SAND2021-11867

# SANDIA REPORT

SAND2021-11867
Unclassified Unlimited Release
Printed September 2021

Sandia National Laboratories

# Trajectory Optimization via Unsupervised Probabilistic Learning On Manifolds

Joshua B. Ortiz, Kelli McCoy, Michael J. Grant, Michael Sparapany, Roger Ghanem, Habib N. Najm, Cosmin Safta

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.



Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

# Trajectory Optimization via Unsupervised Probabilistic Learning On Manifolds

Joshua Ortiz
University of Texas at Dallas
Richardson, TX 75080
Joshua.Ortiz3@UTDallas.edu

Kelli McCoy
University of Southern California
Los Angeles, CA 90089
kjm_422@usc.edu

Michael J. Grant
Sandia National Laboratories
Albuquerque, NM 87185-9999
mjgran@sandia.gov

Michael Sparapany
Sandia National Laboratories
Albuquerque, NM 87185-9999
mjspara@sandia.gov

Roger Ghanem
University of Southern California
Los Angeles, CA 90089
ghanem@usc.edu

Habib N. Najm
Sandia National Laboratories
Livermore, CA 94551-0969
hnnajm@sandia.gov

Cosmin Safta
Sandia National Laboratories
Livermore, CA 94551-0969
csafta@sandia.gov

SAND2021-11867

**ABSTRACT**

This report investigates the use of unsupervised probabilistic learning techniques for the analysis of hypersonic 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 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-planing algorithm. In this framework the controls are determined stage by stage during the flight to adapt to changing mission objectives in real-time. A 3DOF model was employed to generate optimal hypersonic trajectories that comprise the training datasets. The diffusion map algorithm identified that data resides on manifolds of much lower dimensionality compared to the high-dimensional state space that describes each trajectory. In addition to the path-planing workflow we also propose an algorithm that utilizes the diffusion map coordinates along the manifold to label and possibly remove outlier samples from the training data. This algorithm can be used to both identify edge cases for further analysis as well as to remove them from the training set to create a more robust set of samples to be used for the path-planing process.

## ACKNOWLEDGMENT

This work was funded by the Laboratory Directed Research & Development (LDRD) program at Sandia National Laboratories. 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 report describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the report do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

## CONTENTS



## LIST OF FIGURES

## LIST OF TABLES

# 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 hypersonic trajectory optimization problems can be generalized into two categories: indirect methods and direct methods [2]. The indirect methods are based on the Pontryagin's minimum principle and the optimal control is determined by minimizing a Hamiltonian system with respect to the control variables. Indirect 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 [14]. Direct methods discretize trajectories into multiple segments characterized by state and control variables. The optimal control problem is converted into a parameter optimization problem [9], which is typically solved via nonlinear programming [3] or convex optimization methods [33, 34]. However, the computational expense for direct methods cannot be a-priori estimated, 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.

As a result of 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 [35, 6] and agile flight guidance [17]. Deep learning has also found applications in space missions planning. Deep Neural Networks (DNN) are trained on optimal state and control vectors that come from the numerical solution of an equivalent optimal control problem (OCP). During the training, 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 [12] and planetary soft-landing maneuvers [31], as well as the real-time onboard generation of a high number of optimal trajectories for either asteroid landing [5] or atmospheric reentry of hypersonic vehicles [24, 25].

Federici *et al* [10] 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 multi-impulsive rendezvous mission. Zheng and Tsiotras [36] 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 non-stationary 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.

To address the high-dimensionality and computational cost challenges, we propose an unsupervised probabilistic learning framework, employing diffusion maps [7]. This framework assimilates the solution space of flight dynamics model inputs and outputs and both (a) identifies underlying low-dimensional manifolds, and (b) provides a stochastic differential equation model that can efficiently generate many sample trajectories on these manifolds that are probabilistically consistent with the training data. We assemble these techniques into a probabilistic learning framework where trajectories are proposed autonomously, and solution candidates can be used to evaluate risk measures associated with flight conditions that exhibit low probabilities. The diffusion map (DMAP) algorithm will assimilate computed samples adaptively until the basis sets describing the low-dimensional manifolds converge, for a given set of trajectory constraints. Then, in this joint and low-dimensional space, the algorithm will generate solution paths with limited computational resources. These trajectories will be equipped with uncertainty ranges that are consistent with the amount of (or lack of) data.

This report is organized as follows. Section 2 presents the modeling framework for this work, including the 3DOF trajectory model and the optimal control algorithm. Section 4 presents the unsupervised probabilistic learning approach, followed by the results in Section 5. The report ends with conclusions in Section 6 and appendices presenting the continuation schedule for the OCP, in Appendix A and the atmospheric density model in Appendix B.

# 2. MODELING FRAMEWORK

## 2.1. Trajectory Model

For this study, we consider a three-degree-of-freedom (3DOF) model [4] to describe the re-entry 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 $h$ is the altitude from the planet surface to the vehicle position. The three kinematic equations for the vehicle altitude $h$, longitude $\theta$, and latitude $\phi$ are given by

$$\frac{dh}{dt} = v\sin(\gamma) \tag{1}$$

$$\frac{d\theta}{dt} = v\frac{\cos(\gamma)\cos(\psi)}{r\cos(\phi)} \tag{2}$$

$$\frac{d\phi}{dt} = v\frac{\cos(\gamma)\sin(\psi)}{r} \tag{3}$$

In the results presented in this report, 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 $\mathbf{v}$ relative to the planet is expressed in terms of its magnitude $v$ and two angles: the flight path angle $\gamma$ between the velocity vector and the local horizontal plane and heading angle $\psi$ between the projection of $\mathbf{v}$ on the horizontal plane and the local latitude parallel. The force equations for these components are given by

$$\frac{dv}{dt} = \frac{1}{m}F_T - \frac{\mu\sin(\gamma)}{r^2} \tag{4}$$

$$\frac{d\gamma}{dt} = \frac{F_N}{m}\frac{\cos(\sigma)}{v} - \frac{\mu}{r^2 v}\cos(\gamma) + \frac{v}{r}\cos(\gamma) \tag{5}$$

$$\frac{d\psi}{dt} = \frac{F_N}{m}\frac{\sin(\sigma)}{v\cos(\gamma)} - \frac{v}{r}\cos(\gamma)\cos(\psi)\tan(\phi). \tag{6}$$

Here $m$ is the mass of the vehicle, $\mu = 3.986 \times 10^{14}$ m$^3$/s$^2$ is the gravitational parameter, and $(F_T, F_N)$ are the the components of the aerodynamic and propulsive forces along and perpendicular to the velocity vector. This work pertains to non-thrusting flights resulting in

$$F_T = -D, \; F_N = L \tag{7}$$

where $D$ and $L$ are the aerodynamic drag and lift forces, respectively. The bank angle $\sigma$ in Eqs. (5) and (6) accounts for the angle between the direction of the lift force $L$ and the $(\mathbf{r}, \mathbf{v})$ plane formed by the vector from the center of the planet to the vehicle location and the velocity vector.

For the remainder of this report the location and velocity components are grouped into a state vector denoted by $\boldsymbol{x} = \{h, \theta, \phi, v, \gamma, \psi\}$. The angle of attack and the bank angle are grouped into the control vector denoted by $\boldsymbol{u} = \{\alpha, \sigma\}$, and the 3DOF model can be written as $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{f}$ is the right hand side of Eqs. (1-6).

### 2.2. Vehicle Model

For this study the vehicle model is a blunt cone with mass $m = 350$ kg, reference area $A_{ref} = \pi \times 0.305^2\tilde{\mathrm{m}}^2$, and nose radius $r_n = 0.0254$ m [27]. The lift and drag coefficients are given by

$$C_l(\alpha) = c_0\,\alpha \qquad C_d(\alpha) = c_1\,\alpha^2 + c_2. \tag{8}$$

where $\alpha$ is the angle of attack (in radians). For the lift coefficient, the slope is set to $c_0 = 1.5658$. We will explore two sets of trajectories in this report. For the 1st set, the drag coefficient is independent of altitude and the model parameters are given by $c_1 = 1.6537$ and $c_2 = 0.0612$. For the 2nd set the drag coefficient depends on the altitude through the Knudsen number (Kn) defined as the ratio between the atmospheric particle mean free path and the characteristic flowfield dimension. For this work, the Kn number becomes [23]

$$Kn = m_p / \left( \sqrt{2}\pi\sigma_p^2 r_b \rho \right) \tag{9}$$

where $m_p$ is the particle mass, $\sigma_p$ is the particle reference diameter, and $r_b$ is the vehicle base radius. The particle mass is determined from the molecular weight of the air as $m_p = M_a/\mathrm{N}_a$, where $M_a = 28.9$kg/kmol (up to an altitude of 100 km), and $\mathrm{N}_a$ is Avogadro's number. The Kn number is a function of altitude through the atmospheric density (see App. B for more details on the density models).

We assume a hyperbolic tangent model for the $C_d$ as a function of Kn:

$$C_d = C_d^{(l)} + \frac{1}{2}\left[1 + \tanh(.95 + 0.41258 * \log \mathrm{Kn})\right] (C_d^{(h)} - C_d^{(l)}). \tag{10}$$

The superscripts $l$ and $h$ refer to the drag coefficient expressions for low and high altitudes, respectively. For the low altitude end, $C_d^{(l)}$, the drag coefficient employs the same constants' values as above. For the high altitude range the coefficient of drag was set much higher than the low altitide values, $C_d^{(h)} = 4 \times C_d^{(l)}$, to illustrate the impact of model fidelities on the probabilistic learning model. The variation of Kn number with altitude, in Fig. 2-1, mimics the atmospheric density altitude dependence through the Kn number. This results in a transition between lower coefficient of drag values at lower altitudes to higher values starting around 45–50 km, shown in the right frame of Fig. 2-1.

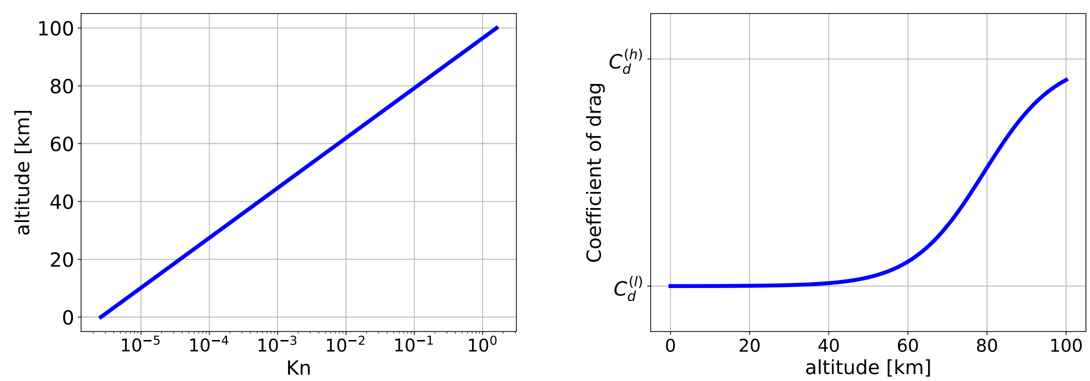


**Figure 2-1 (Left Frame) Kn number dependence on altitude and (Right Frame) Coefficient of drag dependence on altitude.**

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

$$L = 0.5\rho(h)v^2 C_l(\alpha) A_{ref}, \qquad D = 0.5\rho(h)v^2 C_d(\alpha) A_{ref} \tag{11}$$

where $A_{ref}$ is the reference area of the vehicle, assumed independent of the angle of attack. For this study we considered a circular reference area with a radius of 0.305 m. The atmospheric density was approximated with an exponential dependence on height using both 1st- and 5th-order polynomials as arguments for the exponential dependency. Appendix B presents more details about the atmospheric density models.

