### Title

A new scalable algorithm for computational optimal control under uncertainty




### Authors

Lambrianides, Panos
Gong, Qi
Venturi, Daniele



### Publication Date

2020-11-01

### DOI

10.1016/j.jcp.2020.109710

Peer reviewed

# A new scalable algorithm for computational optimal control under uncertainty

Panos Lambrianides[a], Qi Gong[a], Daniele Venturi[a,*]

[a]*Department of Applied Mathematics*
*University of California Santa Cruz*
*Santa Cruz, CA 95064*

**Abstract**

We address the design of optimal control strategies for high-dimensional stochastic dynamical systems. Such systems may be deterministic nonlinear systems evolving from random initial states, or systems driven by random parameters or random noise. The objective is to provide a validated new computational capability for optimal control which will be achieved more efficiently than current state-of-the-art methods. The new framework utilizes direct single or multi-shooting discretization, and is based on efficient vectorized gradient computation with adaptable memory management. The algorithm is demonstrated to be scalable to high-dimensional nonlinear control systems with random initial condition and unknown parameters. Numerical applications are presented and discussed for stochastic path planning problems involving models of unmanned aerial and ground vehicles, and for distributed control of a nonlinear advection-reaction-diffusion equation.

## 1. Introduction

Designing optimal control strategies for high-dimensional stochastic dynamical systems is critical in many engineering applications, such as search of unknown targets with autonomous vehicles, path planning of heterogeneous agents, and quantum control [8, 15, 34, 14, 36]. Such systems may be modeled as nonlinear control systems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad t \in [0, t_f], \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the system's state (random process), $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the (deterministic) control, and $\boldsymbol{x}_0 \in \mathbb{R}^n$ is a random initial state with prescribed probability density function $p_0(\boldsymbol{x})$. As is well known, uncertain parameters in $\boldsymbol{f}$ can always be transferred to the initial condition [30, 29, 27, 4, 5]. The solution to the Cauchy problem (1) is a function of $\boldsymbol{x}_0$ and a functional of the control $\boldsymbol{u}(t)$ [21, 26]. The functional dependence will be denoted by square brackets, i.e., $\boldsymbol{x}(t) = \boldsymbol{x}(t, \boldsymbol{x}_0, [\boldsymbol{u}(t)])$. We aim at designing $\boldsymbol{u}(t)$ by solving the optimal control problem

$$\begin{cases} \min\limits_{\boldsymbol{u}(t)} J([\boldsymbol{x}(t)], [\boldsymbol{u}(t)]) \\ \quad \text{subject to:} \\ \qquad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), & \boldsymbol{x}(0) = \boldsymbol{x}_0 \quad (\boldsymbol{x}_0 \text{ random}) \\ \qquad \boldsymbol{g}(\boldsymbol{u}(t)) \leq \boldsymbol{0}, & (\text{constraints on the control}) \\ \qquad \boldsymbol{h}\left(\mathbb{E}\{\boldsymbol{x}(t)\}\right) \leq \boldsymbol{0}, & (\text{ensemble path constraints}) \end{cases} \tag{2}$$

*Corresponding author
*Email address:* venturi@ucsc.edu (Daniele Venturi)

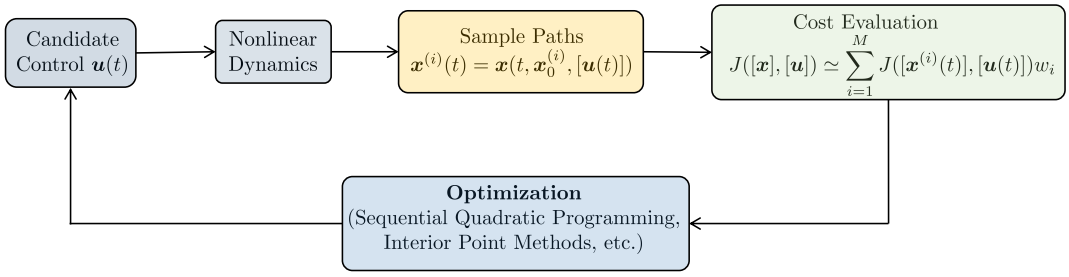



Figure 1: Sketch of the data-driven optimal control architecture. The new control algorithm is based on interior point optimization methods, common sub-expression elimination, and exact gradient information obtained with automatic differentiation and computational graphs.

where $J([\boldsymbol{x}(t)], [\boldsymbol{u}(t)])$ is a cost functional that involves an expectation $\mathbb{E}\{\cdot\}$ over the probability distribution of the initial state $\boldsymbol{x}_0$. For example, $J$ could be of the form

$$J([\boldsymbol{x}(t)], [\boldsymbol{u}(t)]) = \mathbb{E}\{F(\boldsymbol{x}(t_f))\} + \int_0^{t_f} r(\boldsymbol{u}(\tau))d\tau, \tag{3}$$

where $F(\boldsymbol{x})$ and $r(\boldsymbol{u})$ are smooth functions. Other examples of $J$ involve the probability of detecting an unknown target [15], measures of risk [22], or indicator functions of disease propagation in a network of interacting individuals [13]. The optimal control problem (2) is, in general, a high-dimensional (large-scale) *non-convex* optimization problem. State-of-the-art algorithms to solve such problem are largely based on sampling the random initial state $\boldsymbol{x}_0$, e.g., using Gauss quadrature [15, 20] or Monte Carlo methods [6, 17, 28], and then computing a large ensemble of paths corresponding to a particular control $\boldsymbol{u}(t)$. With such ensemble of paths available, one can then evaluate the performance metric $J$ by approximating the expectation operator with an appropriate cubature rule [15, 17, 33, 35], and close the optimal control loop with a suitable optimizer (see Figure 1). The formulation of optimal control strategies based on sample trajectories is straightforward, and it yields an optimal control problem which can be solved by classical algorithms, e.g., pseudospectral methods [7, 9, 10]. However, such algorithms are computationally intensive, and they are not effective even in a moderate number of state variables for the following reasons: First, the set of sample trajectories can propagate in phase space in a very complicated manner. This implies that a large number of paths is usually required to evaluate the performance metric. Secondly, as the number of sample paths increases, the dimension of the discretized optimal control problem rapidly becomes intractable. In particular, for a dynamical system with $n$ state variables and $M$ sample paths, the nonlinear control problem has dimension $nM$ ($M$ copies of the $n$-dimensional dynamical system at each iteration of the optimizer). This is one of the main bottlenecks that limits the applicability of probabilistic collocation methods to systems with low dimensional random input vectors. For systems with large number of random variables, the memory requirements and computational cost becomes unacceptably large.

In this paper, we aim at overcoming these limitations by developing a *new algorithm for computational optimal control under uncertainty* that is applicable to high-dimensional stochastic dynamical systems, and engineering control applications requiring online (real-time) implementations. The new algorithm is based on interior point optimization methods [31, 3], common sub-expression elimination, and exact gradients obtained via automatic differentiation and computational graphs [1, 2]. They key features of the proposed algorithm are:

1. *Multi-shooting optimal control schemes*: Multi-shooting is known for its numerical stability in solving deterministic optimal control problems. Based on piece-wise constant (in time) control approximation, we modified the deterministic algorithm and made it effective for uncertain optimal control

problems. This includes imposing particle-independent continuity conditions across adjacent time segments.

2. *Accurate gradient computations*: The optimization solver we developed combines an interior point method [31, 3, 37] which converges most efficiently using accurate gradient calculations over the controlled ensemble. In this work we compare a number of established techniques for gradient computation, namely traditional operator overloading algorithmic differentiation (ADOLC) [32], more modern graph based methods for gradient calculation (TensorFlow) [1] and the most commonly used finite differences approach. We demonstrate that graph based methods offer precision with considerable speed improvement over more traditional algorithms, albeit at the expense of considerable memory utilization, and slowness in initialization.
3. *Efficient memory utilization*: The algorithm has extremely low memory requirements, which means that it allows us to process a massive number of sample trajectories (in parallel) and determine the performance metric and associated optimal controls in a very efficient way. To achieve these results we developed a common sub-expression elimination (CSE) technique that can substantially reduce memory consumption during computations. As we shall see in section 3.2, CSE considerably reduces the growth of memory requirements with respect to the increase of the time discretization points in multi-shooting. This feature is *essential* for solving high dimensional problems and problems with long time horizon. It also reduces the discretization error in the control approximation, since smaller time steps can be afforded. Most importantly it does so at little to no loss to the performance of the gradient computation, and without any upfront initialization costs compared to graph based methods.

This paper is organized as follows. In section 2 we formulate the optimal control problem (2) in a fully-discrete setting by using multi-shooting methods. This yields a non-convex optimization problem with nonlinear constraints which can be solved using interior point methods. To this end, it is extremely useful to have available fast and accurate gradient information, which we address in section 3 using dataflow graphs, automatic differentiation, and common sub-expression elimination. In section 4 we develop a new criterion for verification and validation of the computed optimal controls using Pontryagin's minimum principles. In section 5 we demonstrate the accuracy and computational efficiency of the proposed control algorithms in applications to stochastic path-planning problems involving Unmanned Ground Vehicles (UGVs), Unmanned Aerial Vehicles (UAVs), and nonlinear PDEs. The main findings are summarized in section 6.

## 2. Multi-shooting schemes for ensemble optimal control

Multi-shooting optimal control algorithms have been used extensively in control of deterministic systems because of their numerical stability and straightforward implementation. In this section we describe an extension of multi-shooting to ensemble optimal control, i.e., control of ensembles of paths generated by the random dynamical system (1). The first step when discretizing (2) with a multi-shooting scheme is to divide control time horizon $[0, t_f]$ into $S$ sub-intervals, which we will call *shooting intervals*,

$$[t_k, t_{k+1}], \qquad k = 1, \ldots, S,$$

with $t_1 = 0$ and $t_{S+1} = t_f$. Within each shooting interval we introduce an evenly-spaced grid of points

$$t_{k,j} = t_k + (j-1)\Delta t_k, \qquad j = 1, 2, \ldots, N_k, \quad k = 1, \ldots, S,$$

where $\Delta t_k$ denotes the time step size used in the $k$th shooting interval $[t_k, t_{k+1}]$. Note that different shooting intervals can have different step size $\Delta t_k$, depending on the size of the temporal grid, i.e., $N_k$. The total number of time discretization points is

$$N = \sum_{k=1}^{S} N_k. \tag{4}$$

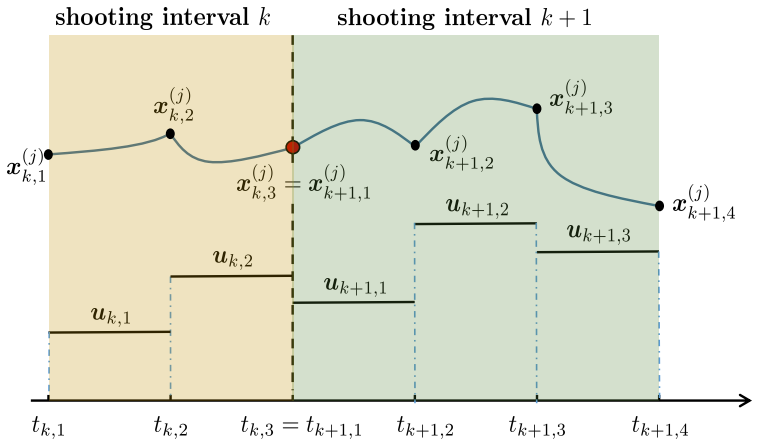



Figure 2: Sketch of the multi-shooting setting for the optimal control of one-dimensional deterministic problem. Shown are the piece-wise constant approximation of the control $\boldsymbol{u}(t)$, and the piece-wise differentiable approximation of the state $\boldsymbol{x}(t)$.

