AAS 25-218

A MONOMIAL METHOD FOR NONLINEAR OPTIMIZATION IN ASTRODYNAMICS

Ethan R. Burnett* **and Francesco Topputo†**

This paper introduces a monomial method for rapidly solving a variety of nonlinear optimization problems astrodynamics. Using differential algebra to compute Taylor expansions of given nonlinear functions, the result is a function of multivariate monomials up to the order of truncation. These monomials are interrelated. We use this fact to approximate nonlinear functions in Euclidean space by linear functions on a Riemannian manifold, wherein the problem can be solved with speed and stability, leveraging numerical schemes for optimization on manifolds. The method could prove advantageous in contexts requiring the repeated solution of similar optimization problems, for which the expansion can be reused, or for particularly tricky optimization problems where a combination of rapid sampling and gradient-based methods need to be used. In this introductory work we lay the foundations and explore two simple example applications: 1) computing periodic orbits, and 2) computing/continuing optimal impulsive transfers.

INTRODUCTION

Many of the tools applied in onboard architectures in spaceflight guidance and control rely inherently on a local, linearized methodology, whereby information from an earlier computed reference trajectory or prior iterate is used to compute new information to regulate towards (in the case of control) or otherwise update (in the case of optimization and filtering) the reference. This paradigm employs well-established, stable, and easily profiled computational tools for solving even fairly large linearized problems. However, recent trends in engineering anticipate the adoption of more advanced approaches incorporating nonlinear information much more broadly for onboard tasks. The use of higher-order tensor expansions has gained popularity, in the form of state transition tensors (STTs). These have recently been proposed for spaceflight guidance, see e.g. Reference 1. In that work, they are computed numerically by integrating the STT dynamics in parallel with the reference, a methodology proposed by Reference 2. In some applications, such expansions can also be computed analytically, such as in spacecraft formation flying and relative motion analysis.[3,4] Their computation is also achievable via automatic differentiation methods, namely differential algebra (DA).[5] Often this information has been used to study the evolution of uncertainty.

With nonlinear expansions being available, local optimization problems can also be solved, with the most noteworthy examples being the use of DA to expand optimal solutions computed via indirect methods, allowing for dispersed problem conditions to be accommodated with minimal additional computational load.[6] Similar application with highly reliable direct methods would also be

*Marie Skłodowska-Curie Postdoctoral Fellow, Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa 34, 20156 Milano IT.
†Professor, Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa 34, 20156 Milano IT.

very attractive, especially given the recent popularization of convex optimization. Modern interior-point methods are so stable and computationally efficient that in practice, convex optimization problems (especially second-order cone programs) are often not much more difficult to solve than linear problems.[7] General nonlinear optimization problems can be solved via a sequential convex optimization approach, extending the use of such methods beyond the standard convex problem forms.[8]

Nonlinear expansions which are pre-computed in advance of their need and rendered in a convex optimization-usable format can be expected to render a solution much faster than a more typical shooting- or collocation-based approach working directly with nonlinear differential equations. In this work we propose such an approach, powered by DA. We provide a general "monomial method" for localized nonlinear optimization, in which nonlinear functions in Euclidean space are rendered linear on a Riemannian manifold of analytic structure. This is done by using the inherent relationship between monomials in a multivariate Taylor expansion. Optimization problems can be solved with speed and stability in such a space, especially by leveraging techniques to limit the dimensionality of the problem. While as general as classical Newton's method or gradient descent, the approach is imagined specifically for applications where optimization problems must be solved repeatedly with slightly different inputs, reusing a given pre-computed expansion. We've developed this out of interest in its computational application to fast onboard spacecraft GNC tasks (including closed-loop guidance), but the methodology and underlying concepts may be of interest for other problems. This paper is intended as a low-level introduction to its use in astrodynamics. The approach is rather non-standard, and the interested reader would benefit from review of a few key topics, particularly some basics of the mathematics of smooth manifolds (e.g. Ref 9), and direct optimization (our choice is convex optimization, see Refs. 7, 8). We use DA as a tool to compute Taylor expansions but it need not be deeply understood for this work (see e.g. Refs. 5, 6 respectively for the basics and for an application in astrodynamics).

THEORY

Nonlinear Problems in Astrodynamics

Many tools and techniques in astrodynamics are rooted in the generic process of the minimization of some cost function subject to general constraints,

$$\begin{aligned} &\min_{\boldsymbol{V}} \ J(\boldsymbol{V}) \\ &\text{subject to } \begin{matrix} \boldsymbol{g}(\boldsymbol{V}) = \boldsymbol{0} \\ \boldsymbol{h}(\boldsymbol{V}) \leq \boldsymbol{0} \end{matrix} \end{aligned} \tag{1}$$

for free variables $\boldsymbol{V} \in \mathbb{R}^N$ and equality and inequality constraints $\boldsymbol{g}(\boldsymbol{V}) \in \mathbb{R}^{M_e}$ and $\boldsymbol{h}(\boldsymbol{V}) \in \mathbb{R}^{M_i}$. The free variables constituting $\boldsymbol{V}$ will often consist of state variables and time. For such nonlinear problems, identifying a solution typically takes the form of correcting an initial guess $\boldsymbol{V}_0$ in an iterative march towards a local minimizer. Via the assignment $J = 0$, $\boldsymbol{h} = \boldsymbol{0}$, and $\boldsymbol{g} = \boldsymbol{F}$, the process of solving a system of nonlinear equations is posed as a degenerate form of Eq. (1):

$$\boldsymbol{F}(\boldsymbol{V}) = \boldsymbol{0} \tag{2}$$

Namely, this is a *feasibility problem*. Both Eqs. (1) and (2) may be solved via a sequence of linear steps $\delta\boldsymbol{V}_i$ which move the free variables $\boldsymbol{V}$ towards the optimizer/solution.

Assuming that J, $\boldsymbol{g}$, and $\boldsymbol{h}$ are sufficiently well-approximated by some j^{th}-order Taylor expansion with $j > 1$, it is possible to directly set up and solve a sub-problem for the step suitable for that case,

regardless of order j. In this work we explore a method to solve localized nonlinear optimization problems via the construction and solving of sub-problems of the following affine form:

$$\begin{aligned} \min_{\boldsymbol{c}_j} \quad & J(\boldsymbol{V}_{i-1}) + \Gamma_J \boldsymbol{c}_j \\ \text{subject to} \quad & \begin{aligned} \boldsymbol{c}_j &\in \mathcal{C}^{(N,j)} \\ \Gamma_g \boldsymbol{c}_j &= -\boldsymbol{g}(\boldsymbol{V}_{i-1}) \\ \Gamma_h \boldsymbol{c}_j &\leq -\boldsymbol{h}(\boldsymbol{V}_{i-1}) \\ \|\boldsymbol{c}_j\| &\leq d_i \end{aligned} \end{aligned} \tag{3}$$

This is a preview. All notation here will be explained more thoroughly in the sections that follow. The vector $\boldsymbol{c}_j$ is a j^{th}-order analog to the linear step in traditional gradient descent/Newton's method, and is solved for directly. It maps to/from $\delta\boldsymbol{V}_i$ via a straightforward mapping, and is constrained to lie on a smooth N-dimensional manifold $\mathcal{C}^{(N,j)}$. The row vector Γ_J, computed efficiently leveraging differential algebra, contains linear and higher-order sensitivities of $J(\boldsymbol{V})$, such that $J\left(\boldsymbol{V}_{i-1}\right) + \Gamma_J \boldsymbol{c}_j$ equals a j^{th}-order Taylor expansion of $J\left(\boldsymbol{V}_{i-1} + \delta\boldsymbol{V}_i\right)$. Similarly, the matrices Γ_g and Γ_h capture the j^{th}-order Taylor expansions of $\boldsymbol{g}(\boldsymbol{V})$ and $\boldsymbol{h}(\boldsymbol{V})$ about the reference or prior iterate. A practical trust region constraint may be placed on $\|\boldsymbol{c}_j\|$ (with the choice of norm to be discussed) to limit the solution search to within the expected domain of validity of the Taylor expansion. Observe that the originally nonlinear cost and constraint functions are rendered *affine* in $\boldsymbol{c}_j$, and the only remaining nonlinearity is a manifold constraint $\boldsymbol{c}_j \in \mathcal{C}^{(N,j)}$. This affine form should remind the reader of a linearized form of Eq. (1), and indeed we can view standard linearization as the trivial order $j = 1$ case of this more general form.

For such a formulation we require only that $J(\boldsymbol{V})$ and any constraint functions $\boldsymbol{g}(\boldsymbol{V})$, $\boldsymbol{h}(\boldsymbol{V})$ are smooth and smoothly differentiable. The high-level logic of this approach is similar to the gradient descent and Newton schemes, consisting of a sequence of three steps: [1] nonlinear update based on a prior iterate $\boldsymbol{V}_{i-1}$, [2] sub-problem setup and solution for the i^{th} $\boldsymbol{c}_j$ (and hence $\delta\boldsymbol{V}_i$), then [3] iterate step $\boldsymbol{V}_i = \boldsymbol{V}_{i-1} + \delta\boldsymbol{V}_i$. The higher-order implementation reduces greatly or even eliminates the need for repeats of this sequence to obtain a solution, but with added computational expense of computing the Taylor expansions. This produces a need for efficient and robust schemes for computing the relevant DA expansions and also for solving for the optimal $\boldsymbol{c}_j$. For this early work, the code is prototyped in Python, making use of daceypy (a Python front-end for the Differential Algebra Computational Engine DACE[10]). We introduce the methodology in a general sense such that it can be applied analogously to linear methods, alternating between updates of the expansion terms Γ_J, Γ_g, Γ_h and solutions for the new nonlinear step $\boldsymbol{c}_j$. However, the required frequency of updates is greatly reduced or eliminated in comparison to a more typical linear step-based approach.

