Low-Thrust Trajectory Optimization with Simplified SQP Algorithm

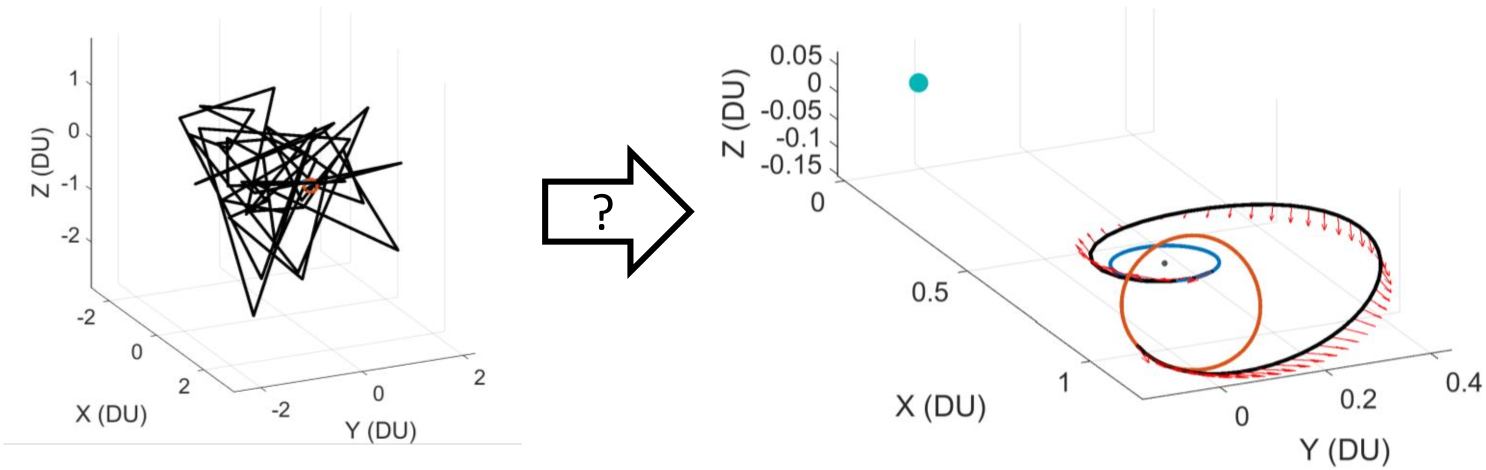
NATHAN L. PARRISH

DANIEL J. SCHEERES

UNIVERSITY OF COLORADO BOULDER

Introduction & Motivation

- Most nonlinear optimization methods require an initial guess “close” to the optimal solution
- Initial guess can be hard to find
- There may be many local optima, and the initial guess dictates which one is found
- General NLP “black box” solvers (IPOPT, SNOPT, …) can be slow
  - Many iterations
  - For large problems, most CPU time is spent in the optimizer

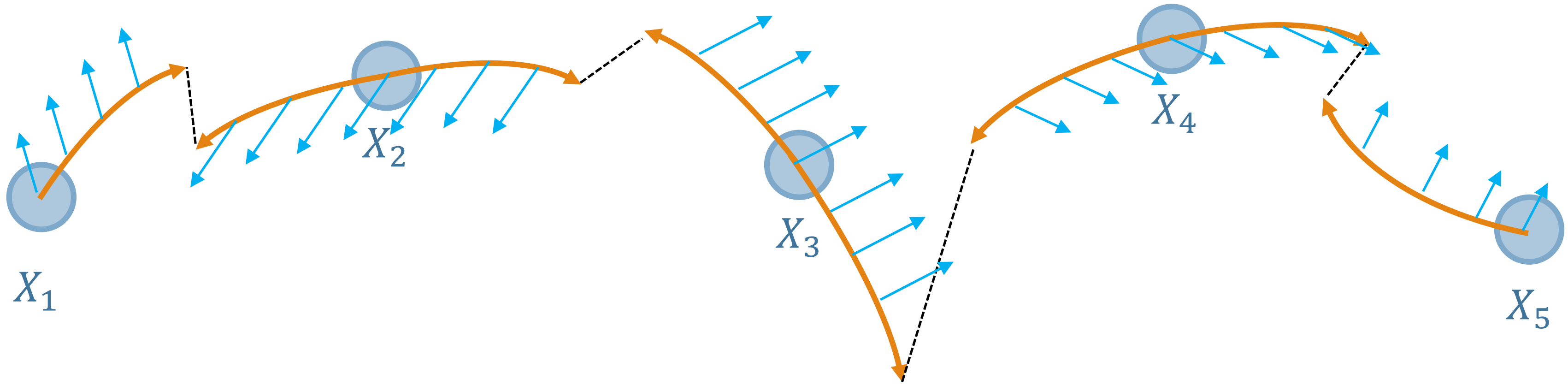


Solution Overview

- Start with very poor initial guess
- Use multiple shooting with ~50-100 nodes
  - Continuous thrust during propagation
  - Constrain defects to go to zero at matchpoints
- Optimize states, controls, and endpoint locations by solving a series of quadratic sub-problems

