National Aeronautics and Space Administration

NASA Langley Research Center

ENGINEERING DIRECTORATE

Atmospheric Flight and Entry Systems Branch

Anthony Williams and Zachary May (NASA Langley Research Center), and Colin Brown (Barrios Technology)

End-to-End Trajectory Optimization Using Copernicus and Program to Optimize Simulated Trajectories II

IEEE Aerospace Conference, March 2-9 2024, Big Sky, Montana

Background

- **Copernicus is a trajectory design and optimization software developed at NASA Johnson Space Center**
  - Used as the primary trajectory design tool for the Orion program
  - 3 degree of freedom (DoF) simulation, focused on interplanetary trajectories, currently models exoatmospheric flight
  - Has a suite of built-in optimizers
  - Generalized plugin capability that allows users to incorporate calls to external applications and to ingest their data

- **Program to Optimize Simulated Trajectories II (POST2) is a widely used, workhorse trajectory simulation tool that has been used to solve a variety of atmospheric ascent and entry problems, developed at NASA Langley Research Center**
  - Significant efforts have led to upgrading POST2 to incorporate modern programming paradigms, such as thread safe computations and increased modularity and flexibility
  - Development of an application programming interface (API), allowing POST2 to be called efficiently from other software
  - Multi-DoF simulation that has varying fidelities, and can include flight software such as guidance, navigation, and control (GNC) models

Motivation

- **Meeting current and future NASA mission requirements may require multiple specialized computational tools to interact to properly assess a system**
  - Leverage the strength of each tool to maximize overall performance
  - Ex: Couple an interplanetary trajectory design tool, Copernicus, with POST2 to allow:
    - Optimal end-to-end trajectory solutions
    - Modeling of ascent/descent, especially during atmospheric flight

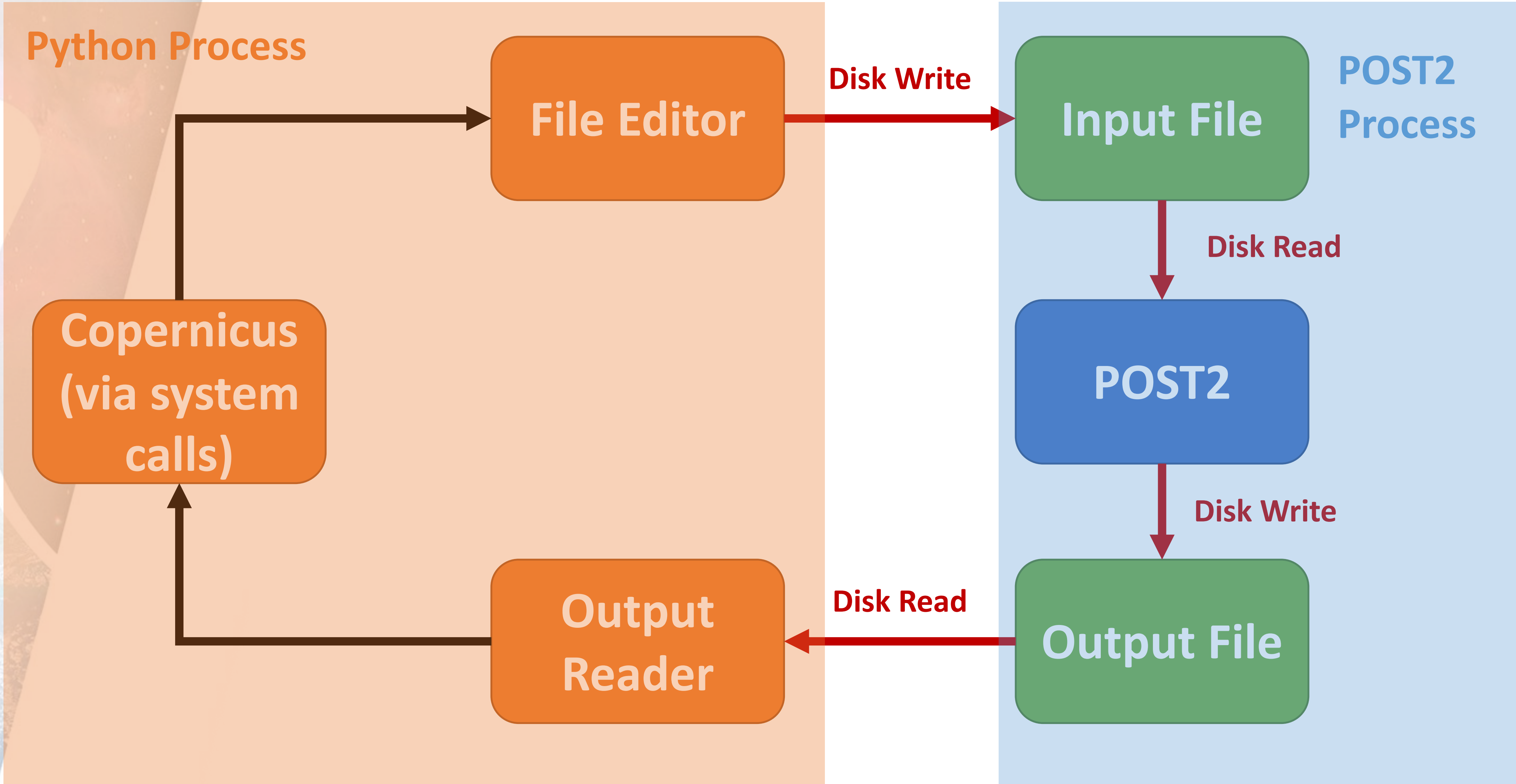
- **Problem:** how can information be passed between Copernicus and POST2 in an accurate and efficient manner?