### 2.3. Optimal Control Problem

We construct a variational problem to generate trajectories based on the models presented in § 2.1 and § 2.2. For the dependent state vector $\boldsymbol{x}$, the variational problem seeks a time dependent angle-of-attack $\alpha$ and bank angle $\sigma$ that maximizes the magnitude of the terminal velocity $v_T$ at the desired endpoint given by $\boldsymbol{\Psi}(\boldsymbol{x}(t_f),t_f) : \mathbb{R}^6 \times \mathbb{R}^1 \mapsto \mathbb{R}^3$ while satisfying the 3DOF model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t)$. Additionally, the initial location of the vehicle is fixed with the initial constraint function $\boldsymbol{\Phi} : \mathbb{R}^6 \times \mathbb{R}^1 \mapsto \mathbb{R}^6$. In the case of $\boldsymbol{\Psi}$, we have the vector-valued function

$$\boldsymbol{\Psi}(\boldsymbol{x}(t),t) = \begin{bmatrix} h(t) - h_f \\ \theta(t) - \theta_f \\ \phi(t) - \phi_f \end{bmatrix} \tag{12}$$

The initial constraint function $\boldsymbol{\Phi}$ is defined similarly with $(h_0, \theta_0, \phi_0, v_0, \gamma_0, \psi_0)$ such that $\boldsymbol{\Phi}(\boldsymbol{x}(t),t) - \boldsymbol{x}_0 = \mathbf{0}$. Finally, the control vector $\boldsymbol{u}$ is restricted to the set of admissible controls $\mathfrak{U}$. The free-final time variational problem is posed as

$$\begin{aligned} \min_{\boldsymbol{u}(t)} J &= \int_{t_0}^{t_f} 0\, dt - v_f^2 \\ \text{Subject to: } \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t) \\ \boldsymbol{\Phi}(\boldsymbol{x}(t_0),t_0) &= \mathbf{0} \\ \boldsymbol{\Psi}(\boldsymbol{x}(t_f),t_f) &= \mathbf{0} \\ \boldsymbol{u} &\in \mathfrak{U} \end{aligned} \tag{13}$$

In the objective function above, we included an integral term that is identically 0 to be consistent with the Lagrange multiplier integral described below. We employ the *beluga* framework [27] to solve the variational problem posed in Eq. (13) using indirect methods [16]. This process is succinctly described here for completeness and in more detail in Ref. [28]. First, the initial, terminal, and dynamic path constraints are adjoined to the cost functional with Lagrange multipliers $\boldsymbol{\xi}_0$, $\boldsymbol{\xi}_f$ and $\boldsymbol{\lambda}$.

$$\min_{\boldsymbol{u}(t)} J^* = \int_{t_0}^{t_f} \boldsymbol{\lambda}^T (\boldsymbol{f} - \dot{\boldsymbol{x}}) dt + \boldsymbol{\xi}_0^T \boldsymbol{\Phi} + \boldsymbol{\xi}_f^T \Psi - v_f^2 \tag{14}$$

Next, the control Hamiltonian, $H \equiv \boldsymbol{\lambda}^T \boldsymbol{f}$, is substituted for convenience and the Euler-Lagrange operator $\mathfrak{E}$ is defined as

$$\mathfrak{E} = \frac{\partial}{\partial \boldsymbol{y}} - \frac{d}{dt}\frac{\partial}{\partial \dot{\boldsymbol{y}}} \tag{15}$$

where $\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda})$. Application of the Euler-Lagrange operator and integrating by parts solves the original variational problem. In *beluga* this map is constructed using symbolic manipulation with *SymPy* [22]. The result is an analytical solution in the form of a Hamiltonian Boundary Value Problem (HBVP),

$$\begin{aligned}
&\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}}, \qquad \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}}, \qquad 0 = \frac{\partial H}{\partial \boldsymbol{u}} \\
&\boldsymbol{\lambda}(t_0) = -\boldsymbol{\xi}_0^T \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{x}}, \qquad \boldsymbol{\lambda}(t_f) = \boldsymbol{\xi}_f^T \frac{\partial \boldsymbol{\Psi}}{\partial \boldsymbol{x}} - 2v_f, \qquad H(t_f) \quad = 0 \\
&\boldsymbol{\Phi}(\boldsymbol{x}(t_0), t_0) = 0, \qquad \boldsymbol{\Psi}(\boldsymbol{x}(t_f), t_f) = 0
\end{aligned} \tag{16}$$

The first three terms in Eq. (16) 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 Eq. (13) may be found *indirectly* by numerically solving Eq. (16). For more details regarding the numerical solution of this system see Refs. [13, 28].

## 3. TRAJECTORY DATA

For this study we consider hypersonic trajectories generated using the model choices presented in § 2.1. Some of the model parameters will be treated as random variables resulting in a number of trajectories corresponding to individual choices for these parameters. Specifically, the initial altitude, $h_0$, velocity magnitude $v_0$, longitude $\theta_0$, latitude $\phi_0$, were sampled from uniform distributions, with ranges presented in Table 3-1. The constant $H$ present in the lower fidelity atmospheric density model will also treated as a uniform random variable. The initial flight path and heading angles will be fixed to $\gamma_0 = \psi_0 = 0$, i.e. trajectories start in a horizontal plane, along the local latitude parallel. The terminal (or final) altitude $h_f = 0$ was also fixed for all data presented in this report.

We generated two trajectory datasets for this study. The first set employs an exponential density model with a 1st-order polynomial dependence on height. This dataset is described in §3.1. The second trajectory dataset, described in §3.2, consists of a matrix of runs that will be used to explore the impact of atmospheric density models (described in Appendix B) as well as the impact of Kn number on the trajectory data.

### 3.1. Single fidelity dataset

Table 3-1 provides ranges for the initial altitude $h_0$, longitude $\theta_0$, latitude $\phi_0$, velocity $v_0$, and density model scale parameter $H$. The terminal location for all trajectories was fixed to $\theta_f = 3^\circ$, $\phi_f = 2^\circ$. In addition to the parameters shown in Table 3-1, the computational model also includes a set of path constraints, resulting in trajectories that avoid regions centered around the two circles shown in the left frame of Fig. 3-1. The intensity of these constraints is controlled by an additional parameter, $\delta$. When setting $\delta = 0$, the path constraints are not activated, resulting in a set of trajectories depicted in gray in Fig. 3-1. The optimization framework uses a numerical continuation algorithm to increase the strength of path constraint expressions, resulting in the end

**Table 3-1 Parameter ranges for the uniform random variables that control the trajectory dataset #1 .**

| | Parameter | | | | |
|---|---|---|---|---|---|
| | $h_0$ [m] | $\theta_0[deg]$ | $\phi_0$ [deg] | $v_0$ [m/s] | $H$ [m] |
| min | $38 \times 10^3$ | $-0.3$ | $-0.3$ | 2000 | 7000 |
| max | $45 \times 10^3$ | 0.3 | 0.3 | 2200 | 8000 |

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.

For this study, we employed $1,100$ samples based on parameter choices drawn from the uniform distributions presented in Table 3-1. For each sample, we generated 41 trajectories corresponding to equally spaced $\delta$ values between $\delta = 0$ and $\delta = 400$.

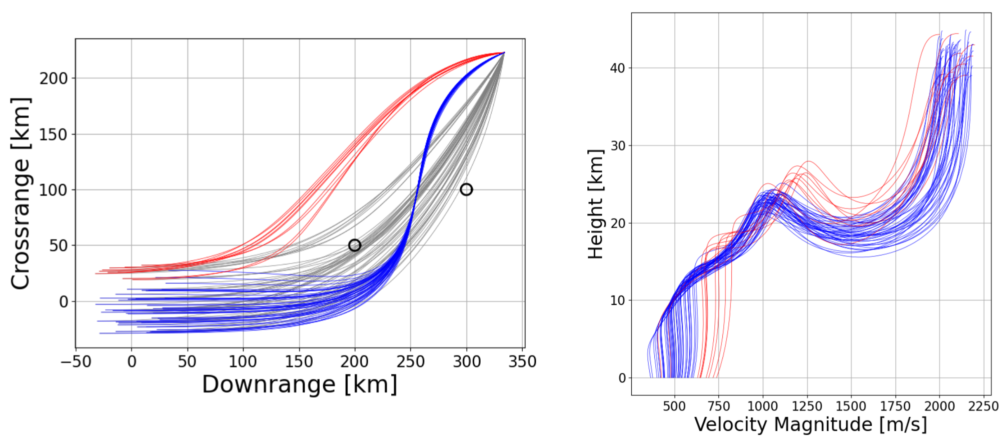


**Figure 3-1 Sample trajectory data. The left frame shows the downrange/crossrange solution components and the right frame shows velocity/height components. The samples shown in gray represent the unconstrained components while the red/blue samples correspond to the maximum path constraint condition, $\delta = 400$.**

For each parameter sample, the time-dependent values for the state vector (location and velocity components) and the control variables (angle of attack and bank angle) 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 are also appended to the solution vector, in addition to the value of path-constraint parameter $\delta$. Each trajectory sample becomes a row in the matrix of samples that will be processed via the algorithms presented in the next 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.

### 3.2. Multifidelity dataset

For these runs, the setup for $\theta_0$, $\phi_0$, and $v_0$ is the same as for the previous dataset, while the range for $h_0$ was changed to 70–85 km. This dataset consists of 4 subsets. All subsets share the same initial conditions. Table 3-2 describes the choice of models for the atmospheric density and Kn number on/off switch that generate different levels of model fidelity for these subsets. The terminal location for all trajectories in these dataset was fixed to $\theta_f = 4°$, $\phi_f = 3°$. Fig. 3-2 shows

**Table 3-2 Choice of models for the multi-fidelity dataset.**

| | | Density | |
|---|---|---|---|
| | | $\rho_L$ | $\rho_H$ |
| Kn | off | $\text{Run}_L$ | $\text{Run}_H$ |
| | on | $\text{Run}_{L,Kn}$ | $\text{Run}_{H,Kn}$ |

a subset of 50 sample pairs from $\text{Run}_{H,Kn}$. Trajectories shown in gray correspond to the unconstrained set, $\delta = 0$, while the ones shown in blue correspond to $\delta = \delta_{\max} = 400$. For this dataset trajectories always pass on one side of the two constrained regions.