Multiple Shooting Formulation

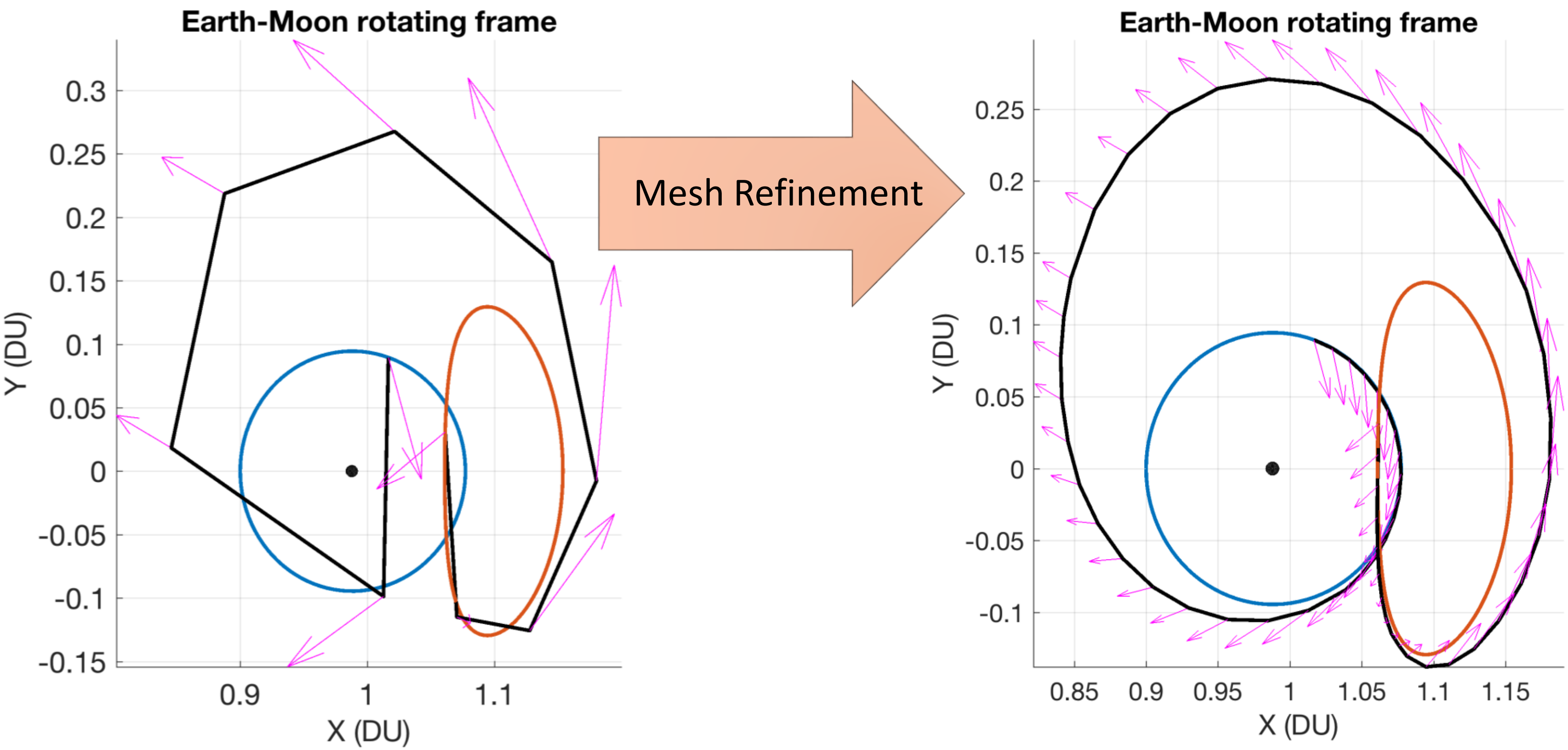
- Trajectory defined by a set of nodes with
  - Position
  - Velocity
  - Mass
  - Continuous control
- Propagate backwards and forwards in time
- Enforce continuity at matchpoints



Multiple Shooting Formulation

- Fixed step size integrator is used, with 5 steps per node
- Why fixed step size?
  - More consistent finite-differenced partial derivatives → faster convergence
  - Faster integration (don't get stuck at a singularity with poor initial guess)
  - Better for parallelization (future work)
- Runge-Kutta 78 numerical integration is used
  - Normally, use the 8^{th} order truncation term to estimate the error in the 7^{th} order step. Then choose the largest step size possible where the error remains within tolerance.
  - Here, we force a fixed step size, but use the truncation term to output the error estimate for use in mesh refinement
- Mesh refinement: Add nodes where the 8^{th} order truncation term for any of the integrator steps is > tolerance

Multiple Shooting Formulation



Then solve again with refined mesh

Traditional SQP Algorithm

- Minimize the Lagrangian:

$$\mathcal{L}(\vec{x}, \vec{\lambda}, \vec{\mu}) = \underbrace{f(\vec{x})}_{\text{objective}} + \vec{\lambda} \cdot \underbrace{\vec{h}(\vec{x})}_{\text{constrain} = 0} + \vec{\mu} \cdot \underbrace{\vec{g}(\vec{x})}_{\text{constrain} \leq 0}$$

- This is some nonlinear function which we don't know how to solve
- We do know how to solve Quadratic Programming problems, so approximate the nonlinear problem as quadratic:
  - Two-term Taylor series expansion of $f(\vec{x})$:

$$f(\vec{x}) \approx f(\vec{x}^k) + \nabla f(\vec{x}^k) \cdot \delta\vec{x} + \frac{1}{2}\delta\vec{x} \cdot Hf(\vec{x}^k) \cdot \delta\vec{x}$$

  - One-term Taylor series expansion of constraints:

$$\vec{h}(\vec{x}) \approx \vec{h}(\vec{x}^k) + \nabla\vec{h}(\vec{x}) \cdot \delta\vec{x}$$

$$\vec{g}(\vec{x}) \approx \vec{g}(\vec{x}^k) + \nabla\vec{g}(\vec{x}) \cdot \delta\vec{x}$$

- Sequential Quadratic Programming
  - Solve a sequence of quadratic programming (QP) problems that approximate the general nonlinear programming problem

SQP Algorithm Variant

- Minimize:

$$f = \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{3} (u_{ij} + \delta u_{ij})^2}_{\text{Truly quadratic objective}}$$

