



RESEARCH ARTICLE

Trajectory design via unsupervised probabilistic learning on optimal manifolds

Cosmin Safta^{1,*} , Roger G. Ghanem² , Michael J. Grant³, Michael Sparapany³ and Habib N. Najm¹

¹Sandia National Laboratories, Livermore, California 94551, USA

²University of Southern California, Los Angeles, California 90089, USA

³Sandia National Laboratories, Albuquerque, New Mexico 87185, USA

*Corresponding author. E-mail: csafta@sandia.gov

Received: 05 January 2022; **Revised:** 12 July 2022; **Accepted:** 14 July 2022

Keywords: Diffusion map; planetary reentry trajectory; probabilistic learning on manifolds; trajectory optimization; unsupervised learning

Abstract

This article illustrates the use of unsupervised probabilistic learning techniques for the analysis of planetary reentry trajectories. A three-degree-of-freedom model was employed to generate optimal trajectories that comprise the training datasets. The algorithm first extracts the intrinsic structure in the data via a diffusion map approach. We find that data resides on manifolds of much lower dimensionality compared to the high-dimensional state space that describes each trajectory. Using the diffusion coordinates on the graph of training samples, the probabilistic framework subsequently augments the original data with samples that are statistically consistent with the original set. The augmented samples are then used to construct conditional statistics that are ultimately assembled in a path planning algorithm. In this framework, the controls are determined stage by stage during the flight to adapt to changing mission objectives in real-time.

Impact Statement

This article demonstrates the use of unsupervised learning techniques to extract low-dimensional manifolds given limited data in high-dimensional configurations. This information is used to estimate conditional statistics that are subsequently employed in designing trajectories under uncertainty without the need for additional model evaluations or data collection campaigns.

1. Introduction

Real-time trajectory optimization for hypersonic vehicles is a difficult task that requires simultaneous accounting for constraints related to flight dynamics, vehicle limitations during flight, variable initial and terminal conditions, and a high-dimensional parameter set for the models employed for these systems. Existing approaches to planetary reentry trajectory optimization problems can be generalized into two categories: indirect methods and direct methods (Betts, 1998). Indirect methods are based on Pontryagin's minimum principle, and optimal control is determined by minimizing a Hamiltonian system with respect to the control variables. These methods can result in high-fidelity solutions through adaptive refinement techniques. Nevertheless, because of high-dimensionality and sensitivity to the initial guess, the resulting boundary-value problems are quite challenging to solve (La Mantia and Casalino, 2006). Direct methods

discretize trajectories into multiple segments characterized by state and control variables. The optimal control problem (OCP) is converted into a parameter optimization problem (Fahroo and Ross, 2002), which is typically solved via nonlinear programming (Betts, 2010) or convex optimization methods (Wang and Grant, 2017, 2018). However, the computational expense for direct methods cannot be estimated a priori, and solution convergence cannot always be guaranteed for hypersonic problems. Despite recent improvements in the efficiency of both direct and indirect methods, their computational expense is high, and convergence challenges limit their adoption for onboard trajectory generation.

Given numerical challenges and computational cost, recent advances in flight dynamics planning algorithms have largely focused on the identification of single trajectory solutions. Nevertheless, during the design process, the envelope of solutions corresponding to a wide range of trajectory constraints is often required. While the computational cost can be afforded during off-line design and planning activities, this approach becomes infeasible when data needs to be processed in real-time, often with limited access to large computing capabilities.

Deep learning techniques have been recently successful in a wide variety of control problems across several research areas including aerospace, in particular for path planning of unmanned aerial systems (Choi and Ahn, 2020; Yan et al., 2020) and agile flight guidance (Loquercio et al., 2020). Deep learning has also found applications in space mission planning. Deep neural networks (DNNs) are trained on optimal state and control vectors that come from the numerical solution of an equivalent OCP. The DNN learns a map from the state vector (e.g., position and velocity) to the corresponding optimal control (e.g., the angle of attack and bank angle), by leveraging the training data provided by the OCP solver. This approach reduces the problem to a supervised learning task provided that a sufficiently large data set of optimal trajectories is available for the problem at hand. Typical applications include the approximation of optimal state-feedback control laws for interplanetary transfers (Izzo et al., 2019) and planetary soft-landing maneuvers (Sánchez-Sánchez and Izzo, 2018), as well as the real-time onboard generation of a high number of optimal trajectories for either asteroid landing (Cheng et al., 2020) or atmospheric reentry of hypersonic vehicles (Shi and Wang, 2020, 2021).

Federici et al. (2021) explored behavioral cloning and reinforcement learning algorithms for real-time optimal spacecraft guidance in presence of both operational constraints and stochastic effects, such as an inaccurate knowledge of the initial spacecraft state and the presence of random in-flight disturbances. The performance of these models is assessed on a linear multiimpulsive rendezvous mission. Zheng and Tsiotras (2021) employed DNNs to learn the optimal feedback control law for online control prediction and generating near-optimal trajectories. Based on the observation that the optimal feedback control law for the finite-time control problem is nonstationary and also may be discontinuous, this work uses the time label as an additional state of the dataset and introduces a clustering approach to sort the training data. Clustering divides the offline trajectories into groups, and a separate DNN model is trained for each group. This approach generates near-optimal trajectories that steer a system from any initial state inside a specific group based on the DNN consistent with the corresponding training data. The training data generation and DNN training are done offline, thus the online computation is minimized. The algorithm has been tested on several systems including a vehicle entry model. In all studies referenced above, the DNN typically requires $O(10^4 - 10^5)$ or more samples to train.

Deep learning frameworks typically require a large number of training samples. This can become a burden depending on the computational complexity of the trajectory model. Instead we will focus on unsupervised learning that aim to assimilate information from a limited number of samples. These class of methods is particularly efficient in high-dimensional settings when the target data describes physical systems that encode correlations and dependencies between the system components. Specifically, we propose to use an unsupervised probabilistic learning framework based on diffusion map (Coifman and Lafon, 2006; Soize and Ghanem, 2020) to assimilate the solution space of flight dynamics model inputs and outputs to (a) identify underlying low-dimensional manifolds, and (b) provide a stochastic differential equation model that can efficiently generate many sample trajectories on these manifolds that are probabilistically consistent with the training data. The diffusion map (DMAP) algorithm assimilates computed samples adaptively until the basis sets describing the low-dimensional manifolds converge for a

given set of trajectory constraints, thus utilizing available computational resources judiciously. Then, in this joint and low-dimensional space, the algorithm generates solution paths with limited computational requirements. We introduce a sequential path planning algorithm that relies on conditional statistics computed on the manifold to generate trajectories that adapt to changing conditions without the need for additional expensive flight dynamics simulations. These trajectories are equipped with uncertainty ranges that are consistent with the amount of data.

This article is organized as follows. Section 2 presents the modeling framework for this work, including a three-degree-of-freedom (3DOF) trajectory model and the optimal control algorithm. Section 3 presents the unsupervised probabilistic learning approach, followed by the results in Section 4. We end with conclusions in Section 5 and an appendix presenting the continuation schedule for the OCP.

2. Modeling Framework

2.1. Trajectory model

We consider a 3DOF model (Busemann et al., 1976) to describe the reentry trajectory of a hypersonic vehicle assumed as a point of mass inside a planetary atmosphere. Further, we assume a spherical planet model with the distance from the planet center to the vehicle location given by $r = r_e + h$, where r_e is the planet radius and h the altitude from the planet surface to the vehicle position. The three kinematic equations for the vehicle altitude h , longitude θ , and latitude ϕ are given by

$$\frac{dh}{dt} = v \sin(\gamma), \quad \frac{d\theta}{dt} = v \frac{\cos(\gamma) \cos(\psi)}{r \cos(\phi)}, \quad \frac{d\phi}{dt} = v \frac{\cos(\gamma) \sin(\psi)}{r}. \quad (1)$$

In the results presented in this article, we will use the longitude/latitude coordinates interchangeably with downrange/crossrange coordinates by conversion from angles to spatial coordinates projected onto the planet surface. The vehicle velocity vector \vec{V} relative to the planet is expressed in terms of its magnitude v and two angles: the flight path angle γ between the velocity vector and the local horizontal plane and heading angle ψ between the projection of \vec{V} on the horizontal plane and the local latitude parallel. The force equations for these components are given by

$$\begin{aligned} \frac{dv}{dt} &= \frac{1}{m} F_T - \frac{\mu \sin(\gamma)}{r^2}, & \frac{d\gamma}{dt} &= \frac{F_N \cos(\sigma)}{m v} - \frac{\mu}{r^2 v} \cos(\gamma) + \frac{v}{r} \cos(\gamma) \\ \frac{d\psi}{dt} &= \frac{F_N \sin(\sigma)}{m v \cos(\gamma)} - \frac{v}{r} \cos(\gamma) \cos(\psi) \tan(\phi). \end{aligned} \quad (2)$$

Here m is the mass of the vehicle, $\mu = 3.986 \times 10^{14} \text{ m}^3/\text{s}^2$ is the gravitational parameter, and (F_T, F_N) are the components of the aerodynamic and propulsive forces along and perpendicular to the velocity vector. This work pertains to nonthrusting flights resulting in $F_T = -D$ and $F_N = L$, where D and L are the aerodynamic drag and lift forces, respectively. The bank angle σ in equation (2) accounts for the angle between the direction of the lift force L and the (\vec{r}, \vec{V}) plane formed by the vector from the center of the planet to the vehicle location and the velocity vector.

For the remainder of this article, the location and velocity components are grouped into a state vector denoted by $\mathbf{x} = \{h, \theta, \phi, v, \gamma, \psi\}$. The angle of attack and the bank angle are grouped into the control vector denoted by $\mathbf{u} = \{\alpha, \sigma\}$, and the 3DOF model can be written as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ where \mathbf{f} is defined by the set of right-hand sides of equations (1) and (2).