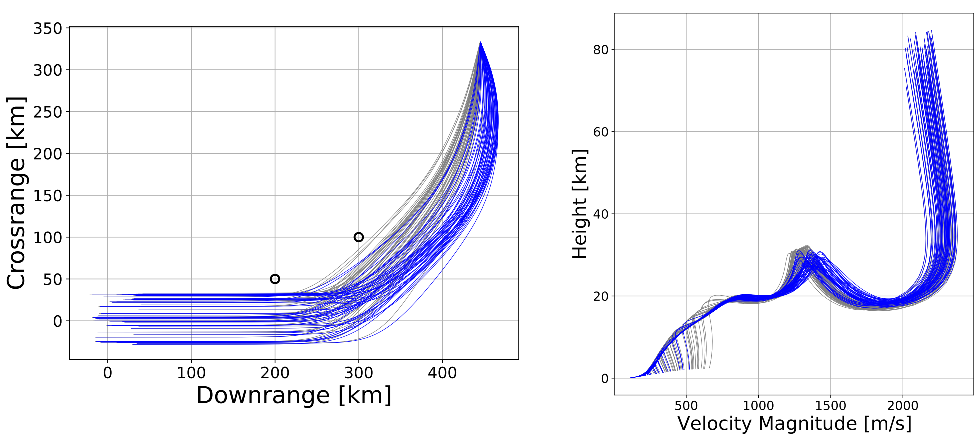


**Figure 3-2 Sample trajectory data from $\text{Run}_{H,Kn}$. The left frame shows the downrange/crossrange solution components and the right frame shows velocity/height components. The samples shown in gray represent the unconstrained trajectories while the blue samples correspond to the maximum path constraint condition, $\delta = 400$.**

## 4. PROBABILISTIC LEARNING ON MANIFOLD

Probabilistic learning on manifolds (PLoM) is a recently developed unsupervised learning technique for augmenting small datasets in a principled manner. An intrinsic structure is first extracted from the initial training dataset and is subsequently used to constrain statistical sample generation. The intrinsic structure takes the form of diffusion coordinates on the graph of the training dataset in its feature space, while the sample generation is in the form of a projected Itô equation constrained to the span of these diffusion coordinates. Each additional sample is a statistical replica of the training dataset, restricted to the same same dominant diffusion coordinates. In this section, we next summarize this construction with the requisite technical details for a self-contained assessment of the paper. A more complete presentation can be found elsewhere [26].

We construe the initial, training, dataset $\boldsymbol{x}^d$, with $N$ samples and $n$ features, as a realization of a $n \times N$ matrix-valued random variable $\boldsymbol{X}$ with $n$ rows and $N$ columns. Our objective is to generate a new dataset $\boldsymbol{x}_{\mathrm{a}}$, the augmented dataset, as realizations of $\boldsymbol{X}$. This new dataset is then construed as a surrogate for the joint probability density function of the $n$ features and is used for various statistical tasks, including the estimation of marginal and conditional density functions and for non-parametric regression. The first step, is to de-correlate the features of $\boldsymbol{X}$, through a linear transformation $\boldsymbol{H} = \mu^{-1/2}\boldsymbol{\Phi}^T(\boldsymbol{X} - \bar{\boldsymbol{x}})$ where $\mu$ is a diagonal matrix with the dominant $\nu$ eigenvalues of the $n \times n$ covariance matrix of $\boldsymbol{X}$m and $\boldsymbol{\Phi}$ is a $n \times \nu$ matrix of the associated eigenvectors. Also, $\bar{\boldsymbol{x}}$ 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 de-correlation step is a straightforward application of the standard PCA procedure to the dataset $x^d$.

We will denote samples of $\boldsymbol{X}$ by $\boldsymbol{x}$ and samples of $\boldsymbol{H}$ by $\boldsymbol{\eta}$. The sample of $\boldsymbol{H}$ associated with $\boldsymbol{x}^d$ 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, $\varepsilon$ is a parameter of the kernel, typically characterizing its bandwidth. We next construct an $N \times N$ diffusion matrix $\boldsymbol{K}$, from the training dataset, such that $\boldsymbol{K}_{ij} = k(\boldsymbol{\eta}_i^d, \boldsymbol{\eta}_j^d)$. We then normalize $\boldsymbol{K}$ so that the sum over each of its rows is equal to 1. The resulting stochastic matrix, $\boldsymbol{P}$, can thus serve as a transition matrix for a Markov chain on the graph. Its eigenvectors are denoted by $\boldsymbol{\psi}^\alpha, \; \alpha = 1, \cdots, N$. Given its construction, the largest eigenvalue of $\boldsymbol{P}$ is equal to 1 and the associated eigenvector is a constant. The $N \times m$ matrix consisting of the $m$ eigenvectors associated with the largest $m$ eigenvalues (not counting the largest unit eigenvalue) is denoted by $\boldsymbol{g}$. It should be noted that matrix $\boldsymbol{P}$ is not symmetrical, and care should therefore be taken in evaluating the associated eigenproblem. The eigenspectrum of $\boldsymbol{P}$ 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, $\boldsymbol{g}$, 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 random variable $\boldsymbol{H}$. 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 (KDE). Then the joint occurrence of the whole graph is assumed to be achieved through the independent occurrence of each of its $N$ vertices. The joint PDF model, in $\mathbb{R}^{N \times \nu}$, for the whole graph is therefore the product of $N$ joint pdfs, each defined in $\mathbb{R}^\nu$ 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,

$$q_{\boldsymbol{H}}(\boldsymbol{\eta}) = \frac{1}{N}\prod_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{h}k(\boldsymbol{\eta}^{d,j},\boldsymbol{\eta}^{i};\xi) \ , \tag{17}$$

where $k(\boldsymbol{\eta},\boldsymbol{\eta}';\xi)$ is a Gaussian kernel on $\mathbb{R}^n$ with bandwidth $\xi$, $\boldsymbol{\eta}^{d,j}$ 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 [26] for selecting the bandwidth $\xi$ 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 an Itô equation whose invariant measure is the above KDE. We then restrict both the potential and the samples of this equation to the span of the eigenvectors $\boldsymbol{g}$. This is accomplished using the following stochastic differential equation,

$$\begin{aligned} d\boldsymbol{Z}(\zeta) &= \boldsymbol{Y}(\zeta)d\zeta \\ d\boldsymbol{Y}(\zeta) &= L(\boldsymbol{Z}(\zeta))d\zeta - \frac{1}{2}f_0\boldsymbol{Y}(\zeta)d\zeta + \sqrt{f_0}d\boldsymbol{W}(\zeta), \\ &\boldsymbol{Z}(0) = \boldsymbol{H}_d\boldsymbol{a} \qquad \boldsymbol{Y}(0) = \boldsymbol{N}\boldsymbol{a} \qquad \boldsymbol{a} = \boldsymbol{g}(\boldsymbol{g}^T\boldsymbol{g})^{-1} \end{aligned} \tag{18}$$

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

After a brief non-stationary period in the Itô dynamics, samples of $\boldsymbol{H}$ are reconstructed from samples of $\boldsymbol{Z}$ as

$$\boldsymbol{\eta}^\ell = \boldsymbol{Z}^\ell\boldsymbol{g}^T \quad \ell = 1,\cdots,n_{\mathrm{MC}} \ . \tag{19}$$

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

## 5. MANIFOLD LEARNING RESULTS

In this section we will characterize the set of trajectories discussed in §3 using the algorithms presented in the previous section. We will first inspect the topology of these manifolds via their corresponding basis vectors in §5.1. We will then verify in §5.2 that the augmented set of trajectories generated via manifold sampling are consistent with the 3DOF model, and then introduce in §5.3 a workflow for sequential path planing using the augmented dataset to generate conditional statistics for the state and control vectors. We conclude this section with an illustration of manifold statistics in a multifidelity setting in §5.4.

### 5.1. Manifold Characterization

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 single fidelity 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 $\delta$. Fig. 5-1 displays the eigenvalue spectra for several values for the kernel bandwidth $\varepsilon$. The results in the left frame of this figure indicate a manifold dimension $m = 52$, for a graph Laplacian using a kernel bandwidth of $\varepsilon = 1000$. For smaller bandwidth values, e.g. for $\varepsilon = 200$, the eigenvalue decay is mild indicating a a less-defined structure in the high-dimensional trajectory space. The topology of the first two basis vectors is presented in Fig. 5-2. The first row in this figure is constructed with the same dataset as for the results presented in the left frame of Fig. 5-1. These plots shows 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, $\varepsilon = 1000$. It appears these samples are significantly different compared to the rest of the training set and their presence impact the magnitude of the basis vectors components resulted from the diffusion manifold algorithm.

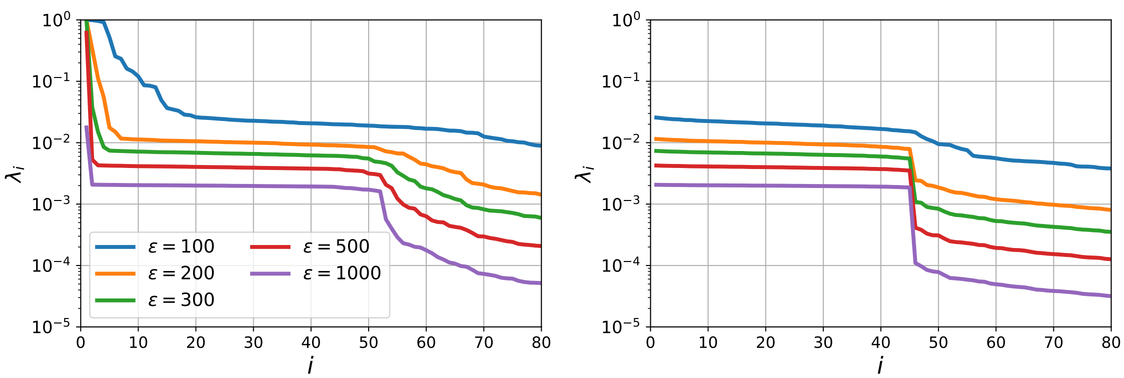


**Figure 5-1 Eigenvalues spectrum for several values for the kernel bandwidth $\varepsilon$: (left frame) results corresponding to the entire training set; (right frame) results obtained after several edge samples were removed from training.**

We proceed to examine the impact of removing these edge (outlier) cases on the diffusion manifold basis vectors. The remaining rows in Fig. 5-2 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 2nd row, removed from training. As more edge samples are removed from the training dataset, the manifold structure, illustrated through a projection on the first two basis vectors, begins to form for smaller bandwidth values. 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 Fig. 5-1. 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, i.e. the training dataset is more compact after removing edge cases identified through a sequence of diffusion manifold basis vectors.