- Subject to:
  - Dynamics constraints:

$$\vec{d} + J \cdot \delta\vec{X} = \vec{0}$$

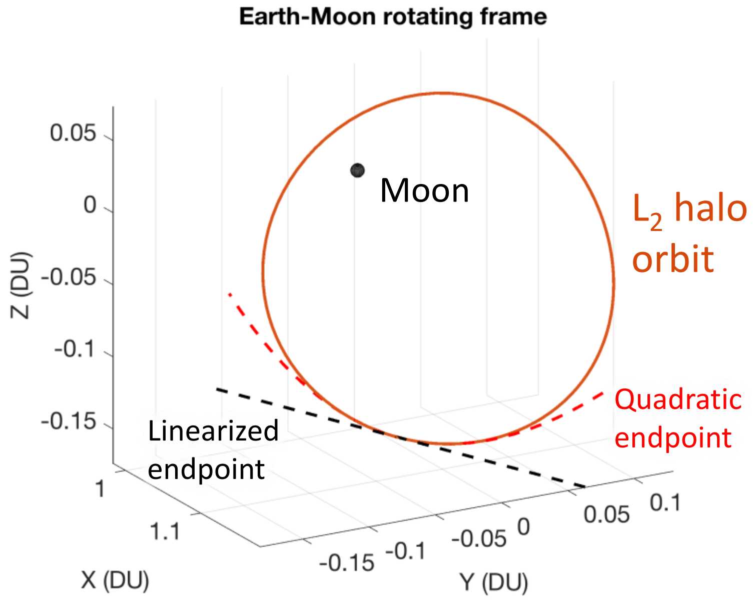
$\vec{d}$ = defects

$J = \frac{\partial \vec{d}}{\partial \vec{X}}$

$\delta\vec{X}$ = update to all optimization variables

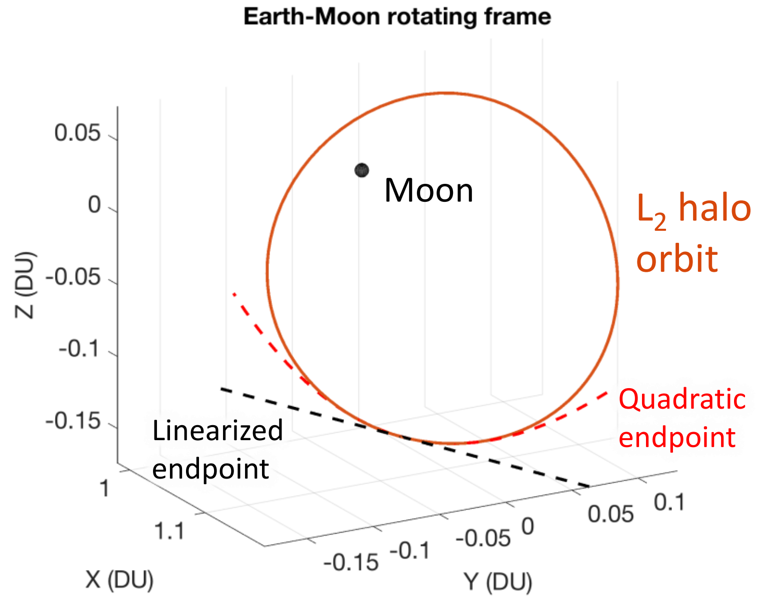
Endpoints

- Previously demonstrated that this approach (or even ordinary least squares) can be used to optimize trajectories when the endpoints and time of flight are fixed
- Now extend to variable endpoints & time of flight
- Easy (fast) to solve problems with:
  - Linear equality constraints
  - Quadratic inequality constraints
  - Quadratic cost
- Hard (slow) to solve problems with any higher order
- Problem: Linearized endpoint does not capture dynamics well
- Solution: use linear equality constraints and add quadratic endpoint term to cost