2.2. Vehicle model

For this study the vehicle model is a blunt cone with mass $m = 350 \text{ kg}$, reference area $A_{\text{ref}} = \pi \times 0.305^2 \text{ m}^2$, and nose radius $r_n = 0.0254 \text{ m}$ (Sparapan and Grant, 2020). The lift and drag coefficients are given by

$$C_l(\alpha) = c_0 \alpha \quad C_d(\alpha) = c_1 \alpha^2 + c_2$$

respectively, where α is the angle of attack (in radians). For the lift coefficient, the slope is set to $c_0 = 1.5658$. For the drag coefficient, the model parameters are given by $c_1 = 1.6537$ and $c_2 = 0.0612$.

The lift and drag forces are functions of the angle of attack, altitude, and velocity magnitude, and are computed as

$$L = 0.5\rho(h)v^2C_l(\alpha)A_{\text{ref}}, \quad D = 0.5\rho(h)v^2C_d(\alpha)A_{\text{ref}}$$

where A_{ref} , defined above, is assumed independent of the angle of attack. The atmospheric density is approximated with an exponential dependence on altitude as

$$\rho(h) = \rho_0 \exp(-h/H)$$

where $\rho_0 = 1.231 \text{ kg/m}^3$ is the atmospheric density at $h = 0$ and H is a scale constant. For this study we have generated data using a range of values $H \in [7000, 8000]\text{m}$ to simulate the uncertainty in the atmospheric density.

2.3. Optimal control problem

We construct a variational problem to generate trajectories based on the models presented in Sections 2.1 and 2.2. For the dependent state vector \mathbf{x} , the variational problem seeks a time-dependent angle-of-attack α and bank angle σ that maximize the magnitude of the terminal velocity v_T at the desired endpoint given by $\mathbf{G}(\mathbf{x}(t_f), t_f) : \mathbb{R}^6 \times \mathbb{R}^+ \mapsto \mathbb{R}^3$

$$\mathbf{G}(\mathbf{x}(t), t) = \begin{bmatrix} h(t) - h_f \\ \theta(t) - \theta_f \\ \phi(t) - \phi_f \end{bmatrix}, \tag{3}$$

while satisfying the 3DOF model $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. Here, (h_f, θ_f, ϕ_f) are the desired terminal values the altitude, longitude, and latitude, respectively.

The initial location of the vehicle is fixed with the initial constraint function $\mathbf{F} : \mathbb{R}^6 \times \mathbb{R}^+ \mapsto \mathbb{R}^6$. \mathbf{F} is similarly defined for the initial values of the state vector components $\mathbf{x}_0 = \mathbf{x}(t_0) = (h_0, \theta_0, \phi_0, v_0, \gamma_0, \psi_0)$ such that $\mathbf{F}(\mathbf{x}(t), t) - \mathbf{x}_0 = \mathbf{0}$. Finally, $\mathbf{U} : \mathbb{R}^6 \mapsto \mathbb{R}^l$ represents l state path-constraints. The free final-time variational problem is posed as

$$\begin{aligned} \min_{\mathbf{u}(t)} J &= \int_{t_0}^{t_f} \mathcal{L} dt - v_f^2 \\ \text{Subject to: } \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{F}(\mathbf{x}(t_0), t_0) &= \mathbf{0} \\ \mathbf{G}(\mathbf{x}(t_f), t_f) &= \mathbf{0} \\ U_{i,\text{MIN}} \leq U_i(\mathbf{x}) &\leq U_{i,\text{MAX}}, \quad i = 1, \dots, l. \end{aligned} \tag{4}$$

In the objective function above, we included an integrand term, \mathcal{L} , that is identically 0 to be consistent with the Lagrange multiplier integral described below. We employ the *beluga* framework (Sparapany and Grant, 2020) to solve the variational problem posed in equation (4) using indirect methods (Longuski et al., 2014). This process is succinctly described here for completeness and in more detail in Sparapany (2020). First, the state path-constraints, \mathbf{U} , are treated using trigonometrization (Mall et al., 2020). This involves adjoining each path-constraint to the path-cost function with small error parameters δ ,

$$\tilde{\mathcal{L}} = \mathcal{L} + \sum_{i=1}^l \delta_i \left(\sec \left(\frac{\pi}{2} \frac{2U_i(\mathbf{x}) - U_{i,\text{MAX}} - U_{i,\text{MIN}}}{U_{i,\text{MAX}} - U_{i,\text{MIN}}} \right) - 1 \right), \tag{5}$$

where \mathcal{L} was chosen to be $\mathcal{L} = 0$ from equation (4). Next, the initial, terminal, and dynamic path constraints are adjoined to the cost functional with Lagrange multipliers ξ_0 , ξ_f , and λ ,

$$\min_{\mathbf{u}(t)} J^* = \int_{t_0}^{t_f} (\tilde{\mathcal{L}} + \lambda^T (\mathbf{f} - \dot{\mathbf{x}})) dt + \xi_0^T \mathbf{F}_0 + \xi_f^T \mathbf{G}_f - v_f^2, \tag{6}$$

where $\mathbf{F}_0 := \mathbf{F}(\mathbf{x}(t_0), t_0)$ and $\mathbf{G}_f := \mathbf{G}(\mathbf{x}(t_f), t_f)$.

Next, we introduce the Hamiltonian, $H \equiv \tilde{\mathcal{L}} + \lambda^T \mathbf{f}$, and define the Euler–Lagrange operator \mathfrak{E} as

$$\mathfrak{E} = \frac{\partial}{\partial \mathbf{y}} - \frac{d}{dt} \frac{\partial}{\partial \dot{\mathbf{y}}}, \tag{7}$$

where $\mathbf{y} = (\mathbf{x}, \mathbf{u}, \lambda)$. Application of the Euler–Lagrange operator and integrating by parts solves the original variational problem. The result is an analytical formulation in the form of a two-point Hamiltonian boundary value problem (HBVP),

$$\begin{aligned} \dot{\mathbf{x}} &= \frac{\partial H}{\partial \lambda}, & \dot{\lambda} &= -\frac{\partial H}{\partial \mathbf{x}}, & 0 &= \frac{\partial H}{\partial \mathbf{u}} \\ \lambda(t_0) &= -\xi_0^T \frac{\partial \mathbf{F}}{\partial \mathbf{x}}, & \lambda(t_f) &= \xi_f^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}} - 2v_f. \\ \mathbf{H}(t_f) &= 0, & \mathbf{F}_0 &= 0, & \mathbf{G}_f &= 0 \end{aligned} \tag{8}$$

The first three terms in equation (8) define the equations-of-motion in a Hamiltonian dynamical system while the latter five terms define values at the boundaries. From this, approximate solutions to equation (4) may be found *indirectly* by numerically solving equation (8). In *beluga* the right-hand sides in equation (8) are constructed using symbolic manipulation with *SymPy* (Meurer et al., 2017). For more details regarding the numerical solution of this system see (Kierzenka and Shampine, 2001; Sparapan, 2020).

3. Probilistic Learning on Manifold

Probabilistic learning on manifolds (PLoM) is a recently developed unsupervised learning technique for augmenting small datasets in a principled manner (Soize and Ghanem, 2016, 2020). The method views a training set as a graph with vertices in feature space, and generates replicas of this graph that are consistent, both structurally and statistically with the training graph. An intrinsic structure is first extracted from the training dataset and is subsequently used to constrain statistical sample generation. This intrinsic structure takes the form of a subspace spanned by the so-called diffusion coordinates associated with the eigenvectors of the graph Laplacian of the training dataset. The sample generation is in the form of a projected Itô equation constrained to the span of these diffusion coordinates. Each additional generated sample is a statistical replica of the training data, restricted to the same dominant diffusion coordinates. In this section, we summarize this construction with the requisite technical details for a self-contained assessment of the article. A more complete presentation can be found elsewhere (Soize and Ghanem, 2016).

We construe the initial, training, dataset \mathbf{x}^d , with N samples and n features, as a realization of a $n \times N$ matrix-valued random variable \mathbf{X} with n rows and N columns. Here, we update our notation for x , extending its meaning from denoting one state along the trajectory to denoting the collection of states that numerically define the entire trajectory. Our objective is to generate a new dataset \mathbf{x}^d , the augmented dataset, as realizations of \mathbf{X} . This new dataset is probabilistically consistent with the original data, it can be used to augment it for subsequent statistical tasks, including the estimation of marginal and conditional density functions (Ghanem and Soize, 2018) and for nonparametric regression. The first step is to decorrelate the features of \mathbf{X} , through a linear transformation $\Theta = \mu^{-1/2} \Phi^T (\mathbf{X} - \bar{\mathbf{x}})$ where μ is a diagonal matrix with the dominant v eigenvalues of the $n \times n$ covariance matrix of \mathbf{X} and Φ is a $n \times v$ matrix of the associated eigenvectors. Also, $\bar{\mathbf{x}}$ denotes the $n \times N$ matrix with duplicate columns that are each equal to the average of the features over the N samples. This initial decorrelation step is a straightforward application of the standard PCA procedure to the dataset \mathbf{x}^d .

