


Multi-Objective Low-Thrust Trajectory Optimization with Robustness to Missed Thrust Events

Chandranth Venigalla* 

University of Colorado Boulder, Boulder, Colorado 80309

Jacob A. Englander†

NASA Goddard Space Flight Center, Greenbelt, Maryland 20771

and

Daniel J. Scheeres‡

University of Colorado Boulder, Boulder, Colorado 80309

<https://doi.org/10.2514/1.G006056>

This paper introduces a new technique for directly controlling the missed thrust recovery margin (MTRM) of a low-thrust spacecraft trajectory. MTRM is defined here as the longest amount of time a spacecraft may coast away from a nominal trajectory while still being able to reach a terminal manifold once thruster operations are resumed. The “virtual swarm” optimization technique developed here simultaneously optimizes the nominal spacecraft trajectory along with many recovery trajectories. The objective can be to maximize the MTRM of the nominal trajectory at its weakest point or to constrain the worst-case MTRM to be at or above a desired level while optimizing a different value (e.g., mass, time of flight). The technique is demonstrated for a direct Earth–Mars transfer and for a gravity assist trajectory to the asteroid Psyche. Further, a method for finding the Pareto front of MTRM, arrival mass, and arrival date is presented to address the related multi-objective optimization problem.

Nomenclature

f	= spacecraft state dynamics function
g	= constraint function
i	= index for spacecraft in the virtual swarm
J	= objective function
j	= index for impulses in a phase
k	= index for phases in a trajectory
l	= function dictating the maximum initial mass that can be launched to a given C_3 value, kg
M	= function used to calculate β
m	= spacecraft mass, kg
N	= total number of virtual spacecraft mass, kg
N_{ph}	= total number of phases in a trajectory
\mathbf{r}	= spacecraft position vector, km
T_{max}	= maximum spacecraft time of flight, days
t_{sd}	= forced shutdown time (missed thrust event length), days
\mathbf{u}	= spacecraft control vector, m/s ²
\mathbf{v}	= spacecraft velocity vector, km/s
\mathbf{x}	= spacecraft state
β	= missed thrust recovery margin: maximum shutdown time allowable at a given point, days
γ	= worst-case missed thrust margin β along a given trajectory, days
$\eta_{underload}$	= underload factor to reduce initial mass

κ	= propellant margin, %
$\nu_{j,k}$	= binary variable set to 1 if a virtual spacecraft is spawned at segment j in phase k
τ	= total number of impulses in a trajectory
τ_k	= total number of impulses in phase k
ψ	= function to constrain initial state of the spacecraft

I. Introduction

LOW-THRUST propulsion methods, especially solar electric propulsion (SEP), are becoming increasingly used in space missions. These methods are useful for their efficiency and ability to, in many cases, deliver more massive payloads to a target than traditional chemical propulsion methods. However, low-thrust methods typically require that a thruster be on for long periods of time, whereas a chemical mission may only require the firing of a thruster for comparatively fewer instances and shorter periods of time in each instance. In the event that some issue prevents a spacecraft with low-thrust propulsion from following its nominal thrust profile (e.g., spacecraft transitions to safe mode and ceases thrusting), the outage has the potential to render the desired target inaccessible given the available propellant on-board and time available to reach the destination. In other words, a surprise loss of thruster operations may cause the mission to fail if the nominal trajectory is not designed to be robust to such losses. In this work, we define the missed thrust recovery margin (MTRM) as the longest amount of time a spacecraft may coast away from a nominal trajectory while still being able to reach a terminal manifold once thruster operations are resumed. In other words, the MTRM at a given point on a trajectory is the length (e.g., days) of the longest forced coast period that still allows the spacecraft to reach its target. While MTRM is important in a variety of contexts, this work specifically focuses on the MTRM problem in interplanetary trajectories. An Earth-orbiting low-thrust spacecraft may rely less on precise timing and orbital periods around Earth are relatively short, whereas an interplanetary low-thrust spacecraft will likely be relying on relative phasing of various celestial bodies and likely cannot afford to wait a full orbital period before resuming operations.

A great deal of work on trajectory optimization has focused on mass- and time-optimal trajectories, but there is a gap in work on directly optimizing the MTRM of a trajectory in the literature. Laipert

Received 11 March 2021; revision received 10 January 2022; accepted for publication 11 January 2022; published online 15 February 2022. Copyright © 2022 by Chandranth Venigalla, Jacob A. Englander, and Daniel J. Scheeres. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 1533-3884 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Ph.D. Student, Department of Aerospace Engineering Sciences, 3775 Discovery Drive; chve1854@colorado.edu. Student Member AIAA.

†Aerospace Engineer, Navigation and Mission Design Branch, 8800 Greenbelt Road; currently Mission Design Engineer, The Johns Hopkins University Applied Physics Laboratory.

‡A. Richard Seebass Endowed Chair Professor, Department of Aerospace Engineering Sciences, 3775 Discovery Drive. Fellow AIAA.

and Longuski [1] address the missed thrust problem by evaluating how much additional propellant is needed to allow a certain amount of MTRM throughout a given nominal trajectory. They also explore the tradeoff between the additional propellant margin and how late the spacecraft arrives at the target. Critically, however, their technique does not enable the reshaping of the nominal trajectory to be more robust to missed thrust events. Although their technique enables the analysis of how much reserve propellant is needed to meet robustness requirements, the result is not necessarily giving the most propellant optimal way of doing so. Laipert and Imken [2] analyze the potential impact of multiple missed thrust events using a Monte Carlo approach given historical data and statistics on missed thrust events in past space missions. This analysis similarly uses propellant margin and lateness to measure the robustness of a given trajectory to the multiple missed thrust events.

Ozaki et al. [3] explore a stochastic differential dynamic programming approach to dealing with uncertainty in low-thrust spacecraft missions. However, the theory in its current form does not appear to have the ability to directly account for the missed thrust problem. Instead, uncertainty is modeled as Gaussian disturbances. Olympio [4] also takes a stochastic approach to the problem. Olympio and Yam [5] address the missed thrust problem with a constrained optimization problem where the MTRM at each point in the nominal, mass-optimized trajectory is constrained to be greater than or equal to some threshold MTRM value. MTRM in the optimization problem is calculated using an approximation that is found by fitting a function to discrete points in state space where MTRM is evaluated. The technique developed in the present work is not totally dissimilar, though does not use a function approximation to constrain MTRM. As reported by Oh et al. [6], the Dawn mission used a “rolling coast” method to ensure that a minimum of 28 days of forced shutdown time was always possible at any point in the nominal trajectory. Ad hoc methods include inserting coast arcs into more sensitive (low MTRM) parts of a nominal trajectory [7] and lowering the planned thruster duty cycle for sensitive parts of a trajectory.

Concurrent with the development of the present work, McCarty and Grebow [8] also developed a similar method to address the missed thrust problem. The “ghost” trajectories discussed in their work are quite similar to the “virtual” trajectories developed in the present work. Ghost trajectories are applied in the three-body problem to make cis-lunar trajectories more robust to missed thrust events. The virtual swarm method developed here in Secs. II and III is generally applicable across dynamic models, but the example results focus on interplanetary trajectories using Keplerian dynamics (Secs. IV–V). McCarty and Grebow use a fixed number of ghost trajectories (five) in targeted locations to improve MTRM at points in the nominal trajectory known to have the worst robustness to missed thrust events. The limited number of ghost trajectories reduces the number of trajectories that must be simultaneously optimized to lessen the computational load, but does not necessarily allow MTRM to be fully constrained or optimized across the entire trajectory. The virtual swarm method developed here requires a variable number of virtual spacecraft to be added to the problem to ensure that worst-case MTRM is controlled across the entire nominal trajectory. Given the more complex dynamics in the three-body problem, applying the virtual swarm method to cis-lunar trajectories would likely be more challenging than the interplanetary example trajectories that are explored in the present work. Notably, the ghost trajectories in [8] also appear to have a fixed thrusting structure, meaning that the optimization process cannot uncover alternative thrust profiles that may be ideal for recovery trajectories. The Sims–Flanagan transcription used in the examples here provides more freedom for the optimizer to select thrusting and coasting arcs.

Rather than relying on ad hoc, heuristic, or potentially overconservative approaches to making a nominal low-thrust trajectory robust, we instead aim to directly optimize or constrain the worst-case MTRM along a nominal, deterministic trajectory. This allows a mission designer to either meet a given robustness constraint throughout a trajectory, or allows them to understand the best

possible value of the worst-case MTRM. The present work introduces a “virtual swarm” technique where a nominal spacecraft trajectory is simultaneously optimized with a discrete number of recovery trajectories with consideration for MTRM. The MTRM can either be constrained in each recovery trajectory to be some minimum value (enforce a shutdown of at least some amount of time) with some other variable being optimized, or alternatively the lower bound constraint on the MTRM for each recovery trajectory can be used as the optimization variable to find a maximally robust nominal trajectory. Note that, in this work, when optimizing MTRM, we specifically are interested in optimizing the worst-case MTRM to ensure that the worst-case recovery time is as long as possible. This is in line with typical requirements for past missions such as the Dawn mission (28 days minimum recovery time at any point [9]), but if more uniformity or certain distributions of MTRM throughout a trajectory are desired, future work may consider a different optimization objective.

In addition to addressing the single-objective problem of optimizing worst-case MTRM along a trajectory, this work also addresses the multi-objective problem because such considerations are critically important in the mission development phase. It is rare for a mission designer to have only a single objective to consider; frequently they are dealing with many different aspects of the mission to find the “best” overall trajectory. Thus, the impact of optimizing worst-case MTRM on other aspects of the trajectory is also of interest. The virtual swarm method developed here allows a multi-objective analysis of the tradeoff between worst-case MTRM along a nominal trajectory, delivered mass, and target arrival date. Specifically, we aim to develop an automated method of generating a Pareto front for these objectives that presents the best-case scenario for one objective given that the other objectives are fixed. Solutions that are represented along a Pareto front cannot improve one of the objectives without performing worse in another. A Pareto front is perhaps one of the best ways to give decision makers the ability to weigh different objectives that may not have a “correct” answer. For example, each mission may prioritize the competing objectives of missed thrust robustness and payload mass in different ways, and an informed decision may be best made by considering the full curve of Pareto optimal solutions. Further, methods that can be automated are important in allowing the efficient exploration of a trade space; too much need for human intervention may make exploration of the trade space too difficult to do in a tractable amount of time.