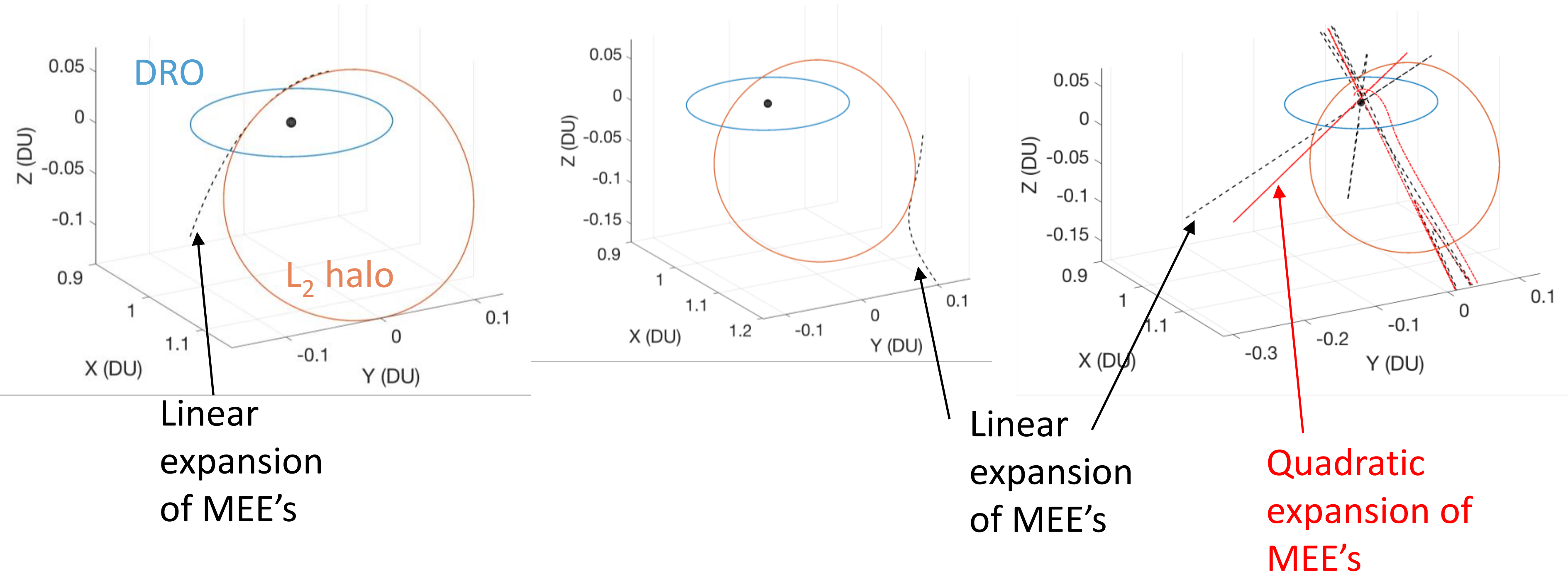
Endpoints

- Define endpoint: $\vec{q}(\tau) =$ L_2 halo orbit, defined by a set of points in a text file
- True endpoint constraint: $\vec{h}_e = \begin{bmatrix} \vec{r}_e \\ \vec{v}_e \end{bmatrix} - \vec{q}(\tau) = \vec{0}$
- Quadratic expansion of endpoint:
  $$\vec{q}(\tau) \approx \vec{q}(\tau^k) + \left.\frac{\partial \vec{q}(\tau)}{\partial \tau}\right|_{\tau_k} \delta\tau + \frac{1}{2}\left.\frac{\partial^2 \vec{q}(\tau)}{\partial \tau^2}\right|_{\tau_k} \delta\tau^2$$
- Linear endpoint equality constraint:
  $$\vec{h}_e = \begin{bmatrix} \vec{r}_e \\ \vec{v}_e \end{bmatrix} - \vec{q}(\tau^k) + \left.\frac{\partial \vec{q}(\tau)}{\partial \tau}\right|_{\tau_k}$$
- Add to objective function:
  $$f = f_{path} + \beta \cdot \left\| \frac{\partial^2 \vec{q}(\tau)}{\partial \tau^2} \right\| \cdot \delta\tau^2$$
- With β too small, solution bounces around optimal τ indefinitely
- With β too large, solution converges prematurely on sub-optimal τ



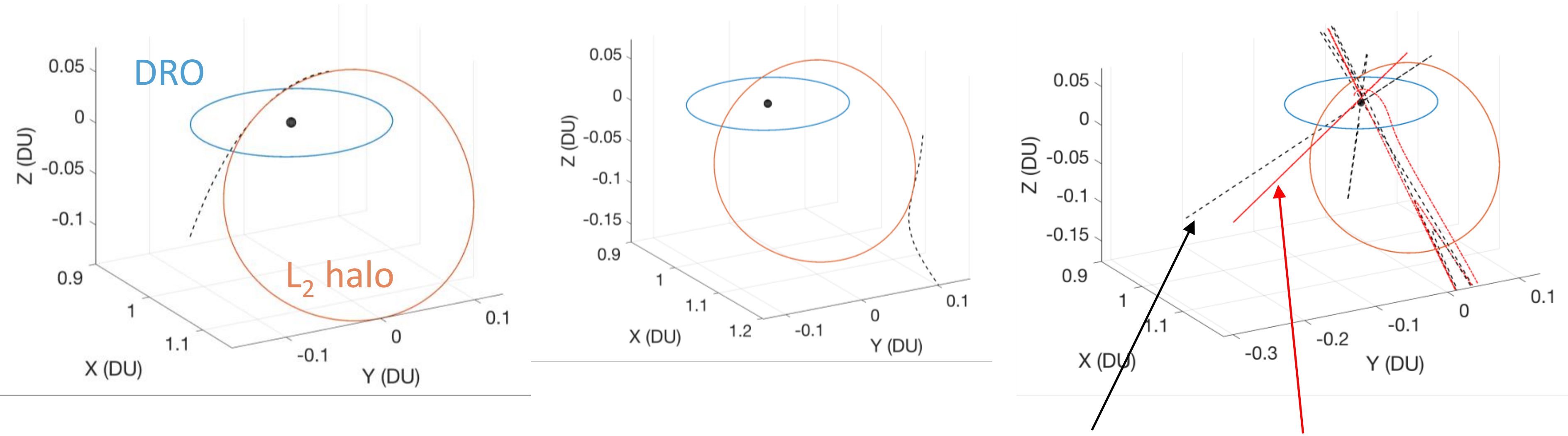
Endpoints

- What if we use a linear expansion with a different set of parameters?
- Tried Modified Equinoctial Elements, unsuccessful
- Works well sometimes (when far from singularities)
- Totally fails sometimes (when close to singularities)



Endpoints

- What if we use a linear expansion with a different set of parameters?
- Tried Modified Equinoctial Elements, unsuccessful
- Works well sometimes (when far from singularities)
- Totally fails sometimes (when close to singularities)



Quadratic expansion of MEE's

Line Search

- Each solution to the QP problem gives us an update $\delta\vec{x}$ to all optimization variables

$$\vec{X}^{k+1} = \vec{X}^{k} + \alpha \cdot \delta\vec{x}$$

- If the problem is sufficiently linear, the QP update is accurate enough to assume $\alpha = 1$
- Why do a line search?
  - We do not trust the solution to the linearized problem
  - $\vec{X}^{k+1} = \vec{X}^{k} + \alpha \cdot \delta\vec{x}$
- For short transfers (<1 revolution), no need to perform line search – the problem is sufficiently linear to converge quickly with full steps