Nonlinear Expansions and Monomial Parameterization

Consider a finite higher-order Taylor expansion of a scalar nonlinear function $g(\boldsymbol{x})$ of free variables $\boldsymbol{x}$ around some reference point $\boldsymbol{x}_r$:

$$g(\boldsymbol{x}) \approx g(\boldsymbol{x}_r) + \sum_{i=1}^{N} \left.\frac{\partial g}{\partial x_i}\right|_{\boldsymbol{x}_r} \delta x_i + \frac{1}{2!} \sum_{i_1 \leq N;\ i_2 \leq N} \left.\frac{\partial^2 g}{\partial x_{i_1} \partial x_{i_2}}\right|_{\boldsymbol{x}_r} \delta x_{i_1} \delta x_{i_2} + \ldots \tag{4}$$

This can equivalently be written in the following form:

$$g(\boldsymbol{x}) \approx g(\boldsymbol{x}_r) + \Gamma_g \boldsymbol{c}_j \tag{5}$$

where $\boldsymbol{c}_j$ simply orders the unique monomial combinations of the components of $\delta\boldsymbol{x}$ up to order j. This is illustrated by the below example for $N = 2$, $j = 3$:

$$\boldsymbol{c}_3 = \left(\delta x_1, \delta x_2, \delta x_1^2, \delta x_1\delta x_2, \delta x_2^2, \delta x_1^3, \delta x_1^2\delta x_2, \delta x_1\delta x_2^2, \delta x_2^3\right)^\top \tag{6}$$

We trade traditional multi-index notation for compact and affine expressions, relying on a logically consistent and well-defined monomial ordering scheme that works for any (N, j). Also, for $j = 1$, $\boldsymbol{c}_j = \boldsymbol{c}_1 = \delta\boldsymbol{x}$, and these arguments reduce to familiar linear algebra. The row vector Γ_g orders, in the same sequence, the associated derivatives of g (and their associated multipliers from the Taylor series):

$$\Gamma_g = \left[\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \frac{1}{2}\frac{\partial^2 g}{\partial x_1^2}, \frac{\partial^2 g}{\partial x_1\partial x_2}, \frac{1}{2}\frac{\partial^2 g}{\partial x_2^2}, \frac{1}{6}\frac{\partial^3 g}{\partial x_1^3}, \frac{1}{2}\frac{\partial^3 g}{\partial x_1^2\partial x_2}, \frac{1}{2}\frac{\partial^3 g}{\partial x_1\partial x_2^2}, \frac{1}{6}\frac{\partial^3 g}{\partial x_2^3}\right] \tag{7}$$

This idea extends naturally to vector functions $\boldsymbol{g}$, for which Γ_g becomes a matrix.

In general, $\boldsymbol{c}_j \subseteq \mathbb{R}^{K_j}$, where K_j is given by the number of unique combinations of differentials up to and including order j:

$$K_j = \binom{N+j}{N} - 1 \tag{8}$$

There will always be $K_j - N$ constraints among the elements of $\boldsymbol{c}_j$. For example, modifying the value of the first element of $\boldsymbol{c}_3$ in Eq. (6) will also necessarily affect elements 3, 4, 6, 7, 8. In this manner, $\boldsymbol{c}_j$ is constrained to an N-dimensional embedded sub-manifold of $\mathbb{R}^{K_j}$, denoted $\mathcal{C}^{(N,j)}$. This will be formalized shortly. The mapping from some $\boldsymbol{c}_1 = \delta\boldsymbol{x} \in \mathbb{R}^N$ to its monomial expansion $\boldsymbol{c}_j = \boldsymbol{E}_j(\boldsymbol{c}_1)$ is an analytic nonlinear mapping, and the inverse is a linear map:

$$\boldsymbol{c}_1 = \begin{bmatrix} I_{N\times N} & 0_{N\times K_j - N}\end{bmatrix}\boldsymbol{c}_j \tag{9}$$

We can frame these arguments in the context of differential algebra, used extensively to compute the expansions in this work. Let $d_k = [x_k]$ denote the higher-order differential structure induced by a j^{th}-order expansion of the k^{th} element of some general coordinates $\boldsymbol{x} \in \mathbb{R}^N$. As discussed by Berz,[5] the Taylor expansion T_f of some function f can be expanded into a vector space:

$$[f] = [T_f] \;=\; \sum_{q_1+\ldots+q_N\leq j} \alpha_{q_1,\ldots,q_N} \cdot d_1^{q_1}\ldots d_N^{q_N} \tag{10}$$

where the d_i are differentials of the i^{th} coordinate. These higher-order differentials exist in a vector space of dimension K_j. The construction of a matrix Γ_g involves computing $\boldsymbol{g}$ as a function of the DA variables $\boldsymbol{V}$, then from the resulting Taylor map sorting the coefficients from the expansion T_g into their proper place. The column-wise ordering of the coefficients in Γ_g must correspond to the ordering of the corresponding monomials in $\boldsymbol{c}_j$. To validate this construction, evaluation of the deviation for for some differential $\delta\boldsymbol{V}$ via the Taylor map, $[T_g](\delta\boldsymbol{V}) - g(\boldsymbol{V})$, should yield numerically the same result as $\Gamma_g\boldsymbol{c}_j^{(\delta\boldsymbol{V})}$, where the superscript on $\boldsymbol{c}_j$ emphasizes that the monomial sequence is constructed from $\delta\boldsymbol{V}$.

Via the monomial formulation, suitable nonlinear functions $\boldsymbol{g}(\boldsymbol{x})$ are rendered affine,

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{x}_r + \delta\boldsymbol{x}) &\approx \boldsymbol{g}(\boldsymbol{x}_r) + \Gamma_g\boldsymbol{c}_j \\ \boldsymbol{c}_j &\in \mathcal{C}^{(N,j)} \\ \|\boldsymbol{c}_j\| &< d \end{aligned} \tag{11}$$

where the final constraint limits $\boldsymbol{c}_j$ (or equivalently $\delta\boldsymbol{x}$) to an estimated domain of validity of the expansion. Thus the problem of solving for some $\delta\boldsymbol{x}$ which satisfies a nonlinear Taylor expansion of $\boldsymbol{g}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{0}$ is transformed to solving for the satisfactory linearly multiplying $\boldsymbol{c}_j$, constrained to a non-Euclidean space of known structure. The flattened formulation Γ_g also allows us to extend some familiar concepts from linear algebra (associated with linear expansion) to tensors (associated with nonlinear expansion). For example, $\boldsymbol{c}_j \neq \boldsymbol{0}$ lies in $\ker(\Gamma_g)$ if the following is satisfied:

$$\begin{aligned} &\Gamma_g \boldsymbol{c}_j = \boldsymbol{0} \\ &\boldsymbol{c}_j \in \mathcal{C}^{(N,j)} \end{aligned} \tag{12}$$

We observe that operations on $\mathcal{C}^{(N,j)}$ tend to be quite computationally efficient and stable. The equation for $\boldsymbol{c}_j$ is an analytic function of the original coordinates $\delta\boldsymbol{x}$ of low complexity. "Moving" on the manifold is done via the tangent spaces and retraction operations. Solving for the optimal $\boldsymbol{c}_j^*$ consists of solving a sequence of simple problems in the tangent bundle of the manifold, $T_{\boldsymbol{c}_j}\mathcal{C}^{(N,j)}$.

This construction introduces an interesting competition between (1) computational schemes leveraging the traditional sequence of "propagate, linearize, repeat", and (2) schemes leveraging higher-order expansions that are more costly to generate but far superior to a linearization. Its most obvious use is for situations where a similar nonlinear optimization problem must be solved repeatedly (i.e. the expansions need to be computed only once, and are then re-used many times). The benefits become clear in situations when the original reference was computationally expensive to generate, and thus approach (1) becomes undesirable.

Monomials on Embedded Sub-manifolds

If we can show that $\mathcal{C}^{(N,j)}$ is a smooth sub-manifold of Euclidean space, then we can borrow rigorous techniques for optimization on such manifolds to solve for the $\boldsymbol{c}_j^* \in \mathcal{C}^{(N,j)}$ solving (or optimizing) our problem of interest. Here we borrow terminology from Reference 9. Let $\mathcal{E}$ be a Euclidean space of dimension d. A non-empty subset $\mathcal{M}$ of $\mathcal{E}$ is a smooth embedded submanifold of $\mathcal{E}$ of dimension n if $n = d - \kappa$ for some $\kappa \geq 1$ and, for each $\boldsymbol{x} \in \mathcal{M}$, there exists a neighborhood U of $\boldsymbol{x}$ in $\mathcal{E}$ and a smooth *defining* function $h : U \to \mathbb{R}^\kappa$ such that (a) if $\boldsymbol{y}$ is in U, then $h(\boldsymbol{y}) = \boldsymbol{0}$ iff $\boldsymbol{y} \in \mathcal{M}$ and (b) $\text{rank}\,(\mathrm{D}h(\boldsymbol{x})) = \kappa$.

In our case, $\mathcal{E}$ is $\mathbb{R}^{K_j}$, thus $d = K_j$, and $\kappa = K_j - N$ is the dimension of the defining function $U(N,j)$ constraining the elements of $\boldsymbol{c}_j$. It is easy to show for any N, j, there exists a suitable defining function that is only zero for all $\boldsymbol{y} \in \mathcal{C}^{(N,j)}$, and that the Jacobian of this function is always rank κ as required. We demonstrate this explicitly for the simple case $N = 2$, $j = 2$:

$$\boldsymbol{c}_2 = \left(x_1, x_2, x_1^2, x_1x_2, x_2^2\right)^\top \tag{13}$$

The smooth defining function is clear, absorbing the definition of all nonlinear monomials from $\boldsymbol{c}_j$:

$$h(\boldsymbol{y}) : U \to \mathbb{R}^{K_j - N} = \begin{pmatrix} y_3 - y_1^2 \\ y_4 - y_1y_2 \\ y_5 - y_2^2 \end{pmatrix}, \text{ thus } h(\boldsymbol{c}_2) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ and } h(\boldsymbol{y}) \neq \boldsymbol{0} \text{ for } \boldsymbol{y} \notin \mathcal{C}^{(N,j)}. \tag{14}$$

$$\mathrm{D}h(\boldsymbol{x}) = \left.\frac{\partial h}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{c}_2} = \begin{bmatrix} -2c_2[1] & 0 & 1 & 0 & 0 \\ -c_2[2] & -c_2[1] & 0 & 1 & 0 \\ 0 & -2c_2[2] & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Note from the above that rank $(\mathrm{D}h(\boldsymbol{x})) = 3 = \kappa$ for all $\boldsymbol{x} = \boldsymbol{c}_2 \in \mathcal{C}^{(2,2)}$. We have thus established that $C^{(N,j)}$ for $N = 2$, $j = 2$ is a smooth two-dimensional sub-manifold of $\mathbb{R}^5$. In general $C^{(N,j)}$ will always be of dimension N.

A straightforward modification of this example can be used to show that if a monomial term does not occur in any Taylor expansion $\Gamma\boldsymbol{c}_j$ for a given problem, then it can be removed from the formulation without compromising the manifold nature of $C^{(N,j)}$. Consider for example the implications that x_1x_2 is removed from $\boldsymbol{c}_2$, so $\kappa = 2$, $d = 4$, and re-compute the new $h(\boldsymbol{y})$ and $Dh(\boldsymbol{x})$. Removing this given monomial reduces κ by one, but reduces also the dimension of $\mathcal{E}$ by one, such that the modified $Dh(\boldsymbol{x})$ is still full-rank. This allows us to exploit any sparseness of the Taylor expansions for a particular coordinate representation. Removing one more coordinate, x_2^2, from $\boldsymbol{c}_2$ in Eq. (13) allows us to physically depict the resulting reduced $\mathcal{C}^{(2,2)}$, which is shown as a parabolic sheet in $\mathbb{R}^3$ in Figure 1. Note that $\boldsymbol{C}^{(N,j)}$ is homeomorphic to $\mathbb{R}^N$ by definition of $\boldsymbol{E}_j$ and its inverse, Eq. (9). In this simple 2D case, that insight can be visualized geometrically – $\mathbb{R}^2$ can be "curled" upwards to produce $\mathcal{C}^{(2,2)}$.