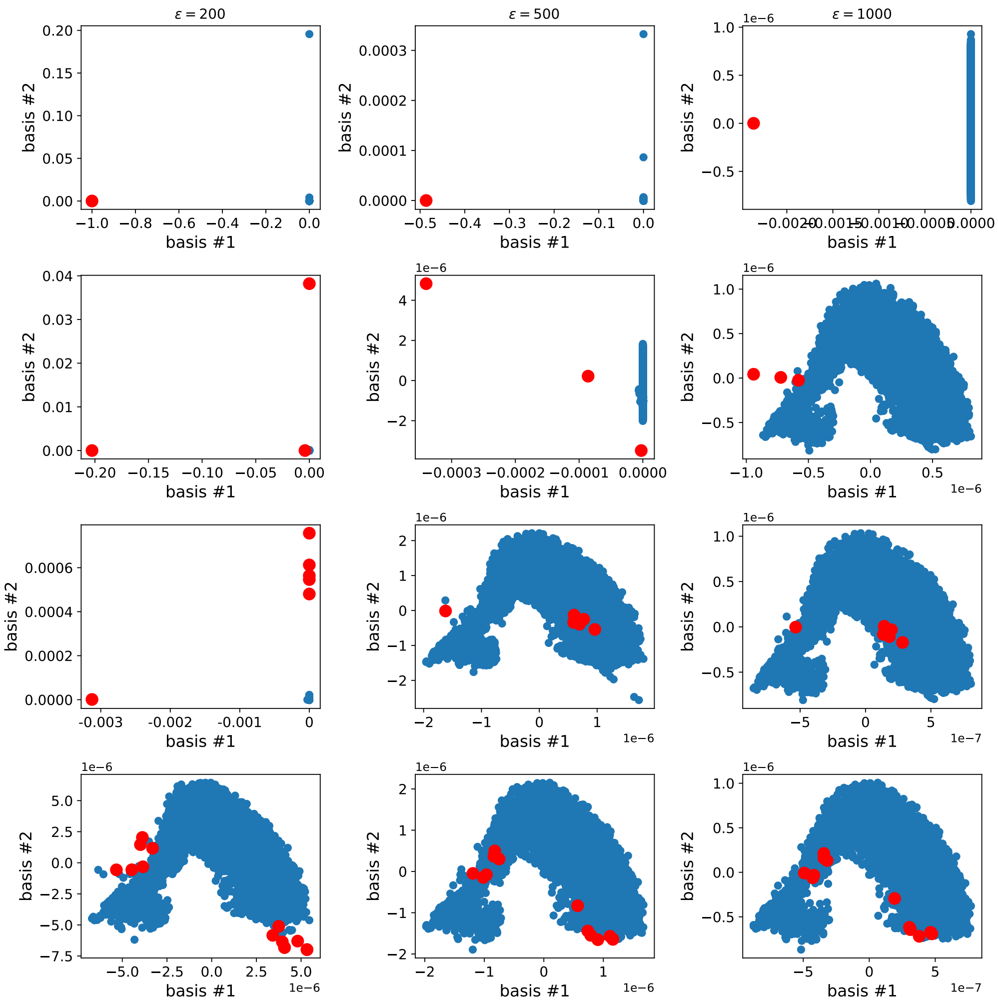


**Figure 5-2 Scatter plots showing the entries in the two most dominant eigenvectors. Left to right columns correspond to diffusion map results based on $\varepsilon = \{200, 500, 1000\}$, respectively. The top row shows the original dataset, while subsequent rows show results with an increasing number of edge cases, identified by red circles, removed at each stage.**

Fig. 5-3 shows the dependency of the manifold dimension $m$ on the kernel bandwidth $\varepsilon$. The results presented in this figure are based on the same sequence of datasets underlying the results shown in Fig. 5-2. For small bandwidth values the set of trajectories behave as an ensemble of independent samples. As the KDE kernel bandwidth increases, a set of dominant basis vectors are revealed. For results corresponding to the original dataset, there is one training sample that is sufficiently different compared to 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, the manifold is revealed for smaller bandwidth values, e.g. $\varepsilon \approx 250$, compared to the previous, much larger bandwidth values. The trend continues as subsequent edge cases are removed.

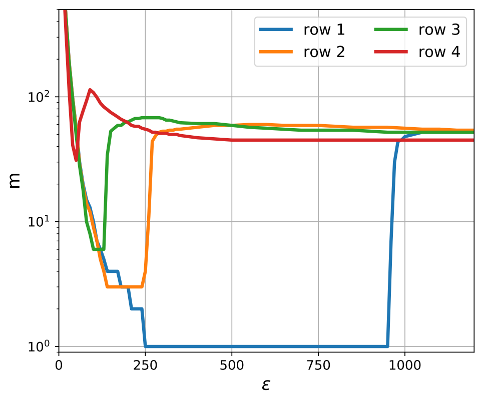


**Figure 5-3 Manifold dimension $m$ dependence on $\varepsilon$. Sets 1 through 4 correspond to the results presented on rows 1 through 4 in Fig. 5-2.**

Figs. 5-4 and 5-5 show the trajectories present in the training sets with gray lines and several sets of edge samples with thick colored lines. The rajectory samples illustrated in physical coordinates reveal characteristics that likely lead to their placement far away from other samples along the diffusion manifold coordinates. The trajectory represented in red correspond to the first detected edge sample. While not being the fastest trajectory, this sample exhibits less variation in the altitude history and the flight path angle $\gamma$. Similarly, edge samples shown with other colors suggest that combinations of features, e.g. trajectories starting in the bottom left in Fig. 5-5, or some of the samples starting at larger initial latitude $\Phi$ values but evolving close to the exclusion zones lead to their placement far away from other samples on the manifold.

## 5.2. Conditional Trajectories

In this section we explore the degree to which the augmented set of samples created via the stochastic differential system in Eq. (18), are consistent with the 3DOF model used to generate optimal trajectories via the OCP. Specifically, for each synthetic trajectory in the augmented set, we evaluate the time derivatives in the left-hand side of Eqs. (1-6) via finite differences, then compute the discrepancy with the right-hand side

$$r_{i,j} = \frac{x_{i,j} - x_{i-1,j}}{t_i - t_{i-1}} - \frac{1}{2}\left(f_j(x_{i-1}, u_{i-1}) + f_j(x_i, u_i)\right) \tag{20}$$

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

$$r_{i,j} \rightarrow r_{i,j} \times \frac{2}{x_{i-1,j} + x_{i,j}} \tag{21}$$

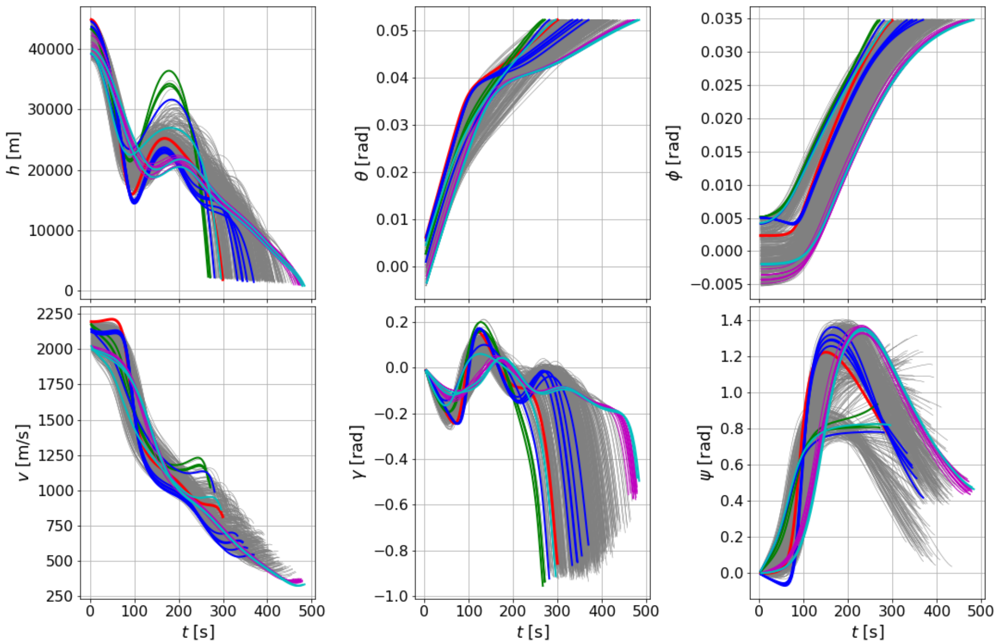


**Figure 5-4 Trajectory data defining the samples used for learning the diffusion map representation: grey lines show a random subset of 100 samples; red/blue/green/magenta/cyan show samples removed from the learning set (in this order).**

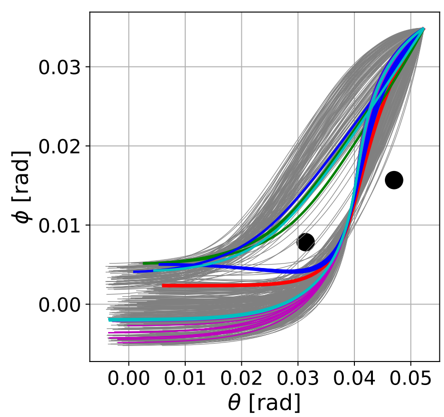


**Figure 5-5 Same dataset and color scheme as in Fig. 5-4, shown in longitude/latitude coordinates.**

Finally, for each synthetic trajectory we evaluate the $L_2$-norm. These results are collected for all augmented trajectory samples constructed via Eq. (18). Fig. 5-6 shows histograms for several sets of trajectories conditioned on select terminal velocity values $v_T$ and path constraint parameter $\delta$ 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 trajectory conditioned on the same terminal velocity but increasingly path constrained, with $\delta = 200$ and 300, respectively. We selected several $\varepsilon$ values to asses the impact the KDE bandwidth for the manifold construction ultimately has on manifold sampled trajectories. For all cases the scaled 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.

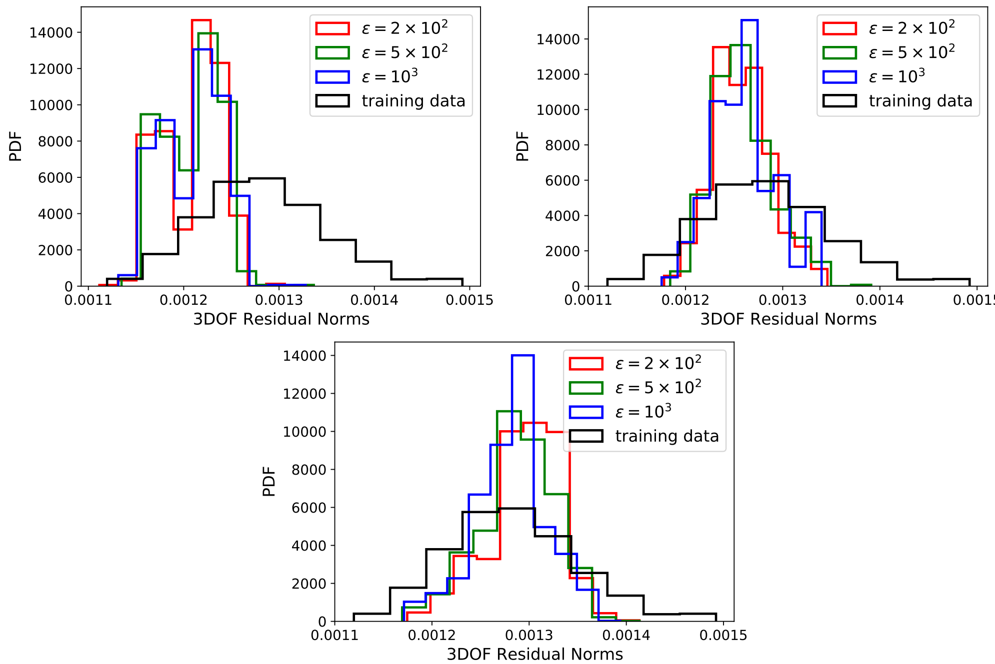


**Figure 5-6 Distribution of 3DOF residual norms for synthetic trajectories conditioned on the terminal velocity of 650 m/s and several values for $\delta$: $100$ (top left frame), $200$ (top right frame), and $300$ (bottom frame).**

### 5.3. Path Planning via PLOM

We continue our analysis with 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 Alg. 1, by defining an objective for the vehicle trajectory. This objective consists of an ensemble of constraints. 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 Eq. (13). Rather, we would like to select from the manifold of optimal solutions, constructed as described in §2.3, the set of samples that satisfy additional constraints related to the path constraint parameter $\delta$ and 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 $\Delta t$ 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, that the expected state vector at $t + \Delta t$, $x^{(e)}_{t+\Delta_t}$, will not match the actual (or measured) solution, $x^{(m)}_{t+\Delta_t}$. If this discrepancy is larger than a lower bound threshold $\varepsilon_{\Delta_t}$, or if the overall objective has to be adjusted, i.e. $\mathrm{obj}_{t+\Delta t} \neq \mathrm{obj}_t$, we proceed to update the set of controls for the next flight segment on line 7.

