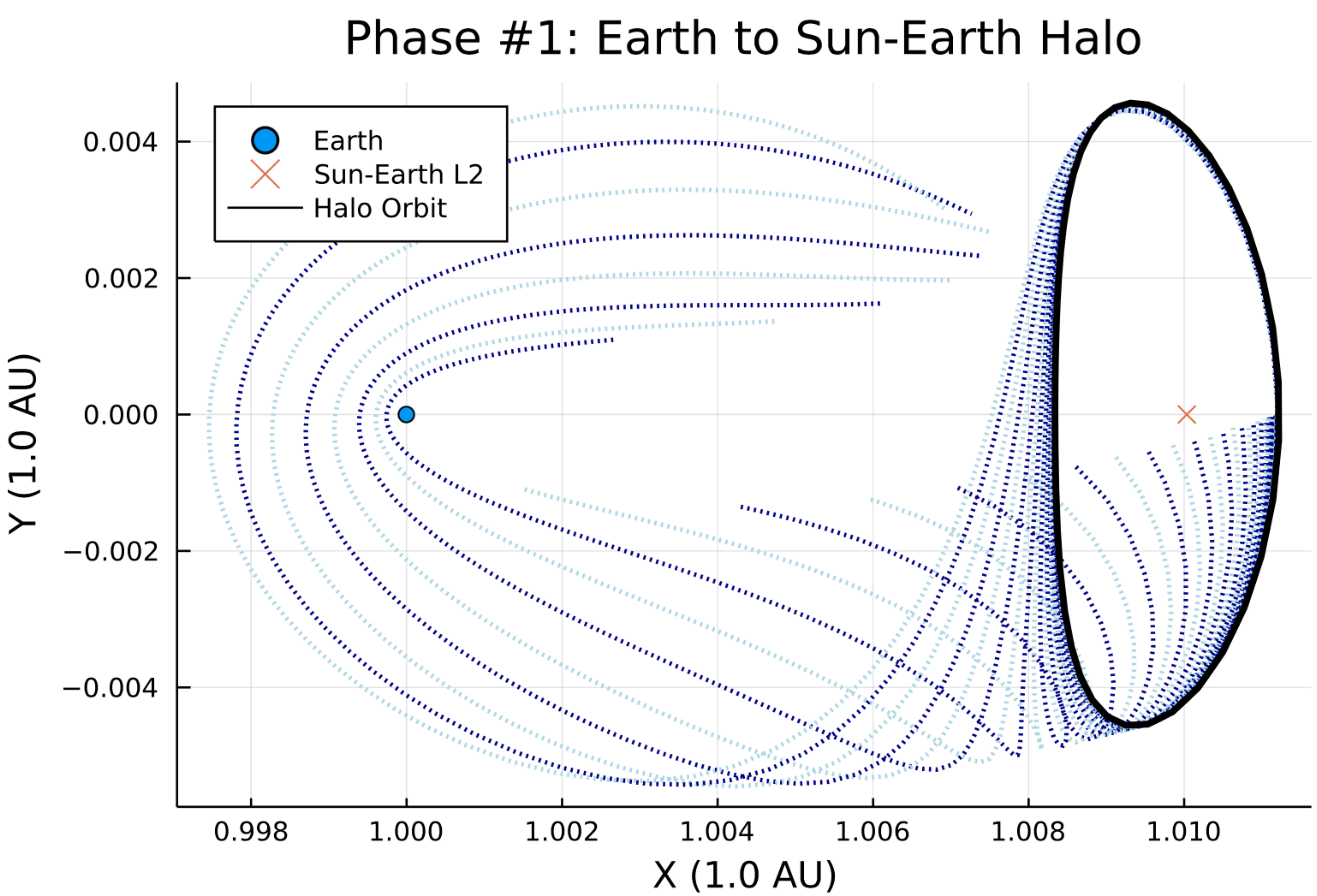

📚 Table of Contents

# Mission Phases

*Manifold-based interplanetary missions are a 3 step process.*

1. Launch from Earth into a **stable manifold** within the Sun-Earth system
2. Perturb from the Sun-Earth Halo onto an **unstable manifold**
3. Transfer onto a **stable manifold** of the desired destination Halo orbit