Within each shooting interval $[t_k, t_{k+1}]$, the control $\boldsymbol{u}(t)$ is approximated by a *piecewise constant function* over the grid $\{t_{k,1}, \dots, t_{k,N_k}\}$. At this point, it is convenient to introduce the following notation:

$$\boldsymbol{x}_{k,j}^{(i)} = \boldsymbol{x}^{(i)}(t_{k,j}) \qquad \boldsymbol{u}_{k,j} = \boldsymbol{u}(t_{k,j}) \tag{5}$$

for the discretized trajectory of the state vector and control, where $i = 1, \dots, M$ labels a specific realization of state process ($M$ sample paths total), $k = 1, \dots, S$, labels the shooting interval $[t_k, t_{k+1}]$, and $j = 1, \dots, N_k$ labels the time instant within the $k$th shooting interval. In Figure 2, we summarize the notation we used for the approximation/discretization of the state vector $\boldsymbol{x}(t)$ and the control $\boldsymbol{u}(t)$. Note that piece-wise constant controls yield continuous sample paths $\boldsymbol{x}^{(i)}(t)$ with cusps at $t_{k,j}$. With the piece-wise constant control approximation available, a sample of the state vector at any time instant can be computed by numerical integration as

$$\boldsymbol{x}_{k,j+1}^{(i)} \simeq \boldsymbol{x}_{k,j}^{(i)} + \int_{t_{k,j}}^{t_{k,j+1}} \boldsymbol{f}(\boldsymbol{x}^{(i)}(\tau), \boldsymbol{u}_{k,j}) d\tau, \qquad i = 1, \dots, M \tag{6}$$

$M$ being the total number of sample paths. To ensure the continuity of each path across different shooting intervals, one can impose the continuity constraints

$$\boldsymbol{x}_{k+1,1}^{(i)} = \boldsymbol{x}_{k,N_k}^{(i)}, \qquad i = 1, \dots, M. \tag{7}$$

However, this formulation introduces a very large number of optimization constraints (one for each sample path), making the discrete optimization problem very hard to solve. In fact, a large number constraints usually increases the computational cost, and the size of unfeasible regions. Therefore, instead of adding one continuity constraint for each sample path, we introduce a single constraint for each dimension and each partition, but across all sample paths.

$$\frac{1}{M} \sum_{l=1}^{M} \left( \boldsymbol{x}_{k,N_k}^{(l)} - \boldsymbol{x}_{k+1,1}^{(l)} \right)^2 = 0 \tag{8}$$

which still guarantees continuity of sample trajectories across different shooting intervals, but in a way that is *agnostic about the trajectory labels*. In other words, across different shooting intervals the constraint (8) allows for a reshuffling of the trajectories labels. As is well know, this does not affect the computation of

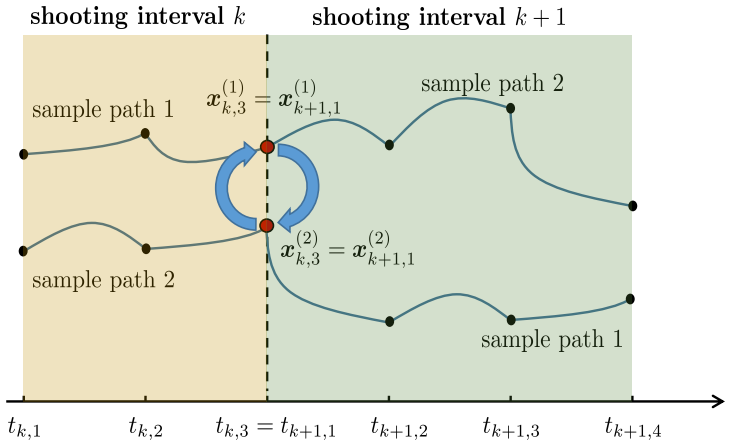



Figure 3: Continuity of sample paths across shooting intervals. The continuity constraint (8) allows us to re-label sample paths at the boundary of each shooting segment. This does not affect ensemble statistical properties at all, but it massively reduces the number of path continuity constraints (equality constraints in the optimizer) from $N$ to essentially one.

ensemble statistical properties (see Figure 3). Approximating the integral in (6), e.g., with an explicit $s$-stage Adams-Bashforth [12] method yields

$$\boldsymbol{x}^{(i)}_{k,j+1} = \boldsymbol{x}^{(i)}_{k,j} + \Delta t_k \sum_{q=0}^{s-1} b_j \boldsymbol{f}\left(\boldsymbol{x}^{(i)}_{k,j-s}, \boldsymbol{u}_{k,j-s}\right). \tag{9}$$

At this point it is convenient to denote the $i$th sample of the fully discrete state process in $[0, t_f]$ as

$$\boldsymbol{X}^{(i)} = \left\{\boldsymbol{x}^{(i)}_{1,1}, \ldots \boldsymbol{x}^{(i)}_{1,N_1}, \boldsymbol{x}^{(i)}_{2,1}, \ldots, \boldsymbol{x}^{(i)}_{2,N_2}, \ldots, \boldsymbol{x}^{(i)}_{S,1}, \ldots, \boldsymbol{x}^{(i)}_{S,N_S}\right\}, \qquad i = 1, ..., M.$$

It is also convenient to define a vector collecting the sample values the process $\boldsymbol{x}(t)$ at the boundaries of the shooting intervals, i.e.,

$$\boldsymbol{X}^{(i)}_b = \left\{\boldsymbol{x}^{(i)}_{1,1}, \boldsymbol{x}^{(i)}_{1,N_1}, \boldsymbol{x}^{(i)}_{2,1}, \boldsymbol{x}^{(i)}_{2,N_2}, \ldots, \boldsymbol{x}^{(i)}_{S,1}, \boldsymbol{x}^{(i)}_{S,N_S}\right\},$$

and the full vectors of states and boundary values

$$\boldsymbol{X} = \left\{\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(M)}\right\}, \qquad \boldsymbol{X}_b = \left\{\boldsymbol{X}^{(1)}_b, \ldots, \boldsymbol{X}^{(M)}_b\right\}. \tag{10}$$

Similarly, we denote the discrete control vector as

$$\boldsymbol{U} = \left\{\boldsymbol{u}_{1,1}, \ldots \boldsymbol{u}_{1,N_1-1}, \boldsymbol{u}_{2,1}, \ldots, \boldsymbol{u}_{2,N_2-1}, \ldots, \boldsymbol{u}_{S,1}, \ldots, \boldsymbol{u}_{S,N_S-1}\right\}. \tag{11}$$

With this notation, we can write the fully discrete form of the optimal control problem (2) as

$$\begin{cases} \min\limits_{\boldsymbol{U}, \boldsymbol{X}_b} J(\boldsymbol{U}, \boldsymbol{X}_b) \\ \quad \text{subject to:} \\ \qquad \displaystyle\sum_{k=1}^{S}\sum_{i=1}^{M} \left\| \boldsymbol{x}^{(i)}_{k,N_k} - \boldsymbol{x}^{(i)}_{k+1,1} \right\|_2^2 = 0 \qquad \text{(path continuity constraint)} \\ \qquad \boldsymbol{g}(\boldsymbol{U}) \leq \boldsymbol{0} \qquad \text{(constraints on the control)} \\ \qquad \boldsymbol{h}(\mathbb{E}\{\boldsymbol{X}\}) \leq \boldsymbol{0} \qquad \text{(ensemble path constraints)} \end{cases} \tag{12}$$

where the intra-interval dynamics defined by equation (9) is used to compute boundary values $\boldsymbol{X}_b$.

**Remark 2.1.** Equation (9) allows us to compute the discrete dynamics corresponding to each initial condition sample under the control vector (11). The solution samples can be vectorized and pushed forward in time simultaneously, in a massively parallel way.

## 3. Fast gradient computations

The large-scale constrained optimization problem (12) can be solved by using optimization algorithms based on gradient information. To this end, it is very important to be able to compute the gradient of the discretized cost function $J(\boldsymbol{U}, \boldsymbol{X}_b)$ with respect to the decision variables $(\boldsymbol{U}, \boldsymbol{X}_b)$, with accuracy and efficiency. In this section we provide technical details on how we implemented such gradient computation using automatic differentiation functions available in graph-based methods such as TensorFlow [1], and backward propagation. Our algorithm leverages the fact that forward time integration can be framed as a recurrent neural network, hence providing exact gradients at a negligible computational cost. Indeed, as we will see, all operations can be performed very efficiently on computational graphs.

### *3.1. Data-flow graphs*

Modern machine learning techniques use decentralized data graphs to map computations in different nodes of a computer cluster [1]. Edges and nodes of such graph usually represent flows of data and input-output maps (operations), respectively. The dataflow graph is pre-compiled, optimized and stored as meta-data in memory for the duration of the calculation. Using reverse automatic differentiation (backpropagation), the computational graph of the gradients with respect to a given cost function is simultaneously constructed and similarly stored. Using these two graphs one can integrate the system in time for a given control and compute gradients of the state trajectories with respect to the controls. The calculations are performed using a data event driven mechanism allowing for nearly simultaneous calculations of gradients as the dynamical system is being integrated. To illustrate the main idea, consider the follow dynamical system modeling the wheel-driving Unmanned Ground Vehicle (UGV) shown in Fig.4(a).

$$\begin{cases} \dot{x}_1 = Ru_1(t)\cos(x_3) \\ \dot{x}_2 = Ru_1(t)\sin(x_3) \\ \dot{x}_3 = Ru_2(t) \end{cases}. \tag{13}$$

Here, $(x_1(t), x_2(t))$ denotes the position of the vehicle on the Cartesian plane while $x_3(t)$ is the heading angle. The controls are $u_1(t)$ (velocity) and $u_2(t)$ (steering angle). For illustration purposes, we discretize (13) with the Euler forward scheme (one-step Adams Bashforth (9)). This yields,

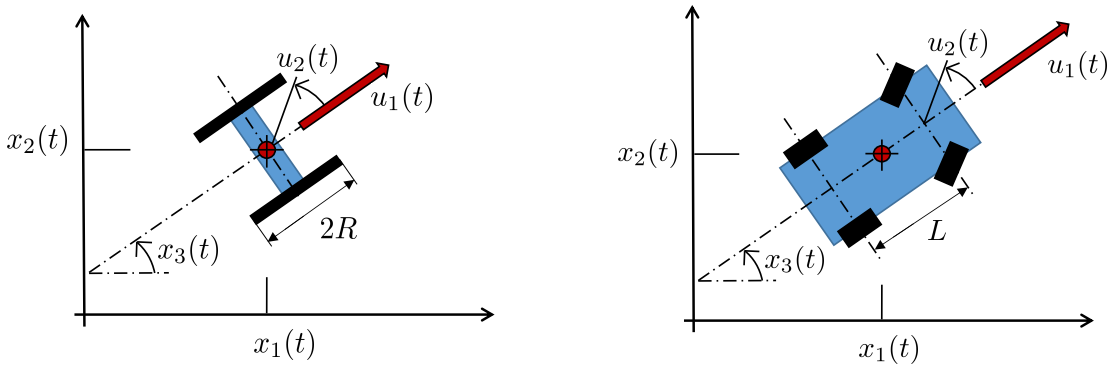
$$\begin{cases} x_1(t_{k+1}) = x_1(t_k) + \Delta t R u_1(t_k)\cos(x_3(t_k)) \\ x_2(t_{k+1}) = x_2(t_k) + \Delta t R u_1(t_k)\sin(x_3(t_k)) \\ x_3(t_{k+1}) = x_3(t_k) + \Delta t R u_2(t_k) \end{cases}. \tag{14}$$

This system of equations defines explicitly an input-output map of the form[1]

$$\boldsymbol{x}_{k+1} = \boldsymbol{Q}(\boldsymbol{x}_k, \boldsymbol{u}_k). \tag{15}$$

Such map takes in $\boldsymbol{x}_k = [x_1(t_k), x_2(t_k), x_3(t_k)]$ and $\boldsymbol{u}_k = [u_1(t_k), u_2(t_k)]$ (which we assumed constant within the time interval $\Delta t$), and sends them to $\boldsymbol{x}(t_{k+1})$. Such nonlinear map can be conveniently represented as a "black box" with inputs $\boldsymbol{x}(t_k)$ and $\boldsymbol{u}(t_k)$, and output $\boldsymbol{x}(t_{k+1})$ as illustrated in Figure 5. It is clear

[1] Explicit Runge-Kutta and linear multistep methods can always be written in the form (15) (see [19, 12, 18]).




(a) UGV modeled by the system (13).

(b) UGV modeled by the system (18).

Figure 4: Sketches of two different Unmanned Ground Vehicles (UGVs). Shown are the state space variables $\{x_i(t)\}_{i=1,2,3}$, and the controls $u_1(t)$ and $u_2(t)$.

at this point that the process of integrating the state vector $\boldsymbol{x}(t_0)$ forward in time from $t_0$ to $t_N = t_f$ for any given piece-wise constant (in time) control

$$\boldsymbol{U} = [\boldsymbol{u}(t_0), \cdots \boldsymbol{u}(t_{N-1})]$$

can be seen as a composition of the same nonlinear map (14). This essentially defines a recurrent network that allows for an extremely low memory footprint and fast gradient computation. Moreover, as clearly seen from the computational graph sketched in Figure 5, portions of the calculations for the forward trajectory propagation as well as the backward gradient calculation are shared. Using reactive programming such shared portions of the graphs are computed only once, while memoization realizes additional substantial computational savings. The exact amount of savings depends on the efficiency of the graph compiler as well as the particular dynamics. Once the compiler creates and compiles the data graph in memory for a single trajectory, adding a dimension for sampling does not involve any changes to the graph or gradient calculations. By choosing a non-adaptive integration scheme, we ensure that the operations can be performed in lock step mode across all samples, and can utilize hardware with wide SIMD capabilities such as GPU, TPU or ASIC cards.