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

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

Fig. 5-7 displays the conditional marginal PDFs for the vehicle location, $p((\theta,\phi)_{t+\Delta t}|x^{(m)}_t)$ (left frame), $p((h,\theta)_{t+\Delta t}|x^{(m)}_t)$ (middle frame), and $p((h,\phi)_{t+\Delta t}|x^{(m)}_t)$ (right frame). Also shown in these figures are the start (left frame, lower left corner) and end (left frame, upper right corner) positions, with "x" marks, and the location of the exclusion regions, with solid circles. These results are based on the same initial condition for the vehicle height, velocity, and longitude $\{4 \times 10^4[\mathrm{m}], 2000[\mathrm{m/s}], 0[\mathrm{rad}]\}$ and the initial density model constant $H = 7700\mathrm{m}$. Four sets of results are shown in this figure corresponding to initial latitude values of $\{0(\mathrm{black}), 0.001(\mathrm{red}), 0.002(\mathrm{green}), 0.003(\mathrm{blue})\}$ (in radians), respectively. The sequence of stages in this figure are constructed using Alg. 1. At each intermediate stage $t$, the algorithm computes the statistics for the set of trajectories at $t + \Delta t$ by conditioning on the current state vector $x^{(m)}$ at time $t$. The algorithm further conditions on the current density model constant $H = H(t)$. For the results displayed in this figure, we assume that $H(t) = 7700 - t \times 300$ m, where time $t$ was normalized by the trajectory duration, i.e. $t \in [0, 1]$. The range of uncertainties that arise from one stage to the next are due the range of training data used for this demonstration. Here we consider results corresponding to $\mathrm{obj}_t \equiv \{50 < \delta \leq 400\}$. The range of path constraint values results in the statistics shown in Fig. 5-7. These results illustrate conditional statistics constrained on a manifold corresponding to a multi-modal behavior. The multi-modality here is induced by set of trajectories that avoid the two regions shown by circles in Fig. 5-7. 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 Alg. 1, the conditional densities for the vehicle location

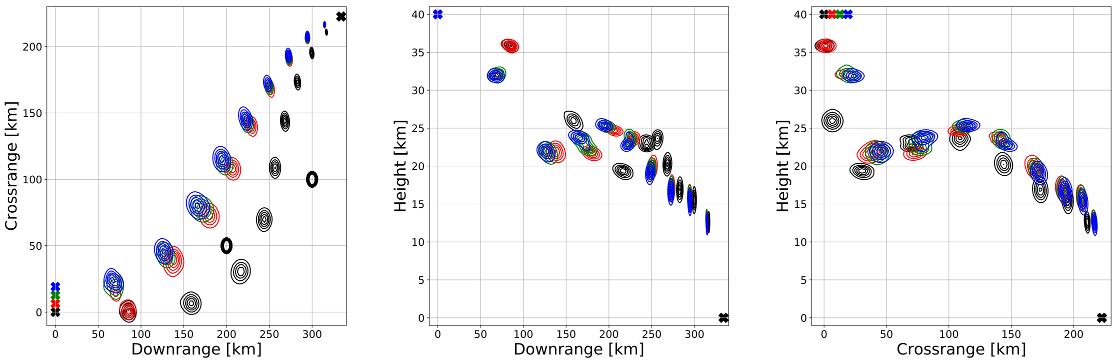


**Figure 5-7 2D pdf's for vehicle location at intermediate locations for trajectories that originate at $h_0 = 40$ [km], $\theta_0 = 0$ [rad], and $\phi_0 = \{0(\text{black}), 0.001(\text{red}), 0.002(\text{green}), 0.003(\text{blue})\}$ [rad]. The "x" symbols mark the start and end points and the large circles mark the location of the exclusion regions.**

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) displays a bi-modal behavior. At later times this trajectory stays on the same side as the trajectories shown in green and blue.

We will further illustrate this workflow with several numerical examples that employ the manifold statistics constructed with trajectory datasets presented in previous sections. Table 5-1 lists choices for several algorithm knobs chosen for these runs. Runs 1 and 2 employ the same objective throughout the entire trajectory, with $v_T > v_{T,\min} = 650$m/s and $\delta > 50$. For Runs 3 and 4, the flight path constraint $\delta$ is adjusted after 30% and 60%, respectively, of the entire trajectory duration. All runs share the same initial condition as the simulation shown in black in Fig. 5-7. 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 $\Delta t$, 5% and 10% respectively. These test (results not shown) revealed a negligible impact for the stage duration on the trajectories generated via Algorithm 1. All results below correspond to $\Delta_t = 10\%$.

Fig. 5-8 displays results for Runs 1 and 3 in a manner similar to Fig. 5-7. In addition to the constraints underlying the results in the previous figure, here we consider a setup for which the lower bound for $\delta$ changes during the flight, for Run 3. The conditional statistics are further constrained on $v_T > 650$ m/s. Results remain similar at the end of the first two stages, $t = 10\%$ and 20%, respectively (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 $\delta$ 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. The subsequent adjustment of $\delta_{\min}$ at $t = 60\%$ has a negligible impact on trajectories beyond this

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

**Table 5-1 Numerical settings for the set of runs chosen to illustrate the workflow in Alg. 1. For all runs $v_{T,\text{min}} = 650$ m/s.**

point since by this time the vehicle is sufficiently far from the region depicted by the upper right circle (results not shown).

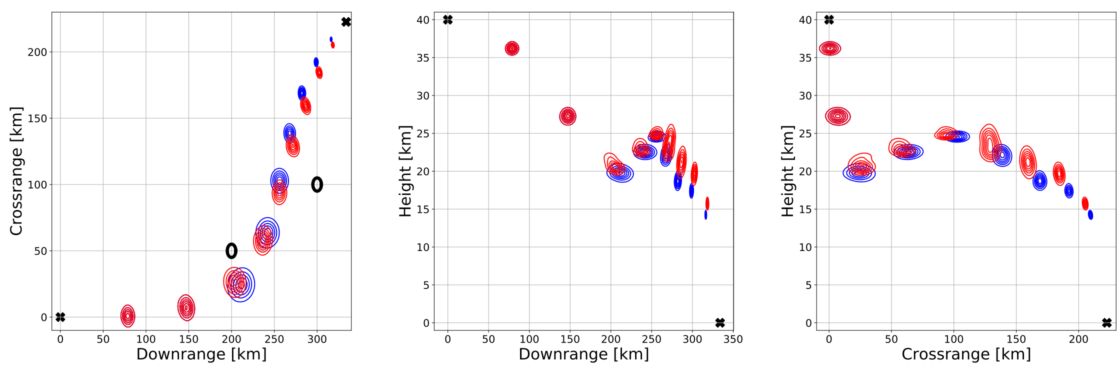


**Figure 5-8 Marginal PDFs for vehicle location at intermediate locations for Runs 1 (red) and 3 (blue) conditional on the location at previous stages. The "x" symbols mark the start and end points and the large circles mark the location of the exclusion regions.**

Further, we will compare conditional statistics results for the terminal velocity, flight path, and heading angles with results from *beluga* simulations. The *beluga*-generated trajectories start from the same intermediate locations and employ the same set of constraints as the conditional statistics using the low-dimensional manifold information. Fig. 5-9 shows, with filled symbols, the expected terminal velocity conditioned on the intermediate conditions at different stages along the trajectory, $\mathbb{E}[v_T|x_t]$. The error bars represent two standard deviations on either side of the mean value. The open symbols represent terminal velocity values computed with beluga, starting from select intermediate stages. In this figure, and for the remainder of this section, red corresponds to Runs 1 and 3 for which a positive multiplicative error bias was added to the altitude at each intermediate stage, while blue corresponds to Runs 2 and 4 for which a negative multiplicative error bias was added to the altitude at each intermediate stage.

We will first discuss the source of uncertainties illustrated in these figures. These are the result of the range of options for the path constraint parameter $\delta$. The resulting ensemble of trajectories and the associated low-dimensional manifold leads to a range of choices for possible paths also illustrated by the joint densities on the intermediate locations in Fig. 5-8. This in turn results in a range of terminal conditions, shown in Fig. 5-9. As the vehicle approaches the target, the uncertainty for $v_T$ shrinks as all trajectories funnel towards one point according to the training data. For Runs 3 and 4, shown in the right frame, the range of path constrain values was shrunk from $[50-400]$ to $[200-400]$ as the vehicle approached the first exclusion region, and further reduced to $[300-400]$ for $t>60\%$. This adjustment changes the expected terminal 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 each with a different random number generator (RNG) seed. Both the RNG seed and the choice of bias, positive vs negative, have a limited impact of the results. Most of the variability here is due to the choice of $\delta$. 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, given the setup of this numerical experiment (results not shown).

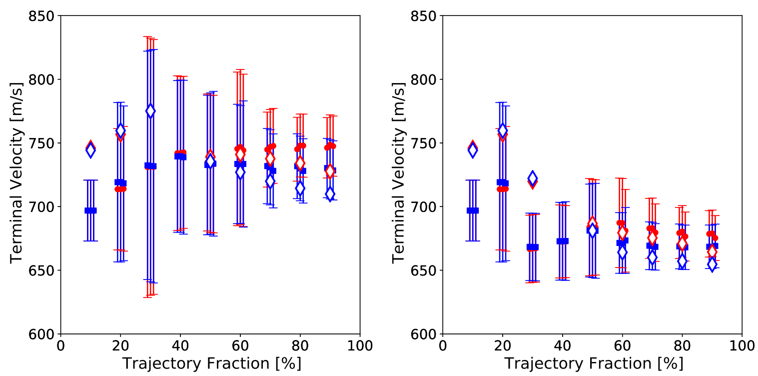


**Figure 5-9 Mean and standard deviations for the terminal velocity $v_T$ conditioned on intermediate conditions along the trajectory: Runs 1 and 2 (left frame) and Runs 3 and 4 (right frame).**

Figs. 5-10 and 5-11 show results for the terminal flight path angle $\gamma_T$ and terminal heading angle $\psi_T$ using the same settings as the results shown in Fig. 5-9. 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 and the RNG seed associated with it as well as the time step size has only a limited impact for the datasets presented in this report. While the *beluga* results for the terminal velocity and heading angle generally fall inside the manifold-based statistics, some discrepancy is observed for the flight path angle, in Fig. 5-10. We attribute this discrepancy to the rapid change in the flight path angle as the vehicle approaches the terminal location, as seen Fig. 5-4.