Maratos effect

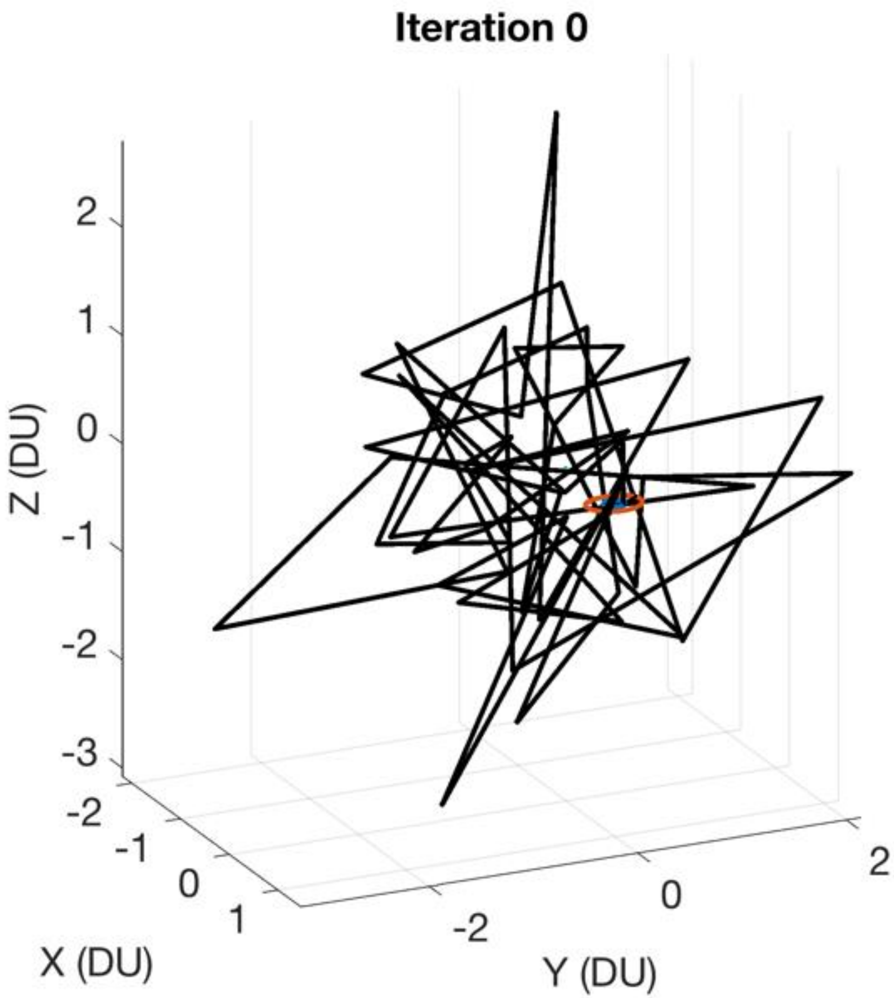
Line Search

- A comment on parameterization
  - Line search is only necessary as the solution takes on more revolutions
  - With a different parameterization (i.e. orbital elements), the revolutions can be “unwound” to keep the problem more linear
  - However, the optimization algorithm is too “smart” for this
    - Every orbital element set has some singularity (or multiple)
    - Optimization algorithm will exploit the singularity to find a non-physical solution with very low cost

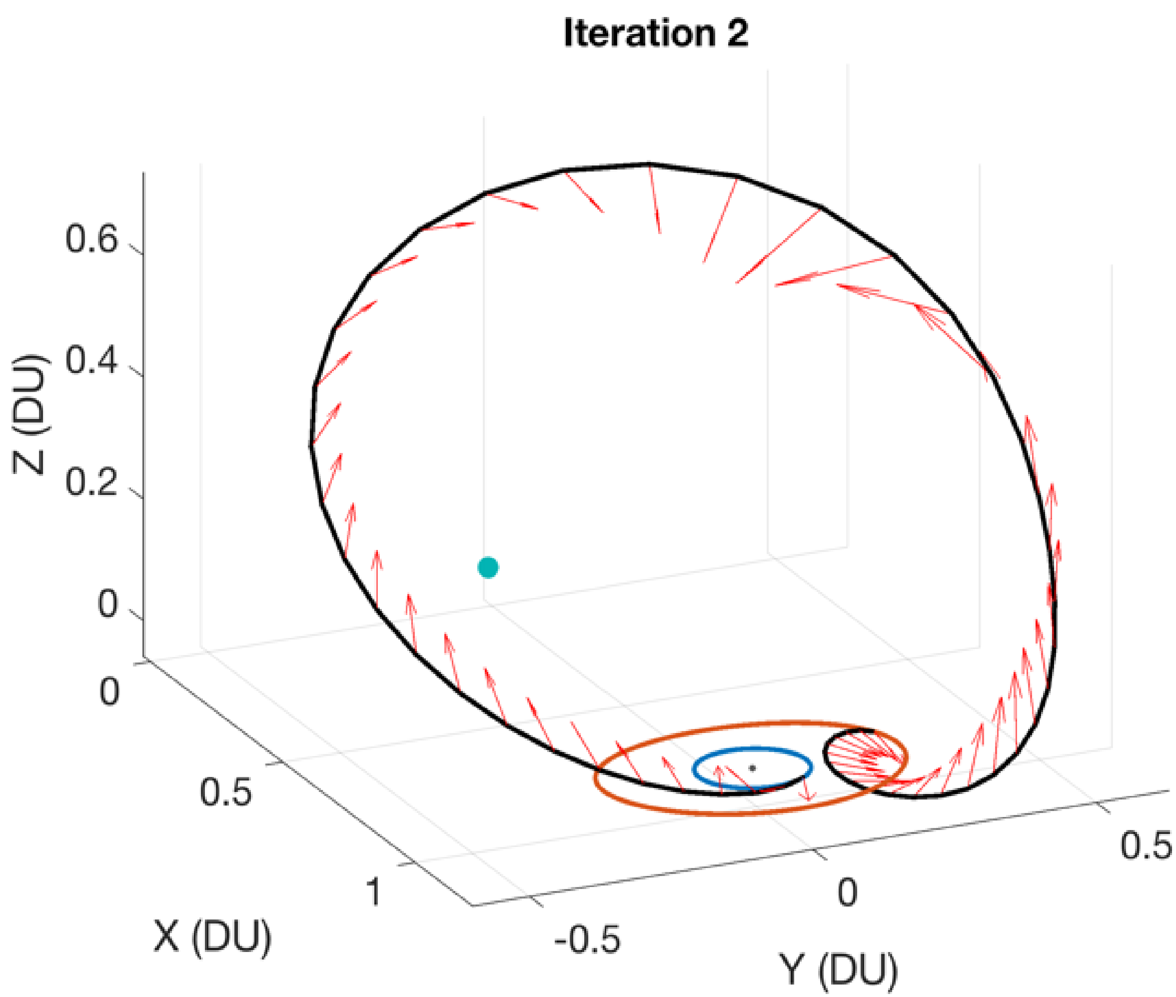
Example applications

- Now, two examples, with CRTBP dynamics
  - DRO (distant retrograde orbit) to L_2 halo orbit
  - DRO to different DRO
- Initial guess is random
- Endpoints and time of flight are variable, but only allowed to change a small amount each iteration, to preserve accuracy of linearization

