

Rapid nonlinear convex guidance via overparameterized monomial coordinates and fundamental solution expansions

Ethan R. Burnett* and Francesco Topputo†

This paper introduces a framework by which the nonlinear trajectory optimization problem is posed as a path-planning problem in a space liberated of dynamics. In this space, general state constraints for continuous and impulsive control problems are encoded as linear constraints on the native overparameterized variables. This framework is enabled by nonlinear expansion in the vicinity of a reference in terms of fundamental solutions and a minimal nonlinear basis of mixed monomials in problem initial conditions. The former can be computed using state transition tensors, differential algebra, or analytic approaches, and the latter is computed analytically. Nonlinear guidance schemes are proposed taking advantage of this framework, including a successive convex programming scheme for delta-V minimizing trajectory optimization. This work enables a stable and highly rapid nonlinear guidance implementation without the need for collocation or real-time integration.

1 Introduction

In the context of modern guidance and control, spacecraft autonomy presents itself as a particularly technically and politically challenging problem. Strict computational and hardware limitations are imposed by the need for radiation-hardened processors and on-board power and mass constraints. Experimentation with autonomous agents is furthermore extremely regulated in comparison to terrestrial ventures in autonomy (such as self-driving cars) due to the high risk of testing autonomous capabilities with no flight heritage on extremely expensive space vehicles. In practice, this has always motivated onboard guidance implementations that are computationally lean, deterministic, and easy to test and validate. Nonetheless recent trends in dropping cost-to-orbit and the proliferation of CubeSats anticipate the deployment of numerous lower-cost deep space spacecraft which will require an increased degree of autonomy to avoid overwhelming human-in-the-loop on-ground tracking and planning capabilities (Di Domenico et al., 2021). Thus it seems likely that efforts in robust computationally constrained guidance will find increasing demand in onboard planning and decision-making applications in the years to come.

The main characteristics of suitable on-board spacecraft guidance, are, according to Starek et al. (2016), (1) that it should be computationally reasonable, (2) it should compute an optimal solution wherever possible, and (3) it should enable verifiability. Convex optimization-based guidance presents an appealing candidate for meeting these challenges because by nature it is fast, extremely stable, computationally well-posed, and thus easy to profile computationally (Boyd and Vandenberghe, 2004). Within the convex optimization framework, it is also possible to develop *stochastic*

*Marie Skłodowska-Curie Postdoctoral Fellow, Department of Aerospace Science and Technology, Politecnico di Milano, ethanryan.burnett@polimi.it

†Professor, Department of Aerospace Science and Technology, Politecnico di Milano

guidance schemes that robustly accommodate expected dispersions in control performance as well as state and parameter estimation errors. Chance-constrained methods are becoming popular for this because they provide performance guarantees with a user-defined confidence level – see e.g. Oguri and McMahon (2021) and Berning Jr. et al. (2023) for recent examples from academia and industry. In addition to stochasticity, nonlinearity poses a perennial and ubiquitous challenge. Even for otherwise convex problems, the general non-convexity of nonlinear dynamics often forces an iterative approach via successive convexification (Mao et al., 2017), whereby a non-convex problem is solved locally as a convex sub-problem subject to trust region constraints to enforce a stable iterative march towards the optimal solution. While extremely powerful, successive convexification implementations tend to require fairly problem-specific details of choice of transcription and trust-region updates to ensure stability, feasibility, and computational efficiency.

Among the aforementioned challenges of on-board guidance, for this work we are focused on methods for fast and easily characterizable onboard guidance for nonlinear systems. Guidance of nonlinear systems invariably comes with some non-trivial computational cost. However, not all of this cost needs to be paid in real-time. Developing computationally feasible on-board guidance thus involves the development of frameworks that keep specifically the real-time computational footprint to a minimum. This is done by enabling pre-computation of useful and reusable information to the maximum extent possible. Along these motivational lines, we introduce a new framework for using and studying nonlinear expansions of problem dynamics that is demonstrably computationally efficient. It requires no real-time integration – either explicitly (as in predictor-corrector methods or multiple shooting) or implicitly (as with collocation schemes). The method requires in real-time only fairly simple computation and manipulation of matrices and vectors of monomials and standard linear algebra. Furthermore it enables geometrically intuitive and easy-to-implement differential correction and successive convexification schemes for trajectory optimization. The truncated nonlinear expansion of the solution, written in a special ordered quasi-linear form, is our “transcription” scheme. The optimization problem is then posed entirely in terms of a special set of coordinates – an overparameterized set of monomials in the osculating initial conditions of the problem. The accuracy of the guidance scheme is then easily characterized by ensuring that the problem, when parameterized in these coordinates, always remains within a region of desired accuracy of the domain of the truncated nonlinear expansions.

Many past works are relevant to the ideas and techniques expounded in this work. First, to facilitate our computations, we make use of a computerized monomial algebra, whereby monomial series of an arbitrary order and number of variables are represented by arrays of their coefficients, whose spatial arrangement matches the ordering of the monomial terms. Similar schemes have been implemented before – Giorgilli and Sansottera (2011) gives a broad overview of commonly used methods. Jorba (1999) develops a computer algebra system in which they compute, store, and manipulate monomials in a manner very similar to ours. Their applications are otherwise starkly different, leveraging manipulation of monomials in a sequence of canonical transformations for the purpose of constructing normal forms and obtaining approximate integrals of Hamiltonian systems.

The other half of our methodology involves the computation of nonlinear fundamental solution expansions about a reference – each associated with a particular unique monomial in the initial conditions. We’ve explored their computation by many means. Firstly, via “State Transition Tensors”, which are an extension of the ubiquitous state transition matrix involved in almost any method involving both linearization and discretization. Park and Scheeres (2006) provide a classic overview, and Boone and McMahon (2021) provide a more recent predictor-corrector guidance implementation for two-burn maneuvers. We have also computed the nonlinear fundamental solutions

leveraging Differential Algebra methods, which facilitates automated computation of derivatives via a structure allowing direct treatment of many topics related to the differentiation and integration of functions (Berz, 1999). This methodology has seen noteworthy use in spaceflight GNC in uncertainty propagation (Valli et al., 2013) and optimal control (Di Lizia et al., 2014; Greco et al., 2020). Lastly, nonlinear fundamental solution expansions can be computed analytically for some specialized problems. This is typically aided nowadays by the use of computational software such as *Mathematica* (Wolfram Research, Inc., 2023), and usually involves the use of perturbation methods (Nayfeh, 2000; Hinch, 1991). In this work, our example application is one for which many such expansions have been analytically derived: the spacecraft relative motion and rendezvous problem. References such as Butcher et al. (2016), Butcher et al. (2017), Willis et al. (2019b), and Willis et al. (2019a) are thus especially relevant to us.

This work makes use of convex optimization for trajectory optimization. In addition to the thorough foundational work of Boyd and Vandenberghe (2004), see also work outlining the successive convexification algorithm SCvx (Mao et al., 2017) and recent improvements (e.g. Oguri (2023) for an augmented Lagrangian formulation to enable a feasibility guarantee of SCvx). Python is our chosen language for convex optimization prototyping, offering mature open-source tools such as CVXPY (Diamond and Boyd, 2016; Agrawal et al., 2018) and ECOS (Domahidi et al., 2013), as well as the recent introduction of CVXPYgen (Schaller et al., 2022) for especially rapid problem-solving via generation of a speedy custom solver implementation in C called directly from the Python implementation as a CVXPY solver method. For a clear and simple introduction for convex optimization methods for spacecraft trajectory optimization, see Wang and Grant (2018), and for a discussion of the necessary collocation schemes, we recommend Kelly (2017). Our work within the DART Lab at Polimi follows some notable other works making use of convex optimization for trajectory optimization. Much of this work falls under the ERC-funded EXTREMA project (Di Domenico et al., 2021) for self-driving interplanetary CubeSats. Hofmann and Topputo (2021) present a computationally simple and robust convex optimization-based algorithm for low-thrust interplanetary trajectories. Morelli et al. (2022) apply convex optimization to a similar problem while considering a homotopic energy-to-fuel optimal approach along with “second-order” trust region methods. Hofmann et al. (2023) perform a wide study of different discretization and trust region methods for convex low-thrust trajectory optimization. In this early work we lay the groundwork for both continuous and impulsive control, but we explore only the latter in-depth. Our example application is the long-range spacecraft rendezvous problem under impulsive thrust – see e.g. Berning Jr. et al. (2023) (with drift safety guarantees) or Burnett and Schaub (2022) (for methods applicable to any general periodic orbits) for applications to the close-range problem with convex programming.

This paper is organized as follows. In Section 2, we highlight the fundamental ideas behind our methodology. We discuss the idea of osculating initial conditions, which is central to this work, along with the useful kinematic constraints that arise from this concept. In Section 3.1 we move on to the applications of our framework to spacecraft trajectory optimization, culminating in an example simple successive convexification implementation whose real-time operations are computationally extremely lean and easy to implement. In Section 4 we provide a thorough example application to the nonlinear spacecraft rendezvous problem. Section 4.1 provides a simple two-stage (linear prediction, nonlinear correction) convex guidance strategy leveraging our developments, and Section 4.2 showcases implementation of the successive convexification scheme from Section 3.1. We make concluding remarks and suggestions for future work in Section 5.

2 Theory

2.1 Osculating Initial Conditions

This work considers controlled dynamical systems with coordinates $\mathbf{x} \in \mathbb{R}^N$ of the following control-affine form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + B(\mathbf{x}, t)\mathbf{u} \quad (1)$$

where $\mathbf{u} \in \mathbb{R}^m$ is a control signal and t is time. For any instantaneous state $\mathbf{x}(t)$, it is always possible to parameterize instead by its initial conditions $\mathbf{x}(0)$ and a time t , such that the two are related by the flow of the natural dynamics (e.g. control-free, $\mathbf{u} = \mathbf{0}$) as $\mathbf{x}(t) = \varphi(\mathbf{x}(0), t, 0)$ or equivalently by the inverse mapping:

$$\mathbf{x}(0) = \varphi^{-1}(\mathbf{x}(t), t, 0) \quad (2)$$

Based on Eq. (2), we could take some controlled state $\mathbf{x}(t)$ at some time $t > 0$ and back-propagate to a corresponding ‘‘osculating’’ initial condition at time 0. This osculating initial condition, when propagated via the natural dynamics, matches up with the controlled state at time t . Depending on the selected ‘‘matching’’ time with the controlled trajectory, the osculating initial condition will assume a different value. We denote this as $\mathbf{c}_1(t)$. In this manner the \mathbf{c}_1 serves as its own coordinate description of the problem, because the mapping to and from $\mathbf{x}(t)$ is always well-defined. Conveniently, this parameterization removes the influence of natural dynamics, so the problem state is stationary unless control is actively being applied.

For a linear dynamical system, the transformation φ is linear and easy to compute, and so past works have explored solving linearized guidance problems leveraging parameterization in terms of the initial conditions (or more generally, integration constants) of the system (Guffanti and D’Amico, 2018; Burnett and Schaub, 2022). This transcribes trajectory optimization problems as a path planning problem in a transformed domain free of natural dynamics. For nonlinear systems, by contrast, we are accustomed to computing the mapping φ or its inverse via relatively costly numerical integration techniques. Because of this, a general reformulation of nonlinear guidance or control problem in terms of initial states previously would not be considered much of an improvement on the problem to be solved. In this work we use the concept of nonlinear fundamental expansions to develop a practical theory valid for nonlinear systems in the vicinity of some reference. We leverage (1) pre-computation of the nonlinear fundamental solution expansions and (2) a special quasi-linear parameterization, in lieu of nonlinear representations such as tensors/summation notations, which greatly facilitates our developments.

2.2 Overparameterized Representation

Instead of the minimal parameterization (linear in the initial conditions) discussed previously, we need to instead parameterize our system by \mathbf{c}_j – a minimal collection of unique mixed monomials up to $\mathcal{O}(x_k(0)^j)$, which is generated from the N initial states $x_1(0), \dots, x_N(0)$. Thus the aforementioned \mathbf{c}_1 is just \mathbf{c}_j for $j = 1$. For $j > 1$, \mathbf{c}_j lies on an N -dimensional constrained surface in \mathbb{R}^{K_j} , with K_j given below:

$$K_j = \sum_{q=1}^j \binom{N+q-1}{q} \quad (3)$$

One strategy for building an ordered \mathbf{c}_j is as follows. For all unique mixed monomials of a given order r , multiply by $x_1(0)$, and append to the list. Then multiply the order- r mixed monomials by

$x_2(0)$, and append in order only the non-repeated values, repeating this procedure through all state variables to $x_N(0)$. This process initializes with $\mathbf{c}_1 = \mathbf{x}(0)$ and $r = 1$, and finishes after $r = j - 1$ to produce \mathbf{c}_j . For example, for $N = 3$, $j = 2$, \mathbf{c}_2 is computed as below:

$$\mathbf{c}_2 = (x_1(0), x_2(0), x_3(0), x_1^2(0), x_1x_2(0), x_1(0)x_3(0), x_2^2(0), x_2(0)x_3(0), x_3^2(0))^\top \quad (4)$$

See e.g. Giorgilli and Sansottera (2011) and Jorba (1999) for more discussion about the computerized algebraic manipulation of sets of monomials. With the construction of \mathbf{c}_j established, we can in general nonlinearly expand the solution of a dynamical system about a fixed point (or an arbitrary reference, w.l.o.g.) in terms of unique monomials and their associated fundamental solutions as below:

$$\begin{aligned} \mathbf{x}(t) &\approx x_1(0)\boldsymbol{\psi}_{x_1}(t) + x_2(0)\boldsymbol{\psi}_{x_2}(t) + \dots + x_N(0)\boldsymbol{\psi}_{x_N}(t) \\ &\quad + x_1^2(0)\boldsymbol{\psi}_{x_1^2}(t) + x_1(0)x_2(0)\boldsymbol{\psi}_{x_1x_2}(t) + \dots + x_N^j(0)\boldsymbol{\psi}_{x_N^j}(t) \\ &= \boldsymbol{\Psi}_j(t)\mathbf{c}_j \end{aligned} \quad (5)$$

The $\boldsymbol{\psi}$ functions are the fundamental solutions of the expanded dynamics, and can be constructed to a given desired order by various means, including using the relevant terms from the associated state transition tensors (STTs) for the problem (Park and Scheeres, 2006; Boone and McMahon, 2021) or the higher-order Taylor map (HOTM) from differential algebra (DA) (Berz, 1999; Valli et al., 2013) – computed numerically, or perturbation solutions (Nayfeh, 2000; Hinch, 1991) which are computed analytically. These topics are discussed in greater detail the Appendix A. An important thing to note is that, at a given order, the fundamental solution/monomial representation is a minimal representation written in a linear form, whereas the STTs are a redundant representation, written in general in summation form.