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 Fig. 5-12 are computed as follows. In the left frame the terminal velocity statistics are conditioned on the intermediate vehicle position and velocity magnitude, $p(v_T|h_t^*,\theta_t,\phi_t,v_t)$, 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 $p(v_T|h_t^*,\theta_t,\phi_t,v_t,\gamma_t^*,\psi_t^*)$. The "*" superscript indicate 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 velocity predicted at later stages are similar across

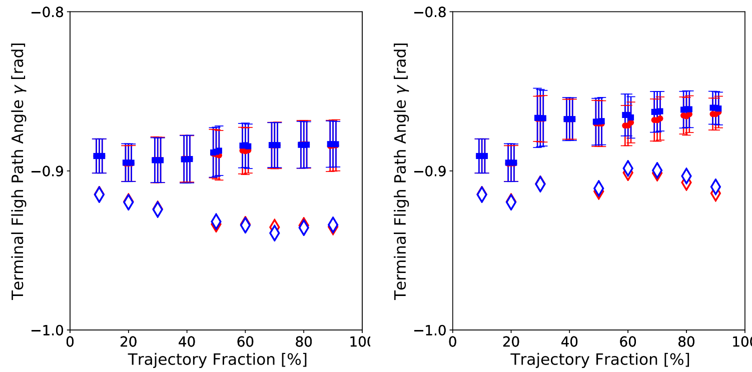


**Figure 5-10 Mean and standard deviations for the terminal flight path angle $\gamma_T$ conditioned on intermediate conditions along the trajectory. The frames setup is the same as for Fig. 5-9.**

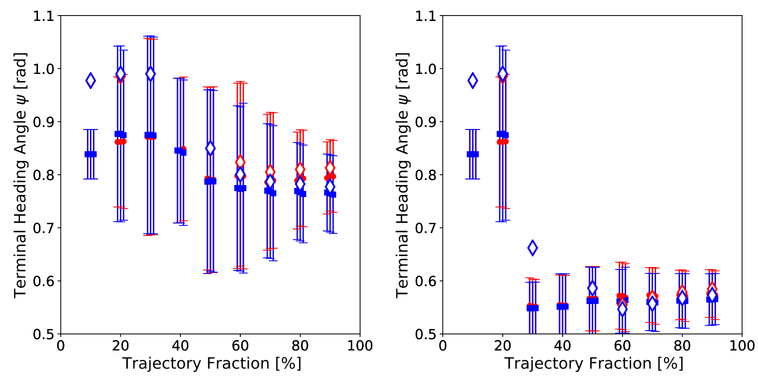


**Figure 5-11 Mean and standard deviations for the terminal heading angle $\psi_T$ conditioned on intermediate conditions along the trajectory. The frames setup is the same as for Fig. 5-9.**

the cases presented in this figure signaling that uncertainties are now driven largely by the choice of path constraint parameter $\delta$.

## 5.4. Diffusion Manifolds in a Multifidelity Framework

In this section we present diffusion manifold results based on the trajectory datasets introduced in §3.2. We will analyze results based on both individual runs, e.g. $\mathrm{Run}_L$ and $\mathrm{Run}_H$, as well as data constructed from combined datasets, e.g. $\mathrm{Run}_H$ and $\mathrm{Run}_{H,Kn}$. We adopt two approaches to combine datasets of various fidelities. In the first approach, referred to as $\Delta_1$ below, individual trajectories are interpolated on uniform time grids. The ensembles of state vectors corresponding to the same trajectory conditions are then joined together to form one sample, i.e. one row in the dataset that feeds into PLoM. Eq. (22) describes this approach for samples corresponding to $\mathrm{Run}_H$ and $\mathrm{Run}_{H,Kn}$

$$\Delta_1:\left(\cup_{i=1}^{N_t} t_i^{(j)}{}_H\right)\cup\left(\cup_{i=1}^{N_t} x_i^{(j)}{}_H\right)\cup\left(\cup_{i=1}^{N_t} u_i^{(j)}{}_H\right)\cup\left(\cup_{i=1}^{N_t} x_i^{(j)}{}_{H,Kn}\right)\cup\left(\cup_{i=1}^{N_t} u_i^{(j)}{}_{H,Kn}\right). \tag{22}$$

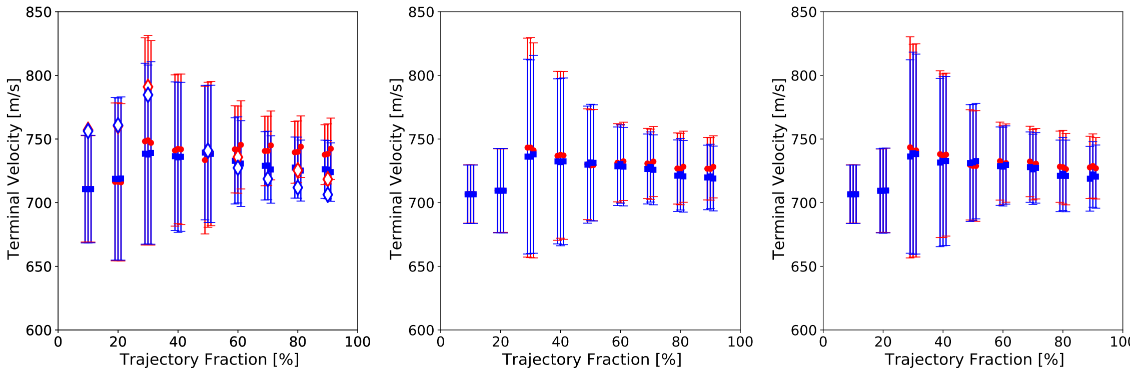


**Figure 5-12 Mean and standard deviations for the terminal velocity $v_T$ corresponding to Runs 1 and 2 conditioned on intermediate vehicle locations: 1st column - marginal over the intermediate flight path and heading angles; 2nd column - conditioned over intermediate flight path and heading angles; 3rd column - conditioned over perturbed intermediate flight path and heading angles.**

Here, $x_i^{(j)}$ and $u_i^{(j)}$ refer to the state and control vectors, respectively, corresponding to time step $t_i$ for sample $j$, and $N_t$ is the number of equally spaced time steps. For the results presented in this section, $N_t$ was set to either 100 or 200.

In the second approach, referred to as $\Delta_2$ below, the state vectors corresponding to adjacent model fidelities are subtracted from each other. Eq. (23) presents this approach

$$\Delta_2 : \left(\cup_{i=1}^{N_t}\left({t_i^{(j)}}_H - {t_i^{(j)}}_{H,Kn}\right)\right) \cup \left(\cup_{i=1}^{N_t}\left({x_i^{(j)}}_H - {x_i^{(j)}}_{H,Kn}\right)\right) \cup \left(\cup_{i=1}^{N_t}\left({u_i^{(j)}}_H - {u_i^{(j)}}_{H,Kn}\right)\right). \quad (23)$$

For this approach the differences between the corresponding state vectors are done component by component, and similar for the control vectors.

Preliminary tests showed that the control angles display discontinuities induced by the solution procedure and $\pm 2\pi$ periodicity for sin and cos functions. Instead of working directly with the control angles we instead employed $\{\alpha_i \sin\sigma_i, \alpha_i \cos\sigma_i\}$ to replace the set of controls $u_{t_i} = \{\alpha_i, \sigma_i\}$. This improved the smoothness in the manifold topology, dropping the manifold dimension from about 50 to less than 35 when assimilating trajectory data from single models.

We also formulate an approach to eliminate outliers/edge samples; this formalizes the visual inspection approach that utilized the components of the first few diffusion map vectors to identify samples with coordinates vastly different than all the others and eliminate them from the learning set. We employ the expression below [7] to compute pairwise distances along the diffusion manifold between samples

$$d_{ij} = \left(\sum_{k=1}^{m} \lambda_k^2 \left(\Psi_{i,k} - \Psi_{j,k}\right)^2\right)^{1/2} \quad (24)$$

where $m$ is the manifold dimension defined in §4, and $\Psi_{i,k}$ and $\Psi_{j,k}$ are the entries for samples $i$ and $j$, respectively, in the diffusion manifold basis vector $k$. Alg. 2 presents a sequence of steps employed to detect and discard samples that are located far from the mean across all median distances from each sample to all other samples. In the examples below, $\tau$ was set to 3 and

$\mathfrak{E} = \{500, 200, 100\}$.

**Algorithm 2:** Sequential outliers removal.

**Input:** dataset $D$, set of bandwidths $\mathfrak{E}$, threshold factor $\tau$

**foreach** $\varepsilon \in \mathfrak{E}$ **do**
    **while** *True* **do**
        evaluate basis vectors $\Psi$ and eigenvalues $\lambda$ for dataset $D$;
        compute $d_{ij}$ for all sample pairs using Eq. (24);
        evaluate median distances $\tilde{d}_i$ for each samples $i$;
        remove samples for which the $\tilde{d}_i > \tau\mu(\tilde{d}_1, \tilde{d}_2, \ldots)$;
        **if** *no outliers found* **then**
            break
        **end**
    **end**
**end**

Fig. 5-13 shows the outcome of applying Alg. 2 to $\mathrm{Run}_L$ and $\mathrm{Run}_H$. In this figure the blue line shows the manifold dimension for the original set of samples, while the rest of the curves correspond to diffusion manifolds with outliers sequentially removed using the kernel bandwidths provided above. These results suggest negligible differences between datasets with 100 time steps and runs using twice more steps. We also observe that the choice of density model does not impact the manifold dimension for the datasets considered here.

Fig. 5-14 shows a similar set of results based on a combination of datasets both using $\rho_H$ with and without Kn number impacting the aerodynamic model. The datasets are combined using the approaches described in Eq. (22) and Eq. (23). Similar to results presented in the previous figure, these results indicate that 100 state vectors, equally spaced in time, are sufficient to obtain converged results. We note that the dimensionality $m$ of the diffusion manifold based the union of datasets corresponding to $\mathrm{Run}_H$ and $\mathrm{Run}_{H,Kn}$ is about 37 which is only slightly larger compared to the dimensionality of manifolds for individual datasets, approximately 32. However, the dimensionality of the manifold based on the joint dataset computed via Eq. (23) is much larger, approximately 105. We attribute this large difference to different initial control choices at higher altitudes motivated by stronger drag forces for $\mathrm{Run}_{H,Kn}$ compared to $\mathrm{Run}_H$. This results in strong nonlinearities introduced by Eq. (23). The alternate approach facilitates correlations across time scales thus reducing the impact of misaligned time dependencies.

