Efficient High-Dimensional Stochastic Optimal Motion Control using Tensor-Train Decomposition

Alex Gorodetsky*, Sertac Karaman*, Youssef Marzouk*
*Massachusetts Institute of Technology
{goroda, sertac, ymarz}@mit.edu

***Abstract*—Stochastic optimal control problems frequently arise as motion control problems in the context of robotics. Unfortunately, all existing approaches that guarantee arbitrary precision suffer from the *curse of dimensionality*: the computational effort invested by the algorithm grows exponentially fast with increasing dimensionality of the state space of the underlying dynamic system governing the robot. In this paper, we propose a novel algorithm that utilizes compressed representations to efficiently solve stochastic optimal control problems with arbitrary precision. The running time of the new algorithms scale *linearly* with increasing dimensionality of the state space! The running time also depends polynomially on the "rank" of the value function, a measure that quantifies the intrinsic geometric complexity of the value function, due to the geometry and physics embedded in the problem instance at hand. The new algorithms are based on the recent analysis and algorithms for tensor decomposition, generalizing matrix decomposition algorithms, *e.g.*, the singular value decomposition, to three or more dimensions. In computational experiments, we show the computational effort of the new algorithm also scales linearly with the discretization resolution of the state space. We also demonstrate the new algorithm on a problem involving the perching of an aircraft, represented by a nonlinear non-holonomic longitudinal model with a seven-dimensional state space, the full numerical solution to which was not obtained before. In this example, we estimate that the proposed algorithm runs more than seven orders of magnitude faster, when compared to the naive value iteration.**

I. Introduction

Motion control problems are embedded and essential in many robotics problems. For example, consider an underwater robot subject to intense underwater currents and disturbances, tasked with docking to its station. The dynamics governing the vehicle under these conditions can be represented by an ordinary stochastic differential equation with substantial drift and diffusion terms. Then, the motion control problem is formulated by designing a suitable cost function that penalizes forbidden states and rewards successful docking [26]. Such stochastic optimal control problems are at the core of many motion control problems of robotics [3, 8, 24], especially when a robot with complex dynamics is operating in a complex environment while subject to external disturbances, such as airplanes operating under harsh weather conditions and high-speed racing cars navigating on dirt roads, among many others.

Analytical solutions are available only for a handful of stochastic optimal control problems, such as the case involving linear dynamics and Gaussian noise [10]. However, most practical instances of robot motion control problems lead to nonlinear and non-holonomic dynamics with multi-modal non-Gaussian noise characteristics; moreover, the geometry of the workspace of the robot often implies non-convex state and input constraints, to further complicate the matters [24]. Hence, computationally-efficient numerical methods are required to solve such instances of motion control problems.

Several numerical methods for this purpose were developed during the past several decades. These approaches can be divided into two categories. First, the *indirect methods* involve deriving the sufficient conditions for optimality, which leads to a boundary value problem for a special Partial Differential Equation (PDE), called the Hamilton-Jacobi-Bellman (HJB) equation. Then, the HJB equation is solved using suitable numerical methods [1, 15]. Second, the *direct methods* construct a sequence of Markov Decision Processes (MDPs), the trajectories of which converge in distribution to the trajectories of the original continuous stochastic optimal control problem. Then, the resulting MDPs are solved to obtain an approximation of the optimal policy for the original problem [13]. In both methods, the algorithms utilize a state-space discretization, often using a regular grid, which renders the number of discrete states exponential in the dimensionality of the state space. This issue is not unique to stochastic optimal control. Whenever a grid-like uniform batch sampling is utilized to construct a discrete approximation of an inherently continuous problem, similar problems arise. For example, algorithms for motion planning (for deterministic systems) or algorithms for Partially-Observable Markov Decision Processes (POMDPs) are also in this class, and they also face similar issues [24].

This computational intractability is known to be inherent in the problem. In fact, even simple special cases of motion planning (and motion control) problems are PSPACE-hard [5, 21]. Hence, in the worst case, the running time of the exact algorithms are doomed to scale exponentially with increasing number of dimensions. This phenomenon is known as the *curse of dimensionality*, and it is embedded in almost all robotics planning and control problems that involve either complex robot dynamics or complex environments [14, 3].

Our contribution in this paper is a tensor-based value iteration framework for overcoming the curse of dimensionality associated in stochastic optimal control resulting from the direct discretization using a regular grid. Our method is applicable in a general optimal stochastic control setting, hence applies to almost any regular dynamical system. The key idea is to represent the stochastic control cost function at each iteration in tensor-train format, and effectively perform value iteration

in this form. Tensor decompositions exploit *low rank* structure commonly found in *separable* functions and allow us to measure the complexity of a problem by the tensor ranks involved instead of by its dimensionality.

In essence, the tensor decomposition allows us to represent the value function in a *compressed* form, which is key to performing value iteration in an efficient manner.

The tensor-based value iteration algorithm presented in this paper exploits the properties of the tensor-train decomposition to create an algorithm which has *polynomial complexity* in dimension, *linear complexity* in the number of discretization nodes in each dimension, and *polynomial complexity* in tensor rank. Furthermore, the algorithm we employ is rank-adaptive and allows us to guarantee a specified level of accuracy for the solution of the stochastic optimal control problem. Only very recently, efficient tensor decomposition algorithms were proposed, and they were successfully utilized to overcome the curse of dimensionality in many fields, such as machine learning and data analysis [6].

We demonstrate tensor-based value iteration on two numerical simulations. The first is a proof-of-concept linear-quadratic-Gaussian control problem. In this problem we are able to match the analytic solution computed using a continuous time algebraic Riccati equation and achieve order of magnitude gains over standard value iteration. Next, we solve a perching problem and create a controller for an unpowered glider whose goal is to perch on a horizontal string. This problem involves a seven dimensional state space and a single control specifying the actions of the glider elevator. We are able to solve the problem when it is discretized in over 30 billion states on a standard 2.0 GHz Macbook Air personal computer. Such performance demonstrates the improvement of the state-of-the-art, and increases the practicality of using optimal stochastic control on a large class of problems.

II. Problem Definition and Background

In this section, we introduce the stochastic optimal control problem, and describe the tensor decomposition algorithms.

A. Stochastic optimal control

Let us denote the set of integers and the set of reals by $\mathbb{Z}$ and $\mathbb{R}$, respectively. We will denote the set of all positive real numbers by $\mathbb{R}_+$. Similarly, the set of positive integers will be denoted by $\mathbb{Z}_+$. Let $d, d_u, d_w \in \mathbb{Z}_+$, $X \subset \mathbb{R}^d$ and $U \subset \mathbb{R}^{d_u}$ be compact sets with smooth boundaries and non-empty interiors, and $\{w(t) : t \geq 0\}$ be a d_w-dimensional Brownian motion defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra, and $\mathbb{P}$ is a probability measure.

