Halo Orbit Simulation of JWST at L_2

CSE 6730 Milestone 3: Project checkpoint 2

Group 15

Tianyang Hu, Haoran Yan, Shiqi Fan, Mark Zhang

GitHub Repository: https://github.com/TimHu-0728/CSE-6730-Final-Project

Current Implementation Progress

1. Numerical Implementation and Validation

Following the Circular Restricted Three-Body Problem (CR3BP) formulation of Weber [1], we have implemented the JWST dynamics in the rotating frame using Python's `control` library [2]. The non-dimensional motion is governed by

$$\begin{aligned}
\ddot{x} - 2\dot{y} &= x - \frac{(1-\pi_2)(x+\pi_2)}{\sigma^3} - \frac{\pi_2(x-1+\pi_2)}{\psi^3} + u_x, \\
\ddot{y} + 2\dot{x} &= y - \frac{(1-\pi_2)y}{\sigma^3} - \frac{\pi_2 y}{\psi^3} + u_y, \\
\ddot{z} &= -\frac{(1-\pi_2)z}{\sigma^3} - \frac{\pi_2 z}{\psi^3} + u_z,
\end{aligned}$$

where σ and ψ are the normalized distances from JWST to the Sun and Earth, respectively:

$$\sigma = \sqrt{(x+\pi_2)^2 + y^2 + z^2}, \qquad \psi = \sqrt{(x-1+\pi_2)^2 + y^2 + z^2}.$$

By defining the state vector $X = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$, we get the non-dimensional CR3BP dynamic equations in the state-space form

$$\begin{aligned}
\dot{X}_1 &= X_4 \\
\dot{X}_2 &= X_5 \\
\dot{X}_3 &= X_6 \\
\dot{X}_4 &= -\frac{1-\pi_2}{\sigma^3}(X_1+\pi_2) - \frac{\pi_2}{\psi^3}(X_1-\pi_1) + 2X_5 + X_1 + u_x \\
\dot{X}_5 &= -\frac{1-\pi_2}{\sigma^3}X_2 - \frac{\pi_2}{\psi^3}X_2 - 2X_4 + X_2 + u_y \\
\dot{X}_6 &= -\frac{1-\pi_2}{\sigma^3}X_3 - \frac{\pi_2}{\psi^3}X_3 + u_z
\end{aligned} \tag{1}$$

which in general vector form is

$$\dot{X} = f(X) + g(u) \tag{2}$$

With zero control input $u = 0$, the Orbit of the JWST is strongly dependent on its initial conditions. We have simulated an unstable Halo Orbit around Earth-Sun L2 point using the the initial condition

$X_{Halo}(0)$ obtained from the JPL Horizons system (non-dimensionalized) [3]:

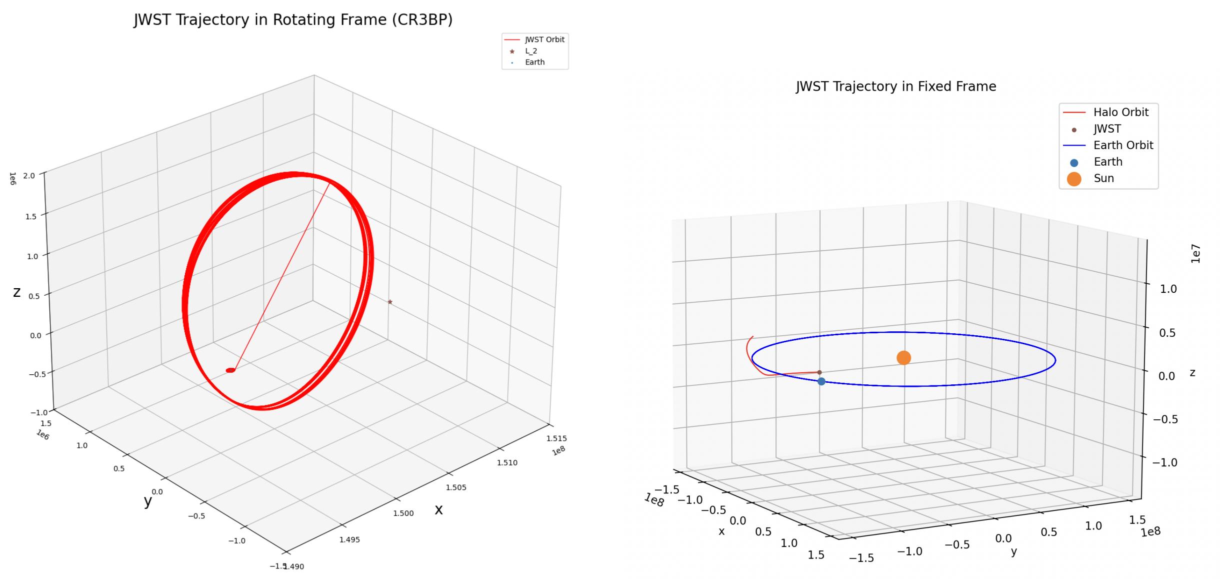
$$X_{Halo}(0) = \begin{bmatrix} 1.006\,201\,041\,659\,247\,6 \\ -2.152\,307\,851\,825\,963\,0 \times 10^{-23} \\ 1.238\,031\,120\,134\,930\,3 \times 10^{-2} \\ -9.807\,436\,982\,071\,242\,3 \times 10^{-16} \\ -1.325\,347\,792\,466\,051\,1 \times 10^{-2} \\ -1.095\,660\,345\,906\,113\,3 \times 10^{-14} \end{bmatrix}$$