II. Problem Statements

First, consider the low-thrust spacecraft nominal trajectory optimization problem of maximizing the total delivered mass to a target. In the terminology of optimal control, this is the problem of maximizing the objective

$$J = m_f \quad (1)$$

with general state dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t; \boldsymbol{\alpha}) \quad (2)$$

where \mathbf{x} is the spacecraft state

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ m \end{bmatrix} \quad (3)$$

with Cartesian position \mathbf{r} , Cartesian velocity \mathbf{v} , and mass m . Control vector \mathbf{u} is provided by the propulsion system, and $\boldsymbol{\alpha}$ is a set containing spacecraft parameters (e.g., solar panel size, thruster performance). There is the terminal constraint to match the state of the target at the final time

$$\mathbf{g}_f(\mathbf{x}_{rv}(t_f), t_f) = \mathbf{x}_{rv}(t_f) - \mathbf{x}_{rv}^{\text{target}}(t_f) = \mathbf{0} \quad (4)$$

where

$$\mathbf{x}_{rv} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} \quad (5)$$

refers only to Cartesian state, and there are inequality constraints on initial and final times

$$t_{0,\min} \leq t_0 \leq t_{0,\max} \quad (6)$$

$$t_{f,\min} \leq t_f \leq t_{f,\max} \quad (7)$$

and potentially an inequality constraint to ensure that the time of flight (TOF) is less than a desired maximum TOF, T_{\max} :

$$t_f - t_0 \leq T_{\max} \quad (8)$$

There is also a constraint on the initial state

$$\mathbf{g}_0(\mathbf{x}(t_0), t_0) = \mathbf{x}(t_0) - \boldsymbol{\psi}(t_0, \boldsymbol{\rho}) = \mathbf{0} \quad (9)$$

where the function $\boldsymbol{\psi}$ determines the initial state of the spacecraft as a function of time and as a function of the set of auxiliary parameters $\boldsymbol{\rho}$. The set $\boldsymbol{\rho}$ can include parameters such as C_3 , launch asymptote, and related parameters. At its simplest, $\boldsymbol{\psi}$ may simply fix the initial state of the spacecraft on some orbit (e.g., Earth's heliocentric orbit). Parameters in $\boldsymbol{\rho}$ may also have associated equality or inequality constraints.

The objective J is optimized by selecting decision variables \mathbf{u} , t_0 , and t_f and potentially some or all of the auxiliary parameters $\boldsymbol{\rho}$. This problem will be referred to as the “reference problem” or “mass-optimal” problem, and it is frequently a baseline problem being solved by mission designers who use a variety of additional methods to take into account other mission objectives and constraints that are not explicitly accounted for in this formulation.

This work focuses on augmenting the reference problem with consideration for MTRM, which can be considered as either a constraint or an objective. The MTRM β at an arbitrary point in state space \mathbf{x} at time t is defined as

$$\beta = M(\mathbf{x}, t; \mathbf{g}(\cdot), \mathbf{f}(\cdot)) = \max_{\mathbf{u}, (\cdot) \in \mathbb{U}} t_{sd} \quad (10)$$

where t_{sd} is the forced shutdown time when the spacecraft cannot thrust. There is the constraint

$$\mathbf{u}(t) = \mathbf{0} \quad \text{for } t_0 \leq t \leq t_{sd} \quad (11)$$

enforcing that the spacecraft cannot thrust until $t > t_{sd}$ and the constraint

$$m_f \geq m_{f,\min} \quad (12)$$

The maximization problem in Eq. (10) is also subject to the same state dynamics $\mathbf{f}(\cdot)$ as the reference problem and the same Cartesian terminal constraint in Eq. (4). The values of $t_{f,\min}$, $t_{f,\max}$, and $m_{f,\min}$, however, can be selected to match the values used and found in the reference problem or selected to be more permissive to allow recovery trajectories to have worse performance than the reference. For example, if $m_{f,\min}$ is chosen to be the optimized value of m_f found in the reference problem and the same time bounds on t_f are used as in the reference problem, the value of β represents the amount of time t_{sd} that is tolerable at the point in state space while still being able to deliver a mass of m_f to the target within the original time bounds. A more permissive approach would calculate β with a longer allowed time of flight and/or lower amount of delivered mass (e.g., if a missed thrust event occurs it is acceptable to arrive at the target 30 days later than the original $t_{f,\max}$).

The set \mathbb{U} contains all control functions $\mathbf{u}(\cdot)$ that generate a state trajectory $\mathbf{x}(\mathbf{u}(t), t) \forall t$ that satisfy the terminal constraints given a starting point \mathbf{x} and t as well as spacecraft parameters such as thruster performance characteristics. Some optimal recovery control function $\mathbf{u}_r^*(\cdot) \in \mathbb{U}$ will both meet the constraint in Eq. (11) and maximize t_{sd} . That is, $\mathbf{u}_r^*(\cdot)$ is a function describing the recovery optimal control that will allow the longest forced shutdown period t_{sd} .

Though β is a quantity that can be defined at arbitrary points in state space, here it is of interest when evaluated along the nominal trajectory of a low-thrust spacecraft. Especially important in mission design is the worst-case missed thrust margin of a nominal spacecraft trajectory. That is, considering all points along a nominal trajectory, what is the shortest amount of time a spacecraft has to recover from a forced shutdown event before it can no longer reach its target? For a given spacecraft state trajectory $\mathbf{x}(\mathbf{u}(t), t)$, the worst-case missed thrust margin γ is

$$\gamma = \min_t M(\mathbf{x}(\mathbf{u}(t), t), t; \mathbf{g}(\cdot), \mathbf{f}(\cdot)) \quad (13)$$

In other words, γ is the lowest value of β found along a nominal trajectory as t is varied from t_0 to t_f .

Frequently, there is some lower bound requirement on γ for a low-thrust mission (e.g., $\gamma \geq 28$ days for the Dawn mission [6]). Thus, the problem of accounting for this can be considered as adding the constraint

$$\gamma \geq \gamma_{\min} \quad (14)$$

to the reference problem such that delivered mass is optimized while the trajectory is still forced to be robust. This is referred to here as the robust-constrained problem. Alternatively, there may be some desired lower-bound constraint on delivered mass (e.g., spacecraft design has been mostly finalized), and a mission designer may wish to maximize γ for a trajectory. This is referred to here as the robust-optimal problem.

Current methods of accounting for these robustness considerations are frequently ad hoc, rely on the experience and intuition of the mission designer, and can require significant time expenditure on the part of the designer. Further, while current methods may enable a designer to meet the constraint in Eq. (14), such methods may not enable the simultaneous optimization of delivered mass or other variables while also meeting the constraint.

In the present work a method is developed that can both optimize spacecraft delivered mass with a constraint on γ as well as optimize γ with constraint on delivered mass. The method can also be automated to enable more efficient exploration of the engineering trade space and Pareto front of solutions. While the present work is focused on the lower bound of β throughout a trajectory, the method developed here is not incompatible with alternative criteria for β or alternative optimization objectives.

III. Virtual Swarm Method

The central idea of the virtual swarm method is to simultaneously optimize a nominal spacecraft trajectory along with its recovery trajectories after simulated forced shutdown events. Each “virtual” spacecraft provides a method of fixing or optimizing β at discrete points along a nominal trajectory. Simultaneously optimizing the nominal trajectory along with its recovery trajectories enables the nominal trajectory to be reshaped in conjunction with values related to the recovery trajectory (e.g., recovery trajectory forced coast time, delivered mass).

A virtual spacecraft is one that has a fixed “spawn point” where its state $\mathbf{x}_0 = \mathbf{x}_{\text{nominal}}(t_{\text{spawn}})$, where t_{spawn} is the time along the nominal trajectory the virtual spacecraft is spawned. Each spawn point can be thought of as a discrete control point for MTRM along the nominal trajectory. The virtual spacecraft themselves can be equivalently thought of as spacecraft that have the same control history as the nominal spacecraft from $t = t_0$ to $t = t_{\text{spawn}}$. At t_{spawn} , the virtual spacecraft has a forced shutdown/coasting period

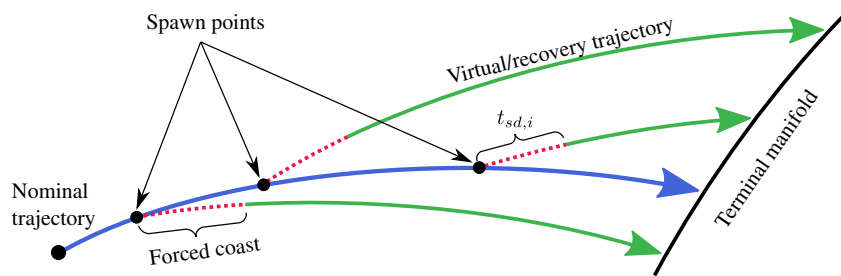


Fig. 1 Illustration of the virtual swarm method.

of t_{sd} days, after which it can resume thrusting with controls and other decision variables (e.g., TOF, m_f) that are independent of the nominal spacecraft trajectory. The nominal and the virtual spacecraft must all satisfy the specified terminal constraints. An arbitrary number N of these recovery trajectories can be added, and here are indexed by the variable i . Care must be exercised when selecting the number N and spawn times of virtual spacecraft; this is discussed later in this section. At each spawn point, a virtual spacecraft has a forced shutdown (coast) time of $t_{sd,i}$, where $i = 1, 2, \dots, N$. Let $i = 0$ refer to the nominal spacecraft. This method is illustrated in Fig. 1.

For the robust-constrained reference problem, the optimization is formulated as the minimization of

$$J = -m_{f,\min} \quad (15)$$

subject to

$$t_{sd,i} = \gamma_{\min} \quad i = 1, 2, \dots, N \quad (16)$$

$$m_{f,i} \geq m_{f,\min} \quad \forall i \quad (17)$$

$$t_{f,\min} \leq t_{f,i} \leq t_{f,\max} \quad \forall i \quad (18)$$

While the nominal spacecraft still has the initial constraints of Eq. (9), each virtual spacecraft has the aforementioned initial constraint

$$\mathbf{x}_{0,i} = \mathbf{x}_{\text{nominal}}(t_{\text{spawn},i}) \quad i = 1, 2, \dots, N \quad (19)$$

with

$$t_{0,i} = t_{\text{spawn},i} \quad i = 1, 2, \dots, N \quad (20)$$

In short, this formulation fixes γ by assigning that value to each $t_{sd,i}$ and then optimizes the lower bound on delivered mass to ensure that the worst-case delivered mass is the highest possible. The lower bound is the focus because the spacecraft design must be able to account for worst-case performance.

The resulting value of $m_{f,\min}$ represents the worst-case delivered mass. While $m_{f,i}$ is a decision variable for all spacecraft, only the limiting case $m_{f,i} = m_{f,\min}$ represents a maximum delivered mass for that spacecraft in the result of the optimization problem. The other virtual spacecraft can be separately optimized to find the maximum delivered mass for each case.

For the robust-optimal problem, the optimization is formulated as the minimization of

$$J = -t_{sd,\min} \quad (21)$$

subject to

$$t_{sd,i} \geq t_{sd,\min} \quad i = 1, 2, \dots, N \quad (22)$$

$$m_{f,i} \geq m_{f,\min} \quad \forall i \quad (23)$$

$$t_{f,\min} \leq t_{f,i} \leq t_{f,\max} \quad \forall i \quad (24)$$

Again, the nominal spacecraft still has the initial constraints of Eq. (9), and each virtual spacecraft has the aforementioned initial constraint in Eq. (19).