Consider a stochastic dynamical system of the following differential form:

$$dx(t) = b(x(t), u(t))dt + F(x(t), u(t))dw(t), \qquad (1)$$

where $b : X \times U \to \mathbb{R}^d$ and $F : X \times U \to \mathbb{R}^{d \times d_w}$ are measurable, continuous, and bounded functions. Strictly speaking, for any admissible control process[1] $\{u(t) : t \geq 0\}$, the solution to this differential form is a stochastic process $\{x(t) : t \geq 0\}$ that solves the following integral equation:

$$x(t) = x(0) + \int_0^t b(x(\tau), u(\tau))\, d\tau + \int_0^t F(x(\tau), u(\tau))\, dw(\tau),$$

for all t with $x(t) \in X$, where the last term on the right hand side is the usual Itô integral [17]; when the process $\{x(t) : t \geq 0\}$ hits the boundary of X, the process stops, i.e., $x(t) = x(T)$ for all $t \geq T$, where T is the time that $x(T)$ is on the boundary of S.

A *(Markov) control policy* is a mapping $\mu : X \to U$ that assigns a control input to each state. For a given policy μ, we define the *first exit time* as follows:

$$T_\mu = \inf\{t : x(t) \notin \text{int}(S) \text{ and } dx(t) = b(x(t), u(t))dt + F(x(t), u(t))dw(t), \text{ where } u(t) = \mu(x(t))\}.$$

In other words, T_μ is the first time that the trajectory of the system governed by the dynamics described by Equation (1) hits the boundary of the set X, when it is under the influence of a given control policy μ.

Then, the expected cost-to-go function under policy μ is a mapping $J : S \to \mathbb{R}$ defined as:

$$J_\mu(z) = \mathbb{E}\Big[\int_0^{T_\mu} e^{-\gamma t} g(x(t), \mu(x(t)))\, dt + h(x(T_\mu)) \;\Big|\; x(0) = z \Big], \quad (2)$$

where $0 < \gamma < 1$ is a discount factor, $g : X \times U \to \mathbb{R}$ and $h : X \to \mathbb{R}$ are measurable, continuous, and bounded functions, called the *cost rate function* and the *terminal cost function*, respectively. An *optimal cost-to-go function* $J^* : X \to \mathbb{R}$ is such that

$$J^*(z) = \inf_{\mu \in \Pi} J_\mu(z), \qquad \text{for all } z \in X.$$

An *optimal control policy* $\mu^* : X \to U$ is such that $J_{\mu^*}(z) = J^*(z)$ holds for all $z \in X$. The stochastic optimal control problem is to compute an optimal policy, given the dynamics described by Equation (1) and the cost function described by Equation (2) . This paper aims to compute approximations of the optimal cost-to-go function and the optimal control policy, for any regular system of the form given in Equation (1). For this purpose, we propose an algorithm that returns a policy μ such that $\|J_\mu - J^*\|_\infty \leq \epsilon$, for any given $\epsilon > 0$.

Numerical methods for stochastic optimal control problems are often solved by analyzing the optimality conditions proposed by Bellman (see [3]). The optimality conditions take the form of a partial differential equation, called the Hamilton-Jacobi-Bellman (HJB) equation, with boundary value conditions. The solution to this boundary value problem is the optimal cost-to-go function. Many numerical approaches to stochastic optimal control discretize and solve the boundary

[1]Suppose the control process $\{u(t) : t \geq 0\}$ is defined on the same probability space $(\Omega, \mathcal{F}, \mathbb{P})$ which the Wiener process $\{w(t) : t \geq 0\}$ is also defined on. Then, $\{u(t) : t \geq 0\}$ is said to be admissible with respect to $\{w(t) : t \geq 0\}$, if there exists a filtration $\{\mathcal{F}_t : t \geq 0\}$ defined on $(\Omega, \mathcal{F}, \mathbb{P})$ such that $u(t)$ is $\mathcal{F}_t$-adapted and $w(t)$ is an $\mathcal{F}_t$-Wiener process. See [13] for the precise measure theoretic definitions.

value problem for the HJB equation [1, 15]. This method is usually called the *indirect method*.

The *direct method* is to discretize the dynamics described by Equation (1) directly and solve the resulting discrete optimization problem, where the cost function is a discretized version of Equation (2). The resulting discrete problem is a Markov Decision Problem (MDP), which can be solved numerically using standard techniques, such as value iteration (VI) or policy iteration [13].

More precisely, an MDP is a tuple $M = (S, A, P, G, H)$, where S is the set of states, A is the set of actions, $P(\cdot|\cdot,\cdot) : S \times S \times A \to [0,1]$ is function that denotes the transition probabilities satisfying $\sum_{s'} P(s'|s,a) = 1$ for all $s \in S$ and all $a \in A$, $G : S \times A \to \mathbb{R}$ is the immediate cost function, and $H : S \to \mathbb{R}$ is the terminal cost function. The MDP describes the discrete stochastic dynamics and the cost associated with its motion. When the process is at state $s \in S$ and action $a \in A$ is applied, the next state of the process is $s' \in S$ with probability $P(s'|s,a)$. A policy is a mapping $\pi : S \to A$ that associates each state with an action. Let $\{\xi : i \in \mathbb{N}\}$ be the resulting (random) sequence of states, when action $\pi(s)$ is applied whenever the process is at state s. Then, the cost associate with motion $\{\xi : i \in \mathbb{N}\}$ is defined as:

$$\sum\nolimits_{i=0}^{N} G(\xi_i, \pi(\xi_i)) + H(\xi_N),$$

where N is the time when the process terminates. The optimal cost-to-go function and the optimal policy are defined similarly to the continuous case.

The direct method, developed by Kushner and co-workers [11, 12, 13], constructs a sequence of discrete MDPs such that the optimal cost-to-go functions of the MDPs converge uniformly to the cost-to-go functions of the original continuous-time continuous-space problem.

Let $\{M_\ell : \ell_\ell \in \mathbb{N}\}$ be a sequence of MDPs, where $M_\ell = (S_\ell, A_\ell, P_\ell, G_\ell, H_\ell)$. Suppose $S_\ell \subset X$ and $A_\ell \subset U$. Define ∂S_ℓ as the subset of S_ℓ that falls on the boundary of X. Let $\{\Delta t_\ell : \ell \in \mathbb{N}\}$ be a sequence of functions, also called *holding times*, in the following form: $\Delta t_\ell : S_\ell \to \mathbb{R}_+$. Let $\{\xi_i^n : i \in \mathbb{N}\}$ be a (random) sequence of states that describe the trajectory of M_ℓ. We use holding times as interpolation intervals to generate a continuous-time trajectory from this discrete trajectory as follows. With a slight abuse of notation, let $\xi^n : \mathbb{R}_{\geq 0} \to S$ denote the continuous-time function defined as follows: $\xi^n(\tau) = \xi_i^n$ for all $\tau \in [t_i^\ell t_{i+1}^\ell)$, where $t_i^\ell = \sum_{k=0}^{i-1} \Delta t^\ell(\xi_k)$. Let $\{u_i^\ell : i \in \mathbb{N}\}$ be a sequence of control inputs, defined for all $\ell \in \mathbb{N}$. Then, we define the continuous time interpolation of $\{u_i^\ell : i \in \mathbb{N}\}$ as $u^\ell(\tau) = u_i^\ell$ for all $\tau \in [t_i^\ell, t_{i+1}^\ell)$.