We denote the N -dimensional surface that \mathbf{c}_j lies on as $\mathcal{C}^{(N)}$, which is embedded in a K_j -dimensional space. Figure 1 depicts for a system $\mathbf{x} = (x, \dot{x})^\top$ (i.e. $N = 2$) the surface obtained with a set of monomials up to order 2 - omitting \dot{x}_0^2 and $x_0\dot{x}_0$ from the second-order monomials (reducing K_j from 5 to 3) to enable a simple 3D example. Note that, given a point $\mathbf{c}_j \in \mathcal{C}^{(N)}$, any physically

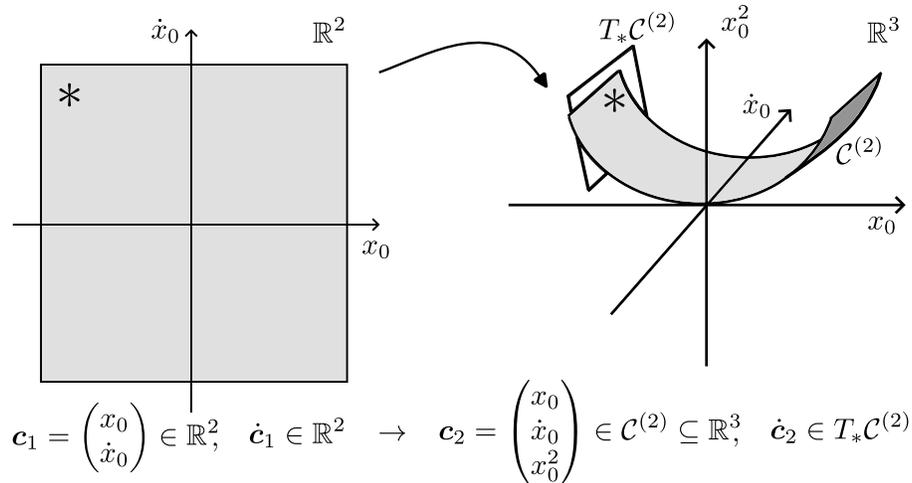


Figure 1: Monomial Coordinates and Initial Conditions

admissible infinitesimal variations $\delta\mathbf{c}_j$ are constrained to lie in the tangent space $T_{\mathbf{c}_j}\mathcal{C}^{(N)}$, because

the monomials are an overparameterized representation of the system. For example, applying a change in the linear component of \mathbf{c}_j associated with $x_1(0)$, the higher-order components of \mathbf{c}_j such as $x_1^2(0)$ are forced to take on a certain value, because they are functionally dependent on the linear components. You can convince yourself of this by computing variations of Eq. (4) based on various values of $\mathbf{x}(0) \in \mathbb{R}^N$, and noting that the higher-order components are nonlinearly dependent on the linear components.

2.3 Variation of Parameters

The overparameterized monomial representation is quite useful, but before explaining we must emphasize that our arguments are locally and not globally valid. The size of the region of validity grows with the order of expansion j , up to the limit of the radius of convergence. Below the j^{th} order approximate representation is restated for the control-free case $\mathbf{u} = \mathbf{0}$:

$$\mathbf{x}(t) \approx \Psi_j(t)\mathbf{c}_j \quad (6)$$

where the approximation notation simply indicates that the nonlinear fundamental solutions only approximate the evolution of the true state from $\mathbf{x}(0)$. We define our operative domain and timescale, $\mathcal{D} \times \mathcal{T}$, to be that in which for our order j and basis Ψ_j the state $\mathbf{x}(t)$ is continuously well-represented by our finite nonlinear expansion throughout that space for any $t \in \mathcal{T}$. In particular, the $\mathbf{x}(t)$ generated by the nonlinear expansion should always satisfy some error metric ε when compared to a numerically integrated counterpart $\mathbf{x}_n(t)$ propagated from the same initial condition $\mathbf{x}(0)$, e.g. $\|\mathbf{x}(t) - \mathbf{x}_n(t)\| < \varepsilon \forall t \in \mathcal{T}$. At an epoch time $t_0 \triangleq 0$, \mathbf{c}_j is generated uniquely from the initial conditions – the state \mathbf{x} at time $t = 0$. The following mapping holds for j^{th} -order approximations of natural trajectories:

$$(\mathbf{x}, t) \in \{\mathcal{D} \times \mathcal{T}\} \subseteq \{\mathbb{R}^N \times \mathbb{R}^{\geq 0}\} \longleftrightarrow \mathbf{c}_j \in \mathcal{C}^{(N)} \subseteq \mathbb{R}^{K_j} \quad (7)$$

If we choose a state vector and time from the left side of Eq. (7), we will have sampled a trajectory evolving in time at that particular time. That trajectory will have its own unique initial conditions $\mathbf{x}(0)$ at the epoch time. On the right is a corresponding stationary state parameterized by an ordered collection of monomials of initial conditions, e.g. components of $\mathbf{x}(0)$. The mapping from right to left is simply Eq. (6). The inverse mapping also exists and is unique – but it isn't analytic.

Let $\mathcal{C}_\varepsilon^{(N)}$ denote the compact sub-domain within $\mathcal{C}^{(N)}$ that, given *any* specified $t \in \mathcal{T}$ and $\mathbf{c}_j \in \mathcal{C}_\varepsilon^{(N)}$, generates a corresponding \mathbf{x} within the domain of validity \mathcal{D} at that time t . In general the validity of our methods can be understood to hold only in the localized context that all points \mathbf{c}_j of interest for a problem are contained in a $\mathcal{C}_\varepsilon^{(N)}$ corresponding to a sufficiently low ε that the approximations are adequate for a given application. This then is easy to verify if, when computing and constructing Ψ_j , we develop also an approximation of its region of validity $\mathcal{C}_\varepsilon^{(N)}$.

Subject to the true nonlinear dynamics with vector field $\mathbf{f}(\mathbf{x}, t) + B(\mathbf{x}, t)\mathbf{u}$, we seek the induced variations $\dot{\mathbf{c}}_j$ such that the true solution can still be represented using our finite functional expansion, i.e. such that Eq. (6) still holds, but now the constants are forced to vary:

$$\mathbf{x}(t) = \Psi_j(t)\mathbf{c}_j(t) \quad (8)$$

In this manner the induced variations in \mathbf{c}_j are analogous, for example, to the induced variations in the classical two-body orbit element description under the influence of control or some other

perturbations. This “osculating” description is similar to a method by which Gauss’ variational equations in can be derived, and the following relationship must hold:

$$\dot{\mathbf{x}}(t) = \frac{\partial \mathbf{x}(t)}{\partial t} + \frac{\partial \mathbf{x}(t)}{\partial \mathbf{c}_j} \dot{\mathbf{c}}_j = \mathbf{f}(\mathbf{x}, t) + B(\mathbf{x}, t)\mathbf{u} \quad (9a)$$

$$\dot{\mathbf{x}}(t) = \dot{\Psi}_j(t)\mathbf{c}_j + \Psi_j(t)\dot{\mathbf{c}}_j = \mathbf{f}(\mathbf{x}, t) + B(\mathbf{x}, t)\mathbf{u} \quad (9b)$$

Within a region of convergence, the term $\dot{\Psi}_j\mathbf{c}_j$ approximates the natural flow of the dynamics, $\mathbf{f}(\mathbf{x}, t)$, with an accuracy that improves with higher orders j , thus the below approximation is obtained after the cancellation of the similar terms:

$$\Psi_j(t)\dot{\mathbf{c}}_j \approx B(\mathbf{x}, t)\mathbf{u} \quad (10)$$

The error in this expression generally grows with time t and distance from the reference (i.e. $\|\mathbf{x}\|$), and shrinks with order j . Recall that infinitesimal variations $\delta\mathbf{c}_j$ lie in the tangent space $T_{\mathbf{c}_j}\mathcal{C}^{(N)}$, thus the following is equivalent to Eq. (10):

$$\Psi_j(t)\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1}\dot{\mathbf{c}}_1 \approx B(\mathbf{x}, t)\mathbf{u} \quad (11)$$

Note that $\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1}$ is also simply a function of monomials in components of $\mathbf{x}(0)$ up to order $j - 1$. For systems of interest to us, the product $\Psi_j(t)\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1}$ can be easily shown to be generally invertible, thus we obtain the following variational equations (dropping the “ \approx ” notation but with the understanding that they are approximations):

$$\dot{\mathbf{c}}_1 = \left(\Psi_j(t)\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right)^{-1} B(\mathbf{x}, t)\mathbf{u} \quad (12a)$$

$$\dot{\mathbf{c}}_j = \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \left(\Psi_j(t)\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right)^{-1} B(\mathbf{x}, t)\mathbf{u} \quad (12b)$$

These equations represent our original dynamical system of Eq. (1) in terms of coordinates that are stationary in the absence of control – either via a minimal representation, \mathbf{c}_1 , or an associated overparameterized representation \mathbf{c}_j , which can easily be computed from \mathbf{c}_1 . Within the region of convergence of all relevant expansions, as the order j is increased, these equations better approximate the variation of osculating initial conditions for some controlled system (e.g. as discussed in Section 2.1).

2.4 Kinematics and Dynamics

Because the monomial coordinates are stationary in the absence of control, any motion of $\mathbf{c}_j \in \mathcal{C}^{(N)}$ only occurs when control is active. While the preceding developments are valid for any state representation of \mathbf{x} (i.e. any choice of coordinates), here we examine the implications for a system specifically parameterized with Cartesian position and velocity coordinates $\mathbf{x}(t) = (\mathbf{r}^\top, \mathbf{v}^\top)^\top$. Consider first the example of continuous control $\mathbf{u}(t)$. The controlled dynamics are given by the equations below, differentiating Eq. (6):

$$\dot{\mathbf{r}} = \dot{\Psi}_{r,j}\mathbf{c}_j + \Psi_{r,j}\dot{\mathbf{c}}_j \quad (13a)$$

$$\dot{\mathbf{v}} = \dot{\Psi}_{v,j}\mathbf{c}_j + \Psi_{v,j}\dot{\mathbf{c}}_j \quad (13b)$$

where $\Psi_{r,j}$ denotes the top $N/2$ (position-associated) rows of Ψ_j , and $\Psi_{v,j}$ the bottom $N/2$ (velocity-associated) rows. The following kinematic identity must be satisfied, noting that by definition of our choice of coordinates \boldsymbol{x} , the nonlinear fundamental solutions satisfy $\Psi_{v,j}(t) = \dot{\Psi}_{r,j}(t)$:

$$\begin{aligned}\dot{\boldsymbol{r}} &= \boldsymbol{v} = \Psi_{v,j} \boldsymbol{c}_j \\ &= \dot{\Psi}_{r,j} \boldsymbol{c}_j\end{aligned}\tag{14}$$

From this we obtain the below constraint on admissible directions of $\dot{\boldsymbol{c}}_j \in T_{\boldsymbol{c}_j} \mathcal{C}^{(N)}$:

$$\Psi_{r,j} \dot{\boldsymbol{c}}_j = \mathbf{0}\tag{15}$$

Then, denoting the natural acceleration \boldsymbol{a} and the control acceleration $\boldsymbol{a}_c = B_L(\boldsymbol{x}, t) \boldsymbol{u}$ where B_L is the bottom $N/2$ rows of control matrix B (and furthermore $B_L = I_{3 \times 3}$ in the general 3D Cartesian case), we obtain also the following relationship from Eq. (13) after noting $\boldsymbol{a} = \dot{\Psi}_{v,j} \boldsymbol{c}_j$:

$$\boldsymbol{a}_c = \Psi_{v,j} \dot{\boldsymbol{c}}_j\tag{16}$$

Note that Eq. (16) can also be obtained directly from Eqs. (11) and (12) upon substitution of the correct form for $B(\boldsymbol{x}, t)$.

For the case of impulsive maneuvers $\Delta \boldsymbol{v}(t_i)$ occurring at various discrete times, the impulsive analogs of the above constraints are:

$$\Psi_{r,j}(t_i) (\boldsymbol{c}_j(t_i) - \boldsymbol{c}_j(t_{i-1})) = \mathbf{0}\tag{17a}$$

$$\Delta \boldsymbol{v}(t_i) = \Psi_{v,j}(t_i) (\boldsymbol{c}_j(t_i) - \boldsymbol{c}_j(t_{i-1}))\tag{17b}$$

Thus the kinematics and dynamics are encoded as fundamental path constraints on admissible paths of \boldsymbol{c}_j on $\mathcal{C}^{(N)}$. Furthermore the cost paid to move in $\mathcal{C}^{(N)}$, i.e. the delta-V, is a simple linear function of the jump in the monomial states. Note that alternate choice of working coordinates for \boldsymbol{x} simply results in a different linear relationship in the form of Eq. (17), as long as the non-linear transformation to Cartesian coordinates admits a regular Taylor expansion. Lastly, as previously stated, if we already have an idea of our domain of validity $\mathcal{C}_\varepsilon^{(N)}$ within $\mathcal{C}^{(N)}$, to ensure accuracy of any methods that use this representation, we can simply check that all $\boldsymbol{c}_j(t_i) \in \mathcal{C}_\varepsilon^{(N)}$. This is much easier than the alternative test of ensuring that $(\boldsymbol{x}, t) \in \mathcal{D} \times \mathcal{T}$ for all states and times of interest.

3 Applications

3.1 Spacecraft trajectory optimization

Consider the optimization of nonlinear spacecraft trajectories composed of (1) impulsive maneuvers, and (2) coast arcs between maneuvers. The monomial coordinates, being effectively constant in the absence of control, are naturally well-suited for posing this trajectory optimization problem as a path-planning problem. We focus on a simple unconstrained delta-V optimal nonlinear control example. We note however that by the linear relationship between \boldsymbol{x} and \boldsymbol{c}_j , constraints on the states $\boldsymbol{x}(t_i)$ at times t_i (and even constraints on control-free drifts $\boldsymbol{x}(\tau_i)$, for times $\tau_i \geq t_i$ with $\boldsymbol{u}(\tau_i) = \mathbf{0}$) are easily inherited by the monomial coordinates $\boldsymbol{c}_j(t_i)$ as well.

The transcription between monomial coordinates $\boldsymbol{c}_j \in \mathcal{C}^{(N)}$ and the trajectory $\boldsymbol{x}(t)$ is depicted in an example 4-burn solution in Figure 2. Note the monomial state is stationary in the absence

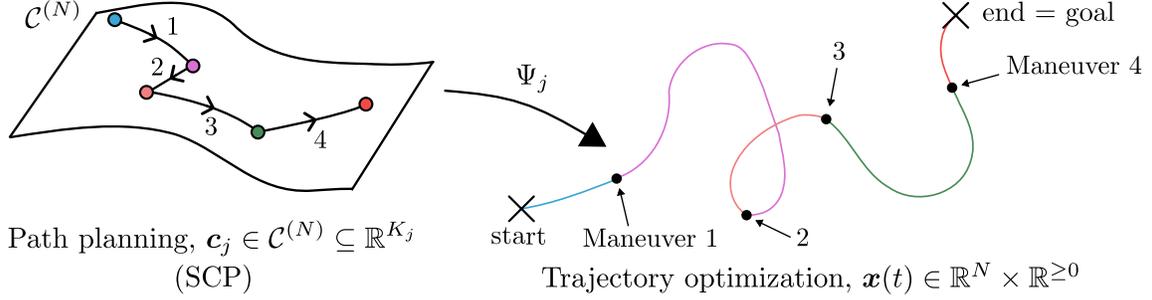


Figure 2: Trajectory Optimization as Path Planning on $\mathcal{C}^{(N)}$

of maneuvers and it is constrained to lie on $\mathcal{C}^{(N)}$, but is otherwise unconstrained, with all nodes free to assume their respective locations such that the overall minimizing path is obtained. We explore two optimization schemes – minimizing $J = \sum_i \|\Delta \mathbf{v}(t_i)\|^2$ (i.e. “energy-minimizing”) and also $J = \sum_i \|\Delta \mathbf{v}(t_i)\|$ (minimizing delta-V). We differentiate between them by their exponent “ \mathcal{P} ”, 2 or 1. The former is parsed explicitly in this work, whereas the objective function for the latter is implemented via `CVXPY` parsing.

3.1.1 Problem formulation

The unconstrained fixed-time optimization problem with impulsive maneuvers is given below:

$$\begin{aligned}
 & \text{minimize} && \sum_{i=1}^K \|\Delta \mathbf{v}(t_i)\|^{\mathcal{P}} \\
 & \text{subject to} && \mathbf{x}(t_0) = \mathbf{x}_0 \\
 & && \mathbf{x}(t_K^+) = \mathbf{x}_f \\
 & && \mathbf{x}(t_{i+1}^-) = \boldsymbol{\varphi}(\mathbf{x}(t_i^+), t_{i+1}, t_i) \\
 & && \mathbf{r}(t_i^+) = \mathbf{r}(t_i^-)
 \end{aligned} \tag{18}$$

where we explore $\mathcal{P} = 1, 2$, and both yield a convex problem. The trajectory is split into up to $K+1$ different points, corresponding to an initial condition plus K times where maneuvers are allowed. For $K \gg N$, the optimal solution will generally have $\Delta \mathbf{v}(t_i) = \mathbf{0}$ for many discretized times t_i . The expression $\boldsymbol{\varphi}(\mathbf{x}(t_1), t_2, t_1)$ denotes the flow of the state $\mathbf{x}(t)$ from t_1 to t_2 , and t_i^- and t_i^+ denote the time t_i at the instants before and after a maneuver $\Delta \mathbf{v}(t_i)$. The first two conditions in Eq. (18) are simple boundary conditions based on prescribed initial and final states of the trajectory. The third condition is a requirement that the state between maneuvers obey the flow of the natural dynamics. The fourth is a continuity condition requiring that the position is unchanged during an instantaneous maneuver.