This formulation fixes a lower bound on delivered mass $m_{f,i}$, ensuring that the nominal and all recovery trajectories deliver sufficient mass while ensuring that the lower bound value γ is as large as possible. This ensures that the weakest point in the trajectory in terms of β has the largest possible value. Similar to the robust-constrained reference problem, $\beta_i = t_{sd,i}$ only for the trajectory where $t_{sd,i} = t_{sd,\min}$. For other trajectories, $\beta_i \geq t_{sd,i}$, and β for each point can be found by re-solving for it at each point along the nominal trajectory. If a sufficient number N of virtual spacecraft are used in proper spawn locations, then $t_{sd,\min} = \gamma$.

Note that both the robust-optimal and robust-constrained problems can both be used to uncover the same solution. Consider a robust-constrained problem where an optimal $m_{f,\min}$ is found with fixed $t_{sd,i}$. Using that same $m_{f,\min}$ as a fixed constraint value in the robust-optimal problem and optimizing $t_{sd,\min}$ will result in $t_{sd,\min} = t_{sd,i}$ from the robust-constrained problem. Solving both problems and ensuring that the solutions match as expected can increase the confidence that a solution has been properly found.

The number and placement of the N virtual spacecraft is extremely important in ensuring that the constraint of γ is properly met or in ensuring that the lower bound γ is properly optimized. Theoretically, an infinite number of virtual spacecraft could be added such that one is spawned at each time of the nominal trajectory. A large number of equally spaced virtual spacecraft could be added throughout the nominal trajectory in lieu of constraining every point. However, a large number of virtual spacecraft will require a large number of decision variables that may slow efforts to numerically converge on an optimal solution. Alternatively, virtual spacecraft can be iteratively added as constraint violations are found. First, N_{init} initial virtual spacecraft can be placed at locations in the nominal trajectory where it is expected to be sensitive to missed thrust events (e.g., right before arrival at the final target, right before a gravity assist). Then, more virtual spacecraft can be iteratively added as shown in Algorithm 1.

Algorithm 1: Iterative scheme of adding virtual spacecraft

Result: Optimized nominal and recovery trajectories

add N_{init} virtual spacecraft;

solve Eq. (15) or Eq. (21);

evaluate β at each point along the trajectory, calculate γ ;

while $\gamma < \gamma_{\min}$ **do**

add N_w virtual spacecraft, one at each the N_w worst violations points where $\beta < \gamma_{\min}$;

solve Eq. (15) or Eq. (21);

evaluate β at each point along the trajectory, calculate γ ;

end

This is the same scheme used for many optimization problems where explicitly implementing all constraints can be computationally expensive. It does not, however, dilute the result of the solution; constraints need not be explicitly enforced as long as they are not violated in the final solution. Algorithm 1 can be automated or done manually by the mission designer.

IV. Low-Thrust Trajectory Transcription

While the virtual swarm method can be implemented numerically using a number of different trajectory transcriptions, in the present work low-thrust trajectories will be described using a multiple phase version of the well-known Sims–Flanagan direct transcription of the low-thrust trajectory optimization problem [10,11]. This transcription is well suited to global optimization problems and enables faster convergence on local optima for low-thrust trajectories. While it is relatively low fidelity, it enables efficient exploration of wide search spaces and provides initial guesses for higher-fidelity methods that are not as able to explore wide regions of state space. This is especially important for efficient exploration of virtual swarm solutions because they can be very high-dimensional problems.

In the Sims–Flanagan transcription, low-thrust control is modeled as a series of impulsive maneuvers spaced throughout a trajectory in discrete segments as seen in Fig. 2. In this work, a single phase is represented by a Sims–Flanagan transcribed trajectory with forward shooting from the initial body, and backward shooting from the final body. The forward and backward parts of the trajectories are constrained to match at the midpoint of the phase, and Keplerian dynamics are used to propagate the trajectory between impulses. The universal variable formulation from Bate et al. [12] is used for the Keplerian propagation. Multiple phases can be connected to give a trajectory with one or more gravity assists; a single-phase mission here goes directly from the initial body to the final body. In a multiphase mission, the final state of one phase is the initial state of the next phase. Phases are indexed from $k = 1, 2, \dots, N_{ph}$. In each phase, there are τ_k impulses numbered from $j = 1, 2, \dots, \tau_k$, each of which is centered in the segment with the same number identifier. The magnitude of each impulsive ΔV is limited in proportion to how much ΔV a low-thrust engine may be able to provide in the time period of the segment. The method is summarized for a single spacecraft in Fig. 2. The total number of impulses τ is

$$\tau = \sum_{k=1}^{N_{ph}} \tau_k \quad (25)$$

and the time period of each segment is

$$q = \frac{t_f - t_0}{\tau} \quad (26)$$

Because continuous periods of low-thrust acceleration are approximated by impulsive ΔV s, virtual spacecraft are only spawned at the start time of a segment. For similar reasons, the maximum number of virtual spacecraft spawned is $N_{max} = \tau$. For $N > \tau$, virtual spacecraft will spawn from points inside segments, where thrust is theoretically being applied but is not actually apparent until the impulse in the center of the segment. These points inside segments have larger differences from a higher-fidelity, finite burn approximation of the

low-thrust trajectory than points at the start of segments. For greater control and fidelity, τ can be increased.

Figure 3 shows a detailed view of how a single recovery trajectory is handled in conjunction with the nominal trajectory. A virtual spacecraft can be added to the start of any segment, and is uniquely described by the nominal phase and segment it spawns from. While in general virtual trajectories immediately diverge from the nominal trajectory at the spawn point (see Fig. 1), when applied in a Sims–Flanagan transcription the virtual trajectory does not deviate from the nominal until one of the two spacecraft applies an impulse. In the notional example shown in Fig. 3, the nominal and virtual trajectories do not deviate until the nominal spacecraft applies an impulse at point B. Alternatively, the deviation could be caused by the first impulse applied by the virtual spacecraft (point C), in which case the full effect of the missed thrust event would not impact the virtual spacecraft trajectory. This occurs when

$$t_{sd,i} + \frac{q_i}{2} < \frac{q_0}{2} \quad (27)$$

where the time length of a single segment of the nominal trajectory is q_0 and the time length of a single segment of the virtual trajectory is q_i . The times q_0 and q_i are frequently of similar magnitude, and usually $t_{sd,i} > 0$ is desired; both factors lead to the condition in Eq. (27) not holding.

For a virtual spacecraft spawned at segment j of phase k , the total number of impulses $\tau_{virtual,j,k}$ it has for its mission lifetime is

$$\tau_{virtual,j,k} = \max\{(\tau_k - j + 1), \tau_{k,min}\} + \sum_{k+1}^{N_{ph}} \tau_k \quad (28)$$

The virtual spacecraft typically gets the same number of impulses as remain in the phase for the nominal trajectory, including the spawn segment, and has the full number of impulses the nominal spacecraft has in subsequent phases. A lower bound $\tau_{k,min}$ on the number of impulses in the spawn phase is included, however, so that virtual spacecraft always have some minimum control authority to change thrust direction. In this work, $\tau_{k,min} = 5$ is generally used, though Fig. 3 shows $\tau_{k,min} \leq 3$.

The transcription introduces a number of constraints. These constraints include explicit matchpoint constraints to ensure that the trajectory is continuous and explicit control magnitude constraints (see [11] for details). Note that the terminal constraint in Eq. (4) and the position component of the initial constraint in Eq. (9) are implicitly satisfied by the transcription and thus do not need to be explicitly stated in the problem set up for the nonlinear programming (NLP) solver.

Launch auxiliary decision variables are shown in Table 1. The launch C_3 and maximum allowable m_0 value are related by a function

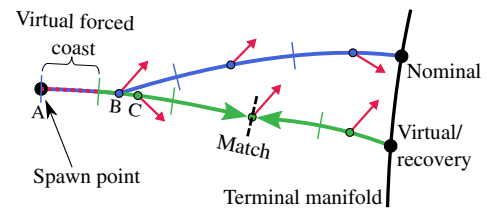


Fig. 3 Detail on a single virtual/recovery trajectory.

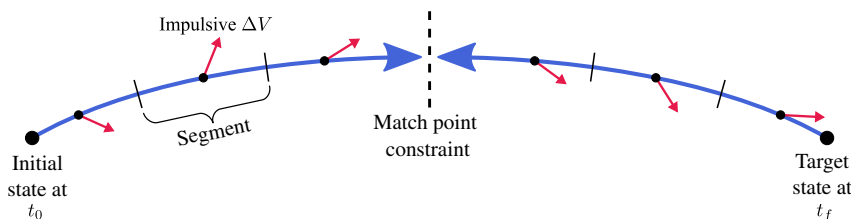


Fig. 2 Single phase represented with a Sims–Flanagan transcription.

Table 1 Launch auxiliary decision variables

Variable	Description
C_3	Characteristic energy
RLA	Right ascension of launch asymptote
DLA	Declination of launch asymptote
$\eta_{\text{underload}}$	Launch mass underload factor

Table 2 Robust-constrained reference problem decision variables

Variable	Description	Number of values
t_0	Initial time at start of each phase	$N + 1$
TOF	Time of flight from t_0 to t_f	$N + 1$
\mathbf{u}	Control vector components	$3\tau + \sum_{k=1}^{N_{\text{ph}}} \sum_{j=1}^{\tau_k} 3\nu_{j,k} \tau_{\text{virtual},j,k}$
m_f	Final mass at end of each phase	$N + 1$
$m_{f,\text{min}}$	Minimum delivered mass constraint	1

$m_{0,\text{max}} = l(C_3)$. Here, a polynomial fit to publicly available launcher data provided by NASA Launch Services Program⁸ is used to model that function. The underload factor $\eta_{\text{underload}} \in (0, 1]$ is used when the initial mass is not fixed, and is instead based on the maximum possible payload mass for the selected launcher and C_3 value. When the underload factor is used, $m_0 = \eta_{\text{underload}} l(C_3)$.

For the robust-constrained reference problem, the additional decision variables are shown in Table 2. The number of entries contributed to the decision vector for each variable is listed in the “Number of Values” column. To express the number of decision variables, a binary variable $\nu_{j,k} \in \{0, 1\}$ is used to express if a virtual spacecraft is spawned at segment j in phase k of the nominal trajectory. If $\nu_{j,k} = 1$, a virtual spacecraft is spawned at that point. The binary variable relates to the total number of virtual spacecraft N with the relation

$$N = \sum_{k=1}^{N_{\text{ph}}} \sum_{j=1}^{\tau_k} \nu_{j,k} \quad (29)$$

For the robust-optimal problem, the additional decision variables are shown in Table 3.