**Mission Phase:** Phase One



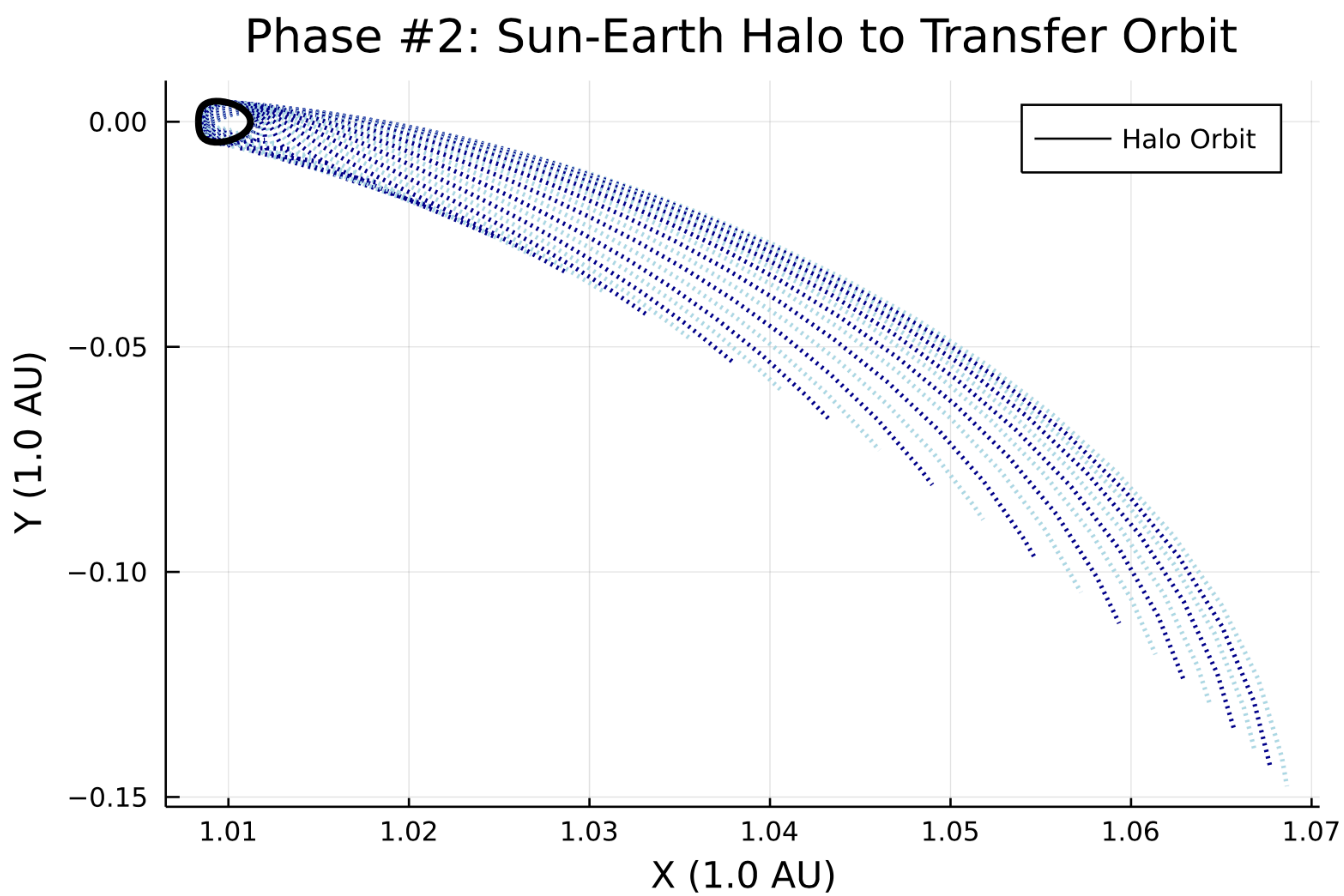

## 📚 Table of Contents

# Mission Phases

*Manifold-based interplanetary missions are a 3 step process.*

1. Launch from Earth into a **stable manifold** within the Sun-Earth system
2. Perturb from the Sun-Earth Halo onto an **unstable manifold**
3. Transfer onto a **stable manifold** of the desired destination Halo orbit