- **Solution:** leveraging Copernicus Python plugin capability, along with the recent API development of POST2

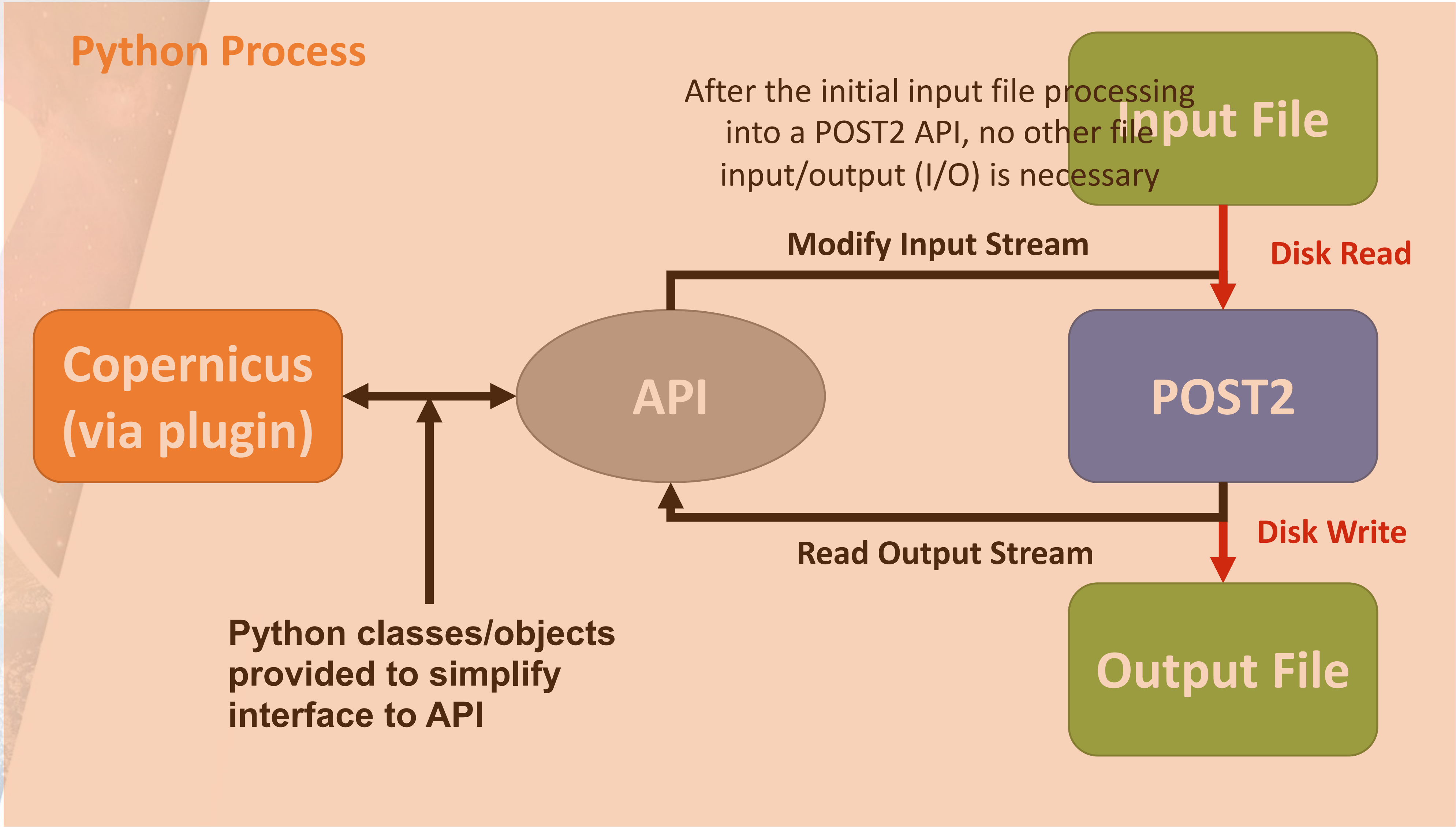
Outline

- **Interfacing Method for Copernicus and POST2**
- **Case Study: Lunar Ascent to Near-Rectilinear Halo Orbit (NRHO)**
  - Hyper grid approach (existing method)
  - Copernicus plugin + POST2 API approach (new method)
- **Solution Comparison**
- **Summary**

