INTERNATIONAL JOURNAL OF CONTROL
https://doi.org/10.1080/00207179.2022.2113438

Taylor & Francis
Taylor & Francis Group

Dimension reduction based adaptive dynamic programming for optimal control of discrete-time nonlinear control-affine systems

Qiang Li and Yunjun Xu

Department of Mechanical and Aerospace Engineering, University of Central Florida, Orlando, FL, USA

ABSTRACT

Dynamic programming based methods have been widely used in solving discrete-time nonlinear constrained optimal control problems. However, applying these methods in real-time is challenging because a large amount of memory is needed and the associated computational cost is high. Here, a search space dimension reduction strategy is proposed for a class of nonlinear discrete-time systems that are control-affine and invertible. Specifically, a bio-inspired motion rule is combined with inverse dynamics to reduce the value iteration search space to one dimension. The corresponding suboptimal control algorithm is developed and its optimality is analysed. An adaptation rule is developed to estimate uncertainties and improve the base policy. The closed-loop system is proven to be asymptotically stable. The advantages of the algorithm including much smaller computational cost and significantly reduced memory usage are demonstrated with two simulation examples.

ARTICLE HISTORY
Received 13 September 2021
Accepted 2 August 2022

KEYWORDS
Nonlinear constrained optimal control; adaptive dynamic programming; real-time control; control-affine system; uncertainty

1. Introduction

Nonlinear constrained optimal control aims to drive a nonlinear system towards a desired value or along a desired trajectory subject to equality and/or inequality constraints, while optimising a user-defined performance index (Betts, 2010; Pontryagin, 1986). This type of problem is ubiquitous and has been seen in many applications such as spacecraft maneuvering (Xin & Pan, 2009), wind turbine optimal control (Gros & Schild, 2017), and autonomous scouting in agricultural fields (N. Li et al., 2015).

Dynamic programming (DP), as one subset of nonlinear constrained optimal control methods, has been widely studied for many decades, especially for those with discrete-time dynamics (Chisci et al., 1998; Howard, 1960; Werbos, 1989a). Based on Bellman's Principle of Optimality, as shown in the seminal work by Bellman (1954), the basic procedure of DP involves two steps of operations over a discretised state-action search space. A value iteration (VI) algorithm is an explicit implementation of such a procedure (Bellman, 1961). In the first step, the Bellman Equation (Howard, 1960) is followed to compute the value function backward in time over the entire state-action space until the optimal value function is reached. In the second step, the optimal value function generates the greedy control actions along the state trajectory (Bellman, 1961). As another implementation of the procedure, in a policy iteration (PI) algorithm (Bellman, 1961), instead of searching all possible control actions, a pre-selected control policy is used in the first step to calculate the value function, which is then used to improve the control policy until an optimal control policy is reached. In the second step, the optimal control policy is followed along the state trajectory (Bellman, 1961). Memory usage and computational cost of DP, especially in Step 1 of both VI and PI, grow exponentially with respect to the dimension of search space and the number of discretised nodes in each dimension – referred to as the curses of dimensionality (Bellman, 1961). This has prevented DP from being utilised in many real-time optimal control problems. For stochastic systems, this problem gets even worse.

For the purpose of mitigating the computational burden of DP, parameterised function approximators (Busoniu et al., 2010; Lendaris et al., 2002; Powell, 2007), primarily neural networks (Bertsekas & Tsitsiklis, 1996; Werbos, 1989b), have been applied to predict the values associated with unvisited states based on those of visited ones. However, the function approximation approaches have their downsides. To start with, it has been pointed out that replacing the lookup table in DP with function approximators sometimes is not robust even for very simple problems (Boyan & Moore, 1995). Besides, there is no guideline about the choice of function approximators, for example the configurations in neural network (Tesauro, 1992). A successful design of a function approximator highly relies on experiences and tuning (Powell, 2007). Additionally, a large amount of data and offline training are typically required to produce a close approximation of the value function (Polak, 1973). Furthermore, to obtain a good approximation, the number of parameters in a function approximator has to be sufficiently large and a good portion of the state-action space needs to be visited (Borghese & Arbib, 1995).

Another set of popular methods to mitigate the computational burden of DP are rollout algorithms (RA) (Bertsekas et al., 1997) and receding horizon control (RHC) (Mayne & Michalsha, 1988) or model predictive control (Clarke

CONTACT Yunjun Xu yunjun.xu@ucf.edu Department of Mechanical and Aerospace Engineering, University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32826, USA

et al., 1987) algorithms. Although RHC is not a DP exclusive method, it and RA utilise similar multi-step lookahead strategies and approximate the original optimal control problem with one of a shorter horizon at each time instant (Bertsekas, 2020). They essentially solve the Bellman equation in a neighbourhood of the current state. From this perspective, these methods can be regarded as subspace based methods that trade optimality for computational cost reductions. The difference between RA and RHC lies in the fact that the former approximates the value function outside the subspace following a base policy (Bertsekas, 2013), whereas the latter truncates the value function beyond the horizon (Michalsha & Mayne, 1993). Depending on the size of the subspace, which is determined by the number of lookahead steps, computational cost can still be a concern for real-time implementation of these methods (Bertsekas, 2013). On the other hand, techniques for reducing computational cost of standard DP can be implemented within the subspace of RA and RHC to further reduce the computational cost.

