Numerical Methods for (Spacecraft Trajectory) Optimization

CK Venigalla

Guest Lectures, ASEN 6020

Personal Background

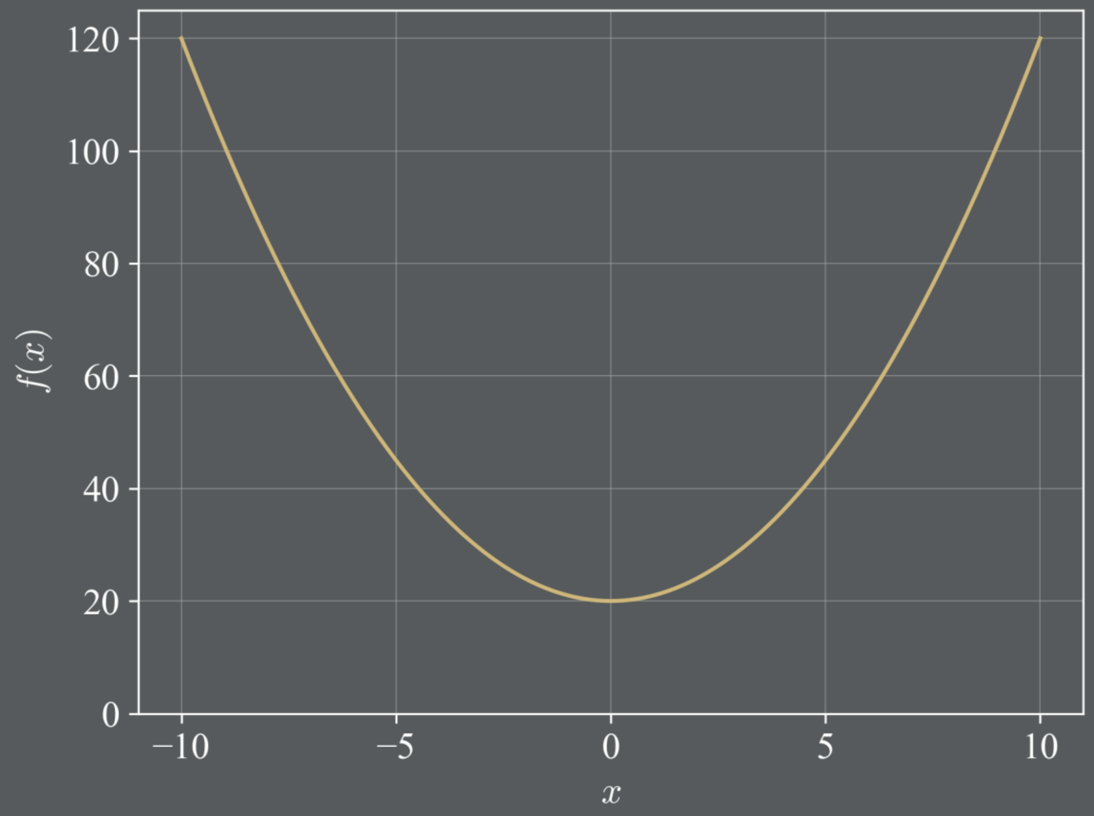
- 2016 – **B.S.** Aerospace Engineering and Physics **University of Virginia**
- 2018 – **M.S.** Aerospace Engineering Sciences **CU Boulder**
- 2021 – **Ph.D.** Aerospace Engineering Sciences **CU Boulder**
    - Thesis title: Multi-Spacecraft Cooperative and Non-Cooperative Trajectory Optimization (advisor: Prof. Scheeres)
- Related experience:
    - NASA Jet Propulsion Lab & NASA Goddard Space Flight Center (research collaboration & mission proposal work)
    - Helped develop several planetary science mission concepts
    - Created a low-thrust interplanetary trajectory optimization tool (NSTOP)

"Roughly speaking, local optimization methods are more art than technology. Local optimization is a well developed art, and often very effective, but it is nevertheless an art."

S. Boyd and L. Vandenberghe, *Convex Optimization*, 2009

Goal: describe some available "paint brushes" to help construct your own solution methods for problems of interest

Nomenclature



x “decision variable”

$f(x)$ “objective function”

$$\min_{x\in\mathbb{R}} f(x) = 20$$

$$\arg\min_{x\in\mathbb{R}} f(x) = 0$$

General Mathematical Programming (Parameter Optimization)

Minimize objective function: $J = f(\boldsymbol{\rho})$

$$\boldsymbol{\rho} = \{\rho_1, \rho_2, \ldots, \rho_N\}$$

Subject to constraints: $\boldsymbol{g}(\boldsymbol{\rho}) \leq 0$

Mathematical Programming Problem (MPP) variants based on properties of f and $\boldsymbol{g}$ include:

- Linear (LP)
- Convex
- Nonlinear (NLP)
- Integer / integer linear (ILP)
- Mixed Integer Nonlinear Programming (MINLP)
- Etc.

MPP Solution Methods

- Gradient Based
  - E.g., sequential quadratic programming (SQP), interior point methods
- Non-gradient based (meta-heuristic)
  - Population-based
    - E.g., genetic algorithms (GAs), particle swarm optimization (PSO)
  - Other
    - E.g., simulated annealing
- Hybrid
  - E.g., non-gradient based outer loop with gradient based inner loop

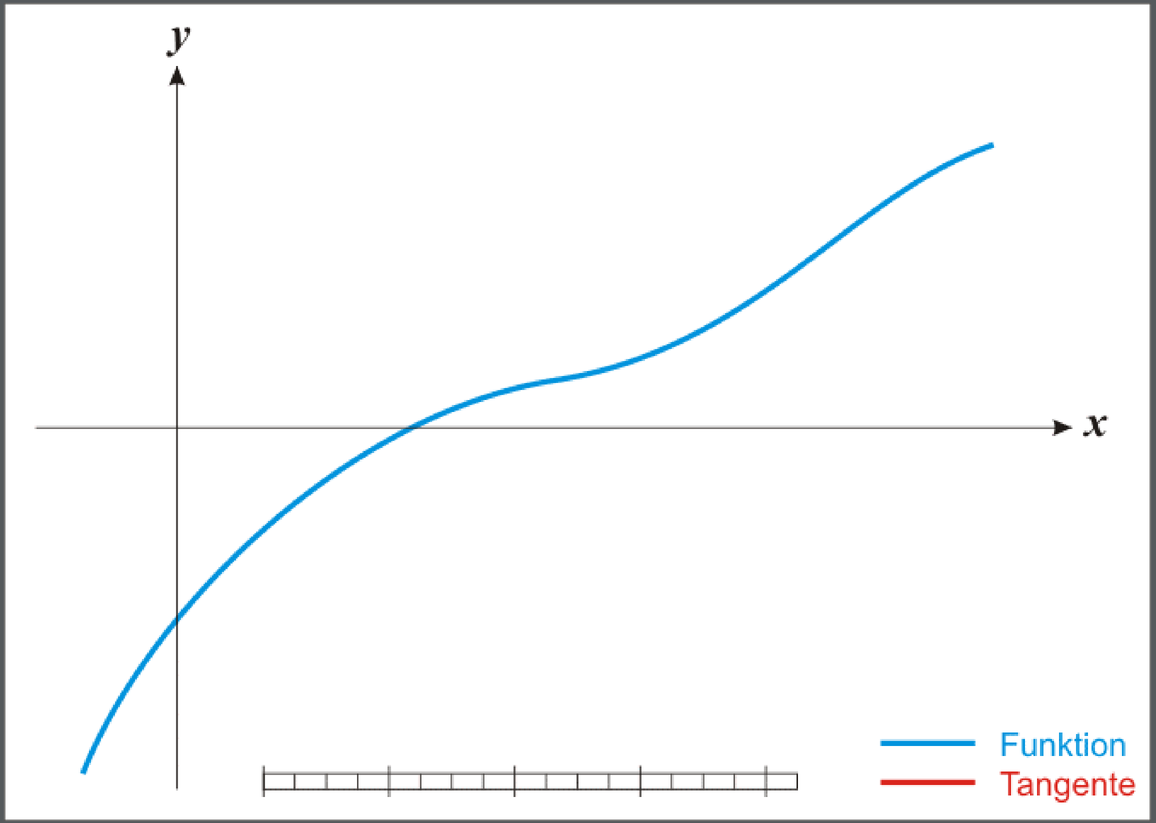
Not a strict hierarchy/classification!
Missing things like:

- simplex method (LP), dynamic programming, branch and bound, etc.
- Breakdown between stochastic and deterministic methods
- Variants of methods that place it into different categories (e.g. gradient-based PSO)
- Etc.

Gradient Based

Gradient-Based NLP Solver Basic Principle

- Use slope of the objective function to climb down/up to the minimum/maximum
- Similar to concept of newton's method of root finding:
  - Iteratively use derivative information and zero-target to step along function towards the zero value
- For optimization, target function value is unknown instead of zero



Newton's method for root finding animated
Image credit: Ralf Pfeifer/Wikipedia

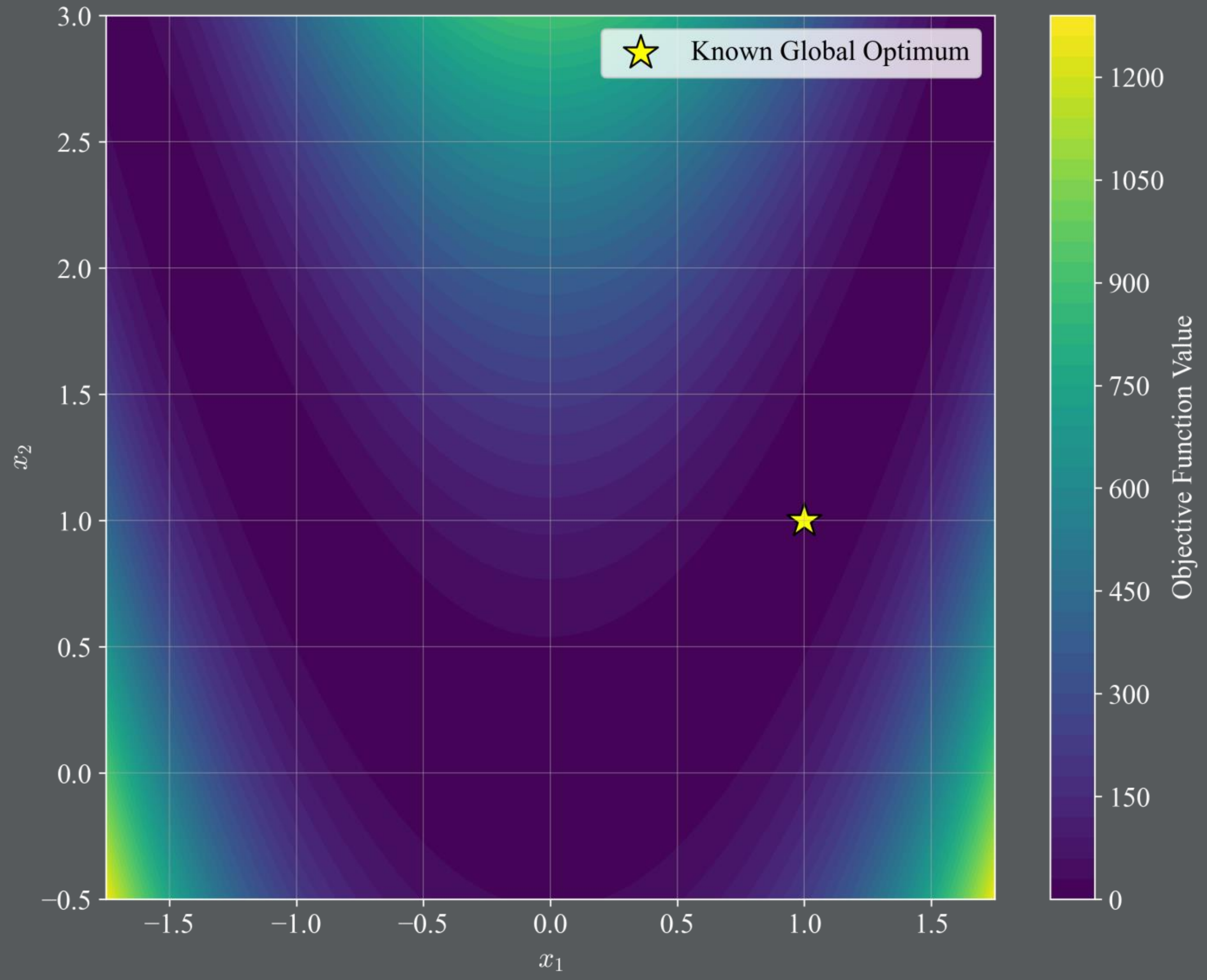
Gradient-Based NLP Solvers

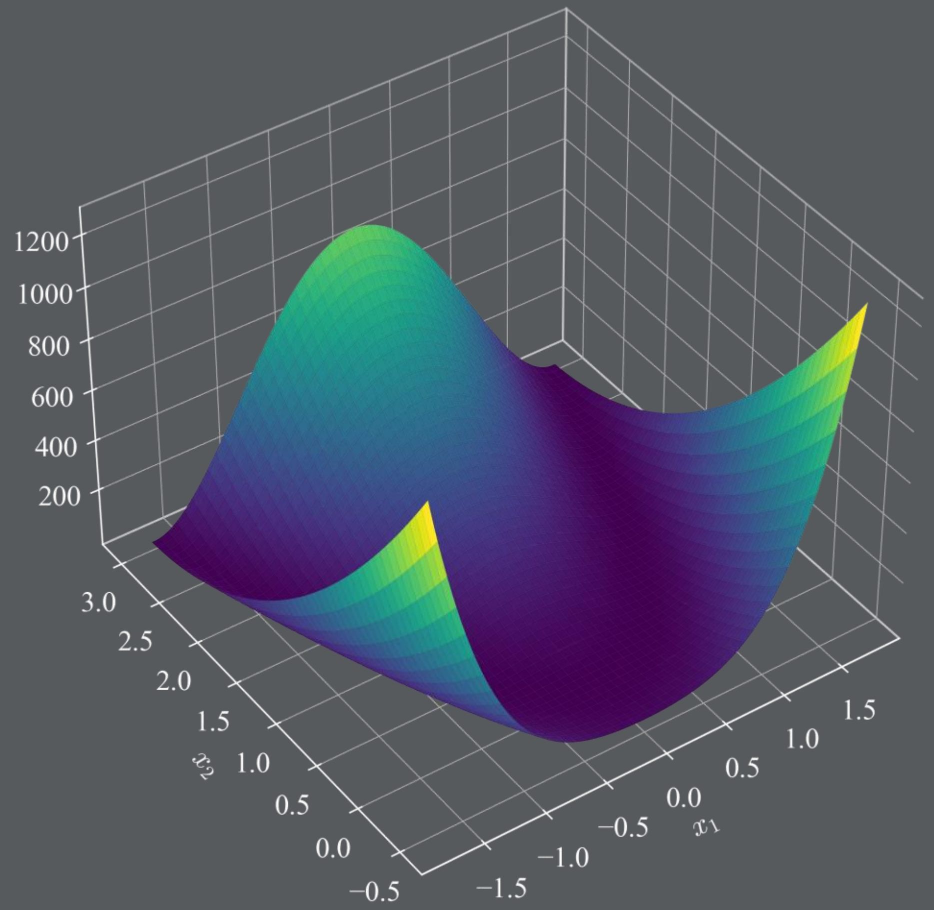
- SNOPT
  - Sequential quadratic method (SQP)
- IPOPT
  - Interior point method
- fmincon (MATLAB)
  - Interior point method by default, SQP and others available
- Many more algorithms and tools exist!

Sample Gradient-Based NLP Solutions

Sample nonlinear objectives to minimize in two dimensions for easy visualization (using IPOPT)