In addition, we have also simulated a lower Earth orbit (LEO) of the JWST using the initial condition

$$X_{LEO}(0) = \begin{bmatrix} 1.000\,205\,272\,126\,932\,3 \\ -3.677\,881\,684\,285\,494\,6 \times 10^{-22} \\ -5.960\,562\,510\,814\,000\,6 \times 10^{-2} \\ -2.198\,524\,397\,434\,676\,5 \times 10^{-11} \\ -1.212\,898\,467\,950\,106\,1 \times 10^{-1} \\ 4.403\,023\,282\,003\,022\,5 \times 10^{-23} \end{bmatrix}$$

The two orbits are plotted in Fig.1(a). At the stage, we only concatenate the two orbits, there is a single line connecting the end position of LEO and starting position of Halo Orbit. We will solve an optimal control problem to find a transfer orbit that bridges the two points later.

By transferring the Halo orbit data to fixed frame, we can visualize the relative motion of the three bodies as shown in Fig. 1(b).



(a) LEO concatenated with Halo Orbit of JWST

(b) Halo Orbit in fixed frame

Figure 1: Comparison of JWST halo-orbit visualizations in different coordinate frames.

2. Visualization System

We developed 3D visualizations of the Sun–Earth–JWST system using the PyVista library [4]. The animations depict the Sun and Earth as spheres, JWST as a compact model, and the halo orbit traced in 3D with dynamic lighting and smooth camera motion. For this checkpoint, we have implemented the fixed lab-frame view; a rotating-frame view (with Earth held fixed and JWST orbiting near L_2) will be added later. The main 3D video can be accessed at JWST_3D_Orbit.mp4. All .mp4 files are stored under the results/ folder in our private GitHub repository.

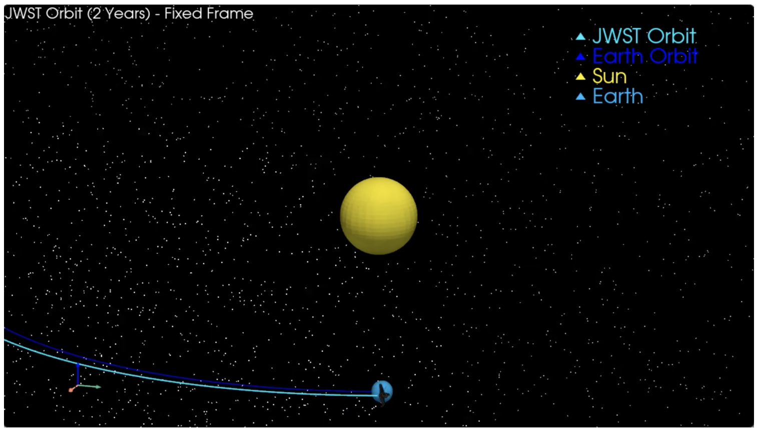
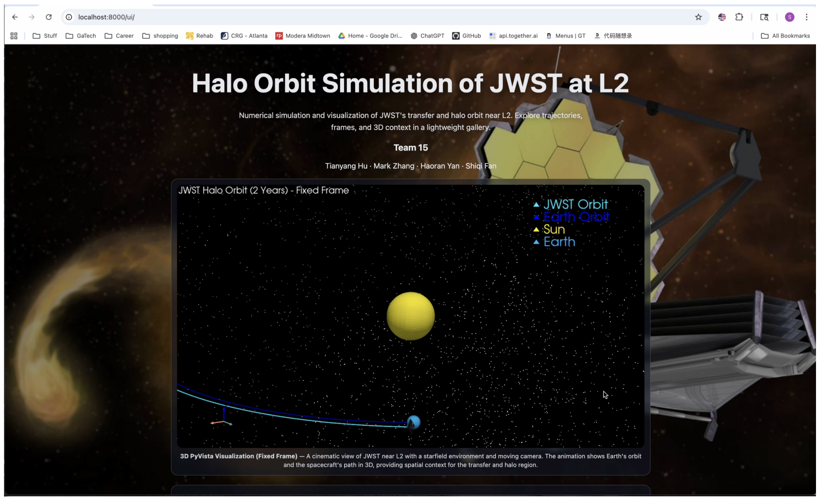


Figure 2: Click image to open the JWST_3D_Orbit.mp4.