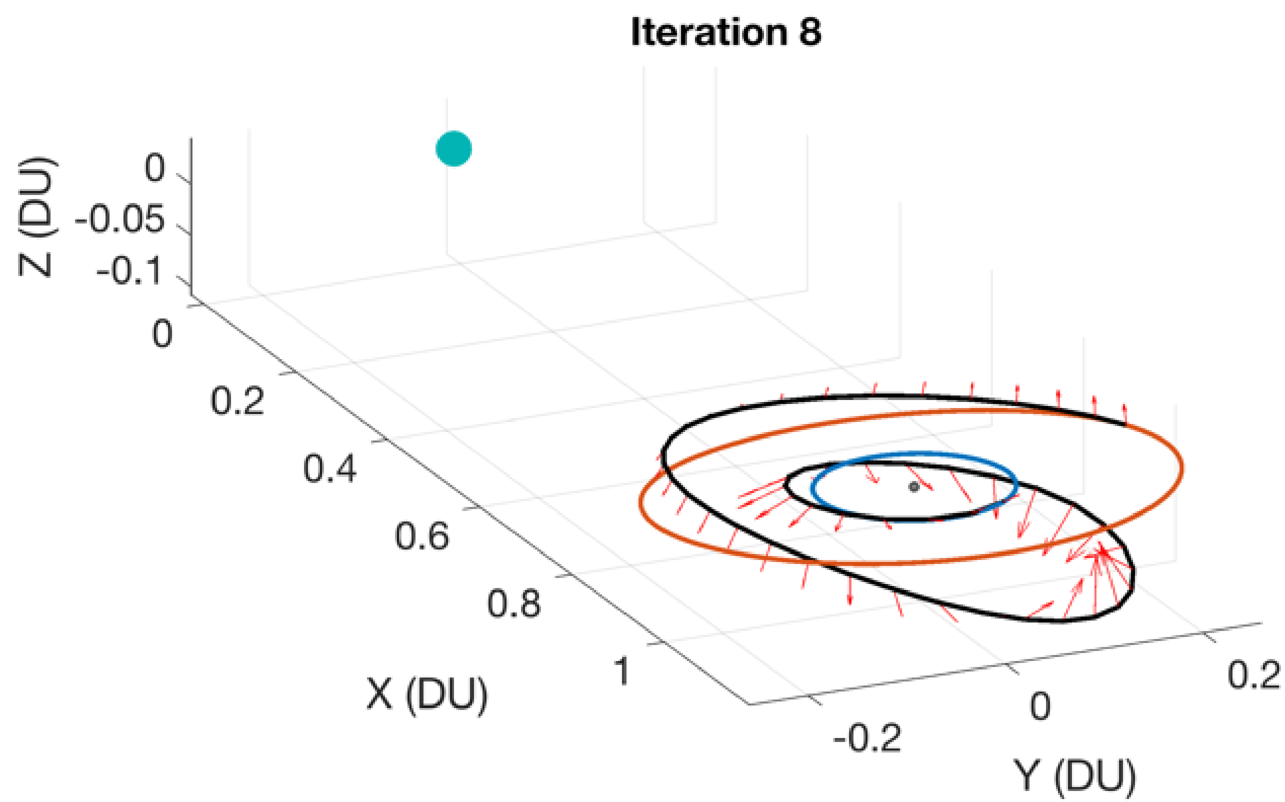
Example applications



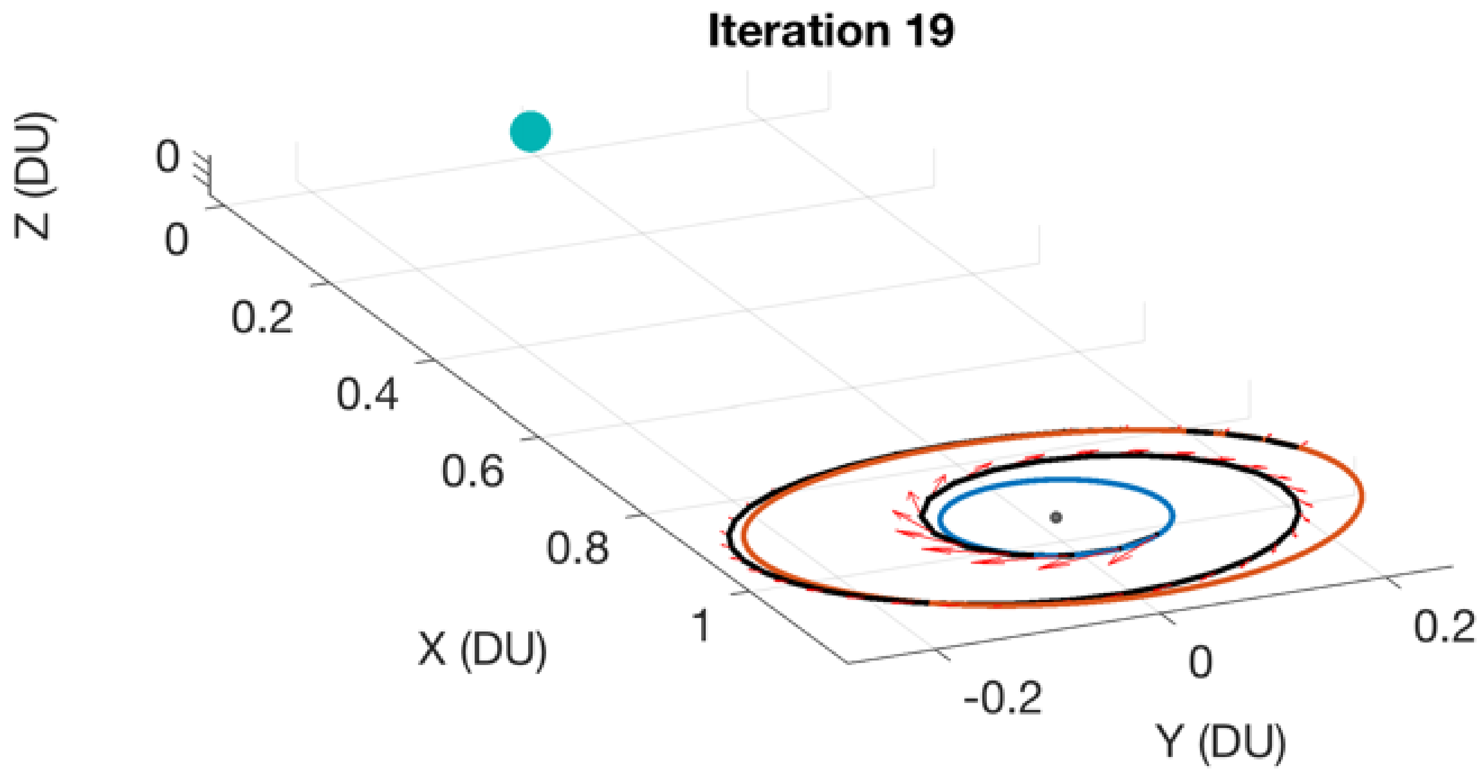
Example applications



Example applications

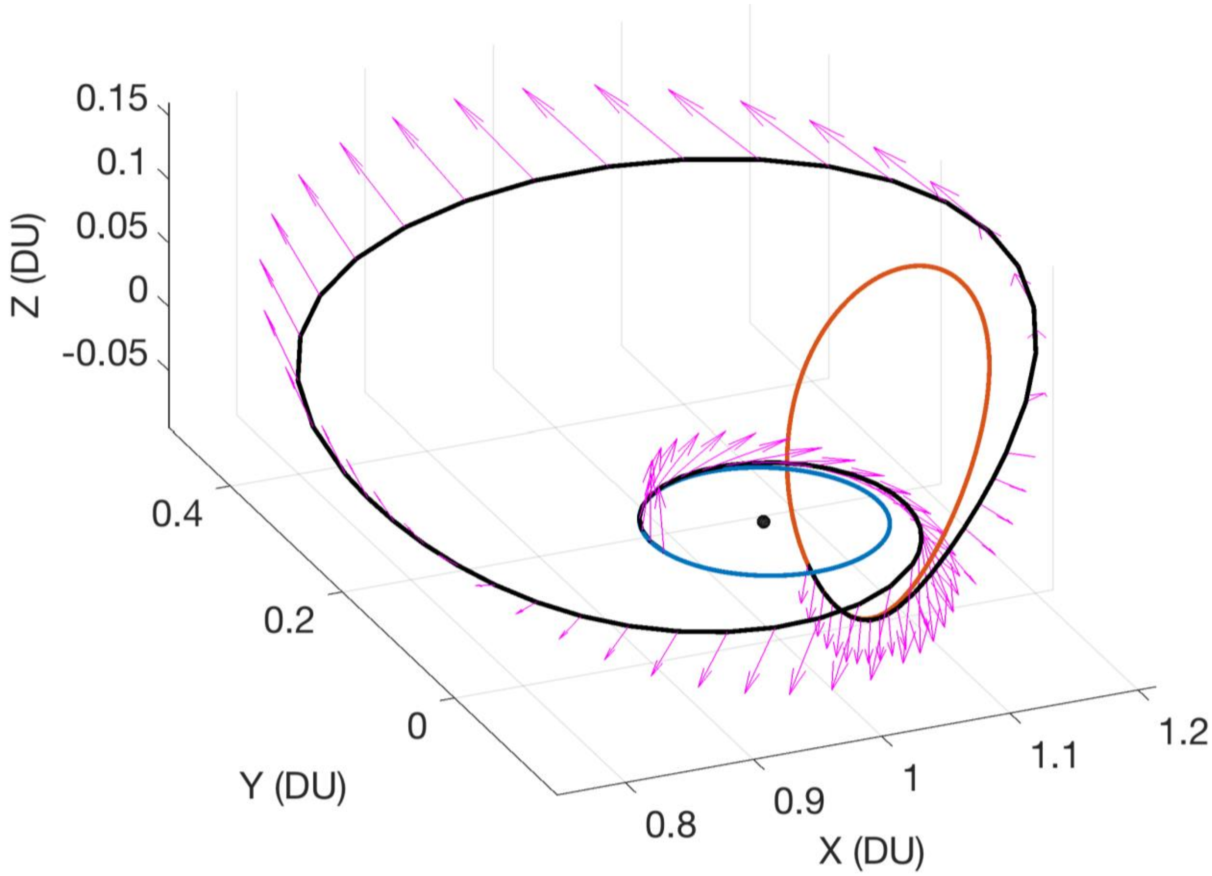


Example applications



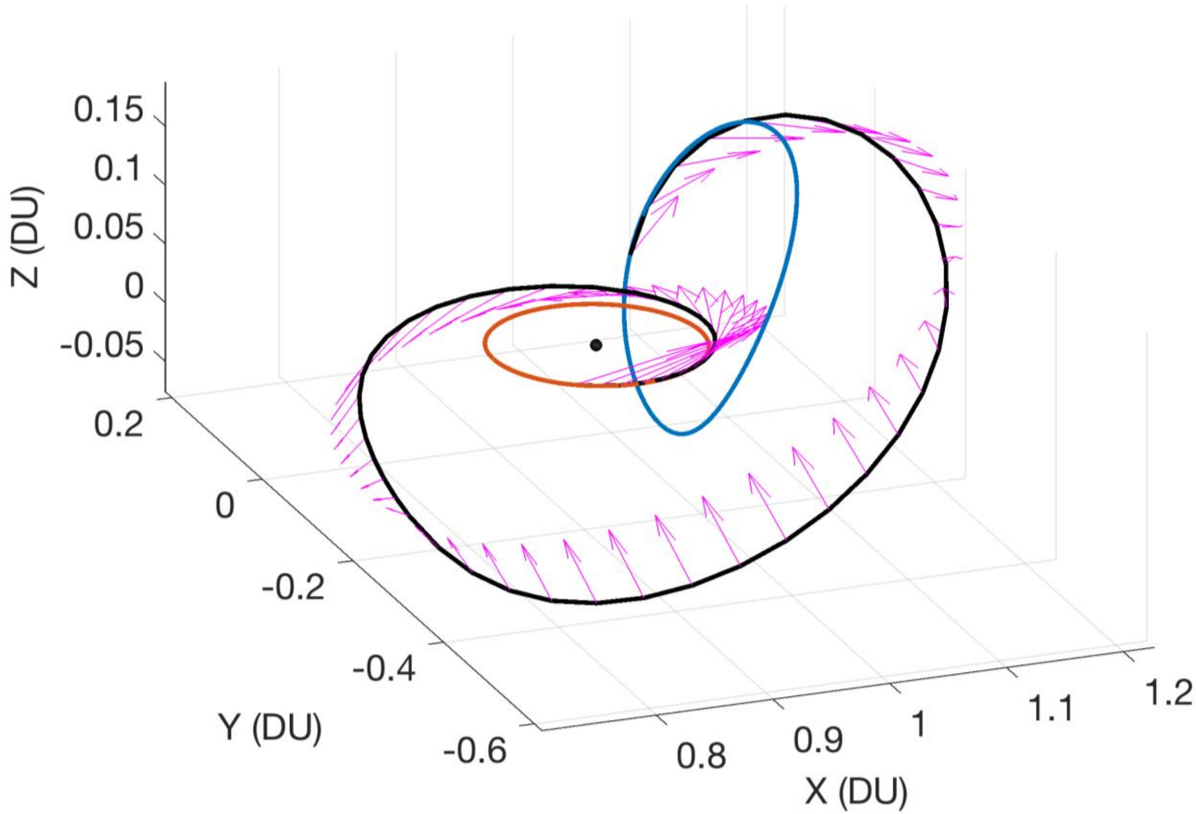
This transfer requires 15 days and an acceleration of 1.7E-4 m/s^2
(equivalently, 170 mN for a 1000 kg spacecraft)

Example applications



This transfer requires 29 days and an acceleration of 2.8E-4 m/s^2 (equivalently, 280 mN for a 1000 kg spacecraft)

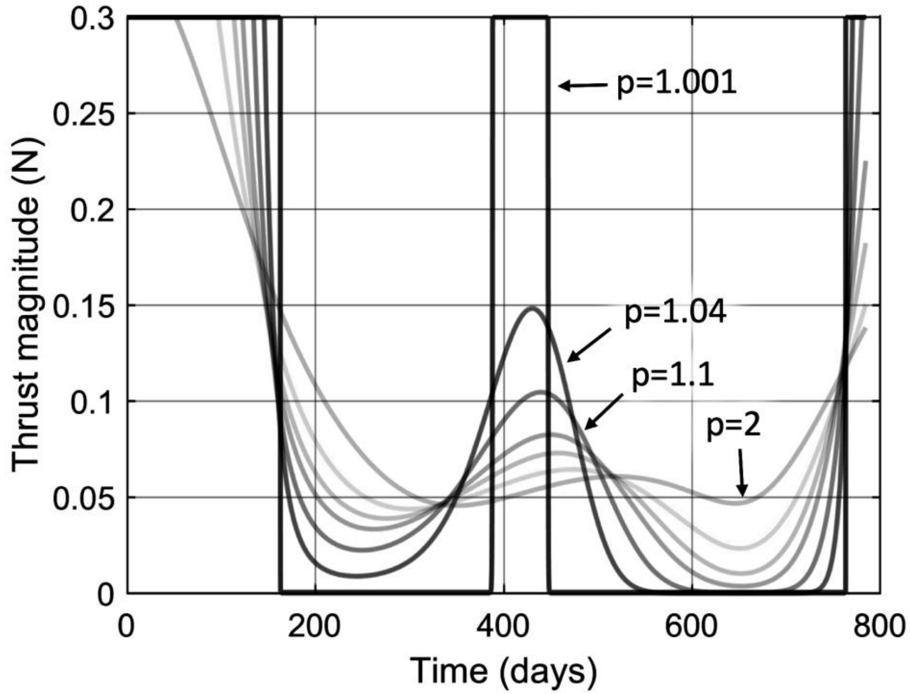
Example applications