### *3.2. Memory reduction and accelerated computations via Common Sub-expression Elimination (CSE)*

Graph based methods for gradient calculations utilized by Deep Neural Net frameworks are scalable and are implemented in a computationally efficient way. However because they are designed to operate on large system for prolonged periods of time, ranging from hours to months, they have the following disadvantages:

1. They consume an inordinate amount of memory of the order of tens of gigabytes, especially for systems that have many constraints.
2. They use lazy initialization which makes the initial ramp up time very slow, often lasting hours for high dimensional systems with long time horizons. This is acceptable for deep neural net architectures, but hampers the response time for solving high dimensional optimal control problem.
3. Because of the hefty memory and time initialization requirements, they are unsuitable for online (real-time) applications, in particular those targeting autonomous vehicles.

The main idea around common sub-expression elimination is the observation that during the construction of the computational graph in systems like Tensorflow, the memory increases linearly with the number of steps. However since the computations are identical from one time step to the next, this motivates the objective

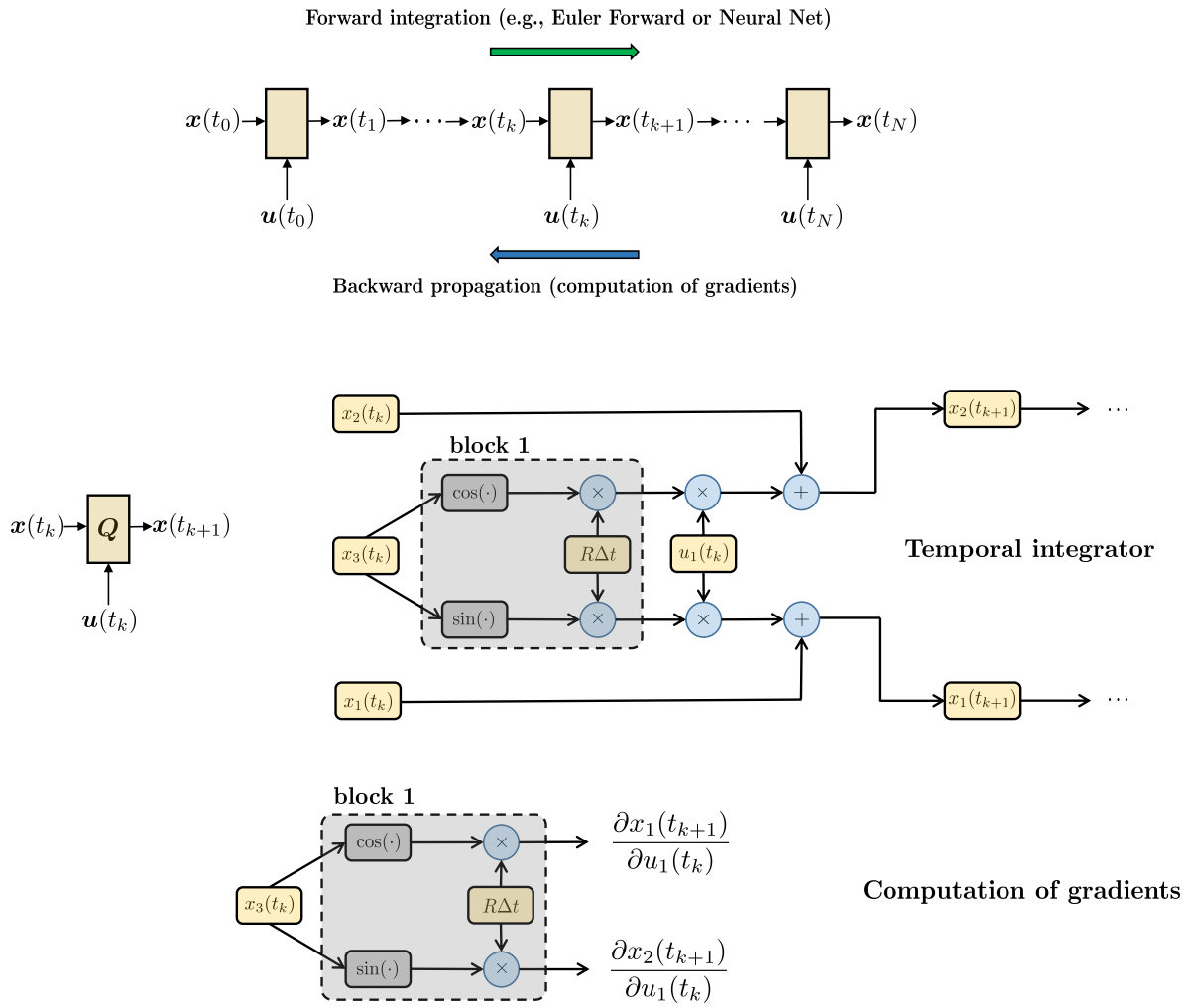



Figure 5: Portion of the data graph for the UGV model discretized with Euler forward time integration and graph the compute the kth component of the gradient of the cost functional. Note that there are sub-graphs that are shared among the computations (boxed graphs). Using reactive programming, all operations within such boxes are computed only once, and then shared across the graph.

of eliminating the common repeated calculations for each trajectory in order to save memory while not sacrificing computational speed, hence the name common sub-expression elimination.

To illustrate the basic idea of CSE in a simple way, let us consider (1) and write the time-discrete form as a symbolic nonlinear one-step recursion of the form (15). Suppose we are interested in minimizing a cost function of the form

$$J(\boldsymbol{U}) = \frac{1}{M}\sum_{i=1}^{M} F(\boldsymbol{x}^{(i)}(t_f)), \tag{16}$$

where $F$ is smooth. The gradient of (16) with respect to $\boldsymbol{u}(t_k)$ can be computed using the chain rule as

$$\frac{\partial J(\boldsymbol{U})}{\partial \boldsymbol{u}(t_k)} = \frac{1}{M}\sum_{i=1}^{M} \frac{\partial F(\boldsymbol{x}_N^{(i)})}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}_{N-1}^{(i)}, \boldsymbol{u}_{N-1})}{\partial \boldsymbol{x}} \cdots \frac{\partial \boldsymbol{Q}(\boldsymbol{x}_{k+1}^{(i)}, \boldsymbol{u}_{k+1})}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{Q}(\boldsymbol{x}_{k}^{(i)}, \boldsymbol{u}_{k})}{\partial \boldsymbol{u}}. \tag{17}$$

This is the well-know *backward propagation formula* in recurrent neural networks. Indeed, the gradient of $J$

is obtained by iteratively applying the Jacobian maps $\partial \boldsymbol{x}(t_j)/\partial \boldsymbol{x}(t_{j-1})$ from $j = N$ to $j = k+1$. Contrary to commonly used patterns in graph methods, here we need to store in memory only one gradient and two Jacobian functions, i.e.,

$$\frac{\partial F}{\partial \boldsymbol{x}}, \qquad \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}, \qquad \text{and} \qquad \frac{\partial \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}},$$

where $\boldsymbol{Q}$ is defined in (15). Even though other memory efficient schemes exist where the Jacobians are not stored explicitly, but rather the Jacobian vector product is done explicitly at each forward step, our numerical experiments indicate that storing the Jacobian explicitly strikes the best balance between computational efficiency and memory consumption. Moreover, the Jacobians for dynamical systems are of much smaller size than the Jacobians for neural nets, especially when the number of hidden layers in the latter is large, which explains why Jacobians are not stored explicitly in neural net frameworks. Taking advantage of the linear nature of the contribution of each trajectory of the ensemble to the end objective, we can leverage this linearity to customize the memory utilization through the mechanism of batching the samples. Specifically, a hundred-dimensional dynamical system over one thousand time steps requires a mere 100MB of memory per sample which is insignificant by today's standards. On a modest embedded system or a GPU with 10GB or more in memory, this allows for the simultaneous propagation of one hundred trajectories per batch using SIMD instructions. For smaller systems, much larger batches of samples can be propagated simultaneously. The specific number of optimal samples that can be propagated in a single batch needs to be determined by experimentation on the particular hardware architecture in question. The overall number of samples propagated can therefore be as large as required by the problem in question, without increase in memory footprint.

*3.2.1. Common Sub-expression Elimination: An example*

It is difficult to quantify theoretically the memory and speed of graph-based methods to compute gradients such as (17) . Hence, in this section we illustrate the memory savings and computational speedup we obtained through a series of numerical experiments. To this end, we consider the following model of wheel-driving UGV

$$\dot{x}_1 = u_1 \cos(x_3), \qquad \dot{x}_2 = u_1 \sin(x_3), \qquad \dot{x}_3 = \frac{u_1}{L} \tan(u_2), \tag{18}$$

where $(x_1(t), x_2(t))$ is the position of the vehicle, $x_3(t)$ is the heading angle, $u_1(t)$ is the forward velocity, and $u_2(t)$ is the steering angle (controls) – see Figure 4(b). We set the following box constraints on the controls

$$-1 \leq u_1(t) \leq 1, \qquad -1 \leq u_2(t) \leq 1 \qquad \forall t \in [0, t_f].$$

We integrate (18) in time from $t_0 = 0$ to $t_f = 100$ using an explicit Runge Kutta 4th order method with step size $\Delta t = 0.1$ (i.e., a total of $N = 1000$ steps) and with a fixed, randomly generated, controls $u_1$ and $u_2$. Recall that in the multi-shooting setting we consider here, the controls are assumed to be constant over each given time interval $[t_i, t_{i+1}]$ (see Figure 2). To compute the gradient (17), hereafter we consider four different approaches, i.e.,

1. Second-order centerd finite differences (FD)
2. Operator overloading algorithmic differentiation (ADOLC [32])
3. Graph-based algorithmic differentiation implemented in TensorFlow [1] (TF)
4. Common Sub-expression Elimination (CSE)

and compare them in terms of memory consumption, speed and accuracy. While it is possible to compute the gradient (17) analytically, for example using the formulas in [11] for Runge-Kutta schemes, here we

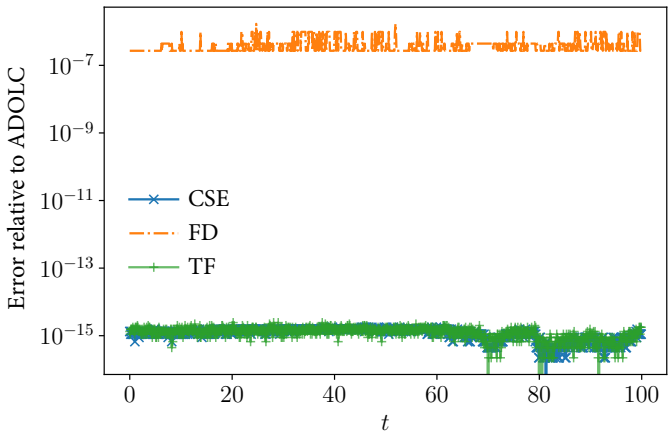



Figure 6: Absolute error of $\partial J/\partial u_1(t)$ with respect to the benchmark gradient computed with ADOLC for CSE, second-order centered finite differences (FD) and TensorFlow (TF). It is seen that tensor flow and CSE are numerically exact.

take a direct numerical approach where we use ADOLC [32], a mature algorithmic differentiation package, to provide the benchmark gradient. In this example we consider an objective that is the simple average of the UGV squared distance to the origin, namely

$$J(\boldsymbol{U}) = \frac{1}{M}\sum_{i=1}^{M}\left[x_1^{(i)}(t_f)\right]^2 + \left[x_2^{(i)}(t_f)\right]^2, \tag{19}$$

where $M$ is the number of sample paths. In Figure 6 we compare the absolute error in computing the gradient $\partial J/\partial u(t_j)$ for each $t_j$ in $[0, 100]$. It is seen that CSE and tensor flow are numerically exact, while second order (centered) finite-differences have accuracy of order $10^{-7}$, corresponds to the threshold we set. Lower precision in gradient calculations typically results in more iterations during optimization when utilizing interior point methods. Next, we compare ADOLC, second-order finite-differences, tensorflow and CSE in terms of speed and memory consumption. This is done in Figure 7 and Figure 8, respectively. All numerical experiments were conducted on dual Xeon E5-2651 v2 processors operating at 1.80GHz with 145 GB of memory. For equivalency no GPUs were used. ADOLC appears to scale best for very large samples, but it is slower for samples below $M = 10^5$ because of its lack of vectorization capabilities. The numerical experiments we present in this section suggest that CSE and ADOLC are both suitable algorithms for ensemble gradient calculations. CSE is faster and more memory efficient and can be vectorized on GPU devices, wheres ADOLC is currently restricted to run on CPUs. Tensorflow is extremely fast and suitable for solving only smaller problems because of its memory consumption and very long initialization times.