Given the discrete optimization variables and constraints from the transcription, the resulting NLP problem is solved here using the commercial optimization package Sparse Nonlinear OPTimizer (SNOPT) [13]. While analytic partial derivatives of objectives and constraints with respect to decision variables are available in prior work [11,14], for fast development the tool created for this analysis (named the N Spacecraft Trajectory Optimizer, or NSTOP) uses automatic differentiation [15,16] and all functionality is written in the Julia programming language. Automatic differentiation provides exact, machine precision partial derivatives without user specification of analytic partials. However, this comes at the cost of slower run time speed as compared with using analytic partials only. The NLP solver is augmented with monotonic basin hopping in an outer loop for a stochastic search, similar to methods previously used [14,17,18]. The stochastic search is parallelized by running multiple processes with a stochastic search at the same time. Each process shares its best result with the other processes, so all workers use the current consensus best solution as a starting point for continued stochastic searching. Basin hopping alone enables convergence on a solution even with a very poor initial guess, but the speed of the process is further improved with parallelization. The use of a stochastic search in general also gives a measure of ability to find an approximate global optimum as opposed to only being able to find a local optimum with the NLP solver alone. NSTOP has been validated against the open-source version of NASA Goddard’s Evolutionary Mission Trajectory Generator (EMTG) [19]

Table 3 Robust-optimal problem decision variables

Variable	Description	Number of values
t_0	Initial time at start of each phase	$N + 1$
TOF	Time of flight from t_0 to t_f	$N + 1$
\mathbf{u}	Control vector components	$3\tau + \sum_{k=1}^{N_{\text{ph}}} \sum_{j=1}^{\tau_k} 3\nu_{j,k} \tau_{\text{virtual},j,k}$
m_f	Final mass at end of each phase	$N + 1$
$t_{\text{sd},i}$	Coast times	N
$t_{\text{sd},\text{min}}$	Minimum coast time	1

and produces nearly identical optimal trajectories when the same problem is posed in both tools.

V. Examples

All examples here use the same polynomial XR-5 Hall thruster model as in earlier work by Laipert and Longuski [1]. Maximum thrust and mass flow rates are estimated by polynomials that are functions of available power. The thrust function is

$$T(P) = (-8.597 + 77.34P - 2.119P^2 - 1.151P^3 + 0.1739P^4) \times 10^{-3} \quad (30)$$

where T is thrust in newtons and P is power in kilowatts limited to the range $0.302 \leq P \leq 4.839$. The mass flow function is

$$\dot{m}(P) = (3.524 + 68.48P - 16.32P^2 + 2.351P^3 - 0.1195P^4) \times 10^{-7} \quad (31)$$

where \dot{m} is the mass flow rate in kg/s. Two thrusters are used on the spacecraft at a 95% duty cycle, and the power per thruster is limited to the range of 0.302–4.839 kW. The power system modeling, thruster switching (maximum number), and thruster smoothing logic used here are described by Ellison et al. [14], with solar array coefficients taken from the example used by Laipert and Longuski [1]. The solar array coefficients used result in the power expression

$$P(r) = \frac{P_0}{r^2} \left(\frac{1.321 - (0.108/r) - (0.117/r^2)}{1 + 0.108r - 0.013r^2} \right) \quad (32)$$

where r is the spacecraft distance from the sun in astronomical units and P_0 is the power available when the spacecraft is at a distance of 1 AU from the sun. The Falcon 9 autonomous spacecraft drone ship model is used for launch vehicle performance and constraints; the polynomial relating C_3 (km^2/s^2) to the maximum possible launch mass (kg) is

$$m_{0,\text{max}}(C_3) = 0.7226C_3^2 - 116.14C_3 + 3310.8 \quad (33)$$

and is valid for $C_3 \in [0, 10]$ km^2/s^2 . In all examples a mandatory 30-day coasting period immediately after launch is enforced. Thus, any virtual spacecraft are only spawned after that initial coast, and evaluation of MTRM occurs afterward. States of target celestial bodies are approximated using spline fits to ephemeris data provided by JPL and the SPICE toolkit.

A. Evaluating Missed Thrust Recovery Margin for a Nominal Trajectory

Given a trajectory transcription, now the optimization problem of evaluating β in Eq. (10) can be solved. Note that in evaluating β , the initial state \mathbf{x}_0 is fixed. In this case, \mathbf{x}_0 is generated by selecting specific points along the nominal trajectory. Evaluating β within a Sims–Flanagan segment has limited utility, because in each segment the impulsive ΔV is a surrogate for distributed thrusting throughout the segment. For this reason, nonphysical discontinuities in β are found when it is evaluated within a segment. Instead, β is only

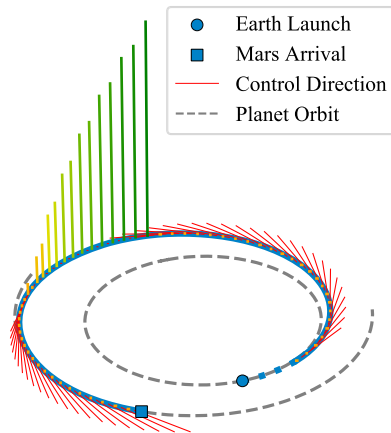
⁸<https://elverf.ksc.nasa.gov/Pages/Query.aspx>.

evaluated at the start points of segments in order to more accurately approximate the value of MTRM along the trajectory. As in the nominal case, a larger number of Sims–Flanagan segments can be used when greater accuracy in modeling the true low-thrust problem is desired. A “nonpermissive” example of evaluating the MTRM along a low-thrust mass optimal trajectory from Earth to Mars can be seen in Fig. 4. Earth launch occurs at the circular point in Fig. 4a, a mandatory 30-day postlaunch coast is shown by a blue dotted line, impulsive ΔV vectors are shown with red lines, and arrival at Mars is shown with a blue square. Vertical lines indicate the MTRM β at the start of each segment, with taller, green lines indicating larger β values and shorter, orange lines indicating smaller β values. Numeric values of β can be seen more clearly in Fig. 4b.

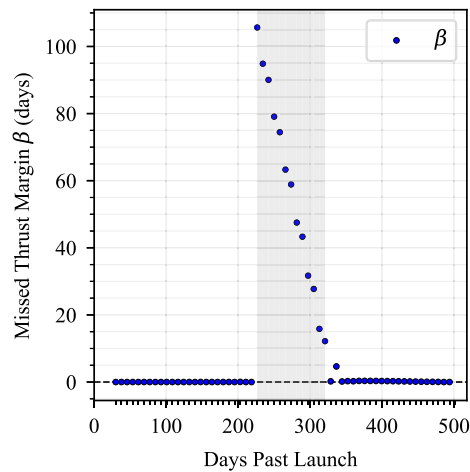
In this nonpermissive example, β is optimized with the same limit $t_{f,\max}$ as the nominal trajectory and had the constraint that $m_{f,\text{recovery}} \geq m_{f,\text{nominal}}$. While in this specific case, the nominal trajectory had $t_f < t_{f,\max}$ by 11 days, the additional time is not enough to allow $\beta > 0$ at all points along the nominal trajectory. Indeed, one indicator that the nominal trajectory is truly mass optimal is that there is at least one point along the trajectory where it cannot withstand any

amount of forced coasting time without inducing an additional mass penalty in the recovery trajectory. From 0.42 TOF ($\beta = 106$ days) to 0.62 TOF ($\beta = 12.17$ days) the spacecraft is coasting, so forced coasts starting at those times overlap with planned nominal coasts and the decline in MTRM should be linear. In this evaluation the decline is close to, but not exactly, linear due to the discrete number of impulses used to approximate the post-forced-shutdown trajectory.

A more permissive example of evaluating β along a low-thrust mass optimal trajectory from Earth to Mars can be seen in Fig. 5. In this case, the same time bound $t_{f,\max}$ from the nominal case is used, but the lower bound on the recovery delivered mass is $m_{f,\text{recovery}} \geq m_{f,\text{nominal}} - 30$ kg. With more permissive bounds, the spacecraft can withstand longer forced shutdown periods at each point along its nominal trajectory, at the expense of using more propellant. This method could be used in a similar manner as in previous work [1] to evaluate the robustness of a nominal trajectory and determine how much additional propellant margin is needed to make a given nominal trajectory robust. However, note that toward the end of this example trajectory very little robustness is gained with 30 kg of additional fuel use; more must be done to gain robustness.

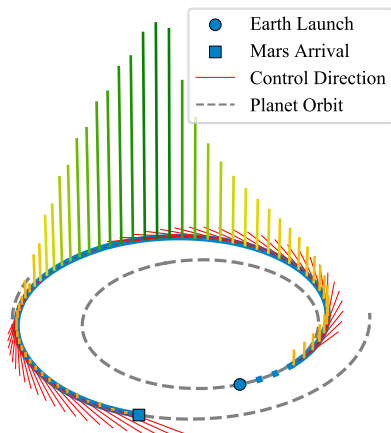


a) Vertical lines are drawn proportional to how much MTRM there is at each point in the trajectory; larger lines indicate more MTRM

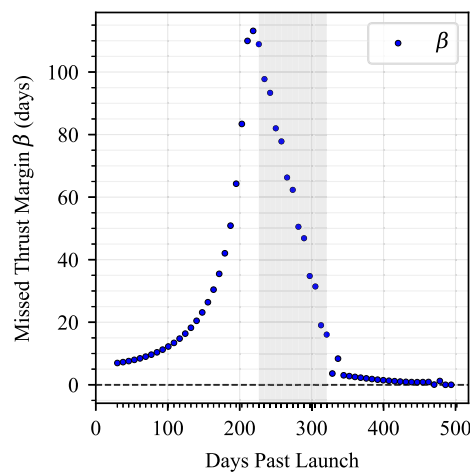


b) Shaded region indicates a coasting period

Fig. 4 Missed thrust recovery margin along an Earth–Mars low-thrust transfer.



a) Vertical lines are drawn proportional to how much MTRM there is at each point in the trajectory; larger lines indicate more MTRM



b) Shaded region indicates a coasting period

Fig. 5 Permissive missed thrust recovery margin along an Earth–Mars low-thrust transfer.

B. Earth–Mars Transfer

1. Single Example

The first example of solving the robust-constrained reference problem shown here is an Earth–Mars transfer using 30 impulses in the nominal spacecraft's trajectory transcription. User-specified values for the reference problem without robustness considerations compared with user-specified values for the related robust-constrained problem are given in Table 4, whereas details of the optimal results are given in Table 5. In this case, the robust-constrained reference problem solved here enforces $\gamma_{\min} = 20$ days by setting a fixed forced shutdown time $t_{sd,i}$ for all virtual spacecraft. The maximum arrival date is set to 25 days later than in the reference problem, and the lower bound on delivered mass for all spacecraft in the swarm is optimized. To use a common measure, the propellant margin κ needed for the robust trajectory can be calculated as

$$\kappa = \frac{m_{f,\text{mass optimal only}} - m_{f,\text{min,swarm}}}{m_{\text{propellant, mass optimal only}}} \quad (34)$$

Here the propellant margin is how much additional propellant, as compared with the mass-optimal-only solution, is needed to recover from the worst-case missed thrust event. However, this requires that the robust-constrained trajectory be flown. If the mass-optimal-only trajectory is flown with the additional propellant margin, it cannot be guaranteed to recover from the worst-case missed thrust event with the given propellant margin.

In the robust-constrained solution, the optimizer has reduced the launch mass from 3038 kg in the reference problem to 2977 kg in the robust-constrained problem. This has the effect of enabling a higher C_3 and giving the low-thrust system more control over acceleration due to the lighter initial mass. The robust-constrained solution, however, has little qualitative difference from the reference solution, as seen in Fig. 6. Figure 6 shows the reference solution in orange and the robust-constrained solution in blue. Figure 6b shows the distance of both solutions from the sun over time, with their respective time limits on t_0 and t_f shown with vertical lines. Although the robust-constrained spacecraft arrives two days later than the reference

spacecraft, it still arrives considerably before its $t_{f,\max}$. This allows virtual/recovery trajectories spawned later in the nominal trajectory to have time to arrive at Mars within the allowed limits.