This transfer requires 28 days and an acceleration of 2.8E-4 m/s^2 (equivalently, 280 mN for a 1000 kg spacecraft)

Fuel Optimal Solutions

- Previously demonstrated we can easily transition from one objective to another
  - $\int |u|^p \, dt$
  - With $p = 2$, large radius of convergence
  - With $p = 1$, small radius of convergence
  - Use homotopy method with control law to transition from $p = 2$ to $p = 1$



Example for Earth-Mars low-thrust rendezvous

Implementation notes

- Implemented in Julia language, with JuMP optimization toolbox and Gurobi as QP optimizer
- Computation time (40-100 nodes):
  - Each iteration:
    - Set up QP problem: 0.2 – 0.5 seconds
    - Solve QP problem: 0.2 – 0.5 seconds
    - Line search: 0.2 – 0.5 seconds
  - Short transfers total time
    - From random initial guess: 10 – 30 seconds
    - From close initial guess: ~1 – 3 seconds
  - Long transfers total time varies
    - Line search becomes necessary, so more iterations required
    - Does not always converge

Low-Thrust Trajectory Optimization with Modified SQP Algorithm

Questions?

NATHAN L. PARRISH

DANIEL J. SCHEERES

UNIVERSITY OF COLORADO BOULDER

This work was supported by a NASA Space Technology Research Fellowship