## 4. Verification and validation of optimal controls

In this section we present two criteria to verify and validate the numerical solution to the optimal control problem (2). The first one is based on the stochastic Pontryagin's minimum principle [20, 15]. To describe the method, consider the optimal control problem (2) and set

$$J([\boldsymbol{x}(t)], [\boldsymbol{u}(t)]) = \mathbb{E}\left\{F(\boldsymbol{x}(t_f))\right\} + \int_0^{t_f} r(\boldsymbol{u}(\tau))d\tau, \tag{20}$$

where $r$ and $F$ are given real-valued (smooth) functions. The second term at the right hand side is usually referred to as "control energy", and it can provide regularization of the optimization problem. Let us define

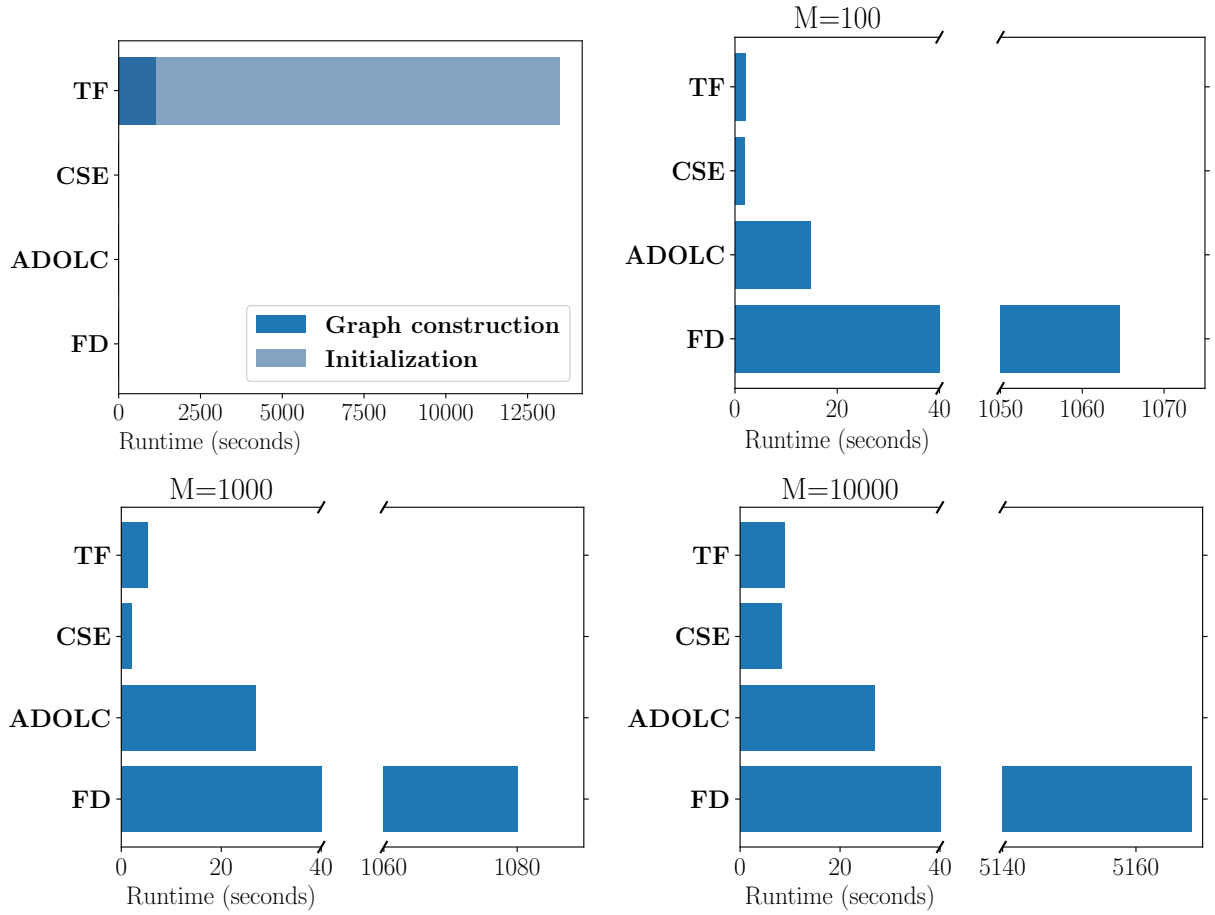



Figure 7: Execution times in seconds for different gradient methods. Here TF stands for tensorflow, and FD represents second-order finite differences. The CSE algorithm uses batch sizes of 100 samples for each thread, whereas ADOLC is parallelized one sample at a time. Tensorflow automatically optimizes its configuration for the target hardware architecture. Note the clear disadvantage in runtime that finite difference methods exhibit. All numerical experiments were conducted on a dual Xeon E5-2651 v2 processor workstation operating at 1.80GHz with 145 GB of memory.

the Hamilton's function

$$H(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\lambda}) = r(\boldsymbol{u}) + \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \tag{21}$$

From this we obtain the Hamilton's equations

$$\begin{cases} \dot{\boldsymbol{x}} = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \\ \dot{\boldsymbol{\lambda}} = -\dfrac{\partial H}{\partial \boldsymbol{x}} = -\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \cdot \boldsymbol{\lambda}. \end{cases} \tag{22}$$

As shown in [20], the system (22) is supplemented with a (random) initial condition for $\boldsymbol{x}(t)$, and a final condition for $\boldsymbol{\lambda}(t)$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \boldsymbol{\lambda}(t_f) = \frac{\partial F(\boldsymbol{x}(t_f))}{\partial \boldsymbol{x}}. \tag{23}$$

In other words, (22)-(23) is a two-point boundary value problem. Given any control $\widehat{\boldsymbol{u}}(t)$, we can solve (22)-(23) using forward-backward propagation as illustrated in Figure 9. How can we check whether $\widehat{\boldsymbol{u}}(t)$ is optimal, i.e., it solves (2) and (20)? A necessary condition is that the control satisfies the following

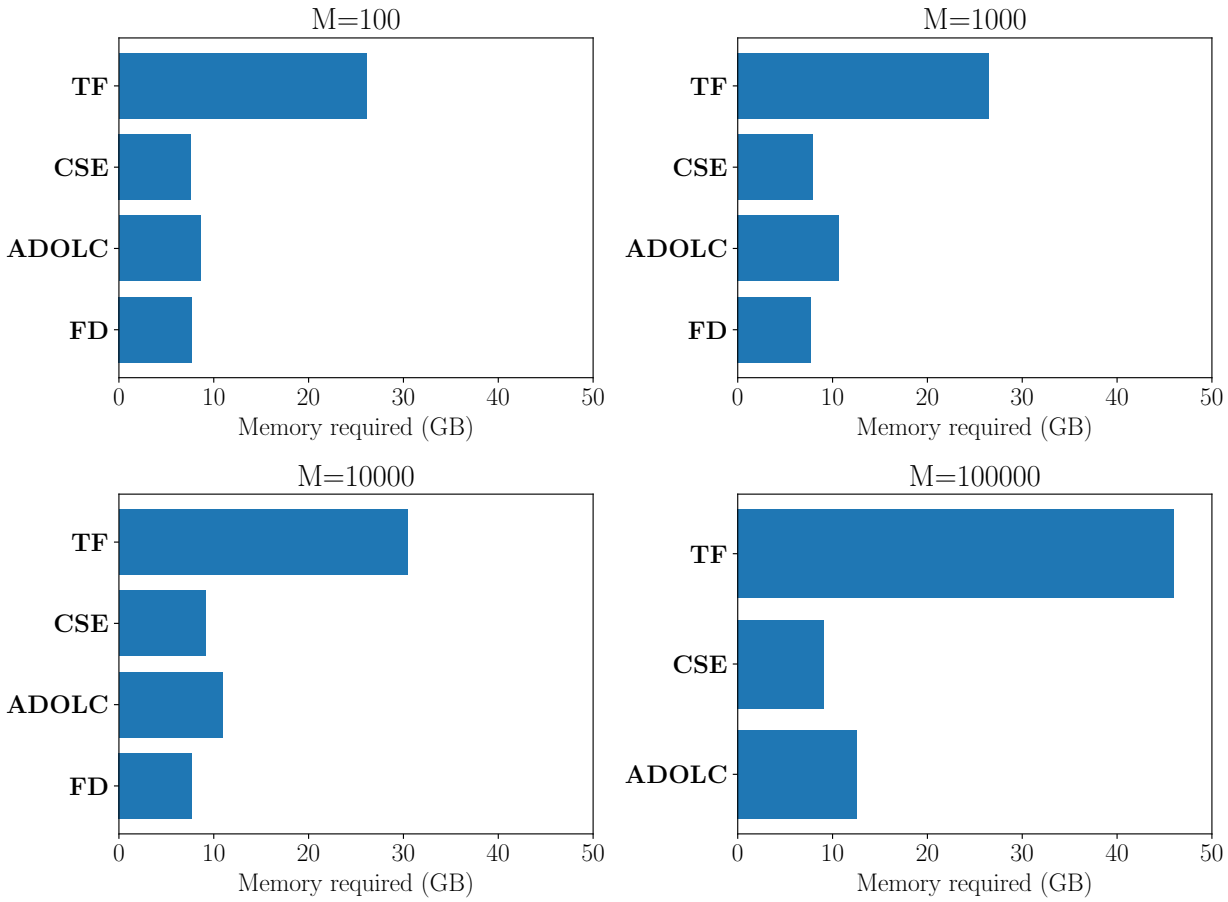



Figure 8: Memory consumption for different gradient calculation methods. Note that Tensorflow (TF) has the highest memory footprint, which grows substantially with sample size. On the other hand, CSE, ADOLC and finite-differences (FD) have fairly constant memory footprints. All numerical experiments were conducted on a dual Xeon E5-2651 v2 processor workstation operating at 1.80GHz with 145 GB of memory.

minimization principle (extended Pontryagin's principle)

$$\boldsymbol{u}^*(t) = \underset{\boldsymbol{u}(t)}{\operatorname{argmin}}\, \mathbb{E}\{H(\boldsymbol{u}(t), \boldsymbol{x}(t), \boldsymbol{\lambda}(t)\} \qquad \forall t \in [0, t_f], \tag{24}$$

subject to the Hamilton's equations (22)-(23), and the constraints $\boldsymbol{g}(\boldsymbol{u}(t)) \leq \mathbf{0}$ on the control $\boldsymbol{u}(t)$. Note that both $\boldsymbol{x}(t)$ and $\boldsymbol{\lambda}(t)$ in (24) are functionals of $\boldsymbol{u}(t)$. The Pontryagin's principle (24) offers us a simple way to verify the optimality of a given . To this end, suppose we have available a solution of (2), say $\widehat{\boldsymbol{u}}(t)$, computed using an optimizer, and we want to verify whether such solution is indeed optimal. A possible way to perform this calculation is to sample a large number of initial states $\boldsymbol{x}_0$ and propagate them through the forward/backward problem (22)-(23). This allows us to obtain many realizations of $\boldsymbol{x}^{(i)}(t)$ and $\boldsymbol{\lambda}^{(i)}(t)$ for the given candidate control $\widehat{\boldsymbol{u}}(t)$. With such realizations available, we can approximate the expectation in (24) using, e.g., Monte Carlo and compute

$$\boldsymbol{u}^*(t) = \underset{\boldsymbol{u}(t)}{\operatorname{argmin}}\, \frac{1}{M} \sum_{i=1}^{M} H\left(\boldsymbol{u}, \boldsymbol{x}^{(i)}(t), \boldsymbol{\lambda}^{(i)}(t)\right). \tag{25}$$

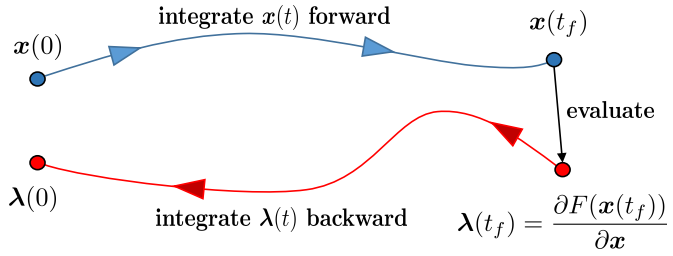



Figure 9: Schematic of the forward-backward integration process for verification and validation of optimal controls.