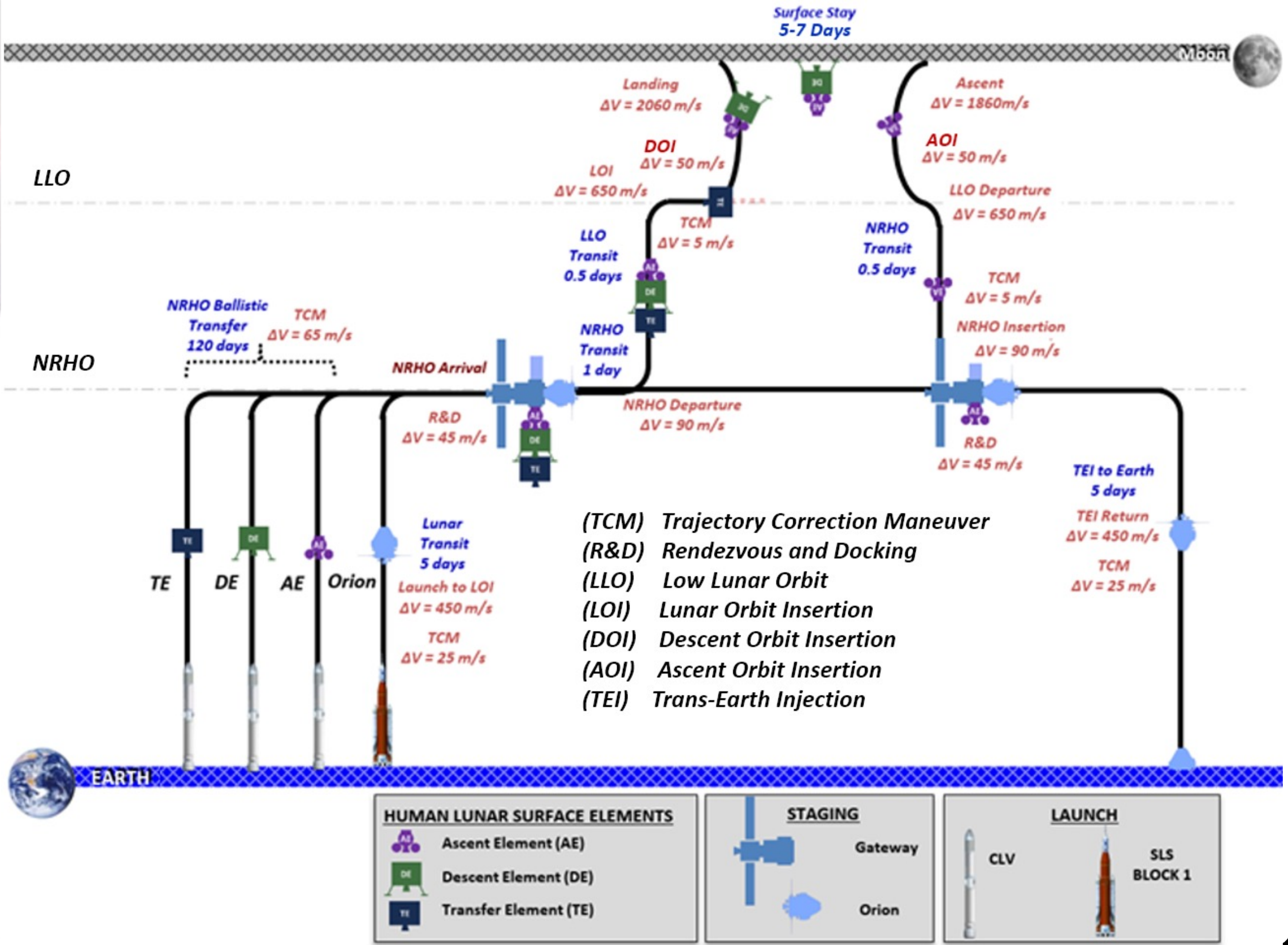
Existing Interfacing Method



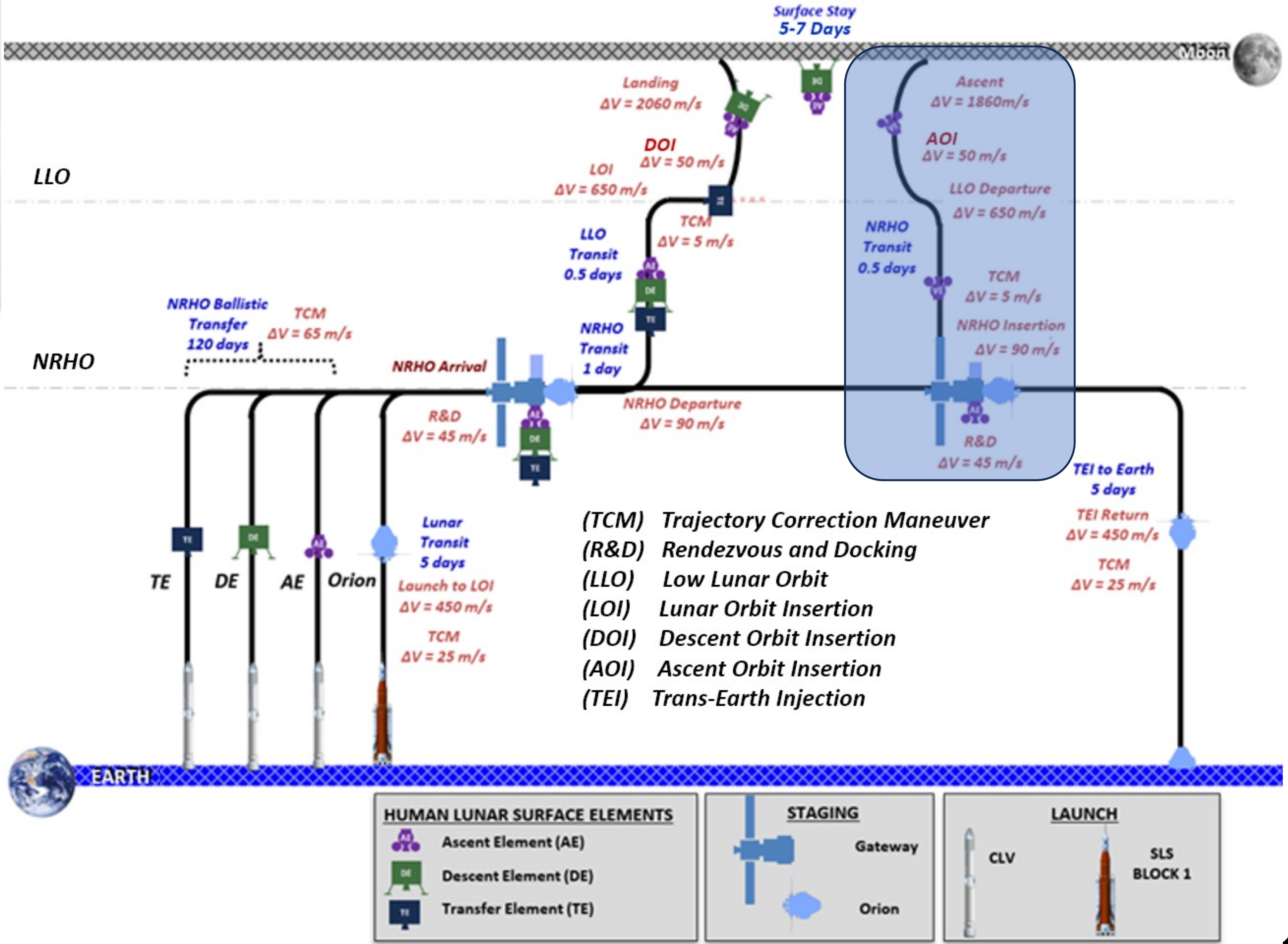
Leveraging POST2 API Method



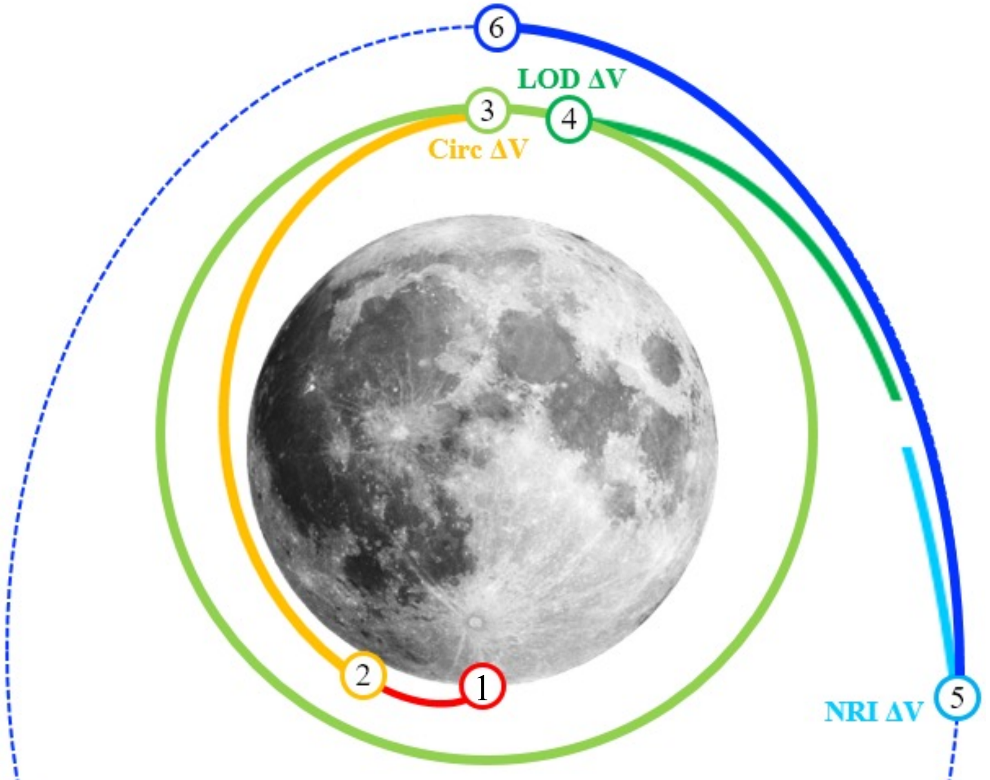
Reference Lunar Landing Mission



Case study: Lunar Ascent to NRHO



Case study: Lunar Ascent to NRHO



1 POST2 Lunar Ascent
2 Coast and Circularization
3 Coast to LOD
4 LOD to NRI, FP
5 NRI to LOD, BP
6 Gateway in NRHO

POST2

Copernicus