**Mission Phase:** Phase Two



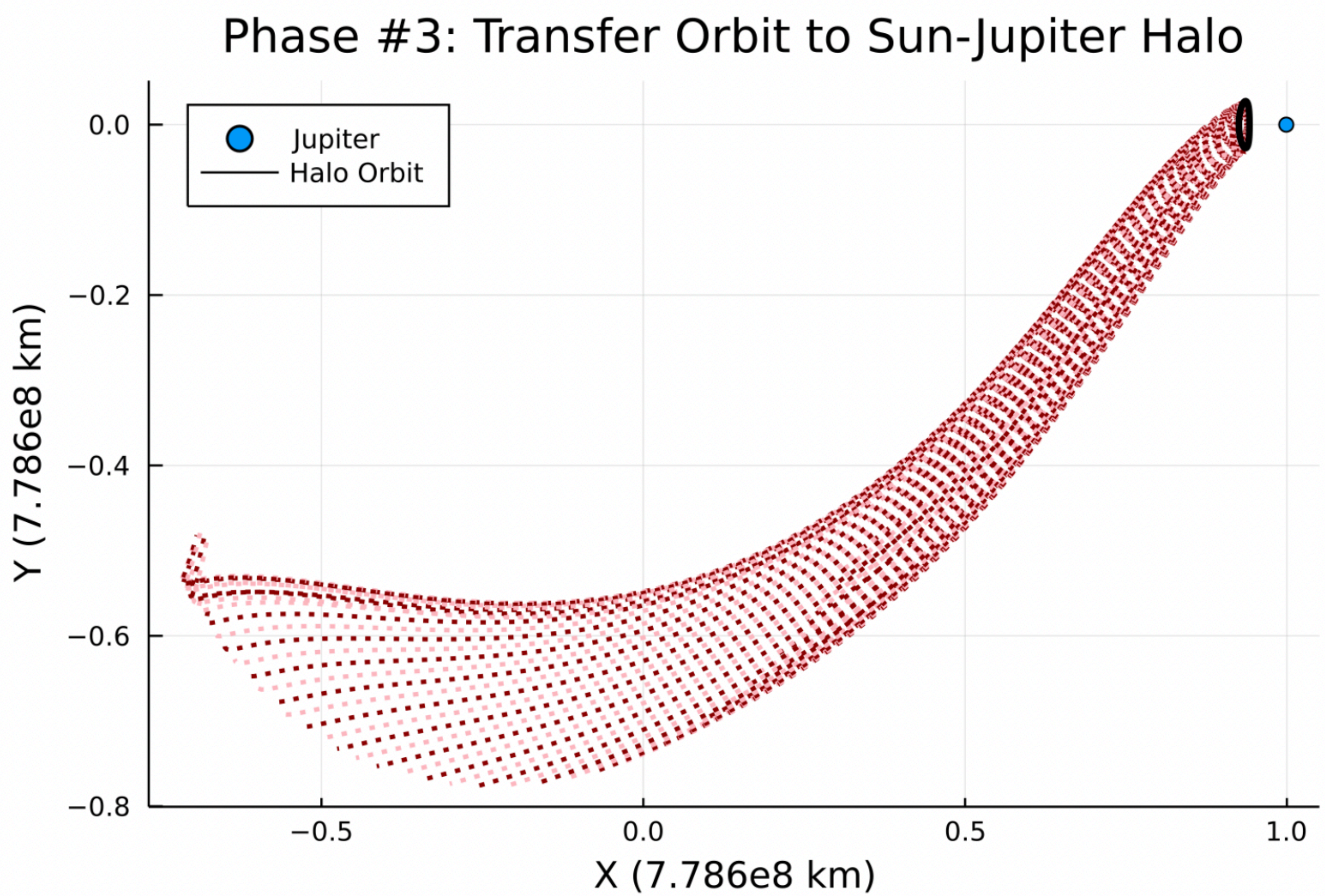

📚 Table of Contents

# Mission Phases

*Manifold-based interplanetary missions are a 3 step process.*

1. Launch from Earth into a **stable manifold** within the Sun-Earth system
2. Perturb from the Sun-Earth Halo onto an **unstable manifold**
3. Transfer onto a **stable manifold** of the desired destination Halo orbit

**Mission Phase:** Phase Three




## Table of Contents

# Phase #1 Design

*Where should we point our launch vehicle?*

- Let's assume a Halo orbit with a $200,000$ km Z-axis amplitude about Sun-Earth $L2$
- We need to find a state within the Halo's **stable manifold** that is **easy** to get to from Earth
- The cost function used to evaluate each orbital state, and the resulting **optimal** Earth-departure orbit are shown below
- **Let's only include orbital states within $1\%$ of Earth's SOI in our search!**

$$\text{COST}({}^{\text{ECI}}r_{\text{sc}}, {}^{\text{ECI}}v_{\text{sc}}, t_{(\text{to halo})}) = {}^{\text{ECI}}r_{\text{sc}}^2 + {}^{\text{ECI}}v_{\text{sc}}^2 + 10t_{(\text{to halo})}^2$$