We will denote samples of \mathbf{X} by \mathbf{x} and samples of Θ by $\boldsymbol{\eta}$. The sample of Θ associated with \mathbf{x}^d will be denoted by $\boldsymbol{\eta}^d$. Next we extract and describe an intrinsic structure from $\boldsymbol{\eta}^d$. The first step is to select a

diffusion kernel $(k_\varepsilon(\boldsymbol{\eta}, \boldsymbol{\eta}'; \varepsilon) : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R})$ that will be used to define proximity on the graph in \mathbb{R}^n with N vertices. Here, ε is a parameter of the kernel, typically characterizing its bandwidth. We next construct an $N \times N$ diffusion matrix \mathbf{K} , from the training dataset, such that $\mathbf{K}_{ij} = k(\boldsymbol{\eta}_i^d, \boldsymbol{\eta}_j^d)$, and we normalize \mathbf{K} so that the sum over each of its rows is equal to 1. The resulting stochastic matrix, \mathbf{P} , can thus serve as a transition matrix for a Markov chain on the graph. Its eigenvectors are denoted by $\boldsymbol{\psi}^\alpha$, $\alpha = 1, \dots, N$. Given its construction, the largest eigenvalue of \mathbf{P} is equal to 1 and the associated eigenvector is a constant. The $N \times m$ matrix consisting of the m eigenvectors associated with the next largest m eigenvalues (excluding the largest unit eigenvalue) is denoted by \mathbf{g} . It should be noted that while matrix \mathbf{P} is nonsymmetric, its eigenvalues and eigenvectors can be shown to be real, and can be evaluated from an associated symmetric matrix (Soize and Ghanem, 2016). The eigenspectrum of \mathbf{P} typically exhibits a sharp, almost discontinuous, decrease after the first few eigenvalues. We assign the index of the eigenvalue corresponding to this drop to the numerical value of m .

With the above procedure, we have now constructed a basis set, \mathbf{g} , called the diffusion coordinates, that localizes $\boldsymbol{\eta}^d$ to an m -dimensional subset in \mathbb{R}^N . We emphasize that this localization is not in \mathbb{R}^n , which is what the PCA usually accomplishes.

The next step in our procedure is to describe an initial representation of the joint probability distribution of the random matrix \mathbf{H} . This is accomplished in two steps. First, the N samples of the n features are used to estimate the joint PDF of these features using an n -dimensional Gaussian mixture model, and kernel density estimation (KDE). Then the joint density of the whole graph is constructed with the assumption of independence among its N vertices. The joint PDF model, in $\mathbb{R}^{N \times v}$, for the whole graph is therefore the product of N joint pdfs, each defined in \mathbb{R}^v and represented as a KDE centered at one of the N vertices. This final representation is in the following form of the product of sums of Gaussian kernels,

$$q_{\mathbf{H}}(\boldsymbol{\eta}) = \frac{1}{N} \prod_{i=1}^N \sum_{j=1}^N \frac{1}{h} k(\boldsymbol{\eta}^{d,j}, \boldsymbol{\eta}^i; \zeta), \tag{9}$$

where $k(\boldsymbol{\eta}, \boldsymbol{\eta}'; \zeta)$ is a Gaussian kernel on \mathbb{R}^n with bandwidth ζ , $\boldsymbol{\eta}^{d,j}$ is the value of $\boldsymbol{\eta}^d$ at the j th vertex, and $\boldsymbol{\eta}^i$ is the value of $\boldsymbol{\eta}$ at the i th vertex. Criteria have been developed (Soize and Ghanem, 2016) for selecting the bandwidth ζ so as to propagate the normalization and orthogonality conditions inherited from the PCA step described previously.

The third and final step in the PLoM procedure is to construct a generator of samples that are constrained by the diffusion coordinates while being informed by the KDE, both of which are synthesized directly from the data. To that end, we start with a Hamiltonian form of the Itô equation whose invariant measure is defined by the above KDE with respect to the Lebesgue measure. The Hamiltonian form of this equation allows to develop an efficient symplectic integration scheme (Soize and Ghanem, 2016). A reduced-order Itô stochastic differential equation (ISDE) corresponding to a change of variables involving the diffusion map eigenvectors \mathbf{g} is derived. The ISDE for $\zeta \in \mathbb{R}^+$ is written as

$$\begin{aligned} d\mathbf{Z}(\zeta) &= \mathbf{Y}(\zeta) d\zeta \\ d\mathbf{Y}(\zeta) &= L(\mathbf{Z}(\zeta)) d\zeta - \frac{1}{2} f_0 \mathbf{Y}(\zeta) d\zeta + \sqrt{f_0} d\mathbf{W}(\zeta), \\ \mathbf{Z}(0) &= \Theta_d \mathbf{a} \quad \mathbf{Y}(0) = \mathbf{N} \mathbf{a} \quad \mathbf{a} = \mathbf{g}(\mathbf{g}^T \mathbf{g})^{-1}, \end{aligned} \tag{10}$$

where $L(\mathbf{Z}) = \nabla \log q(\mathbf{Z} \mathbf{g}^T) \mathbf{a}$ is the projected potential, \mathbf{N} is a $v \times N$ matrix whose N columns are independent copies of a standard Gaussian vector in \mathbb{R}^v , and f_0 is a damping parameter. Further, the columns of \mathbf{W} are N independent copies of a normalized Wiener process projected on the matrix \mathbf{a} .

Similar to (Soize and Ghanem, 2016), here we employ a Störmer–Verlet scheme to integrate equation (10). After a brief nonstationary period in the Itô dynamics, samples of Θ are reconstructed from samples of \mathbf{Z} as

$$\boldsymbol{\eta}^\ell = \mathbf{Z}^\ell \mathbf{g}^T \quad \ell = 1, \dots, n_{\text{MC}}. \tag{11}$$

Realizations of the original random variable X are then obtained by reversing the application of the PCA on η^e . These realizations will augment the original set of samples of X and will be used for the purpose of computing statistics (means, quantiles, and PDFs) conditioned on select observations at intermediate stages along specific trajectories.

4. Results

In this section, we will characterize the set of trajectories discussed in Section 4.1 using the algorithms presented in the previous section. We will first inspect the topology of these manifolds via their corresponding basis vectors in Section 4.2. We will then verify in Section 4.3 that the augmented set of trajectories generated via manifold sampling are consistent with the 3DOF model, and then introduce in Section 4.4 a workflow for sequential path planning using the augmented dataset to generate conditional statistics for the state and control vectors.

4.1. Training data

For this study, we consider planetary reentry trajectories generated using the model presented in Section 2.1 given a number of parameters treated as random variables. Specifically, the initial height, h_0 , velocity magnitude v_0 , longitude θ_0 , latitude ϕ_0 , were sampled from uniform distributions, with ranges presented in Table 1. The constant H present in the atmospheric density model was also treated as a uniform random variable. The initial flight path and heading angles were fixed to 0, $\gamma_0 = \psi_0 = 0$, that is, the trajectory start in a horizontal plane, along the local latitude parallel. The terminal location, $h_f = 0$, $\theta_f = 3^\circ$, $\phi_f = 2^\circ$, was also fixed in this study.

In addition to the parameters shown in Table 1, the computational model also includes a set of path constraints, resulting in trajectories that avoid circular regions centered around the two circles shown in the left frame of Figure 1. The intensity of these constraints is controlled by an additional parameter, δ . When setting $\delta = 0$, the path constraints are not activated, resulting in a set of trajectories depicted in gray in Figure 1. The optimization framework uses a numerical continuation algorithm to increase the strength of path constraint expressions, resulting in the trajectory samples shown in blue and red, respectively. These samples correspond to the same value, $\delta = 400$, and are colored according their topology: the blue samples correspond to paths that go in between the two regions while the red samples avoid these regions and stay on the left side. The appendix provides additional details related to the set of continuation stages employed to generate the trajectory dataset for this study.

For this study, we employed 1,100 samples drawn independently from the uniform distributions presented in Table 1. For each sample, we generated 41 trajectories corresponding to equally spaced δ -values, from $\delta = 0$ to $\delta = 400$.

For each parameter sample, the time-dependent values for the dependent variables (location and velocity components) and the control variables (angle of attack and bank angles) are interpolated on a uniform time grid, and the corresponding solution vectors are concatenated together with the time grid. The parameter values that control the simulation, that is, the ones corresponding to Table 1 are also appended to the solution vector, in addition to the value of path constraint parameter δ . Each sample vector

Table 1. Parameter ranges for the uniform random variables that control the trajectory dataset.

	Parameter				
	h_0 [m]	θ_0 [deg]	ϕ_0 [deg]	v_0 [m/s]	H [m]
Min	38×10^3	-0.3	-0.3	2000	7000
Max	45×10^3	0.3	0.3	2200	8000

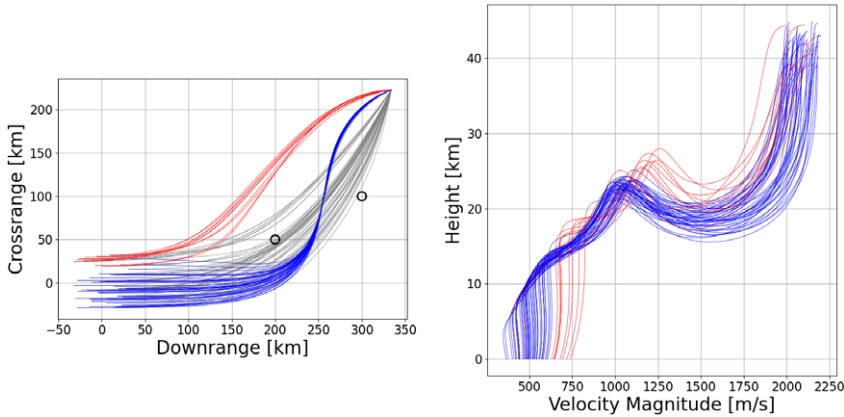


Figure 1. Sample trajectory data. The left frame shows the downrange/crossrange solution components and the right frame shows velocity/height components. The samples shown in gray represent the unconstrained components while the red/blue samples correspond to the maximum path constraint condition, $\delta = 400$.

becomes a column in the matrix of samples that will be processed via the algorithms presented in the previous section. Since all trajectories have the same initial conditions for the flight path and heading angles and the same terminal location, the matrix is rank deficient if these values are included in the solution vector. Instead we choose a time grid that starts at 1% and ends at 99% of the total duration of each trajectory. In order to preserve the information about the total duration, the total time for each trajectory is also appended to each sample vector.