POST2 hands off states to Copernicus

All segments of the trajectory are optimized in an end-to-end sense using SNOPT within Copernicus

LOD: Lunar Orbit Departure
NRI: NRHO Insertion
BP/FP: Backward/forward propagation

Hyper Grid Approach

- **Hyper Grid will be utilized to characterize POST2 lunar ascent trajectory (segment 1 in previous slide)**

- **Hyper Grid is principally a set of multi-dimensional lookup tables or surrogate models to represent a flight phase**

- **The Hyper Grid should only be generated once due to computational cost; *<u>however, any change to ground rules or assumptions (vehicle configuration, etc.) warrants generating a new Hyper Grid</u>***

Hyper Grid Approach

- **For this case, 2-dimensional grid created using launch epoch and azimuth as design variables**
  - For each (epoch, azimuth) pair, 100 cases total, a unique POST2 trajectory was optimized *outside of Copernicus*, from liftoff to low LLO insertion

Must be done a priori

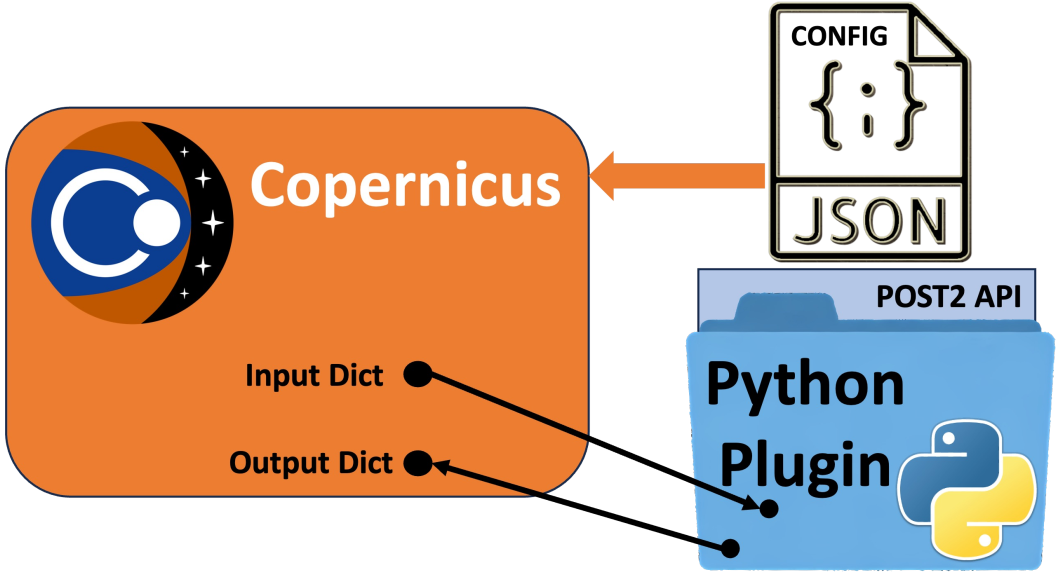
- **Position and velocity, epoch, and AE wet mass at LLO insertion captured for each POST2 trajectory**

- **Utilizing 2-D linear interpolation, AE LLO insertion position, velocity, epoch, and wet mass can be approximated for any point *within* (epoch, azimuth) design span**