We seek to rewrite the problem of Eq. (18) in terms of the monomial coordinates. To do this, we exploit a few useful properties. Recall the following mapping from monomial coordinates to the instantaneous state at some time t :

$$\mathbf{x}(t) = \Psi_j(t, t_0) \mathbf{c}_j \tag{19}$$

Given an instantaneous maneuver $\Delta \mathbf{v}(t_i)$, the corresponding $\Delta \mathbf{c}_j(t_i) = \mathbf{c}_j(t_i) - \mathbf{c}_j(t_{i-1})$ must satisfy the following for a Ψ_j partitioned row-wise into $\Psi_{r,j}$ (top $N/2$ rows - corresponding to position

states) and $\Psi_{v,j}$ (bottom $N/2$ rows - velocity states):

$$\Delta \mathbf{c}_j(t_i) \in \ker(\Psi_{r,j}(t_i, t_0)) \quad (20a)$$

$$\Delta \mathbf{v}(t_i) = \Psi_{v,j}(t_i, t_0) \Delta \mathbf{c}_j(t_i) \quad (20b)$$

$$\mathbf{c}_j(t_{i-1}) + \Delta \mathbf{c}_j(t_i) \in \mathcal{C}^{(N)} \quad (20c)$$

The first constraint is that the position is not changed by the impulsive maneuver. The second constraint is true by definition of the fundamental solution matrix Ψ_j in Cartesian coordinates. The third constraint states that regardless of the change induced in \mathbf{c}_j , the new monomial state must still be on the manifold $\mathcal{C}^{(N)}$. We can now rewrite the problem of Eq. (18):

$$\begin{aligned} & \text{minimize} \quad \sum_{i=1}^K \|\Psi_{v,j}(t_i) (\mathbf{c}_j(t_i) - \mathbf{c}_j(t_{i-1}))\|^{\mathcal{P}} \\ & \text{subject to} \quad \begin{aligned} & \mathbf{c}_j(t_0) = \mathbf{c}_{j,\text{start}} \\ & \mathbf{c}_j(t_K) = \mathbf{c}_{j,\text{goal}} \\ & \mathbf{c}_j(t_i) - \mathbf{c}_j(t_{i-1}) \in \ker(\Psi_{r,j}(t_i)) \\ & \mathbf{c}_j(t_i) \in \mathcal{C}^{(N)}, \quad \forall i \geq 1 \end{aligned} \end{aligned} \quad (21)$$

The problem is reduced to choosing reasonable discrete steps $\Delta \mathbf{c}_j(t_i)$, $i = 1, 2, \dots, K$, achieving the desired path from $\mathbf{c}_{j,\text{start}}$ at t_0 to $\mathbf{c}_{j,\text{goal}}$ at t_f , while minimizing the above cost. This is intended as a simple example, but in general any constraints on the state $\mathbf{x}(t)$ are inherited by constraints on the (linearly related) \mathbf{c}_j .

For this optimal path-planning problem, we write out a vector of (preliminary) decision variables as $\tilde{\mathbf{X}} = (\mathbf{c}_j(t_1)^\top, \mathbf{c}_j(t_2)^\top, \dots, \mathbf{c}_j(t_k)^\top)^\top$. The cost function of the problem given by Eq. (21) can be shown to be quadratic in $\tilde{\mathbf{X}}$, and the first two constraints are clearly linear. The third constraint is also linear, written simply as $\Psi_{r,j}(t_i) \Delta \mathbf{c}_j(t_i) = \mathbf{0}$. The only non-convex part of the problem given by Eq. (21) is the final constraint that the decision variables lie on the manifold $\mathcal{C}^{(N)}$. For this non-convexity we propose a successive convex programming problem.

3.1.2 Successive convex programming

Figure 3 conceptually depicts a trajectory, in terms of 5 distinct points in the monomial coordinates, resulting from 4 impulsive maneuvers. In the vicinity of any of the points in the trajectory shown in Figure 3, we can linearly approximate local variations in \mathbf{c}_j to lie in the tangent space as below:

$$\delta \mathbf{c}_j(t_i) \approx \left. \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right|_{\mathbf{c}_j(t_i)} \delta \mathbf{c}_1(t_i) \in T_{\mathbf{c}_j(t_i)} \mathcal{C}^{(N)} \quad (22)$$

The key to a successive convex programming (SCP) implementation of the problem given by Eq. (21) is to make use of this local tangent plane approximation, and also to choose our free variables for the convex problem as the $\delta \mathbf{c}_1(t_i)$, with the nominal $\mathbf{c}_j(t_i)$ chosen from a prior iteration (or, for iteration 1, from the initial guess of the trajectory). Thus $\tilde{\mathbf{X}} = (\delta \mathbf{c}_1(t_1)^\top, \delta \mathbf{c}_1(t_2)^\top, \dots, \delta \mathbf{c}_1(t_K)^\top)^\top$,

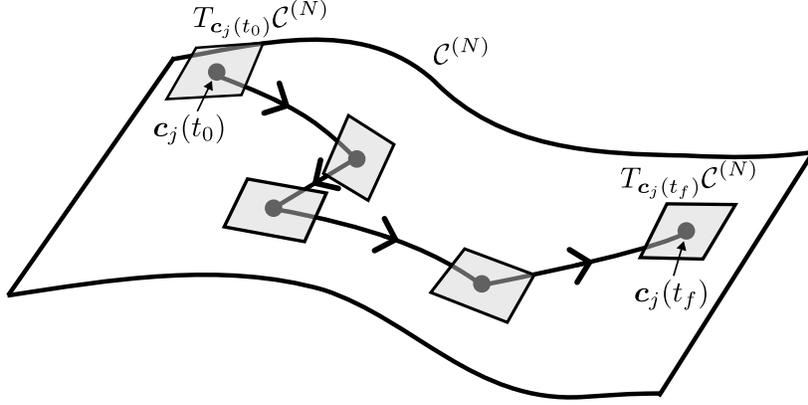


Figure 3: Successive Convexification via Monomial Coordinates on $\mathcal{C}^{(N)}$

and transforming Eq. (21), the resulting convex sub-problem is given below:

$$\begin{aligned}
& \text{minimize} \quad \left\| \Psi_{v,j}(t_1) \left(\mathbf{c}'_j(t_1) + \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_1 \delta \mathbf{c}_1(t_1) - \mathbf{c}_j(t_0) \right) \right\|^{\mathcal{P}} + w \left(\sum_{i=1}^K \|\mathbf{s}_i\|^2 + \|\mathbf{s}_{\text{end}}\|^2 \right) \\
& \quad + \sum_{i=2}^K \left\| \Psi_{v,j}(t_i) \left(\mathbf{c}'_j(t_i) + \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_i \delta \mathbf{c}_1(t_i) - \mathbf{c}'_j(t_{i-1}) - \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_{i-1} \delta \mathbf{c}_1(t_{i-1}) \right) \right\|^{\mathcal{P}} \\
& \text{subject to} \quad \Psi_j(t_k) \left(\mathbf{c}'_j(t_k) + \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_K \delta \mathbf{c}_1(t_k) \right) + \mathbf{s}_{\text{end}} = \mathbf{x}_{\text{goal}}(t_k) \\
& \quad \Psi_{r,j}(t_1) \left(\mathbf{c}'_j(t_1) + \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_1 \delta \mathbf{c}_1(t_1) - \mathbf{c}_j(t_0) \right) + \mathbf{s}_1 = \mathbf{0} \\
& \quad \Psi_{r,j}(t_{i+1}) \left(\mathbf{c}'_j(t_{i+1}) - \mathbf{c}'_j(t_i) + \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_{i+1} \delta \mathbf{c}_1(t_{i+1}) - \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_i \delta \mathbf{c}_1(t_i) \right) + \mathbf{s}_{i+1} = \mathbf{0}, i = 1:K-1
\end{aligned} \tag{23}$$

Note the careful parsing of expressions involving $\mathbf{c}_j(t_0)$, which is fixed in this example and is not part of the decision variables. The $()'$ denotes terms from the solution to the prior iteration, about which the current iteration is expanded. Furthermore $\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_i$ is a shorthand for $\frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \Big|_{\mathbf{c}'_j(t_i)}$.

The enumerated slack variables \mathbf{s}_i are introduced to prevent artificial infeasibility of the convex sub-problem, with a scalar weight $w > 0$ to specify the degree of penalization of slack terms (the converged solution must satisfy $\mathbf{s}_i = \mathbf{0}$). These slack variables can be physically interpreted as the positional defect constraints in the trajectory. There is also an additional slack variable \mathbf{s}_{end} related to the satisfaction of the first listed linear constraint, which is a constraint on the end state, and is similarly penalized. For performance reasons, it is best for the problem states to be rendered non-dimensional, or for the last $N/2$ components of \mathbf{s}_{end} to be rescaled to have the same positional “units” as the \mathbf{s}_i slack variables. In this manner an SCP iteration will not be biased to over/under penalize any components \mathbf{s}_{end} in comparison to the other \mathbf{s}_i .

Between iterations, it is necessary to project the solution to the convex sub-problem (which exists in the union of the tangent planes of points $\mathbf{c}'_j(t_i) \forall i$) back onto the manifold. The most obvious (and computationally easiest) way is to compute for each update $+\delta \mathbf{c}_1(t_i)$ the new $\mathbf{c}_j(t_i)$ given by $\mathbf{c}'_j(t_i) + \delta \mathbf{c}_1(t_i)$, e.g. using the equation for \mathbf{c}_j , which will have some analytic form – Eq. (4) was our earlier example.

Because the tangent-plane approximations of variations $\delta \mathbf{c}_j$ are only locally valid, we must ad-

ditionally impose some kind of trust region constraint preventing the sub-problem from obtaining variations $\delta \mathbf{c}_1(t_i)$ that are too large. We impose a norm constraint on $\tilde{\mathbf{X}}$ to do this:

$$\|\tilde{\mathbf{X}}\| \leq d \quad (24)$$

We can set a fixed trust-region radius d , or alternatively we can update this via a proper trust-region update method such as the one outlined in (Hofmann et al., 2022).

The sub-problem defined by Eq. (23) can be resolved for $\mathcal{P} = 2$ in the form below:

$$\begin{aligned} & \text{minimize } \mathbf{X}^\top P \mathbf{X} + \mathbf{q}^\top \mathbf{X} \\ & \text{subject to } \begin{aligned} & A \mathbf{X} = \mathbf{b} \\ & \|M \mathbf{X}\| \leq d \end{aligned} \end{aligned} \quad (25)$$

For this formulation, we first augment the preliminary decision variables $\tilde{\mathbf{X}}$ with slack variables $\mathbf{S} = (\mathbf{s}_1^\top, \mathbf{s}_2^\top, \dots, \mathbf{s}_K^\top, \mathbf{s}_{\text{end}}^\top)^\top$ to form $\mathbf{X} = (\tilde{\mathbf{X}}^\top, \mathbf{S}^\top)^\top$, which is of length $(\frac{3}{2}K + 1)N$ for assumed even N . Because we only want to constrain $\tilde{\mathbf{X}}$, the form of M is simple:

$$M = \begin{bmatrix} I_{NK \times NK} & 0_{NK \times N(\frac{K}{2} + 1)} \end{bmatrix} \quad (26)$$

Because all constraints in Eq. (23) are linear and fairly simple, it is easy to construct A and \mathbf{b} . The form of P and \mathbf{q} however requires some algebra and will be provided.

While it does not influence the convex sub-problem, we must compute the part of the cost J in Eq. (23) that is not a function of the decision variables of a given iteration – this part of the cost is denoted J' , reusing the prime notation to denote this as the total cost predicted from the prior iteration:

$$J' = \sum_{i=i}^K \mathbf{c}_j'^\top(t_i) O_i \mathbf{c}_j'(t_i) - 2 \mathbf{c}_j'^\top(t_i) O_i \mathbf{c}_j'(t_{i-1}) + \mathbf{c}_j'^\top(t_{i-1}) O_i \mathbf{c}_j'(t_{i-1}) \quad (27a)$$

$$O_i = \Psi_{v,j}^\top(t_i) \Psi_{v,j}(t_i) \quad (27b)$$

Thus the solution to Eq. (25) provides the minimal admissible $\delta J = \mathbf{X}^\top P \mathbf{X} + \mathbf{q}^\top \mathbf{X}$ yielding total cost $J^{(q)} \approx J^{(q-1)} + \delta J^{(q)}$ for iteration q . In reality, after iteration q , the nonlinear projection of the linear step yields the true $J^{(q)}$ after a procedure of (1) nonlinearly updating all $\mathbf{c}_j'(t_i)$, (2) setting all $\delta \mathbf{c}_1(t_i) = \mathbf{0}$ in Eq. (23), then (3) using that to compute the true values of all slack variables, which are simply the true defects in the constraint equations after the nonlinear projection. From this info, we can compute the actual increment in cost δJ . As mentioned previously, we can take advantage of the information provided by the disparity between the predicted (linear) δJ and the actual (nonlinear) value. The two quantities are given below:

$$\delta J_{\text{predict}}^{(q)} = \mathbf{X}^\top P \mathbf{X} + \mathbf{q}^\top \mathbf{X} \quad (28a)$$

$$\delta J_{\text{actual}}^{(q)} = J^{(q)} - J^{(q-1)} \quad (28b)$$

where P , \mathbf{q} are computed using information from iteration $q - 1$.