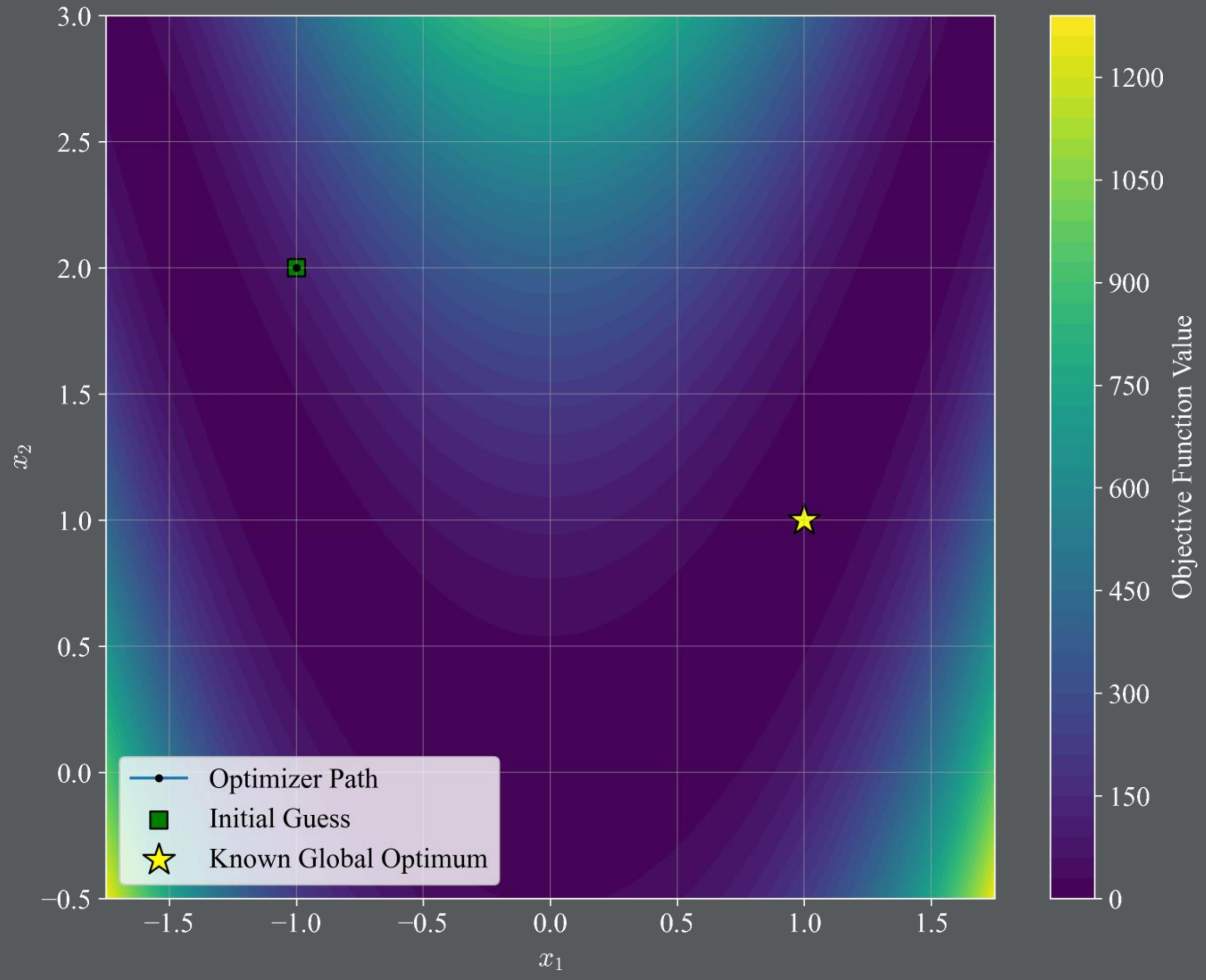
Rosenbrock Function

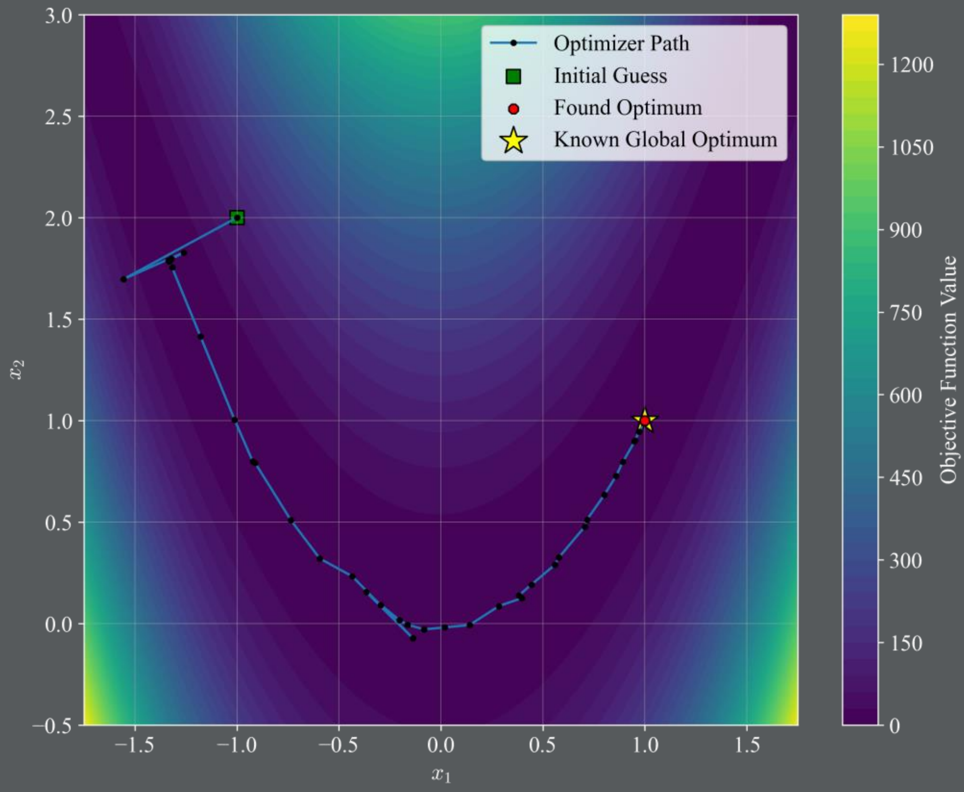




$$f(\boldsymbol{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

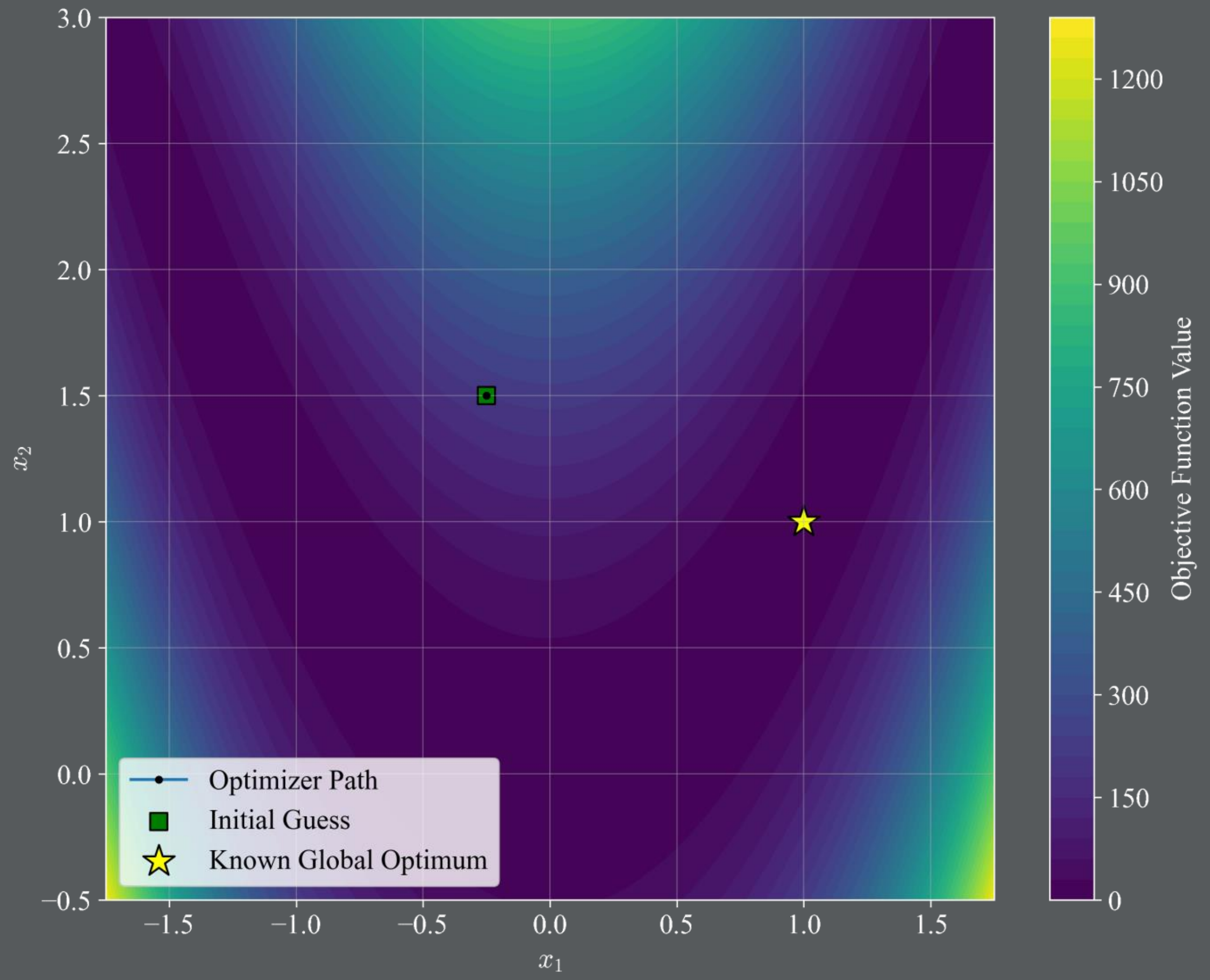
Rosenbrock Function Solution Path 1

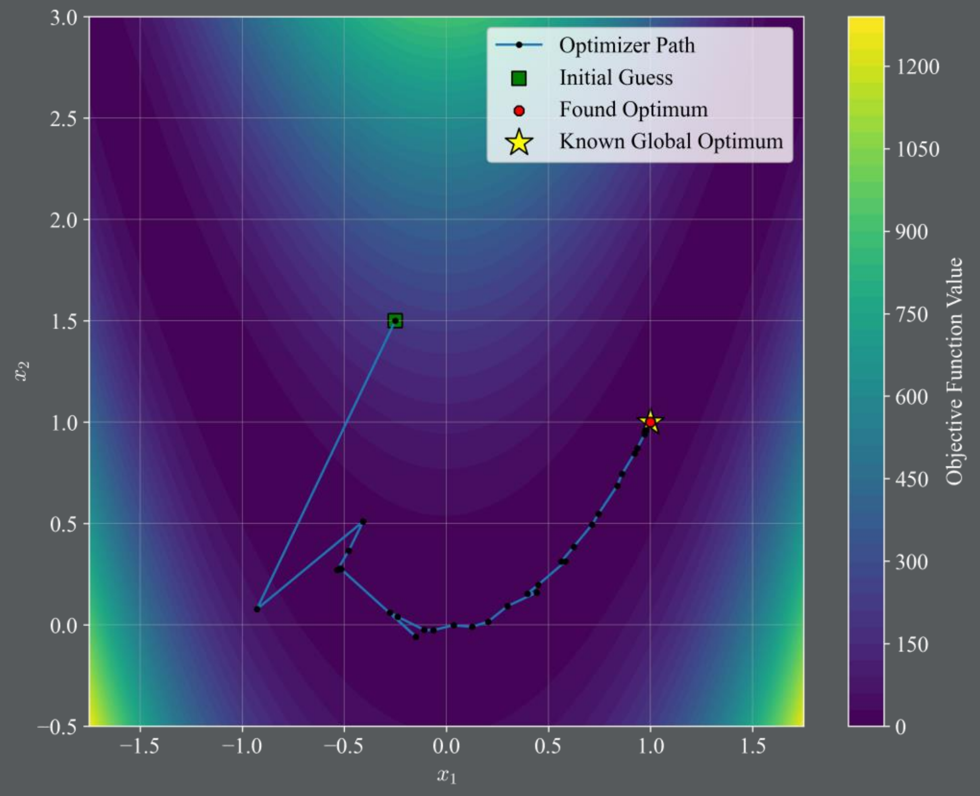




$$f(\boldsymbol{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

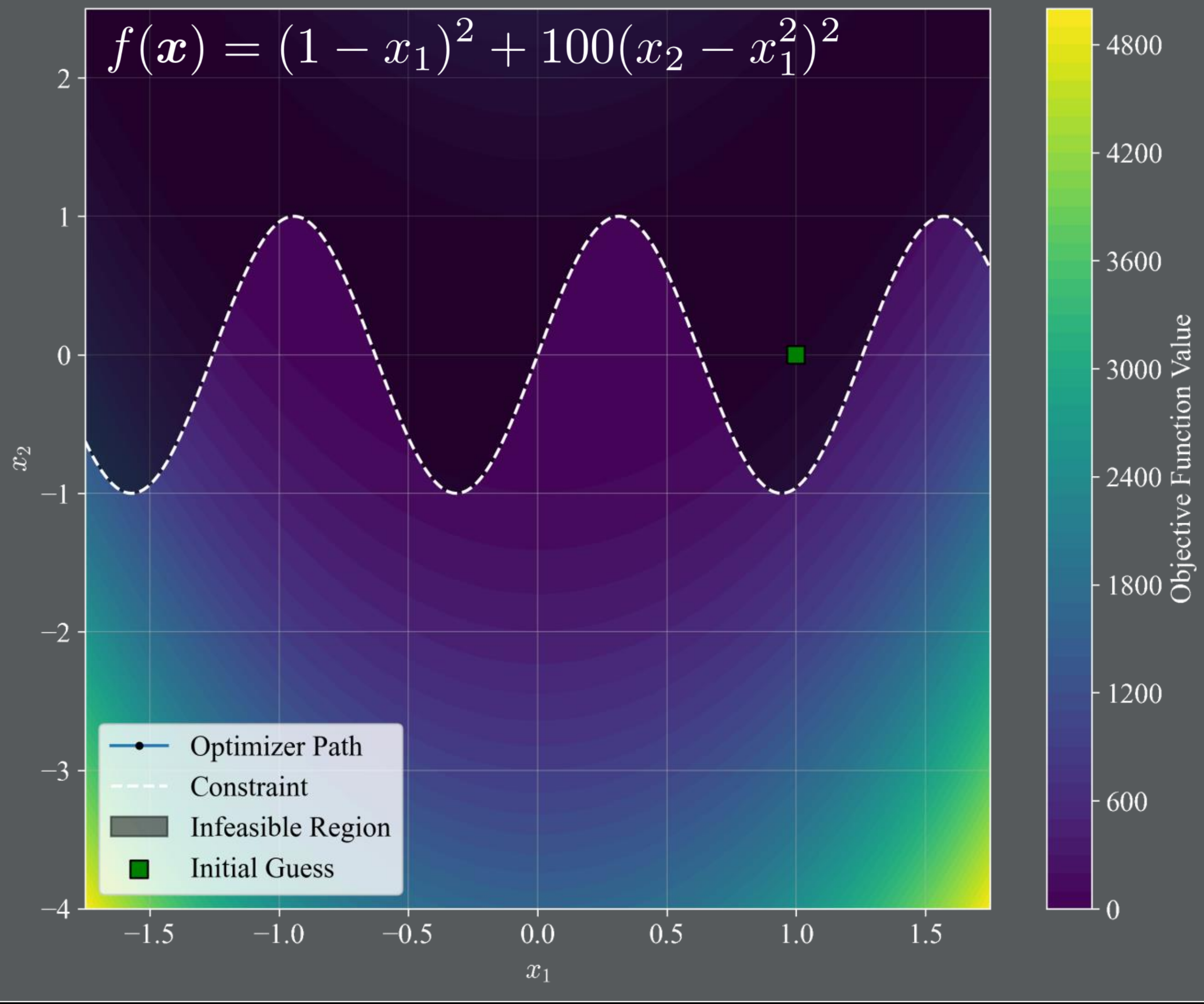
Rosenbrock Function Solution Path 2

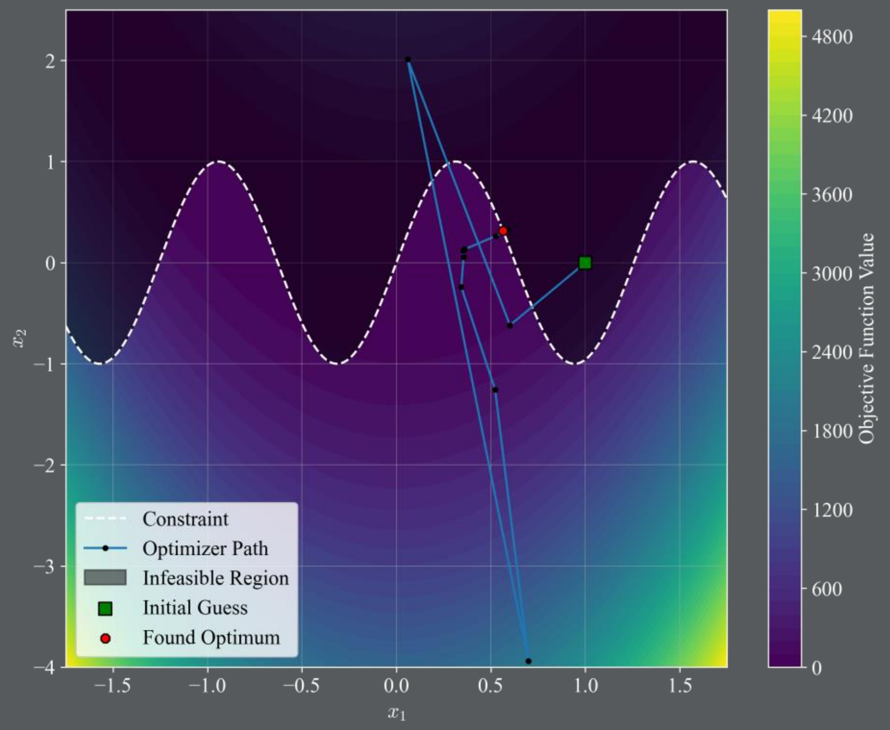




$$f(\boldsymbol{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

Constrained Rosenbrock Function Solution Path

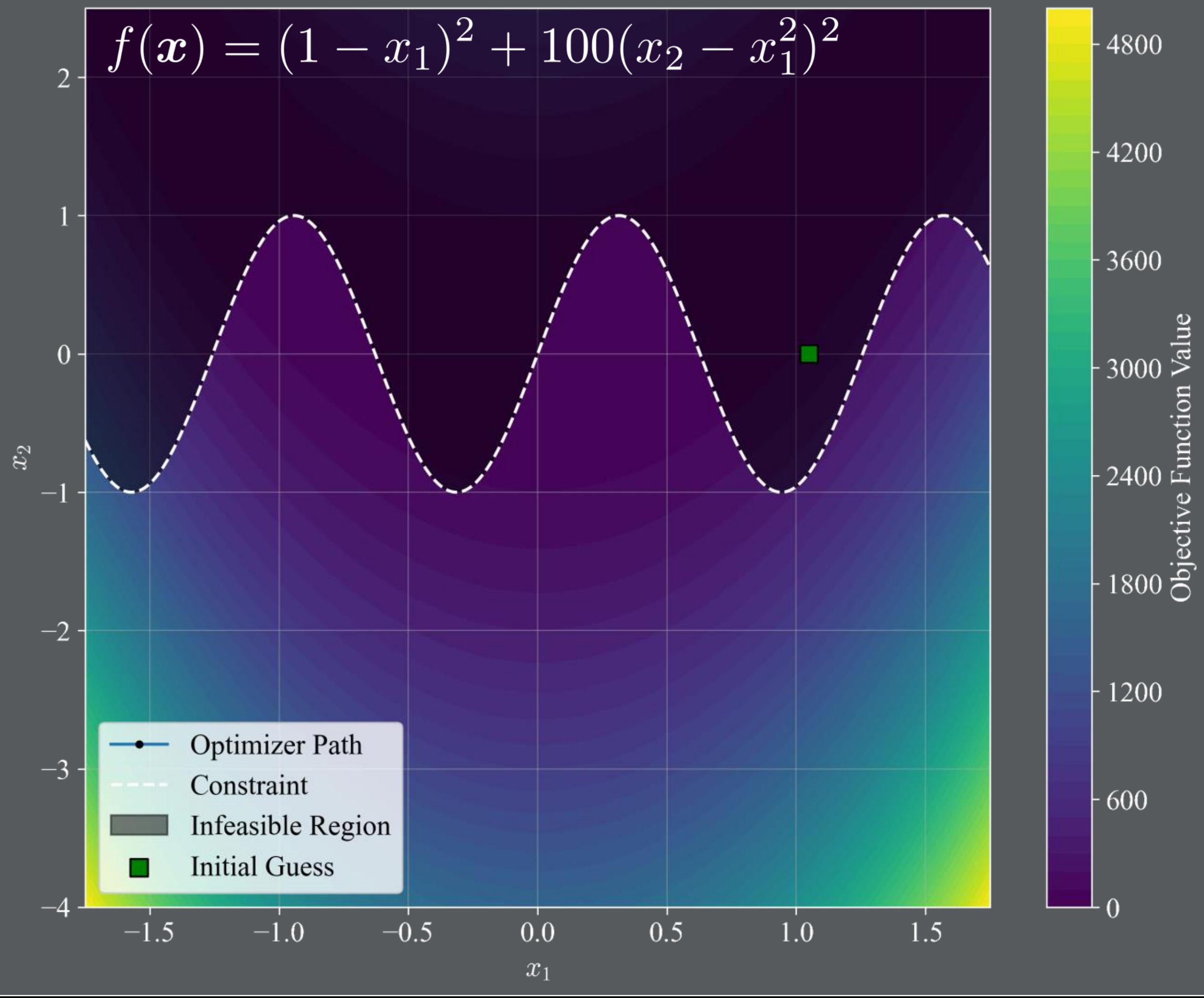


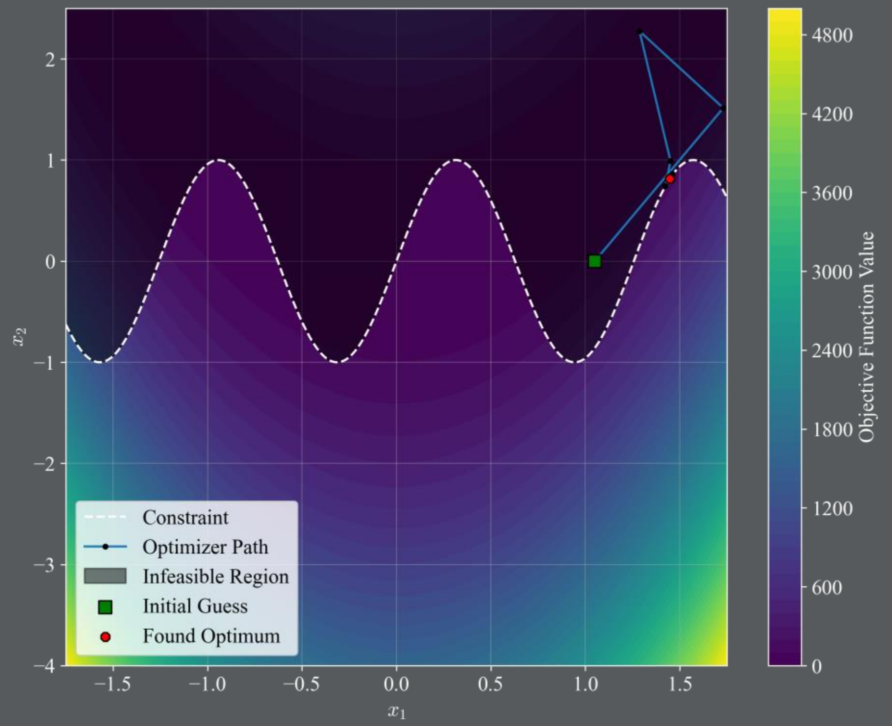


$$g(\boldsymbol{x}) = \sin(5x_1) - x_2 \geq 0$$

Initial guess: (1.0, 0.0)
Found optimum value: 0.1903
Iterations: 16

Constrained Rosenbrock Function Solution Path 2





$$g(\boldsymbol{x}) = \sin(5x_1) - x_2 \geq 0$$

Initial guess: (1.05, 0.0)
Found optimum value: 163.9
Iterations: 10

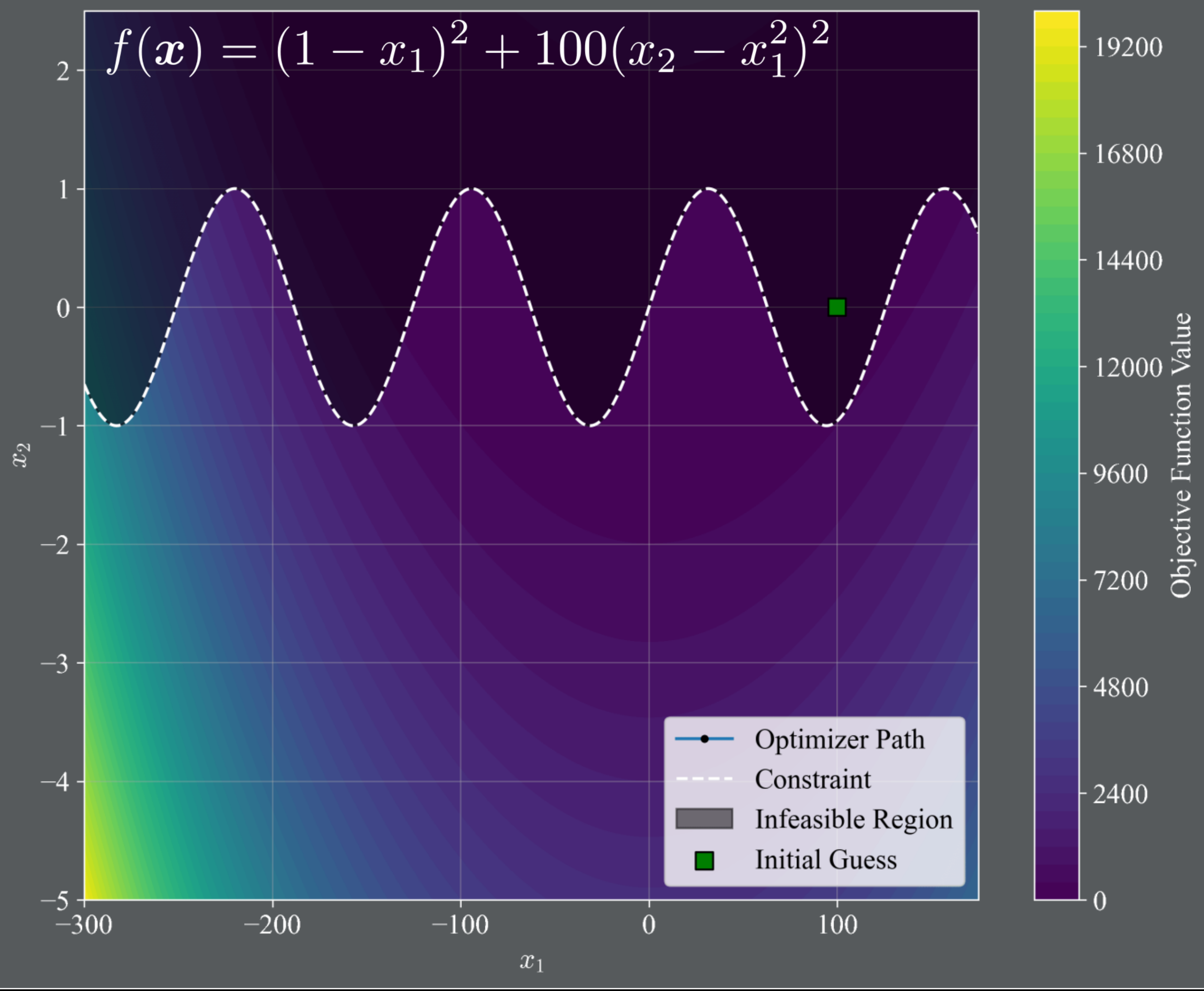
Constrained Rosenbrock Function Solution Poorly Scaled

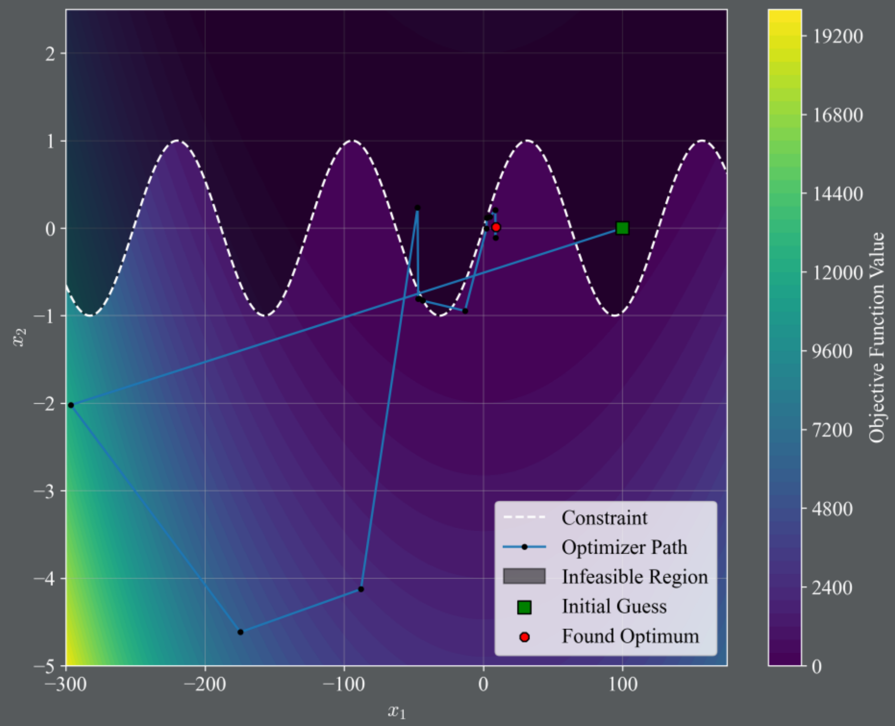
$$f(\boldsymbol{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \longrightarrow f(\boldsymbol{x}) = \left(1 - \frac{x_1}{100}\right)^2 + 100\left(x_2 - \left(\frac{x_1}{100}\right)^2\right)^2$$

$$g(\boldsymbol{x}) = \sin(5x_1) - x_2 \geq 0 \longrightarrow g(\boldsymbol{x}) = \sin\left(5\frac{x_1}{100}\right) - x_2 \geq 0$$

Disable auto-scaling in IPOPT, re-solve problem
(auto-scaling also does not necessarily fix scaling issues!)

Constrained Rosenbrock Function Solution Path Poorly Scaled

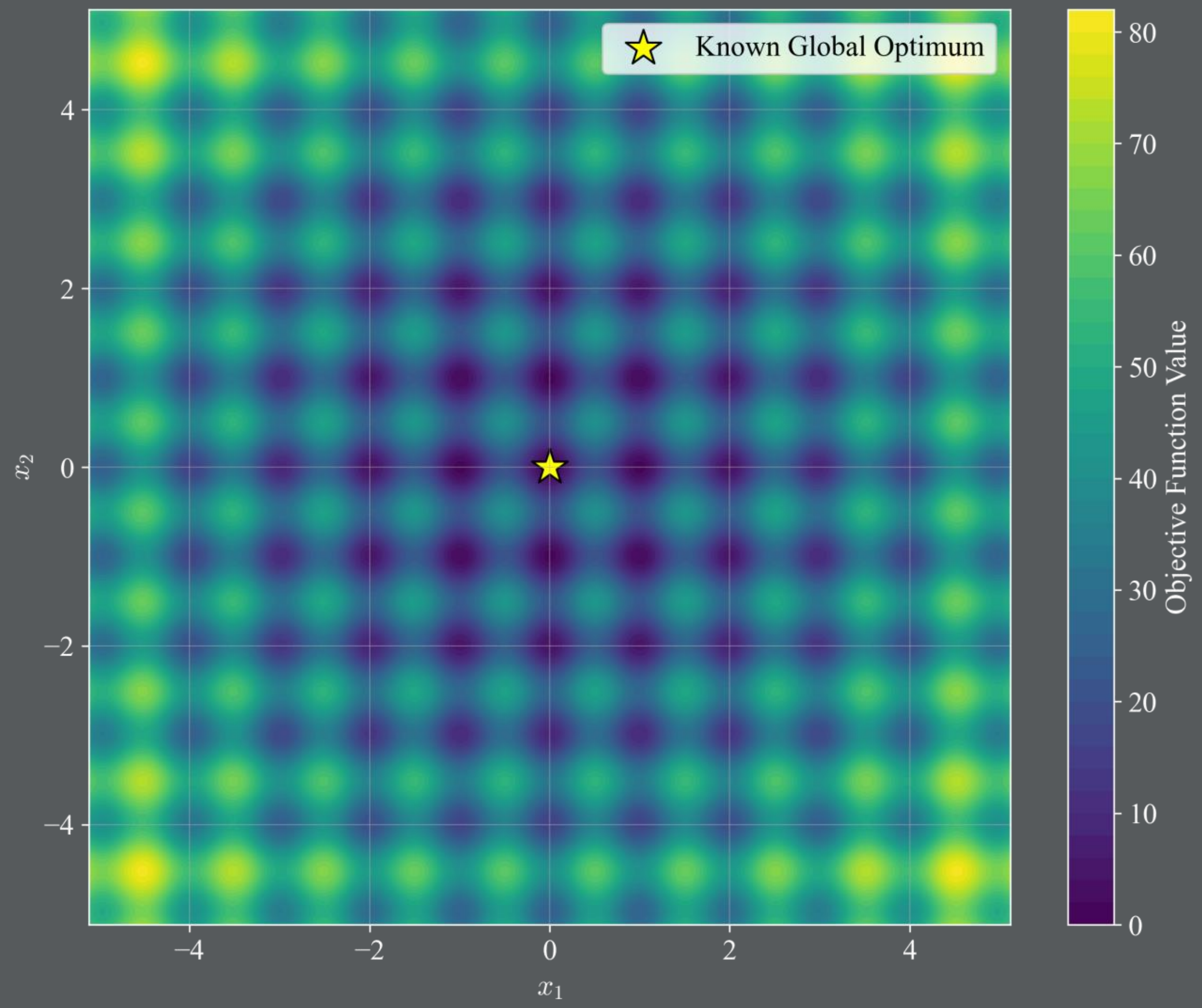


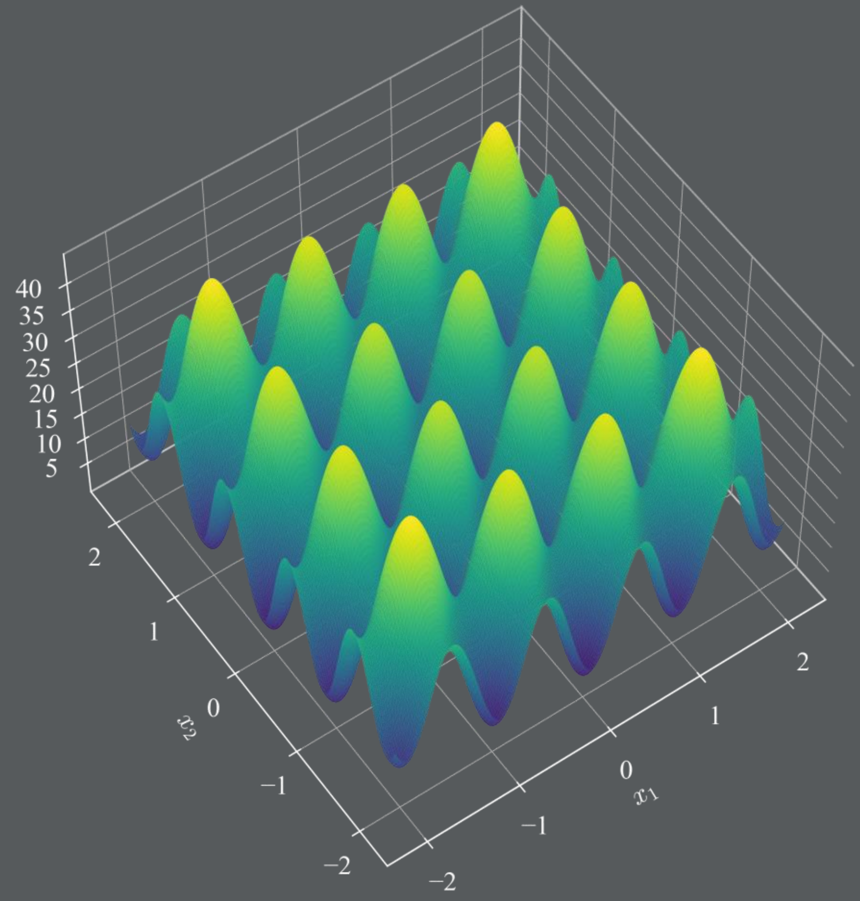


$$g(\boldsymbol{x}) = \sin(5x_1) - x_2 \geq 0$$

Initial guess: (100.0, 0.0)
Final value (exit after failure): 0.8300
Iterations: 27