Possible that optimal solution is outside design span

Plugin + POST2 API Approach

- **Leverages the generalized Copernicus Python plugin capability**
  - Python classes written for POST2 API allow for easier inclusion into plugin
- **Configuration JSON file maps to API methods to:**
  - Set POST2 simulation inputs
  - Propagate trajectory
  - Pass POST2 outputs back to Copernicus
- **Operating all within the same Python process**
  - Optimization of all segments occur in Copernicus
  - More efficient
  - Robust to configuration changes



Approach Comparison

Characteristic	Hyper Grid	POST2 Plugin
Robust To Configuration changes	No, likely requires new Hyper Grid generation	Yes, since trajectory problem definition can be quickly modified in POST2 Plugin

Approach Comparison

Characteristic	Hyper Grid	POST2 Plugin
Robust To Configuration changes	No, likely requires new Hyper Grid generation	Yes, since trajectory problem definition can be quickly modified in POST2 Plugin
Optimization Process	Lunar Ascent segment optimized outside of Copernicus, then remaining segments are optimized with Lunar Ascent solution from lookup table	Solution for all segments are contained in Copernicus, so segments are optimized globally

Approach Comparison

Characteristic	Hyper Grid	POST2 Plugin
Robust To Configuration changes	No, likely requires new Hyper Grid generation	Yes, since trajectory problem definition can be quickly modified in POST2 Plugin
Optimization Process	Lunar Ascent segment optimized outside of Copernicus, then remaining segments are optimized with Lunar Ascent solution from lookup table	Solution for all segments are contained in Copernicus, so segments are optimized globally
Information Passing Process Between POST2 and Copernicus	Through file input/output operations*	All through memory

Approach Comparison

Characteristic	Hyper Grid	POST2 Plugin
Robust To Configuration changes	No, likely requires new Hyper Grid generation	Yes, since trajectory problem definition can be quickly modified in POST2 Plugin
Optimization Process	Lunar Ascent segment optimized outside of Copernicus, then remaining segments are optimized with Lunar Ascent solution from lookup table	Solution for all segments are contained in Copernicus, so segments are optimized globally
Information Passing Process Between POST2 and Copernicus	Through file input/output operations*	All through memory
POST2 Propagation	Front loaded to Hyper Grid generation. Copernicus only interacts with terminal state of Lunar Ascent segment	Done inside of Copernicus, all within same Python process utilizing POST2 API

Approach Comparison

Characteristic	Hyper Grid	POST2 Plugin
Robust To Configuration changes	No, likely requires new Hyper Grid generation	Yes, since trajectory problem definition can be quickly modified in POST2 Plugin
Optimization Process	Lunar Ascent segment optimized outside of Copernicus, then remaining segments are optimized with Lunar Ascent solution from lookup table	Solution for all segments are contained in Copernicus, so segments are optimized globally
Information Passing Process Between POST2 and Copernicus	Through file input/output operations*	All through memory
POST2 Propagation	Front loaded to Hyper Grid generation. Copernicus only interacts with terminal state of Lunar Ascent segment	Done inside of Copernicus, all within same Python process utilizing POST2 API
Data Interpolation	Necessary to provide optimal Lunar Ascent segment output**	N/A

*known to be inefficient, but can also lead to truncation error of data

**can introduce a source of error

Solution Comparison – Design Variables

Design Variable	Hyper Grid	POST2 Plugin
Launch Epoch	Jan 4, 13:52:45 UTC	Jan 4, 13:31:54 UTC
Launch Azimuth	75.5 deg	75.8 deg
Circularization ΔV_{Mag}	19.8 m/s	20.1 m/s
LOD ΔV_{Mag}	648.6 m/s	648.4 m/s
NRI ΔV_{Mag}	58.6 m/s	58.5 m/s
NRI_{Final} AE Propellant	**3108.5 kg**	**3108.5 kg**

- Other than Launch Epoch (difference ~21 min), no significant difference in design variables
- Both approaches converged to same objective function value for AE propellant remaining