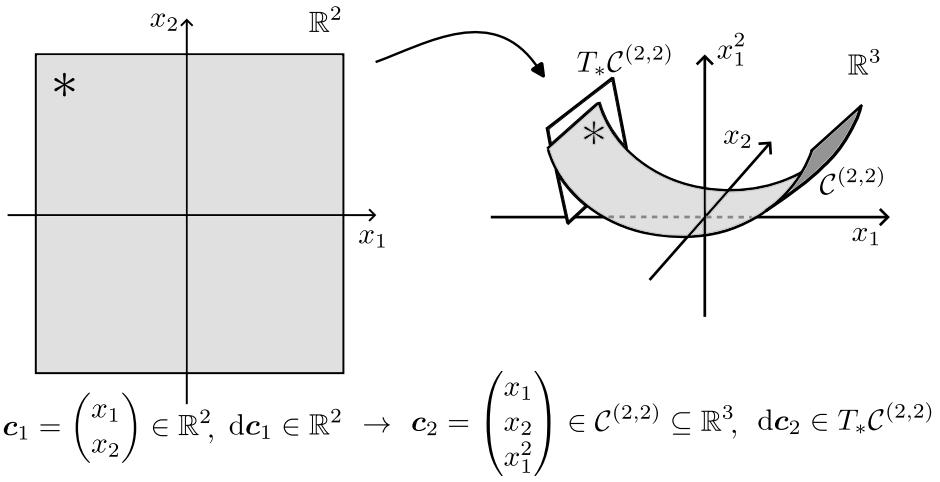


Figure 1. Monomials on a manifold

Figure 1 also shows a plane tangent to $\mathcal{C}^{(2,2)}$ at the operating point "$*$". For higher-dimensional manifolds this generalizes to the notion of a tangent space $T_{\boldsymbol{c}_j}\mathcal{C}^{(N,j)}$ in the vicinity of a point $\boldsymbol{c}_j \in \mathcal{C}^{(N,j)}$. The collection of all tangent spaces at all points is called the tangent bundle $T\mathcal{C}^{(N,j)}$. Consider an infinitesimal variation $\mathrm{d}\boldsymbol{c}_j$ lying in the tangent space at "$*$". This variation is simply,

$$\mathrm{d}\boldsymbol{c}_j = \left.\frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\right|_* \mathrm{d}\boldsymbol{c}_1 \tag{16}$$

The Jacobian of the monomials, e.g. $\partial\boldsymbol{c}_j/\partial\boldsymbol{c}_1$, thus necessary for computations on the tangent spaces of $\mathcal{C}^{(N,j)}$, is simply a linear function of the monomials up to order $j-1$, and hence is itself a linear function of $\boldsymbol{c}_j$. The Jacobian of Eq. (13), for which $j = 2$, is a low-order demonstration:

$$\frac{\partial \boldsymbol{c}_2}{\partial \boldsymbol{c}_1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2x_1 & 0 \\ x_2 & x_1 \\ 0 & 2x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2c_2[1] & 0 \\ c_2[2] & c_2[1] \\ 0 & 2c_2[2] \end{bmatrix} \tag{17}$$

Lastly, because the free variables live on a smooth sub-manifold of a higher-dimensional Euclidean space in which the equality and inequality constraints are affine, there is a clear geometric interpretation of the optimization problem given by Eq. (3). At risk of overusing 3D drawings to visualize high-dimensional spaces, this interpretation is depicted in Figure 2, for which the monomial manifold is represented as a two-dimensional submanifold. For a given pre-computed expansion, the solution is obtained by finding the optimal point from a feasible subset defined by hyperplanes, which themselves are a manifestation of the expansions of $g(\boldsymbol{V}) = 0$ and $h(\boldsymbol{V}) \leq 0$. Adopting the induced metric from $\mathbb{R}^{K_j}$, $\mathcal{C}^{(N,j)}$ becomes a Riemannian submanifold (see e.g. Reference 9). There has been recent progress in practically solving optimization formulations on such spaces.[11] For now, we employ our own on-manifold sequential convex programming (SCP) approach.

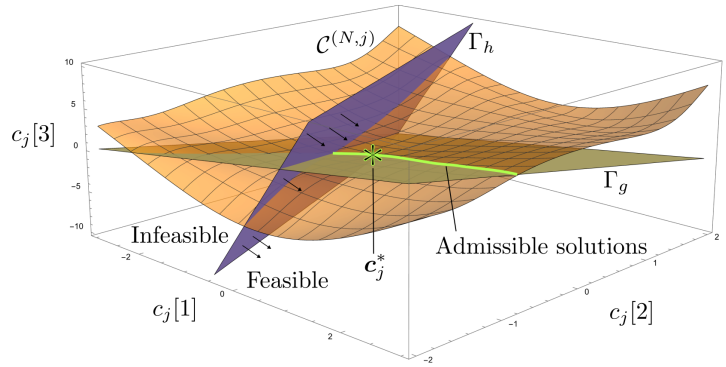


Figure 2. Optimization on the Monomial Manifold

METHODOLOGY

Problem-solving on the Manifold

We approximately solve Eq. (1) under the assumptions that all constraint functions are smooth and differentiable and admit locally convergent Taylor series, and that $\boldsymbol{J}(\boldsymbol{V})$ is convex, for which we seek local minimizers. Eq. (1) is approximated to order j by Eq. (3), linear in $\boldsymbol{c}_j$, for which we seek an optimizer $\boldsymbol{c}_j^*$. Due to the non-Euclidean nature of the space $\mathcal{C}^{(N,j)}$, in lieu of a direct solution, we seek a speedy and reliable sequence of steps $\boldsymbol{c}_j[q] = \sum_{k=1}^{q} \delta\boldsymbol{c}_j[k]$ yielding the optimal $\boldsymbol{c}_j^*$. For a small step, the dimensionality of problem free variables is reduced from K_j back to N via the application of Eq. (16):

$$\delta\boldsymbol{c}_j[k] \approx \left.\frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\right|_{\boldsymbol{c}_j[k-1]} \delta\boldsymbol{c}_1 \tag{18}$$

where $\delta\boldsymbol{c}_1 = \delta\boldsymbol{V}$. We define the norm $\|\boldsymbol{c}_j\| = \sqrt{\boldsymbol{c}_1^\top \boldsymbol{c}_1}$, recalling that $\boldsymbol{c}_j$ is homeomorphic to $\boldsymbol{c}_1$.

Sequential Convex Programming To start, Eq. (3) is rewritten in the following iterable and convex form, with a quadratic truncation of the cost expansion and a linearization of the constraints:

$$\begin{aligned}
\min_{\delta\boldsymbol{c}_1} \quad & J_{[k-1]} + DJ_{[k-1]}\delta\boldsymbol{c}_1 \\
& + \frac{1}{2}\langle \mathrm{D}^2 J_{[k-1]}\left[\delta\boldsymbol{c}_1\right], \delta\boldsymbol{c}_1\rangle + \frac{1}{2} w_e \boldsymbol{s}_e^\top[k]\boldsymbol{s}_e[k] + \frac{1}{2} w_i \boldsymbol{s}_i^\top[k]\boldsymbol{s}_i[k] \\
\text{subject to} \quad & \Gamma_g \tfrac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\delta\boldsymbol{c}_1 + \boldsymbol{s}_e[k] = -\boldsymbol{g}_{[k-1]} \\
& \Gamma_h \tfrac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\delta\boldsymbol{c}_1 + \boldsymbol{s}_i[k] \leq -\boldsymbol{h}_{[k-1]} \\
& \|\delta\boldsymbol{c}_1\| \leq \min\left(d_i - \|\boldsymbol{c}_1[k-1]\|,\ d_{\text{step}}\right)
\end{aligned} \tag{19}$$

where k denotes the iteration step of the SCP, $\boldsymbol{s}_e[k]$ and $\boldsymbol{s}_i[k]$ are equality and inequality slack variables to prevent artificial infeasibility, and w_e, w_i penalize their use. The final constraint enforces that $\|\boldsymbol{c}_1\| \leq d_i$ but also that each convex sub-problem be limited to a step of size d_{step} for stability. For now we assume a cost form $J(\boldsymbol{V}_i) = J(\boldsymbol{V}_{i-1}) + \Gamma_J \boldsymbol{c}_j$. The functions of the prior step on $\mathcal{C}^{(N,j)}$, given below, are not affected by the k^{th} solution for $\delta\boldsymbol{c}_1$:

$$J_{[k-1]} = J(\boldsymbol{V}_{i-1}) + \Gamma_J \boldsymbol{c}_j[k-1] \tag{20a}$$

$$\boldsymbol{g}_{[k-1]} = \boldsymbol{g}(\boldsymbol{V}_{i-1}) + \Gamma_g \boldsymbol{c}_j[k-1] \tag{20b}$$

$$\boldsymbol{h}_{[k-1]} = \boldsymbol{h}(\boldsymbol{V}_{i-1}) + \Gamma_h \boldsymbol{c}_j[k-1] \tag{20c}$$

The vector terms $\mathrm{D}J_{[k-1]}$ and $\mathrm{D}^2 J_{[k-1]}[\boldsymbol{\Delta}]$ are given below:

$$\mathrm{D}J_{[k-1]} = \Gamma_J \left.\frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\right|_{\boldsymbol{c}_j[k-1]} \tag{21a}$$

$$\mathrm{D}^2 J_{[k-1]}[\boldsymbol{\Delta}] = \Gamma_J \left.\frac{\partial}{\partial \boldsymbol{c}_1}\left(\frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\boldsymbol{\Delta}\right)\right|_{\boldsymbol{c}_j[k-1]} \tag{21b}$$

where $\boldsymbol{\Delta}$ is a placeholder vector, representing a direction in $\mathbb{R}^N$, introduced to avoid tensor notation. Alternatively to the quadratic expansion of the cost, a convex cost function $f_J : \boldsymbol{c}_j \to \mathbb{R}^+$ can be evaluated as $f_J(\boldsymbol{c}_j[k-1] + \frac{\partial \boldsymbol{c}_j}{\partial \boldsymbol{c}_1}\delta\boldsymbol{c}_j[k])$ for the SCP sub-problem using CVXPY. The analytic expressions for the derivative terms in Eq. (21), being derivatives of a well-ordered collection of monomials, are easy to compute. Indeed they themselves are linear functions of the monomials, specifically the terms up to order $j-1$. All quantities required for setting up the convex sub-problem of Eq. (19) are thus the pre-computed matrices Γ_g, Γ_h, Γ_J, and the nonlinearly updated $\boldsymbol{c}_j[k-1]$. This makes the setup of each iteration numerically trivial. Between iterations, it is only necessary to do two things. First, apply the nonlinear update, $\boldsymbol{c}_j[k] = \boldsymbol{E}_j(\boldsymbol{c}_1[k-1] + \delta\boldsymbol{c}_1[k])$. This update is a retraction of the linear step from the tangent space back onto the manifold. Second, recompute the monomial Jacobian and Eqs. (20) and (21) with the new $\boldsymbol{c}_j$ as well. A feasible solution is obtained by satisfaction of Eq. (19) with numerically near-zero slack variables, i.e. $\boldsymbol{s}[k] \approx \boldsymbol{0}$, for a nonlinearly updated $\boldsymbol{c}_j[k]$. The algorithm stops when there is no decrement of the cost for any feasible small steps $+\delta\boldsymbol{c}_j$ in the vicinity of a feasible $\boldsymbol{c}_j^*$.