Figs. 5-15 and 5-16 show the diffusion map eigenvalue spectra for the same conditions as the previous two figures and selected bandwidth values. These results show the impact of outlier removal on the sharp drop in the eigenvalue magnitude.

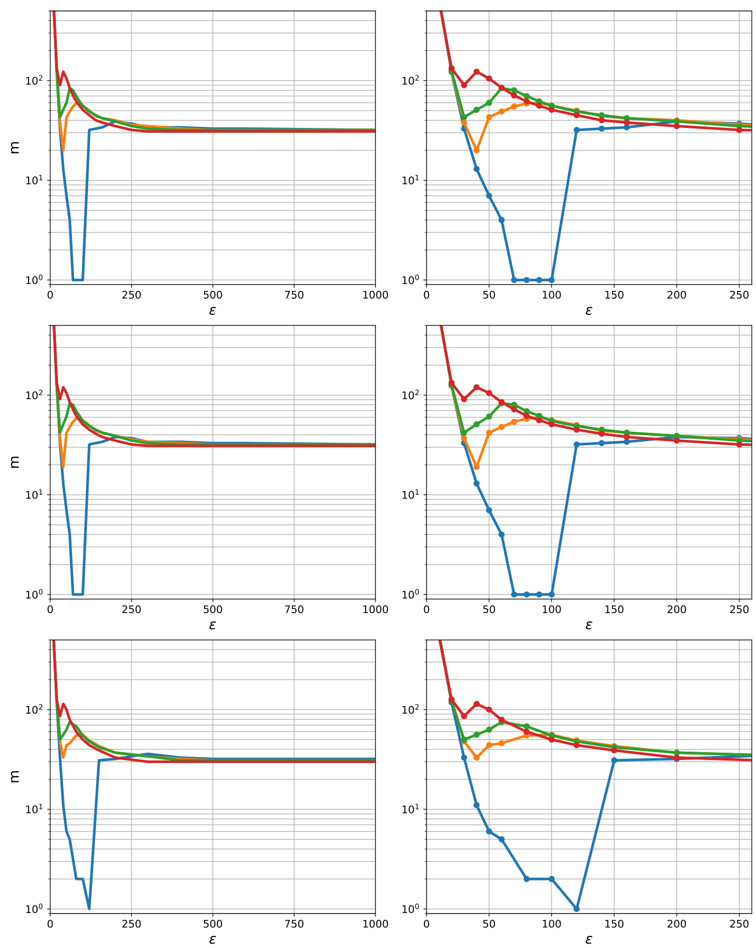


**Figure 5-13 Manifold dimension $m$ dependence on ε. Rows 1 and 2 display results for Run$_L$ with 100 and 200 state vectors, respectively, uniformly distributed along each trajectory, and row 3 displays results for Run$_H$ with 200 state vectors uniformly distributed along each trajectory. The blue, orange, green, and red lines correspond to datasets with an increasing count of outliers removed.**

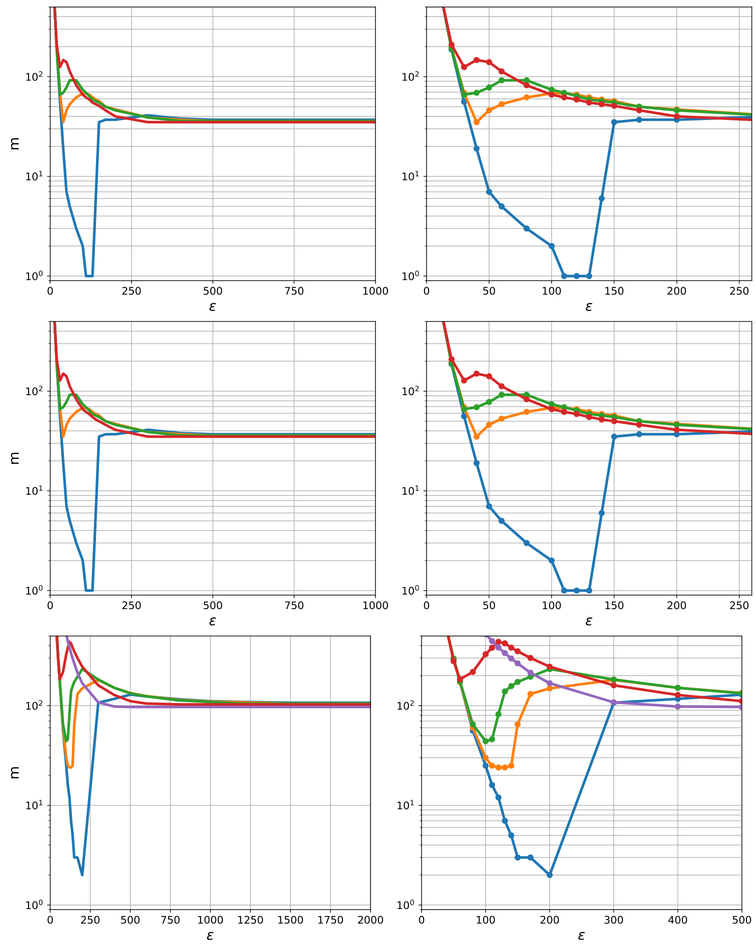


**Figure 5-14 Manifold dimension $m$ dependence on ε. Rows 1 and 2 display results for Run$_{H,Kn,\Delta_1}$ with with 100 and 200 state vectors, respectively, uniformly distributed along each trajectory, and row 3 displays results for Run$_{H,Kn,\Delta_2}$ with 200 state vectors uniformly distributed along each trajectory. The blue, orange, green, and red lines correspond to datasets with an increasing count of outliers removed.**

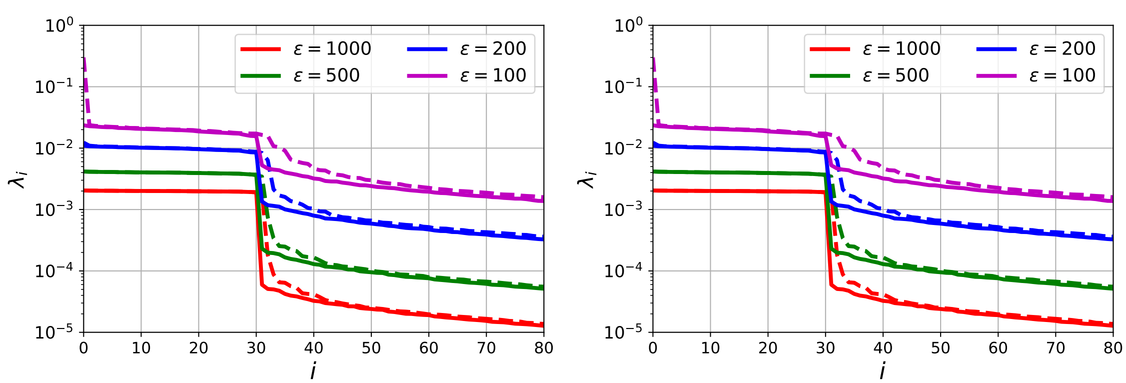


**Figure 5-15 Eigenvalues spectrum for several values for the kernel bandwidth ε: (left frame) results corresponding to Run$_L$; (right frame) results corresponding to Run$_H$. Dashed lines display results based on the original datasets, while solid lines are based on datasets with outliers/edge cases removed.**

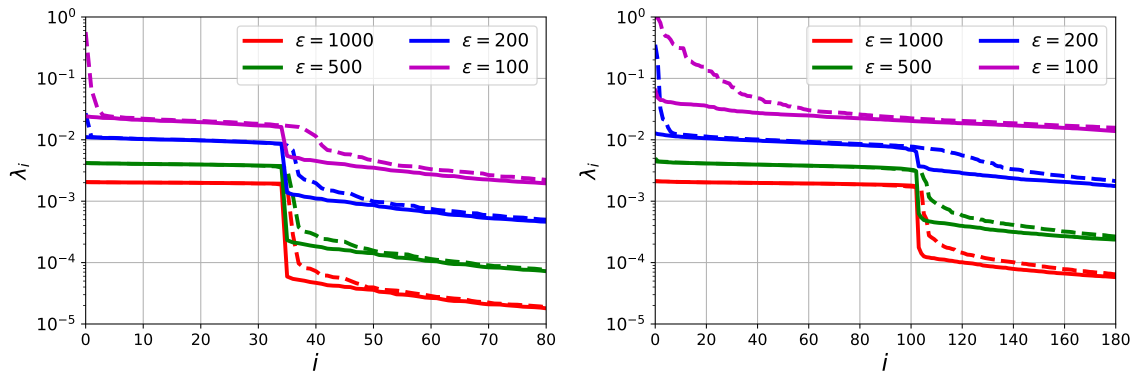


**Figure 5-16 Eigenvalues spectrum for several values for the kernel bandwidth ε: (left frame) results corresponding to Run$_{H,Kn,\Delta_1}$; (right frame) results corresponding to Run$_{H,Kn,\Delta_2}$. Dashed lines display results based on the original datasets, while solid lines are based on datasets with outliers/edge cases removed.**

# 6. CONCLUSION

This report introduces an unsupervised probabilistic learning technique for the analysis of hypersonic 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-planing 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 model considers two options for the atmospheric density dependence on altitude, as well as the effect of Knudsen number on the drag coefficient at higher altitudes. The diffusion map workflow reveals the presence of low-dimensional structures in the high-dimensional datasets. Typical manifold dimensions vary between 30 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 proposed an algorithm to automatically label and, if desired, remove outliers from the set of trajectories used for training.

We propose a novel path-planing workflow that splits the hypersonic 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 is adjusting the controls during subsequent stages to account for errors due to external factors and/or due to evolving mission objectives.

Finally, we illustrate the unsupervised learning framework in a multi-fidelity setting. We find that, for this application space, the diffusion manifold structure lives in a space only slightly higher dimensional when datasets of different fidelities are joined together compared to individual datasets. In contrast, learning the structure from datasets based on discrepancies between datasets of various fidelities leads to a much higher dimensional structure. More tests are required to verify this observation for other datasets and model configurations.

## REFERENCES


[1] U.S. Standard Atmosphere, 1976. Technical Report NASA-TM-X-74335, NASA, October 1976.

[2] John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[3] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*. Society for Industrial and Applied Mathematics, second edition, 2010.

[4] A. Busemann, N. X. Vinh, and R. D. Culp. Hypersonic flight mechanics. Technical Report NASA-CR-149170, NASA, 1976.

[5] Lin Cheng, Zhenbo Wang, Fanghua Jiang, and Junfeng Li. Fast generation of optimal asteroid landing trajectories using deep neural networks. *IEEE Transactions on Aerospace and Electronic Systems*, 56(4):2642–2655, 2020.

[6] Uihwan Choi and Jaemyung Ahn. Imitation learning-based unmanned aerial vehicle planning for multitarget reconnaissance under uncertainty. *Journal of Aerospace Information Systems*, 17(1):36–50, 2020.