The following result by Kushner and co-workers characterizes the conditions under which the trajectories of the discrete MDPs converge to the trajectories of the original continuous-time stochastic system.

Theorem 1 (See Theorem 10.4.1 in [13])**.** *Suppose the sequence $\{M_\ell : \ell \in \mathbb{N}\}$ of MDPs and the sequence $\{\Delta t_\ell : \ell \in \mathbb{N}\}$ holding times satisfy the following conditions: For any sequence of inputs $\{u_i^\ell : i \in \mathbb{N}\}$ and the resulting sequence of trajectories $\{\xi_i^\ell : i \in \mathbb{N}\}$,*

- *for all $z \in X$,*
$$\lim_{\ell \to \infty} \Delta t_\ell(z) = 0,$$
- *for all $z \in X$ and $v \in U$,*
$$\lim_{\ell \to \infty} \frac{\mathbb{E}[\xi_{i+1}^\ell - \xi_i^\ell \,|\, \xi_i^\ell = z, u_i^\ell = v]}{\Delta t_\ell(z)} = f(z,v),$$
$$\lim_{\ell \to \infty} \frac{\mathrm{Cov}[\xi_{i+1}^\ell - \xi_i^\ell \,|\, \xi_i^\ell = z, u_i^\ell = v]}{\Delta t_\ell(z)} = F(z,v),$$

Then, the sequence $\{(\xi^\ell, u^\ell) : \ell \in \mathbb{N}\}$ of interpolations converges in distribution to (x,u) that solves the integral equation with differential form given by Equation (1)*. Let J_ℓ^* denote the optimal cost-to-go function for the MDP M_ℓ. Then, for all $z \in S$,*

$$\lim_{\ell \to \infty} |J_\ell^*(z) - J^*(z)| = 0.$$

The conditions of this theorem are often called the *local consistency conditions*. Roughly speaking, the theorem states that the trajectories of the discrete MDPs will converge to the trajectories of the original continuous-time stochastic dynamical system if the local consistency conditions are satisfied. Furthermore, in that case, the optimal cost-to-go functions of the discrete MDPs also converge to that of the original stochastic optimal control problem. A discretization that satisfies the local consistency conditions is called a *consistent discretization*.

Once a consistent discretization is obtained, the direct method is to utilize the standard dynamic programming algorithms, such as value iteration or policy iteration [2]. Suppose this discretization is obtained by choosing all states on a tensor grid with resolution h, and the resulting set of all discrete states is $\{z_{\mathbf{i}} : \mathbf{i} \in \mathcal{I}\}$. Then, for value iteration, the cost-to-go function (also called the value function) is computed in an iterative manner, by applying the following update function: For all $z_{\mathbf{i}}$,

$$J_h^{(k+1)}(z_{\mathbf{i}}) = \min_u \Big[G(z_{\mathbf{i}}, u) + \gamma_h \sum_{\mathbf{j}} P(z_{\mathbf{j}} | z_{\mathbf{i}}, u) J_h^{(k)}(z_{\mathbf{j}}) \Big], \quad (3)$$

where $0 < \gamma_h < 1$ is a discount rate. In this case, the function $J_h^{(k)}$ is the approximation of the optimal cost-to-go function (for the discrete MDP) in iteration k. The value iteration algorithm guarantees that $J_h^{(k)}$ converges to the optimal cost-to-go function for the discrete MDP as $k \to \infty$. Let us denote the limit by J_h. Then, by Theorem 1, if the discretization is consistent with the original continuous-time stochastic optimal control problem, then the J_h also converges to the optimal cost-to-go function for the continuous-time problem as $h \to 0$.

B. Low-rank tensor decompositions

Tensors are multidimensional arrays. In the context of this work, a tensor arises when a multidimensional function is evaluated on a tensor product grid of input values. For example, the cost-to-go function arising from the direct solution method

can be interpreted as a tensor when the state space is discretized according to a tensor product grid with resolution level h. The primary challenge of the direct method then becomes dealing with high dimensional tensor because of the *curse of dimensionality*. To combat the curse of dimensionality we seek to exploit the *low rank* structure and the *separability* of certain functions to obtain compressed representations of tensors. Once in compressed form, tensors can be used for computation in many algorithms.

Tensor decompositions are compressed representations of multidimensional arrays. These decompositions are multidimensional dimensional analogues of the SVD in the sense that they allow us to represent an array with far fewer elements than its total size. The complexity of algorithms dealing with such representations are dominated by the *rank* of the tensor, and they retain polynomial, and often linear, complexity with dimensions. The rank of a tensor is defined differently by the different tensor decompositions, and it may grow with problem dimension, and in this paper we use the tensor-train (TT) [18] decomposition because it is guaranteed to exist and algorithms for its computation have strong guarantees.

Separation of variables plays an important role in all tensor representations, and this fact is illuminated by first considering continuous analogues of tensor decompositions. Specifically, the continuous analogue of a tensor decomposition is a representation of a multidimensional function as a separable function, or by sums of products of one dimensional functions. The representation of a multidimensional function $f : \mathbb{R}^d \to \mathbb{R}$ in tensor-train format is

$$f(x_1, \ldots, x_d) = \sum_{\alpha_0=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} f_1^{(\alpha_0,\alpha_1)}(x_1) \ldots f_d^{(\alpha_{d-1},\alpha_d)}(x_d),$$

In the TT decomposition, there are distinct tensor ranks $r_k \in \mathbb{Z}_+$ for each dimension, with boundary conditions $r_0 = r_d = 1$. Furthermore, for each dimension $\mathcal{F}_k = \{f_k^{(\alpha_{k-1},\alpha_k)} : \alpha_{k-1} \in [1, \ldots, r_{k-1}],\ \alpha_k \in [1, \ldots, r_k]\}$ represents a set of one dimensional functions of the kth variable. The tensor-train decomposition differs from other tensor decomposition by allowing a greater variety of interaction between the functions in neighboring sets $\mathcal{F}_k, \mathcal{F}_{k+1}$. Thus, each rank r_k, for $0 < k < d$, specifies the number of functions of the kth and $k+1$th dimensional variables that interact with each other.