At this point we can compare $\boldsymbol{u}^*(t)$ with $\widehat{\boldsymbol{u}}(t)$ and verify whether the two controls are close to each other[2].

An alternative criterion for verification and validation of optimal controls was recently proposed in [17] (see also [16]). The criterion is based on the asymptotic behavior of approximate optimal controls relative to the number of sample paths, and it has advantages over the stochastic Pontryagin's principle (24) in terms of ease of implementation, especially for high-dimensional problems. To describe the method, consider the optimal control problem (2), where the performance metric (20) is replaced by the following sample average approximation

$$J_{M_k}([\boldsymbol{x}(t)],[\boldsymbol{u}(t)]) = \frac{1}{M_k}\sum_{k=1}^{M_k} F\left(\boldsymbol{x}^{(i)}(t_f)\right) + \int_0^{t_f} r(\boldsymbol{u}(\tau))d\tau. \tag{26}$$

Let $\boldsymbol{u}^{(k)}(t)$ be the control minimizing (26) for a given number of sample paths $M_k$. It was shown in [17] that the sequence $\{\boldsymbol{u}^{(k)}(t)\}$ epi-converges to a minimizer of (20) as $M_k$ goes to infinity. Based on this result, it is straightforward to generate a Cauchy sequence of optimal controls corresponding to an increasing number of sample paths. The convergence properties of such sequence can be used for verification and validation of optimal controls. For example, one could set a threshold for the distance (in some norm) between two consecutive controls, i.e., $\boldsymbol{u}^{(k+1)}(t)$ and $\boldsymbol{u}^{(k)}(t)$, corresponding to an increasing number of sample paths.

## 5. Numerical applications

In this section we demonstrate the proposed optimal control algorithms in applications to stochastic path planning problems involving models of an unmanned ground vehicles (UGVs) and a fixed-wing unmanned aerial vehicles (UAVs).

### *5.1. UGV stochastic path planning problem*

Consider the systems of equations (13), describing the dynamics of a simple two-wheeled UGV with differential drive. We would like to control the velocity $u_1(t)$ and the steering angle $u_2(t)$ of the UGV in in a way that the vehicle hits a target located at $(x_1, x_2) = (3, 3)$ at time $t_f = 10$. To this end, we first assume that the initial state of the system is deterministic, i.e.,

$$x_1(0) = 0, \qquad x_2(0) = 0, \qquad x_3(0) = 0, \qquad R = 1.25. \tag{27}$$

Moreover, we assume that the controls $u_1(t)$ and $u_2(t)$ are subject to the following constraints

$$-1 \leq u_1(t) \leq 1, \qquad -1 \leq u_2(t) \leq 1. \tag{28}$$

[2]We emphasize that the solution of (25) is *not*, in general, optimal since the expected value of the Hamiltonian is estimated using forward-backward propagation with fixed $\widehat{\boldsymbol{u}}(t)$. However, if $\|\widehat{\boldsymbol{u}}(t) - \boldsymbol{u}^*(t)\|$ is small then $\widehat{\boldsymbol{u}}(t)$ approximately satisfies the first-order necessary conditions for optimality.

We determined the optimal control for this deterministic problem by minimizing the objective functional

$$J([u_1(t), u_2(t)]) = \frac{1}{2}\left\{(x_1(t_f) - 3)^2 + (x_2(t_f) - 3)^2\right\} + \frac{q}{2}\int_0^{t_f} \left[u_1(t)^2 + u_2(t)^2\right] dt. \tag{29}$$

with respect to $u_1(t)$ and $u_2(t)$. Note that $x_1(t_f)$ and $x_2(t_f)$ are both functionals of $u_1(t)$ and $u_2(t)$. We will refer to the controls minimizing (29) subject to the dynamics (13), the initial condition (27) and the constraints (28) as the *nominal controls*. This nominal control is used as the initial guess for the stochastic version of the problem, which can be formulated as follows: find controls $\{u_1(t), u_2(t)\}$ maximizing the expectation that the UGV hits a target[3] located at $(x_1, x_2) = (3, 3)$ at time $t_f = 10$ under uncertain initial conditions and uncertain wheel radius $R$. We model such uncertainty as uniformly distributed independent random variables

$$x_1(0) \sim \mathcal{U}(-0.05, 0.05), \;\; x_2(0) \sim \mathcal{U}(-0.05, 0.05), \;\; x_3(0) \sim \mathcal{U}(-0.05, 0.05), \;\; R \sim \mathcal{U}(1, 1.5). \tag{30}$$

The stochastic version of the functional (29) can be written as

$$J([u_1(t), u_2(t)]) = \frac{1}{2}\mathbb{E}\left\{(x_1(t_f) - 3)^2 + (x_2(t_f) - 3)^2\right\} + \frac{q}{2}\int_0^{t_f} \left[u_1(t)^2 + u_2(t)^2\right] dt, \tag{31}$$

where $\mathbb{E}\{\cdot\}$ denotes an expectation over the (jointly uniform) PDF of $x_1(0)$, $x_2(0)$, $x_3(0)$, and $R$. The optimal control problem reads as follows: minimize (31) subject the dynamics (28), initial conditions (30), and control constraints (28). The integral in both (29) and (31) represents the energy of the control, and it provides a regularization of the control profiles, even for very small values of $q$. To compute the nominal and the optimal controls we utilized the multi-shooting algorithm we described in Section 2, with $S = 2$ shooting intervals, explicit RK4 time integration ($\Delta t = 0.05$), and CSE (section 3.2) to calculate the ensemble gradient of the cost functional (see Eq. (17)). The number of sample paths to approximate the expectation in (31) with a Monte Carlo cubature were set to $M = 10000$. This yields a fully discrete optimal control problem of the form (12) (with no path constraints), which we solved using IPOPT [31, 3, 37] linked with CSE and verified using Tensorflow (see Section 3). The results of our simulations are summarized in Figure 10. We see that, as expected, the final position of the UGV under nominal and optimal controls is clustered around the target located at $(x_1, x_2) = (3, 3)$. However, in the optimal control case, to reduce the variance of the set of trajectories at final time, the UGV does not head directly towards the target, but it initially heads rights and then takes a steep left turn. On the other hand, in the nominal control case, the UGV heads directly towards the target, but the set of trajectories at final time is more spread out around the target than in the optimal control case.

For verification and validation of the optimal controls, we analytically derive the Hamiltonian function for the system (13)

$$H(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\lambda}) = \frac{q}{2}\left(u_1^2 + u_2^2\right) + \lambda_1 R u_1 \cos(x_3) + \lambda_2 R u_1 \sin(x_3) + \lambda_3 R u_2. \tag{32}$$

[3] By Markov's inequality, the probability of the event $\|\boldsymbol{x}(t_f) - \boldsymbol{x}^*\| \geq \epsilon$, where $\boldsymbol{x}^*$ is the location of target, can be bounded as

$$\Pr\left(\|\boldsymbol{x}(t_f) - \boldsymbol{x}^*\| \geq \epsilon\right) \leq \frac{1}{\epsilon^2}\mathbb{E}\left\{\|\boldsymbol{x}(t_f) - \boldsymbol{x}^*\|^2\right\}.$$

Hence, if we define $\Pr\left(\|\boldsymbol{x}(t_f) - \boldsymbol{x}^*\| \geq \epsilon\right)$ as the probability of missing the target located at $\boldsymbol{x}^*$ (at time $t_f$), then minimizing the functional (31) is equivalent to minimize an upper bound on the probability of missing the target. This is not equivalent to minimize the overall probability of missing the target, as there may be different values of $\Pr\left(\|\boldsymbol{x}(t_f) - \boldsymbol{x}^*\| \geq \epsilon\right)$ that are bounded by the same quantity $\min \mathbb{E}\left\{\|\boldsymbol{x}(t_f) - \boldsymbol{x}^*\|^2\right\}/\epsilon^2$.

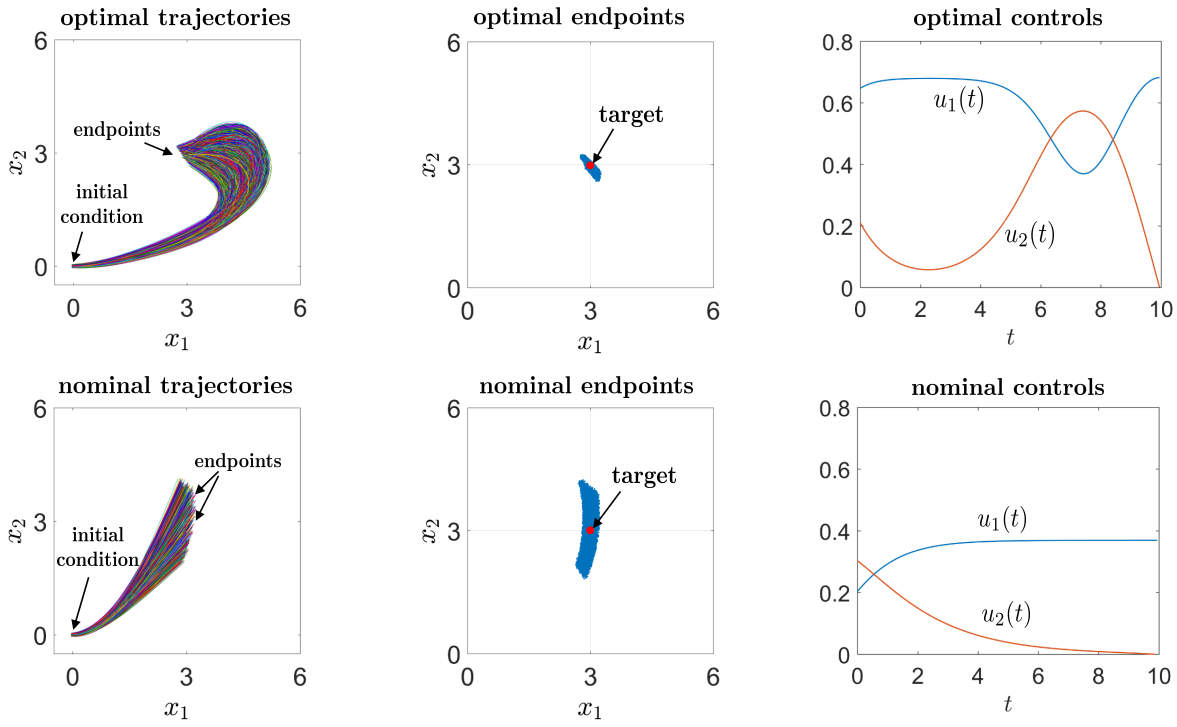



Figure 10: UGV stochastic path planning problem. Stochastic dynamics under optimal and nominal controls. It is seen that in both cases the endpoints of the UGV trajectories are clustered around the target located at $(x_1, x_2) = (3, 3)$. However, in the optimal control case, to reduce the variance of the set of trajectories at final time, the UGV does not head directly towards the target, but it initially heads rights and then takes a steep left turn. On the other hand, in the nominal control case, the UGV heads directly towards the target, but the set of trajectories at final time is more spread out around the target than in the optimal control case.

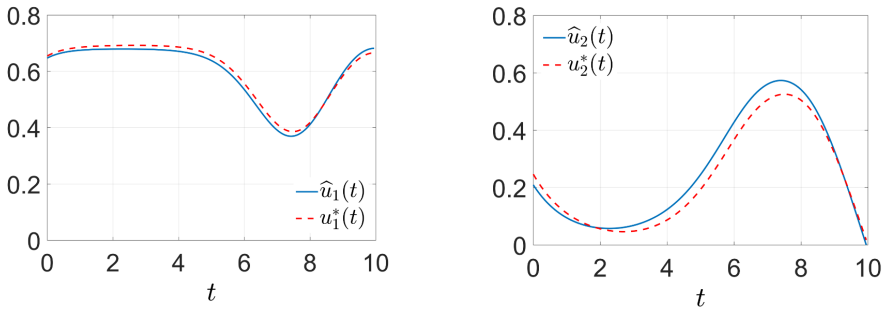



Figure 11: UGV stochastic path planning problem. Verification and Validation of the optimal control using the extended Pontryagin's minimum principle presented in Section 4.