4.2. Manifold construction

In this section, we illustrate the performance of the DMAP component of PLoM while assimilating the high-dimensional space that describes the 3DOF trajectories. This approach reveals a relatively small set of eigenvectors, typically 45 – 52, that are sufficient to describe the corresponding low-dimensional manifold. We also highlight the utility of this algorithm to both detect outliers that are not otherwise evident in the physical space. Upon the removal of these outliers, one can “zoom-in” and inspect relationships between samples, including proximity between samples along the diffusion manifold geodesics.

First, we inspect the impact of kernel bandwidth on the intrinsic dimensionality of the diffusion manifold and the topology of the basis vectors. For this task we select a random subset of trajectories from the dataset. This random subset includes trajectories corresponding to random choices for the initial altitude, longitude, latitude, and velocity, as well as random choices for the path constraint parameter δ . Figure 2 displays the eigenvalue spectra for several values of the kernel bandwidth ε . The results in the left frame of this figure indicate a manifold dimension $m = 52$, for a graph Laplacian using a kernel bandwidth of $\varepsilon = 1,000$. For smaller bandwidth values, for example, for $\varepsilon = 200$, the eigenvalue decay is mild indicating a less-defined structure in the high-dimensional space of trajectories. The topology of the first two basis vectors is presented in Figure 3. The first row in this figure is constructed with the same dataset as for the results presented in the left frame of Figure 2. These plots show that most of the components for the first two basis vectors are concentrated on much smaller values compared to a few select trajectory samples; one of these samples is highlighted in red in these figures as it stands out even for a relatively large bandwidth, $\varepsilon = 1,000$. It appears these samples are significantly different compared to the rest of the training set based on the magnitude of their entries in the diffusion map vectors. These are much larger in magnitude compared to the entries corresponding to rest of the samples resulting in these sample standing out while the rest of the sample cluster together.

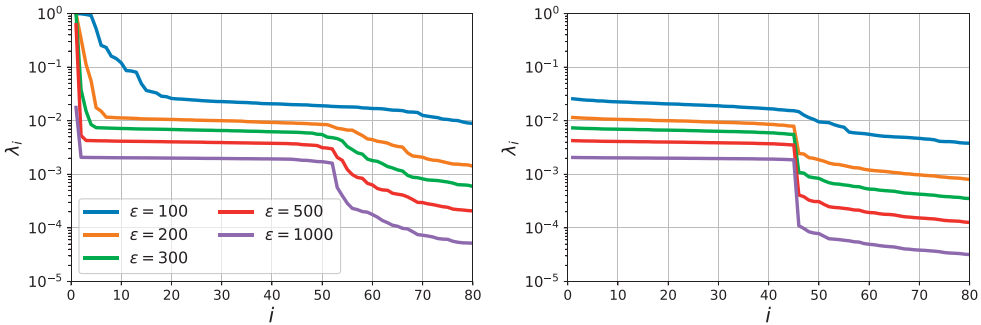


Figure 2. Eigenvalue spectra for several values of the kernel bandwidth ε : (left frame) results corresponding to the entire training set; (right frame) results obtained after several edge samples were removed from training.

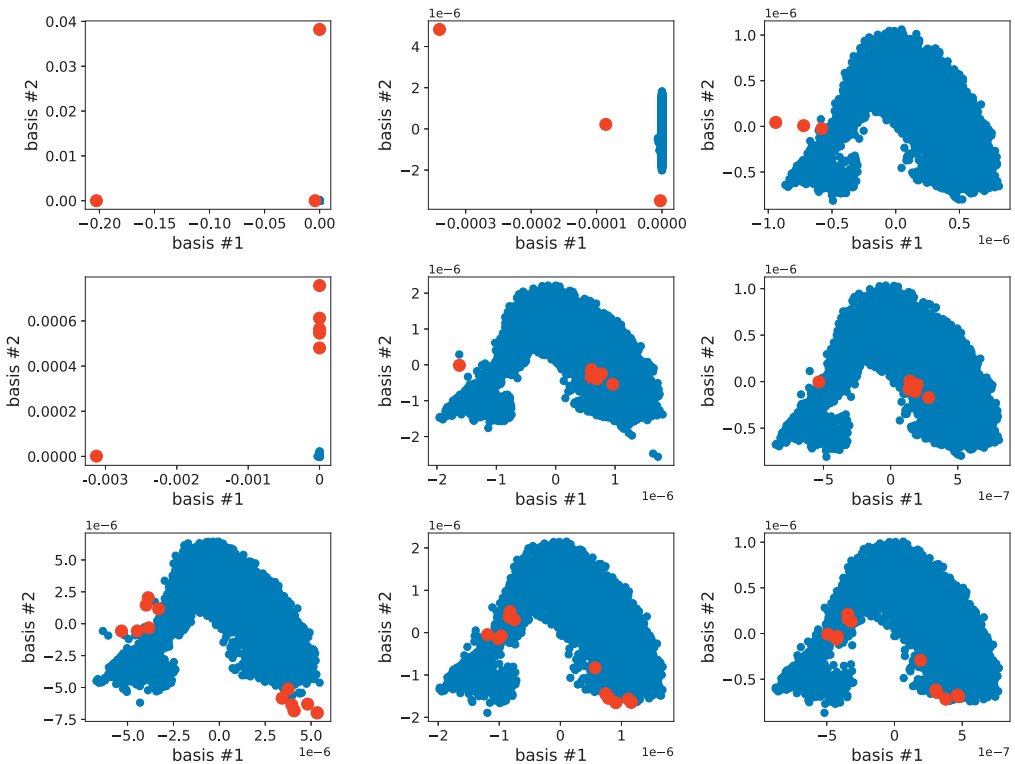


Figure 3. Scatter plots showing the entries in the two most dominant eigenvectors for all samples in blue, with superimposed red circles for edge cases. Left to right columns correspond to diffusion map results based on $\varepsilon = \{200, 500, 1000\}$, respectively. Rows show results after edge cases are sequentially removed from the datasets (top to bottom).

We proceed to examine the impact of removing these edge (outlier) cases on the diffusion manifold basis vectors. The remaining rows in Figure 3 display a sequence of results obtained after edge cases are gradually removed from the training dataset. The results on the second row correspond to a training dataset for which the sample shown in red in the first row was removed, while the results on the third row correspond to a dataset with three additional samples, highlighted in red on the second row, removed from

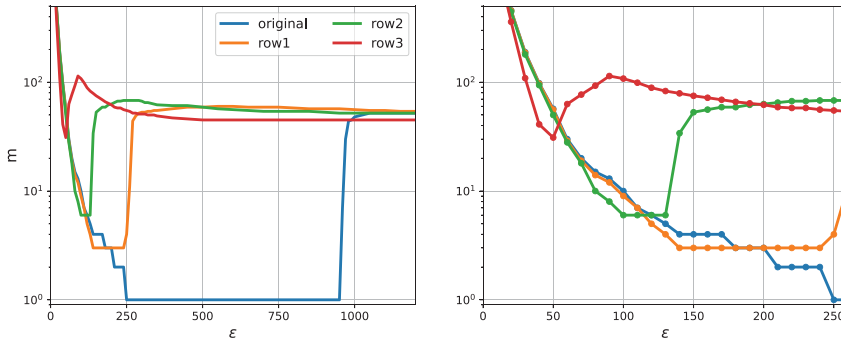


Figure 4. Manifold dimension m dependence on ε . Results labeled “row 1” through “row 3” correspond to the same sequence of datasets underlying the results shown in Figure 2, while results labeled “original” are based on the original dataset before outlier removal.

training. As more edge samples are removed from the training dataset, we are able to “zoom-in” on the manifold structure that contains the bulk of the samples. The results on the bottom row are obtained after several sets of samples are removed from the training dataset. These results correspond to the same analysis displayed in the right frame of Figure 2. The eigenvalue decay presented in this figure indicates now that a diffusion manifold can be defined by a smaller number of basis vectors, approximately 45, and the sharp transition can be obtained with a smaller bandwidth, that is, the training dataset is more compact after removing edge cases identified through a sequence of diffusion manifold basis vectors. This exercise also suggest that the bulk of the manifold structure can be represented with a relatively reduced set of basis vectors, while edge/outliers cases add to the manifold dimensionality, as additional information is needed to replicate low probability regimes.

Figure 4 shows the dependence of the manifold dimension m on the kernel bandwidth ε . For all datasets the manifold dimension exhibit a minimum value at intermediate values for ε . The magnitudes of these minima correspond approximately with the number of outliers observed in the corresponding dataset. For results corresponding to the original dataset, there is one training sample that is sufficiently different from the remaining data. This sample dominates the manifold structure up to $\varepsilon \approx 900$. Beyond this value, the kernels become sufficiently diffuse to diminish the impact of this sample, and the manifold dimension stabilizes around $m \approx 50$. Once this sample is removed from the dataset, the next layer of edge cases become dominant (see results corresponding to row 2). Since these samples are now less removed from the bulk of the samples, one can “zoom-in” on the manifold structure at smaller bandwidth values, for example, $\varepsilon \approx 250$, compared to the previous, much larger bandwidth values. The trend continues as subsequent edge cases are removed.

Figures 5 and 6 show the trajectories present in the training sets with gray lines and several sets of edge samples with thick colored lines. These trajectory samples, illustrated in physical coordinates, do not immediately stand out compared to the bulk of samples, shown in gray. It can be argued that samples displayed in magenta and cyan correspond to trajectories that take the longest to reach the terminal location. Nevertheless, these are the last sets of outliers identified in the sequence presented above. Earlier outlier sets, that are further removed in manifold coordinates, for example, the samples shown in red and blue are not clearly distinct from the bulk. This observation highlights the utility of examining the data in a diffusion map context, thus revealing high-dimensional samples that are structurally different compared to the bulk.