We now define the matrix P and vector \mathbf{q} by sequential accounting instead of explicitly writing the large matrices. In this manner, they are initialized as zeros with the correct size, and consideration

of each successive cost term in Eq. (23) adds a corresponding part to the matrices whose form is to be defined.

$$P = P_0 + \sum \delta P, \quad P_0 = \mathbf{0}_{L \times L} \quad (29a)$$

$$\mathbf{q} = \mathbf{q}_0 + \sum \delta \mathbf{q}, \quad \mathbf{q}_0 = \mathbf{0}_{L \times 1} \quad (29b)$$

$$L = \left(\frac{3}{2}K + 1 \right) N \quad (29c)$$

Starting with P , the first (t_1), second (slack-associated), and third (indexed) cost terms of Eq. (23) produce the following contributions to P :

$$\delta P_1 = \left[\begin{array}{c|c} C_1^\top O_1 C_1 & \mathbf{0}_{N \times (L-N)} \\ \hline \mathbf{0}_{(L-N) \times N} & \mathbf{0}_{(L-N) \times (L-N)} \end{array} \right] \quad (30a)$$

$$C_1 = \left. \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right|_{\mathbf{c}'_j(t_1)}, \quad O_1 = \Psi_{v,j}^\top(t_1) \Psi_{v,j}(t_1) \quad (30b)$$

$$\delta P_2 = \left[\begin{array}{c|c} \mathbf{0}_{NK \times NK} & \mathbf{0}_{NK \times L_S} \\ \hline \mathbf{0}_{L_S \times NK} & w I_{L_S \times L_S} \end{array} \right] \quad (31a)$$

$$L_S = L - KN = N \left(\frac{K}{2} + 1 \right) \quad (31b)$$

$$\delta P_{3,i} = \left[\begin{array}{c|c|c|c} \mathbf{0}_{A \times A} & \mathbf{0}_{A \times N} & \mathbf{0}_{A \times N} & \mathbf{0}_{A \times B} \\ \hline \mathbf{0}_{N \times A} & C_{i-1}^\top O_i C_{i-1} & -C_{i-1}^\top O_i C_i & \mathbf{0}_{N \times B} \\ \hline \mathbf{0}_{N \times A} & -C_i^\top O_i C_{i-1} & C_i^\top O_i C_i & \mathbf{0}_{N \times B} \\ \hline \mathbf{0}_{B \times A} & \mathbf{0}_{B \times N} & \mathbf{0}_{B \times N} & \mathbf{0}_{B \times B} \end{array} \right] \quad (32a)$$

$$C_i = \left. \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right|_{\mathbf{c}'_j(t_i)}, \quad O_i = \Psi_{v,j}^\top(t_i) \Psi_{v,j}(t_i) \quad (32b)$$

$$\mathcal{A} = N(i-2), \quad \mathcal{B} = L - Ni, \quad i = 2:K \quad (32c)$$

where the $\delta P_{3,i}$ is added for every applicable index i . Now \mathbf{q} is defined similarly for the first (t_1) and third (indexed) cost terms:

$$\delta \mathbf{q}_1 = \left[2 \left(\mathbf{c}'_j(t_1) - \mathbf{c}_j(t_0) \right)^\top O_1 C_1 \;\; \mathbf{0}_{1 \times L-N} \right]^\top \quad (33)$$

$$\delta \mathbf{q}_{2,i} = \left[\mathbf{0}_{1 \times \mathcal{A}} \;\; -2 \left(\mathbf{c}'_j(t_i) - \mathbf{c}'_j(t_{i-1}) \right)^\top O_i C_{i-1} \;\; 2 \left(\mathbf{c}'_j(t_i) - \mathbf{c}'_j(t_{i-1}) \right)^\top O_i C_i \;\; \mathbf{0}_{1 \times \mathcal{B}} \right]^\top, \quad i = 2:K \quad (34)$$

Beyond the nonlinear projection, the only other nonlinear calculations needed for the setup of Eq. (23) between iterations are (1) the analytic re-computation of all $\left. \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right|_i$ Jacobians and (2) the computation of any needed norms. This simple implementation liberates the details of dynamics from the problem representation. The converged solution is guaranteed to retain the full accuracy of the j^{th} -order fundamental solution expansion because it serves as our transcription scheme. Note that the only non-convexity in this problem formulation necessitating the SCP is the requirement that the overparameterized monomial states lie on the non-Euclidean constraint surface $\mathcal{C}^{(N)}$.

4 Examples

In this section, we showcase the potential use of the developments in this paper for an example application from space vehicle guidance. We hope to establish both practical and theoretical interest in this overparameterized monomial formulation.

4.1 Nonlinear spacecraft rendezvous in LEO – Two-stage guidance

Before showing results with an SCP implementation, we demonstrate a two-stage guidance scheme (linear predictor, nonlinear corrector) for rapid generation of trajectories that are both fuel-efficient and dynamically feasible. The methods in this paper can be applied in practically any dynamical environment, but we first consider the classical rendezvous equations with a target in a simple circular Earth orbit. Using the framework in this paper, we demonstrate the following convenient two-stage guidance strategy:

Stage 1: Compute a low-fidelity solution using the simplified linear model. This can be done using several state-of-the-art approaches, but we choose a convex optimization-based approach that poses the problem as a second-order cone problem (SOCP).

Stage 2: Rapid nonlinear correction, to accommodate for nonlinearity effects, as well as any desired perturbative effects (e.g. J_2 or target orbit eccentricity). This is done by leveraging computation of the high-fidelity fundamental solution matrix $\Psi_j(t)$ and the analytic expression for the monomial vector \mathbf{c}_j at desired order of nonlinearity j .

The nonlinear fundamental solutions data necessary for Stage 2 can be computed in advance and stored on a spacecraft’s guidance computer. With this approach, implementation of Stage 2 is reduced to the use of pre-programmed analytic functions, data lookup, and manageable linear algebra. In settings where many trajectories need to be computed and characterized, pre-computation of the nonlinear fundamental solution data is substantially computationally liberating. For weakly nonlinear cases, this scheme extends the reach of linearized guidance solutions for a small delta-V penalty. For more strongly nonlinear cases, it provides a convenient feasible (but sub-optimal) initial guess for the SCP solver.

4.1.1 Stage 1

First, a suitable linearized rendezvous guidance scheme is defined. The nonlinear dynamics under consideration are those of relative motion in the vicinity of a simple circular Keplerian orbit. This problem is parameterized by the nonlinear equations below for LVLH relative position $\mathbf{r} = (x, y, z)^\top$ and velocity $\mathbf{v} = (\dot{x}, \dot{y}, \dot{z})$ (Schaub and Junkins, 2018):

$$\ddot{x} - 2n\dot{y} - xn^2 - \frac{\mu}{R^2} = -\frac{\mu}{r^3}(R+x) \quad (35a)$$

$$\ddot{y} + 2n\dot{x} - yn^2 = -\frac{\mu}{r^3}y \quad (35b)$$

$$\ddot{z} = -\frac{\mu}{r^3}z \quad (35c)$$

where R is the circular target orbit radius, $r = \sqrt{(R+x)^2 + y^2 + z^2}$ is the chaser’s instantaneous orbit radius, and velocities are measured with respect to the target-centered LVLH frame. In the vicinity of the target spacecraft, Eq. (35) linearizes to the popular Clohessy-Wiltshire (CW) model

(Clohessy and Wiltshire, 1960):

$$\ddot{x} = 2n\dot{y} + 3n^2x \quad (36a)$$

$$\ddot{y} = -2n\dot{x} \quad (36b)$$

$$\ddot{z} = -n^2z \quad (36c)$$

This linearized model admits a simple analytic solution, given below in a special modal form adapted from Burnett and Schaub (2022) but suited to our notation:

$$\mathbf{x}(t) = \hat{\Psi}(t)\hat{\mathbf{c}}_1 \quad (37)$$

$$\hat{\mathbf{c}}_1 = \begin{pmatrix} y_0 - \frac{2}{n}\dot{x}_0 \\ -6x_0 - \frac{3}{n}\dot{y}_0 \\ 3x_0 + \frac{2}{n}\dot{y}_0 \\ \frac{1}{n}\dot{x}_0 \\ \frac{z_0}{2} \\ \frac{1}{2n}\dot{z}_0 \end{pmatrix} \quad (38)$$

$$V = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_6] \\ = \begin{bmatrix} 0 & -2/3 & -1/2 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -n/2 & 0 & 0 \\ 0 & n & n & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & n \end{bmatrix} \quad (39)$$

$$\hat{\psi}_1 = \mathbf{v}_1 \quad (40a)$$

$$\hat{\psi}_2(t) = \mathbf{v}_1 nt + \mathbf{v}_2 \quad (40b)$$

$$\hat{\psi}_3(t) = 2(\mathbf{v}_3 \cos nt - \mathbf{v}_4 \sin nt) \quad (40c)$$

$$\hat{\psi}_4(t) = -2(\mathbf{v}_3 \sin nt + \mathbf{v}_4 \cos nt) \quad (40d)$$

$$\hat{\psi}_5(t) = -2(\mathbf{v}_5 \cos nt + \mathbf{v}_6 \sin nt) \quad (40e)$$

$$\hat{\psi}_6(t) = 2(\mathbf{v}_6 \cos nt - \mathbf{v}_5 \sin nt) \quad (40f)$$

The $\hat{\mathbf{c}}_1$ is a trivial linear transformation of the initial conditions $\mathbf{c}_1 = \mathbf{x}(0)$, i.e. we can write the relationship $\hat{\mathbf{c}}_1 = L\mathbf{c}_1$, where the constant matrix L is known. This facilitates a factorization of the linear problem into projections along a geometrically intuitive functional basis, in lieu of the columns of the STM which are not as intuitive or convenient (Burnett and Schaub, 2022).

The algorithm for solving for a (linearized) optimal impulsive maneuver sequence, minimizing $J = \sum_{i=1}^k \Delta v(t_i)$ at discrete times $t_i \subseteq [t_0, t_1, \dots, t_f]$, is given below for the unperturbed problem (see e.g. Burnett and Schaub (2022) and Guffanti and D'Amico (2018) and references therein):

1. Solve the following second-order cone program for the optimal value $\boldsymbol{\eta}^*$ corresponding to the desired change in the linear modal state $\Delta\mathbf{c}^* = \hat{\mathbf{c}}_{1,\text{goal}} - \hat{\mathbf{c}}_1(t_0)$:

$$\begin{aligned} & \text{maximize} && \tilde{J} = \boldsymbol{\eta}^\top \Delta\mathbf{c}^* \\ & \text{subject to} && \|B_c(t_q)^\top \boldsymbol{\eta}\| \leq 1 \ \forall t_q \in [t_0, t_1, \dots, t_f] \end{aligned} \quad (41)$$

where $B_c(t_q) = \hat{\Psi}^{-1}(t_q)B_x$ with classical linearized control matrix $B_x = [0_{3 \times 3}, I_{3 \times 3}]^\top$.

- Determine all times $t_i \subseteq [t_0, t_1, \dots, t_f]$ for which $|\|B_c(t_i)^\top \boldsymbol{\eta}^* - 1| < \epsilon$ for some small tolerance ϵ . This will yield a k -maneuver sequence, with $k \leq N$, for which the i^{th} impulse is directed along the unit vector:

$$\hat{\boldsymbol{u}}_i = B_c(t_i)^\top \boldsymbol{\eta}^* \quad (42)$$

- The set of delta-V maneuver magnitudes $\{\Delta v_i\}$ needs to satisfy the linear system of equations:

$$\sum_{i=1}^k B_c(t_i) \hat{\boldsymbol{u}}_i \cdot \Delta v_i = \Delta \boldsymbol{c}^* \quad (43)$$

The conditions of optimality require that we reject any solutions with any $\Delta v_i < 0$, and additionally there are some post-processing steps that we can devise to greatly improve the stability and accuracy of this method. These details are discussed in the Appendix B. The solution of this lightweight algorithm will consist of the following:

- A sequence of (linear) monomial states $\boldsymbol{c}_1(\tau_i)$ at $k + 1$ times $\tau_i \in \mathcal{T}$.
- A list of critical times $\mathcal{T} = [t_0, t_{b_1}, t_{b_2}, \dots, t_{b_k}]$, where $\tau_0 = t_0$ is the initial time and $\tau_i = t_{b_i}$ is the i^{th} burn time. With this notation, $\boldsymbol{c}_1(t_0)$ represents the epoch state before the control action of the first burn at $t_{b_i} \geq t_0$ is taken.
- A table of k associated nonzero delta-V vectors.

The outputs of any other guidance algorithm based on linearization and the impulsive maneuver assumptions (see e.g. Berning Jr. et al. (2023) for a recent example of linearized guidance incorporating the practical constraint of passive safety) can also be easily put into this form. Note the use of lower-case “k” instead of “K” from Section 3.1 because for this example each increment of the index is by definition a (nonzero) maneuver.

4.1.2 Stage 2

This scheme is a rapid high-fidelity correction to the solution of Stage 1 which preserves the burn times $t_{b_i} \in \mathcal{T}_b$ and controllable maneuver positions $\boldsymbol{r}(t_{b_i})$, while altering the magnitude and direction of the delta-Vs $\Delta \boldsymbol{v}(t_{b_i})$. The result is a guidance solution compatible with the dynamics parameterized by a given nonlinear fundamental solution representation $\Psi_j(t)$ and associated over-parameterized monomial set \boldsymbol{c}_j . The solution is trustworthy if the nonlinear fundamental solution parameterization $\Psi_j(t)$ is accurate for the flight environment and domain under consideration.

For this scheme simple Newton-style correction is sufficient. We define a free variable vector \boldsymbol{X} in terms of the (first-order) monomials $\boldsymbol{c}_1(t_{b_i})$, and a constraint vector $\boldsymbol{F}(\boldsymbol{X})$, which is driven to zero as the satisfactory \boldsymbol{X}^* is obtained. This function is linear in the higher-order monomial representation $\boldsymbol{c}_j(t_{b_i})$ and nonlinear in the $\boldsymbol{c}_1(t_{b_i})$. We demonstrate now the form of \boldsymbol{X} and $\boldsymbol{F}(\boldsymbol{X})$:

$$\boldsymbol{X} = \left(\boldsymbol{c}_1(t_{b_1})^\top, \boldsymbol{c}_1(t_{b_2})^\top, \boldsymbol{c}_1(t_{b_3})^\top, \dots, \boldsymbol{c}_1(t_{b_k})^\top \right)^\top \quad (44)$$

$$\mathbf{F}(\mathbf{X}) = \begin{pmatrix} \Psi_{r,j}(\tau_2)\mathbf{c}_j(\tau_2) - \mathbf{r}_G(\tau_2) \\ \Psi_{r,j}(\tau_3)\mathbf{c}_j(\tau_3) - \mathbf{r}_G(\tau_3) \\ \vdots \\ \Psi_{r,j}(\tau_k)\mathbf{c}_j(\tau_k) - \mathbf{r}_G(\tau_k) \\ \hline \Psi_{r,j}(\tau_1)\mathbf{c}_j(\tau_1) - \Psi_{r,j}(\tau_1)\mathbf{c}_j(\tau_0) \\ \Psi_{r,j}(\tau_2)(\mathbf{c}_j(\tau_2) - \mathbf{c}_j(\tau_1)) \\ \Psi_{r,j}(\tau_3)(\mathbf{c}_j(\tau_3) - \mathbf{c}_j(\tau_2)) \\ \vdots \\ \Psi_{r,j}(\tau_k)(\mathbf{c}_j(\tau_k) - \mathbf{c}_j(\tau_{k-1})) \\ \hline \Psi_{v,j}(\tau_k)\mathbf{c}_j(\tau_k) - \mathbf{v}_G(\tau_k) \end{pmatrix} \quad (45)$$

where the nonlinear mapping from $\mathbf{c}_1(\tau_i)$ (and hence from \mathbf{X}) to $\mathbf{c}_j(\tau_i)$ is analytic and straightforward – recall the discussion in Section 2.2. The first set of constraints (above the dashed line) are enforcing that all controllable maneuver locations match the corresponding maneuver location from the guidance solution of Stage 1, $\mathbf{r}_G(\tau_i)$. Because the first maneuver location is not controllable, it is omitted from these constraints. Note that if $\tau_k < t_f$, $\mathbf{x}_G(\tau_k)$ (the state immediately after at the k^{th} and final burn) is related to the goal final condition $\mathbf{x}(t_f)$ simply by the natural dynamics mapping $\mathbf{x}(\tau_k) = \boldsymbol{\psi}^{-1}(\mathbf{x}(t_f), t_f, \tau_k)$. The second set of constraints, between dashed lines, enforce the kinematic constraint that changes in the monomial states at burn nodes (implicitly the result of impulsive maneuvers) should not instantaneously change the position. Note that the very first one is written differently to emphasize that $\mathbf{c}_j(\tau_0)$ is uncontrollable and thus does not appear in the free variables \mathbf{X} . Lastly, the final listed constraint in $\mathbf{F}(\mathbf{X})$ is enforcing that the final goal velocity, immediately after the k^{th} and final burn, is achieved.