The adjoint system is given by

$$\begin{cases} \dot{\lambda}_1 = 0, \\ \dot{\lambda}_2 = 0, \\ \dot{\lambda}_3 = \lambda_1 R u_1(t) \sin(x_3) - \lambda_2 R u_1(t) \cos(x_3). \end{cases} \tag{33}$$

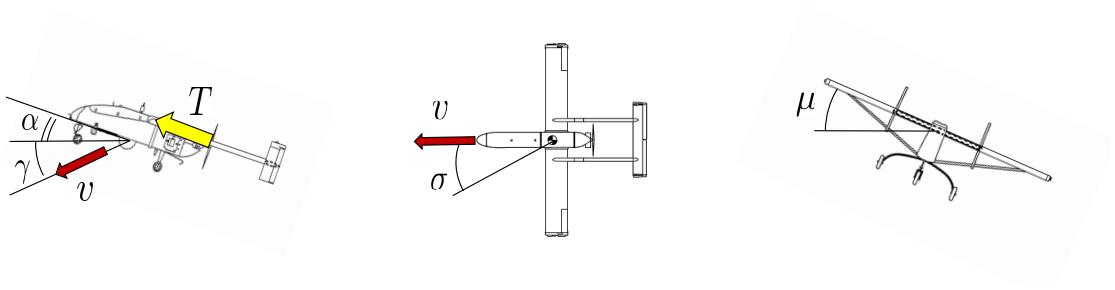



Figure 12: Sketch of the fixed wing Unmanned Aerial Vehicle (UAV). Shown are some of the phase variables appearing in the model (34). The controls act on the trust $T$, the angle of attack $\alpha$, and the bank angle $\mu$.

We follow the process described in section 4 to measure the optimality of the computed solution. This is done in Figure 11, where we compare the control $\boldsymbol{u}^*(t) = \{u_1^*(t), u_2^*(t)\}$ we obtained by solving (25) with optimal control $\widehat{\boldsymbol{u}}$ that minimizes (31). It is seen that the two controls are very close to each other, which means that $\widehat{\boldsymbol{u}}$ is a good approximation to the optimal control.

### *5.2. UAV stochastic path planning problem*

In this section we consider a more challenging problem, namely stochastic path planning for a fixed-wing unmanned aerial vehicle (UAV) model under uncertain initial state such as position, heading angle, angle of attach and aerodynamic forces. The nonlinear dynamical system modeling the UAV is [23, 24]:

$$\begin{cases} \dot{x} = v\cos\gamma\cos\sigma \\ \dot{y} = v\cos\gamma\sin\sigma \\ \dot{z} = v\sin\gamma \\ \dot{v} = \dfrac{1}{m}(-D + T\cos\alpha) - g\sin\gamma \\ \dot{\gamma} = \dfrac{1}{mv}(L\cos\mu + T\cos\mu\sin\alpha) - \dfrac{g}{v}\cos\gamma \\ \dot{\sigma} = \dfrac{1}{mv\cos\gamma}(L\sin\mu + T\sin\mu\sin\alpha) \\ \dot{T} = u_T \\ \dot{\alpha} = u_\alpha \\ \dot{\mu} = u_\mu \end{cases} \tag{34}$$

where $(x(t), y(t), z(t))$ is the position of the UAV in a Cartesian reference frame, $v(t)$ is the velocity, $(\gamma(t), \sigma(t))$ are elevation and heading angles, $T(t)$ is the thrust, $\alpha(t)$ is the angle of attack, and $\mu(t)$ is the bank angle (see Figure 12). The UAV model (34) is valid only in the region of the phase space defined by the following constraints

$$13 \le v(t) \le 42, \quad -\frac{\pi}{6} \le \gamma(t) \le \frac{\pi}{6}, \quad -\pi \le \sigma(t) \le \pi, \quad 3.0 \le T(t) \le 35.0, \quad -\frac{\pi}{8} \le \alpha(t) \le \frac{\pi}{8}. \tag{35}$$

These are effectively linear state space constraints that need to be added to the optimal control problem (see Eqs. (2) and (12)). The other parameters appearing in (34) are the UAV mass $m = 2$ kg, the acceleration of gravity $g = 9.8$ m/s$^2$, and the aerodynamic lift and drag forces given by

$$L = \frac{1}{2}\rho v^2 S C_L, \qquad D = \frac{1}{2}\rho v^2 S C_D. \tag{36}$$

In these expressions $\rho(z) = 1.21e^{-z/8000}$ kg/m$^3$ is the mass density of air, $S = 0.982$ m$^2$ is the wing surface area, and $C_L$ and $C_D$ are the lift and drag coefficients defined as

$$C_L = (C_{x0} + C_{xa}\alpha)\sin\alpha - (C_{z0} + C_{za}\alpha)\cos\alpha, \tag{37}$$

$$C_D = -(C_{x0} + C_{xa}\alpha)\cos\alpha - (C_{z0} + C_{za}\alpha)\sin\alpha. \tag{38}$$

Note that $C_L$ and $C_D$ depend on the parameters $C_{x0}, C_{xa}, C_{z0}$, and $C_{za}$ which are usually unknown and must be estimated from data. In our simulation we model such coefficients as independent random variables with given probability distributions.

We aim at computing optimal controls for the angle of attach $u_\alpha(t)$, the bank angle $u_\mu(t)$ and trust $u_T(t)$ so that the UAV hits the target located at $(x, y, z) = (500, 500, 500)$ under uncertain aerodynamic forces, uncertain initial position/angles, and uncertain initial velocity. The controls are subject to the following box constraints.

$$u_T \in [-1.0, 1.0], \qquad u_\alpha \in [-0.05, 0.05], \qquad u_\mu \in [-0.05, 0.05]. \tag{39}$$

Following the same steps as in the UGV example we studied in section 5.1, we first calculate the nominal control where we minimize the functional

$$\begin{aligned} J\left([u_T(t)], [u_\alpha(t)], u_\mu(t)]\right) =& (x(t_f) - 500)^2 + (y(t_f) - 500)^2 + (z(t_f) - 500)^2 + \\ & \frac{1}{200}\int_0^{t_f} \left[u_T^2(t) + u_\alpha^2(t) + u_\mu^2(t)\right] dt \end{aligned} \tag{40}$$

subject to (34), (35), (39) and the deterministic initial states

$$x(0) = 0, \quad y(0) = 0, \quad z(0) = 0, \quad v(0) = 27.5, \quad \gamma(0) = 0, \quad \sigma(0) = \pi, \quad T(0) = 16.1,$$
$$\alpha(0) = -0.0088, \quad \mu(0) = 0, \quad C_{x0} = -0.03554, \quad C_{xa} = 0.00292, \quad C_{z0} = -0.055, \quad C_{za} = -5.578.$$

Note that this defines a challenging maneuver whereby the UAV has to move from $(x(0), y(0), z(0)) = (0, 0, 0)$ to $(x(t_f), y(t_f), z(t_f)) = (500, 500, 500)$ while the initial heading angle $\sigma$ is in the opposite direction (see Figure 14a).

Next, we consider the stochastic path planning problem, i.e., the problem of computing robust controls that can steer the UAV from an uncertain initial state (position, velocity, and configuration angles) to the final position $(x(t_f), y(t_f), z(t_f)) = (500, 500, 500)$, under random aerodynamics forces. A simplified version of this problem was recently studied by Shaffer *et. al* in [24], where uncertainty was modeled in terms of only one random variable, i.e., $C_{x0}$ in Eqs. (37) and (38). The main reason for studying such simplified model was that the computational control algorithm could not handle higher-dimensional random input vectors. Indeed, the memory requirements and computational cost of the algorithms proposed in [24] are currently beyond the capabilities of a modern workstation. To compute the optimal controls in the stochastic path planning UAV problem we minimize the cost functional

$$\begin{aligned} J\left([u_T(t)], [u_\alpha(t)], [u_\mu(t)]\right) =& \mathbb{E}\left\{(x(t_f) - 500)^2 + (y(t_f) - 500)^2 + (z(t_f) - 500)^2\right\}\Big\} + \\ & \frac{1}{200}\int_0^{t_f} \left[u_T^2(t) + u_\alpha^2(t) + u_\mu^2(t)\right] dt, \end{aligned} \tag{41}$$

subject to (34), (35), the random initial condition

$$x(0) = r\sin(\theta)\cos(\phi), \qquad y(0) = r\sin(\theta)\sin(\phi), \qquad z(0) = r\cos(\phi), \tag{42}$$

$$v(0) \sim \mathcal{U}(25.5750, 29.4250), \qquad \gamma(0) \sim \mathcal{U}(-0.05, 0.05), \qquad \sigma(0) \sim \mathcal{U}(3.1, 3.2), \tag{43}$$

$$\begin{aligned} C_{x0} \sim \mathcal{U}(-0.0380, -0.0330), \qquad C_{xa} \sim \mathcal{U}(0.0027, 0.0031), \\ C_{z0} \sim \mathcal{U}(-0.0589, -0.0548), \qquad C_{za} \sim \mathcal{U}(-5.9685, -5.1875), \end{aligned} \tag{44}$$

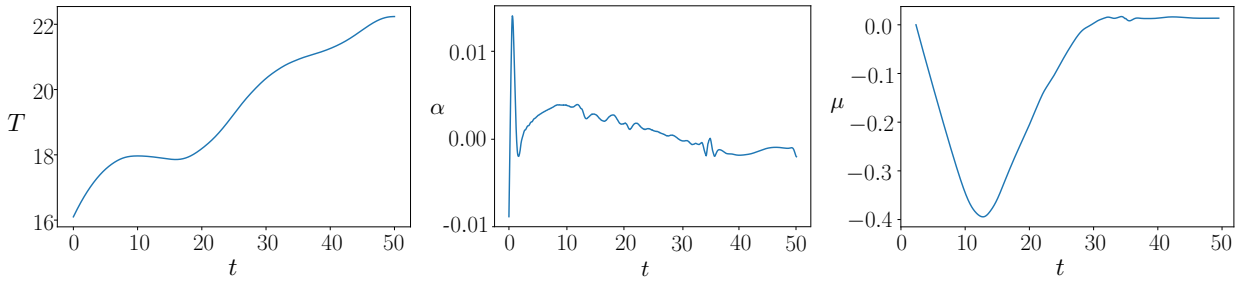



Figure 13: Stochastic path planning problem for the fixed-wing UAV model (34). Shown are the optimal thrust $T(t)$, angle of attack $\alpha(t)$ and bank angle $\mu(t)$ (integrals of $u_T$, $u_\alpha$ and $u_\mu$) we obtained to steer the UAV from the random initial position (42) to the final position $(x, y, z) = (500, 500, 500)$, under random aerodynamic forces.

where $r \sim \mathcal{U}(0, 5.0)$, $\theta \sim \mathcal{U}(0, \pi)$ $\phi \sim \mathcal{U}(0, 2\pi)$, and the ensemble path constraints

$$13 \leq \mathbb{E}\{v(t)\} \leq 42, \qquad -\frac{\pi}{6} \leq \mathbb{E}\{\gamma(t)\} \leq \frac{\pi}{6}, \qquad -\pi \leq \mathbb{E}\{\sigma(t)\} \leq \pi,$$
$$3.0 \leq \mathbb{E}\{T(t)\} \leq 35.0, \qquad -\frac{\pi}{12} \leq \mathbb{E}\{\alpha(t)\} \leq \frac{\pi}{12}.$$

In Eqs. (41) and (34) $\mathbb{E}\{\cdot\}$ is an expectation over the joint PDF of the random variables in the initial state, which we approximate using Monte Carlo cubature rule with $M = 4800$ randomly drawn sample paths. To compute the nominal and the optimal controls we utilized a single shooting algorithm with explicit RK3 time integration ($\Delta t = 0.1$), and CSE to calculate the ensemble gradient of the cost functional (see Eq. (17)). This yields a fully discrete optimal control problem of the form (12), which we solved using IPOPT [31, 3, 37] linked with CSE, and verified using ADOLC [32]. The optimal thrust, angle of attack and bank angle are shown in Figure 13. To verify the performance of the optimal controls under random initial conditions and uncertain parameters, we propagated in time the controlled dynamics of 10000 randomly generated trajectories, with initial conditions and parameters sampled according to (42)-(44). Such trajectories are shown in Figure 14. It is seen that the final positions of the UAV converge to a smaller region about the target location $(x, y, z) = (500, 500, 500)$.