4.3. Augmented dataset consistency with the 3DOF model

The PLoM framework provides, once the diffusion manifolds coordinates are available, a generator of samples that are constrained on the low-dimensional manifold and are probabilistically consistent with the

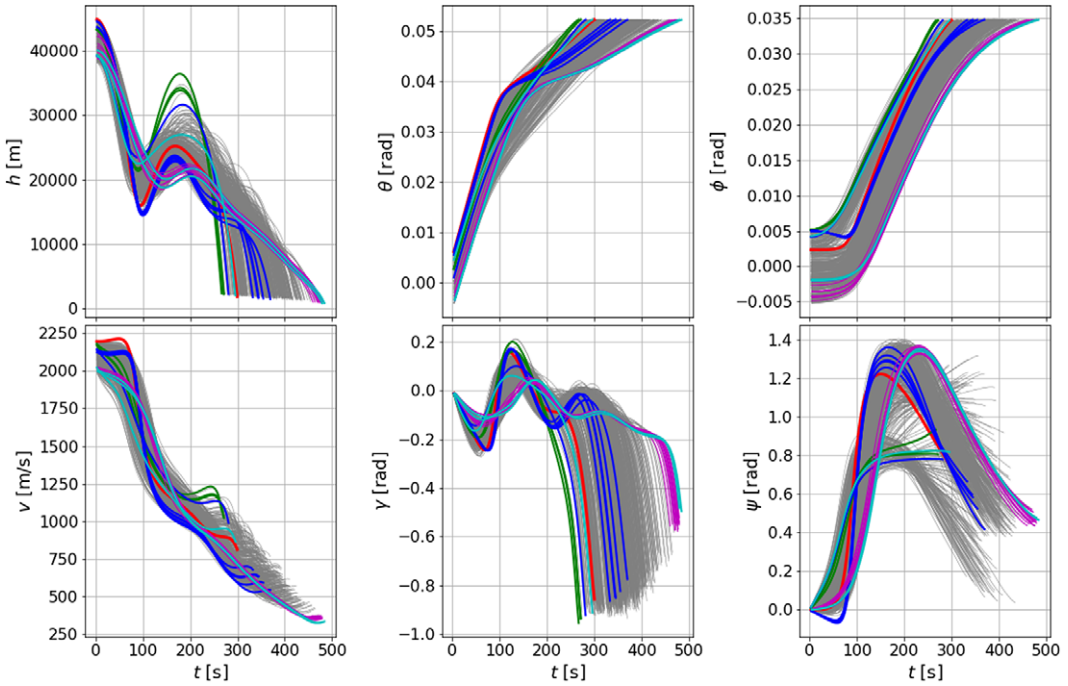


Figure 5. Trajectory data defining the samples used for learning the diffusion map representation: gray lines show a random subset of 100 samples; red/blue/green/magenta/cyan show samples removed from the learning set (in this order).

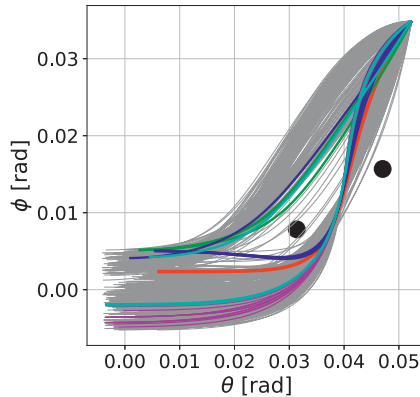


Figure 6. Same dataset and color scheme as in Figure 5, shown in longitude/latitude coordinates.

original data. In this section, we will determine whether or not the samples generated via the stochastic differential system in equation (10), are consistent with the 3DOF model used to generate optimal trajectories via the OCP, that is, the synthetic samples represent realistic planetary reentry trajectories.

We first determine the requisite size for the synthetic dataset. We find that between 10^3 and 2×10^3 replicas are necessary to construct converged marginal probability density estimates for intermediate and terminal state vectors (results not shown). All synthetic samples generated via equation (10) employed a step size $\Delta r = 0.1$ and a damping parameter $f_0 = 1$, according to Soize and Ghanem (2016).

For each synthetic trajectory in the augmented set, we numerically evaluate the time derivatives in the left-hand side of equations (1) and (2) via finite differences, then compute the discrepancy with the right-hand side

$$r_{ij} = \frac{x_{ij} - x_{i-1j}}{t_i - t_{i-1}} - \frac{1}{2} (f_j(x_{i-1}, u_{i-1}) + f_j(x_i, u_i)). \tag{12}$$

Here, subscript i represents the time index and subscript j represents the component of the state vector x . Additionally, for the residuals corresponding to the altitude h and velocity v we scale the residual by the average magnitude over each time interval

$$r_{ij} \rightarrow r_{ij} \times \frac{2}{x_{i-1j} + x_{ij}}. \tag{13}$$

Finally, for each synthetic trajectory we evaluate the L_2 -norm of the residual over all time steps and state vector components. These results are collected for all trajectory samples in the augmented dataset constructed via equation (10).

Figure 7 shows histograms for several sets of trajectories conditioned on select terminal velocity values v_T and path constraint parameter δ values. The samples used to construct the results in the left frame are conditioned on $v_T = 650$ [m/s] and $\delta = 100$, $p(\cdot | v_T = 650, \delta = 100)$. The results in the middle and right frames correspond to trajectories conditioned on the same terminal velocity but increasingly path constrained, with $\delta = 200$ and 300 , respectively. We selected several ϵ values to assess the impact the manifold-construction KDE bandwidth ultimately has on manifold-sampled trajectories. For all cases, the residual L_2 norms are $O(10^{-3})$ and of the same order of magnitude as the numerical residuals for the training dataset, shown with black histograms in this figure.

We conclude that the samples in the augmented dataset are consistent with the underlying 3DOF model and thus follow the same system dynamics as the original dataset generated via the OCP solution. This

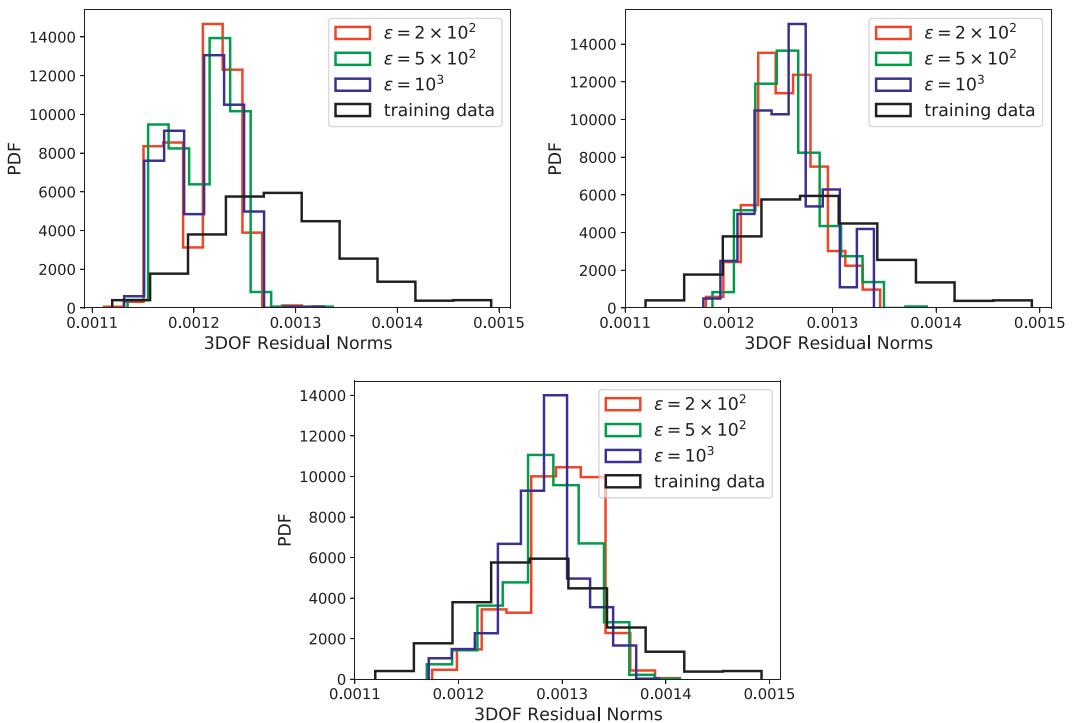


Figure 7. Distribution of 3DOF residual norms for synthetic trajectories conditioned on the terminal velocity of 650 m/s and several values for δ : 100 (top left frame), 200 (top right frame), and 300 (bottom frame).

allows us to employ these synthetically generated trajectories for path-planning exercises without the need for additional, computationally expensive, OCP simulations. We will illustrate such a path-planning algorithm in the next section.

4.4. Path planning via PLoM

We now introduce a workflow that adjusts the vehicle trajectory sequentially using the observed state vector at intermediate locations, including the option to change flight path objectives mid-flight.