Figure 7b demonstrates how the control profile of the robust-constrained trajectory shifts as compared with the mass-optimal reference trajectory. The mass-optimal trajectory for the most part has the familiar bang-bang control structure; the short segment where the thrust is neither full nor zero is likely due to the approximate nature of the Sims–Flanagan transcription. The robust trajectory coasting time is comparatively shorter and earlier, and thrusting is no longer at 100% in the lead up to arrival at the target. Instead, the thrust level steps down as it nears arrival and reaches 50% immediately before t_f . These changes make the trajectory much less sensitive to missed thrust events. Figure 7a shows how the control profiles of the nominal and virtual spacecraft evolve over time. Figure 8 shows individual delivered mass and arrival dates for both the nominal trajectory and each virtual trajectory in the robust-constrained case. Note that nearly all virtual spacecraft spawned in the second half of the trajectory have arrival dates and arrival mass that match the limiting value. In the case of delivered mass shown in Fig. 8a, the values shown do not necessarily represent the maximum possible arrival mass for each recovery trajectory. This is because only the lower bound for the whole swarm is being optimized.

Finally, Fig. 9 shows β evaluated at each segment start point along the nominal robust-constrained trajectory. Evaluated values of β are shown as blue circles, whereas the value of t_{sd} for each virtual spacecraft included in the swarm is shown with red squares. Because each value of $\beta \geq t_{sd}$, the number and placement of virtual spacecraft in this case was sufficient to constrain $\gamma \geq \gamma_{\min} = t_{sd}$.

2. Pareto Front

To explore the optimal tradeoffs between maximum arrival date, γ , and delivered mass, the robust-constrained problem can be solved many times with different fixed values of $t_{f,\max}$ and t_{sd} . This will provide the Pareto front for this trade space. The same information could be gathered by optimizing γ_{\min} and constraining $m_{f,\min}$ and $t_{f,\max}$, though in this example $m_{f,\min}$ is the optimization variable. Other parameters of interest could also be varied to give an understanding of how different parameters impact a trajectory. To give an easy reference for how robust trajectories compare to the mass-optimal-only reference trajectory, the Pareto fronts in Fig. 10 are shown in terms of propellant margin κ instead of in terms of delivered mass. Each point represents a single solution to the robust-constrained problem with different constraints. Each curve has a corresponding γ , which is enforced in each solution through fixed t_{sd} in numerous virtual spacecraft. For each solution, β has been calculated throughout the trajectory to ensure that $\gamma = t_{sd}$. The x axis describes how many days past the reference problem $t_{f,\max}$ that the swarm spacecraft were allowed to arrive at the target. The point solution described in more detail in the previous section is at $\gamma = 20$ days, with an allowable arrival date of 25 days past the reference problem and a propellant margin of 10.2%.

The propellant margin needed if the spacecraft is not allowed to arrive any later than the reference $t_{f,\max}$ can be quite high, especially as γ is increased to 25 days and beyond. The propellant margin drops off relatively quickly as $t_{f,\max}$ is increased from 0, though it does level off with changes from $t_{f,\max} = +50$ to $t_{f,\max} = +100$ being mostly minor. This is likely because a local minimum is found at a certain arrival date due to the relative positions of Earth and Mars at launch. However, as γ is increased, greater benefits in propellant margin κ are seen as $t_{f,\max}$ is increased.

C. Earth–Mars–Psyche Transfer

1. Single Example

The same procedure can be applied to a trajectory that includes a gravity assist. In this example, an Earth–Mars–Psyche mission is used to demonstrate the technique. This trajectory is unrelated to the NASA Psyche mission, and although it is similar to the gravity assist in Laipert and Longuski [1], the reference trajectory is different.

Table 4 Robust-constrained Earth–Mars transfer problem: user specified and constant values

Parameter	Reference problem	Robust-constrained
Nominal SC number of impulses	30	30
Power available at 1AU, kW	10	10
t_0 bounds (lower, upper)	Aug. 11, 2024–Oct. 10, 2024	Aug. 11, 2024–Oct. 10, 2024
t_f bounds (lower, upper)	Nov. 7, 2025–Jan. 6, 2026	Nov. 7, 2025–Jan. 31, 2026
Virtual spacecraft spawn segments	—	1–4, 18–30
Virtual spacecraft t_{sd} , days	—	20
Number of decision variables	97	794
Number of constraints	38	424

Table 5 Robust-constrained Earth–Mars transfer problem: optimizer-selected values

Parameter	Reference problem	Robust-constrained
Launch date	Aug. 11, 2024	Aug. 11, 2024
Nominal SC arrival date	Dec. 25, 2025	Dec. 27, 2025
Nominal SC delivered mass, kg	2343	2304
Nominal SC propellant mass, kg	695	673
Launch mass, kg	3038	2977
C_3 , km ² /s ²	2.38	2.92
Optimized $m_{f,\min}$, kg	—	2272
Propellant margin κ , %	—	10

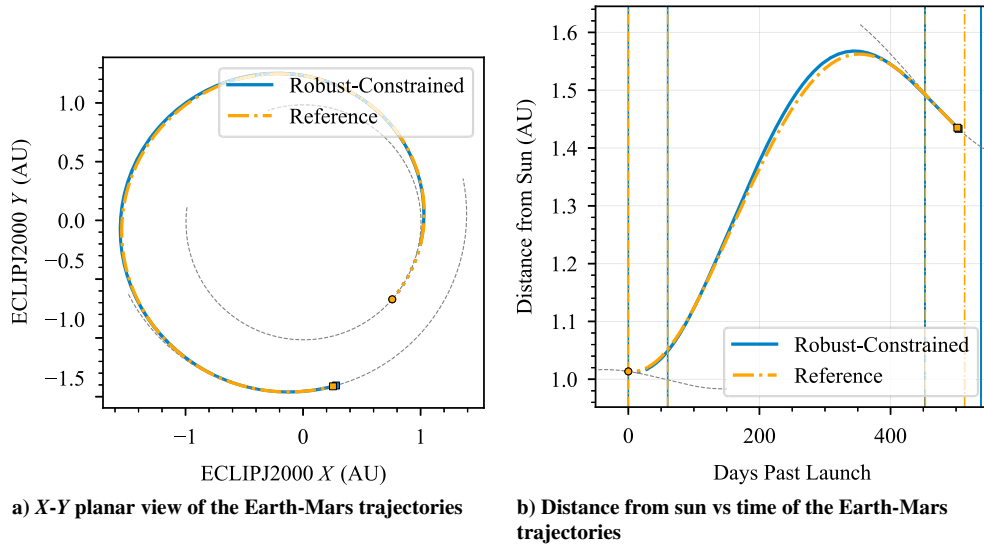


Fig. 6 Reference optimal transfer (orange) vs robust-constrained nominal (blue) Earth-Mars low-thrust transfer. Vertical lines show limits on t_0 and t_f for their corresponding trajectory.

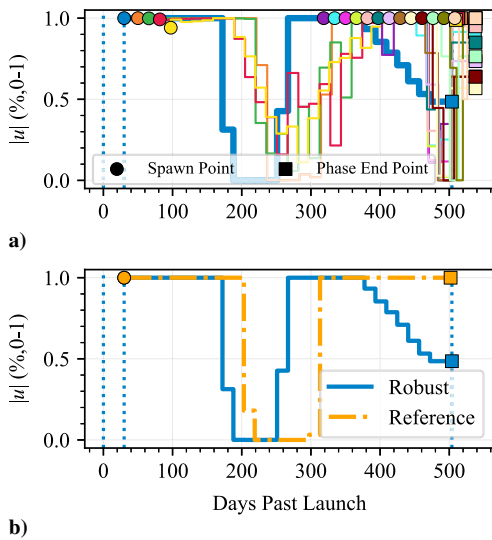


Fig. 7 a) Control history of all swarm (multicolored) and reference (thick blue) spacecraft. b) Control history of reference mass-optimal transfer (orange) vs optimal robust-constrained nominal (blue) Earth-Mars low-thrust transfer.

A 20-day coasting period before arrival at the Mars gravity assist is enforced for all trajectories to allow for navigation and targeting maneuvers to occur. This applies to nominal spacecraft and any virtual spacecraft that spawn before the gravity assist; if a missed thrust event occurs before the gravity assist, the spacecraft must be able to withstand a shutdown of at least γ_{\min} days in addition to the 20-day forced coast before arrival at the gravity assist. It is also possible to allow recovery trajectories to have a shorter pregravity assist coast than the nominal spacecraft to potentially reduce the propellant needed for the worst-case recovery trajectory at the expense of reduced time to perform pregravity assist operations in recovery scenarios. Such a strategy, however, is not used in these results. Details of the problem setup and optimal results for the reference problem without robustness considerations and the robust-constrained problem are given in Tables 6 and 7. The robust-constrained result is allowed to arrive 75 days later than the reference problem, and its lower bound mass $m_{f,\min}$ is optimized with a fixed $t_{sd} = 20$ days for all virtual spacecraft. Note that this robust-constrained solution again has a lower launch mass and higher C_3 than the reference problem, as was found in the Earth-Mars transfer.

In the gravity assist case, as visualized in Fig. 11, qualitative differences between the reference and robust solutions are clearly evident. The nominal gravity assist date has been moved earlier in the robust case, which allows recovery trajectories enough time to make corrections and arrive at the gravity assist before it becomes infeasible. The robust nominal trajectory also remains closer to the sun than the reference trajectory providing more power for thrusting. The control profile shown in Fig. 12 interestingly shows that the robust trajectory does have some thrusting immediately before the gravity assist, whereas the reference trajectory is coasting for a long period of time before its gravity assist. The large spike in applied control immediately before the robust spacecraft's gravity assist is largely applied in the antiveLOCITY direction (see Fig. 11a). This indicates that the control for the robust trajectory at this point is largely being applied to control the timing for the nominal and virtual spacecraft near the gravity assist. In the second phase from Mars to Psyche, the robust trajectory has more control applied earlier in the phase, and again steps the control magnitude down as it approaches its target, much like the Earth-Mars example.

Figure 13 shows the swarm arrival dates and masses for the gravity assist case. Note that in Fig. 13a the virtual spacecraft spawned immediately before the gravity assist and the virtual spacecraft spawned immediately before arrival at Psyche are the limiting factors in the optimization of $m_{f,\min}$. This is in line with our understanding of the problem; the point where the gravity assist occurs is highly sensitive to missed thrust events. Figure 13b shows that several, but not all, spacecraft in the swarm arrive at roughly $t_{f,\max}$. Finally, Fig. 14 shows β along the nominal trajectory to verify that $\gamma = t_{sd}$.

2. Pareto Front and Sensitivity

A Pareto front can be generated for this case using the same method as the Earth-Mars example discussed in Sec. V.B.2. The result is shown in Fig. 15. Note that the lowest propellant margin found in this example is significantly higher than the lowest propellant margin found in the Earth-Mars case seen in Fig. 10. This speaks to the additional sensitivity to missed thrust events introduced by using gravity assists in an interplanetary trajectory. Importantly, individual gravity assist opportunities (i.e., a specific launch date and sequence of flyby locations and dates) will each have varying levels of sensitivity to missed thrust events; these examples are not necessarily representative of how much propellant margin is generally required to ensure trajectories are robust.