Transitioning back from a multidimensional function to a multidimensional array involves discretizing each dimension i into a set $\mathcal{X}_i = \{x_i[1], \ldots, x_i[n_i]\}$ of $n_i \in \mathbb{Z}_+$ points. Now let $F : \mathcal{X}_1 \times \ldots \times \mathcal{X}_d \to \mathbb{R}$ be a tensor with elements $F[i_1, \ldots, i_d] = f(x_1[i_1], \ldots, x_d[i_d])$. In the array setting, each set $\mathcal{F}_k$ can be represented as a three dimensional array F_k, called a core, where each element of each core is equal to $F_k[\alpha_{k-1}, i_k, \alpha_k] = f_k^{(\alpha_{k-1},\alpha_k)}(x_i[i_k])$. Each element of the tensor F represented in TT format can then be computed with a sequence of matrix products

$$F[i_1, i_2, \ldots, i_d] = F_1[:, i_1, :]F_2[:, i_2, :] \ldots F_d[:, i_d, :].$$

The ranks of a TT decomposition are bounded by the ranks of each unfolding matrix of the tensor. For example, rank r_k guaranteed to be not higher than the rank of the $\prod_{i=1}^{k} n_i \times \prod_{i=k+1}^{d} n_i$ unfolding matrix $F^k[i_1, \ldots, i_k; i_{k+1}, \ldots, i_d]$. The proof of this statement is constructive and results in an algorithm which allows one to decompose a tensor into its TT representation by a sequence of singular value decompositions (SVD) [18]. This algorithm also allows us to compute an approximation $\tilde{F}$ to F with a specified accuracy ϵ, i.e., we can compute an approximation such that $\left|\left|\tilde{F} - F\right|\right|_F \leq \epsilon\, ||F||$. Furthermore, this computation can be performed in $\mathcal{O}(dnr^2)$ operations, where for simplicity we let $n_k = n$ and $r_k = r$.

The SVD based compression algorithm requires access to every element of the tensor. In high dimensions, this requirement means that the TT-SVD algorithm is infeasible. To remedy this problem [19] replaces the SVD with a CUR/Skeleton/Cross approximation. Recall that the skeleton decomposition [25] of an $m \times n$ rank r matrix A can written as $A = A[:, C]A[I, C]^\dagger A[C, :]$, where I is a set of rows with $|I| \geq r$ and C is a set of columns with $|C| \geq r$.

In the context of matrices, we see that this decomposition only requires access to particular rows and columns of A. Similarly, in the TT-cross [19] algorithm, only access to particular fibers of the tensor are required. The TT-cross algorithm can also achieve ϵ-level accuracy; however, it requires the specification of upper bounds to each r_k. If the upper bounds are set too low, then errors will occur in the approximation; however, if the upper bounds are set too high then evaluation of too many elements will be required. When specified correctly, this algorithm requires evaluation of $\mathcal{O}(dnr^3)$ tensor elements. Rank-adaptive versions of this algorithm exist and can be found in [23].

Numerical computations with tensors in the TT-format can be readily performed. For example, adding two tensors with ranks $r_k = r$ results in a tensor whose ranks are $\hat{r} = 2r$ and requires virtually no operations. Element-wise multiplication results in a tensor whose ranks are $\hat{r} = r^2$ and requires $\mathcal{O}(dnr^4)$ operations. Even though the rank of a tensor grows after performing addition and multiplication, an approximation with ϵ-level accuracy often remains rank r. A procedure which "rounds" a TT decomposition to one with smaller rank with a prescribed level of accuracy ϵ is called TT-rounding and can be performed in $\mathcal{O}(dnr^2)$ operations [19].

III. Algorithm Description

We now propose a framework for tensor-based value iteration based upon representing the cost function of the stochastic control problem in tensor-train format. The key to this framework is recognizing that the direct method for solving stochastic control problems often requires a tensor product discretization of the state space. Following this discretization all functions defined on a continuous domain become represented on the discrete domain. These functions include the cost function, the policy, the drift function, and the diffusion function. Each of these functions can be interpreted as either a tensor or a vector with tensor elements. For example, the cost-to-go function (or the value function) is a tensor, and the drift function can be

interpreted as a d-dimensional vector with each element of the vector being a tensor.

One previous work has taken advantage of this tensor structure to some degree. In [9], an algorithm based on solving HJB PDEs using the canonical decomposition was proposed. However, the method was applied to particular optimization formulations which can be structured as a linear HJB, which constrains it to control affine drift and control-independent diffusion terms. We do not make any assumptions on the dynamics other than the standard regularity assumptions. Our example in Section V-B does not meet the requirements for a linear HJB directly. Hence, it can not be addressed by the method proposed in [9]. Furthermore, the canonical decomposition used in [9] does not allow existence guarantees and strong analytical results. Instead, our proposed framework is more general because it depends on the direct method, which is capable of handling a wide variety of dynamics and cost functions. In the direct method, we represent the cost function obtained at each iteration of VI in tensor-train format; therefore, we avoid evaluating the expensive optimization problem specified in Equation (3) at every state $z_{\mathbf{i}}$. While there exist approximate dynamic programming [4, 20] methods which can be also used with VI, the proposed framework has the additional feature that we can specify the error ϵ incurred.

We now describe two algorithms which utilize tensor decompositions to perform this update. The first is completely general and can be applied to a wide variety of problems including those with non-quadratic cost, non-additive control, discounted infinite horizon cost, finite horizon cost, etc. The second algorithm shows how using tensors for a specific class of stochastic optimal control problems can yield to further analytic simplifications of the value iteration algorithm.

A. Interpolation approach

The first approach relies on interpolation using `TT-cross`, and it provides a *general solution* to the problem defined by (1) and (2). As we will see, its efficiency depends upon the rank of the cost function. Suppose that $J_h^{(k)}$ is expressed in TT format and is of rank $\hat{r}$. Under such a scenario we are able to evaluate the right hand side of (3) for any $z_{\mathbf{i}}$. Thus we can view $J_h^{(k+1)}$ as a tensor whose elements can be computed by solving the corresponding minimization problem. In this case, we can use the TT-cross based interpolation procedure to build an ϵ-level approximation to $J_h^{(k+1)}$ by evaluating the right hand side of (3) only at certain fibers of $J_h^{(k+1)}$. Using these element-wise evaluations we can perform an approximate VI update to obtain a new estimate for the cost function. This procedure can be repeated each iteration at a complexity of $\mathcal{O}(dn\hat{r}^3)$ evaluations of (3) where n is typically $\mathcal{O}(1/h)$. The full tensor-based value iteration algorithm is shown in 1.

The update step 4 treats the VI update (3) as a black box function into which one feeds a state $z_{\mathbf{i}}$ and obtains an updated cost. The `TT-cross` algorithm takes as inputs a blackbox function and an accuracy parameter, and it returns the full new tensor. For any given state $z_{\mathbf{i}}$ we can then obtain the optimal control as the minimizer of (3). Both simulation examples in Section V are solved using this interpolation approach.