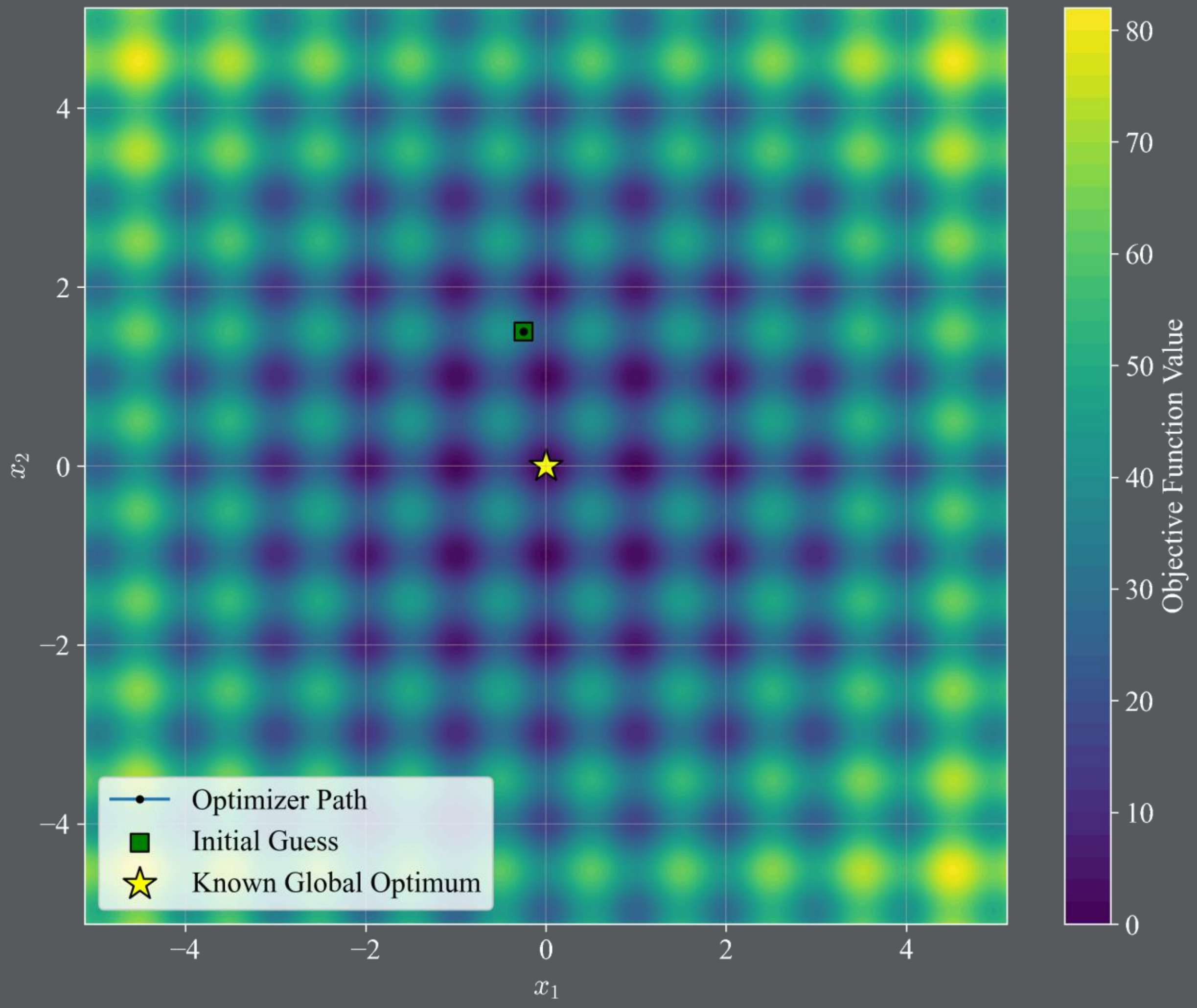
Rastrigin Function

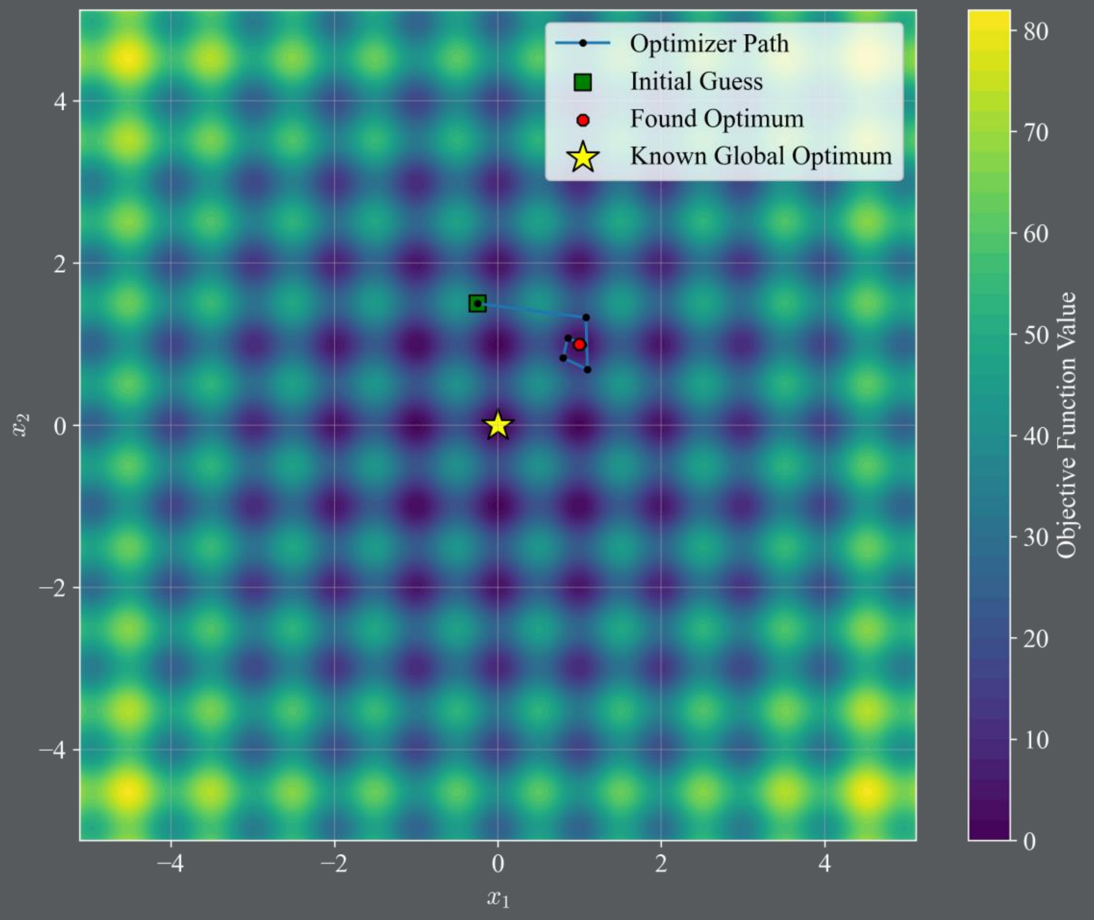




$$f(\boldsymbol{x}) = 20 + x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2)$$

Rastrigin Function Solution Path





$$f(\boldsymbol{x}) = 20 + x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2)$$

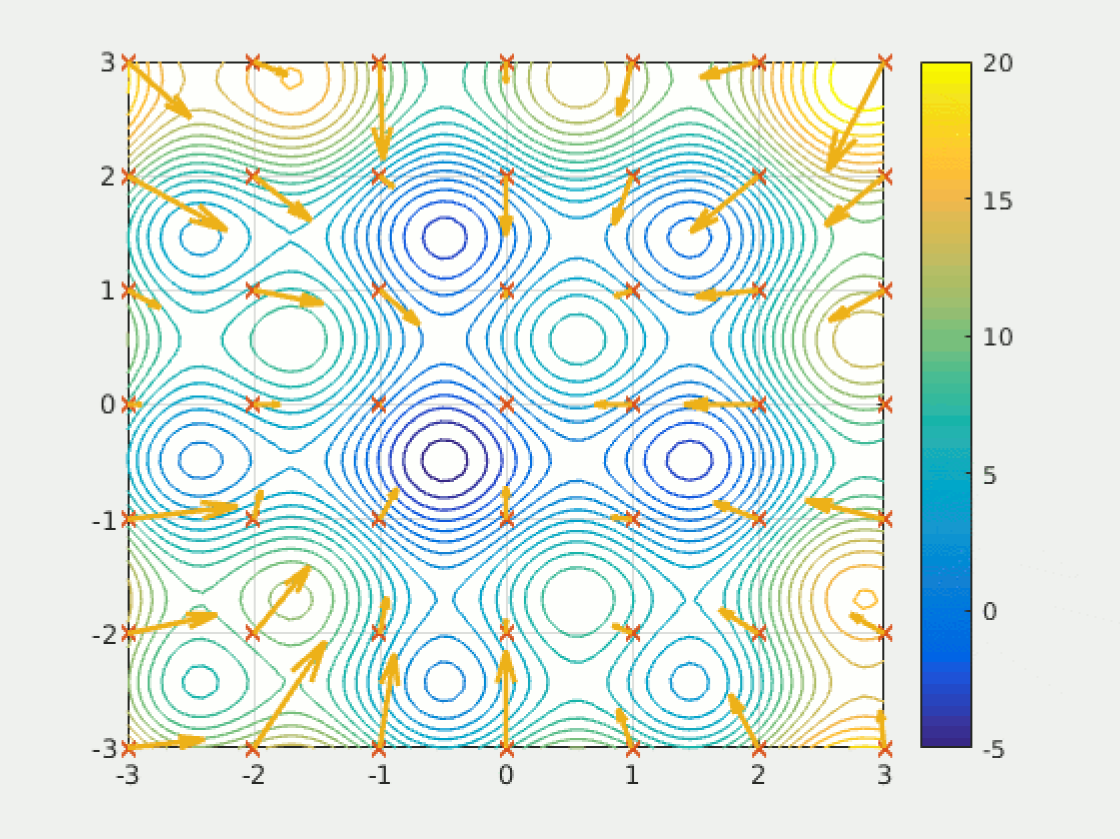
NLP Solution Key Takeaways

- Scaling of decision variables can drastically impact ability to find a good solution
- Gradient based methods tend to find local optima, may miss better solutions
- The more linear the objective and constraints are, the better
- Solving NLP problems is, in general, non-trivial!
  - Example problems shown here are fast to evaluate across the whole domain, easy to “grid search” – not true in general for NLP problems

Non-Gradient Based Methods

Particle Swarm Optimization

- Use a swarm of particles to find an optimal solution
- Each particle evaluates the objective function
- Motion of particles dictated by own best solution and global swarm best-known solution



Credit: Wikipedia user Ephramac
https://en.wikipedia.org/wiki/File:ParticleSwarmArrowsAnimation.gif

Genetic Algorithm

- Mimic natural selection "optimization" process
- A single "chromosome" represents a single solution to the optimization problem (can be binary or real valued)
- GA has a population of chromosomes
- Iterative process:
  - Evaluate "fitness" (objective function) of all chromosomes
  - Select chromosomes from which to make next generation (e.g., more fit genes are more likely to be selected)
  - Create next generation with "crossover" and "mutation" operations on selected chromosomes
- Lots of tuning parameters

Chrom-osome 1	Chrom-osome 2
0.75	0.288
0.043	0.368
0.848	0.434
0.260	0.706
0.359	0.601
0.660	0.942

Real valued chromosomes

Non-Gradient Based Method Key Takeaways

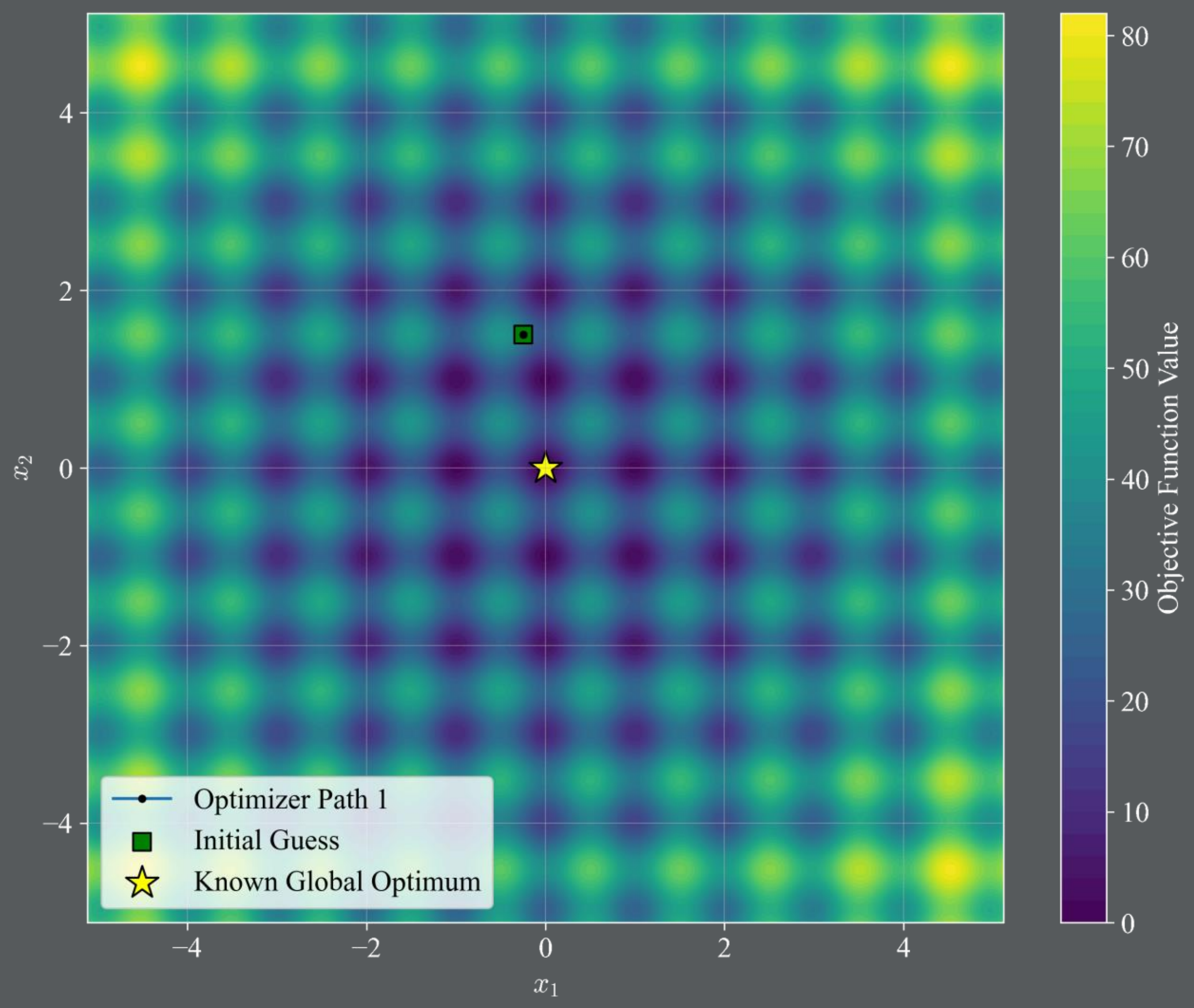
- Can handle non-smooth non-continuous problems
- Can handle discrete (integer) problems
- Frequently use random numbers to explore search space
- Can be computationally expensive if evaluating the objective function is expensive

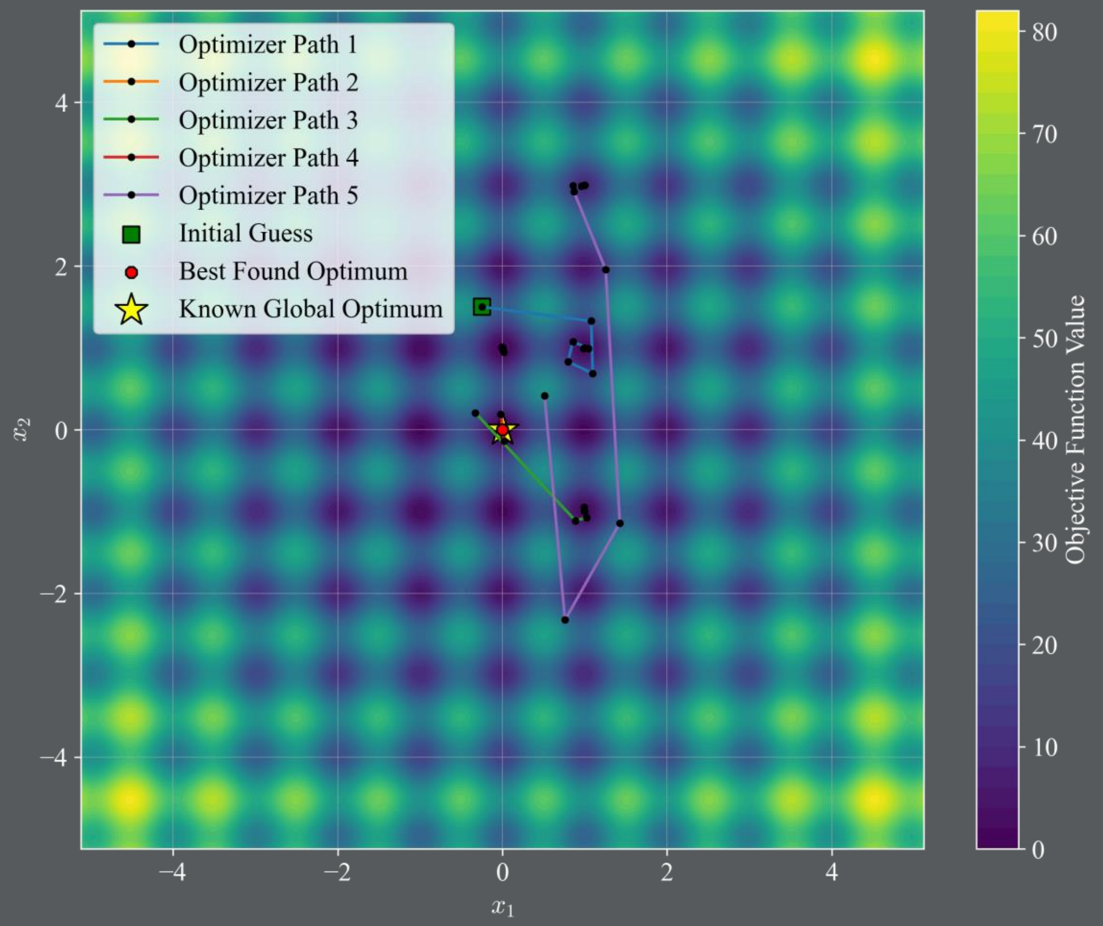
Hybrid Methods

Monotonic Basin Hopping

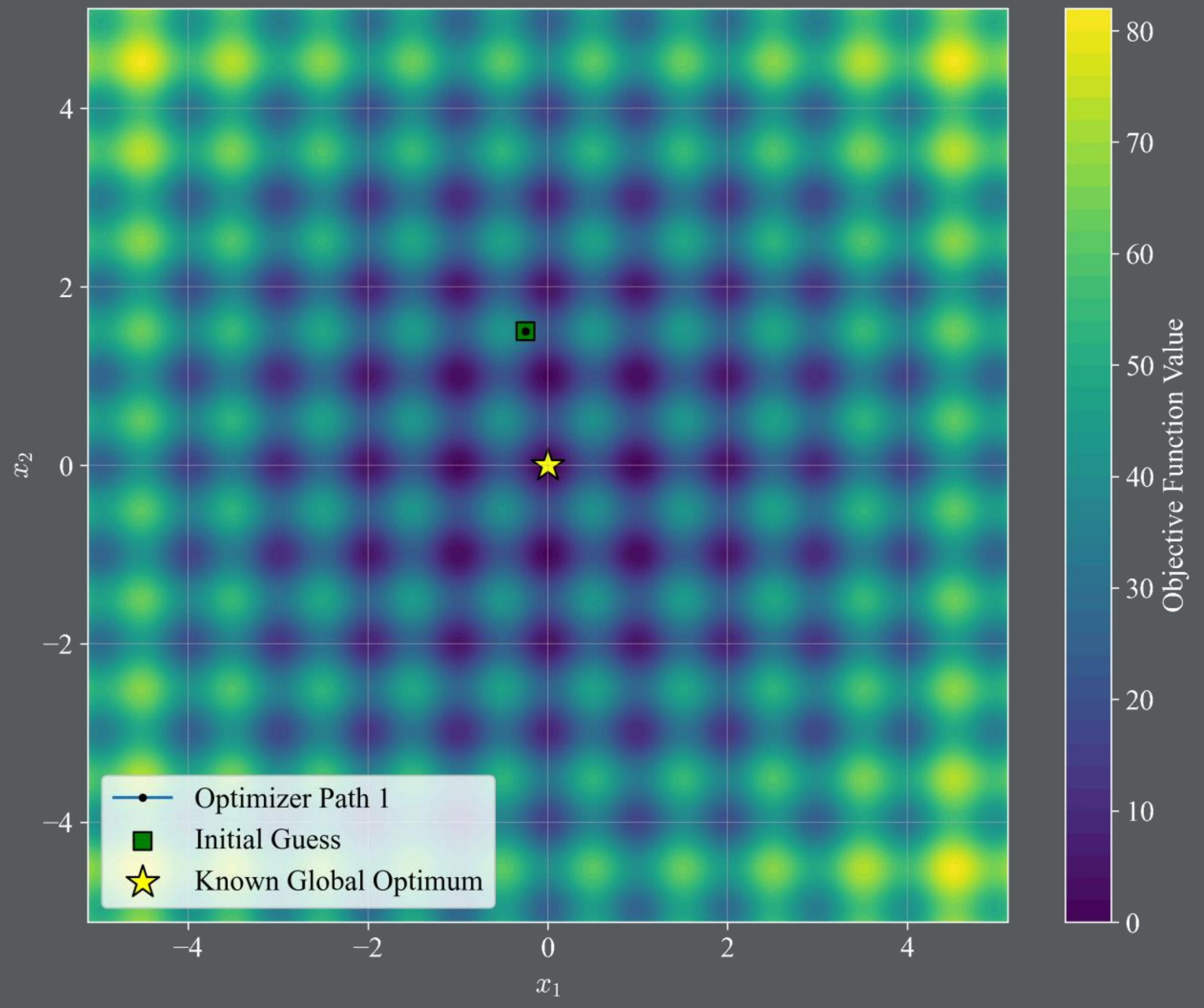
- “Poke” the solution around with random perturbations once you find a local minimum, then re-solve
  - Different random number distributions can be used (e.g. normal distribution, uniform distribution, pareto distribution etc.)
- Method of approximately finding a global optimum
  - True global optimal value is likely unknown
  - Likely exit condition: maximum run time or maximum number of hops
- Reduces reliance on good initial guess

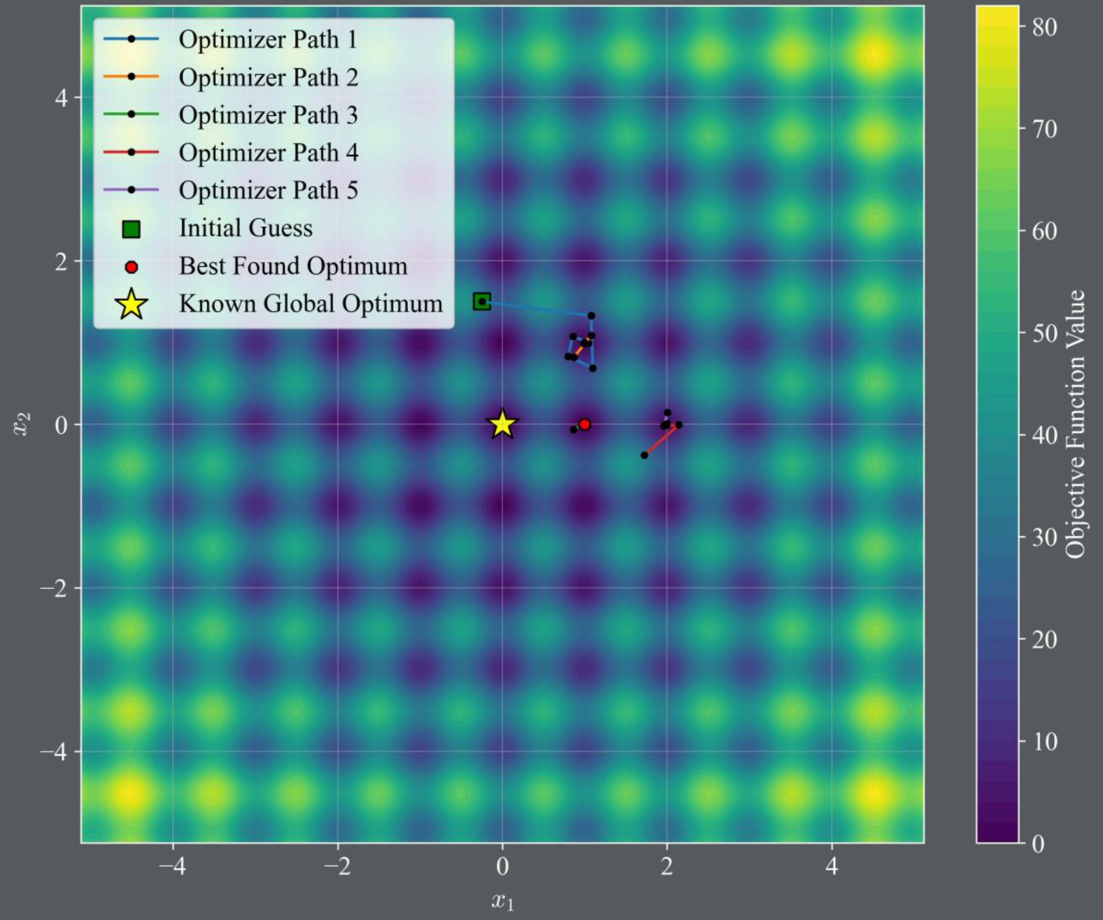
Monotonic Basin Hopping: Rastrigin





Monotonic Basin Hopping: Rastrigin





Other Hybrid Methods

- Basin hopping / simulated annealing + NLP gradient-based solve
  - C. H. Yam, D. D. Lorenzo, and D. Izzo, “Low-thrust trajectory design as a constrained global optimization problem” 2011 https://doi.org/10.1177/0954410011401686
- Genetic algorithm (outer loop) + NLP gradient-based solve (inner loop)
  - J. A. Englander and B. A. Conway, “Automated Solution of the Low-Thrust Interplanetary Trajectory Problem,” Journal of Guidance, Control, and Dynamics 2017 https://doi.org/10.2514/1.G002124
- Grid search + NLP gradient-based solve
  - E.g., grid over launch dates, minimize ΔV for each fixed launch date

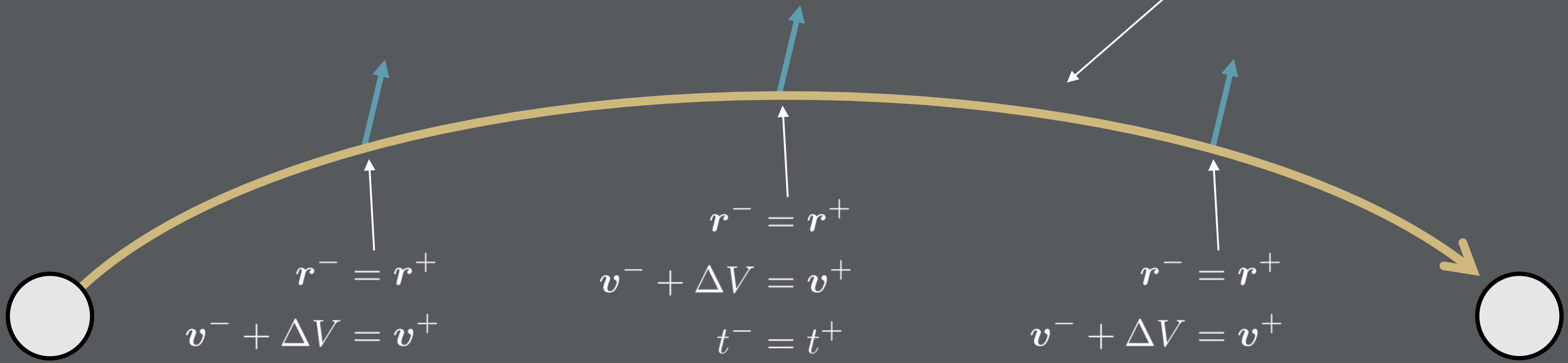
Optimal Control

Transcription Methods: Control

Non-exhaustive!