With the introduction of approximate/adaptive dynamic programming (ADP) (Barto et al., 1983; Prokhorov & Wunsch, 1997; Sutton, 1988; Watkins, 1989; Werbos, 1982), the offline computation of DP is transferred to online and the feedback information is then used to adapt to system uncertainties. ADP is a class of methods (Barto et al., 1983; Prokhorov & Wunsch, 1997; Sutton, 1988; Watkins, 1989; Werbos, 1982) that solve the Bellman equation forward in time through successive steps of policy evaluation and policy improvement. Among ADP methods, heuristic dynamic programming (HDP) (Werbos, 1990) sees the most applications (Bhatnagar, 2010; Borkar, 2005; Liu et al., 2013). It uses an actor network as the control policy for the online forward propagation and a critic network approximating the value function based on system feedback information (Werbos, 1990). The critic network output is then used to evaluate the actor network and improve it accordingly (Werbos, 1990). This procedure is repeated until an optimal control policy is reached (Werbos, 1990). As improvement of the control policy usually requires the calculations of value function derivatives, dual heuristic programming (DHP) is developed to approximate the derivative directly (Werbos, 1992). Furthermore, with globalised DHP (GDHP), not only the value function but also its derivative are approximated (Miller et al., 1995). There are also action-dependent (AD) variants of the aforementioned methods (i.e. ADHDP Watkins & Dayan, 1992, ADDHP Prokhorov & Wunsch, 1997, and ADGDHP Prokhorov & Wunsch, 1997) that assume the value function to be directly related to both state and action. However, because most policy-based ADP methods (Pi et al., 2020; Silver et al., 2014) assume the value function to be differentiable, they are not suitable for constrained optimal control problems where smoothness of the value function sometimes cannot be guaranteed (Yaghmaie & Braun, 2019). Additionally, although the computation of the value function is transferred from offline to online, a sufficient collection of online data is still necessary for the training of the critic network (Luo et al., 2018), resulting in high CPU occupation and memory usage.

In brief, it is still a challenging task to apply DP and its variations to many discrete-time, nonlinear constrained optimal control problems in real-time. Two observations are: (1) the inherent dynamics, many of which are known or partially known, haven't been fully utilised; and (2) knowledge of the environment is not fully exploited, leading to inefficient search over noncontributory regions of the state-action space. A subspace method has been discussed to reduce the computational cost by finding local optimal or suboptimal solutions in Wei et al. (2017).

A two-step search space dimension reduction strategy is presented here to tackle the aforementioned issue for a class of nonlinear discrete-time dynamic systems that are control-affine and invertible. Instead of searching the action space exhaustively, the first dimension reduction step uses an inverse dynamics (ID) policy (Forrest-Barlach & Babcock, 1987; Kumar & Seywald, 1996) as the base policy to calculate the control action based on the current state and the desired state-to-reach in an Euler scheme, resulting in an ID-based VI (ID-VI) method. Furthermore, the state search space is confined using the virtual motion camouflage (VMC) rule, in which the state variation is constrained by the 1-dimensional path control parameter (PCP) (Xu & Basset, 2012). The motion camouflage phenomenon (Srinivasan & Davey, 1995) is observed in mating hoverflies when a male hoverfly is approaching a female one while its motion is shown static on the retina of the female counterpart. Xu and Basset (2012) have utilised this strategy in developing direct collocation based suboptimal controllers for continuous-time optimal trajectory planning problems, which are dramatically different from the problems and approaches in this paper. Here, the VMC rule is used in the second dimension reduction step of DP to project the state space into a 1-dimensional PCP parameter space and the ID-VI method is then developed into the VMC-based ID-VI (ID-VMC-VI) method. The main structure of VI method remains intact. Similar to most VI methods, once the value function has converged, the actual control action will be generated by the greedy policy (Powell, 2007).

The advantages of the resulting ID-VMC-VI algorithm are summarised as follows. (1) Since the search space is 1-dimensional, the optimisation related computation time is low. (2) Since only values corresponding to the 1-dimensional PCP variables are stored, the memory usage is significantly reduced. This can benefit many real-world applications with limited random access memory. (3) It is shown that the closed-loop system is asymptotically stable and the solution optimality is lower bounded by a first-order tracking controller or a second order tracking controller under two different scenarios. (4) When combined with a gradient based ID policy improvement rule, the ID-VMC-VI algorithm can solve constrained optimal control problems with uncertainties. A stability analysis is provided for the overall algorithm.

This study is a significant extension from our previous conference publication (Li & Xu, 2020). (1) Here we focus on the development of ID-VMC-VI method as a general method for discrete-time invertible nonlinear control-affine systems. In comparison, Li and Xu (2020) only applied the preliminary version of ID-VI, ID-VMC-VI, and another algorithm in an unmanned aerial vehicle (UAV) angular velocity control problem. (2) The theoretical aspect of ID-VMC-VI has been revised and extended with respect to the a general class of discrete-time optimal control problems, considering state and control inequality constraints in a systematic manner. (3) Additional sections on dimensionality, stability and optimality analyses

have been developed. (4) A different ID policy improvement method is developed to fit a wider range of problems with different performance indices. Additional convergence and stability analyses are provided. (5) Guidelines on how to apply the algorithm are offered. (6) In this paper, two simulation examples are used for illustration. While the first example is adopted and refined from Li and Xu (2020), the improved algorithm and new ID policy improvement method are applied. Furthermore, the simulation parameters and settings are tuned for a better result demonstration. The second example is a new and challenging application involving obstacle avoidance.

Section 2 formulates the problem and recalls necessary background knowledge on DP. The two search dimension reduction steps are presented in Section 3, along with stability, dimensionality and optimality analyses. Section 4 presents the ID policy improvement rule and its stability and convergence analyses. The overall algorithm is summarised in Section 5 and two simulation examples are shown in Section 6. The study is concluded at the end.