Algorithm 1 Tensor-based Value Iteration (TVI)

Require: Termination criterion $\delta_{\max}$; TT-cross accuracy ϵ; Initial cost function $\hat{J}_h^{(0)}$

Ensure: Residual $\delta = \|\hat{J}_h^{(k)} - \hat{J}_h^{(k-1)}\|^2 < \delta_{\max}$

1: $\delta = \delta_{\max} + 1$.
2: $k = 0$
3: **while** $\delta > \delta_{\max}$ **do**
4: $\quad \hat{J}_h^{(k+1)} = \texttt{TT-cross}\,((3), \epsilon)$
5: $\quad k = k + 1$
6: $\quad \delta = \|\hat{J}_h^{(k)} - \hat{J}_h^{(k-1)}\|^2$
7: **end while**

B. Analytical approach

Although the interpolation algorithm described above is valid for the general stochastic control problem specified by (1) and (2), certain specific problem setups offer an opportunity for avoiding interpolation at each VI step. Avoiding TT based interpolation removes the need for adaptive rank determination and can lead to more efficient computational routines. Here we outline how one can perform all the operations required in a VI update step in TT format for one example of a specific optimization problem setup. Suppose that we have a problem that has diffusion dominated dynamics, linear-additive control, quadratic state and control costs, and an invertible control penalty. Furthermore, suppose that we have obtained TT representations of both the drift and diffusion dynamics and we consider a discounted infinite horizon problem

$$\min_{u(t)} \lim_{T\to\infty} \mathbb{E}\left[\int_0^T e^{-\beta t}\left(x(t)^T\mathbf{Q}x(t) + u(t)^T\mathbf{R}u(t)\right)dt\right]$$
$$\text{s.t } dx = b(x(t))dt + \mathbf{B}u(t)dt + F(x(t))dw,$$

for discount rate $0 < \beta < 1$, state penalty $\mathbf{Q} \in \mathbb{R}^{d\times d}$, control penalty $\mathbf{R} \in \mathbb{R}^{d_u\times d_u}$, and control coefficient $\mathbf{B} \in \mathbb{R}^{d\times d_u}$. Consider now that $F(x)$ is diagonal and recall that for a tensor discretization with resolution h, diffusion dominance requires $F(x)_{ii}^2 - h\left|[\mathbf{B}u(x)]_i\right| - h\left|b(x)_i\right| > 0$ for $i = 1\ldots d$. A general LQG problem, such as the example in Section V-A, can potentially fall into this framework for a sufficiently strong diffusion.

One locally consistent discretization resulting in the optimization problem (3) is

$$p(x, x \pm e_i h\,|u\,) = \frac{F^2(x)_{ii} \pm h\left(b(x)_i + [\mathbf{B}u(x)]_i\right)}{Q(x)}$$
$$Q(x) = 2\text{tr}\left(F^2(x)\right), \quad \Delta t_h = \frac{h^2}{Q(x)}, \quad \gamma_h = e^{-\beta\Delta t_h}$$
$$G(x,u) = \Delta t_h\left(x(t)^T\mathbf{Q}x(t) + u(t)^T\mathbf{R}u(t)\right)$$

Under this discretization the control which minimizes the RHS of (3) is

$$u^*(x) = -\frac{\gamma_h}{2h}\mathbf{R}^{-1}\mathbf{B}^T\tilde{J}^{(k)}(x) \tag{4}$$

where

$$\tilde{J}^{(k)}(x) = \begin{bmatrix} J_h^{(k)}(x+e_1h) - J_h^{(k)}(x-e_1h) \\ \vdots \\ J_h^{(k)}(x+e_dh) - J_h^{(k)}(x-e_dh) \end{bmatrix}$$

Using these expressions we have the following update for the cost function

$$\begin{aligned} J_h^{(k+1)}(x) &= \Delta t_h \left(x^T \mathbf{Q} x + \frac{\gamma^2}{4h^2} \tilde{J}^{(k)}(x)^T b_u^{(k)} \right) \\ &+ \gamma_h \sum_{i=1}^{d} \left[F^2(x)_{ii} + h \left(b(x)_i - \frac{\gamma}{2h} [b_u^{(k)}]_i \right) \right] J_h^{(k)}(x+e_ih) \\ &- \gamma_h \sum_{i=1}^{d} \left[F^2(x)_{ii} - h \left(b(x)_i - \frac{\gamma}{2h} [b_u^{(k)}]_i \right) \right] J_h^{(k)}(x-e_ih), \end{aligned}$$

where $b_u^{(k)}(x) = \frac{\gamma}{2h} \mathbf{B} \mathbf{R}^{-1} \mathbf{B}^T \tilde{J}^{(k)}(x)$. Suppose that the ranks $rank(J_h^{(k)}) = \hat{r}$, and note that each element of $b_u^{(k)}$ is actually a tensor and comes from a linear combination of the elements of $\tilde{J}^{(k)}$. Adding tensors in TT format requires no operations but results in tensors with ranks which are the sum of the ranks of each element of the sum. To remove this rank growth, rounding is required after every addition, and the overall cost to compute the entire vector $b_u^{(k)}(x)$ is $\mathcal{O}(n\hat{r}^3 d^3)$, with the result typically also being rank $\hat{r}$. Here, $\tilde{J}^{(k)}(x)$ may be thought of as a vector of d dimensional tensors, and if we already have $J_h^{(k)}$ in TT format, the computation of each element of $\tilde{J}^{(k)}(x)$ only requires the manipulation of the corresponding core of $J_h^{(k)}$. Therefore,the computation of $\tilde{J}^{(k)}(x)$ requires $\mathcal{O}(dn\hat{r}^2)$ operations. After this process, the ranks of $\tilde{J}^{(k)}(x)$ remain equal to $\hat{r}$. Finally, the rest of the operations involved in the update are either addition or element-wise multiplication, and their complexity has been stated previously. Thus, we see that each update can be performed using algebraic operations in TT-format with a controlled error tolerance.

IV. Analysis

Let $\hat{J}_h^{(k)}$ denote the approximation of the cost-to-go function in the TT format obtained after k iterations using a grid with resolution h. Recall that $J_{u_h^*}$ denotes the optimal cost-to-go function with the same resolution. Our main theoretical result bounds the error between these quantities and the computational effort invested by the proposed algorithms.

Theorem 2. *When the proposed interpolation method are run for k iterations on a grid with resolution h, and TT-cross accuracy ϵ, the number of computational operations performed by the algorithm is $O(kdr^3/h)$ and the resulting approximation error satisfies:*

$$||\hat{J}_h^{(k)} - J_{u_h^*}|| \leq \epsilon \left(\tfrac{R_{max}+\gamma}{1-\gamma} \right) + \gamma^{k+1} \epsilon ||\tilde{J}_h^{(0)}|| + \gamma^k ||\tilde{J}_h^{(0)} - J_{u_h^*}||.$$

Proof: Let $\hat{J}_h^{(k)}$ specify the cost function in TT format obtained by performing k iterations of ϵ-level accuracy VI. Let $J_{u_h^*}$ indicate the solution of the optimal stochastic control problem using the direct method with discretization resolution h. We seek to bound the difference between these two tensors $\epsilon = ||\hat{J}_h^{(k)} - J_{u_h^*}||_F$. To determine the error ϵ, we first consider the error introduced at each iteration of VI.