[7] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[8] James W Demmel, Stanley C Eisenstat, John R Gilbert, Xiaoye S Li, and Joseph WH Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999.

[9] Fariba Fahroo and I. Michael Ross. Direct trajectory optimization by a chebyshev pseudospectral method. *Journal of Guidance, Control, and Dynamics*, 25(1):160–166, 2002.

[10] Lorenzo Federici, Boris Benedikter, and Alessandro Zavoli. Deep learning techniques for autonomous spacecraft guidance during proximity operations. *Journal of Spacecraft and Rockets*, 0(0):1–12, 2021.

[11] Casey Heidrich and Michael J. Grant. Personal communication, July 2020.

[12] Dario Izzo, Ekin Öztürk, and Marcus Märtens. 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*, page 1971–1979, New York, NY, USA, 2019. Association for Computing Machinery.

[13] Jacek Kierzenka and Lawrence F Shampine. A bvp solver based on residual control and the maltab pse. *ACM Transactions on Mathematical Software (TOMS)*, 27(3):299–316, 2001.

[14] Marco La Mantia and Lorenzo Casalino. Indirect optimization of low-thrust capture trajectories. *Journal of Guidance, Control, and Dynamics*, 29(4):1011–1014, 2006.

[15] Xiaoye S Li and James Demmel. A scalable sparse direct solver using static pivoting. In *PPSC*. Citeseer, 1999.

[16] James M Longuski, José J Guzmán, and John E Prussing. *Optimal control with aerospace applications*. Springer, 2014.

[17] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14, 2020.

[18] Kshitij Mall. *Advancing Optimal Control Theory Using Trigonometry For Solving Complex Aerospace Problems*. PhD thesis, Purdue University Graduate School, 2019.

[19] Kshitij Mall and Michael J Grant. Trigonomerization of optimal control problems with bounded controls. In *AIAA Atmospheric Flight Mechanics Conference*, page 3244, 2016.

[20] Kshitij Mall and Michael J Grant. Trigonometrization of optimal control problems with mixed constraints and linear controls. In *AIAA Scitech 2019 Forum*, page 0261, 2019.

[21] Kshitij Mall and Michael James Grant. Epsilon-trig regularization method for bang-bang optimal control problems. *Journal of Optimization Theory and Applications*, 174(2):500–517, 2017.

[22] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[23] Frank J. Regan and Satya M. Anandakrishnan. *Dynamics of atmospheric re-entry*. American Institute of Aeronautics and Astronautics, 1993.

[24] Yang Shi and Zhenbo Wang. A deep learning-based approach to real-time trajectory optimization for hypersonic vehicles. In *AIAA Scitech 2020 Forum*, 2020.

[25] Yang Shi and Zhenbo Wang. Onboard generation of optimal trajectories for hypersonic vehicles using deep learning. *Journal of Spacecraft and Rockets*, 58(2):400–414, 2021.

[26] C. Soize and R. Ghanem. Data-driven probability concentration and sampling on manifold. *Journal of Computational Physics*, 321:242–258, 2016.

[27] M. Sparapany and M. J. Grant. beluga, 2020.

[28] Michael J Sparapany. *Aerospace Mission Design on Quotient Manifolds*. PhD thesis, Purdue University Graduate School, 2020.

[29] Michael J Sparapany and Michael J Grant. Trajectory reduced-order modeling using indirect methods. In *2020 IEEE Aerospace Conference*, pages 1–10. IEEE, 2020.

[30] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 1994.

[31] Carlos Sánchez-Sánchez and Dario Izzo. Real-time optimal control via deep neural networks: Study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018.

[32] Ehsan Taheri and John L Junkins. Generic smoothing for optimal bang-off-bang spacecraft maneuvers. *Journal of Guidance, Control, and Dynamics*, 41(11):2470–2475, 2018.

[33] Zhenbo Wang and Michael J. Grant. Constrained trajectory optimization for planetary entry via sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 40(10):2603–2615, 2017.

[34] Zhenbo Wang and Michael J. Grant. Autonomous entry guidance for hypersonic vehicles by convex optimization. *Journal of Spacecraft and Rockets*, 55(4):993–1006, 2018.

[35] Chao Yan, Xiaojia Xiang, and Chang Wang. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. *Journal of Intelligent & Robotic Systems*, 98(2):297–309, 2020.

[36] Dongliang Zheng and Panagiotis Tsiotras. Near-optimal finite-time feedback controller synthesis using supervised and unsupervised learning. In *AIAA Scitech 2021 Forum*, 2021.

# APPENDIX A. CONTINUATION SCHEDULE FOR THE OPTIMAL CONTROL SOLUTION

Table A-1 displays the continuation stages setup for the solution of the OCP for the dataset #1, described in §3.1. All simulations start with a trajectory state defined by

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

and constraints $\theta_f = \phi_f = 0°$. This essentially leads to the initial trajectory pointing straight down from the initial altitude. Trajectories are then gradually adjusted to satify 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 (expressed as $\theta_f$) to $\theta_f = 0.05°$. The final downrage 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 horizonthal direction and the final downrange location is adjusted to the value setup for this set of simulated trajectories, $\theta_f = 3°$. The final crossrange location (expressed as $\phi_f$) is pushed to the desired location, $\phi_f = 2°$, 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 $\theta_{IC}$ and $\phi_{IC}$, respectively. Stage 6 (if needed) is used to adjust the density model from the low fidelity $\rho_L$ to the high-fidelity $\rho_F$. Finally, Stage 7 adjusts the path-constraint parameter $\delta$ from 0 (no constraint) to the maximum value desired for a particular set of runs, $\delta_{\max}$. These last two stages are shown in gray to indicate they are optional when generating trajectory samples. These stages are turned on/off through command-line options.

**Table A-1 Continuation stages for the optimal control problem #1.**

| Stage | Parameters adjusted | No. of steps |
|---|---|---|
| 1 | $h_f \to 0$, $\theta_f \to 0.05°$ | 21 |
| 2 | $\theta_f \to 0.5°$ | 21 |
| 3 | $\gamma_0 \to 0$, $\theta_f \to 3°$ | 41 |
| 4 | $\phi_f \to 2°$ | 41 |
| 5 | $\theta_0 \to \theta_{IC}$, $\phi_0 \to \phi_{IC}$ | 21 |
| 6 | $\rho = (1-\varepsilon)\rho_L + \varepsilon\rho_H$, $0 \to \varepsilon \to 1$ | 50 |
| 7 | $0 \to \delta \to \delta_{\max}$ | 41 |

Table A-2 displays the continuation stages setup for the solution of the optimal control solution for the dataset #2, described in §3.2. This set of runs starts with an initial estimate setup as for the previous set described above. For set #2, $y_0$ is set to

$$y_0 = \{45 \times 10^3\text{m}, \theta_0 = 0°, \phi_0 = 0°, V_{IC}, \gamma_0 = -90°, \psi_0 = 0°\}.$$

Stages 1 through 4 in Table A-2 are the same as the stages discussed above. These stages are followed by a continuation stage, shown in dark gray in the table, during which the initial height is adjusted from the initial choice of $45 \times 10^3$m to the desired initial condition $h_{IC}$ and the final crossrange and downrange values to $\theta_f \to 4°$ and $\phi_f \to 3°$, respectively. Finally, the last two stages in Table A-2 are the same as the corresponding last two stages in Table A-1 and are used to adjust the density model and the path-constraint parameter. These continuation stages are used both for simulations with the Knudsen number turned off and for simulations with Knudsen number turned on.

**Table A-2 Continuation stages for the optimal control problem #2.**

| Stage | Parameters adjusted | No. of steps |
|---|---|---|
| 1 | $h_f \to 0$, $\theta_f \to 0.05°$ | 21 |
| 2 | $\theta_f \to 0.5°$ | 21 |
| 3 | $\gamma_0 \to 0$, $\theta_f \to 3°$ | 41 |
| 4 | $\phi_f \to 2°$ | 41 |
| 5 | $\theta_0 \to \theta_{IC}$, $\phi_0 \to \phi_{IC}$ | 21 |
| 6 | $h_0 \to h_{IC}$, $\theta_f \to 4°$, $\phi_f \to 3°$ | 21 |
| 7 | $\rho = (1-\varepsilon)\rho_L + \varepsilon\rho_H$, $0 \to \varepsilon \to 1$ | 50 |
| 8 | $0 \to \delta \to \delta_{\max}$ | 41 |

## APPENDIX B. ATMOSPHERIC DENSITY MODEL

This work utilizes two models for the atmospheric density. Both models employ an exponential dependence on polynomials that are functions of altitude. The first model employs a 1$^{st}$-order polynomial

$$\rho_L = \rho_0 \exp(-h/7500), \tag{25}$$

where $h$ is expressed in meters. The second model [11] is constructed as

$$\rho_L = \rho_0 \exp\left(\sum_{i=1}^{5} h^i/c_i\right), \tag{26}$$

with coefficients given by

$$c_1 = 13317.606714653284,\ c_2 = 218603284.61942446,\ c_3 = 9502259870121.428,$$
$$c_4 = 1.0673576770076096e+18,\ c_5 = 3.7690900249546926e+23.$$

Fig. B-1 show a comparison of these models against the data provided by the standard atmosphere model [1]. The relative error results shown in the right frame, computed as

$$|\rho_M - \rho_D|/\rho_D,$$

where $\rho_M$ is either $\rho_L$ or $\rho_H$ and $\rho_D$ is the data, demonstrate a better fidelity for the higher order polynomial approximation in $\rho_H$ compared to the 1-st order polynomial approximation under the exponent in $\rho_L$.

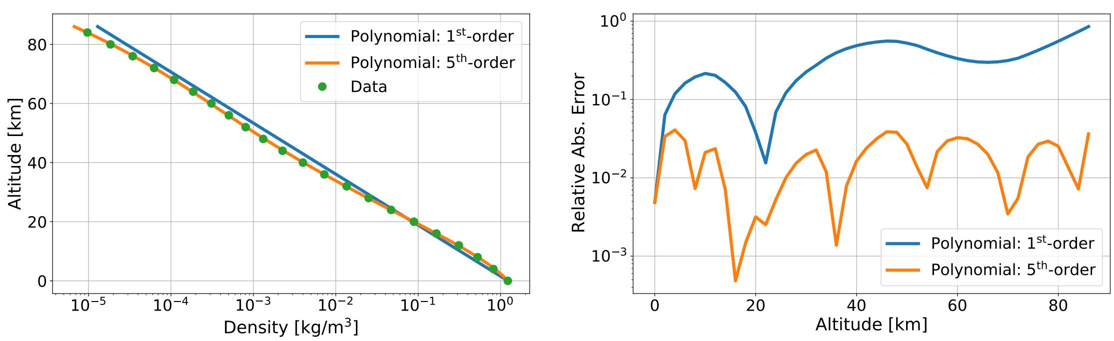


**Figure B-1 Comparison of exponential models for the atmospheric density using $1^{\text{st}}$- and $5^{\text{th}}$-order polynomial arguments: (left frame) atmospheric density vs altitude and (right frame) relative $\ell_1$ error between the models and the reference data.**

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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-NA0003525.