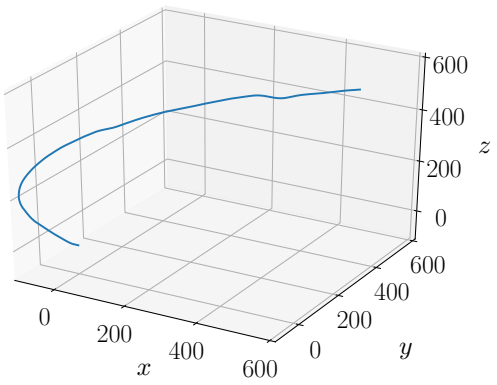
### *5.2.1. Nonlinear advection-reaction-diffusion PDE*

In this section we study open-loop control under uncertainty of nonlinear PDEs. Specifically, we consider the following initial-boundary value problem involving a nonlinear advection-reaction-diffusion equation

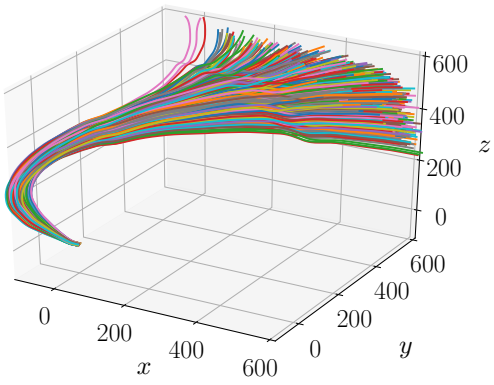
$$\begin{cases} \dfrac{\partial \psi}{\partial t} = \psi \dfrac{\partial \psi}{\partial x} + \dfrac{1}{5}\dfrac{\partial^2 \psi}{\partial x^2} + \dfrac{3}{2}\psi e^{-\psi/10} + I_\Omega(x)u(t), \\ \psi(t, -1) = \psi(t, 1) = 0, \\ \psi(0, x) = \psi_0(x), \end{cases} \tag{45}$$

where $\psi(x, t) : [-1, 1] \times [0, t_f] \mapsto \mathbb{R}$ is a random field that is a functional of $\psi_0$ (random initial condition) and $u(t) : [0, t_f] \to \mathbb{R}$ (control). Note that $u(t)$ is actuated only on the spatial domain $\Omega = [-0.5, -0.2]$, which is the support of the indicator function $I_\Omega(x)$. We aim at determining the optimal control $u(t)$ that minimizes the cost functional

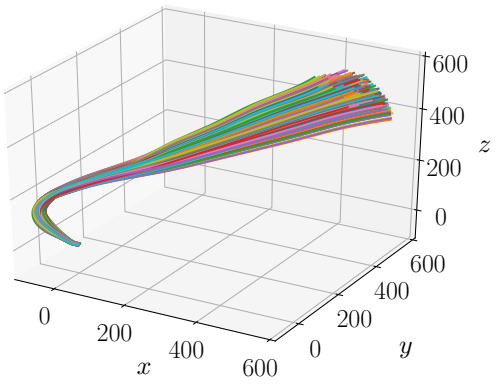
$$J([\psi], [u]) = \frac{1}{2}\mathbb{E}\left\{\int_{-1}^{1} \psi(t_f, x)^2 dx + 2\int_0^{t_f}\int_{-1}^{1} \psi(\tau, x)^2 dx d\tau\right\} + \frac{1}{20}\int_0^{t_f} u(\tau)^2 d\tau, \qquad t_f = 8, \tag{46}$$




(a) Nominal trajectory.




(b) Random trajectories under nominal controls.




(c) Random trajectories under optimal controls.

Figure 14: UAV stochastic path planning problem. Shown are trajectories obtained by sampling the random initial condition and the random parameters of the model (34) and then propagating them forward in time using nominal and optimal controls. It is seen that the optimal controls can mitigate the effects of uncertainty, and they yield final states clustered around the target located at $(x, y, z) = (500, 500, 500)$. Figures drawn to the same scale for comparison.

where $\mathbb{E}\{\cdot\}$ is an expectation over the probability distribution of the initial state $\psi_0(x)$. To this end, we first discretize (45) in space using a Chebyshev pseudo-spectral method [25]. This yields the semi-discrete form

$$\begin{cases} \dot{\boldsymbol{\psi}} = \boldsymbol{\psi} \circ \boldsymbol{D}\boldsymbol{\psi} + \dfrac{1}{5}\boldsymbol{D}^2\boldsymbol{\psi} + \dfrac{3}{2}\boldsymbol{\psi} \circ e^{-\boldsymbol{\psi}/10} + \boldsymbol{I}_\Omega u(t), \\ \boldsymbol{\psi}(0) = \boldsymbol{\psi}_0, \end{cases} \tag{47}$$

where $\boldsymbol{\psi}(t) = [\psi(t, x_1), ..., \psi(t, x_n)]^T$ collects the values of the solution $\psi(t, x)$ at the (inner) Chebyshev nodes

$$x_j = \cos\left(\frac{j\pi}{n+1}\right) \qquad j = 1, ..., n. \tag{48}$$

In equation (47), the circle "$\circ$" denotes element-wise multiplication (Hadamard product), $\boldsymbol{I}_\Omega$ is the discrete indicator function, while $\boldsymbol{D}$ and $\boldsymbol{D}^2 \in \mathbb{R}^{n\times n}$ are, respectively, the first- and second-order Chebyshev differentiation matrices. These matrices are obtained by deleting the first and last rows and columns of the full differentiation matrices. In Figure 15 we plot two solution samples of the initial-boundary value problem (45) we obtained by integrating the discretized system (47) in time. The cost functional (46) can be

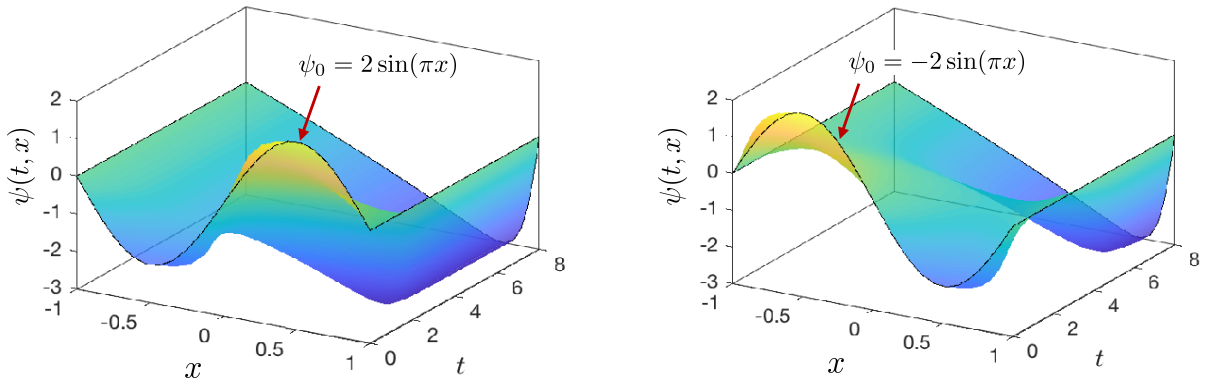



Figure 15: Solution samples of the initial-boundary value problem (45) corresponding to different initial conditions. Here we set the control $u(t)$ equal to zero (uncontrolled dynamics).

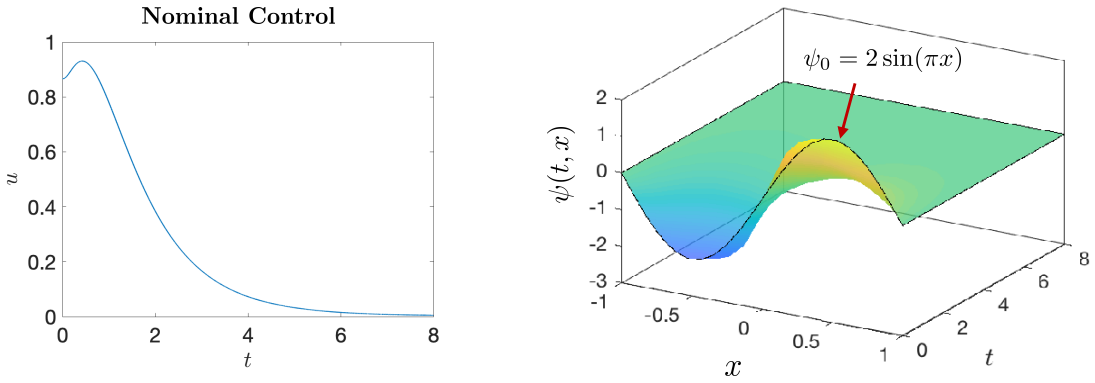



Figure 16: Open loop control of the nonlinear PDE (45). Shown is the nominal control minimizing (46) for the deterministic initial condition $\psi_0 = 2\sin(\pi x)$ and the corresponding solution dynamics. It is seen that the control $u(t)$ sends $\psi(t,x)$ to zero after a small transient.

discretized as

$$J = \frac{1}{2M}\sum_{i=1}^{M} \boldsymbol{w}^T \left[\boldsymbol{\psi}^{(i)}(t_f)^2 + 2\int_0^{t_f} \boldsymbol{\psi}^{(i)}(\tau)^2 d\tau\right] + \frac{1}{20}\int_0^{t_f} u(\tau)^2 d\tau, \tag{49}$$

where $\boldsymbol{w}$ are Clenshaw-Curtis quadrature weights (column vector), and $M$ is the total number of samples.

To quantify the effectiveness of the ensemble optimal control algorithm we propose, we first determine the nominal control corresponding to the deterministic initial condition $\psi_0(x) = 2\sin(\pi x)$. Such control minimizes the functional (49) with $M = 1$ and $\psi_j^{(1)}(0) = 2\sin(\pi x_j)$, and sends $\psi(t,x)$ to zero after a small transient (see Figure 16). Next, we introduce uncertainty in the initial condition. Specifically, we set

$$\boldsymbol{\psi}_{0j} = 2\sin(\pi x_j) + \epsilon_j \qquad \epsilon_j \sim \mathcal{N}(0, \sigma^2), \tag{50}$$

where $\epsilon_j$ are independent and identically-distributed normal random variables with mean zero and variance $\sigma^2$. This makes the solution to (45) and (47) random. As seen in Figure 17 the introduction of a small uncertainty in the initial condition, i.e., $\sigma = 10^{-3}$, causes large perturbations to the system solution under nominal control especially at $t = 8$. To compute the optimal control, we minimized the functional (49) subject to $M = 1000$ replicas of the dynamical system (47). Specifically, we considered the single-shooting

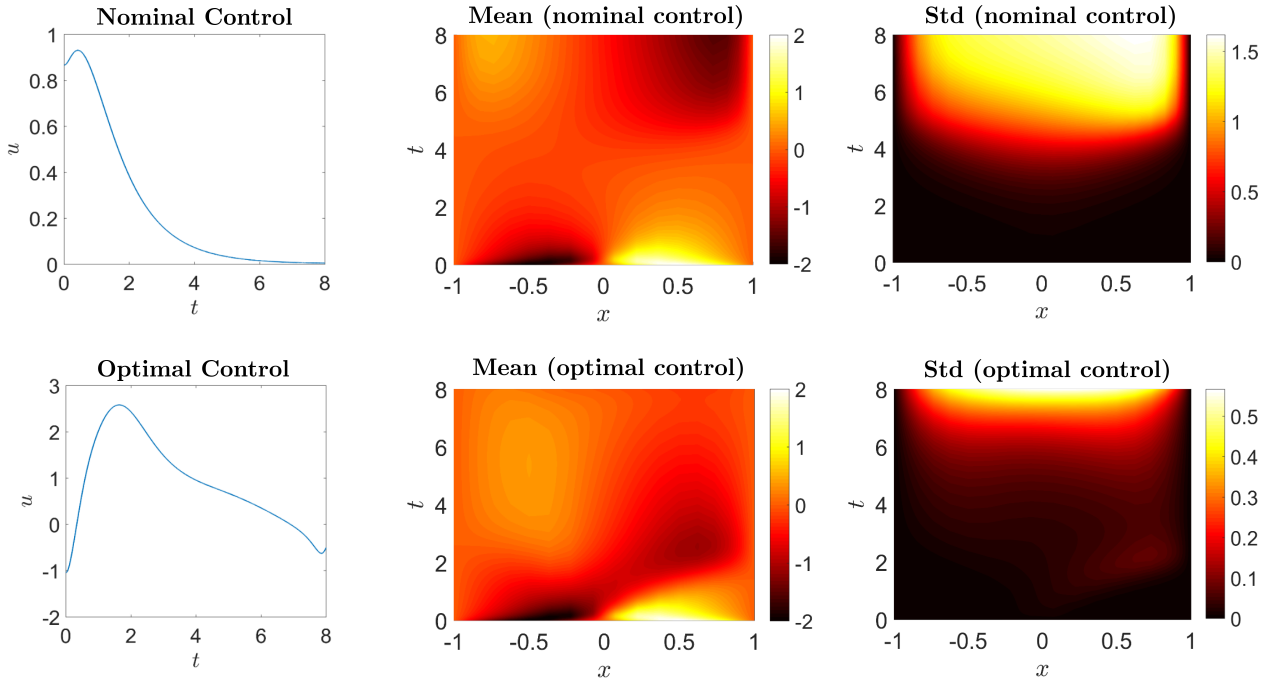



Figure 17: Comparison between the mean and the standard deviation of the stochastic solution to the PDE (45) under optimal or nominal controls. The initial condition here is set to be random as in (50), with $\sigma = 10^{-3}$. It is seen that the optimal control is more effective in driving the stochastic solution to zero. In fact, both the mean and standard deviation of the optimally controlled solution are closer to zero. Note also that the optimal control differs substantially from the nominal control.

method described in section 2 with $\Delta t = 0.0005$, Euler-forward time integrator, and CSE gradient algorithm. In Figure 17 we compare the mean and the standard deviation of the solution we obtain by using nominal and optimal controls. It is seen that the optimal control is more effective in driving the solution ensemble to zero. In fact, both the mean and standard deviation of the optimally controlled dynamics are closer to zero. Note also that the optimal control differs substantially from the nominal control. To further illustrate the performance of the method, we increase the uncertainty in (50) by several order of magnitude, i.e., we set $\sigma = 10^{-1}$ in (50). This yields the mean and standard deviation shown in Figure 18, where it is seen that the optimal control performs much better than the nominal control in driving the stochastic solution closer to zero for this increased level of uncertainty. Next, we verify the optimality of the control. Since the dimension of the control problem is relatively large, we perform such verification by using use asymptotic criterion explained at the end of section 4. To this end, we approximate the sample average in (49) using an increasing number of sample paths $M$, and compute the corresponding sequence of optimal controls. This sequence is shown in Figure 19 for $M$ ranging from 1000 to 12000. It is seen that the optimal controls converge to a well-defined control as the number of sample paths increases.