To implement this SCP scheme, we use the popular CVXPY[12,13] to process the convex sub-problem, calling ECOS[14] or Clarabel.[15] To represent and manipulate the monomials, we employ a computerized monomial algebra in which the set of monomials is represented as an array, tracking the leading coefficient as well as the exponents of each constituent variable, e.g. $qx_1^a x_2^b x_3^c$ becomes $[q, a, b, c]$. This allows automation of differentiation operations. We developed Python-based tools for these operations, as well as for other tasks, e.g. computing matrices Γ_g, Γ_h, and Γ_J. See References 16, 17 for more information on computerized representation and manipulation of monomials. All examples in this paper are solved using the same custom solver based on the monomial method.

Strategies for Accommodating Larger Problems

Here we discuss strategies for managing the number of required monomial computations, vital to ensure practical implementation for problems of many variables.

Pruning As was previously noted, removing monomials from the formulation whose associated expansion coefficients for $J, \boldsymbol{g}, \boldsymbol{h}$ are all zero reduces the dimension of the Euclidean space in which the problem manifold is embedded, without affecting the problem computations or the submanifold nature of $\mathcal{C}^{(N,j)}$. Specifically, removing any unused monomials has the effect of reducing the number of monomial computations needed by the retraction operation $\boldsymbol{c}_j = \boldsymbol{E}_j(\boldsymbol{c}_1)$.

Product Manifolds It is often the case that not all free variables will be nonlinearly correlated. Consider the ubiquitous task of executing a two-burn transfer from initial state $\boldsymbol{X} = (\boldsymbol{r}_0^\top, \boldsymbol{v}_0^\top)^\top$ at $t = 0$ to a goal state $\boldsymbol{X}_{\text{goal}} = (\boldsymbol{r}_{\text{goal}}^\top, \boldsymbol{v}_{\text{goal}}^\top)^\top$ at $t = t_f$, for which $\boldsymbol{V} = (\Delta\boldsymbol{v}_1^\top, \Delta\boldsymbol{v}_2^\top)^\top$:

$$\begin{aligned} \min_{\Delta\boldsymbol{v}_1, \Delta\boldsymbol{v}_2} \quad & \|\Delta\boldsymbol{v}_1\| + \|\Delta\boldsymbol{v}_2\| \\ \text{subject to} \quad & \boldsymbol{r}_{\text{goal}} = \boldsymbol{\varphi}_{r,t_f}(\boldsymbol{r}_0, \boldsymbol{v}_0 + \Delta\boldsymbol{v}_1) \\ & \boldsymbol{v}_{\text{goal}} = \boldsymbol{\varphi}_{v,t_f}(\boldsymbol{r}_0, \boldsymbol{v}_0 + \Delta\boldsymbol{v}_1) + \Delta\boldsymbol{v}_2 \end{aligned} \tag{22}$$

where $\boldsymbol{\varphi}_t$ denotes the flow of the dynamics for some duration t, and this is factored into its position and velocity components for the orbit problem. In this case, a nonlinear expansion of the cost and constraints yields no mixing terms between $\Delta\boldsymbol{v}_1 \in \mathbb{R}^3$ and $\Delta\boldsymbol{v}_2 \in \mathbb{R}^3$. Thus it is not necessary to consider mixing terms such as $\delta\Delta v_{1,x}^2 \delta\Delta v_{2,y}$. To optimize this problem, we instead seek the optimal $\boldsymbol{c}_{j,1}^{(\Delta\boldsymbol{v}_1)}$ and $\boldsymbol{c}_{j,2}^{(\Delta\boldsymbol{v}_2)}$ satisfying the following, transforming the constraints of Eq. (22):

$$\Gamma_{g,1}\boldsymbol{c}_{j,1} + \Gamma_{g,2}\boldsymbol{c}_{j,2} = -\boldsymbol{g}(\boldsymbol{V}_{i-1}) \tag{23}$$

where $\boldsymbol{c}_{j,1} \in \mathcal{C}_1^{(3,j)}$ and $\boldsymbol{c}_{j,2} \in \mathcal{C}_2^{(3,j)}$ live on two manifolds of identical structure, each embedded in two identical higher-dimensional Euclidean spaces of dimensions K_j. The solution to this problem is a stationary point on the *product manifold*[9] $\mathcal{C}^{(3,j)} \times \mathcal{C}^{(3,j)}$, which is itself a Euclidean submanifold, whose tangent bundle is the union of the tangent bundles of the constituent sub-manifolds. Posing the problem in this way allows us to keep the monomial expansion compact, e.g. for order $j = 4$, Eq. (23) has unified $\Gamma_g = [\Gamma_{g,1}, \Gamma_{g,2}] \in \mathbb{R}^{6\times 68}$ for unified compact $\boldsymbol{c}_j = (\boldsymbol{c}_{j,1}^\top, \boldsymbol{c}_{j,2}^\top)^\top \in \mathbb{R}^{68}$ instead of $\Gamma_g \in \mathbb{R}^{6\times 209}$ for a unified sparse $\boldsymbol{c}_j \subseteq \mathbb{R}^{209}$. This approach notably saves computation time in the DA-based step where problem partials are computed.

Problem Reformulation It is often possible to reformulate the problem to limit dimensionality, by carefully designing the description to avoid unnecessary couplings. This can often be done at the cost of introducing extra constraints. Multiple shooting is one way of breaking a problem into decoupled variables, and we provide an example later in the paper.

Identifying Dominant Nonlinearities Now we discuss modifying the monomial set to keep only the dominant nonlinear terms. Similar to pruning, we seek to reduce the number of monomial computations, but this time we accept marginal error by removing monomials whose effect is nonzero but sub-dominant. This can be done systematically, which we now discuss. For the sake of simplicity, consider a scalar nonlinear function $g(\boldsymbol{x}_r + \delta\boldsymbol{x}) \approx g(\boldsymbol{x}_r) + \Gamma_g \boldsymbol{c}_j^{(\delta\boldsymbol{x})}$. Without loss of generality, let the problem be normalized such that all coordinates are of equal magnitude and furthermore the region of validity is $\|\delta\boldsymbol{x}\| \leq d < 1$. Then, within a certain region of interest $\|\delta\boldsymbol{x}\| \leq \alpha \leq d$, replace all terms in the expansion by their associated maximum scale, shown below to order 3:

$$\Gamma_g\boldsymbol{c}_3 \sim \Gamma_g[1]\alpha + \ldots + \Gamma_g[N]\alpha + \Gamma_g[N+1]\alpha^2 + \ldots + \Gamma_g[K_2]\alpha^2 + \Gamma_g[K_2+1]\alpha^3 + \ldots + \Gamma_g[K_3]\alpha^3 \tag{24}$$

where ellipses imply the inclusion of all terms at each order, $\Gamma_g[i]$ denotes the absolute value of the i^{th} element of Γ_g, and K_j denotes the number of elements (number of unique monomials) up to and

including some integer order j. Consider some small term $\Gamma_{g,\varepsilon}$ at order q, much smaller than other terms in Γ_g at the same order. Its term in the expansion will thus be $\sim \Gamma_{g,\varepsilon}\alpha^q$. We can remove this component from Γ_g and its associated monomial from $\boldsymbol{c}_j$, with the expectation of inducing error of $\mathcal{O}(\delta x^{q+b})$ in g for some higher order $q+b > q$ if the following holds:

$$|\Gamma_{g,\varepsilon}| < |\Gamma_g[\eta]|\alpha^b \; \forall \; \eta \in [K_{q+b-1}+1, K_{q+b}] \tag{25}$$

Similar logic applies to the case of a vector function $\boldsymbol{g}$, and also such a scaling analysis should be performed taking into account all problem functions $J, \boldsymbol{g}, \boldsymbol{h}$. This process can be automated.

Coordinate Transformations Often times, simple problems in astrodynamics will exhibit one or more dominant *stretching directions* in state space, $\boldsymbol{\xi} \in \mathbb{R}^N$, for which a function of interest $\boldsymbol{g}$ will be particularly sensitive to projections of $\delta\boldsymbol{x}$. For example, in References 18 and 19, $\boldsymbol{g} = \boldsymbol{\varphi}_{t_f-t_0}(\boldsymbol{x}_r + \delta\boldsymbol{x}) - \boldsymbol{\varphi}_{t_f-t_0}(\boldsymbol{x}_r)$, where $\delta\boldsymbol{x}$ is some initial state deviation, and the most sensitive direction is identified as the eigenvector associated with the maximum eigenvalue of the Cauchy-Green tensor $C_{(t_f,t_0)} = \Phi^\top(t_f,t_0)\Phi(t_f,t_0)$. In such problems, for the higher-order terms, it makes sense to prioritize the projections along the dominant eigendirection(s) of the Cauchy-Green tensor, reducing significantly the size of the tensors Γ_g and also the size of the monomial basis. Here this idea is generalized for some arbitrary function $\boldsymbol{g}$, facilitating a linear change-of-basis from $\boldsymbol{c}_j^{(\delta\boldsymbol{x})}$ to one $\boldsymbol{c}_j^{(\delta\boldsymbol{z})}$ which can be easily pruned.