The constraint Jacobian, $G(\mathbf{X}) = \frac{\partial}{\partial \mathbf{X}}(\mathbf{F}(\mathbf{X}))$, is computed analytically, and its derivation from Eq. (45) is straightforward:

$$G(\mathbf{X}) = \begin{bmatrix} \mathbf{0}_{\frac{N}{2} \times N} & \Psi_{r,j}(\tau_2)C_{j,2} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} \\ \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \Psi_{r,j}(\tau_3)C_{j,3} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} \\ & & \vdots & & & & \\ \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \Psi_{r,j}(\tau_k)C_{j,k} \\ \hline \Psi_{r,j}(\tau_1)C_{j,1} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} \\ -\Psi_{r,j}(\tau_2)C_{j,1} & \Psi_{r,j}(\tau_2)C_{j,2} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} \\ \mathbf{0}_{\frac{N}{2} \times N} & -\Psi_{r,j}(\tau_3)C_{j,2} & \Psi_{r,j}(\tau_3)C_{j,3} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} \\ & & \vdots & & & & \\ \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & -\Psi_{r,j}(\tau_k)C_{j,k-1} & \Psi_{r,j}(\tau_k)C_{j,k} \\ \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \mathbf{0}_{\frac{N}{2} \times N} & \cdots & \mathbf{0}_{\frac{N}{2} \times N} & \Psi_{v,j}(\tau_k)C_{j,k} \end{bmatrix} \quad (46)$$

where $C_{j,i} = \left. \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1} \right|_{\mathbf{c}_1(\tau_i)}$ which can be computed analytically by exploiting the known monomial structure of the \mathbf{c}_j . This $kN \times kN$ matrix is full rank for our problem of interest. Thus the Newton step can be assessed as below:

$$\delta \mathbf{X} = -\gamma G^{-1} \mathbf{F}(\mathbf{X}), \quad 0 < \gamma \leq 1 \quad (47)$$

The proper γ can, if needed, be chosen by evaluating the error norm of $\mathbf{F}(\mathbf{X} + \delta \mathbf{X})$ for a pre-determined sequence of γ , i.e. in a line-search, because of the extremely low overhead in computing

$\mathbf{F}(\mathbf{X})$. For our examples, the simple choice $\gamma = 1$ was always sufficient. This linear problem is initialized by constructing $\mathbf{X}^{(0)}$ (i.e. the 0th iteration) directly from the $\mathbf{c}_1(\tau_i)$ outputs from Stage 1 of guidance. In this particular example, because Stage 1 solves for a sequence of modified constants $\hat{\mathbf{c}}_1(\tau_i)$, to construct $\mathbf{X}^{(0)}$ each must first be linearly mapped to the initial guess of the corresponding $\mathbf{c}_1(\tau_i)$. This is accomplished by the following equation:

$$\mathbf{c}_1(\tau_i) = \Psi_1^{-1}(\tau_i) \hat{\Psi}(\tau_i) \hat{\mathbf{c}}_1(\tau_i) \quad (48)$$

In this manner, the influence of any desired perturbations not modeled in Stage 1 can be introduced by ensuring their linear effects are accounted for in Ψ_1 . Because there is no need for any complicated numerics in the analytic computations of Eqs. (46) and (47), Stage 2 is extremely rapid. In our experience, the total time taken by all needed successive corrections of Stage 2 are much faster than the small SOCP solved by stage 1.

We now apply the aforementioned procedure to an example short-range low-Earth orbit (LEO) rendezvous and proximity operations scenario. For this example, a linearized rendezvous trajectory is generated which requires correction for the nonlinear dynamics of relative motion in low-Earth orbit. The problem details are provided in Table 1. Also included in that table are the runtime details for generation of the nonlinear fundamental solutions via the numerical propagation of the STTs. Note that in lieu of using numerically propagated STTs, for this particular problem there exist analytic nonlinear expansions in literature. In particular, exact nonlinear expansions suitable for computing $\Psi_j(t)$ up to order $j = 3$ can be found (in normalized form) in Butcher et al. (2016). See also Butcher et al. (2017) and Willis et al. (2019b) and also references therein. Such analytic solutions unburden onboard algorithms from the task of pre-computing STTs to render the nonlinear fundamental solutions. More details can be found in Appendix A.

Table 1: Parameters for Rendezvous Example 1

Parameter	Values
Target spacecraft	LEO orbit, $a = 6378$ km, $e = 0$, $T \approx 1.4$ hrs
Control interval	$t_0 = 0.1T$, $t_f = 2.3T$, discretized into 220 times
Relative state, $t = 0$	$\mathbf{x}(0) = [-1266.6, -12000, 1000, 0, 2.9748, 0]$ (units: m, m/s)
Relative state, t_f	$\mathbf{x}(t_f) = [-589.6, 383.2, -1825.9, 2.3747, 1.4617, -1.3499]$
STT info	Duration: $2.3T$. Discretization: 230 times. Size: 474 kb Compute time (order): 129s (3), 6.69s (2), 0.747s (1)
Linear guidance	Burn indices: [0, 93, 142, 201, 219]. Delta-V: 2.336 m/s. Runtime: 0.108s
Stage 2 guidance	Delta-V: 2.353 m/s. Runtime: 0.0119s

Figure 4 gives the nominal guidance solution produced by Stage 1, propagated with the linearized dynamics assumed therein (in this case, the CW dynamics). This is generated on a 2023 MacBook Pro (M2 chip) in Python 3, using CVXPY (Diamond and Boyd, 2016; Agrawal et al., 2018) with ECOS (Domahidi et al., 2013) to solve the SOCP in ~ 0.1 s. After a sequence of 5 burns, the goal state is achieved: entry onto a tilted target-centered relative orbit. This linearized guidance solution requires a maneuver sequence with a total delta-V of 2.336 m/s. In subsequent figures the nominal linear guidance solution appears as a gray curve.

To produce Figure 5, the original Stage 1 guidance solution (nominal delta-Vs applied at pre-planned times) is followed in an open-loop fashion subject to the nonlinear dynamics, with no allowance for corrective maneuvers – yielding the orange curve. The goal state is clearly not

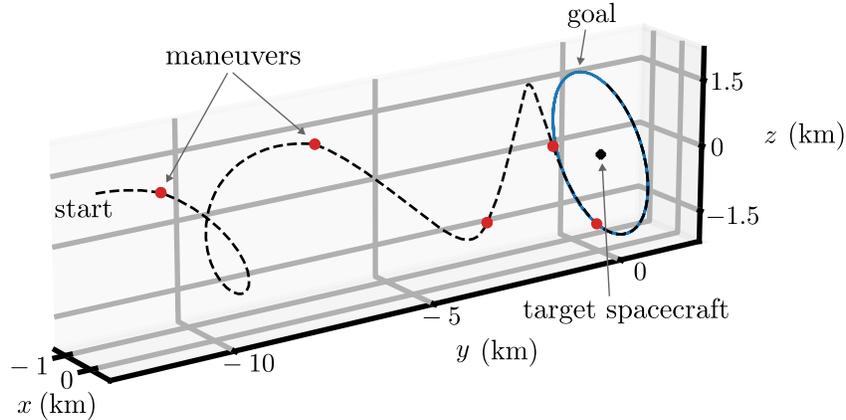


Figure 4: Nominal linear guidance trajectory, linearized dynamics (rendezvous ex. 1)

achieved. This illustrates a well-known limitation of using linearized dynamics for rendezvous guidance: the nominal trajectories predicted by linearization will have to be corrected for accuracy in a high-fidelity model. There are various means of doing this, such as refinement via successive convexification, post-processing with sequential Lambert targeting (restricted to two-body problems), or correction via multiple-shooting schemes. These come at some non-trivial computational cost that must be paid each time a new trajectory is devised.

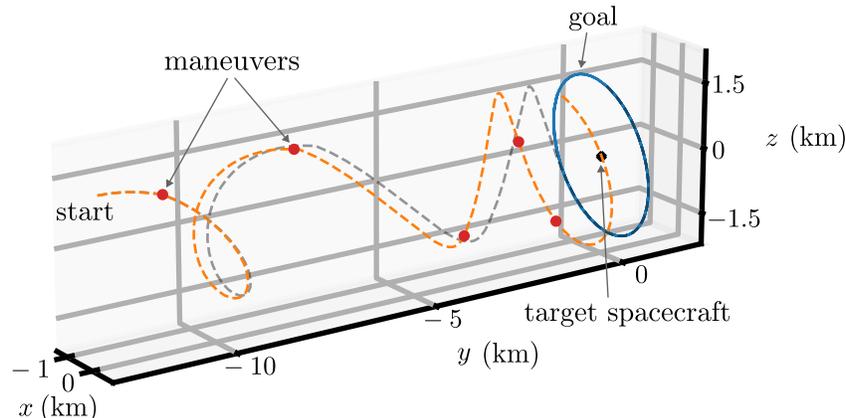


Figure 5: Executed open-loop linear guidance, nonlinear dynamics (rendezvous ex. 1)

In Figure 6, we show the results of Stage 2. This correction is achieved in ~ 0.01 s with 2 iterations of the Newton solver scheme (~ 0.005 s per iteration) – making use of analytic monomial expressions and the pre-saved $\Psi_j(t)$ evaluated at the critical times. The resulting nominal trajectory is again simulated in open-loop with the nonlinear dynamics, but now satisfies the final conditions to a high degree of accuracy (0.37% error in final along-track position, $< 0.1\%$ error in all other states).

In our tests, we find that this two-stage guidance scheme is quite stable, even when the linearized initial guess (Stage 1) is highly erroneous. One more LEO example with the two-stage guidance scheme serves as a nice illustration of this. We consider a rendezvous from an initial along-track offset of 75 km – well beyond the range of validity of the CW problem (especially in its Cartesian instead of curvilinear representation). The problem details for this example are provided in Table 2,

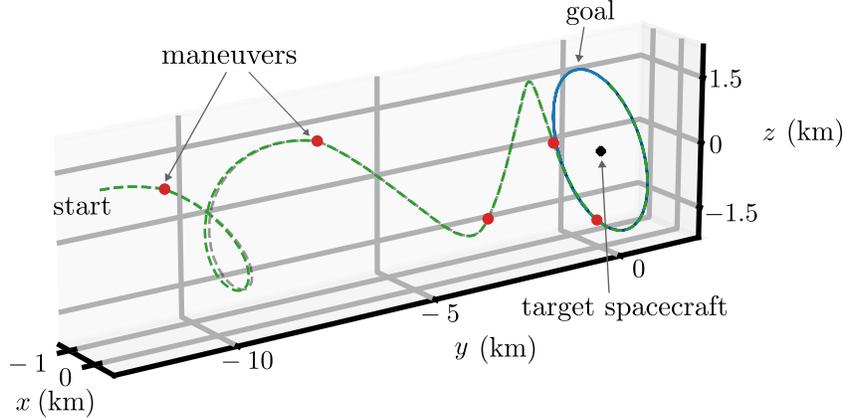


Figure 6: Executed open-loop nonlinear guidance, nonlinear dynamics (rendezvous ex. 1)

and we reuse the same $\Psi_j(t)$ computed for the prior example. Figure 7 gives the open-loop execution of the delta-Vs predicted by linearized guidance solution with the nonlinear orbital dynamics, which completely fails to reach the target point, due to the long range extending beyond the linear regime.

Table 2: Parameters for Rendezvous Example 2

Parameter	Values
Target spacecraft	LEO orbit, $a = 6378$ km, $e = 0$, $T \approx 1.4$ hrs
Control interval	$t_0 = 0.1T$, $t_f = 2.3T$, discretized into 220 points
Relative state, $t = 0$	$\mathbf{x}(0) = [-9000, -75000, 1000, 0, 17.353, 0]$ (units: m, m/s)
Relative state, t_f	$\mathbf{x}(t_f) = [0, 500, 0, 0, 0, 0]$
STT info	Reused from Example 1
Linear guidance	Burn indices: [17, 62, 117, 163, 215]. Delta-V: 5.112 m/s. Runtime: 0.105s Open-loop error: > 30 km range, trajectory failed
Stage 2 guidance	Delta-V: 8.124 m/s. Runtime: 0.018s. Open-loop t_f error: 1.06 km, 6.5 cm/s

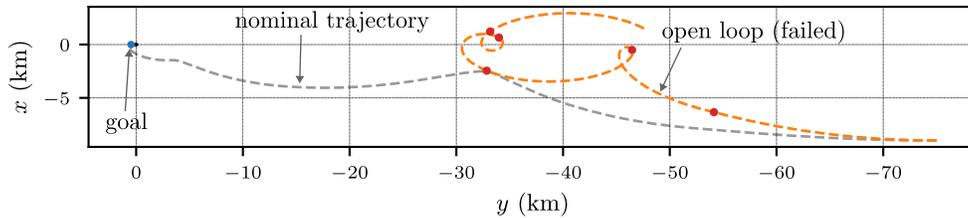


Figure 7: Executed open-loop linear guidance, nonlinear dynamics (rendezvous ex. 2)

Figure 8 shows the open-loop result achieved using the delta-Vs predicted by Stage 2 guidance, which converged in 3 Newton steps, with a total time of ~ 0.018 seconds to compute the correction in Python 3. The result is a dramatic improvement on the linearized guidance solution, rendering a reasonable reference trajectory, achieving a sub-km error in the targeting of the final along-track point in open-loop execution. The residual error can easily be accommodated with a small correction to the penultimate burn. This small error however illustrates the important point that the guidance solutions produced by Stage 2 will only be as accurate as the nonlinear solution expansions $\Psi_j(t)$ that were used to generate them. In this case, at 75 km of range and 2.2 target orbit periods of

total duration, the linearized CW model is totally invalid but the third-order expansion is starting to show only some error.

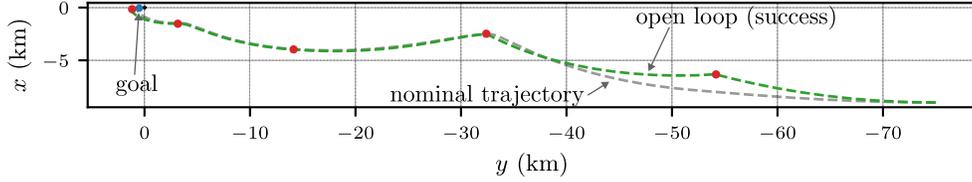


Figure 8: Executed open-loop two-stage guidance, nonlinear dynamics (rendezvous ex. 2)

4.2 Nonlinear spacecraft rendezvous in LEO – Successive convexification-based optimal guidance

Now we provide a simple demonstration of the successive convexification scheme introduced in Section 3.1. For this example we consider a new LEO rendezvous case whose details are provided in Table 3. This example combines long-range along-track separation with some out-of-plane motion which must be regulated to achieve a final stationary along-track state with 1.5 km offset from the target spacecraft. As before, we first compute a linearized guidance solution (Stage 1) and then a nonlinear guidance solution (now via the SCP scheme). The linearized guidance solution, and its failed open-loop execution, are depicted in Figure 9. This solution is computed in ~ 0.05 s.