Let $R_{max} = \max_{u(x),x} r(x, u(x))$, $\tilde{J}_h^{(k)}$ be a cost function approximation obtained by performing exact VI for k iterations, $J_h^{(k+1)}$ be a cost function approximation obtained by performing approximate VI for k iterations followed by a single iteration of exact VI, and $\mathbf{P}$ be the probability transition matrix of the Markov process. The difference between $\hat{J}_h^{(k+1)}$ and $\tilde{J}_h^{(k+1)}$ can then be bounded by

$$\begin{aligned} &||\hat{J}_h^{(k+1)} - \tilde{J}_h^{(k+1)}|| = ||\hat{J}_h^{(k+1)} - J_h^{(k+1)} + J_h^{(k+1)} - \tilde{J}_h^{(k+1)}|| \\ &\leq ||\hat{J}_h^{(k+1)} - J_h^{(k+1)}|| + ||J_h^{(k+1)} - \tilde{J}_h^{(k+1)}|| \\ &\leq \epsilon ||J_h^{(k+1)}|| + ||\gamma \mathbf{P} \hat{J}_h^{(k)} - \gamma \mathbf{P} \tilde{J}_h^{(k+1)}|| \\ &\leq \epsilon ||R_{max} + \gamma \mathbf{P} \hat{J}_h^{(k)}|| + \gamma ||\hat{J}_h^{(k)} - \tilde{J}_h^{(k)}|| \\ &\leq R_{max}\epsilon + \epsilon\gamma ||\hat{J}_h^{(k)}|| + \gamma ||\hat{J}_h^{(k)} - \tilde{J}_h^{(k)}|| \\ &\leq R_{max}\epsilon + \epsilon\gamma ||\hat{J}_h^{(k)}|| + \gamma \left(R_{max}\epsilon + \epsilon\gamma ||\hat{J}_h^{(k-1)}|| + \gamma ||\hat{J}_h^{(k-1)} - \tilde{J}_h^{(k-1)}|| \right), \end{aligned}$$

where the first inequality follows from the triangle inequality, the third inequality follows from contraction, and the fourth inequality follows from the fact that all eigenvalues of $\mathbf{P}$ are less than or equal to one. Continuing all the way to $k = 0$ we can bound the error as

$$||\hat{J}_h^{(k+1)} - \tilde{J}_h^{(k+1)}|| \leq \begin{cases} R_{max}\epsilon \sum_{i=0}^{k} \gamma^{k-i} + \epsilon\gamma \sum_{i=1}^{k} \hat{J}_h^{(i)} \gamma^{k-i} + \gamma^{k+1} \epsilon \tilde{J}_h^{(0)}, & \text{if } k \geq 0 \\ \epsilon J_*^{(0)}, & \text{otherwise.} \end{cases}$$

which simplifies to

$$||\hat{J}_h^{(k+1)} - \tilde{J}_h^{(k+1)}|| \leq \begin{cases} \epsilon \left(\tfrac{R_{max}+\gamma}{1-\gamma} \right) + \gamma^{k+1} \epsilon \tilde{J}_h^{(0)}, & \text{if } n \geq 0 \\ \epsilon J_*^{(0)}, & \text{otherwise.} \end{cases} \quad (5)$$

Utilizing this total error we can finally make a statement about the convergence of the TT-cross approximation to the true solution $J_{u_h^*}$ by using the triangle inequality $||\hat{J}_h^{(k)} - J_{u_h^*}|| \leq ||\hat{J}_h^{(k)} - \tilde{J}_h^{(k)}|| + ||\tilde{J}_h^{(k)} - J_{u_h^*}|| \leq \epsilon \left(\tfrac{R_{max}+\gamma}{1-\gamma} \right) + \gamma^{k+1} \epsilon ||\tilde{J}_h^{(0)}|| + \gamma^k ||\tilde{J}_h^{(0)} - J_{u_h^*}||$. The total computational cost is $\mathcal{O}\left(kdr^3/h\right)$, which was already argued in Section III-A. ∎

A few remarks are in order. First, notice that the error term can be set arbitrary close to zero, by choosing ϵ and h small enough and k large enough. Thus, by also Theorem 1, we conclude that the proposed algorithm solves the original continuous-time stochastic optimal control problem with arbitrary precision. Second, notice that the computational effort invested by the algorithm scales linearly with the dimensionality of the state space! Compare this result with the exponential increase in complexity for standard value iteration on a naive discretization of the state space. Furthermore, the error bound decreases rapidly with increasing number of VI iterations (k). The computational effort also scales with rank (r), which quantifies the 'complexity' of the problem instance at hand. Notice that this scaling is also polynomial. Finally, let us note that the non-interpolation, analytical, method also achieves the

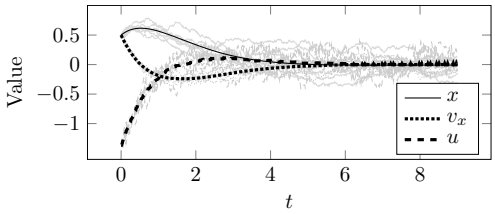

Fig. 1. Simulation of state and control variables under an interpolated control policy found from tensor-based value iteration applied to the LQG example with parameters $h = 10^{-2}$ and $\epsilon = 10^{-3}$. A maximum error of 3.1×10^{-2} is achieved across each variable and all time. The grey traces represent realizations of the controlled process.

same error bound. For the example presented in Section III-B, the computational effort is $O(kd^3\hat{r}^3/h)$ as discussed in the same section.

V. Simulation Examples

In this section we show the performance of tensor-based value iteration on two problems. The first is a two dimensional linear-quadratic-Gaussian optimization problem, and the second is seven-dimensional problem seeking to find a controller which will allow a glider to perch on a horizontal string.

A. Linear-Quadratic-Gaussian (LQG) stochastic control

We seek a feedback policy μ for the solution of an infinite horizon linear quadratic Gaussian (LQG) optimization problem with a two dimensional state space (x, v_x)

$$\min_{u(t)} \lim_{T\to\infty} \mathbb{E}\Big[\int_0^T x(t)^2 + v_x(t)^2 + u(t)^2 dt | (x(0), v_x(0)) = z \Big]$$

$$\text{subject to} \qquad dx = v_x dt, \quad dv_x = u(t)dt + 10^{-1} dw,$$

for all $z \in [-1, 1]^2$ such that $u(t) = \mu(x(t))$ are control realizations. Infinite horizon LQG problems have analytic solutions which can be determined analytically by solving the related continuous time algebraic Riccati equation (CARE). Reflective boundary conditions are chosen [13] for the implementation of the direct method. Because the boundary conditions are only specified in the direct method, we expect to see a small deviation between the solutions of the direct and analytical methods.