## 6. Summary

In this paper we developed a new scalable algorithm for computational optimal control under uncertainty. The new algorithm combines multi-shooting discretization methods, interior point optimization, common sub-expression elimination, and exact gradients obtained via automatic differentiation and computational graphs. It also allows for point-wise control constraints and ensemble state-space constraints. The algorithm has an extremely low memory footprint, which allows us to process a large number of sample trajectories

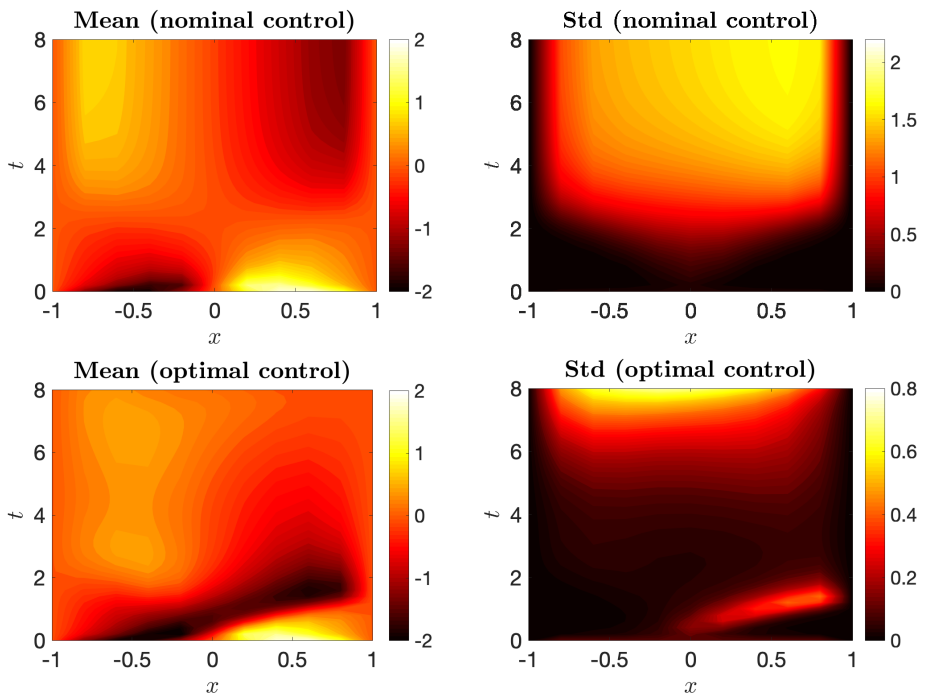



Figure 18: Comparison between the mean and the standard deviation of the stochastic solution to the PDE (45) under optimal or nominal controls for increased uncertainty. The initial condition here is set to be random as (50), with $\sigma = 10^{-1}$. It is seen that the optimal control performs much better than the nominal control in driving the stochastic solution closer to zero also for this increased level of uncertainty. The nominal control is the same as in Figure 17.

and controls random dynamical systems over long time horizons. We also developed a new criterion for verification and validation of optimal control based on the stochastic version of the Pontryagin's minimum principle. We demonstrated the accuracy and computational efficiency of the new algorithm in applications to stochastic path planning problems involving models of unmanned aerial and ground vehicles, and in distributed control of a nonlinear advection-reation-diffusion PDE.

**Acknowledgements** This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) grant FA8650-18-1-7842.

## References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, et al., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[3] L. T. Biegler and V. M. Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[4] H. Cho, D. Venturi, and G. E. Karniadakis. Adaptive discontinuous Galerkin method for response-excitation PDF equations. *SIAM J. Sci. Comput.*, 5(4):B890–B911, 2013.

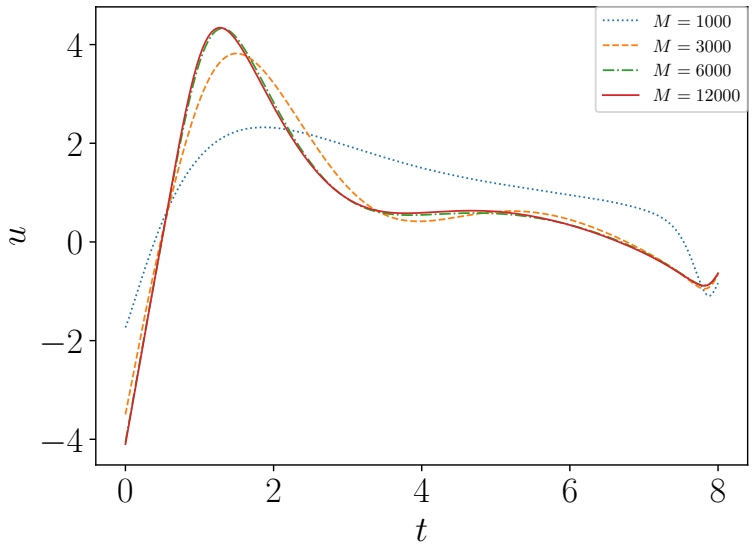



Figure 19: PDE control problem (45)-(46). Verification and Validation of the optimal control minimizing (49) using the asymptotic criterion based on the number of samples (see section 4). It is seen that the sequence of optimal controls corresponding to an increasing number of samples converges to a well-defined control.


[5] H. Cho, D. Venturi, and G. E. Karniadakis. Numerical methods for high-dimensional probability density function equation. *J. Comput. Phys*, 315:817–837, 2016.

[6] J. Dick, F. Y Kuo, and I. H. Sloan. High-dimensional integration: The quasi-Monte Carlo way. *Acta Numerica*, 22:133–288, 2013.

[7] F. Fahroo and I. M. Ross. Costate estimation by a Legendre pseudospectral method. *AIAA Journal of Guidance, Control and Dynamics*, 24(2):270–277, 2001.

[8] J. Foraker, J. O. Royset, and I. Kaminer. Search-trajectory optimization: Part I, formulation and theory. *Journal of Optimization Theory and Applications*, 169(2):530–549, 2016.

[9] Q. Gong, W. Kang, I. M. Ross, and F. Fahroo. A pseudospectral method for the optimal control of constrained feedback linearizable systems. *IEEE Trans. On Automatic Control*, 51(7):1115–1129, 2006.

[10] Q. Gong, I. M. Ross, and F. Fahroo. Spectral and pseudospectral optimal control over arbitrary grids. *Journal of Optimization Theory and Applications*, 169(3):759–783, 2016.

[11] W. W. Hager. Runge-kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282, 2000.

[12] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer, second edition, 2006.

[13] M. J. Keeling and K. T. D. Eames. Networks and epidemic models. *J. R. Soc. Interface*, 2(4):295–307, 2005.

[14] J. S. Li, J. Ruths, T. Y. Yu, H. Arthanari, and G. Wagner. Optimal pulse design in quantum control: A unified computational method. *Proceedings of the National Academy of Sciences*, 108(5):1879–1884, 2011.

[15] C. Phelps, Q. Gong, J. O. Royset, C. Walton, and I. Kaminer. Consistent approximation of a nonlinear optimal control problem with uncertain parameters. *Automatica*, 50(12):2987–2997, 2014.

[16] C. Phelps, J. O. Royset, and Q. Gong. Sample average approximations in optimal control of uncertain systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1958–1965. IEEE, 2013.

[17] C. Phelps, J. O. Royset, and Q. Gong. Optimal control of uncertain systems using sample average approximations. *SIAM Journal on Control and Optimization*, 54(1):1–29, 2016.

[18] S. C. Reddy and L. N. Trefethen. Stability of the method of lines. *Numer. Math.*, 62:235–267, 1992.

[19] A. Rodgers and D. Venturi. Stability analysis of hierarchical tensor methods for time-dependent pdes. *J. Comp. Phys*, 409:109341, 2020.

[20] I. M. Ross, R. J. Proulx, M. Karpenko, and Q. Gong. Riemann–Stieltjes optimal control problems for uncertain dynamic systems. *AIAA Journal of Guidance, Control, and Dynamics*, 38(7):1251–1263, 2015.

[21] W. J. Rugh. *Nonlinear system theory: the Volterra/Wiener approach*. Johns Hopkins University Press, 1981.

[22] A. Ruszczyński and A. Shapiro. Optimization of risk measures. In G. Calafiore and F. Dabbene, editors, *Probabilistic and randomized methods for design under uncertainty*, pages 119–157. Springer, 2006.

[23] M. Savage. Design and hardware-in-the-loop implementation of optimal canonical maneuvers for an autonomous planetary aerial vehicle. Master's thesis, Naval Postgraduate School, Monterey, California, 2012.

[24] R. Shaffer, M. Karpenko, and Q. Gong. Unscented guidance for waypoint navigation of a fixed-wing UAV. In *2016 American Control Conference (ACC)*, pages 473–478, July 2016.

[25] L. N. Trefethen. *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, 2000.

[26] D. Venturi. The numerical approximation of nonlinear functionals and functional differential equations. *Physics Reports*, 732:1–102, 2018.

[27] D. Venturi, H. Cho, and G. E. Karniadakis. The Mori-Zwanzig approach to uncertainty quantification. In R. Ghanem, D. Higdon, and H. Owhadi, editors, *Handbook of uncertainty quantification*. Springer, 2016.

[28] D. Venturi, M. Choi, and G. E. Karniadakis. Supercritical quasi-conduction states in stochastic Rayleigh-Bénard convection. *Int. J. Heat and Mass Transfer*, 55(13-14):3732–3743, 2012.

[29] D. Venturi and G. E. Karniadakis. Convolutionless Nakajima-Zwanzig equations for stochastic analysis in nonlinear dynamical systems. *Proc. R. Soc. A*, 470(2166):1–20, 2014.

[30] D. Venturi, T. P. Sapsis, H. Cho, and G. E. Karniadakis. A computable evolution equation for the joint response-excitation probability density function of stochastic dynamical systems. *Proc. R. Soc. A*, 468(2139):759–783, 2012.

[31] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[32] A. Walther, A. Griewank, and O. Vogel. ADOL-C: Automatic differentiation using operator overloading in c++. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 2, pages 41–44. Wiley Online Library, 2003.

[33] C. Walton, Q. Gong, I. Kaminer, and J. O. Royset. Optimal motion planning for searching for uncertain targets. In *the 19th World Congress of the International Federation of Automatic Control*, Cape Town, South Africa,, August 2014.

[34] C. Walton, P. Lambrianides, I. Kaminer, J. Royset, and Q. Gong. Optimal motion planning in rapid-fire combat situations with attacker uncertainty. *Naval Research Logistics (NRL)*, 65:101–119, 2018.

[35] C. Walton, C. Phelps, Q. Gong, and I. Kaminer. A numerical algorithm for optimal control of systems with parameter uncertainty. In *10th IFAC Symposium on Nonlinear Control Systems, (NOLCOS)*, Monterey, CA, August 2016.

[36] S. Wang and J. S. Li. Free-endpoint optimal control of inhomogeneous bilinear ensemble systems. *Automatica*, 95:306–315, 2018.

[37] E. Xu. Pyipopt. https://github. com/xuy/pyipopt, 2014.