Let $\Gamma_{g,j}$ denote Γ_g up to and including order j. The linearly dominant direction, i.e. the direction along which a change $\delta\boldsymbol{x}$ will yield the greatest change in $\|\boldsymbol{g}\|$, can be derived as the maximum eigenvalue of $C_g = \Gamma_{g,1}^\top\Gamma_{g,1}$. Because C_g is real and symmetric, all eigenvalues are real and the basis is orthonormal. Let the eigenbasis be sorted as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$, each λ_i having a corresponding $\boldsymbol{\xi}_i$, with the resulting coordinate transformation rewriting $\delta\boldsymbol{x}$ as a projection along each basis direction:

$$\delta\boldsymbol{x} = \sum_{i=1}^{N} \boldsymbol{\xi}_i \delta z_i \tag{26}$$

where $\delta z_i = \delta\boldsymbol{x} \cdot \boldsymbol{\xi}_i$. A linear transformation $\delta\boldsymbol{x} = R\delta\boldsymbol{z}$ can thus be defined, with R built column-wise from the $\boldsymbol{\xi}_i$. The dominant stretching direction is along coordinate δz_1; the second most important stretching direction is along δz_2, etc. In this regard, at second-order, it may be true that keeping only δz_1^2 in the representation $\boldsymbol{c}_j^{(\delta\boldsymbol{z})}$ captures the most important second-order effects, or in the case of two dominant directions, the terms δz_1^2, $\delta z_1\delta z_2$, and δz_2^2. This can extend to higher orders as well, inspiring significant reduction in the dimensionality of $\boldsymbol{c}_j^{(\delta\boldsymbol{z})}$. To inform how a tensor is transformed from its form in the old coordinates $\Gamma_g^{(\delta\boldsymbol{x})}$ to its form in the new coordinates $\Gamma_g^{(\delta\boldsymbol{z})}$, we discuss briefly how its components are updated. Adopting index notation, a chain rule procedure similar to Reference 18 can be applied, demonstrated explicitly below for linear and quadratic terms:

$$\delta x^{\kappa_1} = \frac{\partial \delta x^{\kappa_1}}{\partial \delta z^{\gamma_1}} \delta z^{\gamma_1} = R^{\kappa_1,\gamma_1}\delta z^{\gamma_1} \tag{27a}$$

$$\frac{\partial g^i}{\partial \delta z^{\gamma_1}} = \frac{\partial g^i}{\partial \delta x^{\kappa_1}} R^{\kappa_1,\gamma_1} \tag{27b}$$

$$\frac{\partial^2 g^i}{\partial \delta z^{\gamma_1}\partial \delta z^{\gamma_2}} = \frac{\partial^2 g^i}{\partial \delta x^{\kappa_1}\partial \delta x^{\kappa_2}} R^{\kappa_1,\gamma_1} R^{\kappa_2,\gamma_2} \tag{27c}$$

where the first expression is the component-wise equivalent of Eq. (26) and we use the usual superscript notation for tensor components. In this regard, an expansion Γ_g can be computed in many variables in $\boldsymbol{c}_j^{(\delta \boldsymbol{x})}$, then used to identify a more compact representation in $\boldsymbol{c}_j^{(\delta \boldsymbol{z})}$. One limitation of the aforementioned transformation is that it only leverages the linearized information in Γ_g to render new working coordinates. Any nonlinear coordinate transformation between $\delta \boldsymbol{x}$ and new coordinates $\delta \boldsymbol{z}$ admitting a convergent Taylor series can also be approximated in terms of monomials. Thus, the same pre-computed information can be leveraged to identify fortuitous order-j nonlinear transformations to new working coordinates, with a corresponding transformation of the constraint equations. Discussion of this goes beyond the scope of this introductory work.

EXAMPLE APPLICATIONS

Example 1, A Feasibility Problem: Computing Periodic Orbits

For an approachable first example of how our method is used to solve a problem of interest, consider first the familiar task of computing periodic orbits in the circular restricted three-body problem (CR3BP). See Appendix A for the necessary equations. We seek an augmented set of initial conditions and propagation time $\boldsymbol{V} = (\boldsymbol{X}_0^\top, T)^\top$ such that the orbit is periodic,

$$\boldsymbol{\varphi}_T(\boldsymbol{X}_0) - \boldsymbol{X}_0 = \boldsymbol{0} \tag{28}$$

There is no cost function, so we are solving a *feasibility problem* in the form of Eq. (2). With this formulation, $\boldsymbol{V} \in \mathbb{R}^6 \times \mathbb{R}^+$ has 7 free variables but Eq. (28) enforces a periodicity condition of only 6 constraints. Appending a requirement specifying the desired Jacobi constant $C_{J,\text{spec}}$:

$$\underset{\boldsymbol{X}_0, T}{\text{solve}} \quad \boldsymbol{F} = \begin{pmatrix} \boldsymbol{\varphi}_T(\boldsymbol{X}_0) - \boldsymbol{X}_0 \\ C_J(\boldsymbol{X}_0) - C_{J,\text{spec}} \end{pmatrix} = \boldsymbol{0} \tag{29}$$

In this example, a highly elliptical cislunar trajectory is computed. Integrating to obtain the higher-order expansions, a corrected member of the Southern Halo family is desired.

We now discuss the computation of Γ_F and Γ_J. The initial state $\boldsymbol{X}_0$ is initialized as a vector of DA variables using daceypy, along with the orbit propagation time T. The propagation time can be accommodated as a DA variable via a transformation of time to $\tau = t/T$,[20] for which the equations of motion become $\frac{\mathrm{d}}{\mathrm{d}\tau}(\boldsymbol{X}) = T\boldsymbol{f}(\boldsymbol{X})$. After integration to the final time and evaluation of $\boldsymbol{F}$ and J, the Taylor maps are obtained, which are expansions with respect to the elements of $\delta \boldsymbol{X}_0$ and δT. These are put in the matrix forms Γ_F and Γ_J, row-wise based on their location in Eq. (29), and column-wise by following the specified monomial ordering. The numerical performance may be improved by setting the smallest number of free variables $\boldsymbol{V}$ possible, so we can also choose $\boldsymbol{V}$ composed of the nonzero elements at apoapsis, plus orbit propagation time T:

$$\boldsymbol{V} = (x_0, z_0, \dot{y}_0, T)^\top \tag{30}$$

We explore expansions up to order $j = 6$, but show explicitly the first few terms of $\boldsymbol{c}_2$ below:

$$\boldsymbol{c}_2 = (\delta x_0, \delta z_0, \delta \dot{y}_0, \delta T, \delta x_0^2, \delta x_0 \delta z_0, \delta x_0 \delta \dot{y}_0, \delta x_0 \delta T, \delta z_0^2, \ldots)^\top \subseteq \mathbb{R}^{14} \tag{31}$$

Meanwhile the expansion up to 3rd-order would have $\boldsymbol{c}_3 \subseteq \mathbb{R}^{34}$, and up to fourth-order would have $\boldsymbol{c}_4 \subseteq \mathbb{R}^{69}$. By contrast, the original formulation of $\boldsymbol{V} \in \mathbb{R}^6 \times \mathbb{R}^+$ would have $\boldsymbol{c}_4 \subseteq \mathbb{R}^{329}$.

Using the free variable choice of Eq. (30), we compute expansions for an initial guess given in Table 1, cataloguing runtime and accuracy of the expansion vs. expansion order. In particular, we're interested in approximating a "nearby" lunar southern L2 NRHO with a Jacobi constant of $C_J = 3.0499728$, selected from the JPL Three-Body Periodic Orbit Catalog. This corresponds to a NASA Gateway-like orbit period of approximately 6.5 days (1.466695 TU). Its initial conditions are also provided in Table 1. Fig. 3 provides at every order j the error norm $\|\overline{\boldsymbol{X}}_f - \boldsymbol{X}_f\|$, computed from approximation of the final state $\boldsymbol{X}_f = \boldsymbol{\varphi}_{T+\delta T}(\boldsymbol{X}_0+\delta\boldsymbol{X}_0)$ via the expansion $\overline{\boldsymbol{X}}_f = \boldsymbol{\varphi}_T(\boldsymbol{X}_0)+\Psi_j\boldsymbol{c}_j$, as well as the compute time. Note their opposing trends as expansion order is increased. The matrix $\Psi_j \in \mathbb{R}^{6\times K_j}$ contains the information of a flattened representation of a higher-order "state transition tensor", but also includes terms (columns) to capture the effect of a variation in integration time δT, including all mixed terms e.g. $\delta z_0 \delta T^2$.

Table 1. Parameters for CR3BP Orbit (Example 1)

Parameter	Values
Mass ratio μ	$\mu = 1.215058560962404 \times 10^{-2}$
Reference trajectory	$x_0 = 1.02166$, $z_0 = -0.18566$, $\dot{y}_0 = -0.0955$, $T = 1.462$
Reference type	Lunar, highly elliptical, non-periodic
NRHO initial conditions	$x_0 = 1.01866$, $z_0 = -0.17967$, $\dot{y}_0 = -0.095814$, $T = 1.466695$

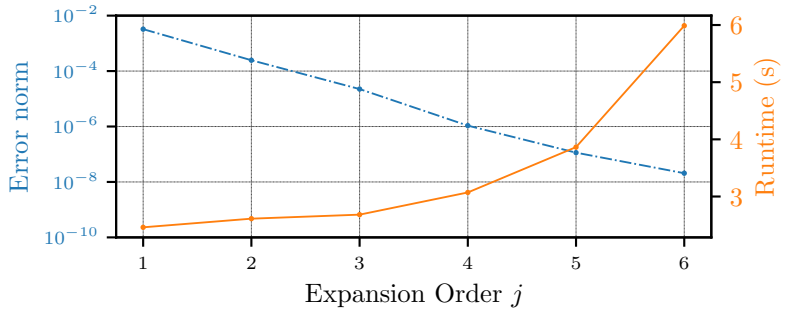


Figure 3. Expansion Error and Compute time vs. Order in CR3BP (Example 1)

Using the pre-computed expansion at order $j = 5$, we now seek to compute nearby periodic orbits belonging to the L2 Southern Halo family using the SCP scheme defined in Eq. (19). Because the relevant expansion info has already been pre-computed, namely Γ_F, orbits of a desired Jacobi constant can be computed quite quickly from that information. For this application, the state-dependent cost term is null, $J(\boldsymbol{c}_j) = 0$, thus the optimizer will approximate a feasible solution by minimizing the equality constraint slack cost, $\frac{1}{2}w_e\boldsymbol{s}^\top\boldsymbol{s}_e$. We apply the trust region constraints $d_{\text{step}} = 2.5 \times 10^{-3}$ and $d_i = 0.1$. All parameters for the SCP are listed in Table 2. Tasked with computing a periodic orbit at the value of Jacobi constant $C_J = 3.05$, the SCP scheme converges (meeting specified slack variable norm tolerance) in 5 iterations, executing in a total time of only $\sim$0.033s using a 2023 MacBook Pro with the Apple M2 Pro chip. The norm of each step $\|\delta\boldsymbol{c}_1[k]\|$ and the quadratic slack cost are given in Figure 4. This confirms the proper functioning of the trust region, with the first four iterations hitting the norm bound. Increasing d_{step} to 10^{-2}, the solution is reached in 3 iterations, executing within $\sim$0.019s. This is in contrast to the runtime to compute a given periodic orbit via more standard methods. Consider a differential correction implementation in Python on the same machine. Integrating the state transition matrix in parallel with the equations of motion using solve_ivp from scipy.optimize, and constructing the Jacobian for each iteration etc.,

just a single differential correction step takes ∼0.1s or more.