Tensor-based value iteration is performed using an interpolation scheme based on TT-cross for various discretization levels h and for various error tolerances ϵ. Recall that each dimension is discretized into $n = \frac{2}{h}$ nodes, and that the total number of discretized states is $n^2 = \frac{4}{h^2}$. Solution verification is performed in two steps. First, the state and control trajectories of the simulated system starting from $z = (0.5, 0.5)$ resulting from the direct and analytic methods are compared. These time traces are shown for $h = .03125$ or $n = 64$ and $\epsilon = 10^{-3}$ in Figure 1, where we obtain a maximum error $\max_t [\max [|x - x^*|, |v_x - v_x^*|, |u - u^*|]] = 3.1 \times 10^{-2}$, where x, v_x, u denote the mean position, velocity and controls computed with tensor-based value iteration, and and x^*, v_x^*, u^* denote the states and controls of the analytical solution.

Second, we compare of the norm of the cost function defined by $||J^*|| = \left(\int \left(J^*(z)\right)^2 dz \right)^{\frac{1}{2}}$ for the analytical and direct methods. The norm $J^*(z)$ for the analytical solution is computed with a uniform quadrature rule on $160,000$ quadrature nodes. The norm of the cost function in the direct method is computed as $||J^*|| \approx ||J_h(z)||_F / \sqrt{n^2}$, where n is the number of discretization nodes used in each direction. Figure 2 shows a trace of this cost for $h = 10^{-3}$ for various ϵ values as a function of the total number of minimization problems (3) solved. The slightly underestimated value of $||J^*||$ obtained through the numerical solutions may be attributed to boundary conditions. In this Figure, we can also see order of magnitude gains for the same cost function norm; for example, using $\epsilon = 10^{-1}$ rather than $\epsilon = 10^{-12}$ results in achieving a cost function norm of 1.4 with an order of magnitude less evaluations. Since the results of VI with $\epsilon = 10^{-12}$ are almost equivalent (see Section IV), these results suggest more than an order of magnitude gains over exact VI. The result also suggests that future work on ϵ-adaptive algorithms could focus their adaptation of ϵ when the VI algorithm begins to converge.

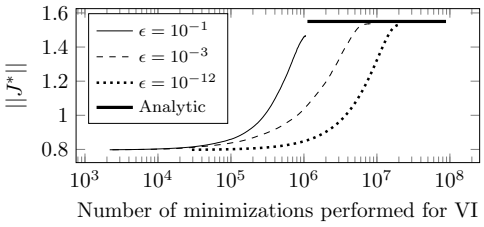

Fig. 2. Norm of the LQG objective obtained with $h = 10^{-2}$ for varying levels of ϵ as a function of total number of optimization solutions. For reference, the analytical solution is provided by the thick black line.

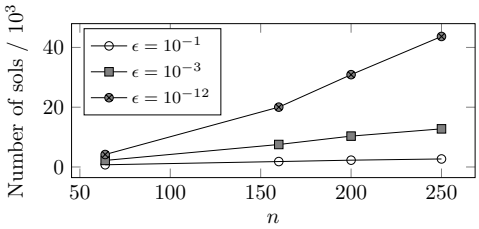

Fig. 3. Average number of states visited by the TT-cross algorithm over the course of VI as a function of discretization level for the LQG problem.

Finally, we can see linear growth in the average number of states visited per iteration of VI in Figure 3. Exact VI would result in quadratic growth in the number of states visited. For example, when $n = 250$ the total number of discretized states, and therefore the number of states visited each iteration with exact VI, is $62,500$. However, tensor-based VI with $\epsilon = 10^{-3}$ visits approximately $12,000$ states on average per iteration.

B. Perching

We now apply the tensor-based value iteration (Algorithm 1) on a seven dimensional stochastic control problem modeling an unpowered glider attempting to perch on a horizontal string [7, 22, 16]. The glider is described by flat-plate model in a two dimensional plane involving seven state variables $(x, y, \theta, \phi, v_x, v_y, \dot{\theta})$ specifying its x-position, y-position, angle of attack, elevator angle, horizontal speed, vertical speed, and the rate of change of the angle of attack respectively. The input

control is the rate of change of the elevator angle $u = \dot{\phi}$. A successful perch is defined by a horizontal velocity between 0 and 2 m/s, a vertical velocity between -1 and -3 m/s, and the x and y positions of the glider within a 5cm radius of the perch. Under these conditions, the experimental aircraft in [7] can attach to the string. For a diagram of this glider refer to either [22] or [16]. To achieve a perch, we solve the following optimization problem:

$$J^*(z) = \min_{u(t)} \mathbb{E}\Big[\int_0^T \bar{x}^T \mathbf{Q} \bar{x} + 0.1u^2 dt + \bar{x}(T)^T \mathbf{Q}_f \bar{x}(T) \Big]$$

subject to

$$\begin{aligned}
\mathbf{x}_w &= [x - l_w c_\theta, y - l_w s_\theta], \quad \dot{\mathbf{x}_w} = [\dot{x} + l_w \dot{\theta} s_\theta, \dot{y} - l_w \dot{\theta} c_\theta] \\
\mathbf{x}_e &= [x - l c_\theta - l_e, c_{\theta+\phi}, y - l s_\theta - l_e s_{\theta+\phi}] \\
\dot{\mathbf{x}}_e &= [\dot{x} + l\dot{\theta} s_\theta + l_e(\dot{\theta} + u) s_{\theta+\phi}, \dot{y} - l\dot{\theta} c_\theta - l_e(\dot{\theta} + u) c_{\theta+\phi}] \\
\alpha_w &= \theta - \tan^{-1}(\dot{y}_w, \dot{x}_w), \quad \alpha_e = \theta + \phi - \tan^{-1}(\dot{y}_e, \dot{x}_e) \\
f_w &= \rho S_w |\dot{\mathbf{x}}_w|^2 \sin(\alpha_w), \quad f_e = \rho S_e |\dot{\mathbf{x}}_e|^2 \sin(\alpha_e) \\
m\ddot{x} &= -f_w s_\theta - f_e s_{\theta+\phi} + F dw \\
m\ddot{y} &= f_w c_\theta + f_e c_{\theta+\phi} - mg + F dw \\
I\ddot{\theta} &= -f_w l_w - f_e(l c_\phi + l_e) + F dw
\end{aligned}$$

where ρ is the density of air, m is the mass of the glider, I is the moment of inertia of the glider, S_w and S_e are the surface areas of the wing and tail control surfaces, l is the length from the center of gravity to the elevator, l_w is the half chord of the wing, l_e is the half chord of the elevator, c_γ denotes $\cos(\gamma)$, and s_γ denotes $\sin(\gamma)$. The parameters of the model are chosen to be the same as those in [22]

Our setup differs slightly from that reported in [16]. We have added a stochastic diffusion term to model external disturbances, and we solve for a global policy that provides a control for each state value. However, the diffusion term is specified as $F = 10^{-4}$ to obtain a problem instance that is similar to the one found in the literature. The state penalties in our cost function differ slightly from those in the literature, and they were chosen to obtain the desired behavior of our glider. We have chosen $\mathbf{Q}_f = \text{diag}(600, 400, \frac{1}{9}, \frac{1}{9}, 1, 1, \frac{1}{9})$ and $\mathbf{Q} = \text{diag}(20, 50, 10, 1, 1, 11)$.