Table 3: Parameters for Rendezvous Example 3a (Min. sum of delta-V squared, $\mathcal{P} = 2$)

Parameter	Values
Target spacecraft	LEO orbit, $a = 6378$ km, $e = 0$, $T \approx 1.4$ hrs
Control interval	$t_0 = 0.1T$, $t_f = 1.1T$, discretized into 4 points (burn times fixed)
Relative state, $t = 0$	$\mathbf{x}(0) = [-3666.7, -62000, -4000, -1.239, 7.437, 2.479]$ (units: m, m/s)
Relative state, t_f	$\mathbf{x}(t_f) = [0, 1500, 0, 0, 0, 0]$
STT info	Reused from Example 1
Linear guidance	Burn indices: $[0, 12, 64, 99]$. Delta-V: 10.04 m/s. Runtime: 0.052s Open-loop error: >10 km range, trajectory failed
SCP guidance	Delta-V: 10.82 m/s. Runtime: 0.1s. Open-loop t_f error: 0.0529 km, 6.0 cm/s
SCP parameters	Fixed trust region, $d = 3$. Slack var. penalty weight $w = 20$. Method: ECOS Convergence condition: $\ \tilde{\mathbf{X}}\ < 10^{-4}$, No. iterations needed: 5

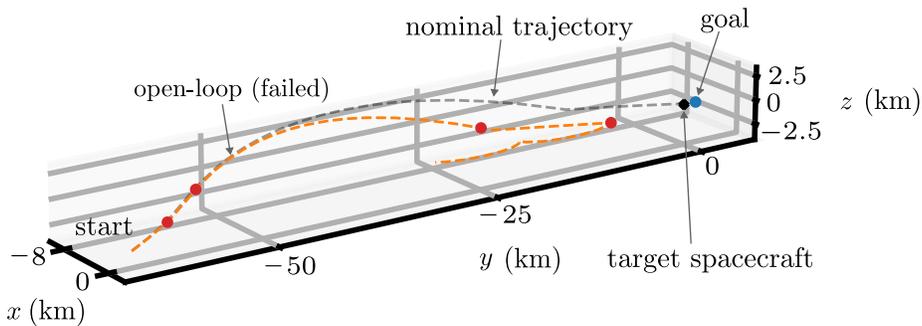


Figure 9: Executed open-loop linear guidance, nonlinear dynamics (rendezvous ex. 3a)

Figure 10 gives the nonlinear guidance solution obtained via the SCP framework, minimizing sum of delta-V norm squared, thus $\mathcal{P} = 2$, where the burn nodes and delta-V vectors are allowed to change via implementation of the scheme discussed in Section 3.1. To facilitate the most rapid solution we inherit the optimal burn times of the linearized guidance solution, which minimizes the dimensionality of decision variables (although the SCP framework allows for the times to be optimized as well). The SCP solution, which is generated using the linearized guidance solution as an initial guess, converges in 5 iterations in a total runtime of ~ 0.1 s, with a runtime per iteration of ~ 0.02 s. The total solve time (linear SOCP + nonlinear SCP) to generate this nonlinear guidance solution was thus ~ 0.15 s. The resulting guidance solution requires 10.82 m/s of total delta-V, compared to the linearized prediction of 10 m/s, and a two-stage guidance requirement of 11.2 m/s. Figures 11 and 12 give the norm of the (non-slack) free variables, $\|\tilde{\mathbf{X}}\|$, and the total cost

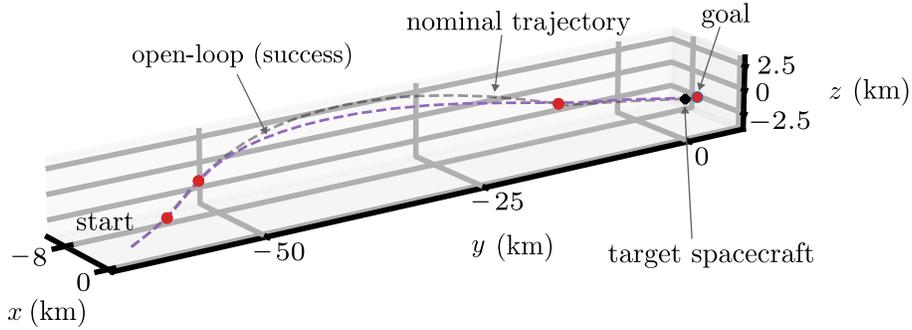


Figure 10: Executed open-loop SCP-based guidance, nonlinear dynamics (rendezvous ex. 3a)

J vs. iteration number. The steady drop in the norm of the free variables per iteration, as well as decreasing cost, correspond with our expectations of nominal behavior of the SCP scheme converging to a feasible minimum. For this problem, the norm of the vector of slack variables, $\|\mathbf{S}\|$, never exceeded 10^{-6} .

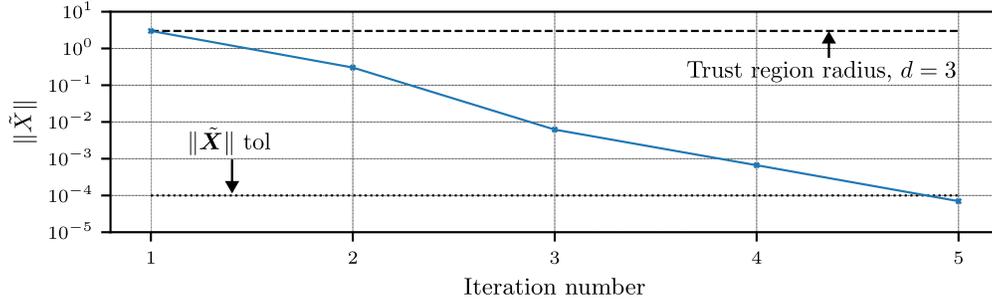


Figure 11: SCP norm of non-slack free vars. vs. iter no. (rendezvous ex. 3a)

For this same problem, we also compute via SCP the $\mathcal{P} = 1$ delta-V minimizing solution, and in addition we allow for the burn times to be selected from 100 candidate discretization points. The details for the new solver scheme are given in Table 4. The final SCP solution still has only 4 maneuvers, but the timing of the inner two is shifted slightly. The other discretization times are automatically identified (as a by-product of the geometrically optimal path on $\mathcal{C}^{(N)}$) to be sub-optimal burn times, so the optimal result gives only 4 non-null jumps in the $\mathbf{c}_j(t_i)$ across the 100 discretizations of the path. Note the increase in runtime to 2.5s as a result of the twenty-five

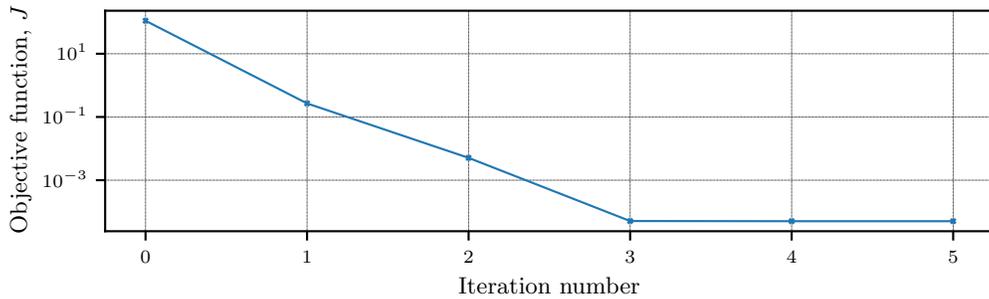


Figure 12: SCP total cost vs. iter no. (rendezvous ex. 3a)

fold expansion of the dimensionality of free variables for this example. The nominal delta-V is also slightly reduced. Figure 13 gives the new guidance solution, with the SCP solution in purple, the discretization of the problem shown by the gray dots (showing only 50 of the 100 points for clarity), and the selected burn nodes shown as red dots.

Table 4: Parameters for Rendezvous Example 3b (Min. sum of delta-V, $\mathcal{P} = 1$)

Parameter	Values
Target spacecraft	LEO orbit, $a = 6378$ km, $e = 0$, $T \approx 1.4$ hrs
Control interval	$t_0 = 0.1T$, $t_f = 1.1T$, discretized into 100 points (burn times free)
Linear vs. SCP	Burn indices: Linear, [0, 12, 64, 99]. SCP solution, [0, 13, 65, 99]
SCP guidance	Delta-V: 10.73 m/s. Runtime: 2.5s. Open-loop t_f error: 0.108 km, 14.6 cm/s
SCP parameters	Fixed trust region, $d = 10$. Slack var. penalty weight $w = 5$. Method: ECOS Convergence condition: $\ \tilde{\mathbf{X}}\ < 10^{-4}$, No. iterations needed: 6

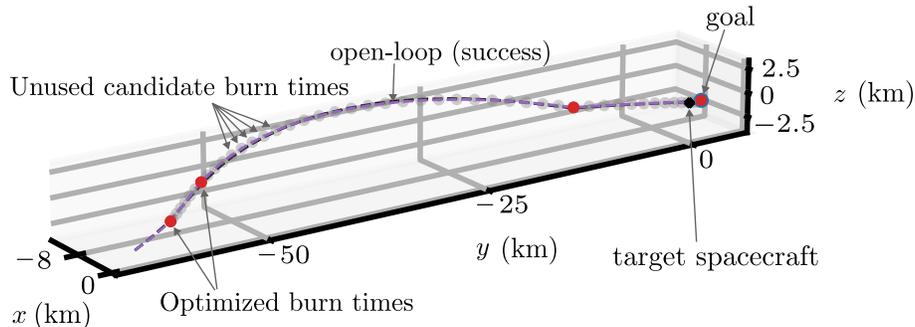


Figure 13: Executed open-loop SCP-based guidance, nonlinear dynamics (rendezvous ex. 3b)

All guidance examples in this paper were generated using the same pre-computed third order nonlinear expansion, and represent a small sample of the trade space that can be explored leveraging that same data. The two-stage nonlinear guidance and SCP-based nonlinear guidance implementations all perform without the need for any real-time numerical integration, and their robustness to starting with a poor linearization-based initial guess is encouraging and interesting. Both offer nonlinear correction in a comparable amount of time that was required to generate the linearized initial guess, but of course the SCP-based implementation comes with the added delta-V minimizing benefit. Note that our implementation in Python is not optimized for speed but for proof of

concept, and a `CVXPYgen` implementation of the SCP could potentially improve performance by removing significant overhead from successive `CVXPY` calls (Schaller et al., 2022).

5 Conclusions

This paper introduces a method of rapid guidance for nonlinear systems leveraging overparameterized monomial coordinates and fundamental solution expansions. By this methodology, trajectory optimization problems can be solved with very low real-time computational footprint. The solutions inherit the accuracy of the nonlinear fundamental solution expansions encoded by the the special nonlinear fundamental solution matrix $\Psi_j(t)$. We devise a simple two-stage (linear predict, nonlinear correct) approach and a simple successive convexification scheme to demonstrate the usefulness of the methodology. The spacecraft relative motion and rendezvous problem (and relatedly, orbit regulation) are particularly well-suited for this approach because the reference trajectory needed for the nonlinear solution expansions is clear: the target spacecraft orbit. We present several examples where our methodology is used to rapidly compute guidance solutions that are effective when propagated in a numerical model of nonlinear problem dynamics. In general, like linear methods, this method is still only “locally” valid, though admitting a much larger region of validity than linearization. For this reason, application to more general orbit transfer problems requires the computation of reasonable reference trajectories for the transfer, which are not always clear. It’s also worth noting that, depending on order of nonlinearity j , the time to compute the fundamental solutions numerically can be non-negligible. We envision a GNC implementation where these solutions are computed hours, days, or more in advance of when needed in a mission. In the context of our chosen example problem of spacecraft rendezvous, incorporation of the analytic fundamental solution expansions (in lieu of our numerically computed expansions) keeps the total computational footprint extremely low (solve times of $\mathcal{O}(1s)$ or shorter for our examples).

This is an early showcasing of a new methodology, so the opportunities for future work are significant. First, it is possible to extend this work to trajectory optimization problems with continuous thrust, the kinematic constraints of which were already outlined. Incorporating path constraints is straightforwardly inherited from more traditional implementations into our methodology because the relationship between the instantaneous state vector (or functions of the state vector) and its overparameterized monomial expression is linear. There are some other future developments which require more theoretical work. First, the error analysis outlined in this paper for problems transcribed in $\mathcal{C}^{(N)}$ can and should be extended so that formal guarantees on performance can be made. In this work, we merely provide early arguments towards that end, rooted in the fact that the feasible region for accurate use of our method is static. In reality this feasible region can be described in \mathbb{R}^N because the \mathbf{c}_j are functionally generated from \mathbf{c}_1 . Relatedly, more advanced methods for sizing the trust region(s) should be explored, and indeed the trust region can be chosen to ensure the guidance solution remains in the region of validity of its predictions. Extending this framework to accommodate stochasticity is also desirable for robust onboard applications. There is a path forward to this, because the nonlinear fundamental solutions can also be expanded in terms of uncertain parameters.

Acknowledgements

This work was performed as part of the Marie Skłodowska-Curie Actions (MSCA) Postdoctoral Fellowship, and funded by the European Commission. It is further made possible by technical

and administrative support from the hosting organization: Department of Aerospace Science and Technology (DAER), Politecnico di Milano, Italy.

6 Code and Data Availability

Data and links to code to reproduce select examples from this paper will be made available via the open-access digital repository Zenodo at <https://zenodo.org/communities/faast-msca/about> as part of the MSCA project “Facilitating Autonomy in Astrodynamics for Spacecraft Technology”.

References

- Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018. doi:10.1080/23307706.2017.1397554.
- Andrew W. Berning Jr., Ethan R. Burnett, and Stefan Bieniawski. Chance-constrained, drift-safe guidance for spacecraft rendezvous. In *AAS Rocky Mountain Guidance, Navigation, and Control Conference*. American Astronautical Society, 2023. doi:10.48550/arXiv.2401.11077.
- Martin Berz. Chapter 2 - Differential Algebraic Techniques. In Peter Hawkes, editor, *Modern Map Methods in Particle Beam Physics*, volume 108 of *Advances in Imaging and Electron Physics*, pages 81–117. Elsevier, 1999. doi:10.1016/S1076-5670(08)70228-3.
- Spencer Boone and Jay McMahon. Orbital Guidance Using Higher-Order State Transition Tensors. *Journal of Guidance, Control, and Dynamics*, 44(3):493–504, 2021. doi:10.2514/1.G005493.
- Spencer Boone and Jay McMahon. Directional State Transition Tensors for Capturing Dominant Nonlinear Effects in Orbital Dynamics. *Journal of Guidance, Control, and Dynamics*, 46(3): 431–442, March 2023. doi:10.2514/1.G006910.
- Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, 2004.
- E. R. Burnett, E. A. Butcher, A. J. Sinclair, and T. A. Lovell. Linearized Relative Orbital Motion Model About an Oblate Body Without Averaging, AAS 18-218. *Advances in the Astronautical Sciences*, 167(AAS 18-218):691–710, 2018.
- Ethan R. Burnett and Hanspeter Schaub. Spacecraft relative motion dynamics and control using fundamental modal solution constants. *Journal of Guidance, Control, and Dynamics*, 45(10): 1786–1799, 2022. doi:10.2514/1.G006603.
- Eric A. Butcher, T. Alan Lovell, and Andrew Harris. Third order cartesian relative motion perturbation solutions for slightly eccentric chief orbits. In *AAS/AIAA Spaceflight Mechanics Meeting*, volume 158, pages 3435 – 3454, 2016. URL <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85007356529&partnerID=40&md5=d2756042f95ce0dc7988721b36b77d3c>.
- Eric A. Butcher, Ethan R. Burnett, and T. Alan Lovell. Comparison of Relative Orbital Motion Perturbation Solutions in Cartesian and Spherical Coordinates. In *AAS/AIAA Spaceflight Mechanics Meeting*, number AAS 17-202. American Astronautical Society, 2017.
- S. Casotto. The Equations of Relative Motion in the Orbital Reference Frame. *Celestial Mechanics and Dynamical Astronomy*, 124(3):215–234, 2016. doi:10.1007/s10569-015-9660-1.