To visualize how the sensitivity of the trajectory to missed thrust events changes from the mass-optimal case to the robust constrained case, sensitivity plots are given in Figs. 16–18. In these plots, missed thrust events are simulated at various points along the nominal

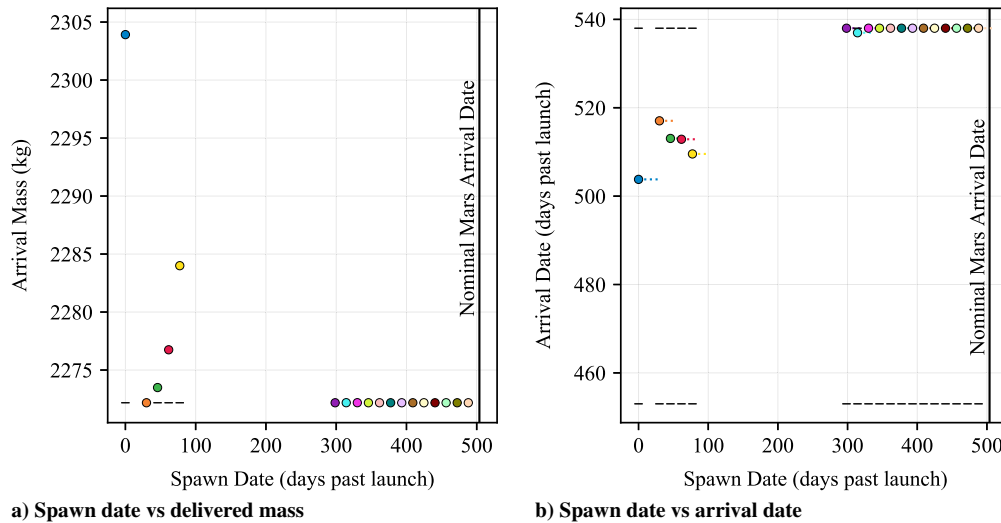


Fig. 8 Results for nominal and virtual spacecraft in the robust-constrained Earth-Mars problem; colors correspond to the virtual spacecraft colors in Fig. 7a.

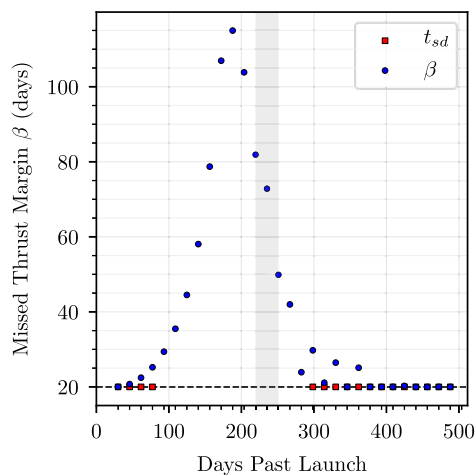


Fig. 9 β values calculated along the robust-constrained Earth-Mars trajectory. The shaded region indicates coasting, and a dashed horizontal line is placed at the desired minimum β value of 20 days.

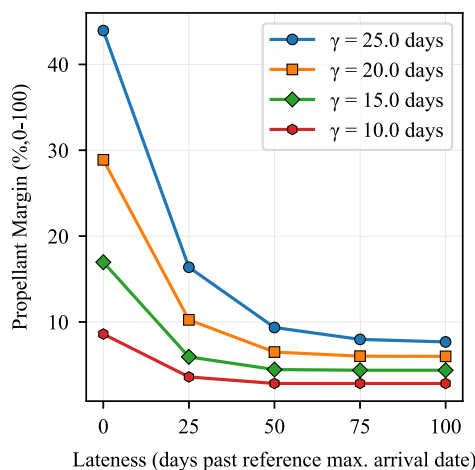


Fig. 10 Earth-Mars Pareto front, γ , $t_{f, \max}$, and propellant margin.

trajectory with varying lengths. After the simulated missed thrust event, a mass-optimal trajectory is generated with the initial state fixed to the post-shutdown state and no constraints on robustness. This gives insight into what the optimal delivered mass will be for

Table 6 Robust-constrained Earth-Mars-Psyche transfer problem: user specified & constant values

Parameter	Reference problem	Robust-constrained
Nominal SC number of impulses	60	60
Power available at 1 AU, kW	20	20
t_0 bounds (lower, upper)	Aug. 5, 2024–Feb. 1, 2025	Aug. 5, 2024–Feb. 1, 2025
t_f bounds (lower, upper)	Oct. 4, 2028–Feb. 11, 2029	Oct. 4, 2028–April 27, 2029
Virtual SC phase 1 (Earth-Mars) spawn segments	—	28, 30
Virtual SC phase 2 (Mars-Psyche) spawn segments	—	1, 21, 23, 25–30
Virtual SC t_{sd} , days	—	20
Number of decision variables	194	680
Number of constraints	78	358

Table 7 Robust-constrained Earth-Mars-Psyche transfer problem: optimizer-selected values

Parameter	Reference problem	Robust-constrained
Launch date	Aug. 5, 2024	Aug. 5, 2024
Nominal SC Mars gravity assist date	May 19, 2026	April 10, 2026
Nominal SC Psyche arrival date	Feb. 11, 2029	April 26, 2029
Nominal SC delivered mass, kg	1891	1580
Nominal SC propellant mass, kg	1333	1523
Launch mass, kg	3224	3103
C_3 , km^2/s^2	0.75	1.81
Optimized $m_{f, \min}$, kg	—	1575
Propellant margin κ , %	—	24

missed thrust events occurring at different points and for different lengths of time. The sensitivity of the Psyche mass-optimal reference trajectory is shown in Fig. 16, where each recovery trajectory is allowed to arrive at Psyche up to 75 days late but the nominal trajectory still arrives at the date given in Table 7. The sensitivity of the 20-day robust and up to 75-day late Earth-Mars-Psyche trajectory discussed in detail in this section is shown in Figs. 17 and 18. While the robust trajectory was only constrained to be 20 days robust, shutdown events of up to 25 days were simulated to understand the sensitivity beyond the constrained robustness. In the contour plots (Figs. 17 and 16), white unshaded space indicates that a feasible

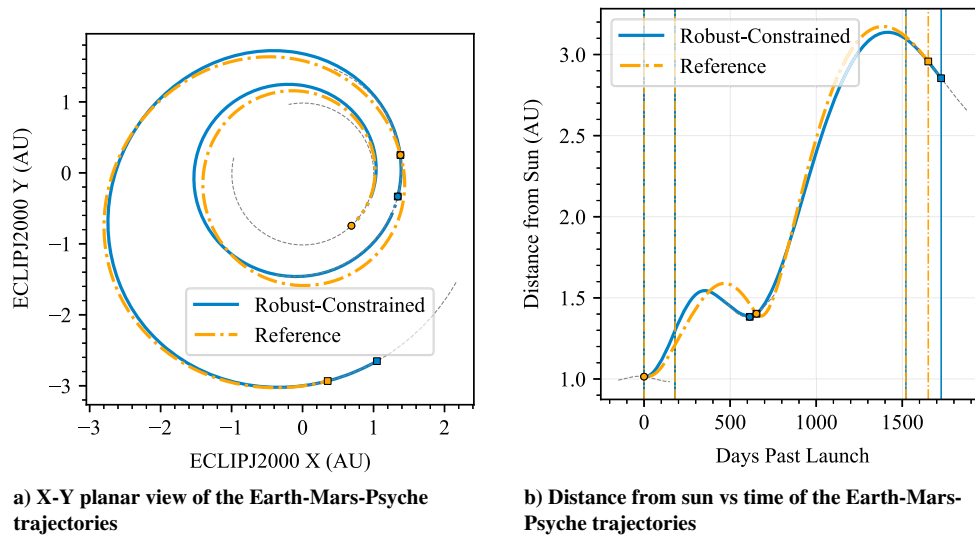


Fig. 11 Reference optimal transfer (orange) vs robust-constrained nominal (blue) Earth-Mars-Psyche low-thrust transfer. Mars gravity assists are marked by square-enclosed circles; vertical lines show limits on t_0 and t_f for their corresponding trajectory.

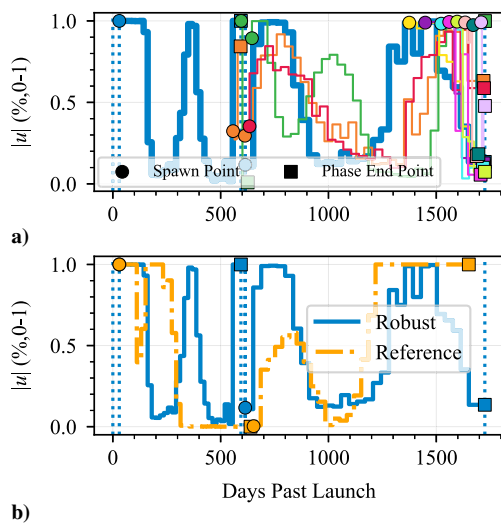


Fig. 12 a) Control history of all swarm spacecraft (thin multicolored) and reference spacecraft (thick blue). b) Control history of reference optimal transfer (orange) vs robust-constrained nominal (blue) Earth-Mars-Psyche low-thrust transfer.

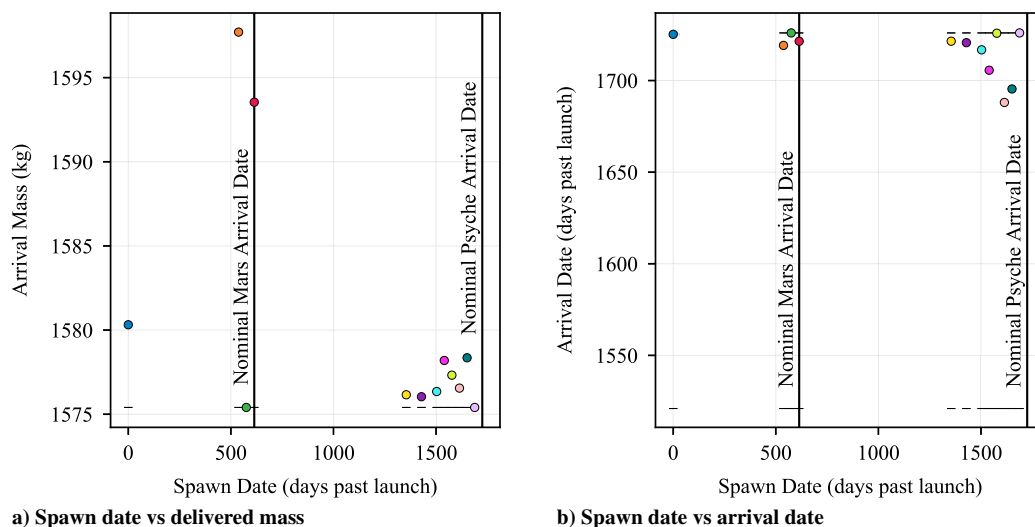


Fig. 13 Results for nominal and virtual spacecraft in the robust-constrained Earth-Mars-Psyche problem; colors correspond to the virtual spacecraft colors in Fig. 12a.

solution was not found, likely indicating that such a trajectory is not possible. The discrete points where optimization problems were solved in order to generate the contours are shown with black points, and the contour colors correspond to the optimized mass delivered to Psyche. Figure 18 shows the same data as Fig. 17 but in a different format to more clearly show which case is limiting.