We start the workflow, presented below in Algorithm 1, by defining an objective for the vehicle trajectory, obj_t . This objective consists of an ensemble of constraints, either deterministic or probabilistic. In this section, we will display several examples, using either the same set of constraints throughout the flight or a set of constraints that adjusts during the flight. It should be noted that these objectives should not be confused with the cost function defined in Section 2.3. Rather, we would like to select from the manifold of optimal solutions, constructed as described in Section 2.3, the set of samples that satisfy additional constraints without having to recompute the OCP. These constraints restrict the range of admissible solutions to a subset on the manifold previously computed. For the examples presented below, obj_t corresponds to reduced ranges for the path constraint parameter δ and limits on the terminal velocity v_T . Once the initial objective $\text{obj}_{t=0}$ is defined, we estimate the trajectory that satisfies this objective as a *conditional expectation* solution *constrained* on the manifold of optimal trajectories. This yields the set of state vectors $x^{(e)}$ and controls $u^{(e)}$ required to realize this trajectory. We then enter a control loop corresponding to rows 3–8, during which the control u is adjusted over a series of stages to correct for errors in the vehicle path, as well as to adjust the controls in case the set of constraints imposed on the workflow changes during the flight. In the workflow below, the time step Δt is represented as a fraction of the trajectory duration. Inside the control loop, we first proceed with estimating the trajectory over a specific stage $[t, t + \Delta t]$ (row 4). It is possible that, due to model errors or incomplete knowledge of the environmental conditions, the expected state vector at $t + \Delta t$, $x_{t+\Delta t}^{(e)}$, will not match the actual (measured) solution, $x_{t+\Delta t}^{(m)}$. If this discrepancy is larger than a lower bound threshold $\epsilon_{\Delta t}$, or if the overall objective has to be adjusted, that is, $\text{obj}_{t+\Delta t} \neq \text{obj}_t$, we proceed to update the set of controls for the next flight segment on line 7.

Algorithm 1: Sequential path-planning algorithm using conditional sampling on manifolds.

```

1 select initial objective, e.g.  $\text{obj}_{t=0}$ ;
2 Compute expected trajectory  $\mathbb{E}[x, u | \text{obj}_{t=0}] \rightarrow (x^{(e)}, u^{(e)})$ ;
3 while  $t < t_{end}$  do
4     advance from  $t$  to  $t + \Delta t$  using the expected control  $u_{[t, t+\Delta t]}^{(e)}$ ;
5     retrieve position at  $t + \Delta t$ :  $x_{t+\Delta t}^{(m)}$ ;
6     if  $\|x_{t+\Delta t}^{(e)} - x_{t+\Delta t}^{(m)}\| > \epsilon_{\Delta t} \vee (\text{obj}_{t+\Delta t} \neq \text{obj}_t)$  then
7         re-compute control  $\mathbb{E}[u | \text{obj}_{t+\Delta t}, x_{t+\Delta t}^{(m)}] \rightarrow u_{t+\Delta t, t_{end}}^{(e)}$ ;
8     end
9      $t \rightarrow t + \Delta t$ ;
10 end

```

Figure 8 displays the conditional marginal PDFs for the vehicle location, $p((\theta, \phi)_{t+\Delta t} | x_t^{(m)})$ (left frame), $p((h, \theta)_{t+\Delta t} | x_t^{(m)})$ (middle frame), and $p((h, \phi)_{t+\Delta t} | x_t^{(m)})$ (right frame). The sequence of stages in this figure are constructed using Algorithm 1. These results are based on the same initial condition for the vehicle height, velocity (2,000 m/s), and longitude, and density model constant. The initial latitude values for these simulations were set as indicated in the figure.

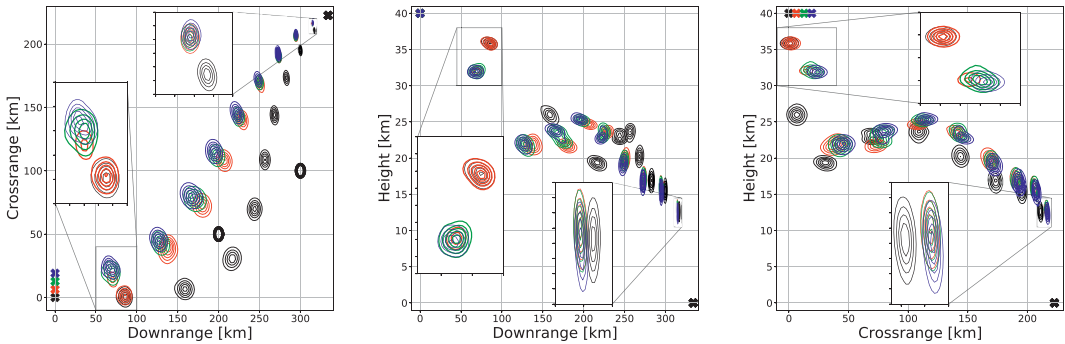


Figure 8. 2D PDFs for vehicle location at intermediate locations for trajectories that originate at $h_0 = 40$ km, $\theta_0 = 0$ rad, and $\phi_0 = \{0(\text{black}), 0.001(\text{red}), 0.002(\text{green}), 0.003(\text{blue})\}$ rad. The initial latitude values correspond to $\{0, 6, 12, 18\}$ [km] in the crossrange coordinates. The “x” symbols mark the start and end points and the large circles mark the location of the exclusion regions.

At each intermediate stage t , we compute the statistics for the set of trajectories at $t + \Delta t$ by conditioning on the current state vector $x_t^{(m)}$ at time t . For this example we adopt, and condition on, a time varying model for the density model constant $H = H(t)$ to demonstrate our approach for cases where the atmospheric conditions change during the flight. For the results displayed in this figure, we assume that $H(t) = (7, 700 - 300t)$ m, where time t was normalized by the trajectory duration, that is, $t \in [0, 1]$. The range of uncertainties that arise from one stage to the next are due to the range of training data used for this demonstration. Here, we consider results corresponding to $\text{obj}_t \equiv \{50 < \delta \leq 400\}$. The range of path constraint values results in the statistics shown in the figure below.

These results illustrate conditional statistics constrained on a manifold corresponding to a multimodal behavior. The multimodality here is induced by set of trajectories that avoid the two regions shown by circles in Figure 8. These exclusion regions partition the training data into two subsets, with one set of trajectories passing in between the two regions and the other avoiding the two regions on the left. Depending on the start of each trajectory assembled via Algorithm 1, the conditional densities for the vehicle location evolve on either of these paths. For the run shown in black, the sequence of intermediate locations point to an ensemble of paths going in between the two circles. For the simulation shown in red, the location at the 10% mark (the first set of contours near the start points, details also shown in the inset frames) displays a bimodal behavior. At this time stamp the trajectory finds itself in the mode that is near the set of results for the simulations shown in blue and green. These sets of results evolve together and as trajectories funnel toward the terminal location. The second inset in all frames focuses on the marginal densities at the 90% mark.

We will further illustrate this workflow with several numerical examples that employ the manifold statistics constructed with trajectory datasets presented in previous sections. Table 2 lists choices for several algorithm knobs chosen for these runs. All runs employ conditioning with $v_T > v_{T,\min} = 650$ m/s. Runs 1 and 2 employ the same objective throughout the entire trajectory, with $\delta > 50$. For Runs 3 and 4, the flight

Table 2. Numerical settings for the set of runs chosen to illustrate the workflow in Algorithm 1.

Run ID	$\delta_{\min,0-30}$	$\delta_{\min,30-60}$	$\delta_{\min,60-100}$	Altitude bias
1	50	50	50	+
2	50	50	50	-
3	50	200	300	+
4	50	200	300	-

Note. For all runs $v_{T,\min} = 650$ m/s.

path constraint δ is adjusted after 30 and 60% of the entire trajectory duration, respectively. All runs share the same initial condition as the simulation shown in black in Figure 8. For each set of runs, we introduce random noise into the height measured at the end of each stage $t + \Delta t$ to simulate the effect of noise in the atmospheric density model. We explore the impact of positive bias, in Runs 1 and 3—shown in red in figures below, and negative biases, in Runs 2 and 4—shown in blue. We also explored unbiased noise (results not shown) and observed trends that are in-between the positive and negative biased noise results. For all runs, the noise level is about 10% for the entire span of the trajectory. Finally, we compared two sets of runs, given two choices for the time step Δt , 5 and 10% respectively. These tests (results not shown) revealed a negligible impact for the stage duration on the trajectories generated via Algorithm 1. All results below correspond to $\Delta t = 10\%$.

Figure 9 displays results for Runs 1 and 3 in a manner similar to Figure 8. Results remain similar at the end of the first two stages, $t = 10$ and 20% (near the lower end of the vertical axis in the left frame and near the top in the other two frames). The increase in the lower bound for δ for Run 3 results in a shift of 2D marginal densities further away from the region depicted by the lower left circle in the left frame. These results are highlighted in the figure by the insets shown in the lower left corner of each frame, corresponding to $t = 30\%$. The upper insets mark the 60% time stamp with corresponds to the second increase of δ_{\min} for Run 3.

Further, Figures 10–12 compare conditional statistics results for the terminal velocity, flight path, and heading angles with corresponding results computed directly from *beluga* simulations. The range of uncertainties in these figures are the result of the range of options for the path constraint parameter δ . The resulting ensemble of trajectories and the associated low-dimensional manifold lead to a range of choices for possible paths also illustrated by the joint densities on the intermediate locations in Figure 9. This in turn results in a range of terminal conditions, shown in Figures 10–12. The filled symbols represent conditional expectations $\mathbb{E}[\cdot|x_{t-\Delta t}]$. The conditional PDFs are available through PLoM, however, we choose to show error bars only since the PDFs are nearly normal. The *beluga*-generated terminal conditions, shown with open symbols, are based on trajectories that start from the same intermediate locations and employ the same set of constraints as the conditional statistics using PLoM. The *beluga* results are based on the lower bound of δ_{\min} given a specific objective and thus do not exhibit any uncertainty. In these figures, and for the remainder of this section, red corresponds to Runs 1 and 3, while blue corresponds to Runs 2 and 4.