2. Problem definition and background information

A discrete-time, invertible, and control-affine system can be formulated as

$$\boldsymbol{x}_{t_{i+1}} = F(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}) + G(\boldsymbol{x}_{t_i}, \boldsymbol{\theta})\boldsymbol{u}_{t_i}, \quad \boldsymbol{x}_{t_i} \in \mathcal{S}^n,\ \boldsymbol{u}_{t_i} \in \mathcal{A}^m,\ \boldsymbol{\theta} \in \mathcal{P}^r \tag{1}$$

where $\boldsymbol{x}_{t_i}$ is the state vector at time instant t_i, $i = 0, 1, 2, \ldots$. $\boldsymbol{u}_{t_i}$ is the control vector. $\boldsymbol{\theta}$ consists of uncertain parameters. $\mathcal{S}^n \subset \mathbb{R}^n$ is the constrained state vector space. $\mathcal{A}^m \subset \mathbb{R}^m$ is the constrained control vector space. $\mathcal{P}^r \subset \mathbb{R}^r$ is the uncertain parameter vector space. Assuming that $G(\boldsymbol{x}_{t_i}, \boldsymbol{\theta})$ has a rank of m, its Moore-Penrose inverse $G^{-1}(\boldsymbol{x}_{t_i}, \boldsymbol{\theta})$ exists (Penrose, 1954).

Because of the uncertainties in $\boldsymbol{\theta}$, given state $\boldsymbol{x}_{t_i}$ and control $\boldsymbol{u}_{t_i}$, the predicted system state $\hat{\boldsymbol{x}}_{t_{i+1}}$ based on $\hat{\boldsymbol{\theta}}_{t_i}$, which is the estimate of $\boldsymbol{\theta}$ at t_i, will differ from the actual state feedback $\boldsymbol{x}_{t_{i+1}}$. Define $\tilde{\boldsymbol{x}}_{t_{i+1}} = \boldsymbol{x}_{t_{i+1}} - \hat{\boldsymbol{x}}_{t_{i+1}}$ and let $e_{t_{i+1}} = \tilde{\boldsymbol{x}}_{t_{i+1}}^T \tilde{\boldsymbol{x}}_{t_{i+1}}/2$. The estimation problem (**E1**) is defined as: Solve for $\hat{\boldsymbol{\theta}}_{t_{i+1}}$ to minimise $e_{t_{i+1}}$.

The control problem (**C1**) is defined as: Solve for the optimal control sequence $\{\boldsymbol{u}^*\}_{k=i}^{N-1}$ so that the following performance index, or value function, is minimised (Howard, 1960):

$$V(\boldsymbol{x}_{t_i}) = \sum_{k=i}^{N-1} L(\boldsymbol{x}_{t_k}, \boldsymbol{u}_{t_k}) \tag{2}$$

subject to the equality constraint (1) and the boundary conditions:

$$\boldsymbol{x}_{t_i} = \boldsymbol{x}_{t_i} \quad \text{and} \quad \boldsymbol{x}_{t_N} = \boldsymbol{x}_{d,t_i} \tag{3}$$

where $L(\boldsymbol{x}_{t_k}, \boldsymbol{u}_{t_k}) \geq 0$ is the step cost of taking $\boldsymbol{u}_{t_k}$ in $\boldsymbol{x}_{t_k}$. $\boldsymbol{x}_{d,t_i} \in \mathcal{S}^n$ is the desired state-to-reach from $\boldsymbol{x}_{t_i}$. $L(\boldsymbol{x}_{t_k}, \boldsymbol{u}_{t_k}) = 0$ only when $\boldsymbol{x}_{t_k} = \boldsymbol{x}_{d,t_i}$ and $F(\boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}_{t_k} = \boldsymbol{x}_{d,t_i}$.

The basic VI scheme solves for the optimal value function $V^*(\boldsymbol{x}_{t_i})$ in **C1** over the $(m+n)$-dimensional state-action space using the Bellman equation (Powell, 2007)

$$V_{j+1}(\boldsymbol{x}_{t_k}) = \min_{\boldsymbol{u}_{t_k} \in \mathcal{A}^m} \left\{ L(\boldsymbol{x}_{t_k}, \boldsymbol{u}_{t_k}) + V_j(\boldsymbol{x}_{t_{k+1}}) \right\}, \quad \forall\, \boldsymbol{x}_{t_k} \in \mathcal{S}^n \tag{4}$$

where $k \geq i$. With (4), $V_j(\boldsymbol{x}_{d,t_i}) = 0$, $j = 0, 1, 2, \ldots$, is guaranteed. State constraints are introduced in the form of penalty in the value function, i.e. $V_j(\boldsymbol{x}_{t_k}) = \mathit{Penalty}$, $\forall\, \boldsymbol{x}_{t_k} \notin \mathcal{S}^n$.

In this paper, we consider the tabular approximation of the value function. We define $\mathcal{S}^n$ to always refer to the discrete state vector space, and $\mathcal{A}^m$ to refer to the discrete action vector space.

For an actual system, the discrete vector spaces $\mathcal{S}^n$ and $\mathcal{A}^m$ are finite. Therefore, within finite iterations, the value iteration stopping condition, $V_{j+1}(\boldsymbol{x}_{t_k}) = V_j(\boldsymbol{x}_{t_k})$, $\forall\, \boldsymbol{x}_{t_k} \in \mathcal{S}$, will be met and the optimal value function $V^*(\boldsymbol{x}_{t_k})$, $\forall\, \boldsymbol{x}_{t_k} \in \mathcal{S}$, will be reached. The optimal value function at t_i satisfies

$$V^*(\boldsymbol{x}_{t_k}) = \min_{\boldsymbol{u}_{t_k} \in \mathcal{A}^m} \left\{ L(\boldsymbol{x}_{t_k}, \boldsymbol{u}_{t_k}) + V^*(\boldsymbol{x}_{t_{k+1}}) \right\}, \quad \forall\, \boldsymbol{x}_{t_k} \in \mathcal{S}^n, \tag{5}$$

and the optimal control to take in state $\boldsymbol{x}_{t_i}$ with respect to (2) is given as Powell (2007)

$$\boldsymbol{u}_{t_i}^* = \arg\min_{\boldsymbol{u}_{t_i} \in \mathcal{A}^m} \left\{ L(\boldsymbol{x}_{t_i}, \boldsymbol{u}_{t_i}) + V^*(\boldsymbol{x}_{t_{i+1}}) \right\}. \tag{6}$$

Discussions on the existence and uniqueness of $\boldsymbol{u}_{t_i}^*$ can be found in Powell (2007). In general, it is assumed that 'arg min' includes all necessary operations to find the best control vector.