### Optimal Earth-Departure Orbit

Time to Halo: $3.7992$ years

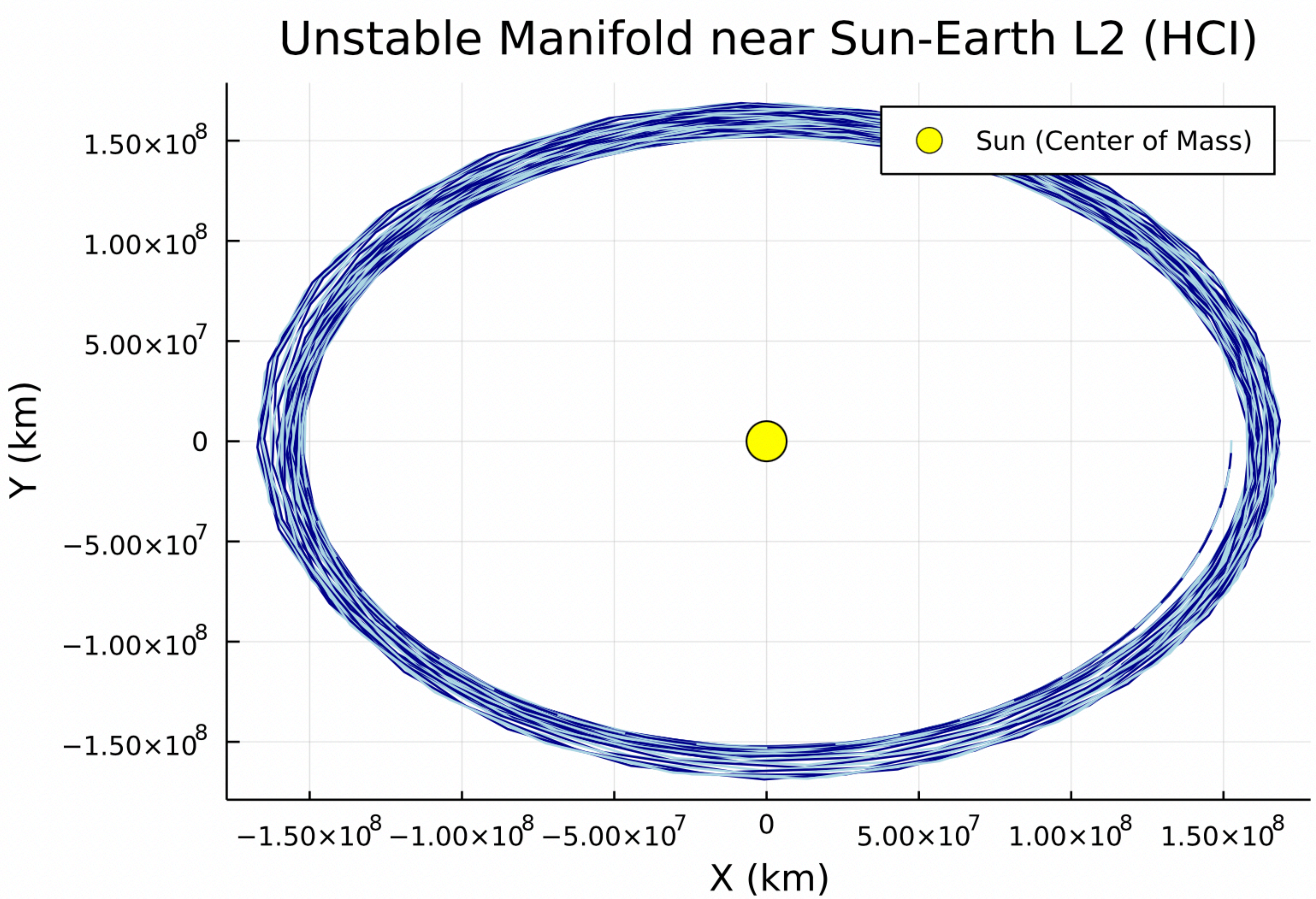
```
Elliptical Restricted Two-body Orbit (Float64):

  ECI Cartesian State:

    t =  -1.19892297638199e8 s
    r = [-827383.871059 394727.776642 -99291.628123] km
    v = [-0.056047 -0.027744 0.012576] km s⁻¹

  Restricted Two-body System (Earth):

    μ = 398600.4354360959 km³ s⁻²

  Orbit Properties:

    e =  0.9937738640475346
   C3 = -0.8604980835118132 km² s⁻²
```

## 📚 Table of Contents

# Phase #2 Design

*Where can the Sun-Earth manifold take us?*

- Compute the **unstable manifold** near your Sun-Earth Halo orbit
- The **unstable manifold** shows trajectories up to $4529.2794$ days, with a $94.8078$ m/s perturbation cost