In the mass-optimal case, Fig. 16, there are very significant regions where a missed thrust event would render the target unreachable, even with significant additional propellant expenditure. Contrast this with the robust-constrained case in Figs. 17 and 18, where nearly all points are feasible though many have significant propellant requirements to arrive at the target. The point immediately before the gravity assist (575 days past launch, the purple steeply downward-sloped line in Fig. 18b) indicates that a missed thrust event just before the gravity assist is the limiting case where the least amount of mass can be delivered to Psyche. A missed thrust event immediately before Psyche arrival is similarly sensitive (but slightly less so) to 20-day missed thrust events than the point before the gravity assist. The significantly steeper slope of the pre-gravity assist delivered mass vs shutdown length curve demonstrates how much more sensitive the point is than others. Information in Fig. 13a similarly shows these two points to be sensitive, but does not show how much mass might be saved if a shorter missed thrust event occurs at those points. Information like this can be

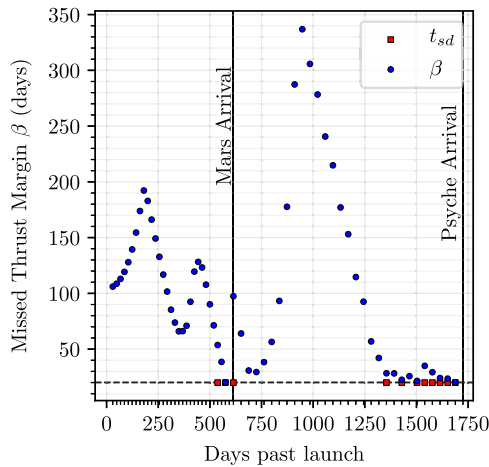


Fig. 14 β values calculated along the robust-constrained Earth–Mars–Psyche trajectory.

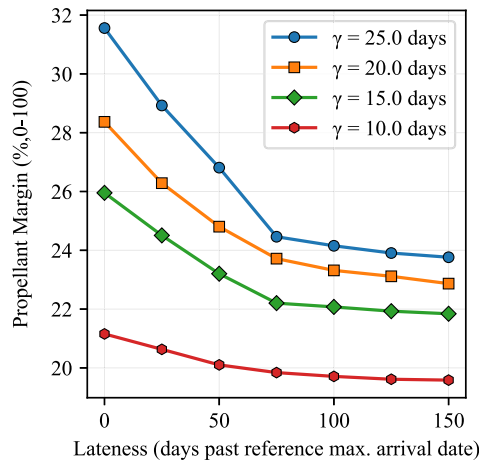


Fig. 15 Earth–Mars–Psyche Pareto front for γ , $t_{f,max}$, and propellant margin.

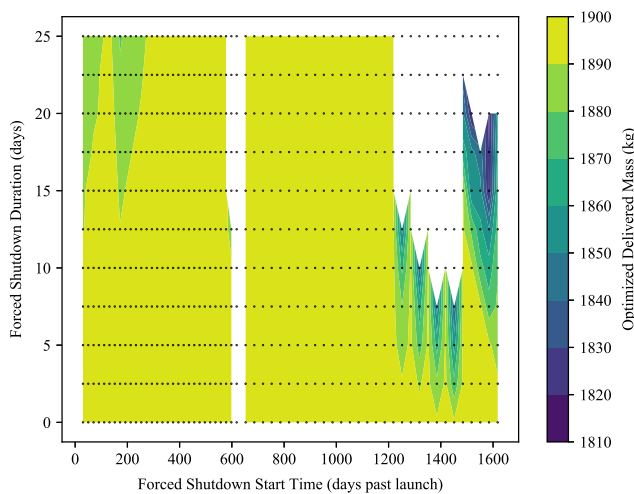


Fig. 16 Sensitivity of a reference mass-optimal Earth–Psyche trajectory to missed thrust events; recovery trajectories can arrive up to 75 days later than the nominal maximum arrival date.

used to decide if a lesser guarantee on allowable missed thrust event length can be used at highly sensitive points in the trajectory in order to reduce the amount of contingency propellant needed to account for missed thrust events. With the robust-constrained virtual swarm method, each point can have different constraints on the missed thrust

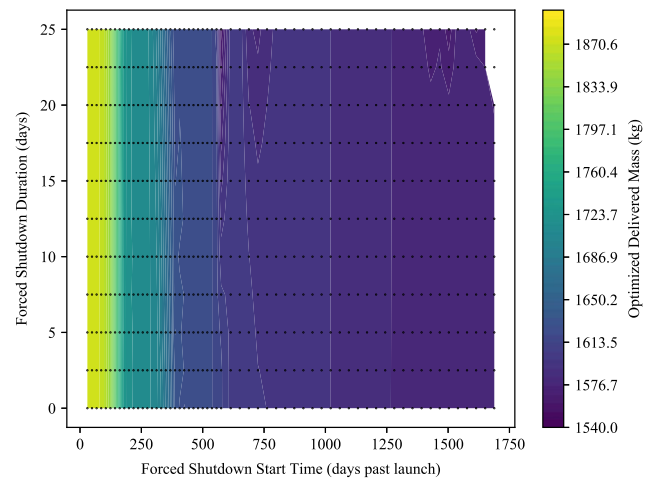


Fig. 17 Sensitivity of a robust-constrained (20-day robustness, up to 75 days late arrival) Earth–Psyche trajectory to missed thrust events.

margin, so in this case the 20 day requirement may be relaxed for the pre-gravity assist point if the resulting decrease in contingency propellant mass is deemed more important. Alternative mitigations for missed thrust events at such points may include standing up additional ground support resources during times when the trajectory is less robust and additional spacecraft testing to ensure nominal operations during less robust points in the trajectory. These system-level decisions can be supported by using the virtual swarm method to understand the optimal propellant tradeoffs with time, robustness, and other parameters of interest.

VI. Discussion

The virtual swarm method that has been developed here is able to describe the optimal tradeoffs between γ , $t_{f,max}$, and delivered mass, without requiring excessive user intervention. Additional tradeoffs can be explored in a similar manner by varying other parameters of interest (e.g., launch vehicle, thruster model) and optimizing the problem. Pareto fronts using the robust-constrained formulation could also consider optimizing other continuous variables not considered here, such as target flyby speed in missions where arrival at the final target is a flyby. The case without a gravity assist, which is less sensitive than the gravity assist case, is particularly amenable to automation. Using a poor initial guess, even with a large number of virtual spacecraft included in the swarm, does not prevent the optimizer from converging on a solution in a relatively short amount of time. Because of this, the method is well suited for trade space exploration. Note that neither the Earth–Mars transfer nor the Psyche transfer examples shown here are specifically selected for amenability to being robust to missed thrust events. Rather, these examples show how a nominal trajectory is transformed once MTRM is constrained along the trajectory. Wide searches using this method can potentially be used to identify specific launch opportunities where the propellant margin penalty for a robust mission as compared with a mass-optimal-only mission is less than the penalties found in the examples presented here.

The gravity assist (multiphase) case presents greater challenges in searching for solutions. While controlling the robustness of the trajectory immediately before arrival at the gravity assist introduces high sensitivities into the problem, controlling this point alone does not require too much computational time. However, in this case each virtual spacecraft can potentially have many more impulses, and consequently many more decision variables, than virtual spacecraft spawned in a single-phase trajectory. The introduction of more decision variables creates a wider search space and, additionally, slows linear algebra operations performed with the Jacobian matrix of partial derivatives of constraints with respect to decision variables. In the present work these linear algebra operations are performed by the NLP solver SNOPT. Further, because automatic differentiation is

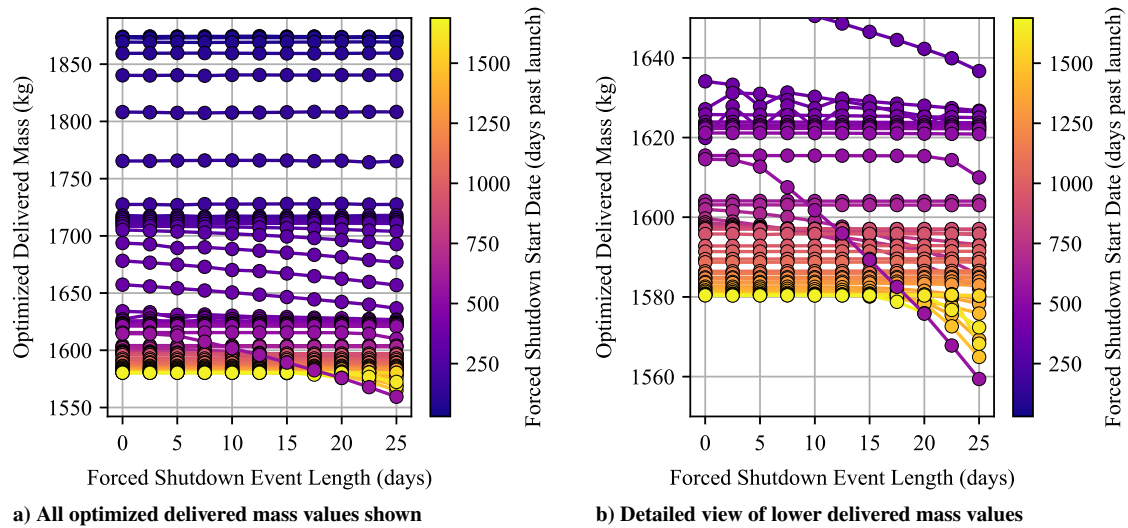


Fig. 18 Sensitivity of a robust-constrained (20-day robustness, up to 75 days late arrival) Earth–Psyche trajectory to missed thrust events, line plot views.

Table 8 Computation times for each example

Case	Number of decision variables	Number of constraints	Constraint evaluation time, ms	Constraint Jacobian evaluation time, ms
Earth–Mars mass optimal	97	38	0.7004	12.0108
Earth–Mars robust-constrained (17 virtual spacecraft)	794	424	9.0954	139.7524
Earth–Mars–Psyche mass optimal	194	78	1.7909	43.3259
Earth–Mars–Psyche robust-constrained (11 virtual spacecraft)	680	358	6.8641	665.0238

being used in this work, the inclusion of more decision variables translates into a greater amount of time spent calculating derivatives.