Table 2. SCP Scheme Info for CR3BP Orbits (Example 1)

Parameter	Values
Solver and platform	CVXPY with ECOS, 2023 MacBook Pro (Apple M2 Pro)
Runtimes: per iteration, total	$\sim 6 \times 10^{-3}$ s, ~ 0.02 s
Cost function	Equality constraint slack penalty, $J = \frac{1}{2} w_e \boldsymbol{s}_e^\top \boldsymbol{s}_e$
Trust region constraints	range $0.001 < d_{\text{step}} \leq 0.01$, $d_i = 0.1$
Weights	$w_e = 10^3$
Stopping condition	Feasibility only, $\|\boldsymbol{F}(\boldsymbol{V}_{i-1}) + \Gamma_F \boldsymbol{c}_j\| < 9 \times 10^{-8}$

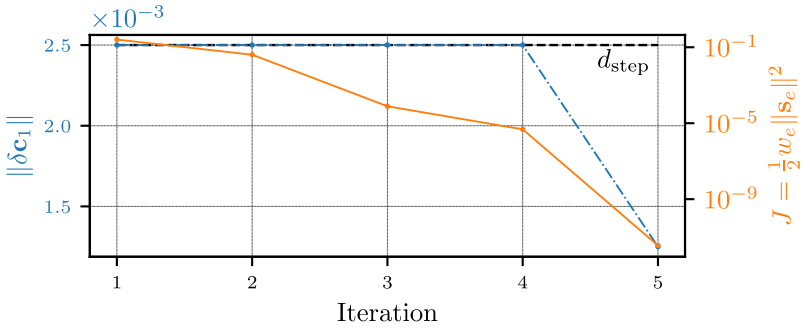


Figure 4. Example SCP Results for computing an NRHO Orbit (Example 1)

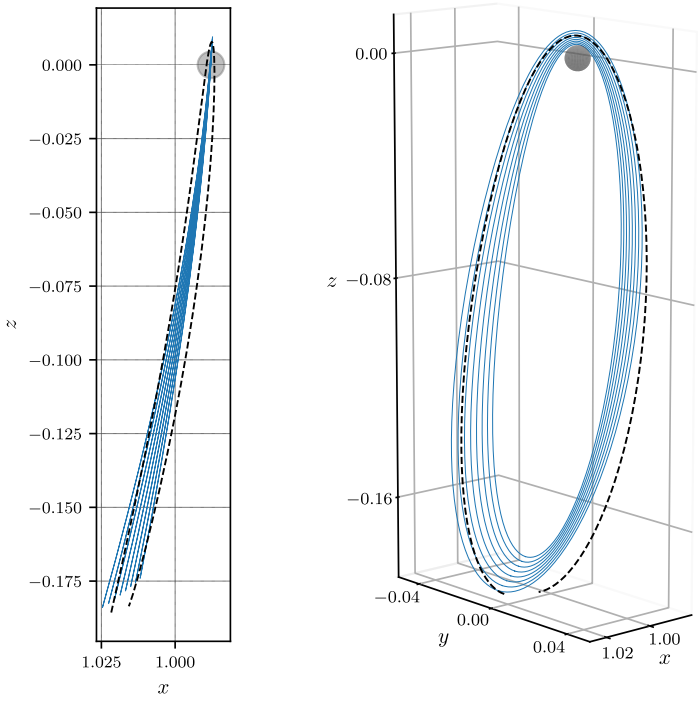
Using the pre-computed expansion and SCP scheme, a few orbits in the range $C_J \in [3.044, 3.058]$ are computed, each in comparable time as the prior example, ~ 0.02s. These are provided in Figure 5, where the non-periodic reference trajectory is shown as a dashed line, the periodic orbits are given as solid lines, and the moon as a gray sphere. The optimizer is able to approximate all members of the family within the established trust region bound $\|\boldsymbol{c}_j\| < 0.1$. This first example is meant to provide an approachable introduction to using the monomial method, showing that one nonlinear expansion can be reused to compute a variety of desired solutions that are nearby in state space. However, we emphasize that other DA approaches also exist for computing periodic orbit families – see e.g. Ref. 20, where expansion about a converged periodic orbit in terms of a single DA variable, a natural parameter s, is used to construct locally part of the halo family. Our methodology here is a bit more general because the expansions from a non-periodic reference are used.

Example 2, An Optimality and Continuation Problem: Orbital Transfers

Now we explore the efficient computation of general impulsive transfers in the CR3BP, with a primary focus on fixed-time "point A to point B" transfers. The methodology developed in this paper is localized, so we need of a strategy for efficiently generating a reasonable initial guess. Such a strategy is now briefly outlined. The vector field in the CR3BP is Lipschitz continuous everywhere except at the primaries, and the flow of the dynamics is a diffeomorphism. Thus, dimensionality and smoothness of submanifolds will be preserved by the flow of the dynamics. Defining a 2D submanifold for maneuvers of a fixed magnitude applied at a particular point A in phase space:

$$\boldsymbol{X}_0 = \{\boldsymbol{r} : \boldsymbol{r} = \boldsymbol{r}_{\text{A}}\} \times \{\boldsymbol{v} : \|\boldsymbol{v} - \boldsymbol{v}_{\text{A}}\| = \Delta v\} \tag{32}$$

The departure maneuver $\Delta \boldsymbol{v} \in \mathcal{B}_{\Delta v}$ lives on the surface of a hollow (2D) ball, and it can be shown that $\boldsymbol{X}_0$ is constrained to a smooth 2D submanifold of $\mathbb{R}^6$, leveraging Eq. (32) to create a



(a) (b)

Figure 5. Lunar Reference Orbit and Recovered NRHO Orbits (Example 1)

defining function, and reusing the earlier sub-manifold test from Reference 9. We can consider fixed-time reachability from point A for a particular departure delta-V magnitude Δv and transfer time $\Delta t = t_f - t_0$ via the following mapping from initial delta-V to final state submanifolds:

$$\mathcal{B}_{\Delta v} \underset{\varphi_{\Delta t}}{\rightarrow} \mathcal{S}_{\Delta v, \Delta t} \tag{33}$$

where the final state $\boldsymbol{X}_f \in \mathcal{S}_{\Delta v_1, \Delta t}$ lies on a smooth 2D submanifold of phase space, representable as a 2D surface in 3D position space. As Δv is increased, the volume enclosed by this surface expands, and the surface deforms, possibly also developing self-intersections (note velocities $\boldsymbol{v}_f$ remain unique for such intersecting positions). Initializing by sampling on $\mathcal{B}_{\Delta v}$ for a small Δv, then repeatedly re-sampling for higher delta-V magnitudes with a prioritization of points moving towards point B, an iterative search for the minimal delta-V to reach point B can be performed efficiently. Thus the departure delta-V norm for which $\boldsymbol{r}_{\text{goal}}$ is first reachable may be identified. Such a scheme executes in a few seconds in Python for nearby orbits. Figure 6 shows an example planar transfer initial guess from a specified starting point on an L_1 Lyapunov orbit with $C_J = 3.079$ to a specified end point on a Distant Prograde orbit with $C_J = 3.048$ in $\Delta t = 2.6$ TU. For this planar example, $\mathcal{B}_{\Delta v}$ and $\mathcal{S}_{\Delta v, \Delta t}$ are one-dimensional, aiding visualization. Contours for 4 guesses of departure delta-V magnitude, $\Delta v_4 > \Delta v_3 > \Delta v_2 > \Delta v_1$, are provided, with Δv_4 reaching $\boldsymbol{r}_{\text{goal}}$ from $\boldsymbol{r}_0$.

The monomial method is now leveraged in a scheme to optimize impulsive transfer orbits in the CR3BP. An initial guess can be generated from a prototype of the aforementioned method in $\sim$10 s of computation, minimizing $\tilde{J} = \|\Delta \boldsymbol{v}_1\|$. We start instead with a rather poor initial guess to demonstrate superiority to standard linear correction. For an optimal ballistic Lambert-type solution, we

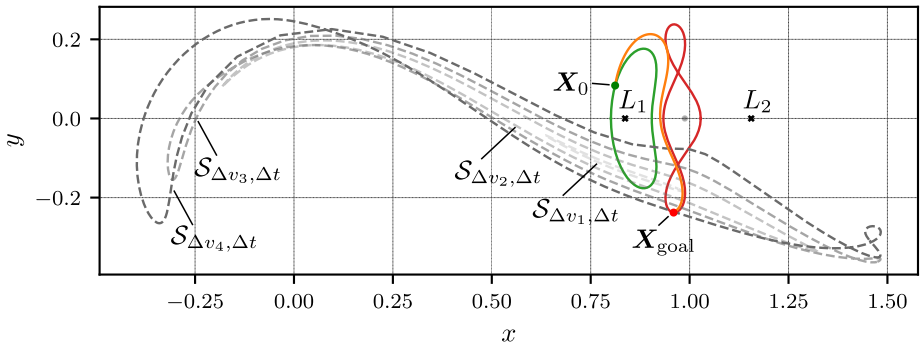


Figure 6. Visualizing Transfer Initial Guess Generation via Reachability Submanifolds

seek the minimizer of $J = \|\Delta \boldsymbol{v}_1\| + \|\Delta \boldsymbol{v}_1\|$.* This problem, already previously outlined in Eqs. (22) - (23), is now solved. We seek an optimized fixed-time impulsive two-burn transfer from an L2

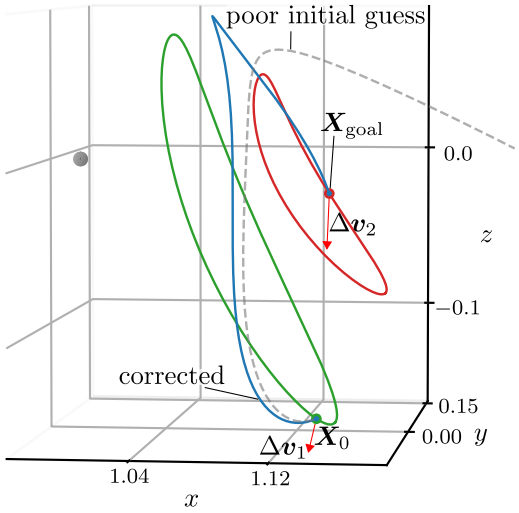


Figure 7. Halo-to-Halo transfer initial guess and corrected (Example 2)

Southern Halo of $C_J \sim 3.06126$ to another of $C_J \sim 3.12603$ in 2.6 TU. The departure orbit phase, in time since apolune, is 0.25 TU. The arrival orbit phase is ∼2.52 TU post-apolune, chosen arbitrarily. The initial guess has a total delta-V of 0.2673 DU/TU and does not meet the equality constraint $\boldsymbol{X}_f = \boldsymbol{X}_{\text{goal}}$, with egregious violation visible in Fig. 7. Nonetheless we expand about this solution. The time to pre-compute Γ_g varies from ∼1 s for $j = 1$ (e.g. linear) to ∼1.3 s for $j = 5$, ∼1.9 s for $j = 7$, and ∼3.5 s for $j = 9$. The monomial SCP scheme is applied for order $j = 7$ with the parameters in Table 3, solving in 12 iterations, total time ∼0.2 s, and achieving error of $\|\boldsymbol{g}\| \sim 10^{-4}$. The violation decreases further with higher order j, and conversely increases with lower order. Hence a single linearized correction achieves an error of $\|\boldsymbol{g}\| \sim 0.07$, requiring further trajectory updates and re-linearizations. Meanwhile, higher-order expansions can yield solutions accurate to the limits of the convex solver, requiring no reference trajectory updates at all. The corrected delta-V is 0.2737

*We note, for this particular fixed-time two-burn example, that locally the optimal solution seems to also be the only feasible solution, so the cost function is not necessary to compute the transfer – the problem could be solved similarly to Example 1, recovering the same result. It's not obvious *a priori* in the CR3BP how many feasible transfers will exist, but in the case of such "Lambert" transfers, we posit that they will tend to be isolated from one another (like the two in the Keplerian case), and in typical cases at most one satisfactory transfer will tend to be available within a given tensor expansion about an initial guess.