The direct method requires the discretization of the state space. To this end, we we first restrict our state space to a hypercube bounded by $x \in [-4, 0]$, $y \in [-1, 1]$, $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\phi \in [-\frac{2\pi}{9}, \frac{2\pi}{9}]$, $v_x \in [0, 7]$, $v_y \in [-5, 5]$, and $\dot{\theta} \in [-10, 10]$. We uniformly discretize each variable into 32 states, resulting in a discrete state space of 3.4×10^{10} states. We use $\epsilon = 0.1$.

Figure 4 illustrates the state trajectories of a simulation. These results match qualitatively with those reported in the literature. This controller is successful, because the position of the glider enters the blue ellipse indicating a 5cm radius from the perching string, the horizontal velocity is approximately 0 m/s, and the vertical velocity is -2m/s, at the end. This vertical speed is in the middle of the range required for the perch.

Finally, the cost function is indeed low rank, and Figure 5 shows the average number of states evaluated per value iteration. For example, when $n = 16$ the total number of discretized states

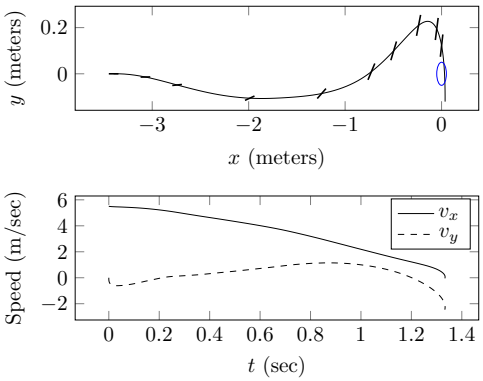

Fig. 4. Simulated state variables under the control policy found from tensor-based value iteration applied to the perching optimization problem with parameters $n = 32$, 3.4×10^{10} total states, and $\epsilon = 10^{-1}$. The thick black line segments in the top panel indicate the glider's angle of attack, and the blue circle indicates the region which the perching mechanism of [7] requires.

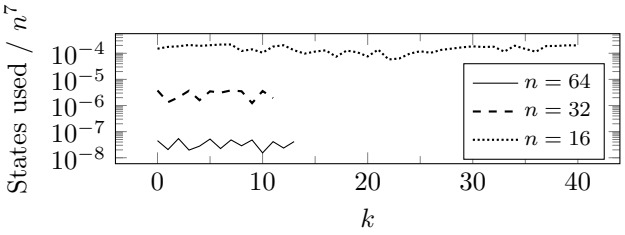

Fig. 5. Fraction of number of discretized states per iteration of VI for various discretization levels n and $\epsilon = 0.1$

is $n^7 \approx 2.7 \times 10^8$ of which approximately 10^4 are used every iteration. At the other extreme, when $n = 64$ the total number of discretized states is 4.4×10^{12} of which approximately 10^5 are used each iteration. Therefore, we see gains between four and seven orders of magnitude per iteration of VI. In fact, the gains in this problem are large enough that this problem can be solved on a personal computer with a 2 GHz Intel Core i7 processor and 8 GB of RAM.

VI. Conclusion

We have proposed, analyzed, and demonstrated a tensor-based value iteration algorithm which mitigates the curse of dimensionality that is associated with stochastic control problems, by taking advantage of the low-rank structure embedded in the value function of the problem. Through theoretical and computational analysis, we have verified and demonstrated an algorithm that achieves several orders of magnitude improvement when compared to the standard, discretization-based stochastic optimal control algorithms. The benefits are substantial enough that the computational requirements for both obtaining and storing the solutions are met by a personal computer in challenging examples, such as the longitudinal aircraft perching control. Since the direct method can also be applied in estimation problems, our future work will focus on utilizing the framework described here to that effect. Overall, our approach provides a flexible and scalable framework for use in a variety of optimal stochastic control and estimation problem instances that were previously considered intractable.

References

[1] M Bardi and I Capuzzo-Dolcetta. Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations, 2008.

[2] D. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, 2007.

[3] D P Bertsekas. *Dynamic Programming and Optimal Control*, volume II. Athena Scientific, 4th edition, 2012.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, 1996.

[5] J. Canny. *The Complexity of Robot Motion Planning*. The MIT Press, 1988.

[6] A. Cichocki. Tensor Networks for Big Data Analytics and Large-Scale Optimization Problems. *arXiv preprint arXiv:1407.3124*, pages 1–36, 2014. URL http://arxiv.org/abs/1407.3124.

[7] R. Cory and R. Tedrake. Experiments in fixed-wing uav perching. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 1–12. AIAA Reston, VA, 2008.

[8] M J Grimble, R J Patton, and D A Wise. The Design of Dynamic Ship Positioning Control Systems using Stochastic Optimal Control Theory. *Optimal Control Application and Methods*, 1:167–202, 1980.

[9] M. B. Horowitz, A. Damle, and J. W. Burdick. Linear Hamilton Jacobi Bellman Equations in High Dimensions. *arXiv preprint arXiv:1404.1089*, 2014. URL http://arxiv.org/abs/1404.1089.

[10] H J Kushner. *Introduction to Stochastic Control*. Holt, Rinehart and Winston, Inc., 1971.

[11] H. J. Kushner. *Probability methods for approximations in stochastic control and for elliptic equations*, volume 129. Academic Press New York, 1977.

[12] H. J. Kushner. Numerical methods for stochastic control problems in continuous time. *SIAM Journal on Control and Optimization*, 28(5):999–1048, 1990.

[13] H. J. Kushner and P. Dupuis. *Numerical methods for stochastic control problems in continuous time*, volume 24. Springer, 2001.

[14] S. M. LaValle. *Planning algorithms*. Cambridge Univ Pr, May 2006.

[15] P L Lions. Optimal Control of Diffusion Processes and Hamilton-Jacobi-Bellman Equations: The Dynamic Programming Principles and Applications, 1983.

[16] J. Moore and R. Tedrake. Control synthesis and verification for a perching uav using lqr-trees. In *CDC*, pages 3707–3714. Citeseer, 2012.

[17] B Oksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer Verlag, 2003.

[18] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[19] I. V. Oseledets and E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[20] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[21] J. Reif. Complexity of the Generalized Mover's Problem. In J Schwartz, J Hopcroft, and M. Sharir, editors, *Planning, Geometry, and Complexity of Robot Motion*, pages 267–281. Ablex Publishing Corp., 1987.

[22] J. W Roberts, R. Cory, and R. Tedrake. On the controllability of fixed-wing perching. In *American Control Conference, 2009. ACC'09.*, pages 2018–2023. IEEE, 2009.

[23] D. Savostyanov and I.V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In *Multidimensional (nD) Systems (nDs), 2011 7th International Workshop on*, pages 1–8. IEEE, 2011.

[24] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. The MIT Press, 2005.

[25] E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64(4):367–380, 2000.

[26] J Yuh. Design and Control of Autonomous Underwater Robots: A Survey. *Autonomous Robots*, 8:7–24, 2000.