Figure 10 shows result terminal velocity at intermediate stages along the trajectory planned via PLoM. As the vehicle approaches the target, the uncertainty for v_T shrinks as all trajectories funnel toward one point according to the training data. For Runs 3 and 4, shown in the right frame, the range of path constraint values shrinks from $[50 - 400]$ to $[200 - 400]$ as the vehicle approaches the first exclusion region, and further reduces to $[300 - 400]$ for $t > 60\%$. This adjustment changes the expected terminal

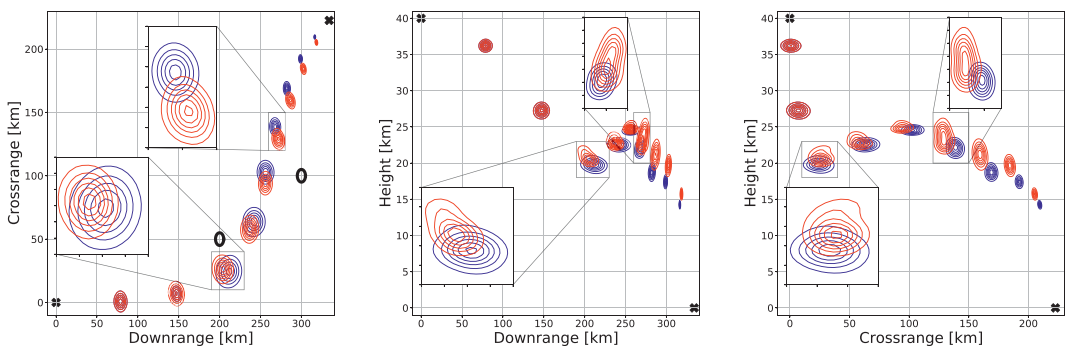


Figure 9. Marginal PDFs for vehicle location at intermediate locations for Runs 1 (red) and 3 (blue) conditional on the location at previous stages. The “x” symbols mark the start and end points and the large circles mark the location of the exclusion regions.

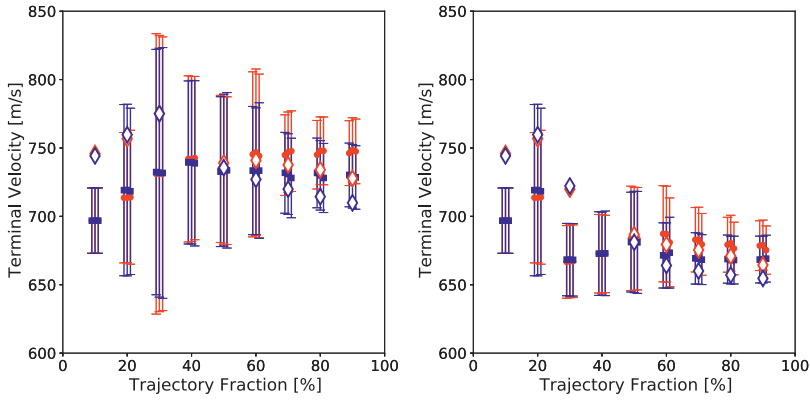


Figure 10. Means (with filled symbols) and error bars (± 2 standard deviations) for the terminal velocity v_T conditioned on intermediate conditions along the trajectory: Runs 1 and 2 (left frame) and Runs 3 and 4 (right frame). Beluga results are shown with open symbols.

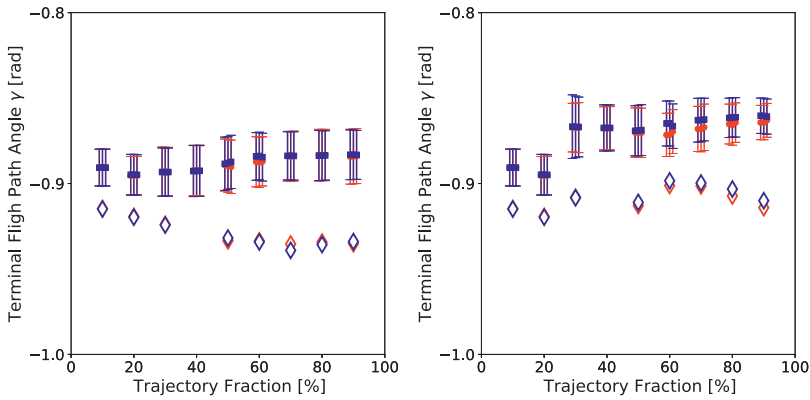


Figure 11. Mean and standard deviations for the terminal flight path angle γ_T conditioned on intermediate conditions along the trajectory. The frames setup is the same as for Figure 10.

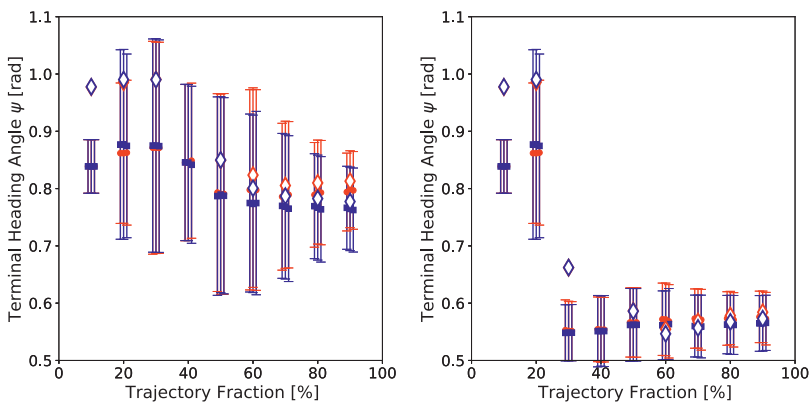


Figure 12. Mean and standard deviations for the terminal heading angle ψ_T conditioned on intermediate conditions along the trajectory. The frames setup is the same as for Figure 10.

velocity for Runs 3 and 4 compared to Runs 1 and 2 and reduces the uncertainty in these estimates. Each run was conducted three times with a different random number generator (RNG) seed. Both the RNG seed and the choice of bias, positive versus negative, have a limited impact on the results. Most of the variability here is due to the choice of δ . We also compared results using $\Delta t = 5\%$ and 10% , respectively. Similar to the previous observation, the stage length does not have a sizeable impact on the terminal conditions (results not shown).

Figures 11 and 12 show results for the terminal flight path angle γ_T and terminal heading angle ψ_T using the same settings as the results shown in Figure 10. These results indicate that the biggest driver for the range of terminal values predicted at various stages remains the range of paths taken to avoid the two exclusion regions. Similarly to the previous observations, the bias added to altitude measurements, as well as the time step size, have only a limited impact on the results. While the *beluga* results for ψ_T generally fall inside the manifold-based statistics, some discrepancy is observed for γ_T , in Figure 11. We attribute this discrepancy to the rapid change in the flight path angle as the vehicle approaches the terminal location, as seen Figure 5. Since these results represent conditional statistics constructed based on globally optimal data, we suspect these discrepancies are generated by the difference between a dataset of globally optimal trajectories and subsequent solutions that are optimal starting at intermediate points along the flight path. The magnitude of γ changes from a range of about $-0.2 \dots 0.2$ radians during the bulk of the flight to around -0.9 over the last 5% fraction of the trajectory duration. In contrast, the variation velocity and terminal heading angle changes over the last fraction of the trajectory are less nonlinear. We leave the exploration of this observation for subsequent work.

We conclude this section with a discussion on the impact of intermediate flight path and heading angles on the conditional statistics for the terminal conditions. The results presented in Figure 13 are computed as follows. In the left frame the terminal velocity statistics are conditioned on the intermediate vehicle position and velocity magnitude, $p(v_T|h_t^*, \theta_t, \phi_t, v_t)$, while the middle frame statistics correspond to $p(v_T|h_t^*, \theta_t, \phi_t, v_t, \gamma_t, \psi_t)$, and the right frame to $p(v_T|h_t^*, \theta_t, \phi_t, v_t, \gamma_t^*, \psi_t^*)$. The “*” superscript indicates that the corresponding variable was perturbed at time t to mimic either sensor inaccuracies or the impact of unaccounted external factors that lead to discrepancies between the predicted and realized vehicle trajectory over the stage $[t - \Delta t, t]$. These results indicate that constraining on all state vector components narrows the range on uncertainties at an early stage during the flight, $t = 10$ and 20% . Terminal velocities predicted at later stages are similar across the cases presented, signaling that uncertainties are now driven largely by the choice of path constraint parameter δ .

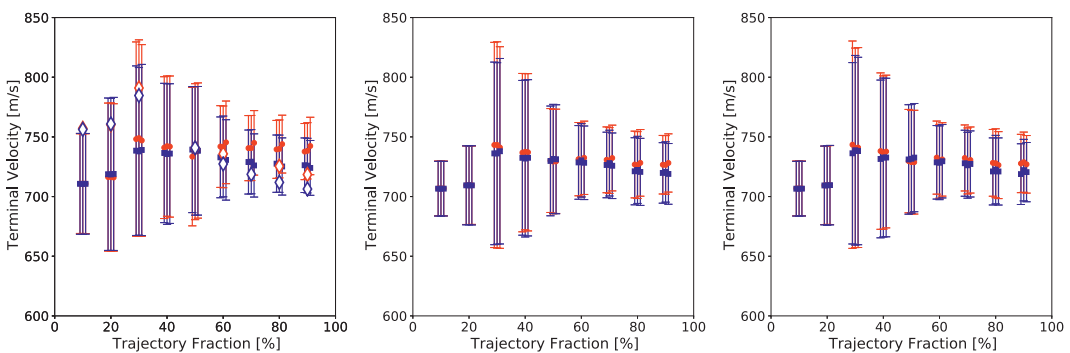


Figure 13. Mean and standard deviations for the terminal velocity v_T corresponding to Runs 1 and 2 conditioned on intermediate vehicle locations: first column—marginal over the intermediate flight path and heading angles; second column—conditioned over intermediate flight path and heading angles; and third column—conditioned over perturbed intermediate flight path and heading angles.