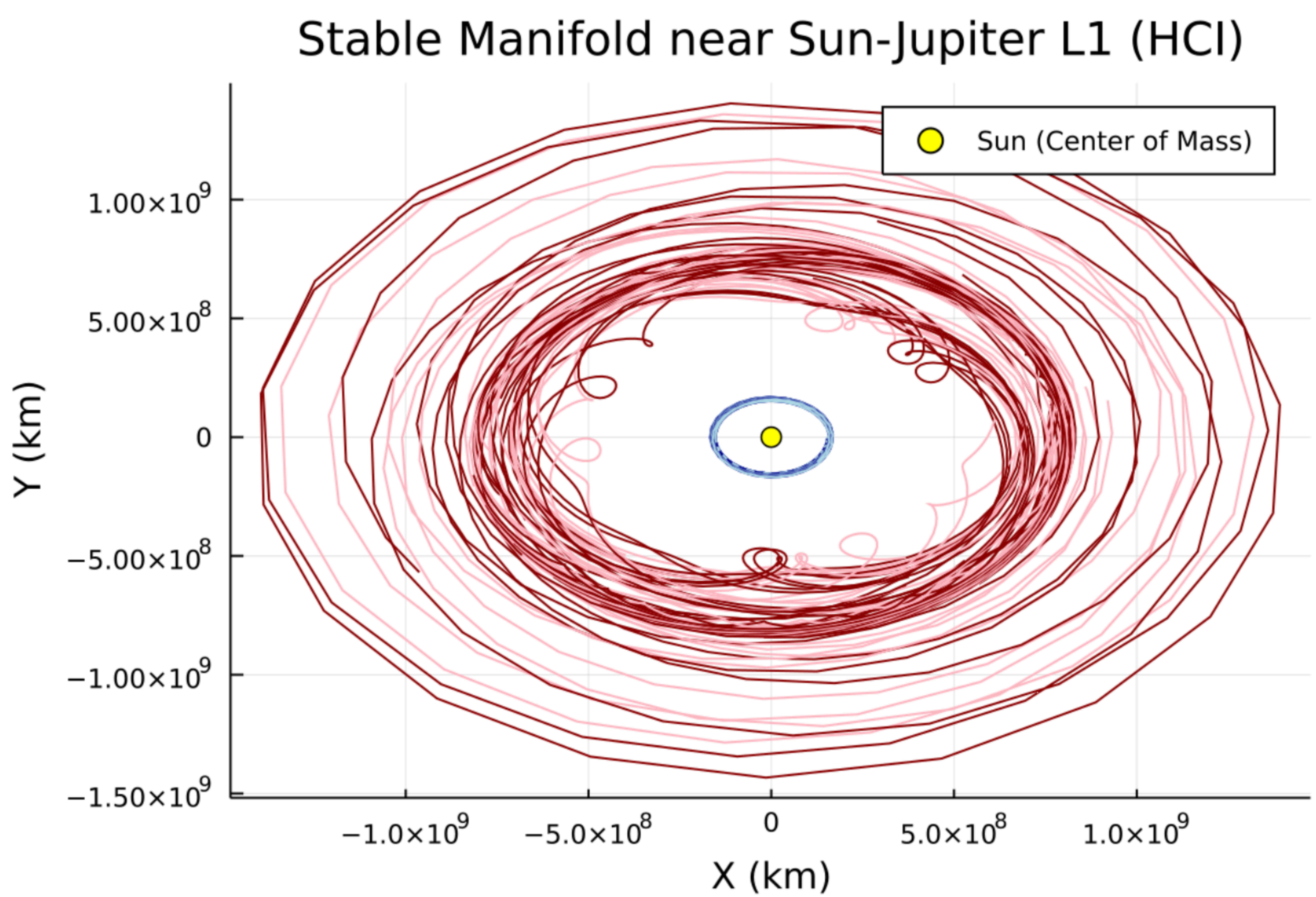



## 📚 Table of Contents

# Phase #3 Design

*How can we get to the Sun-Jupiter manifold?*

- Compute the **stable manifold** near your Sun-Earth Halo orbit
- The **stable manifold** shows trajectories up to $50899.4193$ days, with a $415.7727$ m/s perturbation cost
- **Lambert-scan result:** $6.0$ years, $37.5473$ km/s $\Delta v$




## 📚 Table of Contents

# Lessons Learned

## Mission Results

- Total Duration: $105.6238$ years
- Total $\Delta v$: $37.2263 \ \frac{\text{km}}{\text{s}}$

## Issues

- These results are likely incorrect due to book-keeping, including...
  - CR3BP to HCI
  - Lambert scanner issues

## Forward Work

- Fix Lambert issues, confirm CR3BP to HCI calculations
- Compare correct manifold-transfer mission results to Hohmann, Lambert (without manifold)
- Use nonlinear optimizer to find optimal departure and arrival Halo orbit amplitudes
- Use ephemeris data for realistic mission planning

## Package Announcement

- New Julia package: `GeneralAstrodynamics`
- Features include:
  - R2BP, CR3BP, and NBP definitions
  - Position, velocity, zero-velocity-curve, Lagrange point plotting
  - Ephemeris interpolation
  - Orbit propagation
  - Kepler, Lambert, and Halo solvers
  - More to come!