These problems are not insurmountable and have not prevented automated searches for multiphase trajectories as was done to generate Fig. 15. These barriers do, however, motivate more careful selection of initial guesses to speed up convergence and enable wider searches of the state space as problems become more complex. For more complex problems each virtual spacecraft can be initialized by optimizing its mass or initial coast time based on a static initial state on a nominal reference trajectory. This would provide the optimizer a feasible trajectory as a starting point, instead of it initially having to make the trajectories both feasible and optimal. Complex missions may also benefit from using fewer virtual spacecraft to start with and only adding one or two spacecraft at a time where constraint violations occur (use a small N_w in Algorithm 1). There is a balance to be struck there, however, because although it can take some time to converge on a solution with many virtual spacecraft, there is also a time cost to repeatedly evaluating β along the nominal trajectory. Another strategy is to start with a smaller number of Sims–Flanagan segments to reduce the number of decision variables, then progressively increase the fidelity as solutions are found.

The general computation speed of the virtual swarm method is difficult to describe for a number of reasons. First, varying problems will have varying sensitivities to missed thrust events, and thus will require different numbers of virtual spacecraft to adequately constrain the nominal trajectory. Further, the additional computational complexity introduced by each virtual spacecraft is different depending on its spawn point when using the Sims–Flanagan transcription; virtual spacecraft that spawn earlier will have more control parameters to add to the problem than virtual spacecraft that spawn later. The

stochastic monotonic basin hopping method also makes each optimization run nondeterministic, and sometimes a fortuitous hop occurs early in the process that greatly speeds up how soon a solution is found. Because of this stochasticity, reporting single run times from start to finish are of limited utility. To give a rough idea of how computationally expensive adding the additional virtual spacecraft is, however, Table 8 includes the time to evaluate all problem constraints and constraint Jacobian for each example. The constraints are the most expensive part of each iteration in the NLP solver; the objective function and gradient of the objective function are trivial to compute because the objective is directly a decision variable. Each time is an average over 100 trials of the same computation for a more accurate measurement, and a desktop machine with an Intel Core i7 9700k processor is used for the calculations. Finally, note that the NSTOP tool developed for this work was primarily developed with the intention of proving the virtual swarm concept to be viable, and was not specifically optimized for fast evaluation in large trade studies or large individual problems. There are many areas in which the code could be improved to have a faster run time, such as using analytic derivatives or optimizing the speed of the trajectory propagation function that is run many times as the NLP solver iterates.

Also note that models of varying fidelity can be used in the virtual swarm method. For example, virtual spacecraft could use smaller numbers of Sims–Flanagan segments or analytic approximations for low-thrust dynamics. Conversely, where a higher-fidelity solution is of interest, larger numbers of Sims–Flanagan segments could be used or the low-thrust equations of motion could be integrated using finite-burn thruster modeling. The correct model to use depends largely on the use case and the overall goals of the mission designer, but the virtual swarm method can support a number of different options. The low-fidelity Sims–Flanagan transcription is the focus of the present work to efficiently search a wider space and to provide better initial guesses for higher-fidelity optimization problems.

The virtual swarm technique can also theoretically account for an arbitrary number of sequential missed thrust events. Throughout this work, only a single missed thrust event is accounted for, but it is possible that a recovery trajectory also has a missed thrust event along its nominal path so a second layer of virtual spacecraft could be used to constrain robustness to the second event. The recovery trajectory from the second missed thrust event could also have a third missed thrust event, so a third layer of virtual spacecraft could be spawned, and so on. The NSTOP tool built to do the analysis in this paper can compute solutions with an arbitrary number of missed thrust events, though including too many virtual spacecraft will present challenges to finding optimal and feasible solutions. The exponential growth of the number of virtual spacecraft needed to adequately constrain robustness across multiple missed thrust events will make handling multiple missed thrust events with this method difficult.

One potential solution is to use the method as described in the present work as a way to provide an initial guess for a series of indirect optimization problems for spacecraft in the swarm. An indirect formulation of the optimization problem is more sensitive and difficult to find solutions for, but has fewer decision variables to optimize and thus may scale to larger numbers of spacecraft better. Once indirect solutions are found for the swarm at one level of missed thrust, the next level of missed thrust robustness might be constrained with either direct- or indirect-optimized virtual spacecraft. Alternatively, it may be most prudent to use real world data [2] to guide the placement of virtual spacecraft to ensure that β is constrained at the most important locations along the trajectory. If the robustness need not be constrained at all points along a trajectory, accounting for multiple missed thrust events with virtual spacecraft becomes much easier. The method presented here takes a deterministic view of the robustness problem, which has the benefit of giving concrete recovery trajectories, but makes accounting for an arbitrary number of missed thrust events more difficult. A fully stochastic optimal control approach would be ideal to constrain successful arrival probability (e.g., 99% chance of successfully arriving given a known distribution of multiple possible missed thrust events), but current methods are not able to incorporate missed thrust events into a stochastic optimal control problem with chance constraints.

This method could also be used to analyze the safe-mode robustness of spacecraft using solar sails for propulsion. During a safe mode event for a spacecraft equipped with a solar sail, the thrust level will not necessarily be zero until the spacecraft resumes normal operations. An interesting avenue to explore is what the ideal attitude or attitude control law would be during a safe-mode event, assuming that the spacecraft can maintain some desired attitude instead of simply pointing radially away from the sun. This safe-mode attitude would need to be balanced against other spacecraft health needs such as pointing solar panels toward the sun. Finally, the tools developed for this work easily extend to real spacecraft swarms. Perhaps the most directly applicable real swarm that this method extends to is one in which a low-thrust parent spacecraft launches one or more child spacecraft along its trajectory. These child spacecraft could be destined for the same or different targets than the parent.

VII. Conclusions

The MTRM of a low-thrust spacecraft trajectory can be constrained or optimized using the virtual swarm method developed here. The virtual swarm method simultaneously optimizes a nominal trajectory with its recovery trajectories, enabling the nominal trajectory to be reshaped to account for robustness constraints or objectives. This has the benefit of allowing a mission designer to either optimize delivered mass with path constraints for a minimum missed thrust margin along the way, or it allows a mission designer to optimize worst-case MTRM with a constraint that nominal and recovery trajectories must deliver a minimum amount of mass. The process can be automated to enable efficient search space exploration and the generation of Pareto fronts to give decision makers information about optimal tradeoffs between different objectives.

Acknowledgments

This work was supported by a NASA Space Technology Research Fellowship under grant number 80NSSC17K0147.

References

- [1] Laipert, F. E., and Longuski, J. M., "Automated Missed-Thrust Propellant Margin Analysis for Low-Thrust Trajectories," *Journal of Spacecraft and Rockets*, Vol. 52, No. 4, 2015, pp. 1135–1143. <https://doi.org/10.2514/1.A33264>
- [2] Laipert, F. E., and Imken, T., "A Monte Carlo Approach to Measuring Trajectory Performance Subject to Missed Thrust," *2018 Space Flight Mechanics Meeting*, AIAA Paper 2018-0966, 2002. <https://doi.org/10.2514/6.2018-0966>
- [3] Ozaki, N., Campagnola, S., Funase, R., and Yam, C. H., "Stochastic Differential Dynamic Programming with Unscented Transform for Low-Thrust Trajectory Design," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 377–387. <https://doi.org/10.2514/1.G002367>
- [4] Olympio, J. T., "Designing Robust Low-Thrust Interplanetary Trajectories Subject to One Temporary Engine Failure," *Proceedings of the 20th AAS/AIAA Space Flight Meeting*, Univelt, Inc., Escondido, CA, 2010, pp. 10–171.
- [5] Olympio, J. T., and Yam, C. H., "Deterministic Method for Space Trajectory Design with Mission Margin Constraints," *Proceedings of the 61th International Astronautical Congress*, International Astronautical Federation Paper IAC-10.C1.9.12, Prague, 2010.
- [6] Oh, D., Landau, D., Randolph, T., Timmerman, P., Chase, J., Sims, J., and Kowalkowski, T., "Analysis of System Margins on Deep Space Missions Using Solar Electric Propulsion," *44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, AIAA Paper 2008-5286, 2008. <https://doi.org/10.2514/6.2008-5286>
- [7] Rayman, M. D., and Williams, S. N., "Design of the First Interplanetary Solar Electric Propulsion Mission," *Journal of Spacecraft and Rockets*, Vol. 39, No. 4, 2002, pp. 589–595. <https://doi.org/10.2514/2.3848>
- [8] McCarty, S. L., and Grebow, D. J., "Missed Thrust Analysis and Design for Low Thrust Cislunar Transfers," *2020 AAS/AIAA Astrodynamics Specialist Conference*, AAS Paper 20-535, 2020.
- [9] Rayman, M. D., Fraschetti, T. C., Raymond, C. A., and Russell, C. T., "Coupling of System Resource Margins Through the Use of Electric Propulsion: Implications in Preparing for the Dawn Mission to Ceres and Vesta," *Acta Astronautica*, Vol. 60, No. 10, 2007, pp. 930–938. <http://www.sciencedirect.com/science/article/pii/S0094576506004255>. <https://doi.org/10.1016/j.actaastro.2006.11.012>
- [10] Sims, J., and Flanagan, S., "Preliminary Design of Low-Thrust Interplanetary Missions," *Advances in the Astronautical Sciences*, Vol. 103, Univelt, Inc., Escondido, CA, 1999, pp. 538–548.
- [11] Ellison, D. H., Conway, B. A., Englander, J. A., and Ozimek, M. T., "Analytic Gradient Computation for Bounded-Impulse Trajectory Models Using Two-Sided Shooting," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 7, 2018, pp. 1449–1462. <https://doi.org/10.2514/1.G003077>
- [12] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, Courier Dover Publ., New York, 1971, Chap. 4.
- [13] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, Vol. 47, No. 1, 2005, pp. 99–131. <https://doi.org/10.1137/S0036144504446096>
- [14] Ellison, D. H., Conway, B. A., Englander, J. A., and Ozimek, M. T., "Application and Analysis of Bounded-Impulse Trajectory Models with Analytic Gradients," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 8, 2018, pp. 1700–1714. <https://doi.org/10.2514/1.G003078>
- [15] Revels, J., Lubin, M., and Papamarkou, T., "Forward-Mode Automatic Differentiation in Julia," arXiv:1607.07892 [cs.MS], 2016, <https://arxiv.org/abs/1607.07892>.
- [16] Rall, L. B., and Corliss, G. F., "An Introduction to Automatic Differentiation," *Computational Differentiation: Techniques, Applications, and Tools*, Vol. 89, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996, pp. 1–18.
- [17] Englander, J. A., and Englander, A. C., "Tuning Monotonic Basin Hopping: Improving the Efficiency of Stochastic Search as Applied to Low-Thrust Trajectory Optimization," *24th International Symposium on Space Flight Dynamics*, Johns Hopkins Univ., Applied Physics Lab., Paper GSFC-E-DAA-TN14154, 2014.
- [18] Yam, C., Lorenzo, D., and Izzo, D., "Low-Thrust Trajectory Design as a Constrained Global Optimization Problem," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 225, No. 11, 2011, pp. 1243–1251. <https://doi.org/10.1177/0954410011401686>
- [19] Englander, J. A., Knittel, J. M., Williams, K., Stanbridge, D., and Ellison, D. H., "Validation of a Low-Thrust Mission Design Tool Using Operational Navigation Software," *27th AAS/AIAA Space Flight Meeting*, AAS Paper 17-204, 2017.