Impulsive

For two body dynamics, very low computational cost to propagate between impulses



$$\boldsymbol{r}^- = \boldsymbol{r}^+$$
$$\boldsymbol{v}^- + \Delta V = \boldsymbol{v}^+$$
$$t^- = t^+$$

$$\boldsymbol{r}^- = \boldsymbol{r}^+$$
$$\boldsymbol{v}^- + \Delta V = \boldsymbol{v}^+$$
$$t^- = t^+$$

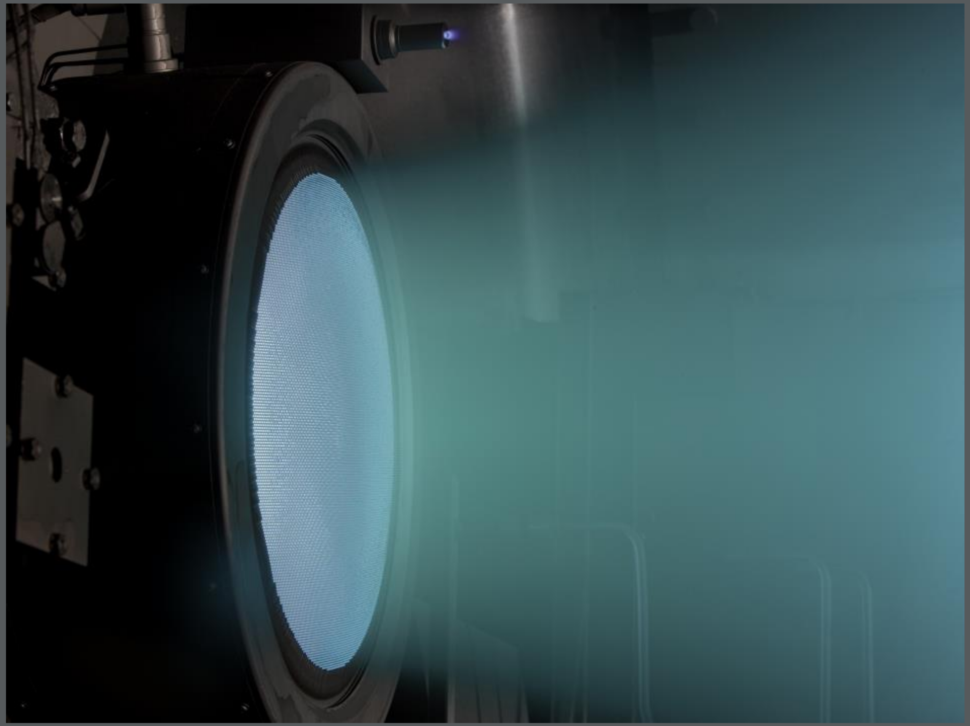
$$\boldsymbol{r}^- = \boldsymbol{r}^+$$
$$\boldsymbol{v}^- + \Delta V = \boldsymbol{v}^+$$
$$t^- = t^+$$

ΔV vector components are decision variables

Unbounded, impulsive ΔV generally used for preliminary chemical maneuver design

Low Thrust: Why Not Use The Standard Impulsive Assumption?

- Low thrust propulsion
  - Solar electric propulsion (SEP)
    - Accelerate ions out of engine at high speed to produce thrust
  - Solar sailing
    - Passive acceleration due to solar radiation pressure
- Highly efficient, but have very low magnitude of thrust
- Large, impulsive ΔV maneuvers are no longer a good model

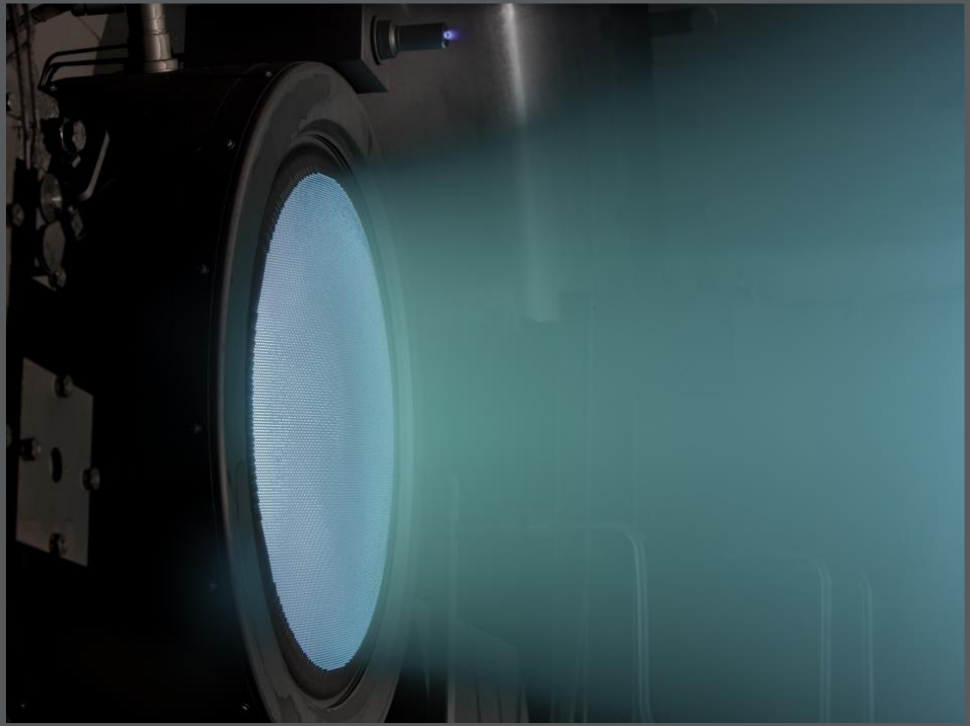


NEXT thruster during testing at NASA Glenn

Low Thrust: Why Not Use The Standard Impulsive Assumption?

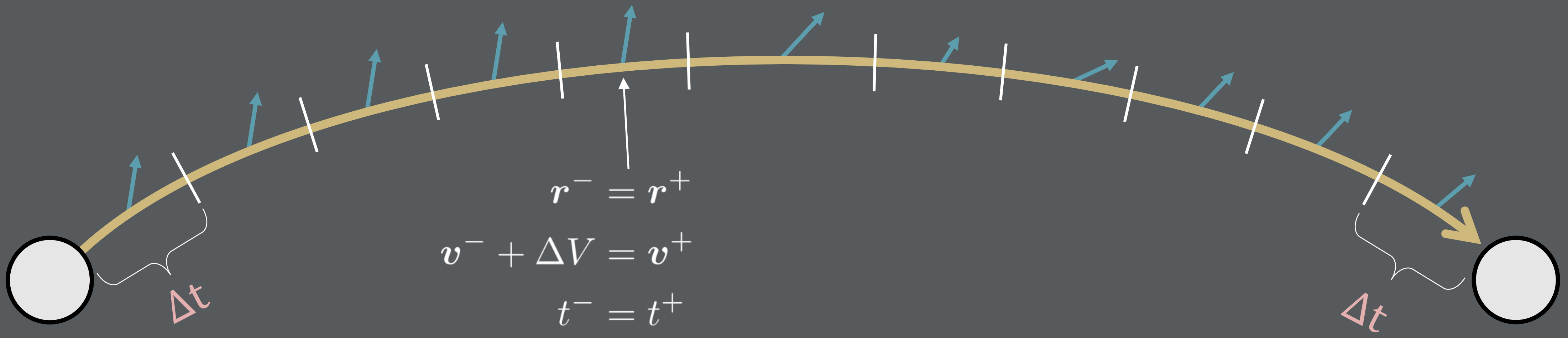
Engine	Max Thrust (N)	Efficiency / I_{sp} (sec.)
Space Shuttle Orbital Maneuvering (AJ10)	2,670	316
NASA Evolutionary Xenon Thruster (NEXT)	0.236	4,100

These two thrusters are designed for very different applications!



NEXT thruster during testing at NASA Glenn

Impulsive: Limited Magnitude

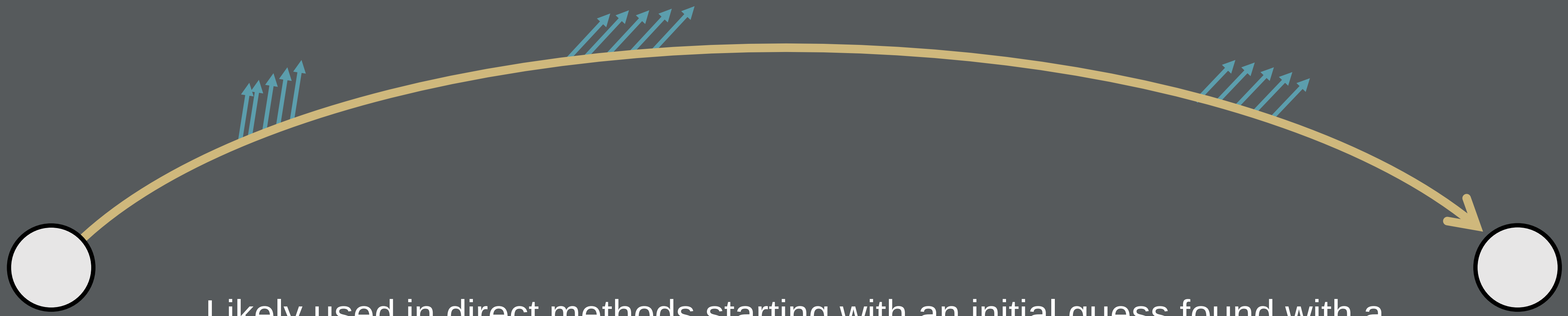


$$
\begin{aligned}
\boldsymbol{r}^- &= \boldsymbol{r}^+ \\
\boldsymbol{v}^- + \Delta V &= \boldsymbol{v}^+ \\
t^- &= t^+ \\
|\boldsymbol{\Delta V}| &< \Delta V_{\max}
\end{aligned}
$$

ΔV_{max} is an estimate of the maximum ΔV the low thrust engine can provide in the time Δt

Finite Burn

- Integrate equations of motion with acceleration matching what real thrusters can do
  - Slow speed of integration limits the number of iterations an optimizer can perform in a given amount of time



Likely used in direct methods starting with an initial guess found with a lower fidelity method
Indirect methods typically use finite burns as well

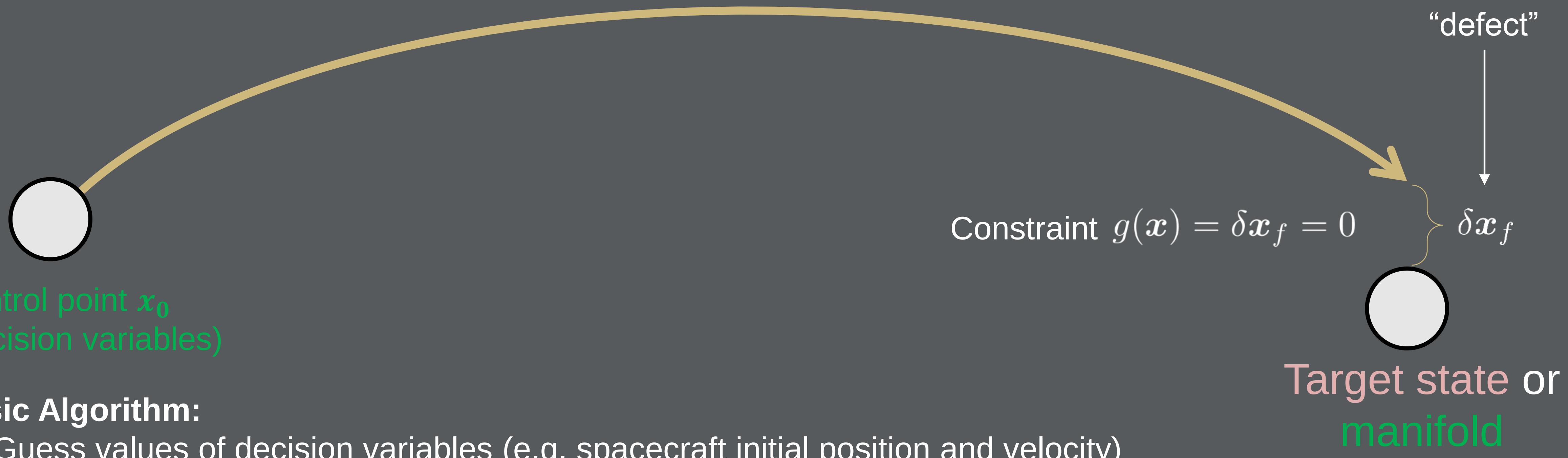
Control Law

- Parameterize control with a function
  - E.g. closed loop control law that selects thrusting direction and magnitude based on current state and time $\boldsymbol{u}(t) = \boldsymbol{f}(\boldsymbol{x})$
- Control law tuning parameters may be decision variables in an optimization problem
- Control law may handle difficult segment of larger control problem
  - E.g. spiral out from Earth SOI with control law, then interplanetary transfer elsewhere
- Common for low-thrust: Q-law

Transcription Methods: Dynamics

Methods of managing non-linearity between decision variables and the objective & constraints

Single Shooting



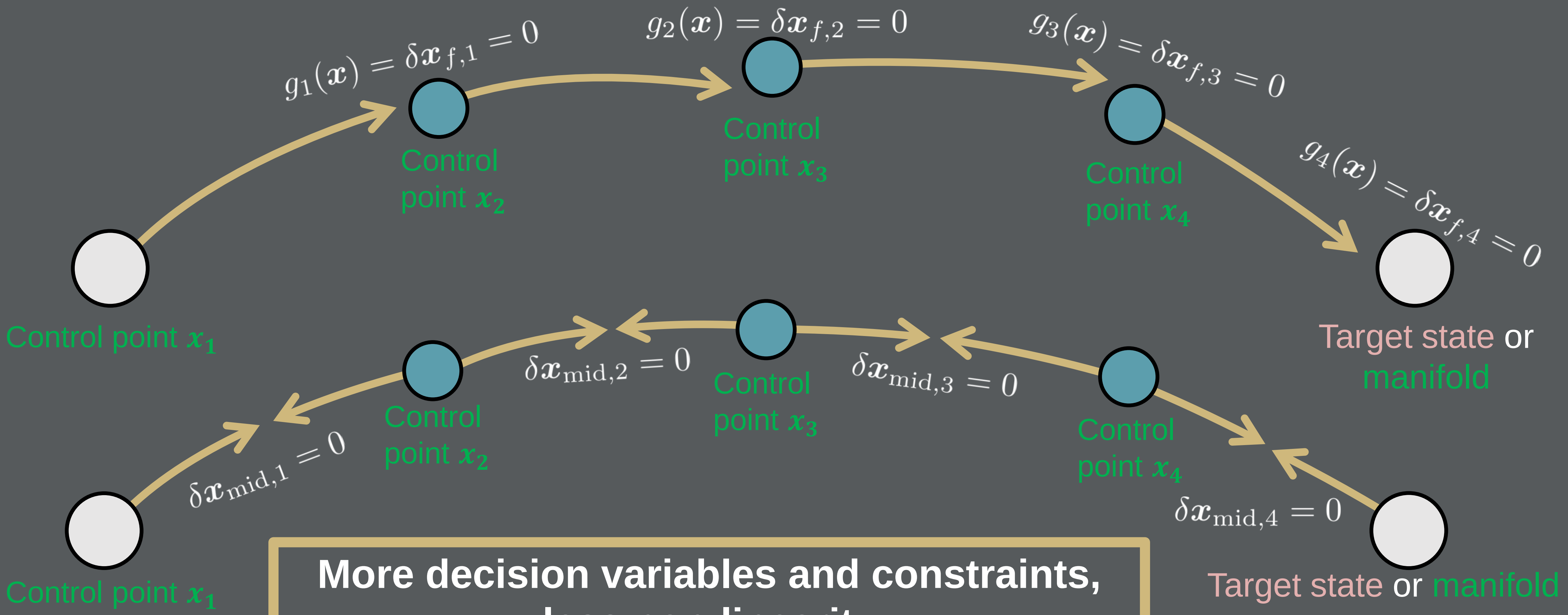
Basic Algorithm:

1. Guess values of decision variables (e.g. spacecraft initial position and velocity)
2. "Shoot:" propagate dynamics to the final time, calculate error in final state
3. Modify guess for decision variables based on "miss distance," repeat until error is zero
   - Modification likely uses a linear approximation of dynamics

$$\delta \boldsymbol{x}_f \approx \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_0}\right|_{\boldsymbol{x}=\boldsymbol{x}_0} \delta \boldsymbol{x}_0 \qquad \delta \boldsymbol{x}_0 = \left[\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_0}\right|_{\boldsymbol{x}=\boldsymbol{x}_0}\right]^{-1} \delta \boldsymbol{x}_f \qquad \boldsymbol{x}_{0,\text{new}} = \boldsymbol{x}_0 - \delta \boldsymbol{x}_0$$

Alternative: use NLP solver to select decision variables and enforce constraint, subject to some objective

Multiple Shooting

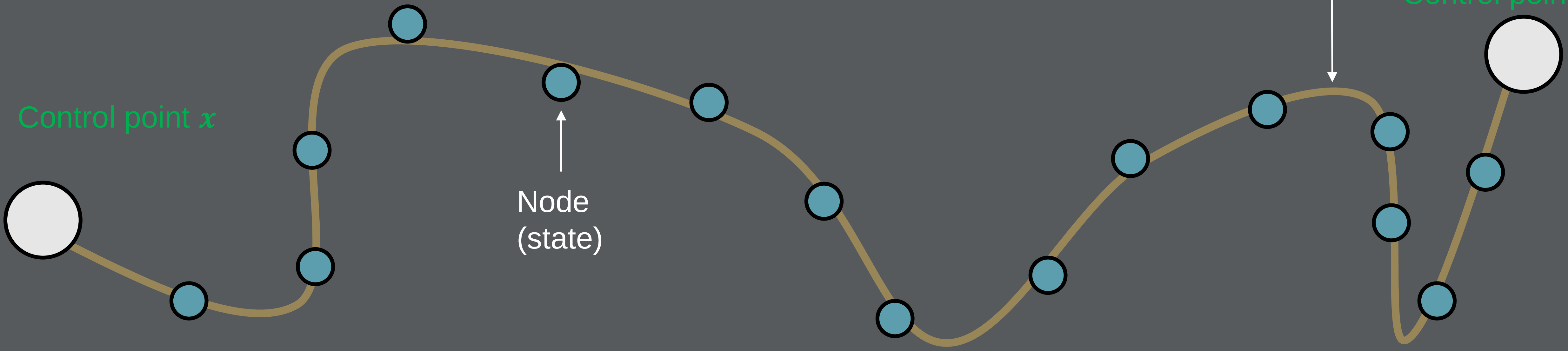


More decision variables and constraints, less non-linearity

Collocation

Single or multiple polynomials representing the problem dynamics

Decision variables: node states
Constraints: dynamics are followed at each point



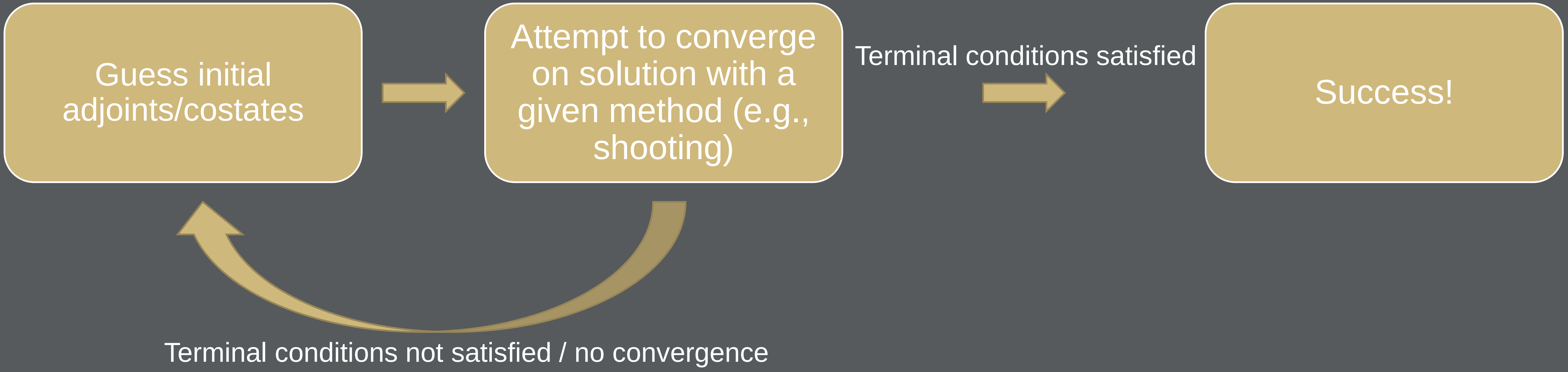
Even more decision variables and constraints, even less non-linearity

GPOPS, DIDO, GMAT (CSALT) use this method. “Pseudospectral” methods are a type of collocation

Indirect

Solution Process

- Indirect method transforms optimization problem into two-point or multi-point boundary value problem (TPBVP / MPBVP)



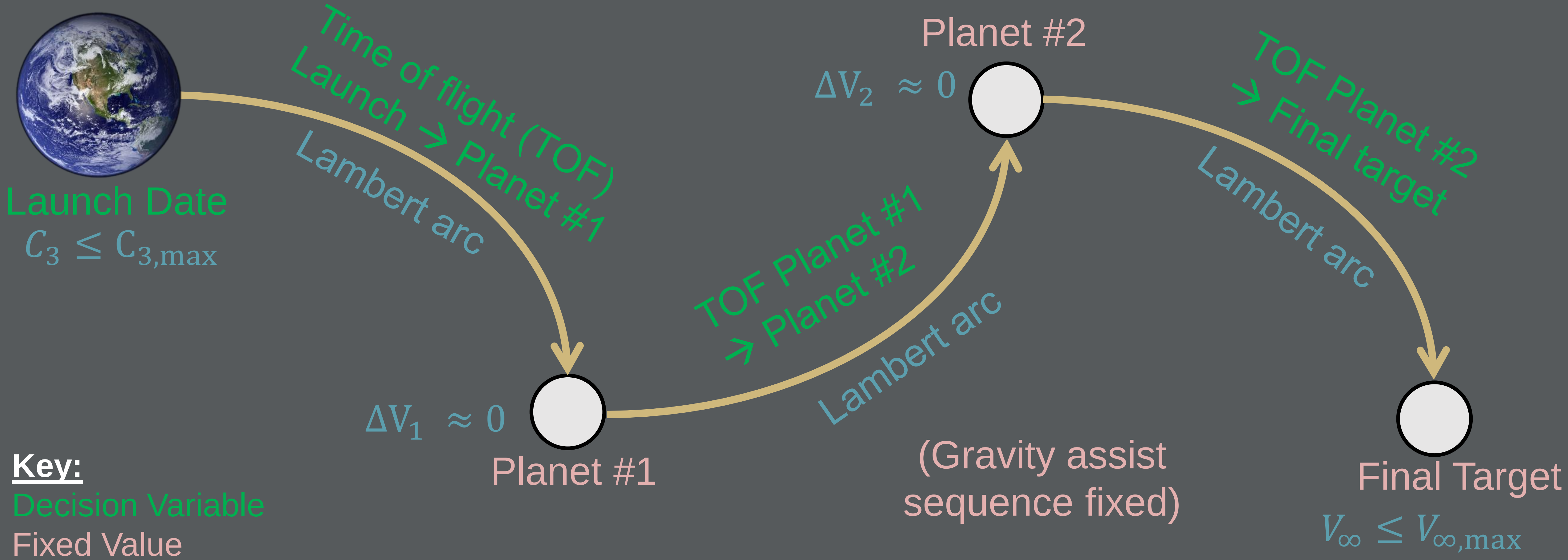
Indirect Solution Tips

- Orbit-element state representations drastically reduce the sensitivity of problem w.r.t. initial guess of costates
  - J. L. Junkins and E. Taheri, "Exploration of Alternative State Vector Choices for Low-Thrust Trajectory Optimization," Journal of Guidance, Control, and Dynamics 2019 https://doi.org/10.2514/1.G003686
- Smooth transition between zero thrust and maximum thrust
  - Start with smooth transition, use continuation to progressively make the transition steeper
  - E. Taheri and J. L. Junkins, "Generic Smoothing for Optimal Bang-Off-Bang Spacecraft Maneuvers," Journal of Guidance, Control, and Dynamics 2018 https://doi.org/10.2514/1.G003604
- Multiple shooting or collocation can further reduce sensitivity

Sample Direct Transcriptions for Interplanetary Problems

Putting it all together

Sample Ballistic Interplanetary Transcription



Key:

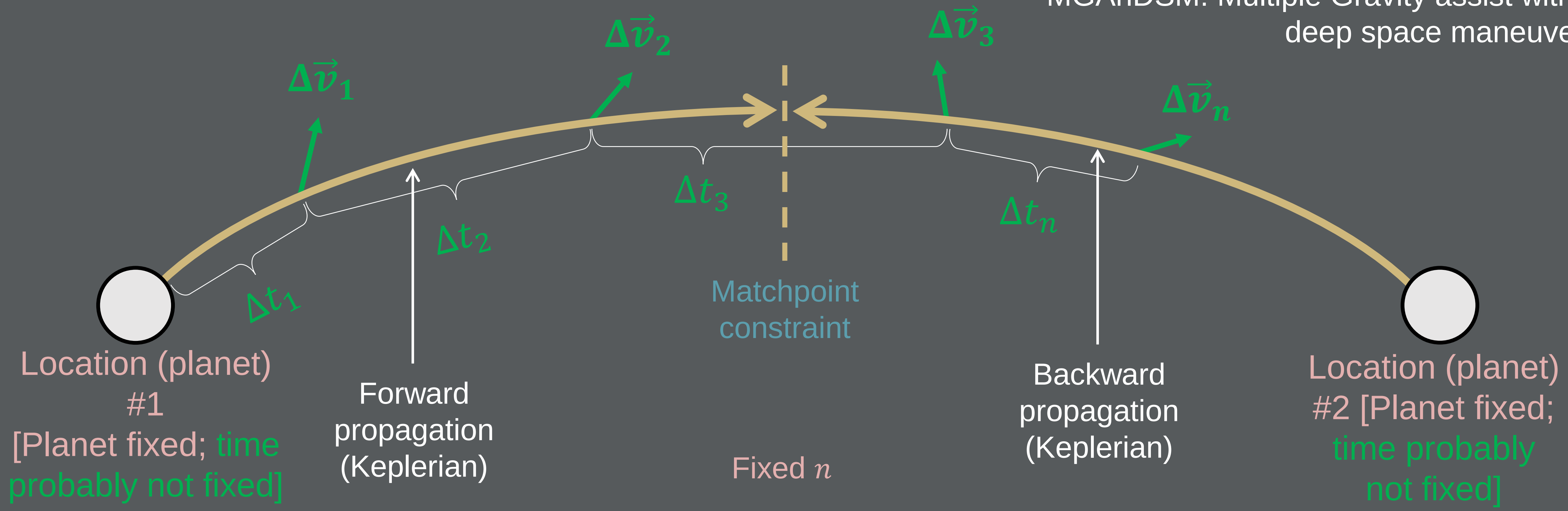
Decision Variable

Fixed Value

Calculated Value (based on decision & fixed values)