- W. H. Clohessy and R. S. Wiltshire. Terminal Guidance System for Satellite Rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, Sept. 1960. doi:10.2514/8.8704.
- Gianfranco Di Domenico, Eleonora Andreis, Andrea Morelli, Gianmario Merisio, Vittorio Franzese, Carmine Giordano, Alessandro Morselli, Paolo Panicucci, Fabio Ferrari, and Francesco Topputo. Toward Self-Driving Interplanetary CubeSats: the ERC-Funded Project EXTREMA. In *International Astronautical Congress*, number IAC-21,D4,1,1,x65902, 2021.
- P. Di Lizia, R. Armellin, A. Morselli, and F. Bernelli-Zazzera. High order optimal feedback control of space trajectories with bounded control. *Acta Astronautica*, 94(1):383–394, 2014. ISSN 0094-5765. doi:10.1016/j.actaastro.2013.02.011.
- Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016. doi:10.48550/arXiv.1603.00943.
- Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: An SOCP solver for embedded systems. pages 3071–3076, 07 2013. doi:10.23919/ECC.2013.6669541.
- E. Gilbert and G. Harasty. A class of fixed-time fuel-optimal impulsive control problems and an efficient algorithm for their solution. *IEEE Transactions on Automatic Control*, 16(1):1–11, 1971. doi:10.1109/TAC.1971.1099656.
- A. Giorgilli and M. Sansottera. Methods of algebraic manipulation in perturbation theory. *Workshop Series of the Asociacion Argentina de Astronomia*, 3:147–183, January 2011. doi:10.48550/arXiv.1303.7398.
- Cristian Greco, Marilena Di Carlo, Massimiliano Vasile, and Richard Epenoy. Direct multiple shooting transcription with polynomial algebra for optimal control problems under uncertainty. *Acta Astronautica*, 170:224–234, 2020. doi:10.1016/j.actaastro.2019.12.010.
- Tommaso Guffanti and Simone D’Amico. Integration constants as state variables for optimal path planning. In *2018 European Control Conference (ECC)*, pages 1–6, 2018. doi:10.23919/ECC.2018.8550448.
- E. J. Hinch. *Perturbation Methods*. Cambridge University Press, Cambridge, England, 1991. doi:10.1017/CBO9781139172189.
- Christian Hofmann and Francesco Topputo. Rapid Low-Thrust Trajectory Optimization in Deep Space Based on Convex Programming. *Journal of Guidance, Control, and Dynamics*, 44(7):1379–1388, 2021. doi:10.2514/1.G005839.
- Christian Hofmann, Andrea C. Morelli, and Francesco Topputo. *On the Performance of Discretization and Trust-Region Methods for On-Board Convex Low-Thrust Trajectory Optimization*. Number 1892. American Institute of Aeronautics and Astronautics, 2022. doi:10.2514/6.2022-1892.
- Christian Hofmann, Andrea C. Morelli, and Francesco Topputo. Performance assessment of convex low-thrust trajectory optimization methods. *Journal of Spacecraft and Rockets*, 60(1):299–314, 2023. doi:10.2514/1.A35461.
- Dario Izzo, Francesco Biscani, Carlos Sánchez, Jörg Müller, and Mike Heddes. darioizzo/audi: Update to the new obake API and tbb API (v1.9.2). Zenodo, June 2022.
- Àngel Jorba. A Methodology for the Numerical Computation of Normal Forms, Centre Manifolds

- and First Integrals of Hamiltonian Systems. *Experimental Mathematics*, 8(2):155–195, 1999. doi:10.1080/10586458.1999.10504397.
- Matthew Kelly. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Review*, 59(4):849–904, 2017. doi:10.1137/16M1062569.
- E. B. Lee and L. Markus. *Foundations of Optimal Control Theory*. Wiley, New York, 1967.
- Yuanqi Mao, Daniel Dueri, Michael Szmuk, and Behçet Açıkmeşe. Successive Convexification of Non-Convex Optimal Control Problems with State Constraints. *IFAC-PapersOnLine*, 50(1):4063–4069, 2017. ISSN 2405-8963. doi:10.1016/j.ifacol.2017.08.789. 20th IFAC World Congress.
- Andrea Carlo Morelli, Christian Hofmann, and Francesco Topputo. Robust Low-Thrust Trajectory Optimization Using Convex Programming and a Homotopic Approach. *IEEE Transactions on Aerospace and Electronic Systems*, 58(3):2103–2116, 2022. doi:10.1109/TAES.2021.3128869.
- A.H. Nayfeh. *Perturbation Methods*. Wiley, 2000. doi:10.1002/9783527617609.
- Kenshiro Oguri. Successive Convexification with Feasibility Guarantee via Augmented Lagrangian for Non-Convex Optimal Control Problems. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 3296–3302, 2023. doi:10.1109/CDC49753.2023.10383462.
- Kenshiro Oguri and Jay W. McMahon. Robust Spacecraft Guidance Around Small Bodies Under Uncertainty: Stochastic Optimal Control Approach. *Journal of Guidance, Control, and Dynamics*, 44(7):1295–1313, 2021. doi:10.2514/1.G005426.
- Ryan S. Park and Daniel J. Scheeres. Nonlinear Mapping of Gaussian Statistics: Theory and Applications to Spacecraft Trajectory Design. *Journal of Guidance, Control, and Dynamics*, 29(6):1367–1375, 2006. doi:10.2514/1.20177.
- John Prussing. *Primer Vector Theory and Applications*, chapter 2, pages 16–36. Cambridge University Press, 08 2010. ISBN 9780521518505. doi:10.1017/CBO9780511778025.003.
- Christopher W. T. Roscoe, Jason J. Westphal, Jacob D. Griesbach, and Hanspeter Schaub. Formation establishment and reconfiguration using differential elements in J2-perturbed orbits. In *2014 IEEE Aerospace Conference*, pages 1–19, 2014. doi:10.1109/AERO.2014.6836272.
- Maximilian Schaller, Goran Banjac, Steven Diamond, Akshay Agrawal, Bartolomeo Stellato, and Stephen Boyd. Embedded Code Generation With CVXPY. *IEEE Control Systems Letters*, 6:2653–2658, 2022. doi:10.1109/LCSYS.2022.3173209.
- Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA, 4th edition, 2018. doi:10.2514/4.105210.
- Joseph A. Starek, Behçet Açıkmeşe, Issa A. Nesnas, and Marco Pavone. *Spacecraft Autonomy Challenges for Next-Generation Space Missions*, pages 1–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-47694-9. doi:10.1007/978-3-662-47694-9_1.
- M. Valli, R. Armellin, P. Di Lizia, and M. R. Lavagna. Nonlinear Mapping of Uncertainties in Celestial Mechanics. *Journal of Guidance, Control, and Dynamics*, 36(1):48–63, 2013. doi:10.2514/1.58068.
- Zhenbo Wang and Michael J. Grant. Minimum-fuel low-thrust transfers for spacecraft: A convex approach. *IEEE Transactions on Aerospace and Electronic Systems*, 54(5):2274–2290, 2018. doi:10.1109/TAES.2018.2812558.

Matthew Willis, Kyle T. Alfriend, and Simone D’Amico. Second-order solution for relative motion on eccentric orbits in curvilinear coordinates. In *AAS/AIAA Astrodynamics Specialist Conference*, number AAS 19-810. American Astronautical Society, 2019a.

Matthew Willis, T. Alan Lovell, and Simone D’Amico. Second-Order Analytical Solution for Relative Motion on Arbitrarily Eccentric Orbits. In *AAS/AIAA Spaceflight Mechanics Meeting*, number 19-364 in *Advances in the Astronautical Sciences*. Univelt, January 2019b.

Wolfram Research, Inc. *Mathematica*, Version 13.3, 2023. URL <https://www.wolfram.com/mathematica>. Champaign, IL, 2024.

A Generating and Formatting the Mixed Monomials and the Non-linear Fundamental Solution Expansions

A.1 Computerized generation and manipulation of monomial representations

To implement our methodology, one needs a well-ordered and unambiguous way of constructing and manipulating monomial sequences. We use a positional power representation of monomials and we represent sequences of monomials in 2D arrays. The easiest way to explain this is with a full end-to-end example, but Giorgilli and Sansottera (2011) and Jorba (1999) give more context on similar techniques. Here we drop the initial condition notation “ $x_i(0)^j$ ” in favor of the more compact x_i^j . Consider a 3-state system with states x_1, x_2, x_3 . The array representation of the monomial $x_1^a x_2^b x_3^c$ for integer powers $a, b, c \geq 0$ is $[a, b, c]$. These are stacked row-wise to reproduce sequences of monomials as needed. For example, we represent $\mathbf{c}_1 = \mathbf{x} = (x_1, x_2, x_3)^\top$ as the following array:

$$\text{arr}(\mathbf{c}_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (49)$$

The following functional code snippet constructs all rows of a desired $\text{arr}(\mathbf{c}_j)$ for a system with N states by initializing $\text{arr}(\mathbf{c}_1)$ and then, order-by-order, appending the array representation (itself row-by-row) for higher-order terms up to and including nonlinearity order j . We take the array representation of a prior order (“`base_block`”) and generate all unique monomial power permutations for a new order (“`newblock`”), then step to the next order with `base_block` \leftarrow `newblock`.

```
import numpy as np
Identity_N = np.identity(N, dtype='int ')
Kj = compute_Kj(N, j)

# Create arr(c1):
c_vec_array = np.identity(N, dtype='int ')

# Extend c_vec_array to create arr(cj):
base_block = c_vec_array[0:N,0:N] # Initialize
k_prev = base_block.shape[0] # No. rows of base_block
for idx0 in range(j-1): # This order
    for idx1 in range(N): # This state
        term_mult = Identity_N[idx1,:]
        for idx2 in range(k_prev): # Which row of base_block?
            new_row = base_block[idx2,:] + term_mult # Make candidate
```

```

if idx1 == 0 and idx2 == 0:
    newblock = new_row.astype('int') # Initialize block
else:
    if not (new_row.tolist() in newblock.tolist()):
        newblock = np.row_stack([newblock, new_row])
c_vec_array = np.row_stack([c_vec_array, newblock]) # New order done
base_block = newblock # Step
k_prev = base_block.shape[0]

```

As an example output, we provide explicitly the third-order array representation of \mathbf{c}_j for the case of $N = j = 3$ below, with sub-blocks of orders 1, 2, and 3 partitioned for clarity:

$$\text{arr}(\mathbf{c}_3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ \hline 3 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 2 \\ 0 & 3 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{bmatrix} \quad (50)$$

where e.g. the fourth row (first second-order component) is x_1^2 , the eighth row is x_2x_3 , and the eleventh is $x_1^2x_2$. Note that for nonlinearities of order q , it can be shown that the resultant sub-block will have the following number of terms:

$$\dim(\text{sub-block } q) = \binom{N+q-1}{q} \quad (51)$$

Adding up the number of rows of all sub-blocks to the max order j , we recover Eq. (3).

With a computerized array representation of \mathbf{c}_j , the column-wise arrangement of all fundamental solutions to make Ψ_j is also specified. We defer the reader to our code for any more details in that regard. We now briefly discuss how the Jacobians $C_j = \frac{\partial \mathbf{c}_j}{\partial \mathbf{c}_1}$ are computed in a computerized fashion leveraging the array representation of \mathbf{c}_j . This Jacobian should be of size $K_j \times N$ and will be composed completely of monomials and scalars. To compute the elements of this it's easy to differentiate a row of $\text{arr}(\mathbf{c}_j)$ with respect to a state by removing a one from the entry corresponding to the state being differentiated, and retaining the prior power as the new multiplying coefficient. For example, $\frac{\partial}{\partial x_3}(1 \times [1, 0, 2]) = 2 \times [1, 0, 1]$, and $\frac{\partial}{\partial x_3}(1 \times [1, 1, 0]) = 0 \times [1, 1, -1]$. In this manner derivatives on the monomials are reduced to simple array operations.

A.2 Computing the nonlinear fundamental solution expansion from STTs

This is the first method for constructing $\Psi_j(t)$. Park and Scheeres (2006) introduces the state transition tensors (STTs) as a generalization of the more familiar state transition matrix. They map initial deviations of a system to deviations at some time t , and are characterized by the scalar expansions below, borrowing their notation:

$$\delta x_i(t) = \sum_{p=1}^m \frac{1}{p!} \Phi_{i,k_1\dots k_p} \delta x_{k_1}^0 \dots \delta x_{k_p}^0, \quad i = 1 : N \quad (52a)$$

$$\delta \dot{x}_i(t) = \sum_{p=1}^m \frac{1}{p!} f_{i,k_1\dots k_p}^* \delta x_{k_1}^0 \dots \delta x_{k_p}^0 \quad (52b)$$

$$\delta \dot{x}_i(t) = \sum_{p=1}^m \frac{1}{p!} \dot{\Phi}_{i,k_1\dots k_p} \delta x_{k_1}^0 \dots \delta x_{k_p}^0 \quad (52c)$$

where summation convention is used, with $k_j \in \{1, \dots, N\}$, and subscripts k_j denote the k_j^{th} component of the state vector:

$$\Phi_{i,k_1\dots k_p} = \frac{\partial^p x_i}{\partial x_{k_1}^0 \dots \partial x_{k_p}^0} \quad (53)$$

$$f_{i,k_1\dots k_p}^* = \left. \frac{\partial^p f_i}{\partial x_{k_1}^0 \dots \partial x_{k_p}^0} \right|_* \quad (54)$$

The STTs have their own unique order-dependent dynamics. Park and Scheeres (2006) presents the dynamics for the STTs again element-by-element, up to fourth order. We compute them this way, getting the necessary Jacobian terms of Eq. (54) (necessary for STT numerical integration) via automatic differentiation.

This STT representation is fundamentally redundant, e.g. the 3rd-order STT components associated with $\delta x_1^0 \delta x_2^0 \delta x_1^0$, $\delta x_1^0 \delta x_1^0 \delta x_2^0$, $\delta x_2^0 \delta x_1^0 \delta x_1^0 \dots$ are all equal yet appear individually within the STT structure. We compute a fundamental solution from its associated STT terms as below:

$$\psi_{i,\mathcal{O}(x^q)} = \frac{\mathcal{R}}{q!} \Phi_{i,k_1\dots k_q} \quad (55)$$

where $i = 1:N$ denotes the i^{th} component (row) of the fundamental solution, $\mathcal{O}(x^q)$ denotes some particular column of Ψ_j associated with the order- q STT component of interest, and \mathcal{R} is the number of repeats of the relevant STT term – for example, for $N = 2$, $j = 2$, and $\psi_{i,x_1(0)x_2(0)}$, we compute $\mathcal{R} = 2$ because we have two repeated STTs, $\Phi_{i,1,2} = \Phi_{i,2,1}$. Lastly recall that the ordering of the columns of Ψ_j corresponds to the ordering of the corresponding monomials in \mathbf{c}_j .

A.3 Computing the nonlinear fundamental solution expansions analytically

Now we discuss a second method for constructing $\Psi_j(t)$. The spacecraft relative motion problem, our example in this work, is extremely well-studied in literature due to its historical significance and enduring importance in spaceflight. For this problem there has been an enormous amount of past work to develop completely analytic nonlinear approximations of the solution of the satellite relative motion equations. These can be adopted directly to replace the numerically computed nonlinear functions used in this work. Here we provide a brief discussion about how such solutions can be derived. These “solutions” approximate, via low-order series expansion in the initial states, the

true behavior of the nonlinear differential equations of spacecraft relative motion given by Casotto (2016). Typically these are obtained by perturbation methods (Nayfeh, 2000; Hinch, 1991), most typically a straightforward expansion. Here we preview such a methodology briefly with a shortened discussion of how the second-order analytic STTs are derived (Butcher et al., 2016). In general, the state can be expanded in a perturbation series:

$$\mathbf{x}(\tau) = \mathbf{x}_1(\tau) + \varepsilon \mathbf{x}_2(\tau) + \varepsilon^2 \mathbf{x}_3(\tau) + \dots \quad (56)$$

where ε is a small parameter, $\tau = nt$, and typically the states are rendered in a non-dimensional form. The function $\mathbf{x}_1(\tau)$ is the solution to the unperturbed linearized (CW) dynamics, and the subsequent functions correct for nonlinearities in the dynamics, and possibly also target orbit eccentricity.

First, the dimensionless CW equations are solved to obtain $\mathbf{x}_1(\tau)$. Then, we perform a nonlinear expansion of the nonlinear equations of relative motion (see e.g. Schaub and Junkins (2018) for Keplerian or Casotto (2016) for non-Keplerian) up to a desired order in the states, then substitute the expansion of Eq. (56) into the dynamics. The dynamics are then parsed order-by-order starting with the unperturbed linear problem $\mathcal{O}(\varepsilon^0)$ and continuing to all other powers of ε . From the $\mathcal{O}(\varepsilon^1)$ part, the model equations are rearranged into a form below separating all perturbations up to second-order in the states (constituting F_2) from the dimensionless Clohessy-Wiltshire (CW) ODE equations:

$$\begin{pmatrix} x_2'' - 2y_2' - 3x_2 \\ y_2'' + 2x_2' \\ z_2'' + z_2 \end{pmatrix} = F_2(\tau, x_1, y_1, z_1, x_1', y_1', z_1') \quad (57)$$

The solutions to these equations can be found using symbolic software to evaluate the following inverse Laplace transform:

$$\begin{pmatrix} x_2(\tau) \\ y_2(\tau) \\ z_2(\tau) \end{pmatrix} = \mathcal{L}^{-1} \left(\begin{bmatrix} s^2 - 3 & -2s & 0 \\ 2s & s^2 & 0 \\ 0 & 0 & s^2 + 1 \end{bmatrix}^{-1} \mathcal{L}(F_2(\tau)) \right) \quad (58)$$

The inverted matrix is the transfer matrix of the CW system, and $F_2(\tau)$ is obtained by substituting the normalized CW solution into all states in the right side of Eq. (57). The velocity states $x_2'(\tau)$, $y_2'(\tau)$, $z_2'(\tau)$ are then obtained by simple symbolic differentiation of the solutions above.