To make the results easy to view, we implemented a lightweight web-based UI that acts as a video gallery for our simulation outputs. Each scenario (e.g., halo orbit view, fixed-frame vs. rotating-frame perspectives) is presented as a card that combines the animation with a short explanatory caption, so users can simply scroll through the page and understand each visualization. A short demo of the UI is available at JWST_VideoGallery_Demo.mp4. All videos used in the UI are generated by our Python/PyVista pipeline and loaded from the results/ directory.



Halo Orbit Simulation of JWST at L2

Numerical simulation and visualization of JWST's transfer and halo orbit near L2. Explore trajectories, frames, and 3D context in a lightweight gallery.

Team 15

Tianyang Hu · Mark Zhang · Haoran Yan · Shiqi Fan

3D PyVista Visualization (Fixed Frame) — A cinematic view of JWST near L2 with a starfield environment and moving camera. The animation shows Earth's orbit and the spacecraft's path in 3D, providing spatial context for the transfer and halo region.

Figure 3: Click the image to open the JWST_VideoGallery_Demo.mp4.

Remaining Work & Next Steps

To find a reasonable trajectory $X(t)$ and control $u(t)$ that bridges the end states of LEO, and the starting states of Halo Orbit. We can solve a Optimal Control problem, which minimizes a cost function in a feasible input space $\mathcal{U}$, subject to dynamics (1). We choose the integral and terminal cost functions to be both quadratic in states and inputs, then the problem is

$$\begin{aligned} \min_{u \in \mathcal{U}} \int_0^T (X(\tau) - X_0)^\top Q (X(\tau) - X_0) + (u(\tau) - u_0)^\top R (u(\tau) - u_0) d\tau \ + \ (X(T) - X_0)^\top P (X(T) - X_0) \\ s.t. \qquad \dot{X} = f(X) + g(u) \end{aligned} \tag{3}$$

where the weighting matrices, $Q \succeq 0$, $P \succeq 0$, and $R \succ 0$, are assumed to be constant.

Instead of solving using classical optimal control methods, we used the function control.optimal.solve_optimal_trajectory in [2] to generate the optimal trajectory $X(t)$ and $u(t)$. This function cast the original problem to a nonlinear optimization problem by discretizing the input vector u at each time points and treat them as the coefficeints (inputs) of the optimization problem. Then using numerical optimization algorithm to solve the nonlinear optimization problem. This result to a large scale problem as the time points becomes large to approximate a continuous input. To efficiently solve this problem, we plan to use Autodiff and modify some part of the control module to efficiently calculate the gradient of the cost functionw with respect to the inputs at each time points. With the gradient, it is possible to numerically solve the optimal u(t). We will explore this for the remaining of the semester.

In parallel, we will refine both the 3D visualization and the web-based video gallery UI. On the visualization side, we will add realistic textures to the Sun and Earth and adjust lighting for clearer orbit depiction. On the UI side, we will tidy the layout, standardize captions, and integrate any new animations from the optimization experiments.

Division of Work

Member	**Focus Area**	**Current Deliverables**
Tianyang Hu	System implementation	CR3BP dynamics; simulation pipeline
Mark Zhang	3D rendering using `PyVista`	JWST halo orbit scenes; 3D `.mp4` animations
Haoran Yan	Trajectory optimization	Experiments on refined halo/transfer trajectories
Shiqi Fan	Video gallery; documentation	Write-up; web-based results gallery

References

[1] Ben Weber. Circular restricted three-body problem. https://orbital-mechanics.space/the-n-body-problem/circular-restricted-three-body-problem.html. Accessed: 2025-10-23.

[2] Richard M. Murray, Steven L. Brunton, et al. The python control systems library (python-control). https://python-control.readthedocs.io/, 2022. Version 0.9.4; accessed 2025-10-23.

[3] J. D. Giorgini. Three-body periodic orbits. https://ssd.jpl.nasa.gov/tools/periodic_orbits.html#/periodic. Accessed: 2025-10-26.

[4] C. B. Sullivan and A. Kaszynski. Pyvista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk). *Journal of Open Source Software*, 4(37):1450, 2019.