📚 Table of Contents

# Project References


- NASA,NASAsLunarExplorationProgramOverview,2020.
- Rund, M. S., “Interplanetary Transfer Trajectories Using the Invariant Manifolds of Halo Orbits,” , 2018.
- Richardson, D., “Analytical construction of periodic orbits about the collinear points of the Sun-Earth system.” asdy, 1980, p. 127.
- Koon,W.S.,Lo,M.W.,Marsden,J.E.,andRoss,S.D.,“Dy- namical systems, the three-body problem and space mission design,” Free online Copy: Marsden Books, 2008.
- Howell, K. C., “Three-dimensional, periodic,haloorbits,” Ce- lestial mechanics, Vol. 32, No. 1, 1984, pp. 53–71.
- Carpinelli, J., “Exploring Invariant Manifolds and Halo Orbits,” https://github.com/cadojo/Halo-Orbit-Solvers, 2020- 2021.
- Carpinelli, J., “GeneralAstrodynamics.jl,” https://github.com/cadojo/GeneralAstrodynamics.jl, 2021.
- 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.
- van der Plas, F., “Pluto.jl,” https://github.com/fonsp/Pluto.jl, 2020.
- Williams, J., Lee, D. E., Whitley, R. J., Bokelmann, K. A., Davis, D. C., and Berry, C. F., “Targeting cislunar near recti- linear halo orbits for human space exploration,” 2017.
- Vallado, D. A., Fundamentals of astrodynamics and applica- tions, Vol. 12, Springer Science & Business Media, 2001.
- Lara,M.,Russell,R.,andVillac,B.,“Classificationofthedis- tant stability regions at Europa,” Journal of Guidance, Control, and Dynamics, Vol. 30, No. 2, 2007, pp. 409–418.
- Zimovan-Spreen, E. M., Howell, K. C., and Davis, D. C., “Near rectilinear halo orbits and nearby higher-period dynam- ical structures: orbital stability and resonance properties,” Ce- lestial Mechanics and Dynamical Astronomy, Vol. 132, No. 5, 2020, pp. 1–25.


📚 Table of Contents

# Extras

📚 Table of Contents

# Dependencies

```julia
begin

    using DrWatson

    using CSV
    using Plots
    using Revise
    using PlutoUI
    using Crayons
    using Latexify
    using DataFrames
    using LaTeXStrings
    using LinearAlgebra
    using ConcreteStructs
    using GeneralAstrodynamics
    using Unitful, UnitfulAngles, UnitfulAstro

    macro terminal(expr)
        quote
            with_terminal() do
                $(esc(expr))
            end
        end
    end

end;
```

📚 Table of Contents

# Support Types

## Halos

```julia
begin
    struct Halo{O<:CR3BPOrbit, T<:Real}
        orbit::O
        period::T
    end

    function Halo(param::P; kwargs...) where P<:Union{Real, CR3BPSystem}
        orbit, T = halo(param; kwargs...)
        return Halo(orbit, T)
    end
end;
```

## Transfers

```julia
@concrete terse mutable struct Transfer
    departure
    arrival
    Δv₁
    Δv₂
    Δv
    TOF
    m
end;
```

## Ephemeris

```julia
earth, moon, jupiter = let
    path  = joinpath(datadir(), "exp_pro", "ephemeris", "wrt_sun")

    earth   = interpolator(joinpath(path, "Earth.txt"))
    moon    = interpolator(joinpath(path, "Moon.txt"))
    jupiter = interpolator(joinpath(path, "Jupiter.txt"))

    earth, moon, jupiter
end;
```