For the analytic version of the third-order nonlinear solutions that we numerically computed to build Ψ_j , in local Cartesian coordinates, consult Butcher et al. (2016). Note their solution is trigonometrically sorted for compactness instead of in terms of the monomials. Butcher et al. (2017) explores a similar approach in both Cartesian coordinates and curvilinear coordinates, which offer greater accuracy because they better accommodate the natural curvature of the orbit geometry. These solutions also apply corrections for nonzero target orbit eccentricity, which improves their fidelity in a true flight setting. Note that the eccentricity-linear part of the curvilinear solutions are derived from an equation which contains an error, which is addressed in the Appendix of Willis et al. (2019a). That work also derives its own curvilinear solutions and provides a nice discussion of other analytic solutions in literature for this problem. Note that for long-duration problems, it becomes necessary to account for non-Keplerian perturbations such as J_2 (Burnett et al., 2018). The problem of deriving nonlinear analytic solutions to this problem presents a higher degree of difficulty than the Keplerian case. The appealing possibilities in use of these solutions in guidance applications (particularly if they're time-explicit), as demonstrated by this paper, might breathe new life into the academic efforts for their derivation.

A.4 Connections to Differential Algebra

It’s well-known that the STTs and Differential Algebra (DA) can be used to provide the same information about the nonlinear expansions in the vicinity of a reference. Thus DA constitutes a third method for building $\Psi_j(t)$. Indeed, we have tested this, computing Ψ_j using `Pyaudi` (Izzo et al., 2022), which offers speed advantages over the STTs. That is the first and most obvious connection of this work to DA. However in our literature review we’ve noted deeper connections of our work to the DA framework, particularly upon reading in Berz (1999) the map methods for use in particle beam physics. Here we highlight some of these connections, in anticipation of future work where our developments may be further commensurated with the formalisms of DA. For this discussion we borrow and synthesize Berz’ notation where needed. We defer the unfamiliar reader to their thorough introduction to DA so as not to risk the confusion from a compressed summary.

First we note that our K_j -dimensional space $\mathcal{C}^{(N)}$ (facilitated by the j^{th} -order nonlinear expansion) is directly related to the vector space ${}_jD_N$. In particular, let $d_k = [x_k]$ denote the higher-order differential structure induced by a j^{th} -order expansion of the k^{th} element of coordinates $\mathbf{x} \in \mathbb{R}^N$. Then the Taylor expansion T_f of some function f can be expanded into the vector space ${}_jD_N$:

$$[f] = [T_f] = \sum_{q_1 + \dots + q_N \leq j} \alpha_{q_1, \dots, q_N} \cdot d_1^{q_1} \dots d_N^{q_N} \quad (59)$$

Berz computes, via consideration of the tuples q_1, \dots, q_N , that the dimensionality of ${}_jD_N$ is as follows:

$$\dim({}_jD_N) = \binom{N+j}{N} \quad (60)$$

We observe the following similarity, where $\mathcal{C}^{(N)} \subseteq \mathbb{R}^{K_j}$, and K_j is given by our Eq. (3):

$$\dim({}_jD_N) = K_j + 1 \quad (61)$$

The difference here is that ${}_jD_N$, as a differential algebra, contains the “Real” component (the reference point about which the nonlinear expansion occurs) by definition, but $\mathcal{C}^{(N)}$ does not. In other words, elements of ${}_1D_1$ are always of the form (r_0, r_1) where r_0 is the reference point and r_1 is the (first-order) differential. The notation “Real” differentiates numbers of the form $(1, 0)$ from their “not real” differential quantity $d = (0, 1)$, which in this analogy plays a similar role to the imaginary unit i .

Other differences between $\mathcal{C}^{(N)}$ and ${}_jD_N$ are as follows. First, in our operations on $\mathcal{C}^{(N)}$ we consider real projections along the differential part of the vector space i.e. ${}_jD_N - \{0\}$, whereas the DA framework hierarchically distinguishes between the “Real” component and the differentials, and then also among different orders of the differentials themselves. Second, as an algebra, ${}_jD_N$ is always defined with dimensionality $K_j + 1$, whereas our formulation allows for the reduction of dimensionality of \mathbf{c}_j and Ψ_j by exclusion of monomials whose corresponding fundamental solutions ψ_i are zero. Indeed we can even exclude components whenever their corresponding ψ_i is negligible from the perspective of satisfying a total error requirement in the representation $\mathbf{x} = \Psi_j \mathbf{c}_j$. We don’t do that in this work, but we could, and the recent STT-based guidance work of Boone and McMahan (2023) makes such a truncation implicitly.

B Stable Linearized Guidance Solution Algorithm

Here we discuss some considerations of how the linearized rendezvous guidance briefly described in Section 4.1 can be made as numerically efficient and robust as possible. First, it is best to

contextualize the solution to the SOCP in the more familiar context of optimal control theory, which provides a more intuitive interpretation than the standard geometric descriptions of e.g. Lee and Markus (1967) and Gilbert and Harasty (1971). To do this, we highlight a connection of the SOCP to standard primer vector theory (Prussing, 2010) applied specifically to the relative motion problem. Following the notation of Casotto (2016), the general equations of relative motion in the LVLH frame are given below:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{pmatrix} = \begin{bmatrix} 0_{3 \times 3} & I_{3 \times 3} \\ H & K \end{bmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{v} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ a\hat{\mathbf{u}} \end{pmatrix} \quad (62)$$

for thrust direction $\hat{\mathbf{u}}$, hence $a(t) \geq 0$ always. We seek to minimize the total delta-V, which is the integral of the control acceleration:

$$J = \int_{t_0}^{t_f} a(t) dt \quad (63)$$

We construct the Hamiltonian with the position and velocity co-states written separately:

$$\mathcal{H} = a + \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top (H\mathbf{r} + K\mathbf{v} + a\hat{\mathbf{u}}) \quad (64)$$

The Hamiltonian is instantaneously minimized by the choice of control direction $\hat{\mathbf{u}} = \mathbf{p}(t)/p(t)$ where $\mathbf{p}(t) = -\boldsymbol{\lambda}_v$. This analysis holds for continuous control and also applies to the limit case of impulsive control e.g. as $a(t_{b_i}) \rightarrow \infty$ at some burn time(s) t_{b_i} with $a(t) = 0$ otherwise. Applying the first-order conditions of optimality,

$$\dot{\boldsymbol{\lambda}}_r = -H^\top \boldsymbol{\lambda}_v \quad (65a)$$

$$\dot{\boldsymbol{\lambda}}_v = -\boldsymbol{\lambda}_r - K^\top \boldsymbol{\lambda}_v \quad (65b)$$

Making use of Eq. (65) and the identity $\dot{\mathbf{p}} = \boldsymbol{\lambda}_r + K^\top \boldsymbol{\lambda}_v$, the following independent differential equation is obtained for the primer vector:

$$\ddot{\mathbf{p}} = \left(H^\top - \dot{K}^\top \right) \mathbf{p} - K^\top \dot{\mathbf{p}} \quad (66)$$

This holds regardless of the form of the linearized dynamics, e.g. regardless of the form of the H and K matrices in Casotto's formulation. In the classical continuous thrust formulation, the norm of the primer vector $p(t)$ is used as a switching function, with optimal thrusting times only occurring when $p(t) > 1$. In the impulsive thrust case, which is a limit case of the continuous thrust problem, the optimal firing times happen only when $p(t) = 1$.

For the impulsive problem, the conditions of an optimal firing sequence are as follows. First, we require the existence of a primer vector $\mathbf{p}(t)$ satisfying Eq. (66). Furthermore the magnitude $p(t) \leq 1$ at all times, with optimal maneuver times located at times with $p(t) = 1$, and $\dot{\mathbf{p}} \cdot \mathbf{p} = 0$ at interior impulses (e.g. not first and last), with the thrust always directed along $\hat{\mathbf{u}}(t) = \mathbf{p}(t)/p(t)$ and never anti-parallel. It can be shown that the solution $\bar{\mathbf{p}}(t_i) = B_c(t_i)^\top \boldsymbol{\eta}$ from the Stage 1 SOCP satisfies Eq. (66), even though the optimal control problem is defined in continuous time but the SOCP is set up in discretized time. The optimal discretized firing times predicted by the norm of $\bar{\mathbf{p}}(t_i) = B_c(t_i)^\top \boldsymbol{\eta}$ equaling unity will only approximate the truly optimal firing times (which in general occur between sampled discretized times). This will manifest in general as small overshoots $p(t) > 1$ of the $\mathbf{p}(t)$ numerically propagated from $\bar{\mathbf{p}}(t_0) = B_c(t_0)^\top \boldsymbol{\eta}$, at local maxima of $p(t)$ which will appear in the vicinity of the SOCP-predicted firing times where $\|\bar{\mathbf{p}}(t_i)\| = 1$. See Roscoe et al. (2014) and references therein for a practical discussion of this overshoot phenomenon and

its mitigation (though their work does not make use of the SOCP formulation). It is possible to leverage the overshoots (which are explicitly violations of an equality constraint of optimality) to differentially correct (e.g. in a simple Newton scheme) the $\boldsymbol{\eta}$ from the SOCP to obtain the true optimal $\boldsymbol{\eta}^*$. In this manner the size of the SOCP specified by Stage 1 can be made fairly small (with a coarse discretization) and subsequently differentially corrected to rapidly obtain the true linearly optimal solution. In this work, because we are interested in feasible (and possibly optimal) solutions for the *nonlinear* dynamics, given the stability of our methods to a poor initial guess (e.g. a poor solution to Stage 1 linearized guidance), this correction scheme was not retained.

There is however another numerical detail of Stage 1 guidance that absolutely requires consideration to generate reasonable solutions: how to determine unambiguously the firing times predicted by the SOCP? From the discretized solution to the SOCP, it is necessary to identify possible points where $p(t)$ is numerically sufficiently close to 1. In general, we require a k -burn candidate solution with $k \leq 6$, but based on a given tolerance ϵ for $|p(t) - 1| < \epsilon$, there will often be more than 6 candidate points distributed among k intervals. To demonstrate this we revisit now the linear solution to the final example in Section 4.2. The $p(t)$ generated by the SOCP solution is plotted in Figure 14 with the proper burn times as red dots and all other possible burn points marked with a black \times . In this case $p(t) \approx 1$ at four distinct intervals in time, so $k = 4$. A candidate

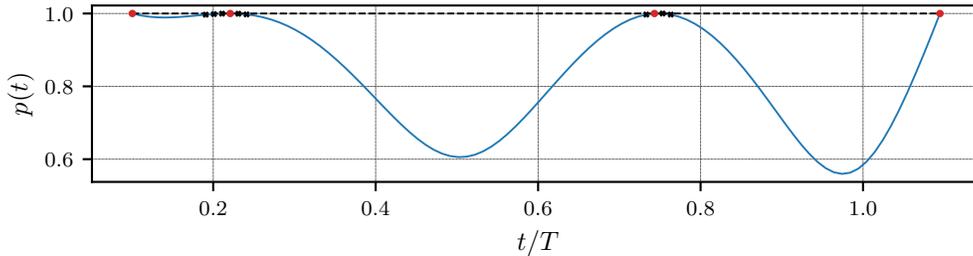


Figure 14: Linearized Guidance primer vector norm and burn times (Example 3)

solution (a choice of up to $k \leq 6$ points from the k intervals where $p(t)$ gets sufficiently close to 1) is not guaranteed to satisfy Eq. (43) because $\Delta\mathbf{c}^*$ has 6 quantities and we fire k times, so the linear problem is therefore underdetermined when $k < 6$. Thus, the resultant linearized guidance solution resulting from a general candidate solution is not guaranteed to achieve its goal. We require a scheme that identifies all possible burn times from the SOCP solution, creates reasonable candidate solutions, and tests them based on conditions of optimality and how closely they achieve $\Delta\mathbf{c}^*$, keeping only the best (or least bad) guidance solution. We summarize such a scheme with the following algorithm for a set of possible burn time indices \mathcal{I} , composed of indices i corresponding to all t_i for which $|p(t_i) - 1| < \epsilon$:

-
- 1: **procedure** SOLUTION_SORT($\mathcal{I}, \boldsymbol{\eta}, p(t_i); B_c(t_i) \forall t_i \in \mathcal{I}, \Delta\mathbf{c}^*$) \triangleright Get best solution $\mathcal{I}^* \subseteq \mathcal{I}$
 - 2: $\mathcal{G} \leftarrow \text{sort_groups}(\mathcal{I}, P)$ \triangleright Sort neighboring indices in \mathcal{I} into k groups of $\leq P$ points each
 - 3: $\Omega \leftarrow \mathcal{G}_1 \times \dots \times \mathcal{G}_k$ \triangleright Construct the Cartesian product Ω of all points in the k groups
 - 4: $q=1$
 - 5: **while** $q \leq$ (number of combos in Ω) **do** \triangleright Check all combinations in Ω
 - 6: $\mathcal{S}_q \leftarrow \text{make_candidate}(\Omega[q], B_c(t_i) \forall t_i \in \Omega[q])$ \triangleright Create candidate solution \mathcal{S}_q
 - 7: $e_q \leftarrow \text{get_error}(\mathcal{S}_q, \Delta\mathbf{c}^*)$ \triangleright Compute (linearized) error metric in achieving $\Delta\mathbf{c}^*$ (float)
 - 8: $a_q \leftarrow \text{assess_accuracy}(\mathcal{S}_q, \text{tol})$ \triangleright Check if error $e_q < \text{tol}$ (Boolean)
 - 9: $O_q \leftarrow \text{check_optimality}(\mathcal{S}_q)$ \triangleright Check optimality conditions, e.g. all $\Delta v_i > 0$ (Boolean)

```

10:      $\Delta v_q \leftarrow \text{compute\_delta\_v}(\mathcal{S}_q)$  ▷ Total delta-V for this candidate
11:     q++
12: end while
13: if  $\exists q \mid (a_q = \text{True}) \ \& \ (O_q = \text{True})$  then ▷ We have acceptable candidates
14:      $q_{\text{best}} = q \mid (a_q = \text{True}) \ \& \ (O_q = \text{True}) \ \& \ \Delta v_q$  is min of all acceptable  $q$ 
15: else ▷ We don't have acceptable candidates
16:     if  $\nexists q \mid (a_q = \text{True})$  then ▷ None of the candidates meet error tol
17:          $q_{\text{best}} = q \mid e_q$  is min of all ▷ Pick least erroneous
18:     else ▷ Some solutions meet error tol but none meet optimality conds.
19:          $q_{\text{best}} = q \mid \Delta v_q$  is min of all ▷ Pick least expensive
20:     end if
21: end if
22: return  $\mathcal{S}_{q_{\text{best}}}, \Omega[q_{\text{best}}]$  ▷ Return the details of best solution
23: end procedure

```

We find that this routine is extremely stable, with negligible runtime for reasonable choices of primer vector tolerance ϵ and max points per group P . For clarity we now apply it with all possible burn indices in the linearized guidance solution from Example 3. For this example, $\epsilon = 5 \times 10^{-3}$, $P = 2$, and $\text{tol} = 10^{-3}$. For this problem the control interval is discretized into 100 times, thus $i \in [0, 99]$. The possible burn time indices \mathcal{I} , down-selected groups \mathcal{G} , and combinations Ω are given below:

$$\mathcal{I} = [0, 9, 10, 11, 12, 13, 14, 63, 64, 65, 66, 99] \tag{67a}$$

$$\mathcal{G} = \{\{0\}, \{11, 12\}, \{64, 65\}, \{99\}\} \tag{67b}$$

$$\Omega = \{\{0, 11, 64, 99\}, \{0, 11, 65, 99\}, \{0, 12, 64, 99\}, \{0, 12, 65, 99\}\} \tag{67c}$$

From this set of points, SOLUTION_SORT keeps the candidate solution parameterized by $\{0, 12, 64, 99\}$.