Because $\boldsymbol{x}_{d,t_i}$ and $\hat{\boldsymbol{\theta}}_{t_i}$ are time-varying, **C1** needs to be solved at each time instant t_i. However, due to the curse of dimensionality, solving **C1** via basic VI scheme can be impracticable when the system dimension is high, even if $\mathcal{S}^n$ and $\mathcal{A}^m$ are discrete and finite. We present a two-step search space dimension reduction strategy in the following.

3. Search space dimension reduction

3.1 Action space elimination

In the first dimension reduction step, the control action is calculated by an ID policy (Forrest-Barlach & Babcock, 1987) based on the current state and a state-to-reach. Following this procedure, instead of searching the $(m+n)$-dimensional state-action space to find the optimal control action, we only need to search the m-dimensional state space for the optimal state-to-reach. The optimal control action can then be calculated by the ID policy.

The ID policy based control to take in $\boldsymbol{x}_{t_k}$ to reach $\boldsymbol{x}_{d,t_k}$ in one time step is calculated based on the estimate $\hat{\boldsymbol{\theta}}_{t_i}$ as

$$\begin{aligned} \boldsymbol{u}_{\pi,t_k} &= G^{-1}(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_i})[\boldsymbol{x}_{d,t_k} - F(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_i})] \\ &\triangleq \pi(\boldsymbol{x}_{t_k}, \boldsymbol{x}_{d,t_k}, \hat{\boldsymbol{\theta}}_{t_i}) \end{aligned} \tag{7}$$

where $\boldsymbol{x}_{d,t_k}$ is assumed reachable within one time step without considering the constraints. $\boldsymbol{u}_{\pi,t_k}$ is then saturated to satisfy the control constraint $\boldsymbol{u}_{\pi,t_k} \in \mathcal{A}^m$ before used for the state propagation in the value iteration.

Choosing $\boldsymbol{x}_{d,t_k} = \boldsymbol{x}_{d,t_i}$ and substituting (7) into (4), the Bellman equation for ID-VI can be written as

$$V_{\pi,j}(\boldsymbol{x}_{t_k}) = L(\boldsymbol{x}_{t_k}, \pi(\boldsymbol{x}_{t_k}, \boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{\boldsymbol{t_i}})) + V_{\pi,j-1}(\boldsymbol{x}_{t_{k+1}}), \quad \forall\, \boldsymbol{x}_{t_k} \in \mathcal{S}^n \tag{8}$$

where we follow the notation from Sutton and Barto (2018) and use V_π to denote the value function of following policy $\boldsymbol{u}_{t_k} = \pi(\boldsymbol{x}_{t_k}, \boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{\boldsymbol{t_i}})$ during value iteration.

Compared with (4), which needs to be solved in the $(m+n)$-dimensional state-action space, (8) is solved in the n-dimensional state space. Thus, the action space is eliminated from the search space following ID-VI and the search space dimension is reduced from $(m+n)$ to n.

In a discrete and finite $\mathcal{S}^n$, the convergence of ID-VI is guaranteed. Once V_π has converged to V_π^*, which is the optimal value function following policy π, we search for the predicted optimal state-to-reach $\boldsymbol{x}_{t_{i+l}}^{s*}$ by solving

$$\boldsymbol{x}_{t_{i+l}}^{s*} = \underset{\boldsymbol{x}_{t_{i+l}} \in \mathcal{S}^n}{\arg\min} \left\{ \sum_{p=0}^{l-1} L(\boldsymbol{x}_{t_i+p}, \pi(\boldsymbol{x}_{t_{i+p}}, \boldsymbol{x}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) + V_\pi^*(\boldsymbol{x}_{t_{i+l}}) \right\}. \tag{9}$$

where l is the number of steps it takes to propagate from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_{i+l}}$, and l varies depending on $\boldsymbol{x}_{t_{i+l}}$. Because of the control constraints, some states in $\mathcal{S}^n$ are not reachable from $\boldsymbol{x}_{t_i}$ in one time step. Thus, the l-step lookahead strategy (Sutton, 1988) is used in (9) to evaluate those states as the state-to-reach. During the l-step lookahead, controls are constrained and the state constraints are introduced as penalty.

Remark 3.1: Because of the state constraints, e.g. collision avoidance and obstacle avoidance, the reachable set of states from $\boldsymbol{x}_{t_i}$ following the ID policy will be different from the reachable set from $\boldsymbol{x}_{t_{i+1}}$. As a result, even if $\boldsymbol{x}_{d,t_i} = \boldsymbol{x}_{d,t_{i+1}}$, the predicted optimal state-to-reach at different time step will not necessarily be the same.

The ID-VI based control $\boldsymbol{u}_{t_i}^{s*}$ is calculated by substituting $\boldsymbol{x}_{t_{i+l}}^{s*}$ from (9) as the desired state-to-reach from $\boldsymbol{x}_{t_i}$ into (7), i.e.

$$\boldsymbol{u}_{t_i}^{s*} = \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_{i+l}}^{s*}, \hat{\boldsymbol{\theta}}_{t_i}). \tag{10}$$

3.2 State space projection via VMC

To further reduce the dimension of the search space, the VMC rule (Srinivasan & Davey, 1995; Xu & Basset, 2012) is applied to confine the state search space to a VMC subspace $\mathcal{S}_{\upsilon,t_i}^n \subseteq \mathcal{S}^n$ that is shaped by a reference state $\boldsymbol{x}_{ref,t_i}$ and a desired state-to-reach $\boldsymbol{x}_{d,t_i}$. The VMC rule uniquely determines a state $\boldsymbol{x}_{t_k}^\upsilon \in \mathcal{S}_{\upsilon,t_i}^n$ by a PCP υ_{t_k} from the 1-dimensional PCP space $\mathcal{V}^1$. Following this procedure, searching for the optimal state-to-reach in the m-dimensional state space is transferred into searching for the optimal PCP-to-go in the 1-dimensional PCP space, which is then projected back into the state space to obtain the optimal state-to-reach.