DU/TU. There was little marginal pre-compute cost for nonlinear expansions of order $1 < j \leq 7$. The SCP solve time also scales favorably with order, increasing only from $\sim$0.1 s at order $j = 5$ to $\sim$0.19 s for order $j = 9$. The advantages are thus clear. In terms of computational time $\mathcal{T}$, we observe the trend $\mathcal{T}(\text{initial guess}) > \mathcal{T}(\text{tensor expansion}) \gg \mathcal{T}\left(\text{solution on } \mathcal{C}^{(N,j)}\right)$.

Table 3. SCP Scheme Info for Halo-to-Halo transfer (Example 2)

Parameter	Values
Solver and platform	CVXPY with Clarabel, 2023 MacBook Pro (Apple M2 Pro)
Runtimes: per iteration, total	$\sim 10^{-2}$ s, $\sim$ 0.1-0.2 s (depending on order j)
Cost function	Min. delta-V + slack penalty, $J = \|\Delta \boldsymbol{v}_1\| + \|\Delta \boldsymbol{v}_2\| + \frac{1}{2} w_e \boldsymbol{s}_e^\top \boldsymbol{s}_e$
Trust region constraints	$d_{\text{step}} = 0.003$, $d_i = 0.03$
Weights	$w_e = 5 \times 10^5$
Stopping condition	"Cost stall", $\lvert\delta J\rvert / J < 10^{-4}$ for 5 or more iterations; $\|\boldsymbol{g}\| < g_{\text{tol}}$

The corrected two-burn solution is sub-optimal when the option for more than two maneuvers is allowed. We now compute the true impulsive optimal, using the monomial method with a multiple-shooting procedure. We use the behavior of the primer vector of the two-impulse transfer to inform whether interior impulses are required, and roughly when they should be (e.g., Refs. 21, 22). The multiple-shooting states are defined on a pruned product manifold leveraging the following equations shown for the general case of K intermediate burns (hence $K + 2$ total burns):

$$\boldsymbol{c}_1 = \left(\Delta \boldsymbol{v}_1^\top,\ \Delta t_1,\ \delta \boldsymbol{X}_{m_1}^\top,\ \Delta t_2,\ \ldots,\ \delta \boldsymbol{X}_{m_K}^\top,\ \Delta t_{K+1},\ \Delta \boldsymbol{v}_{K+2}^\top\right)^\top \tag{34a}$$

$$\boldsymbol{g} = \begin{pmatrix} \Delta t_1 + \Delta t_2 + \Delta t_3 - \Delta T \\ \boldsymbol{\varphi}_r(\boldsymbol{X}_0^+, \Delta t_1) - [I_{3\times3}\ 0_{3\times3}]\, \boldsymbol{X}_{m_1} \\ \boldsymbol{\varphi}_r(\boldsymbol{X}_{m_1}, \Delta t_2) - [I_{3\times3}\ 0_{3\times3}]\, \boldsymbol{X}_{m_2} \\ \vdots \\ \boldsymbol{\varphi}_r(\boldsymbol{X}_{m_{K-1}}, \Delta t_K) - [I_{3\times3}\ 0_{3\times3}]\, \boldsymbol{X}_{m_K} \\ \boldsymbol{\varphi}_r(\boldsymbol{X}_{m_K}, \Delta t_{K+1}) - [I_{3\times3}\ 0_{3\times3}]\, \boldsymbol{X}_{\text{goal}} \\ \boldsymbol{\varphi}_v(\boldsymbol{X}_{m_K}, \Delta t_{K+1}) + \Delta \boldsymbol{v}_{K+2} - [0_{3\times3}\ I_{3\times3}]\, \boldsymbol{X}_{\text{goal}} \end{pmatrix} = \boldsymbol{0} \tag{34b}$$

$$J = \|\Delta \boldsymbol{v}_1\| + \left\|[0_{3\times3}\ I_{3\times3}]\left(\boldsymbol{X}_{m_1} - \boldsymbol{\varphi}(\boldsymbol{X}_0^+, \Delta t_1)\right)\right\| + \sum_{p=2}^{K} \left\|[0_{3\times3}\ I_{3\times3}]\left(\boldsymbol{X}_{m_p} - \boldsymbol{X}_{m_{p-1}}\right)\right\| + \|\Delta \boldsymbol{v}_{K+2}\| \tag{34c}$$

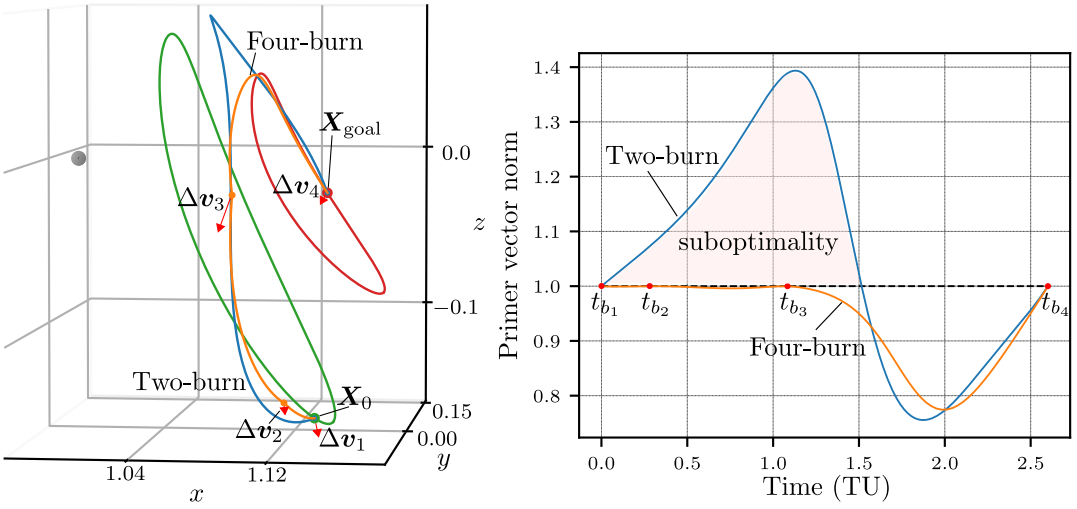
where ΔT denotes the (fixed) total transfer time, $\boldsymbol{X}_{m_p}$ is the state after the p^{th} intermediate burn, t_q is a corresponding propagation time after the q^{th} burn, $\boldsymbol{\varphi}_r$ selects just the position components of the propagated state, and $\boldsymbol{\varphi}_v$ the velocity, and $\boldsymbol{X}_0^+ = \boldsymbol{X}_0 + (\boldsymbol{0}_{3\times1}^\top\ \Delta \boldsymbol{v}_1^\top)^\top$. Note that the burn times are optimization decision variables, which is necessary to recover the true optimal solution. Because this is a multiple shooting approach, there is no nonlinear state coupling between node states/times. The constraint and cost equations in Eq. (34) are expanded about in a Taylor series and rendered in the monomial framework. For the case of $j = 4$ with four total burns (thus $K = 2$ inner nodes), the total number of monomials is thus 761 instead of 12649, therefore $\Gamma_g \in \mathbb{R}^{13\times761}$. Burns are added recursively: using the primer vector of the bi-impulse case to identify a candidate inner burn time and then differentially correcting, we produce a three-burn solution, then apply again a

similar primer vector-based procedure to correct to the four-burn solution, which is optimal and marginally less costly than the three-burn solution. Information about the two-, three-, and four-burn trajectories is summarized in Table 4. Note the interesting property that the final burn onto the arrival halo is almost removed in the optimal case.

Table 4. Halo-to-Halo Impulsive Transfer Solutions (Example 2)

Solution	Delta-V (DU/TU)	Maneuver Times (TU)
Two-burn	Total 0.27374; per maneuver (0.1189, 0.1549)	(0.0, 2.6)
Three-burn	0.21216; (0.08911, 0.1199, 0.0032)	(0.0, 1.0414, 2.6)
Four-burn	0.21203; (0.0652, 0.0316, 0.1119, 0.0034)	(0.0, 0.2733, 1.0862, 2.6)

The optimal transfer trajectory is shown in orange in Fig. 8(a). Its optimality is established by the requirement that $\|\boldsymbol{p}_r(t)\| \leq 1 \; \forall t$, with maneuvers occuring when $\|\boldsymbol{p}_r(t)\| = 1$, verified by inspection of Fig. 8(b). We truncate the expansion order to $j = 4$ and constrain $\|\boldsymbol{c}_1[i]\| \leq d_i$ with $d_i = 0.4$ in the call to the SCP, thus requiring repeatedly re-computing the higher-order expansion and solving a series of SCP problems until convergence to the optimal solution, because the needed change to the free variables exceeds the trust region of a single SCP call at order $j = 4$. In this manner the monomial method is applied sequentially, similarly to a standard linearization-based differential correction procedure. Unlike linearized differential correction, each intermediate solution $\boldsymbol{c}_1[i]$ yields a feasible trajectory, e.g. $\boldsymbol{g}[i] \approx \boldsymbol{0}$ (satisfied to an order $j = 4$ expansion in the free variables), but not satisfying the optimality conditions. The optimal solution is thus obtained by *continuation* through a family of feasible but sub-optimal solutions of decreasing cost. We observe typically a time of ∼2s to pre-compute the expansions, plus a runtime of ∼0.1s per call to the convex solver, with on average 10 calls per SCP problem, with a total sequence of 15 SCP problems solved to render the optimal trajectory. This is just a proof-of-concept with no code optimization or fine-tuning. A tradeoff study varying the order j and size of d_i and exploring resulting stability, accuracy, and total runtime could be conducted, but we leave this to future work.