MGAnDSM Transcription: Chemical

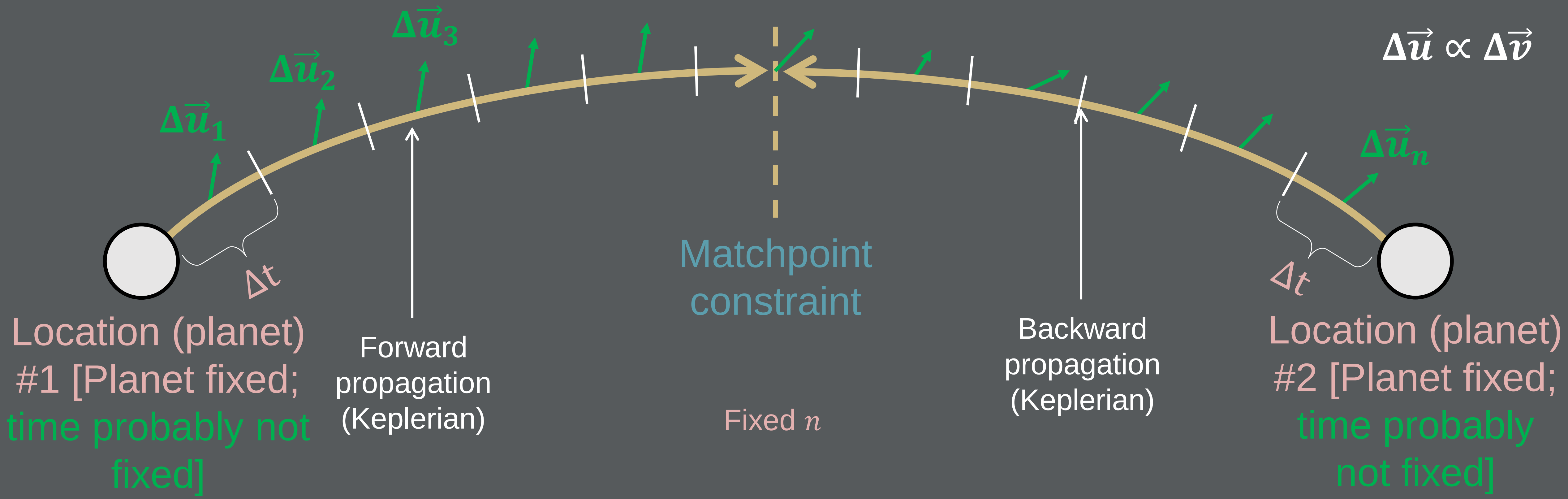
MGAnDSM: Multiple Gravity assist with n deep space maneuvers



Key:
Decision Variable
Fixed Value
Calculated Value (based on decision & fixed values)

Donald H. Ellison, et. al. "Analytic Gradient Computation for Bounded-Impulse Trajectory Models Using Two-Sided Shooting," *Journal of Guidance, Control, and Dynamics* 2018 https://doi.org/10.2514/1.G003077

Sims-Flanagan Transcription: Low-Thrust



Key:

Decision Variable

Fixed Value

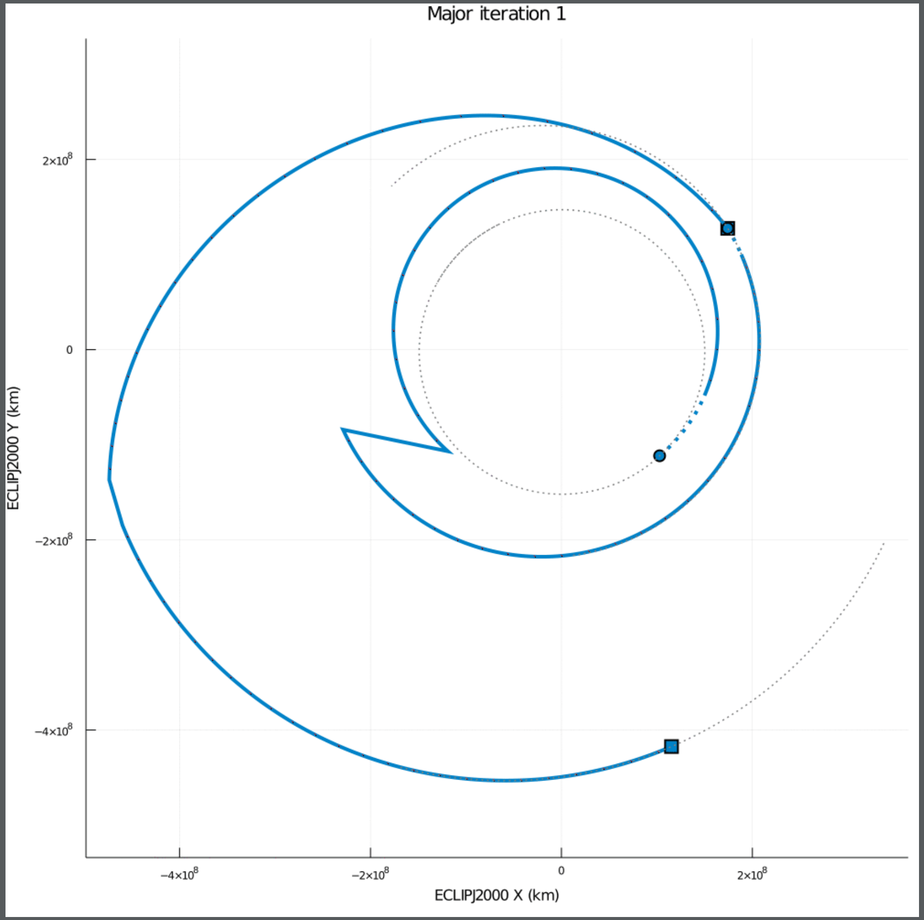
Calculated Value (based on decision & fixed values)

Sims, J. A., and Flanagan, S. N., “Preliminary Design of Low-Thrust Interplanetary Missions (99-338),” Advances in the Astronautical Sciences, Vol. 103, Univelt Inc., Escondido, CA, 1999, pp. 538–548.
Ellision 2018 paper in previous slide has nice additional detail about the problem

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche

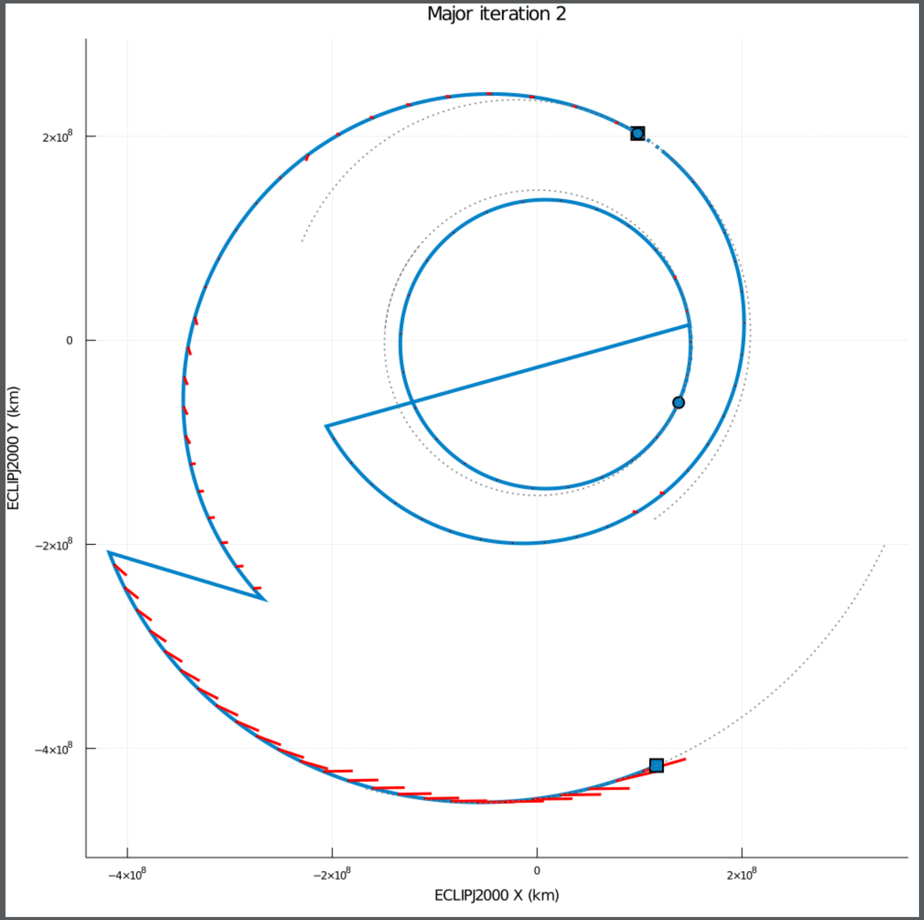
Using the NSTOP tool I developed and a Sims-Flanagan transcription, find a mass optimal trajectory to Psyche with a Mars gravity assist

NLP Solver: SNOPT



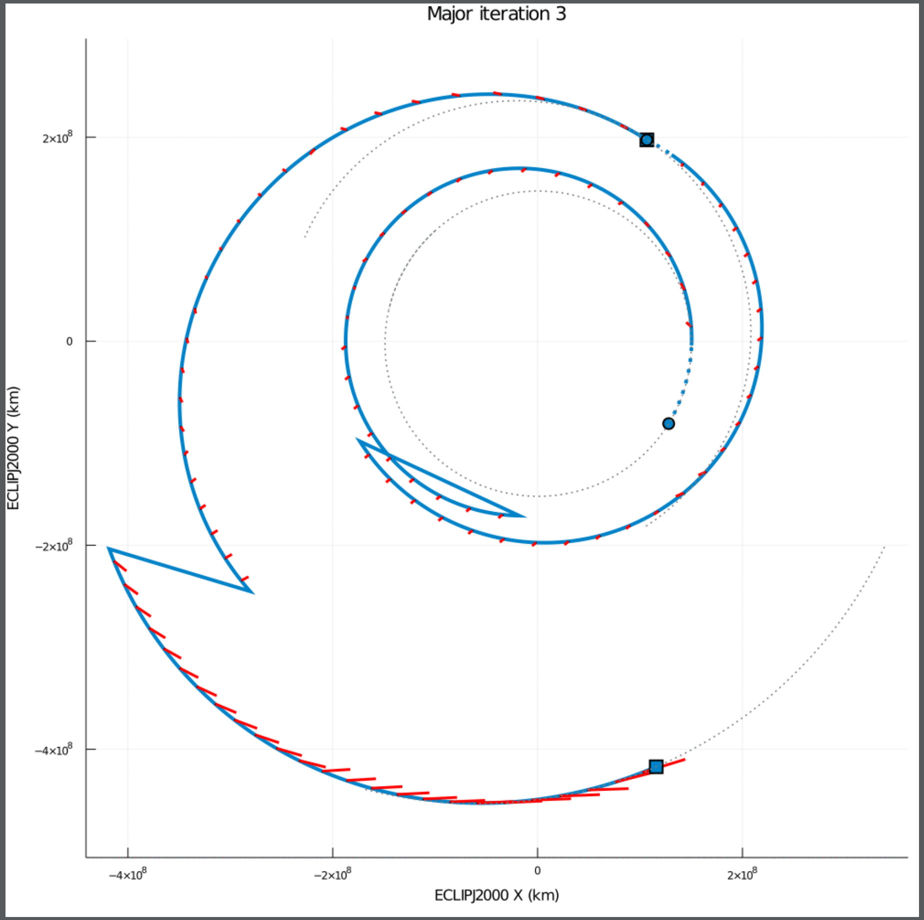
Bad initial guess: no control applied (though initial launch/flyby/arrival dates are not too far from the local optimum)

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



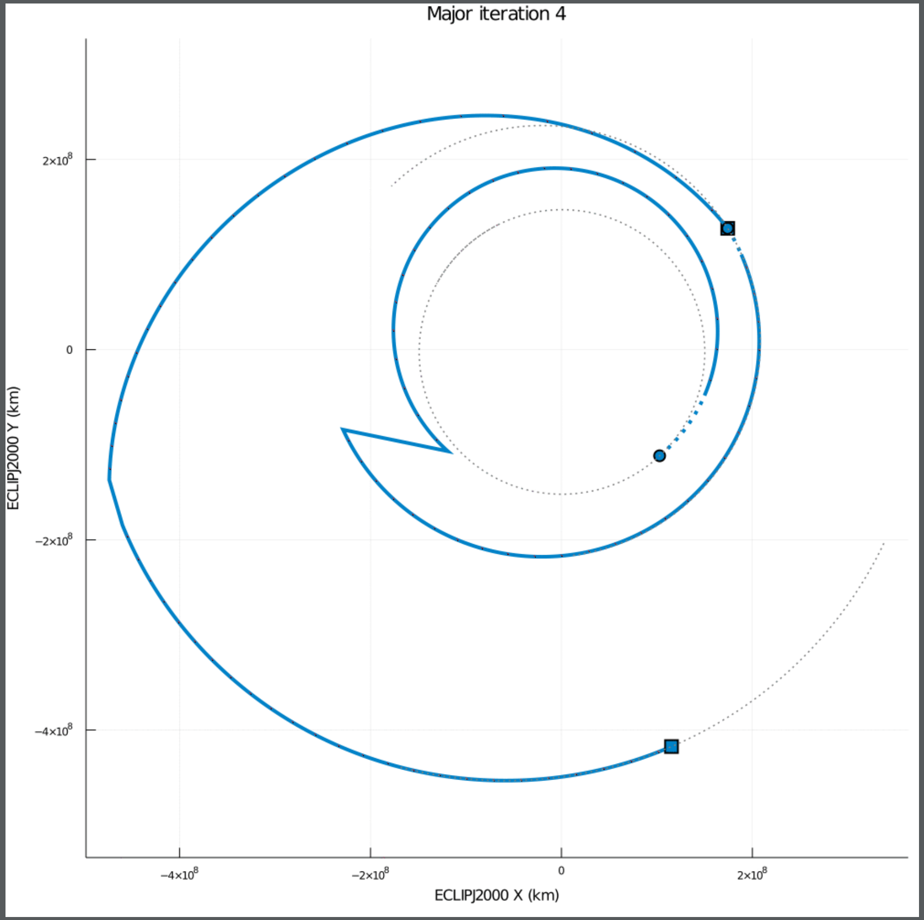
Iteration 2

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



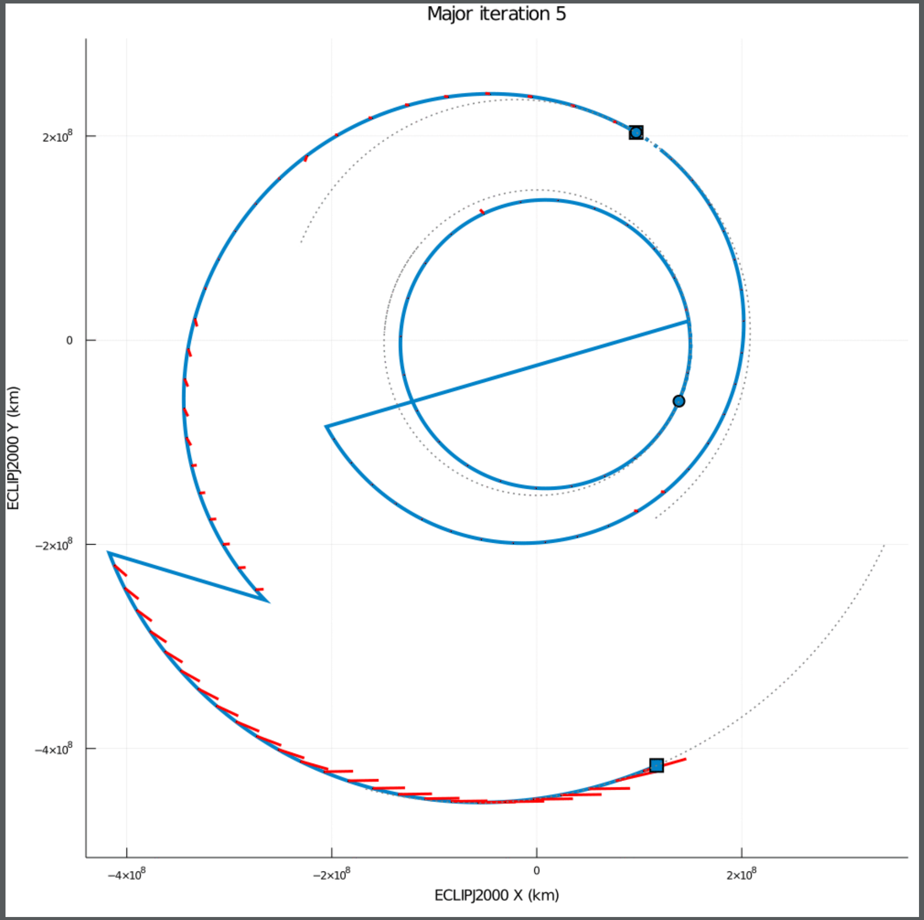
Iteration 3

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



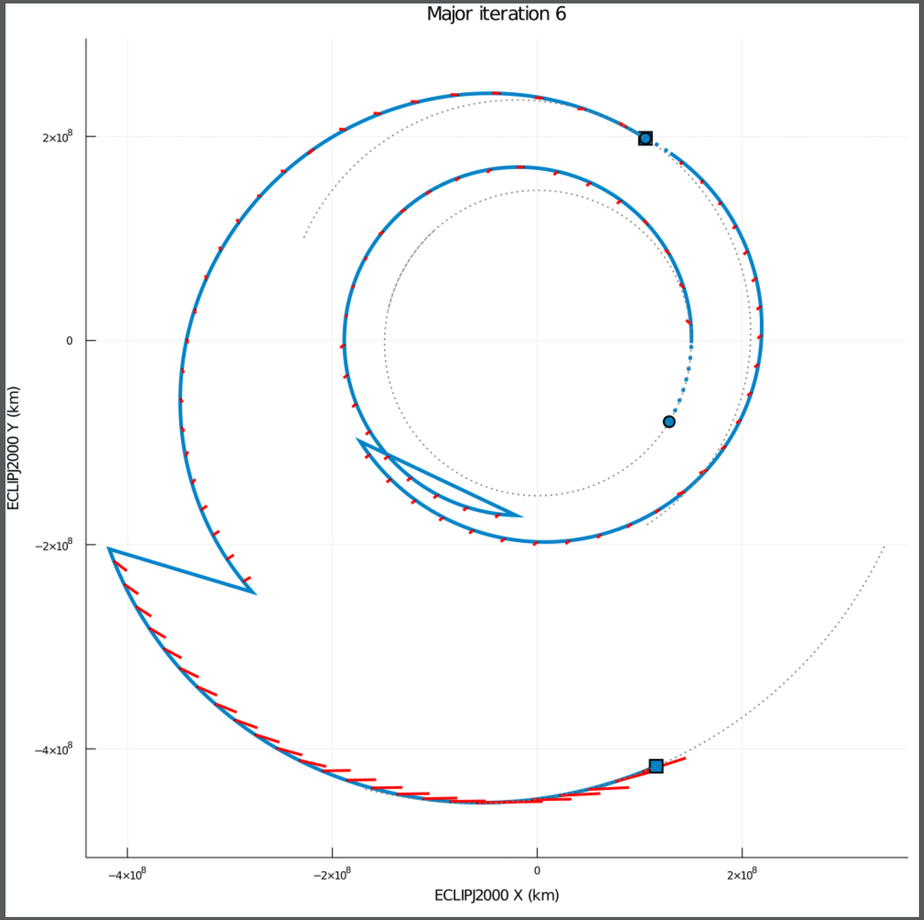
Iteration 4

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



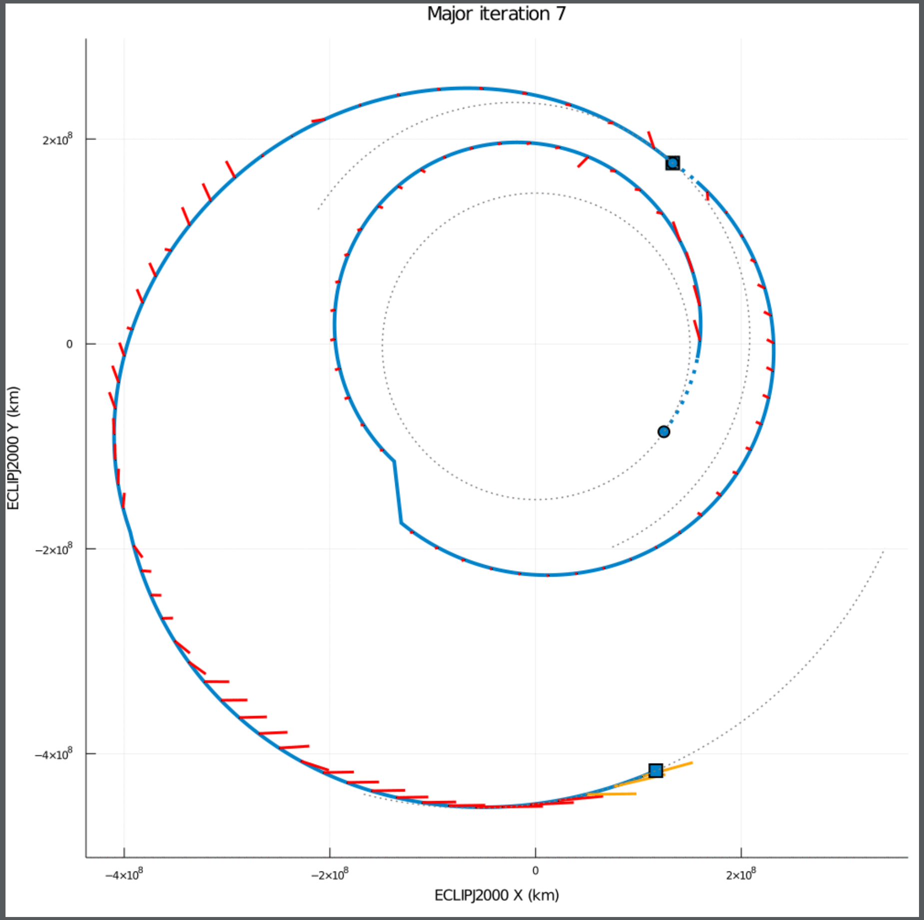
Iteration 5

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



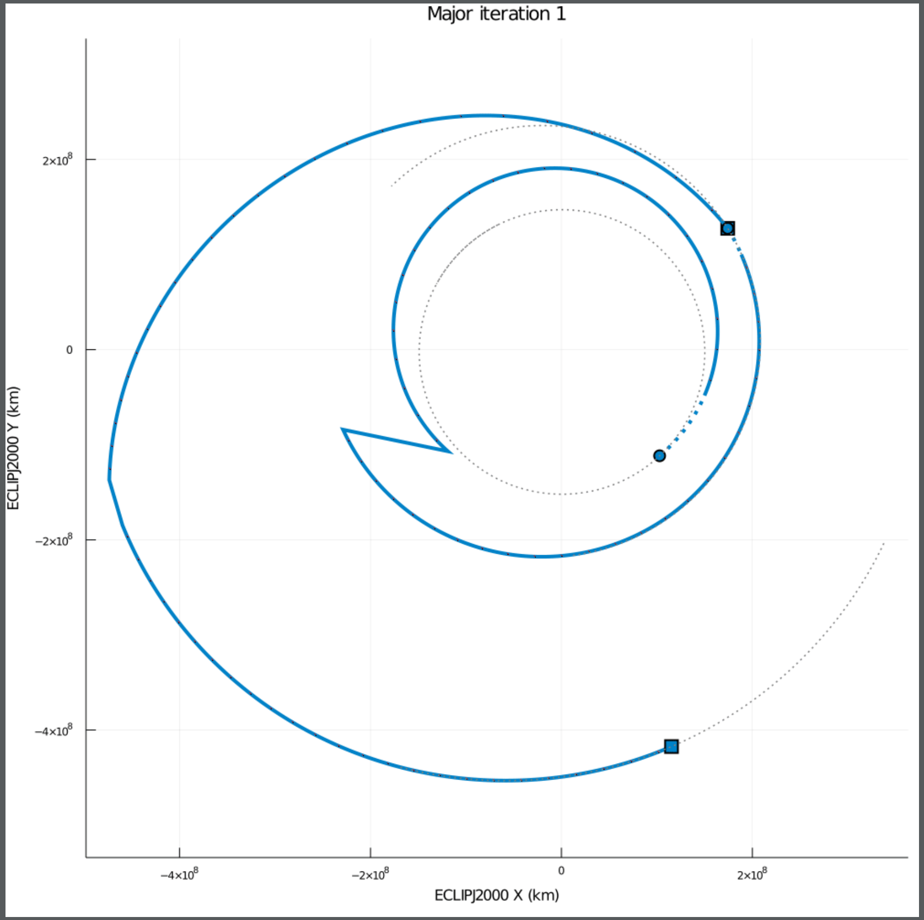
Iteration 6

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



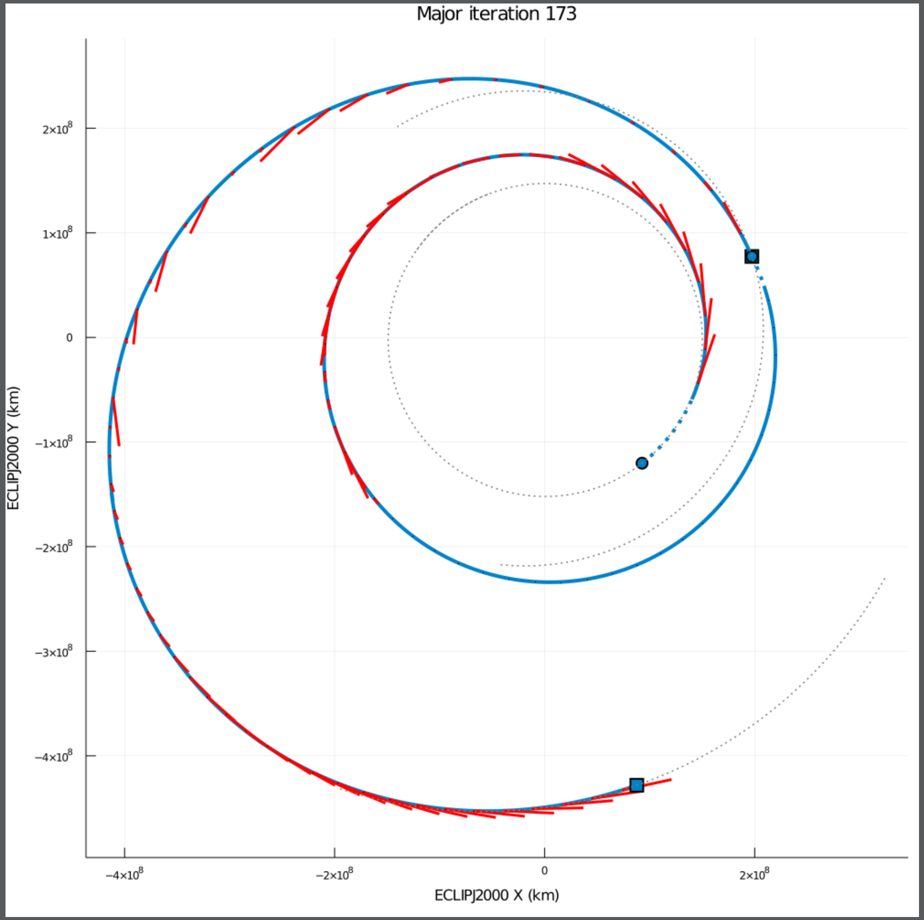
Iteration 7

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



Earth-Mars-Psyche trajectory

Sims-Flanagan Solution Convergence: Earth → Mars → Psyche



Final iteration

Recap

- Reviewed several MPP solution methods
  - Gradient-based NLP solvers most common for SC trajectory optimization problems (with direct methods)
  - Alternative methods can be used to deal with non-smooth & discrete problems
- Discussed indirect and direct methods for optimal control
  - Direct: more robust to poor initial guesses
  - Indirect: more compact representation of solution
- Discussed transcription methods
  - Dynamics transcriptions can be used for both direct and indirect methods
- Showed example trajectory transcriptions and convergence process