Following the VMC rule, a $\boldsymbol{x}_{t_k}^\upsilon \in \mathcal{S}_{\upsilon,t_i}^n$ can be determined by a PCP $\upsilon_{t_k} \in \mathcal{V}^1$ as Xu and Basset (2012)

$$\begin{aligned} \boldsymbol{x}_{t_k}^\upsilon &= \boldsymbol{x}_{ref,t_i} + \upsilon_{t_k}(\boldsymbol{x}_{d,t_i} - \boldsymbol{x}_{ref,t_i}) \\ &\triangleq \boldsymbol{x}(\upsilon_{t_k}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i}), \quad \upsilon_{t_k} \in \mathcal{V}^1 \end{aligned} \tag{11}$$

where $\boldsymbol{x}_{ref,t_i}$ corresponds to a PCP value of 0, and $\boldsymbol{x}_{d,t_i}$ corresponds to a PCP value of 1. It is practical to let $\mathcal{V}^1 = [0, \upsilon_{\max}]$, where $\upsilon_{\max}$ can be chosen based on desired system characteristics. For example, $\upsilon_{\max} = 1$ corresponds to a desired state response without overshoot, while $\upsilon_{\max} = 1.2$ allows for a 20% overshoot. Different from Xu and Basset (2012), $\boldsymbol{x}_{t_i}$ does not necessarily belong to $\mathcal{S}_{\upsilon,t_i}^n$. The reference state $\boldsymbol{x}_{ref,t_i}$ can be selected in such a way that $\mathcal{S}_{\upsilon,t_i}^n \subseteq \mathcal{S}^n$, i.e. every $\boldsymbol{x}_{t_k}^\upsilon \in \mathcal{S}_{\upsilon,t_i}^n$ satisfies the state constraints. When the state constraints are too complicated to satisfy, desired waypoint states-to-go can be inserted to divide the original problem into sub-problems.

Equation (11) is a bijection, meaning for each $\upsilon_{t_k} \in \mathcal{V}^1$, there is a corresponding $\boldsymbol{x}_{t_k}^\upsilon \in \mathcal{S}_{\upsilon,t_i}^n$, and vice versa. The inverse projection from the VMC subspace $\mathcal{S}_{\upsilon,t_i}^n$ to the PCP space $\mathcal{V}^1$ is

$$\begin{aligned} \upsilon_{t_k} &= \left\langle \boldsymbol{x}_{t_k}^\upsilon - \boldsymbol{x}_{ref,t_i}, \boldsymbol{x}_{d,t_i} - \boldsymbol{x}_{ref,t_i} \right\rangle / \left\| \boldsymbol{x}_{d,t_i} - \boldsymbol{x}_{ref,t_i} \right\|_2 \\ &\triangleq \upsilon(\boldsymbol{x}_{t_k}^\upsilon, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i}), \quad \boldsymbol{x}_{d,t_i} \neq \boldsymbol{x}_{ref,t_i},\ \boldsymbol{x}_{t_k}^\upsilon \in \mathcal{S}_{\upsilon,t_i}^n. \end{aligned} \tag{12}$$

Substituting (11) into (7) with $\boldsymbol{x}_{d,t_i}$ as the desired state-to-reach, the ID policy under the VMC rule becomes

$$\boldsymbol{u}_{\pi,t_k}^\upsilon = \pi(\boldsymbol{x}(\upsilon_{t_k}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i}), \boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{t_i}). \tag{13}$$

The Bellman equation for ID-VMC-VI is obtained by substituting (11) and (13) into (8) as

$$\begin{aligned} &V_{\pi,j}(\boldsymbol{x}(\upsilon_{t_k}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})) \\ &\quad = L(\boldsymbol{x}_{t_k}^\upsilon, \boldsymbol{u}_{\pi,t_k}^\upsilon) + V_{\pi,j-1}(\boldsymbol{x}(\upsilon_{t_{k+1}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})), \\ &\quad \forall\, \upsilon_{t_k} \in \mathcal{V}^1, \end{aligned} \tag{14}$$

or simply

$$V_{\pi\upsilon,j}(\upsilon_{t_k}) = L(\boldsymbol{x}_{t_k}^\upsilon, \boldsymbol{u}_{\pi,t_k}^\upsilon) + V_{\pi\upsilon,j-1}(\upsilon_{t_{k+1}}), \quad \forall\, \upsilon_{t_k} \in \mathcal{V}^1 \tag{15}$$

where we use the additional subscript υ to illustrate that the value function is obtained under the VMC rule. $\upsilon_{t_{k+1}}$ is the PCP corresponding to $F(\boldsymbol{x}_{t_k}^\upsilon, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}_{t_k}^\upsilon, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}_{\pi,t_k}^\upsilon$ and is calculated using (12).

ID-VMC-VI solves (15) for the optimal value function $V_{\pi\upsilon}^*$ in the 1-dimensional PCP space. After this reduction step, the search space dimension is further reduced from n in ID-VI to 1.