Solution Comparison – Run Time

Optimization Step	Hyper Grid	POST2 Plugin
Transfer and Launch State	9.7 sec	9.7 sec
POST2 Lunar Ascent	1225.8 sec	3.9 sec
End-to-End	2.7 sec	11.1 sec
Total	1238.2 sec	24.7 sec

- POST2 Plugin approach, leveraging API, was approximately *50 times faster* than the Hyper Grid approach
- Hyper Grid solution was run serially, which run time could have been improved if running trajectory simulations in parallel

Solution Comparison – Hyper Grid Interpolation Error

LLO Insertion	Hyper Grid	POST2	Error
R_x	81341 m	82904 m	-1563 m
R_y	307499 m	307144 m	355 m
R_z	-1724252 m	-1724252 m	-1 m
V_x	411 m/s	419 m/s	-8 m/s
V_y	1612 m/s	1610 m/s	2 m/s
V_z	306 m/s	306 m/s	0 m/s
AE Wet Mass	11327 kg	11327 kg	0 kg
Epoch	01/04/24 13:38:11 UTC	01/04/24 13:59:1 UTC	20 min 50s

- Small errors in AE state
- Hyper Grid solution had an LLO insertion epoch that was *before* the launch epoch

Design Variable	Hyper Grid
Launch Epoch	Jan 4, 13:52:45 UTC

Summary

- **An efficient and robust method of optimizing an end-to-end trajectory was presented**
  - Coupling Copernicus and POST2 using Copernicus' python plugin and POST2's newly developed API
  - Increased trajectory consistency due to direct communication of Copernicus with POST2
  - Demonstrated on a Lunar Ascent to NRHO problem
  - Enabling generalized capability for many projects
  - Capability can be extended to other tools, such as MONTE

- **Leveraging this method has multiple advantages over the existing, Hyper Grid method:**
  - More efficient in run time (**approximately 50 times faster**)
  - Each segment, including POST2 segment, is optimized within Copernicus (SNOPT)
  - No interpolation is needed, which lead to errors in the Hyper Grid Method
  - Much more robust to configuration changes

Backup

Future Work

- **Additional tools are currently in development for coupling with POST2, namely MONTE**

- **POST Explorer is a tool that allows for rapid design space exploration given a POST2 input deck**
    - Allows user to directly tweak inputs and visualize outputs
    - Useful for parametric sweeps or sensitivity studies
    - Built on top of the POST2 API

Ground Rules and Assumptions

POST2	
Gravity Model	Moon J2
Numerical Integrator	Runge Kutta 4th Order
Copernicus	
Gravity Model	Moon J2 Earth and Sun Point Masses
Numerical Integrator	Variable Order, Variable Step Adams Method
Lunar Landing Site	
Longitude	0.0°
Centric Latitude	-89.9°
Centric Radius	1737.4 km
Epochs	
Landing Epoch	December 29, 2023 15:58:00 UTC
Optimization Reference Epoch	January 04, 2024 15:58:00 UTC
AE Parameters	
Landed Inert Mass	6000 kg
Landed Propellant Mass	14000 kg
Surface Boiloff Rate	50 kg/day
Main Engine Isp	340 sec
Landed T/W_{Lunar}	1.2

Optimization Problem Definition

Objective = Maximize Propellant Remaining after NRI		
Segment	**Design Variables**	**Constraints**
1	Launch Epoch Launch Azimuth LTS* Burn Duration LTS* $Pitch_{Initial}$ Angle LTS* $Pitch_{Final}$ Angle	$Perilune_{Final}$ = 15.24 km $Apolune_{Final}$ = 100 km
2	Coast Duration Prograde ΔV_{Mag}	$Eccentricity_{Final} = 0$
3	Coast Duration	
4	LOD ΔV_x LOD ΔV_y LOD ΔV_z	Seg 4-5 R_x Continuity Seg 4-5 R_y Continuity Seg 4-5 R_z Continuity Seg 4-5 V_x Continuity Seg 4-5 V_y Continuity Seg 4-5 V_z Continuity
5	NRI ΔV_x NRI ΔV_y NRI ΔV_z	
Total	**14**	**9**