📚 Table of Contents

# Support Functions

## CR3BP to R2BP

```julia
function R2BP(orbit::CR3BPOrbit, body_index::Int, body::R2BPSystem;
              frame = Inertial)

    # Position of celestial body
    rᵦ, vᵦ = let

        normalized_orbit = normalize(orbit)

        if body_index == 1
            synodic_body_position = primary_synodic_position(normalized_orbit)
        elseif body_index == 2
            synodic_body_position = secondary_synodic_position(normalized_orbit)
        else
            throw(ArgumentError("Second argument `body_index` must be 1 or 2."))
        end

        inertial_body_position = inertial(synodic_body_position,
                                          epoch(normalized_orbit))
        inertial_body_velocity = (
            (normalize(inertial_body_position) × [0, 0, -1]) .*
                scalar_velocity(normalized_length_unit(orbit.system) *
                    normalized_mass_parameter(orbit.system),
                    normalized_length_unit(orbit.system) *
                    normalized_mass_parameter(orbit.system),
                    primary_mass_parameter(orbit.system)))

        redimensionalize_length.(inertial_body_position,
                                 normalized_length_unit(orbit.system)),
        inertial_body_velocity
    end


    pos = (position_vector ∘ redimensionalize ∘ inertial)(orbit)
    vel = (velocity_vector ∘ redimensionalize ∘ inertial)(orbit)
    tp  = epoch(redimensionalize(orbit))

    pos = uconvert.(u"km", pos .- rᵦ)
    vel = uconvert.(u"km/s", vel)
    tp  = uconvert(u"s", tp)

    state = CartesianState(pos, vel, tp, frame)
```

# 📚 Table of Contents

```
    pos = (position_vector ∘ redimensionalize ∘ inertial)(orbit)
    vel = (velocity_vector ∘ redimensionalize ∘ inertial)(orbit)
    tp  = epoch(redimensionalize(orbit))

    pos = uconvert.(u"km", pos .- rᵦ)
    vel = uconvert.(u"km/s", vel)
    tp  = uconvert(u"s", tp)

    state = CartesianState(pos, vel, tp, frame)

    return R2BPOrbit(state, body)

end;
```

## Cost Function

```
function cost(orbit; P = 1, D = 1, T = 10)
    r = position_vector(orbit) - secondary_synodic_position(orbit)
    r = redimensionalize_length.(r, normalized_length_unit(orbit.system))
    r = r |> norm |> upreferred |> ustrip

    v = velocity_vector(orbit) - secondary_synodic_position(orbit)
    v = v |> norm |> upreferred |> ustrip

    t = (ustrip ∘ upreferred ∘ epoch ∘ redimensionalize)(orbit)

    return (P * r^2) + (D * v^2) + (T * t^2)
end;
```

📚 Table of Contents

# Julia vs. MATLAB

*What's the difference?*

- MATLAB is commercial software, while Julia is free and open-source
- Julia has a robust package manager, and can run faster than MATLAB **during execution**
- Julia is *just-ahead-of-time* compiled, so you pay by **waiting for your code to compile** the first time you execute it (microseconds to seconds)

| Task | MATLAB | Julia |
|---:|---:|---:|
| `Print to the console` | `disp('Hello, world!')` | `print("Hello, world!")` |
| `Make a random 3×3 matrix` | `M = rand(3,3)` | `M = randn(3,3)` |
| `Take the determinant` | `det(M)` | `det(M)` |
| `Matrix multiplication` | `M * M` | `M * M` |
| `Element-wise multiplication` | `M .* M` | `M .* M` |
| `Two-norm of a vector` | `norm(vec)` | `norm(vec)` |
| `Vectorize function (1/2)` | `for i = vec, f(i), end` | `for i = vec; f(i); end` |
| `Vectorize function (2/2)` | `n/a` | `f.(vec)` *or* `map(f, vec)` |
| `Numerical integration` | `ode45(f, x0, p t)` | `solve(ODEProblem(f, x₀, p, t))` |
| `Symbolic math` | `syms x y z` | `@variables x y z` |
| `Manual` | `help f` | `@doc f` |

## 📚 Table of Contents