For a discrete and finite $\mathcal{V}^1$, ID-VMC-VI is guaranteed to converge. Once $V_{\pi\upsilon}^*$ is reached, we search $\mathcal{V}^1$ for the predicted optimal PCP-to-go from $\boldsymbol{x}_{t_i}$ as

$$\upsilon_{t_{i+l}}^* = \underset{\upsilon_{t_{i+l}} \in \mathcal{V}^1}{\operatorname{argmin}} \left\{ \sum_{p=0}^{l-1} L(\boldsymbol{x}_{t_{i+p}}, \pi(\boldsymbol{x}_{t_{i+p}}, \boldsymbol{x}_{t_{i+l}}^\upsilon, \hat{\boldsymbol{\theta}}_{t_i})) + V_{\pi\upsilon}^*(\upsilon_{t_{i+l}}) \right\} \tag{16}$$

where $\boldsymbol{x}_{t_{i+l}}^\upsilon = \boldsymbol{x}(\upsilon_{t_{i+l}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})$. l is the number of steps taken to reach $\boldsymbol{x}_{t_{i+l}}^\upsilon$ from $\boldsymbol{x}_{t_i}$, and l is free. The l-step lookahead (Sutton, 1988) is warranted by not only control constraints, but

also the fact that $\boldsymbol{x}_{t_i}$ may not belong to $\mathcal{S}^n_{\upsilon,t_i}$. Either way, the l-step lookahead is necessary for finding the optimal PCP-to-go over the 1-dimensional PCP space. Similar to ID-VI, the control and state constraints are considered in the l-step lookahead in (16).

Remark 3.2: Because of the state constraints, the optimal PCP-to-go changes as the system state propagates, even if the VMC subspace stays unchanged. Choosing $\upsilon^*_{t_{i+l}}$ as the optimal PCP-to-go at t_i does not necessarily mean that the state trajectory will eventually reach $\boldsymbol{x}(\upsilon^*_{t_{i+l}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})$ at t_{i+l}.

Substituting (16) into (11), the optimal l-step state-to-reach predicted by ID-VMC-VI is

$$\boldsymbol{x}^{\upsilon*}_{t_{i+l}} = \boldsymbol{x}(\upsilon^*_{t_{i+l}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i}). \tag{17}$$

The ID-VMC-VI based control $\boldsymbol{u}^{\upsilon*}_{t_i}$ is then calculated by substituting (17) as the desired state-to-reach from $\boldsymbol{x}_{t_i}$ into (7), i.e.

$$\boldsymbol{u}^{\upsilon*}_{t_i} = \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}(\upsilon^*_{t_{i+l}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i}), \hat{\boldsymbol{\theta}}_{t_i}). \tag{18}$$

3.3 Stability analysis

Proposition 3.1: *At t_i, under the assumption that $\hat{\boldsymbol{\theta}}_{t_i} = \boldsymbol{\theta}$, $\boldsymbol{x}_{d,t_i}$ is the asymptotically stable point of the system* (1) *under control* (18).

Proof: Let $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{d,t_i}$. It is straight forward to get $\boldsymbol{x}_{d,t_i} = F(\boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}^{\upsilon*}_{t_i}$. Therefore, $\boldsymbol{x}_{d,t_i}$ is an equilibrium point of the system (1) under control (18).

At t_i, the optimal value function $V^*_{\pi\upsilon}$ is stationary. From (16), we define the value function of state $\boldsymbol{x}_{t_i}$ under control (18) as

$$\begin{aligned} V_{\pi\upsilon}(\boldsymbol{x}_{t_i}) &\triangleq \sum_{p=0}^{l-1} L(\boldsymbol{x}_{t_{i+p}}, \pi(\boldsymbol{x}_{t_{i+p}}, \boldsymbol{x}^{\upsilon*}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\upsilon^*_{t_{i+l}}) \\ &= L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon*}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) \\ &\quad + \sum_{p=1}^{l-1} L(\boldsymbol{x}_{t_{i+p}}, \pi(\boldsymbol{x}_{t_{i+p}}, \boldsymbol{x}^{\upsilon*}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\upsilon^*_{t_{i+l}}), \end{aligned} \tag{19}$$

in which

$$\begin{aligned} &\sum_{p=1}^{l-1} L(\boldsymbol{x}_{t_{i+p}}, \pi(\boldsymbol{x}_{t_{i+p}}, \boldsymbol{x}^{\upsilon*}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\upsilon^*_{t_{i+l}}) \\ &\overset{l_2=l-1}{=} \sum_{p=0}^{l_2-1} L(\boldsymbol{x}_{t_{i+1+p}}, \pi(\boldsymbol{x}_{t_{i+1+p}}, \boldsymbol{x}^{\upsilon*}_{t_{i+1+l_2}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\upsilon^*_{t_{i+1+l_2}}) \\ &\geq \min_{\upsilon_{t_{i+1+l_3}} \in \mathcal{V}^1} \left\{ \sum_{p=0}^{l_3-1} L(\boldsymbol{x}_{t_{i+1+p}}, \pi(\boldsymbol{x}_{t_{i+1+p}}, \boldsymbol{x}^{\upsilon}_{t_{i+1+l_3}}, \hat{\boldsymbol{\theta}}_{t_i})) \right. \\ &\quad \left. + V^*_{\pi\upsilon}(\upsilon_{t_{i+1+l_3}}) \right\} \\ &= V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}). \end{aligned} \tag{20}$$

From the definition of the step cost function following (3), it is clear that $V_{\pi\upsilon}(\boldsymbol{x}_{t_i}) \geq 0$, $\forall\, \boldsymbol{x}_{t_i} \in \mathcal{S}^n$, with $V_{\pi\upsilon}(\boldsymbol{x}_{t_i}) = 0$ only when $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{d,t_i}$.