The path-planning workflow presented in this section employs conditional sampling on low-dimensional manifolds to generate controls that adapt sequentially to current measured state vector values along the trajectory. The PLoM framework generates solutions that probabilistically consistent with the training data for the velocity and heading angles, which the results for the flight path angles are off by 10–20%.

The computational cost of the overall approach can be split into two stages, the offline stage and the online stage. In the offline stage, trajectories are generated with *beluga* to assemble a training dataset. This is then processed using the PLoM framework described in Section 3. The resulting augmented set of trajectories and the associated diffusion vectors serve as inputs to the online path-planning algorithm presented in this section. In the online stage, this algorithm is approximately *two to three orders of magnitude less expensive* compared to the computational cost of *beluga* simulations employed to adjust the trajectory parameters. We expect the computational cost differential to become wider with increased trajectory model complexity.

5. Conclusion

We have introduced an unsupervised probabilistic learning technique for the analysis of planetary reentry trajectories. The algorithm first extracts the intrinsic structure in the data via a diffusion map approach. Using the diffusion coordinates on the graph of training samples, the probabilistic framework then augments the original data with samples that are statistically consistent with the original set. The augmented samples are used to construct conditional statistics that are ultimately assembled in a path-planning algorithm. The algorithm is designed to adjust the controls mid-flight to adapt the trajectory to changing mission objectives in real-time.

We employ a 3DOF model to generate optimal trajectories that satisfy path-constraints and maximize impact velocities. The diffusion map workflow reveals the presence of low-dimensional structures in the high-dimensional datasets. Typical manifold dimensions vary between 40 and 50 depending on the formulation employed for the control parameters.

The diffusion map algorithm revealed the presence of outliers (or edge cases). The outlier samples display diffusion coordinates that place these samples outside the “cloud” where the bulk of the samples reside. Based on this observation, we sequentially label and, if desired, remove outliers from the set of trajectories used for training.

We proposed a novel path-planning workflow that splits the planetary reentry trajectories into a set of stages. The probabilistic learning framework then utilizes conditional statistics to generate a set of controls for a specific flight stage conditioned on the information available at the beginning of each stage. The algorithm adjusts the controls during subsequent stages to account for errors due to external factors and/or due to evolving mission objectives. The bulk of the computational cost for the probabilistic control estimates corresponds to offline computations, that is, the generation of the augmented dataset based on the original training data. The online computational costs are about two to three orders of magnitude less than the cost of the OCP.

Acknowledgments. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA-0003525. This article describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Competing Interests. The authors declare no competing interests exist.

Data Availability Statement. Code available upon request.

Author Contributions. All authors contributed equally to this manuscript and have approved the final submitted draft.

Funding Statement. This work was funded by the Laboratory Directed Research & Development (LDRD) program at Sandia National Laboratories.

References

- Betts JT** (1998) Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics* 21(2), 193–207.
- Betts JT** (2010) *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd Edn. Philadelphia, PA: Society for Industrial and Applied Mathematics.
- Busemann A, Vinh NX and Culp RD** (1976) *Hypersonic Flight Mechanics*. Technical Report NASA-CR-149170, NASA.
- Cheng L, Wang Z, Jiang F and Li J** (2020) Fast generation of optimal asteroid landing trajectories using deep neural networks. *IEEE Transactions on Aerospace and Electronic Systems* 56(4), 2642–2655.
- Choi U and Ahn J** (2020) Imitation learning-based unmanned aerial vehicle planning for multitarget reconnaissance under uncertainty. *Journal of Aerospace Information Systems* 17(1), 36–50.
- Coifman RR and Lafon S** (2006) Diffusion maps. *Applied and Computational Harmonic Analysis* 21(1), 5–30.
- Fahroo F and Ross IM** (2002) Direct trajectory optimization by a Chebyshev pseudospectral method. *Journal of Guidance, Control, and Dynamics* 25(1), 160–166.
- Federici L, Benedikter B and Zavoli A** (2021) Deep learning techniques for autonomous spacecraft guidance during proximity operations. *Journal of Spacecraft and Rockets* 58(6), 1–12.
- Ghanem R and Soize C** (2018) Probabilistic nonconvex constrained optimization with fixed number of function evaluations. *International Journal for Numerical Methods in Engineering* 113, 719–741.
- Izzo D, Öztürk E and Märten M** (2019) Interplanetary transfers via deep representations of the optimal policy and/or of the value function. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. New York: Association for Computing Machinery, pp. 1971–1979.
- Kierzenka J and Shampine LF** (2001) A BVP solver based on residual control and the Matlab PSE. *ACM Transactions on Mathematical Software (TOMS)* 27(3), 299–316.
- La Mantia M and Casalino L** (2006) Indirect optimization of low-thrust capture trajectories. *Journal of Guidance, Control, and Dynamics* 29(4), 1011–1014.
- Longuski JM, Guzmán JJ and Prussing JE** (2014) *Optimal Control with Aerospace Applications*. New York: Springer.
- Loquercio A, Kaufmann E, Ranftl R, Dosovitskiy A, Koltun V and Scaramuzza D** (2020) Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics* 36(1), 1–14.
- Mall K, Grant MJ and Taheri E** (2020) Uniform trigonometrization method for optimal control problems with control and state constraints. *Journal of Spacecraft and Rockets* 57(5), 995–1007.
- Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, Kumar A, Ivanov S, Moore JK, Singh S, Rathnayake T, Vig S, Granger BE, Muller RP, Bonazzi F, Gupta H, Vats S, Johansson F, Pedregosa F, Curry MJ, Terrel AR, Roučka Š, Saboo A, Fernando I, Kulal S, Cimrman R and Scopatz A** (2017) Sympy: Symbolic computing in python. *PeerJ Computer Science* 3, e103.
- Sánchez-Sánchez C and Izzo D** (2018) Real-time optimal control via deep neural networks: Study on landing problems. *Journal of Guidance, Control, and Dynamics* 41(5), 1122–1135.
- Shi Y and Wang Z** (2020) A deep learning-based approach to real-time trajectory optimization for hypersonic vehicles. In *AIAA Scitech 2020 Forum*. Orlando, FL: AIAA.
- Shi Y and Wang Z** (2021) Onboard generation of optimal trajectories for hypersonic vehicles using deep learning. *Journal of Spacecraft and Rockets* 58(2), 400–414.
- Soize C and Ghanem R** (2016) Data-driven probability concentration and sampling on manifold. *Journal of Computational Physics* 321, 242–258.
- Soize C and Ghanem R** (2020) Probabilistic learning on manifolds. *Foundations of Data Science* 2(3), 279–307.
- Sparapany MJ** (2020) *Aerospace Mission Design on Quotient Manifolds*. PhD Thesis, Purdue University Graduate School.
- Sparapany M and Grant MJ** (2020) *beluga*. Available at <https://github.com/Rapid-Design-of-Systems-Laboratory/beluga> (accessed September 4, 2020).
- Wang Z and Grant MJ** (2017) Constrained trajectory optimization for planetary entry via sequential convex programming. *Journal of Guidance, Control, and Dynamics* 40(10), 2603–2615.
- Wang Z and Grant MJ** (2018) Autonomous entry guidance for hypersonic vehicles by convex optimization. *Journal of Spacecraft and Rockets* 55(4), 993–1006.
- Yan C, Xiang X and Wang C** (2020) Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. *Journal of Intelligent & Robotic Systems* 98(2), 297–309.
- Zheng D and Tsiotras P** (2021) Near-optimal finite-time feedback controller synthesis using supervised and unsupervised learning. In *AIAA Scitech 2021 Forum*. Orlando, FL: AIAA.

A. Appendix: Setup of Continuation Schedule for the Optimal Control Solution

Table A1 displays the continuation stages setup for the solution of the OCP for the dataset described in Section 4.1. All simulations start with a trajectory state defined by

$$x_0 = \{h_{IC}, \theta_0 = 0^\circ, \phi_0 = 0^\circ, v_{IC}, \gamma_0 = -90^\circ, \psi_0 = 0^\circ\},$$

Table A1. Continuation stages for the optimal control problem.

Stage	Parameters adjusted	No. of steps
1	$h_f \rightarrow 0, \theta_f \rightarrow 0.05^\circ$	21
2	$\theta_f \rightarrow 0.5^\circ$	21
3	$\gamma_0 \rightarrow 0, \theta_f \rightarrow 3^\circ$	41
4	$\phi_f \rightarrow 2^\circ$	41
5	$\theta_0 \rightarrow \theta_{IC}, \phi_0 \rightarrow \phi_{IC}$	21
6	$0 \rightarrow \delta \rightarrow \delta_{\max}$	41

and constraints $\theta_f = \phi_f = 0^\circ$. This essentially leads to the initial trajectory pointing straight down from the initial altitude. Trajectories are then gradually adjusted to satisfy the desired initial, intermediate, and final constraints through a set of continuation stages. During Stage 1 the final height is pulled to $h_f = 0$ and the final downrange location to $\theta_f = 0.05^\circ$. The final downrange location is pushed further out during Stage 2. During Stage 3 the initial flight path angle is adjusted from the initial straight down direction to horizontal direction and the final downrange location is adjusted to the value setup for this set of simulated trajectories, $\theta_f = 3^\circ$. The final crossrange location is pushed to the desired target, $\phi_f = 2^\circ$, during Stage 4, followed by Stage 5, during which the longitude and latitude for the start point are adjusted to the set of initial conditions θ_{IC} and ϕ_{IC} , respectively. Finally, Stage 6 adjusts the path-constraint parameter δ from 0 (no constraint) to the maximum value desired for a particular set of runs, δ_{\max} .