Implementing Trajectory Optimization: NSTOP

Created for a research project on robust optimization

Implementation: N Spacecraft Trajectory Optimizer (NSTOP)

- NSTOP written in the Julia programming language: ~7,000+ lines of code
- Can pose (but not necessarily solve!) problems with arbitrary number of:
  - Spacecraft
  - Phases (i.e. gravity assists)
  - Missed thrust events
- Primary function: transform problem into NLP for SNOPT to solve (+ augment with basin hopping, seed sharing)

NLP Solver Interface

- N decision variables $\boldsymbol{x}$
- m constraints $\boldsymbol{g}$

$$\min_{x \in \mathbb{R}^N} f(\boldsymbol{x})$$

$$\boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U$$

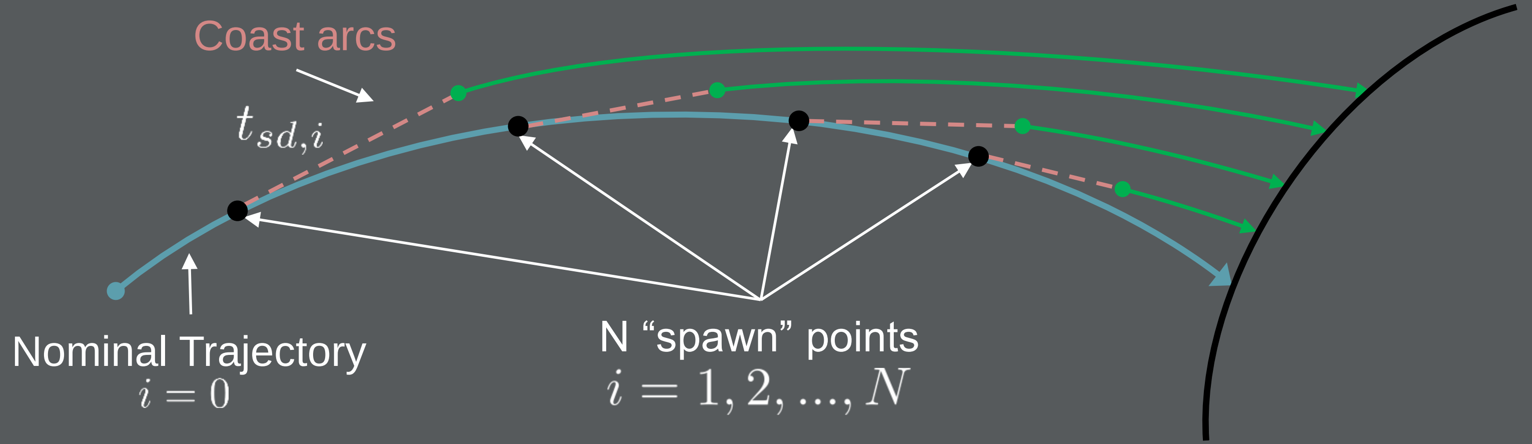
$$\boldsymbol{g}^L \leq \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{g}^U$$

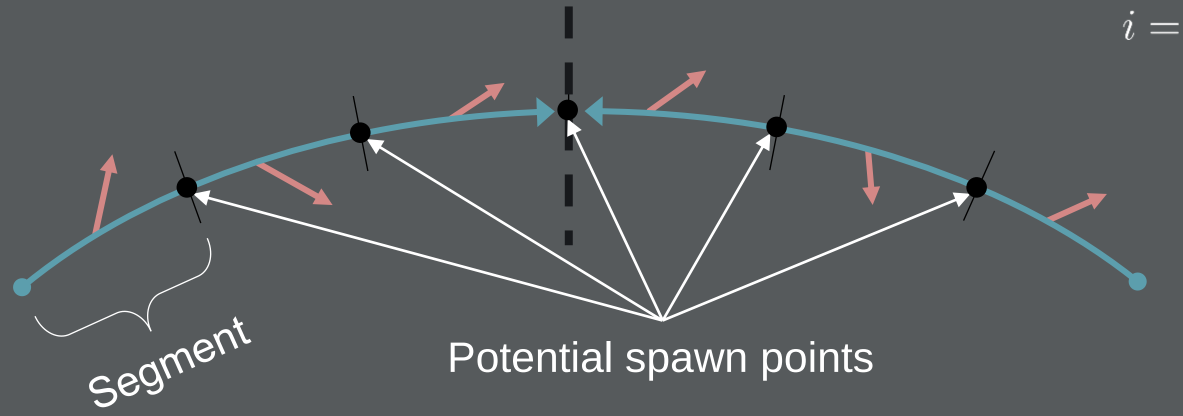
$$\frac{\partial f}{\partial \boldsymbol{x}}$$ N-dimensional vector

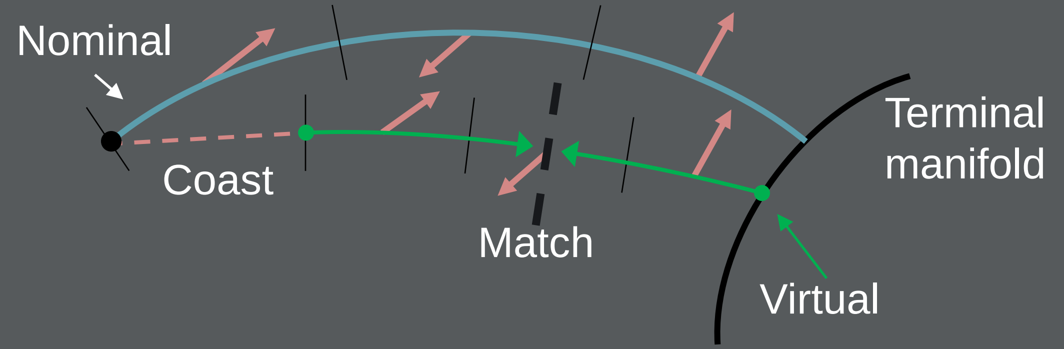
$$\frac{\partial g}{\partial \boldsymbol{x}}$$ $m \times N$ Jacobian matrix

Tip: Whiteboard Your Problem Before Implementing!

- Early whiteboard drawings for the robustness problem looked roughly like these Powerpoint diagrams
- Explicitly list all
  - Decision variables
  - Constraints







NLP Solver Interface

- Must provide to the NLP solver:
  - Bounds on decision variables ("box" constraints) $\boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U$
    - E.g. if x_1 is launch date, x_1^L might be the earliest allowable launch date and x_1^U might be the latest allowable launch date
  - Bounds on constraint values $\boldsymbol{g}^L \leq \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{g}^U$
    - E.g. if $g_1 = \Delta V_{1x}^2 + \Delta V_{1y}^2 + \Delta V_{1z}^2$, may have $g_1^L = 0$ and $g_1^U = \Delta V_{1,max}^2$
  - Scalar function $f(\boldsymbol{x})$ to map decision variables to objective value
  - Function $\boldsymbol{g}(\boldsymbol{x})$ to map decision variables to constraint values
  - Function to compute $\frac{\partial f}{\partial \boldsymbol{x}}$ $(N \times 1)$
  - Function to compute $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$ $(m \times N)$

May be automatically computed with finite differencing

Note on Sparsity

- The constraint Jacobian $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$ $(m \times N)$ can grow quite large
  - Large size consumes lots of computer memory, slows linear algebra operations
- Jacobian structure is frequently “sparse” – has many zeros

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 11 & 12 & 13 \end{bmatrix}$$

Note on Sparsity

- Instead of holding all values in memory, use alternative representation

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 11 & 12 & 13 \end{bmatrix}$$

5x8 Matrix (2D array)

Row Vector	Column Vector	Value Vector
1	1	1
1	2	2
1	3	3
2	1	4
2	2	5
2	3	6
3	4	7
4	5	8
…	…	…

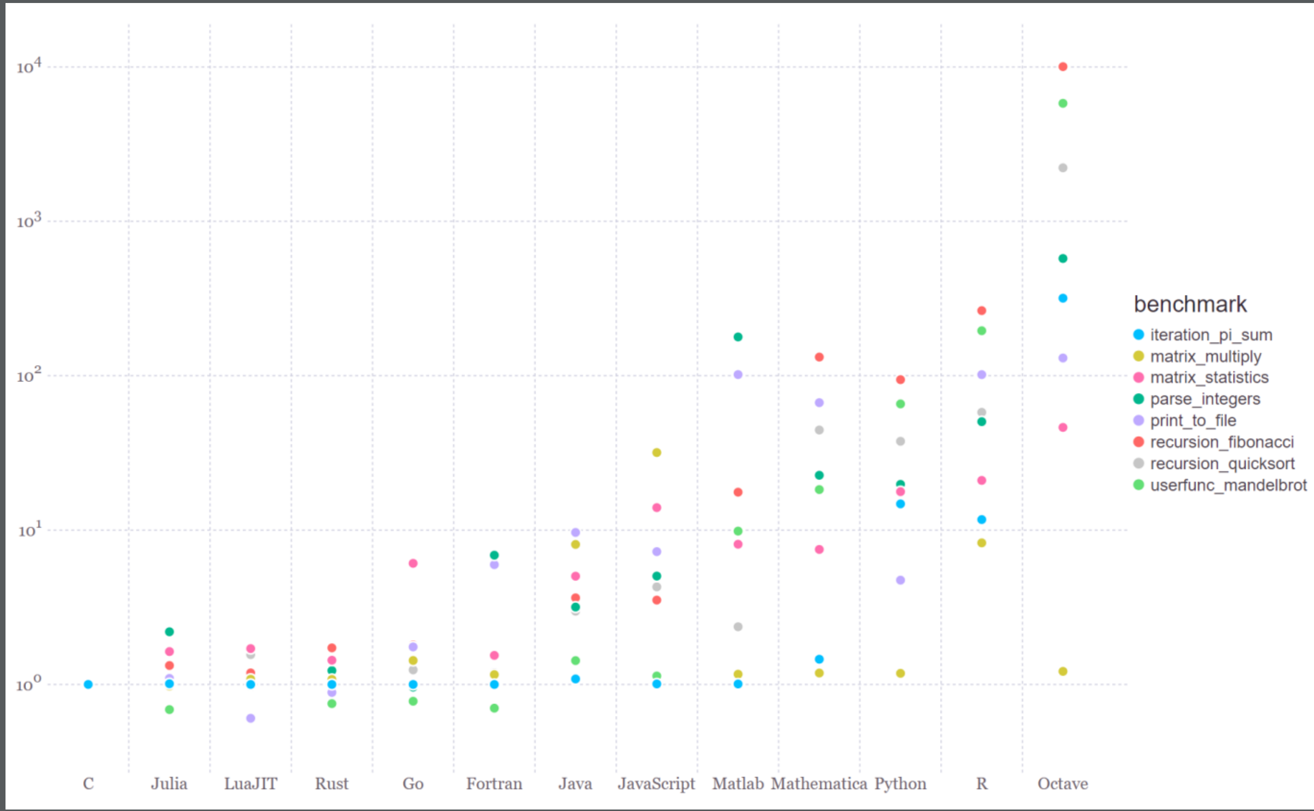
Automatic Differentiation

- Generates partial derivatives of functions accurate to machine-precision
  - Not the same as symbolic differentiation through Mathematica/SymPy/etc.
- Good implementation usually computes derivates faster than finite differencing
- Tools to do this are widely available
  - New in Matlab 2020b! (Linked blog post with additional information)
  - Packages exist for Julia, Python, C++, etc.
  - NSTOP uses ForwardDiff.jl in the Julia programming language

https://blogs.mathworks.com/loren/2020/10/06/automatic-differentiation-in-optimization-toolbox/, https://github.com/JuliaDiff/ForwardDiff.jl

Julia Programming Language

- High-level, high-performance computing language
- Built to solve the “two-language” problem
- Lots of scientific computing packages and users in the ecosystem
- https://julialang.org/learning/



https://julialang.org/benchmarks/

Alternative Partial Derivative Methods

- Analytic
  - Hand-derived or computer aided (e.g. Mathematica, SymPy, etc.)
  - Best computation speed
  - Can be difficult to calculate or book-keep
  - Evolutionary Mission Trajectory Generator (EMTG) takes this approach
- Finite Difference
  - Can tune this for specific problems
  - Highly flexible
  - Potentially inaccurate, slow
  - Copernicus SC trajectory optimization tool provides this capability

Ephemeris Data

- NSTOP uses planetary ephemeris data provided by JPL through “SPICE” kernels
- Not ideal to call SPICE within optimization loop
- Common technique (lower-fidelity stage): fit ephemeris data to smooth spline functions
  - Query states from spline

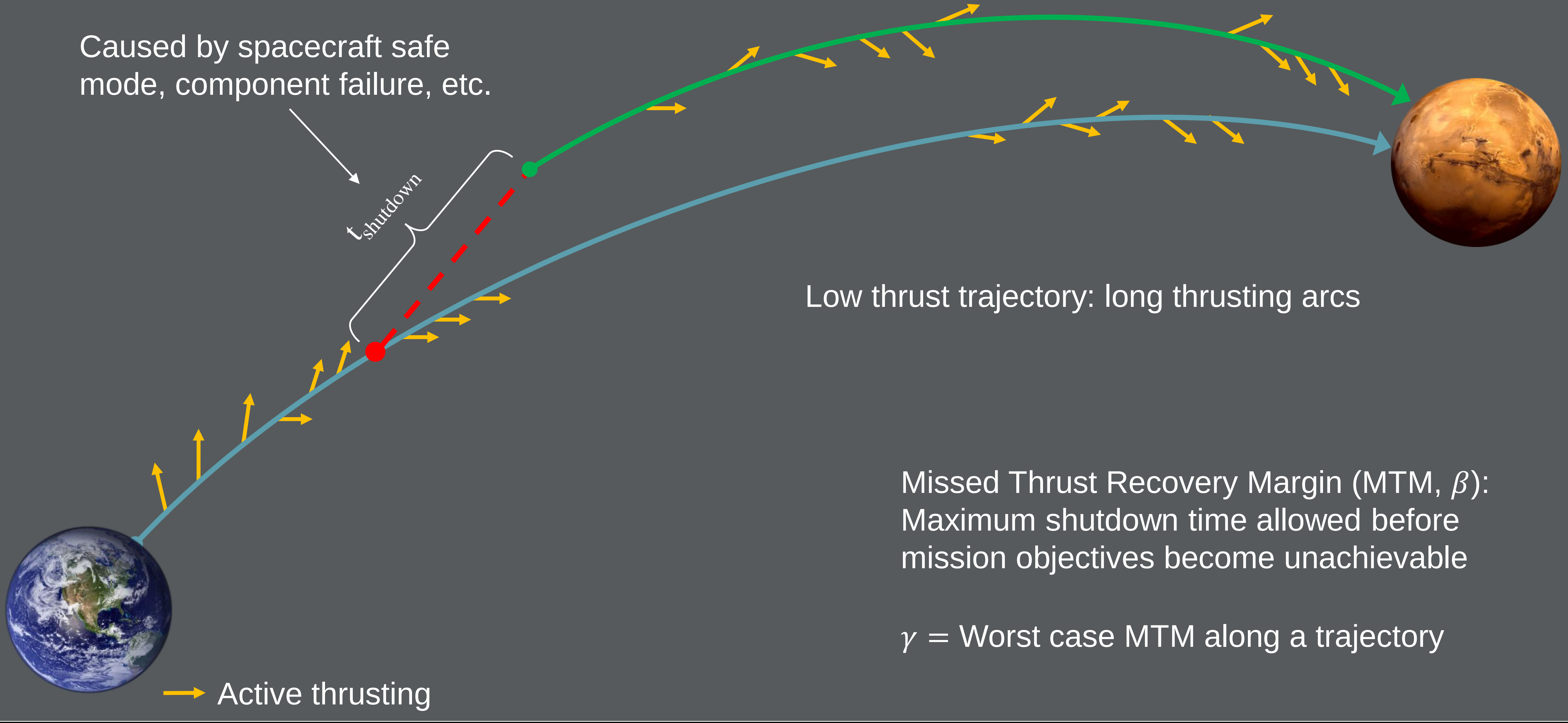
Miscellaneous Notes

- Scaling
  - Distance Unit (DU) is 1 AU (1.496e9 km)
  - Time Unit (TU) set to $\sqrt{\frac{DU^3}{\mu}}$
  - Propagate with real units, present scaled units to optimizer
- User defines “Mission” and “Phase” structs
- NSTOP creates functions to generate decision vector from structs, and update structs from updated decision vectors each iteration

Low Thrust Trajectory Optimization with Robustness to Missed Thrust Events

Research project working with Dr. Jacob Englander and Prof. Scheeres

Missed Thrust Problem



Missed Thrust Recovery Margin (MTM, β):
Maximum shutdown time allowed before mission objectives become unachievable

γ = Worst case MTM along a trajectory

Optimization Objectives of Interest

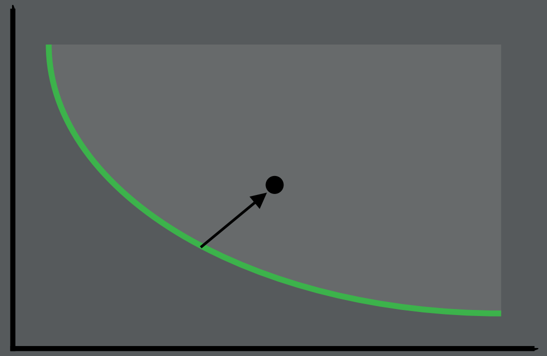
- Delivered mass/propellant mass
- Time of flight
- Missed thrust recovery margin

State of the Art (simplified)

1. Find optimal trajectory for maximum delivered mass or other objective
2. Evaluate trajectory for “weakest” point MTM value γ (lowest MTM)

3a. If “weakest point” is unacceptable, modify the nominal

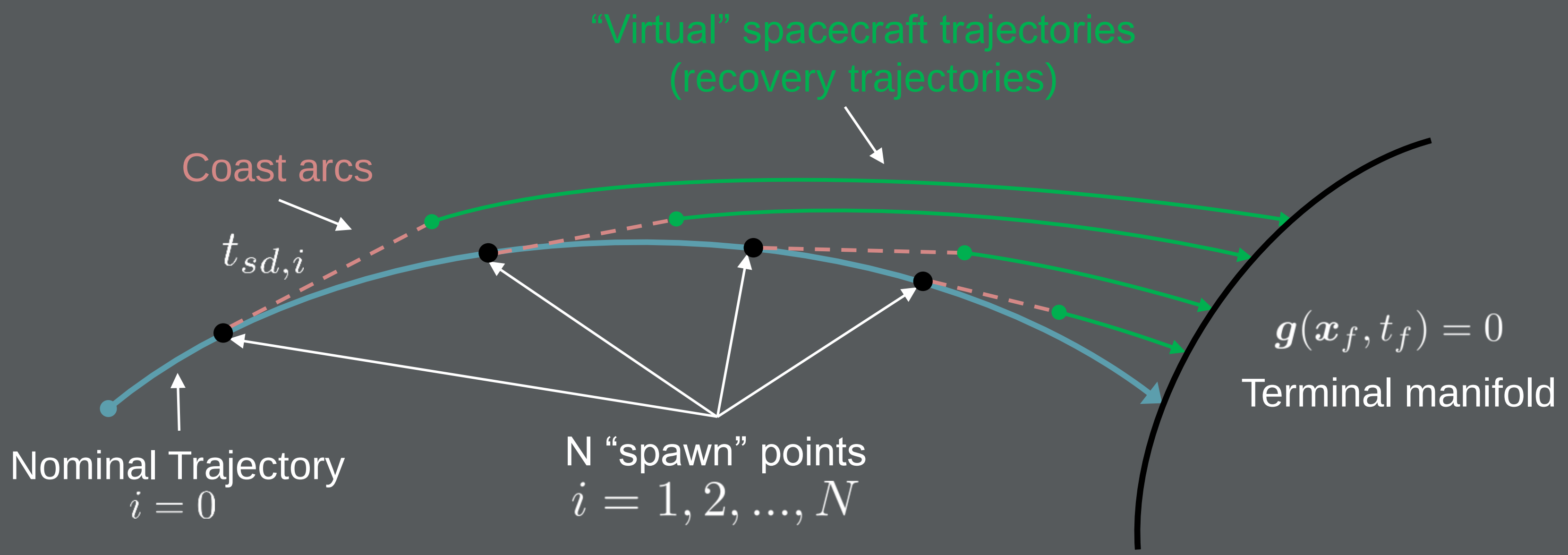
- Enforce coast (zero thrust) at weakest point, re-optimize
- Enforce “rolling coast”
- Modify maximum thrust levels
- Bias nominal arrival date to be earlier
- Etc.


Pareto front

3b. AND/OR carry more reserve fuel, increase time of flight

Problem: Cannot directly optimize or constrain MTM at each point along the trajectory

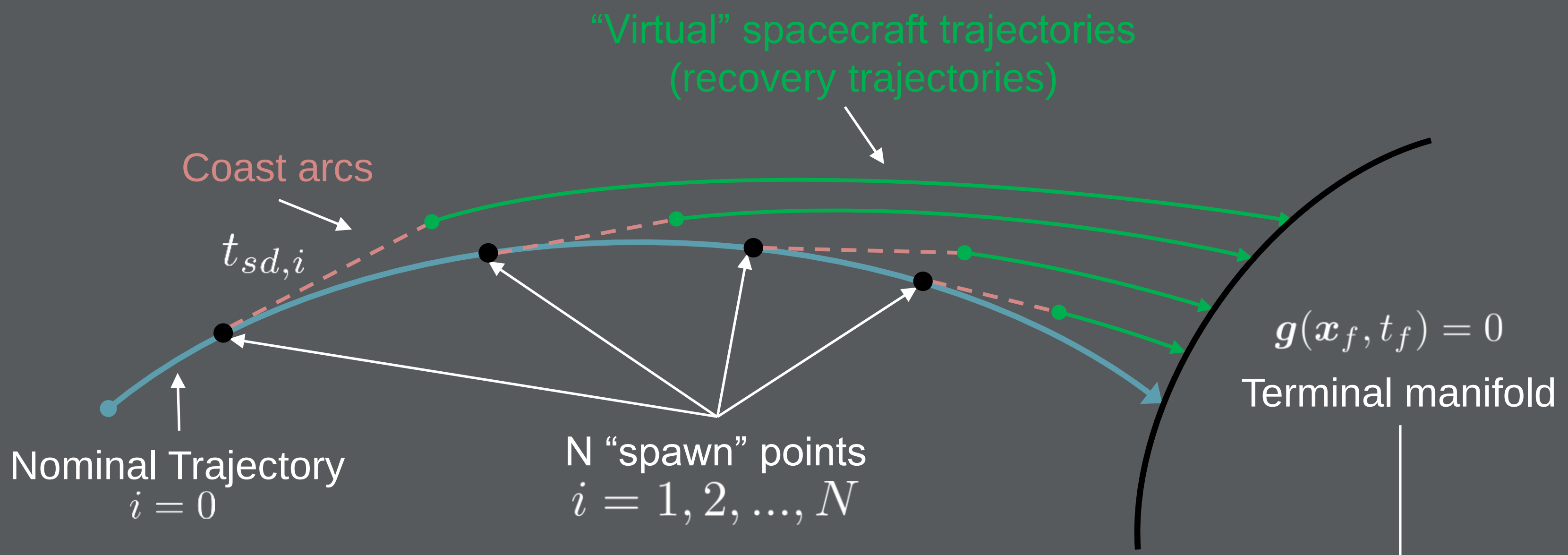
Virtual Swarm Method



- Spawn points: discrete locations where MTM can be controlled
- Virtual spacecraft decision variables and control profiles are independent of the nominal once spawned
- If the number and placement of virtual spacecraft is insufficient, $t_{sd,min} \neq \gamma$

  (worst case MTM is not controlled)