Combining (19) and (20) yields

$$\begin{aligned} &V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_i}) \leq -L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon*}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) \leq 0, \\ &\quad \forall\, \boldsymbol{x}_{t_i} \in \mathcal{S}^n. \end{aligned} \tag{21}$$

The second equality in (21) holds only when $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{d,t_i}$. Therefore, $V_{\pi\upsilon}(\boldsymbol{x}_{t_k})$ is a Lyapunov function for the system (1) under control (18) at t_i. Based on the discrete-time Lyapunov theorem (Luenberger, 1979) and the discrete-time LaSalle's invariance principle (Sundarapandian, 2003), the equilibrium point $\boldsymbol{x}_{d,t_i}$ is asymptotically stable. ■

3.4 Optimality analysis

Lemma 3.2: *When applied to solve **C1**, $\boldsymbol{u}^{s*}_{t_i}$ is suboptimal to $\boldsymbol{u}^*_{t_i}$.*

Proof: From (5), we have

$$\begin{aligned} V^*(\boldsymbol{x}_{t_i}) &= L(\boldsymbol{x}_{t_i}, \boldsymbol{u}^*_{t_i}) + V^*(F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}^*_{t_i}) \\ &\leq L(\boldsymbol{x}_{t_i}, \boldsymbol{u}^{s*}_{t_i}) + V^*(F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}^{s*}_{t_i}). \end{aligned} \tag{22}$$

Therefore, $\boldsymbol{u}^{s*}_{t_i}$ is suboptimal to $\boldsymbol{u}^*_{t_i}$. ■

As mentioned earlier, the selection of reference point in the second dimension reduction step is mainly exploited for addressing the state constraints. For the purpose of optimality analysis, $\boldsymbol{x}_{t_i}$ will be regarded as the reference point in this subsection.

Lemma 3.3: *When applied to solve **C1**, $\boldsymbol{u}^{\upsilon*}_{t_i}$ is suboptimal to $\boldsymbol{u}^{s*}_{t_i}$.*

Proof: Choosing $\boldsymbol{x}_{ref,t_i} = \boldsymbol{x}_{t_i}$, then from (9), we have

$$\boldsymbol{x}^{s*}_{t_{i+1}} = \underset{\boldsymbol{x}_{t_{i+1}} \in \mathcal{S}^n}{\arg\min} \left\{ L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi}(\boldsymbol{x}_{t_{i+1}}) \right\}, \tag{23}$$

and from (16) and (17), we have

$$\begin{aligned} \boldsymbol{x}^{\upsilon*}_{t_{i+1}} &= \underset{\boldsymbol{x}^{\upsilon}_{t_{i+1}} \in \mathcal{S}^n_{\upsilon,t_i}}{\arg\min} \left\{ L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\boldsymbol{x}^{\upsilon}_{t_{i+1}}) \right\} \\ &\geq \underset{\boldsymbol{x}^{\upsilon}_{t_{i+1}} \in \mathcal{S}^n_{\upsilon,t_i}}{\arg\min} \left\{ L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi}(\boldsymbol{x}^{\upsilon}_{t_{i+1}}) \right\}. \end{aligned} \tag{24}$$

Because $\mathcal{S}^n_{\upsilon,t_i} \subseteq \mathcal{S}^n$, comparing (23) and (24) yields

$$\begin{aligned} &L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{s*}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi}(\boldsymbol{x}^{s*}_{t_{i+1}}) \\ &\quad \leq L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon*}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\boldsymbol{x}^{\upsilon*}_{t_{i+1}}). \end{aligned} \tag{25}$$

From (10) and (18), we have $\boldsymbol{u}^{s*}_{t_i} = \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{s*}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})$ and $\boldsymbol{u}^{\upsilon*}_{t_i} = \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon*}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})$. Therefore, $\boldsymbol{u}^{\upsilon*}_{t_i}$ is suboptimal to $\boldsymbol{u}^{s*}_{t_i}$. ■

Corollary 3.4: *For $\forall\, \boldsymbol{x}^{\upsilon}_{t_{i+1}} \in \mathcal{S}^n_{\upsilon,t_i}$, $\pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})$ is suboptimal to $\boldsymbol{u}^{\upsilon*}_{t_i}$ in solving* **C1**.

Proof: The following inequality is a direct result from (24)

$$L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon*}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\boldsymbol{x}^{\upsilon*}_{t_{i+1}}) \leq L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\boldsymbol{x}^{\upsilon}_{t_{i+1}}), \tag{26}$$

which proves the statement. ■

Corollary 3.5: *If $\upsilon \in [0, 1]$, the optimality of using $\boldsymbol{u}^{\upsilon*}_{t_i}$ in **C1** is lower bounded by that of a first-order, exponential decay state tracking controller.*

Proof: A first-order state trajectory in the VMC subspace can be formulated as Ogata (2010)

$$\boldsymbol{x}_{t_k} = \boldsymbol{x}_d - (\boldsymbol{x}_d - \boldsymbol{x}_i)e^{a(t_i - t_k)}, \quad k = i, \ldots, N-1 \tag{27}$$

with $1/a > 0$ being the time constant.

Substituting (27) into (12), the PCP-to-go can be derived as

$$\upsilon_{t_k} = 1 - e^{a(t_i - t_k)}, \quad k = i, \ldots, N-1. \tag{28}$$

It is clear from (28) that $0 < \upsilon_{t_{i+1}} < 1$, $\forall\ a > 0$. Based on Corollary 3.4, the tracking controller following this predefined PCP is suboptimal to $\boldsymbol{u}^{\upsilon*}_{t_i}$. ■

Corollary 3.6: *If υ can be larger than 1, the optimality of using $\boldsymbol{u}^{\upsilon*}_{t_i}$ is lower bounded by that of a second-order state trajectory tracking controller.*

Proof: A second-order state trajectory can be formulated as Ogata (2010)

$$\boldsymbol{x}_{t_k} = \boldsymbol{x}_d - (\boldsymbol{x}_d - \boldsymbol{x}_i)\frac{e^{\omega_d(t_i - t_k)\zeta/\sqrt{1-\zeta^2}}}{\sqrt{1-\zeta^2}}\sin\chi, \quad 0 < \zeta < 1,\ k = i, \ldots, N-1 \tag{29}$$

where $\chi = \omega_d(t_k - t_i) + \tan^{-1}(\sqrt{1-\zeta^2}/\zeta)$. ζ is the damping ratio and ω_d is the damped natural frequency.