(a) Two-burn and four-burn transfers

(b) Four-burn optimality based on primer vector

Figure 8. Halo-to-Halo optimal impulsive transfer (Example 2)

Example 3: Accommodating Stochasticity

A strong benefit of the monomial-based direct optimization approach is that we can leverage the traditional uncertainty quantification strengths of DA for the purposes of stochastic optimization. For example, consider again the nominal optimal 4-burn guidance policy from example 2, subject to some dispersion in initial conditions, in addition to the possibility of navigation and execution errors. In this dispersed context, the objective of stochastic guidance is that the dispersion in the final state is mitigated to a specified tolerance. For example, incorporating pre-specified dispersions in the initial state and departure burn, the problem defined by Eq. (34) is modified by the removal of $\boldsymbol{c}_{1,1} = (\Delta\boldsymbol{v}_1^\top, \Delta t_1)^\top$ from the optimization variables, and the modification of $\boldsymbol{g}$, with the final requirement $\mathbb{E}(\boldsymbol{X}_{K+2}) - \boldsymbol{X}_{\text{goal}} = \boldsymbol{0}$ (where $\mathbb{E}$ denotes the expected value) as well as the augmentation of constraints on the final state covariance P_f at arrival time t_f. Solving the resulting nonlinear stochastic optimal guidance problem is beyond the scope of this work, but the important information to solve it, namely the means for nonlinear propagation of uncertainty distributions, is *already* pre-computed in the implementation to obtain the nominal solution. As a brief preview of this capability, consider a pre-defined distribution of states at time $t_1 = \Delta t_1$ (the timing of the nominal second burn). Samples from this distribution can be propagated extremely efficiently to the next burn time $t_2 = \Delta t_1 + \Delta t_2$ via the following equation:

$$\boldsymbol{X}(t_2) = \boldsymbol{X}_r(t_2) + \Gamma_\varphi \boldsymbol{c}_{j,2} \tag{35}$$

where $\boldsymbol{c}_{j,2}$ is built from the monomials of $\boldsymbol{X}_{m_1}$, and Γ_φ is built from a Taylor expansion of the flow $\boldsymbol{\varphi}(\boldsymbol{X}_{m_1}, \Delta t_2)$ which was already used to compute part of Γ_g for example 2. In other words this matrix was "salvaged" from the larger Γ_g. This matrix can be viewed simply as a 2D equivalent of the familiar state transition tensor. The compute time for nonlinearly "propagating" 1000 sample points of $\boldsymbol{c}_{j,2} \in \mathcal{C}^{(6,4)}$ in our Python routine is $\sim$ 0.2s. Note that in a stochastic optimal guidance setting, we would have to ensure that the expansion order is chosen properly to accommodate expected dispersions based on navigation and propulsion system performances.

CONCLUSIONS

This paper presented a monomial method for nonlinear optimization in astrodynamics, using differential algebra (DA) to approximate nonlinear functions with multivariate monomials, developing a framework and numerical tools to pose and solve optimization problems numerically efficiently, leveraging the fact that the problem state lies on a smooth Riemannian submanifold. The approach is designed for onboard applications where computational efficiency and predictability is key, such as onboard closed-loop guidance, or any other other repeated optimization tasks which reuse the underlying expansions. The ability to perform nonlinear sampling in the decision space at minimal extra cost will be applied to develop a stochastic approach, but could have additional applications for ensuring solver stability in challenging settings where sampling can augment the SCP approach. The methodology allows for a range of different balances of computational speed and accuracy based on expansion order.

The results demonstrate the method's capability to compute periodic orbits in the CR3BP with very low runtime and to solve more complex problems such as the optimization of impulsive transfers. The CR3BP was chosen simply as a convenient starting point for these astrodynamics examples, and the general methodology will also work in high-fidelity scenarios without fundamental modification. While sharing similarities with traditional DA-based approaches, the monomial

method offers broad flexibility, efficiently leveraging affine representations of higher-order expansions and providing a hierarchical approach for the use of such expansions in the context of direct optimization problems. Future work includes improvements to the prototype solver to improve runtime and stability, exploring more complex stochastic optimization examples, and enhancing the pruning and nonlinearity reduction strategies.

DISCLAIMER

Funded by the European Union (Horizon Europe, FAAST-MSCA, Grant ID: 101063274). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union, the European Research Executive Agency, or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] S. Boone and J. McMahon, "Orbital Guidance Using Higher-Order State Transition Tensors," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 3, 2021, pp. 493–504, 10.2514/1.G005493.

[2] R. S. Park and D. J. Scheeres, "Nonlinear Mapping of Gaussian Statistics: Theory and Applications to Spacecraft Trajectory Design," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1367–1375, 10.2514/1.20177.

[3] M. Willis, K. T. Alfriend, and S. D'Amico, "Second-Order Solution for Relative Motion on Eccentric Orbits in Curvilinear Coordinates," *AAS/AIAA Astrodynamics Specialist Conference*, No. AAS 19-810, American Astronautical Society, 2019.

[4] E. A. Butcher, T. A. Lovell, and A. Harris, "Third order Cartesian relative motion perturbation solutions for slightly eccentric chief orbits," *Advances in the Astronautical Sciences*, Vol. 158, No. AAS 16-496, 2016, pp. 3435–3454.

[5] M. Berz, "Chapter 2 - Differential Algebraic Techniques," *Modern Map Methods in Particle Beam Physics*, Vol. 108 of *Advances in Imaging and Electron Physics*, pp. 81–117, San Diego, CA: Academic Press, 1999, 10.1016/S1076-5670(08)70228-3.

[6] P. Di Lizia, R. Armellin, A. Morselli, and F. Bernelli-Zazzera, "High order optimal feedback control of space trajectories with bounded control," *Acta Astronautica*, Vol. 94, No. 1, 2014, pp. 383–394, 10.1016/j.actaastro.2013.02.011.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge University Press, 2004.

[8] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive Convexification of Non-Convex Optimal Control Problems with State Constraints," *IFAC-PapersOnLine*, Vol. 50, No. 1, 2017, pp. 4063–4069. 20th IFAC World Congress, 10.1016/j.ifacol.2017.08.789.

[9] N. Boumal, *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023, 10.1017/9781009166164.

[10] M. Rasotto, A. Morselli, A. Wittig, M. Massari, P. Di Lizia, R. Armellin, C. Valles, and G. Ortega, "Differential algebra space toolbox for nonlinear uncertainty propagation in space dynamics," *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, Darmstadt, Germany, 2016.

[11] Z. Lai and A. Yoshise, "Riemannian Interior Point Methods for Constrained Optimization on Manifolds," *Journal of Optimization Theory and Applications*, Vol. 201, No. 1, 2024, pp. 433–469, 10.1007/s10957-024-02403-8.

[12] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, Vol. 17, No. 83, 2016, pp. 1–5, 10.48550/arXiv.1603.00943.

[13] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, Vol. 5, No. 1, 2018, pp. 42–60, 10.1080/23307706.2017.1397554.

[14] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," *2013 European Control Conference (ECC)*, European Control Association, July 2013, pp. 3071–3076, 10.23919/ECC.2013.6669541.

[15] P. J. Goulart and Y. Chen, "Clarabel: An interior-point solver for conic programs with quadratic objectives," 2024, 10.48550/arXiv.2405.12762.

[16] A. Giorgilli and M. Sansottera, "Methods of algebraic manipulation in perturbation theory," *Workshop Series of the Asociacion Argentina de Astronomia*, Vol. 3, Jan. 2011, pp. 147–183, 10.48550/arXiv.1303.7398.

[17] À. Jorba, "A Methodology for the Numerical Computation of Normal Forms, Centre Manifolds and First Integrals of Hamiltonian Systems," *Experimental Mathematics*, Vol. 8, No. 2, 1999, pp. 155–195, 10.1080/10586458.1999.10504397.

[18] S. Boone and J. McMahon, "Directional State Transition Tensors for Capturing Dominant Nonlinear Effects in Orbital Dynamics," *Journal of Guidance, Control, and Dynamics*, Vol. 46, March 2023, pp. 431–442, 10.2514/1.G006910.

[19] E. L. Jenson and D. J. Scheeres, "Semianalytical Measures of Nonlinearity Based on Tensor Eigenpairs," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 4, 2023, pp. 638–653, 10.2514/1.G006760.

[20] P. D. Lizia, R. Armellin, and M. Lavagna, "Application of high order expansions of two-point boundary value problems to astrodynamics," *Celestial Mechanics and Dynamical Astronomy*, Vol. 102, No. 4, 2008, pp. 355–375, 10.1007/s10569-008-9170-5.

[21] G. Bucchioni, G. Gemignani, F. Lombardi, A. Bellome, J. P. F. Leitão, S. Lizy-Destrez, and M. Innocenti, "Optimal time-fixed impulsive non-Keplerian orbit to orbit transfer algorithm based on primer vector theory," *Communications in Nonlinear Science and Numerical Simulation*, Vol. 124, September 2023, 10.1016/j.cnsns.2023.107307.

[22] G. Grossi, C. Buonagura, C. Giordano, and F. Topputo, "On optimal three-impulse Earth–Moon transfers in a four-body model," *Celestial Mechanics and Dynamical Astronomy*, Vol. 136, No. 3, 2024, p. 22, 10.1007/s10569-024-10193-4.

[23] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, *Dynamical Systems, The Three-Body Problem, and Space Mission Design*. New York: Springer, 2017.

APPENDIX A: CIRCULAR RESTRICTED THREE-BODY PROBLEM

For more information, see e.g. Ref. 23. The CR3BP is normalized via the parameters m^*, l^*, t^*:

$$m^* = \tilde{M}_1 + \tilde{M}_2 \tag{36}$$

$$l^* = \tilde{R}_1 + \tilde{R}_2 \tag{37}$$

$$t^* = \left(\frac{(l^*)^3}{\tilde{G} m^*} \right)^{1/2} \tag{38}$$

where $\tilde{G}$ is the gravitational constant, $\tilde{M}_1$ and $\tilde{M}_2$ are the masses of bodies 1 and 2, and $\tilde{R}_1$ and $\tilde{R}_2$ their distances from the barycenter. Normalization of dimensional quantities $\tilde{(\cdot)}$ and time t is as below, along with the dimensionless mass ration μ:

$$X = \frac{\tilde{X}}{l^*},\ Y = \frac{\tilde{Y}}{l^*},\ Z = \frac{\tilde{Z}}{l^*},\ \tau = \frac{t}{t^*} \tag{39}$$

The normalized rotating frame equations of motion for the CR3BP are provided below:

$$\ddot{x} = 2\dot{y} + x - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3} \tag{40a}$$

$$\ddot{y} = -2\dot{x} + y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} \tag{40b}$$

$$\ddot{z} = -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3} \tag{40c}$$

$$r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2} \tag{41a}$$

$$r_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2} \tag{41b}$$

where $\dot{(\,)} = \frac{\mathrm{d}}{\mathrm{d}\tau}(\,)$, the plane $z = 0$ defines the plane in which the primary bodies orbit, and x is the direction from the larger to smaller primary. The CR3BP conserves the Jacobi constant:

$$C_J = x^2 + y^2 + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} - \dot{x}^2 - \dot{y}^2 - \dot{z}^2 \tag{42}$$