Virtual Swarm Method – Robustness as a Constraint



Maximize delivered mass lower bound: $J = m_{f,min}$

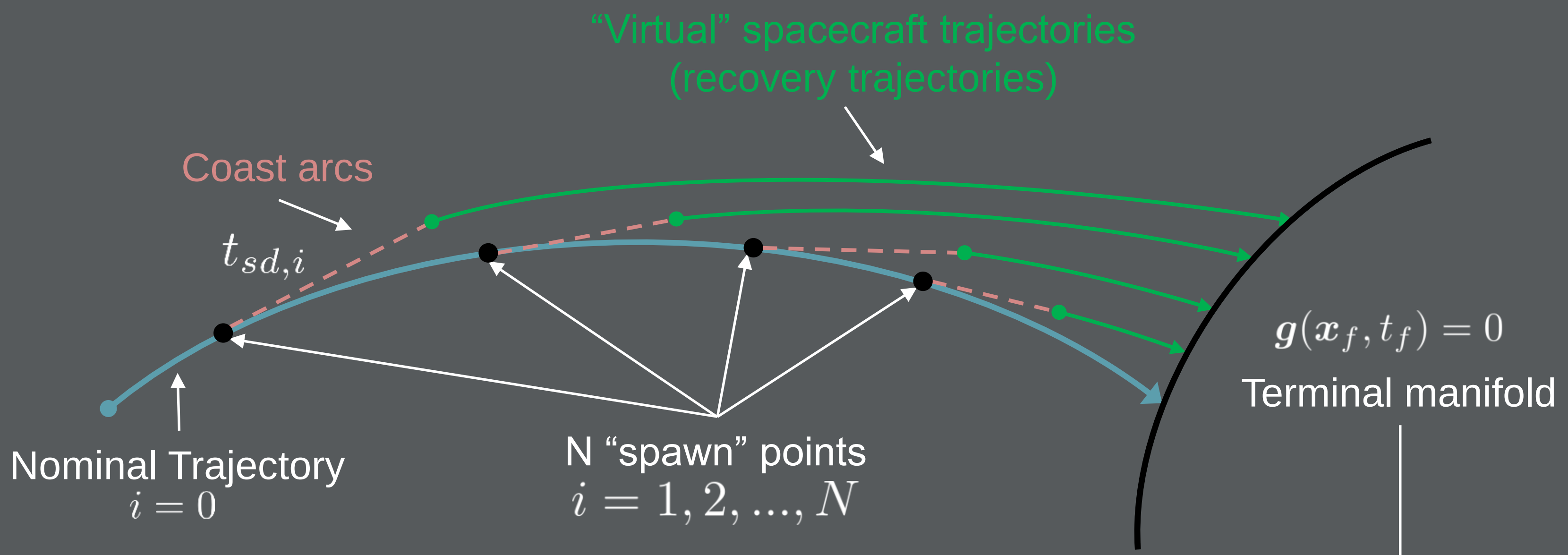
With Constraints: $t_{sd,i} = \gamma_{min} \quad i = 1, 2, ..., N$

$t_{f,min} \leq t_{f,i} \leq t_{f,max} \quad \forall i$

$m_{f,i} \geq m_{f,\min} \quad \forall i$

Including controls: $\boldsymbol{u}_i(\cdot) \quad \forall i$

Virtual Swarm Method – Optimize Worst Case Robustness



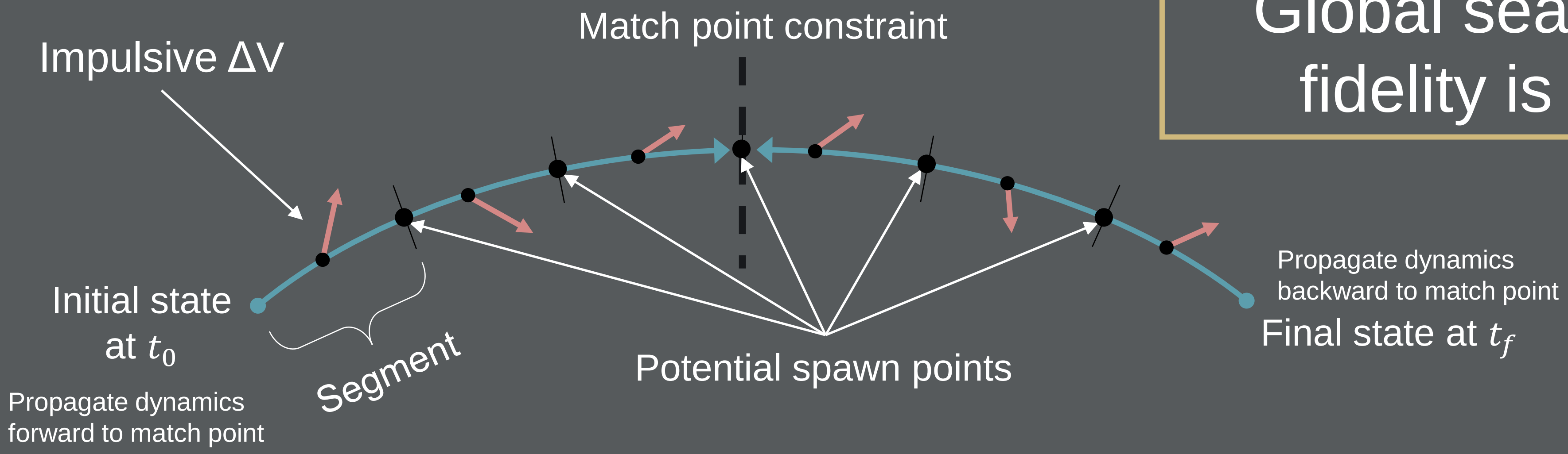
Maximize shutdown time lower bound: $J = t_{sd,min}$

With Constraints: $t_{sd,i} \geq t_{sd,min} \quad i = 1, 2, \ldots, N$ $\qquad t_{f,min} \leq t_{f,i} \leq t_{f,max} \quad \forall i$

$m_{f,i} \geq m_{f,\min} \quad \forall i$

Including controls: $\boldsymbol{u}_i(\cdot) \quad \forall i$

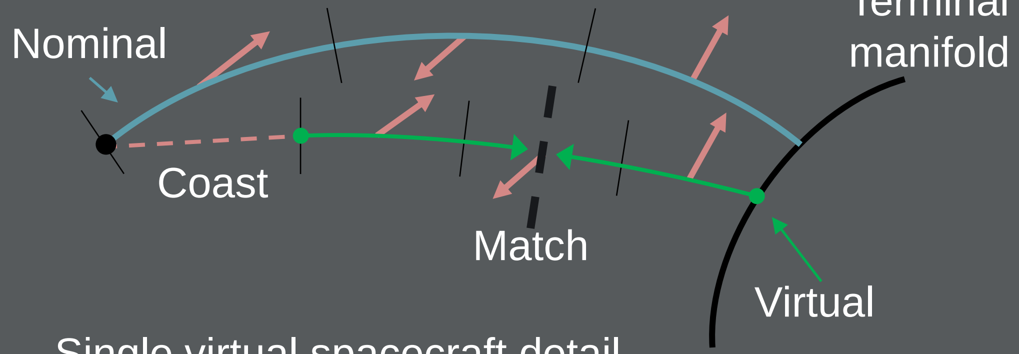
Underlying Transcription (Sims-Flanagan)



Global search: reduced fidelity is acceptable

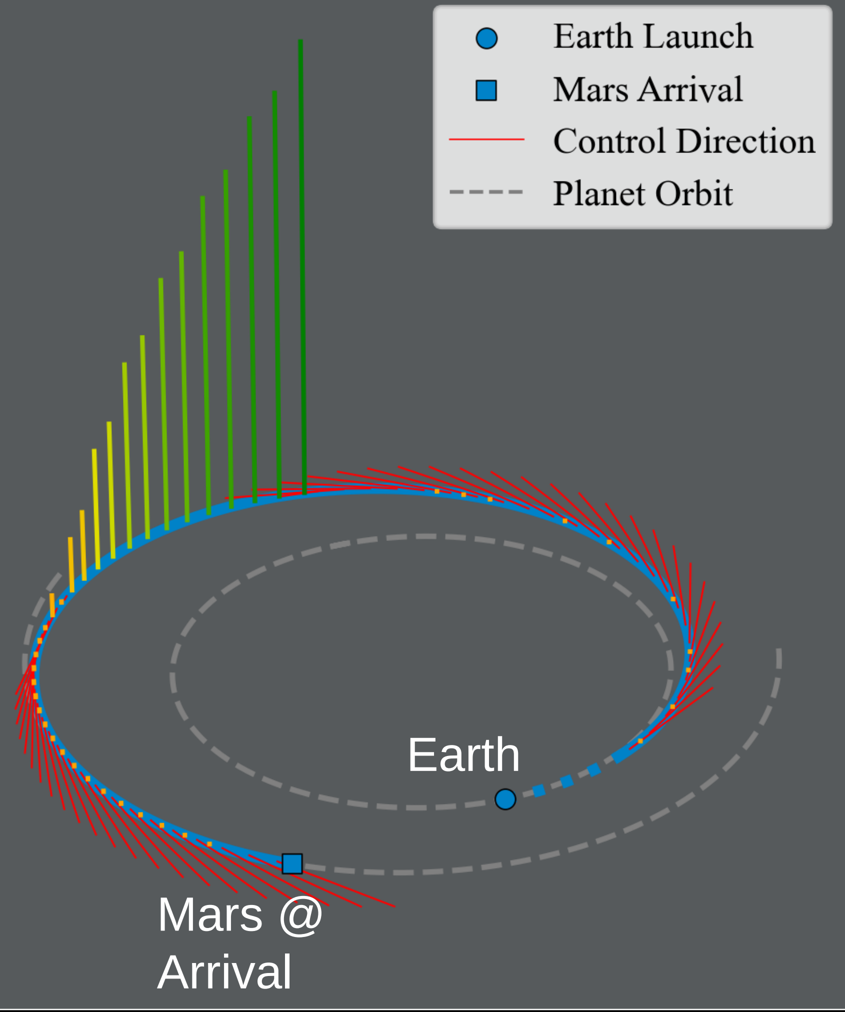
Decision Variables (Robust-Constrained):

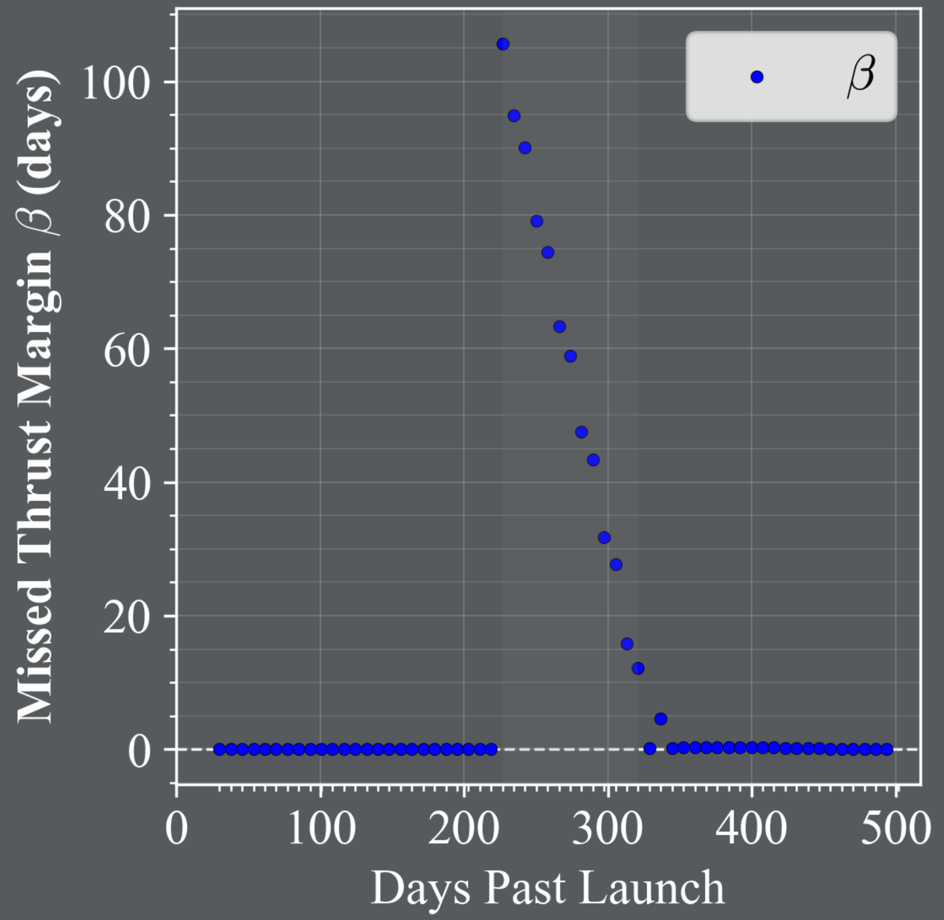
- All spacecraft (N+1):
  - Each impulse magnitude & direction
  - Time of flight
  - Final mass
- Minimum delivered mass constraint



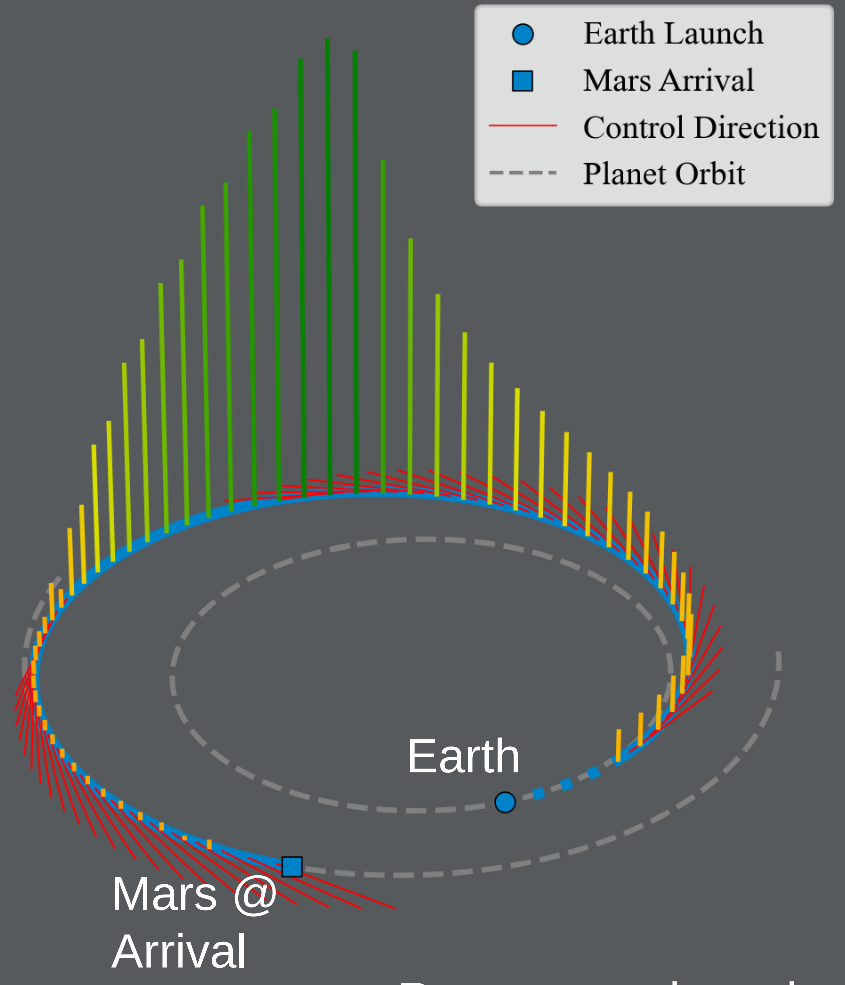
Single virtual spacecraft detail

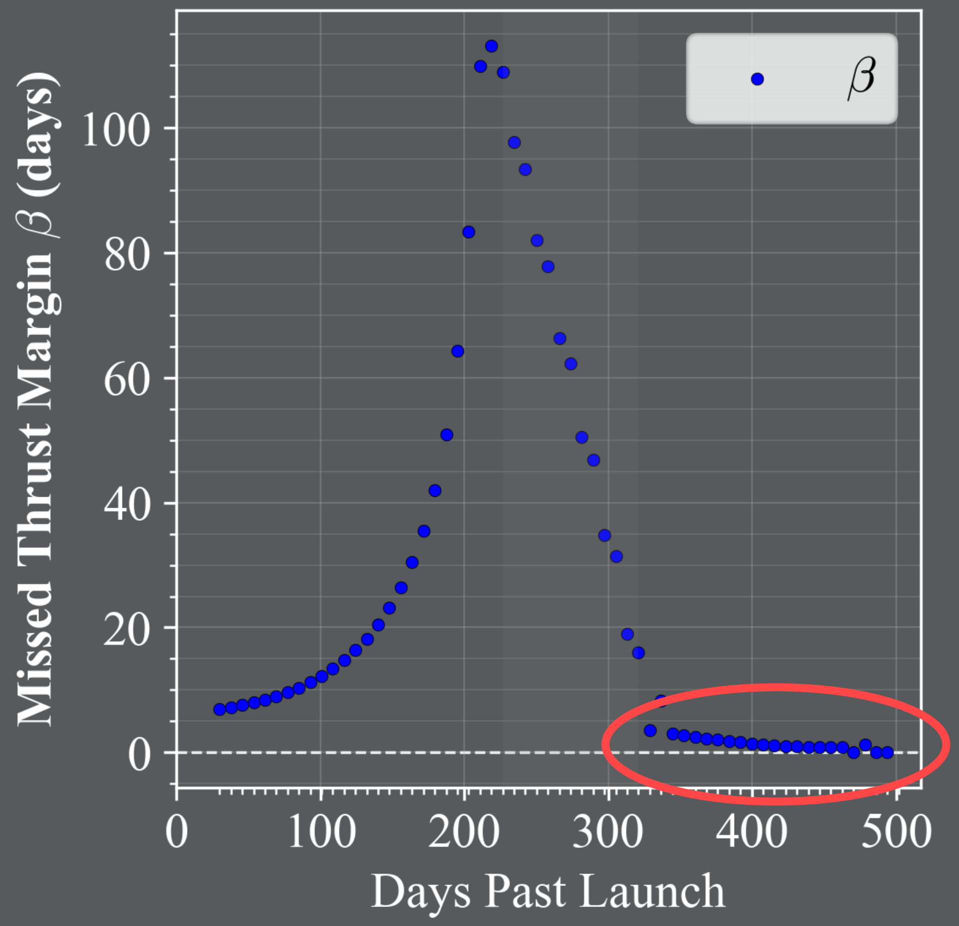
Missed Thrust Margin for Mass Optimal Trajectory: Reference





Missed Thrust Margin for Mass Optimal Trajectory: Permissive



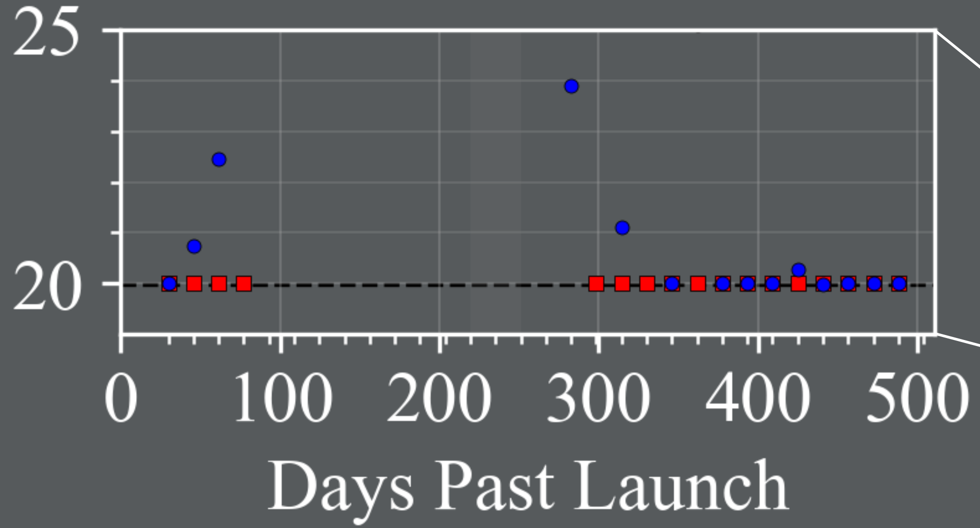


Recovery trajectories are allowed to deliver 30 kg less mass to the target

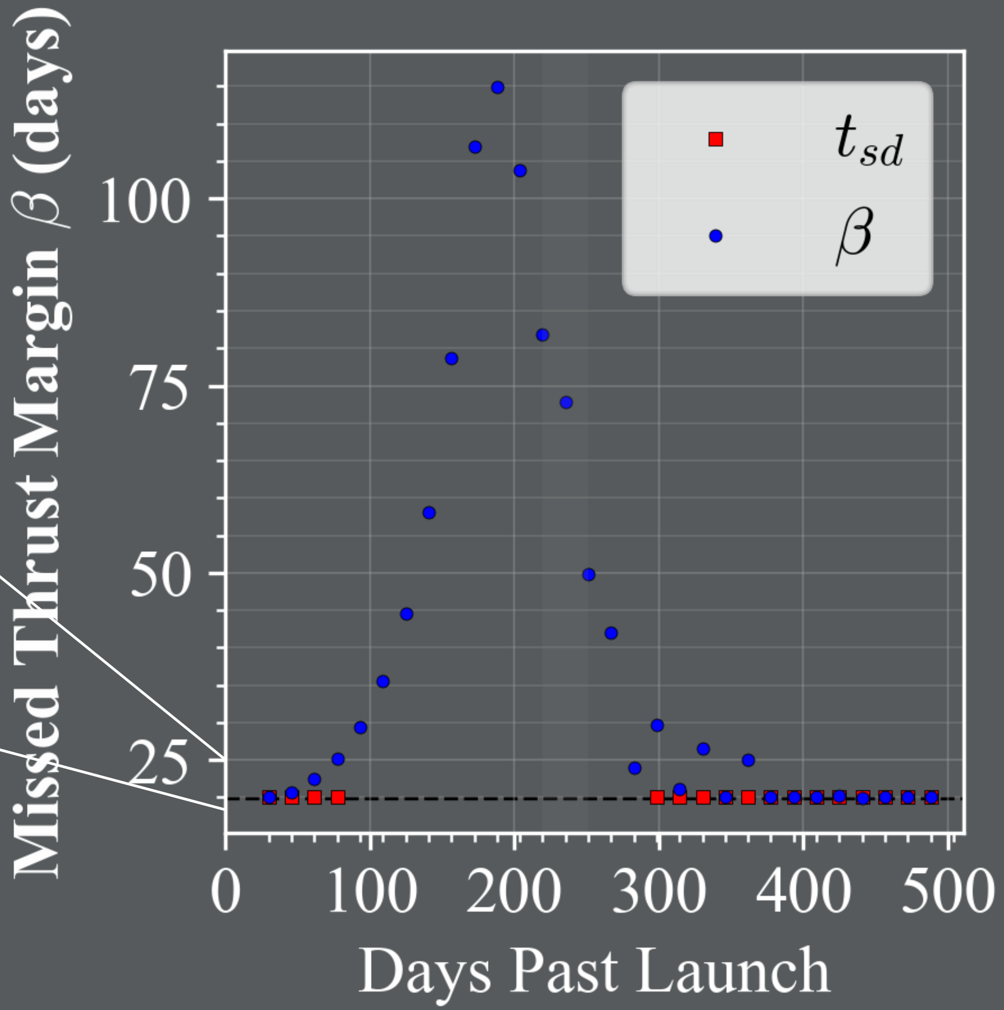
Robust-Constrained Earth-Mars Transfer

Robust constrained trajectory:

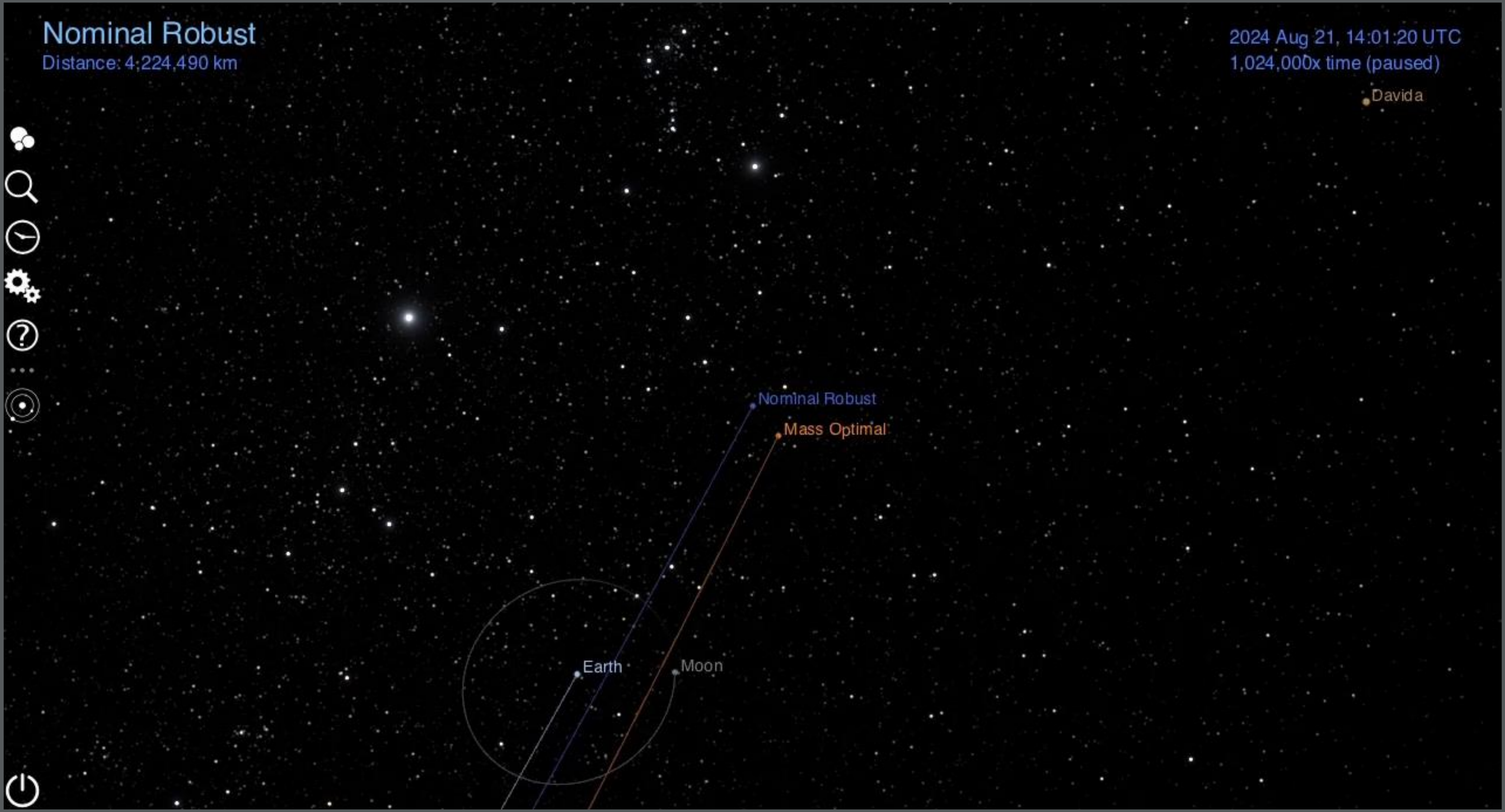
- 20 days “lateness”
- 10% propellant margin compared to mass-optimal reference



Detail view, β from 19 to 25 days

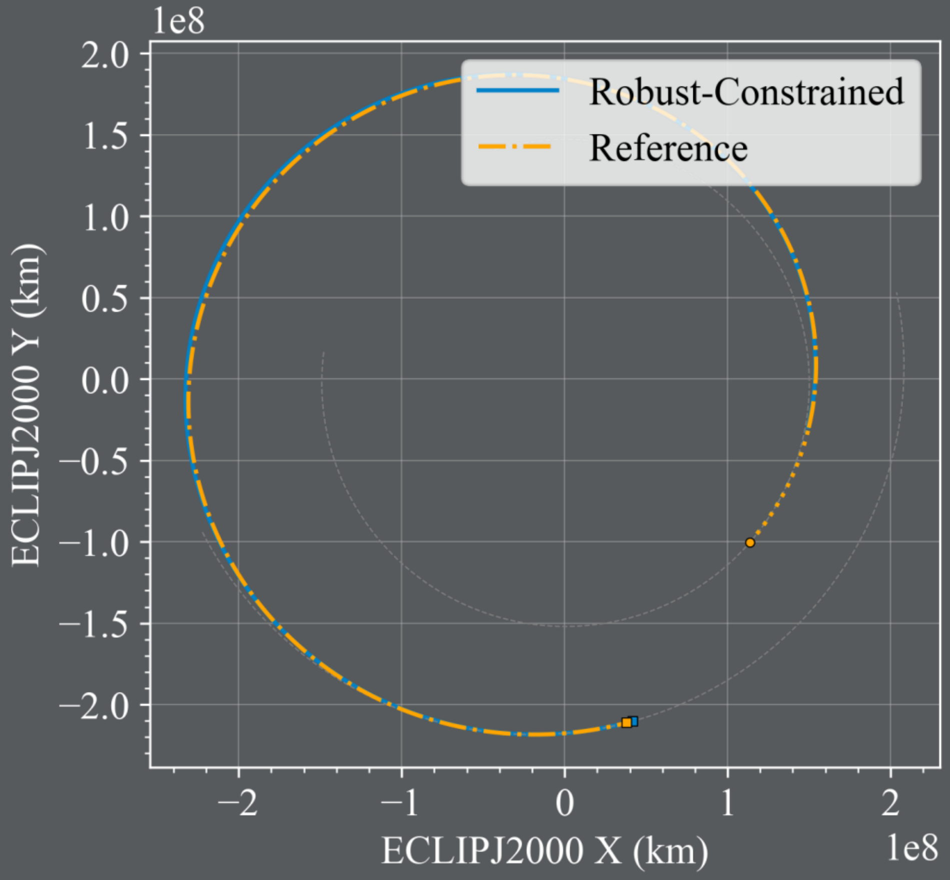


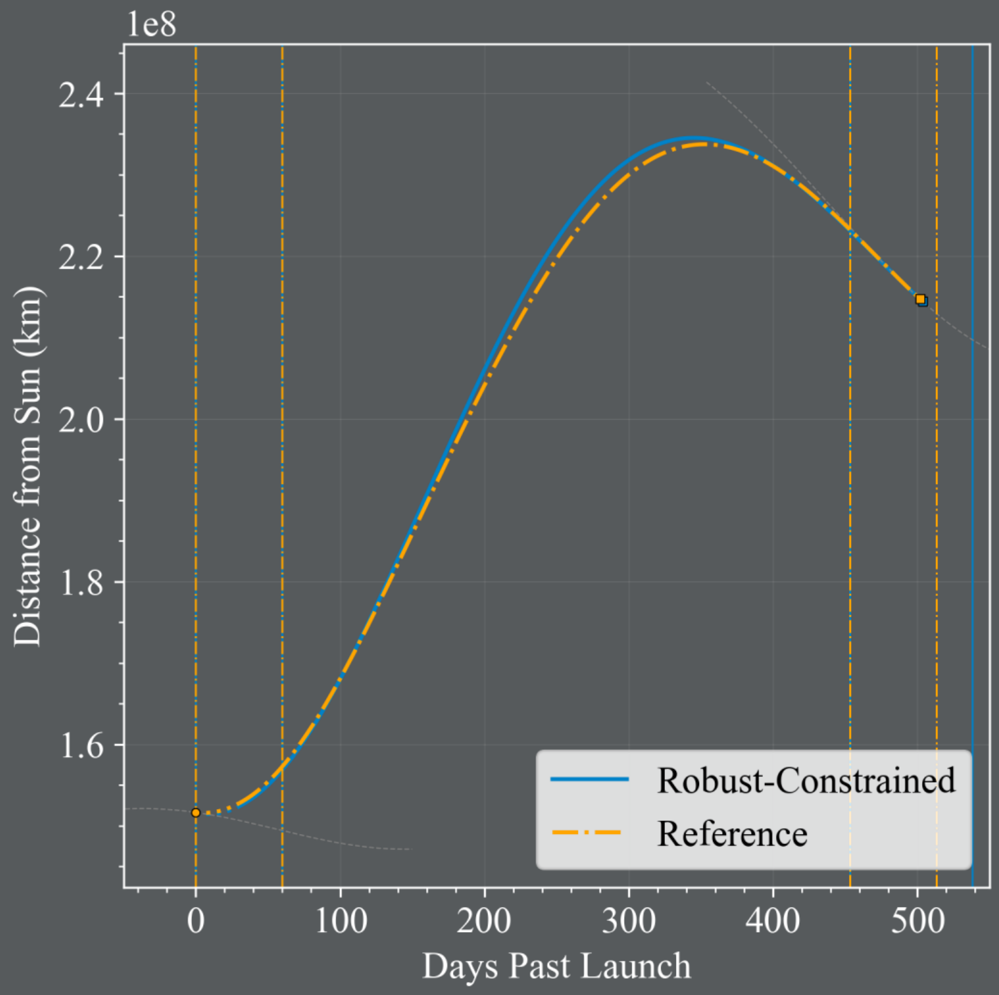
Visualization



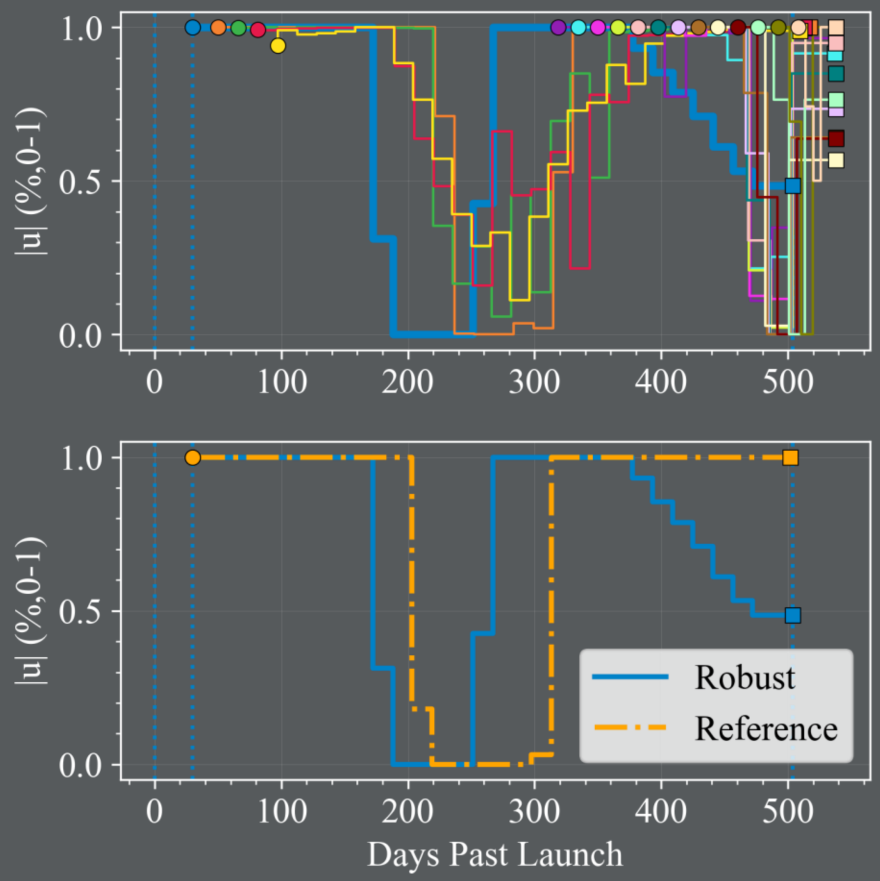
Created in Cosmographia https://naif.jpl.nasa.gov/naif/cosmographia.html

Robust Constrained Earth-Mars Transfer



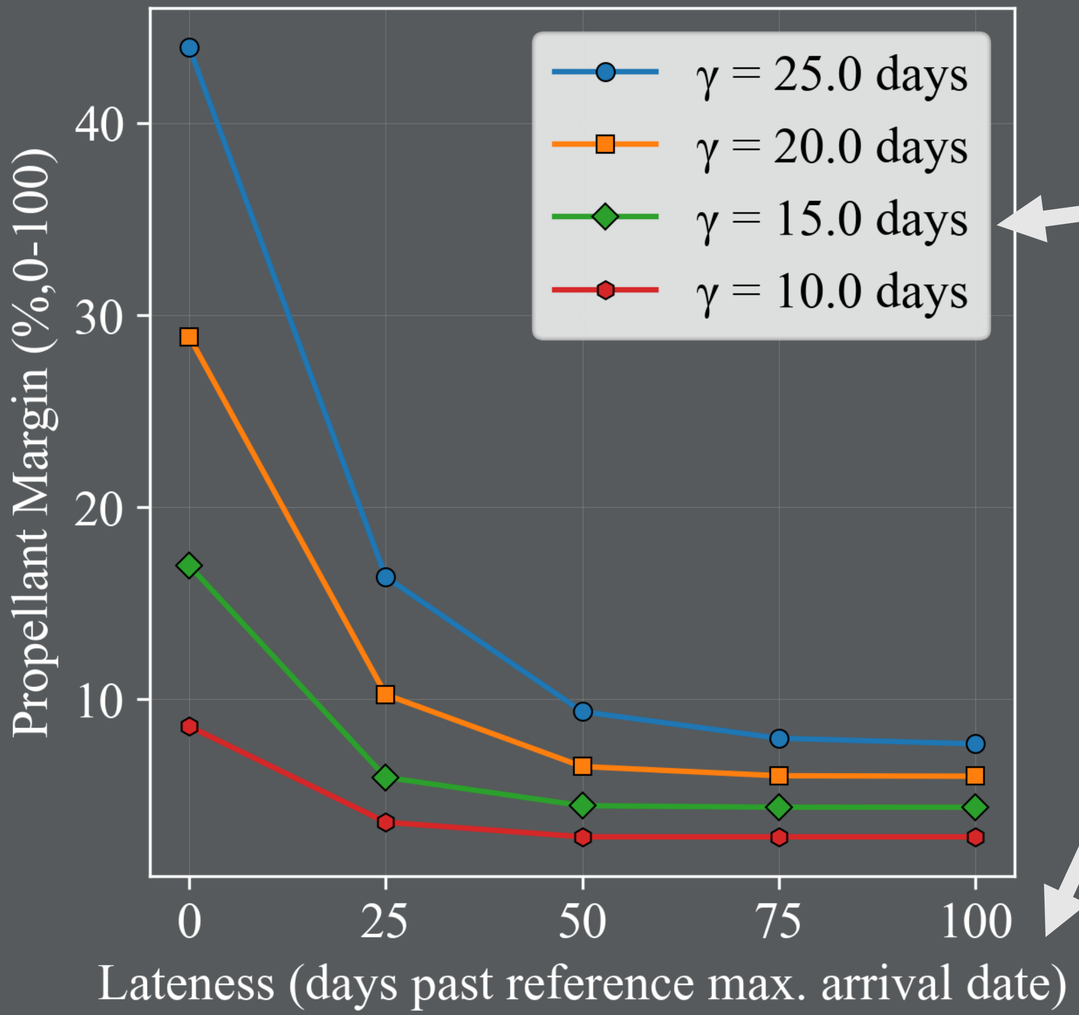


Robust-Constrained Earth-Mars Transfer



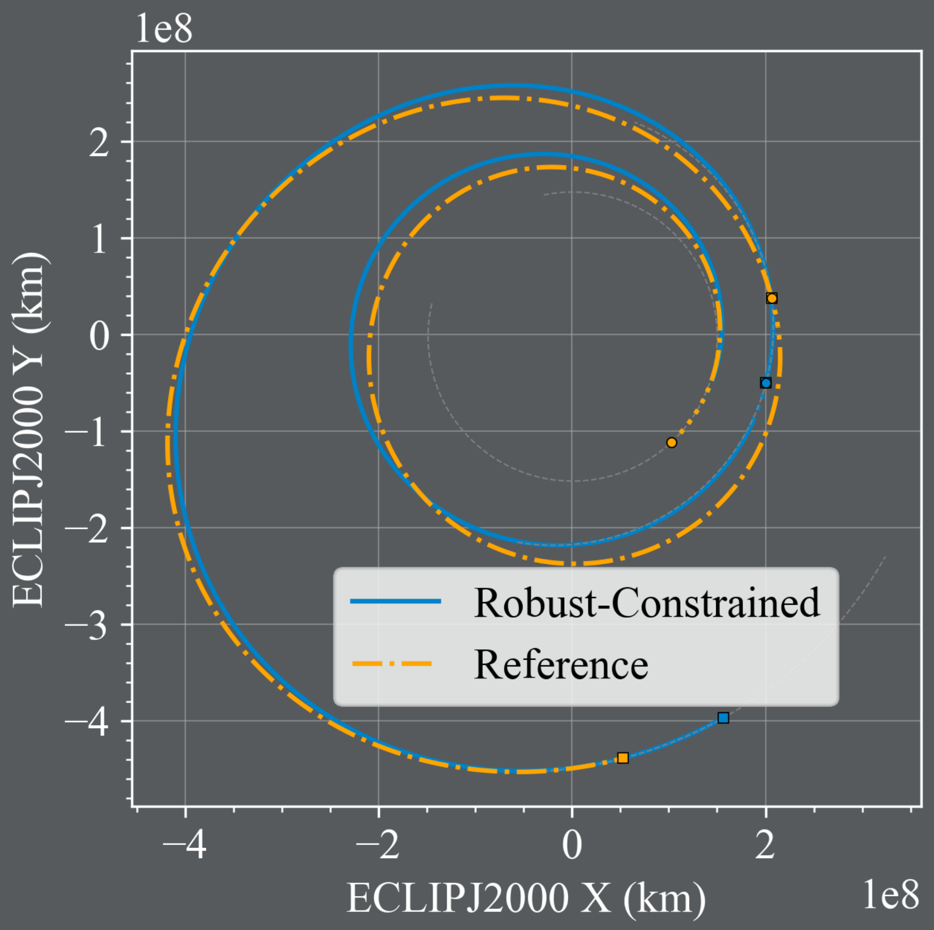
Earth-Mars Transfer Pareto Front

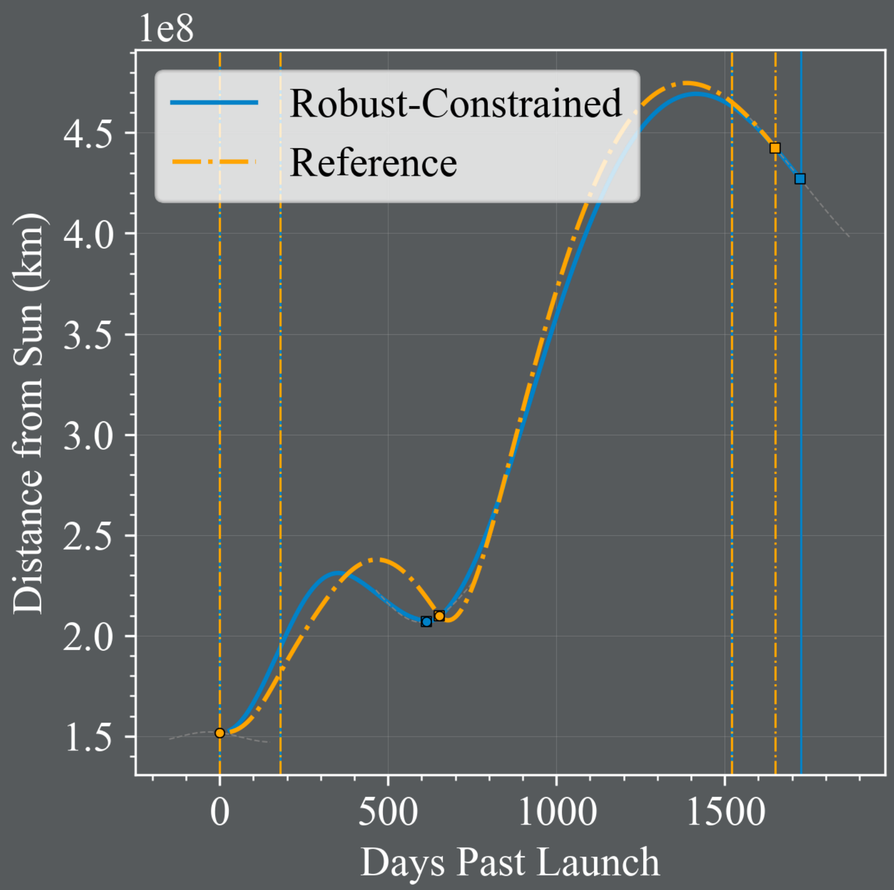
Propellant margin minimized by NLP solver



Constrained/fixed values

Earth-Mars-Psyche Transfers

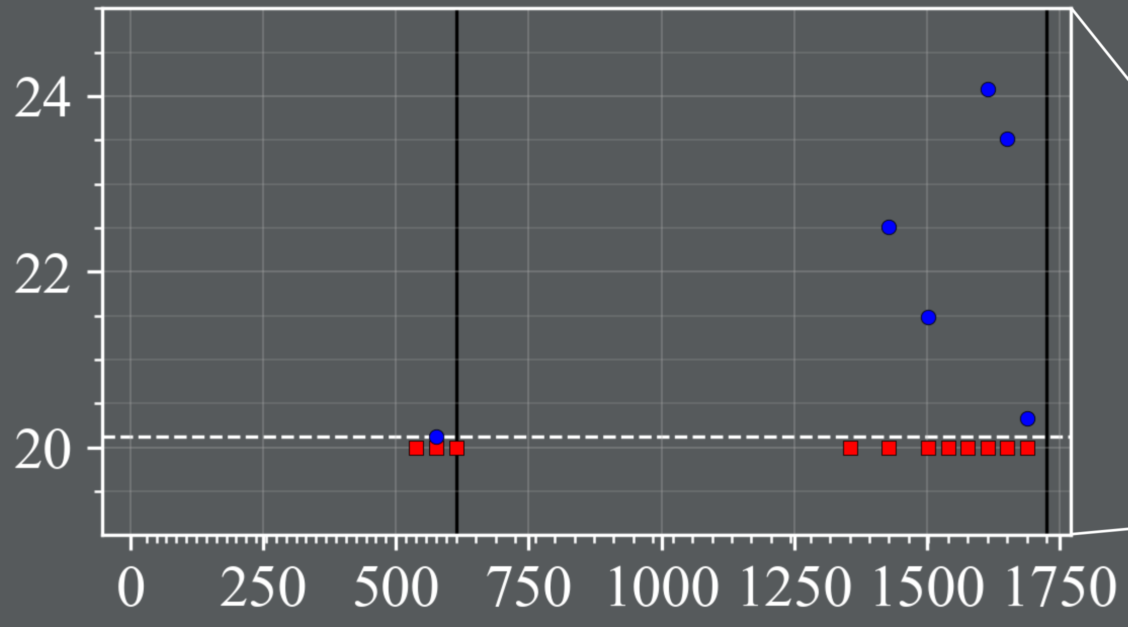




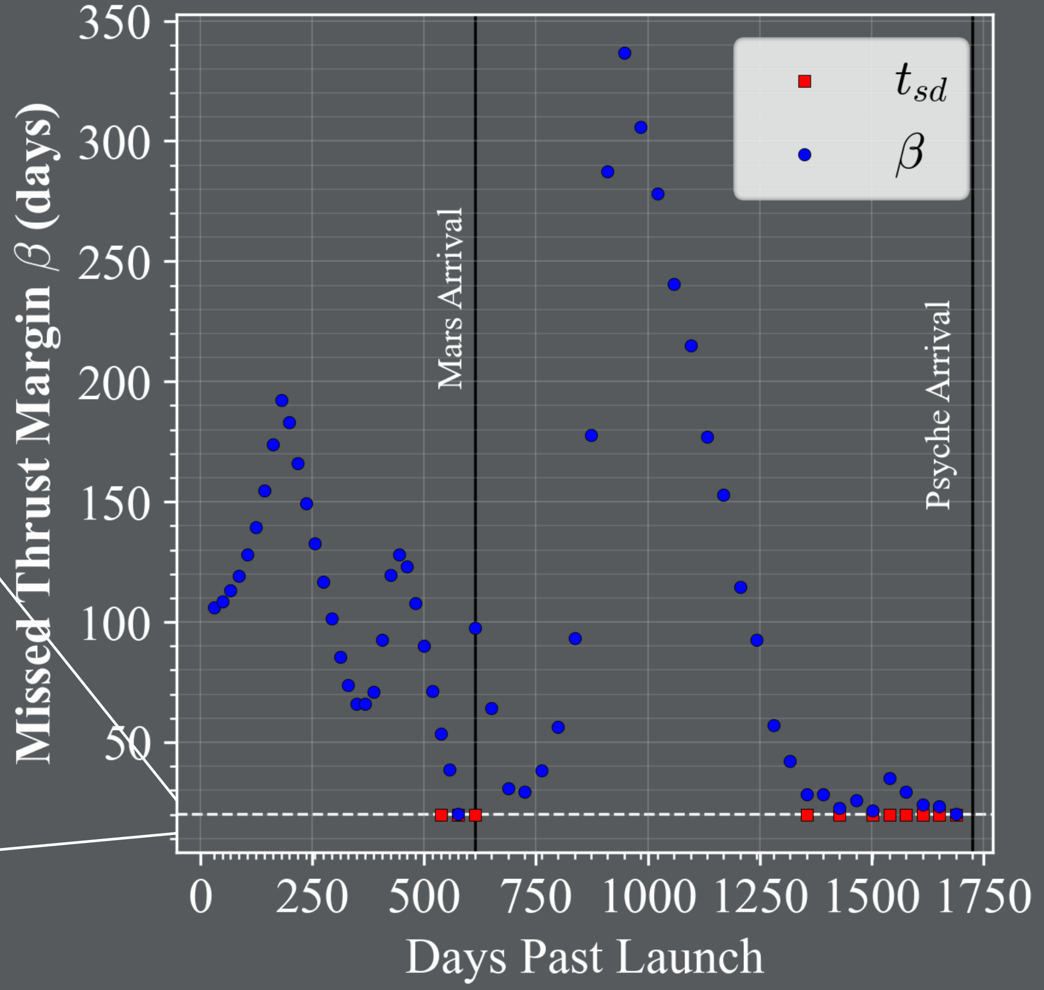
Robust Constrained Earth-Mars-Psyche Transfer

Robust constrained trajectory:

- 75 days "lateness"
- 24% propellant margin compared to mass-optimal reference

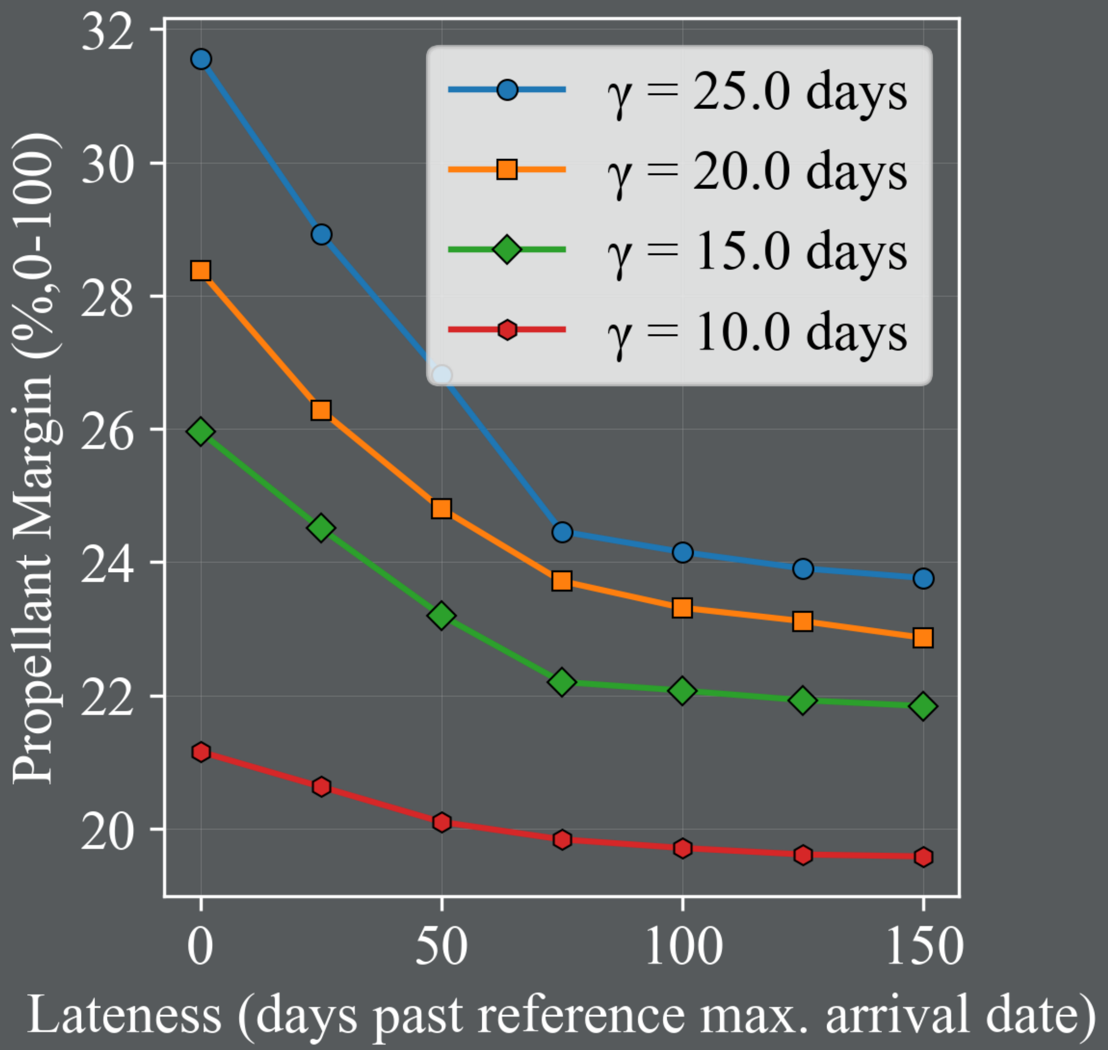


Detail view, β from 19 to 25 days



Earth-Mars-Psyche Pareto Front

Propellant margin: additional mass needed as compared to mass-optimal reference



Summary

- “Virtual swarm” method: new transcription to enable robustness analysis in multi-objective framework
- Enables a mission designer to answer the question: “What is the fuel optimal trajectory to reach a target given a robustness constraint?”
- Method can be used for “real” multi-spacecraft problems as well

Questions?

chve1854@colorado.edu