The PCP-to-go following this second-order trajectory can be derived as

$$\upsilon_{t_k} = 1 - \frac{e^{\omega_d(t_i - t_k)\zeta/\sqrt{1-\zeta^2}}}{\sqrt{1-\zeta^2}} \cdot [\sin\omega_d(t_k - t_i)\cot\chi + \cos\chi], \quad 0 < \zeta < 1,\ k = i, \ldots, N-1. \tag{30}$$

Choosing $k = i+1$ in (30) gives the PCP-to-go $\upsilon_{t_{i+1}}$. Following Corollary 3.4, a controller tracking this predefined PCP is suboptimal to $\boldsymbol{u}^{\upsilon*}_{t_i}$. ■

Remark 3.3: The performance advantage of the proposed ID-VMC-VI control scheme over typical trajectory tracking controllers lies in the optimisation over PCPs. Furthermore, in ID-VMC-VI, we have the flexibility to choose the reference point $\boldsymbol{x}_{ref,t_i}$ to address state constraints.

3.5 Dimension analysis

Following the two-step search space dimension reduction strategy, the dimension of search space for the optimal control is reduced from $m+n$ in basic VI to n in ID-VI, and eventually reduced to 1 in ID-VMC-VI.

Assume that there are D discretised nodes per dimension in the search space. Because the size of a tabular value function is proportional to the number of discretised nodes in the search space, the memory consumption of ID-VMC-VI will be D^{n-1} and D^{m+n-1} times less than those of ID-VI and VI, respectively.

Even though the computational cost of a VI algorithm depends on many factors besides the size of the search space and no rigid relationship can be drawn between the two, ID-VMC-VI still dramatically reduces the computational cost compared with ID-VI and the basic VI, as will be shown in the simulation results.

4. ID policy improvement

4.1 Uncertain parameter adaptation via gradient descent

E1: The problem of estimating the uncertain parameter $\boldsymbol{\theta}$ to improve the ID policy, as stated in Section 2, is solved by the gradient descent method (Curry, 1944)

$$\hat{\boldsymbol{\theta}}_{t_{i+1}} = \hat{\boldsymbol{\theta}}_{t_i} - \boldsymbol{\alpha}\frac{\mathrm{d}e_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} \tag{31}$$

where $\boldsymbol{\alpha}$ is the step size matrix. $\mathrm{d}e_{t_{i+1}}/\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}$ is derived as

$$\frac{\mathrm{d}e_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}\frac{\mathrm{d}e_{t_{i+1}}}{\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}} = \frac{\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}\tilde{\boldsymbol{x}}_{t_{i+1}} \tag{32}$$

where

$$\frac{\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} - \frac{\mathrm{d}\hat{\boldsymbol{x}}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}. \tag{33}$$

Given $\boldsymbol{x}_{t_i}$, $\boldsymbol{x}_{t_{i+1}} = F(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}) + G(\boldsymbol{x}_{t_i}, \boldsymbol{\theta})\boldsymbol{u}_{t_i}$, where $\boldsymbol{u}_{t_i} = G^{-1}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})[\hat{\boldsymbol{x}}_{t_{i+1}} - F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})]$. The derivative of $\boldsymbol{x}_{t_{i+1}}$ with respect to $\hat{\boldsymbol{\theta}}_{t_i}$ via the chain rule is

$$\frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\mathrm{d}\boldsymbol{u}_{t_i}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}\frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}} = \left(\frac{\partial\boldsymbol{u}_{t_i}}{\partial\hat{\boldsymbol{\theta}}_{t_i}} + \frac{\mathrm{d}\hat{\boldsymbol{x}}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}\frac{\partial\boldsymbol{u}_{t_i}}{\partial\hat{\boldsymbol{x}}_{t_{i+1}}}\right)\frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}} \tag{34}$$

where

$$\frac{\partial\boldsymbol{u}_{t_i}}{\partial\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\mathrm{d}G^{-1}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}[\hat{\boldsymbol{x}}_{t_{i+1}} - F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})] - \left[\frac{\mathrm{d}F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}}\right]^T[G^{-1}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})]^T. \tag{35}$$

On the other hand, $\hat{\boldsymbol{x}}_{t_{i+1}}$ satisfies $\hat{\boldsymbol{x}}_{t_{i+1}} = F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}_{t_i}$, into which substituting $\boldsymbol{u}_{t_i} = G^{-1}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})[\hat{\boldsymbol{x}}_{t_{i+1}} - F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})]$

yields $\hat{\boldsymbol{x}}_{t_{i+1}} = \hat{\boldsymbol{x}}_{t_{i+1}}$. Therefore, $\hat{\boldsymbol{x}}_{t_{i+1}}$ is invariant with respect to $\hat{\boldsymbol{\theta}}_{t_i}$, i.e. $d\hat{\boldsymbol{x}}_{t_{i+1}}/d\hat{\boldsymbol{\theta}}_{t_i} = \mathbf{0}$. Hence, from (33),

$$\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}/\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i} = \mathrm{d}\boldsymbol{x}_{t_{i+1}}/\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}, \tag{36}$$

and from (34),

$$\frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\partial \boldsymbol{u}_{t_i}}{\partial \hat{\boldsymbol{\theta}}_{t_i}} \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}}. \tag{37}$$

Combining (37) and (36), we have

$$\frac{\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\partial \boldsymbol{u}_{t_i}}{\partial \hat{\boldsymbol{\theta}}_{t_i}} \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}}. \tag{38}$$

Substituting (38) into (32), we have

$$\frac{\mathrm{d}e_{t_{i+1}}}{\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i}} = \frac{\partial \boldsymbol{u}_{t_i}}{\partial \hat{\boldsymbol{\theta}}_{t_i}} \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}} \tilde{\boldsymbol{x}}_{t_{i+1}}, \tag{39}$$

which, when substituted into (31), yields

$$\hat{\boldsymbol{\theta}}_{t_{i+1}} = \hat{\boldsymbol{\theta}}_{t_i} - \boldsymbol{\alpha} \frac{\partial \boldsymbol{u}_{t_i}}{\partial \hat{\boldsymbol{\theta}}_{t_i}} \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}} \tilde{\boldsymbol{x}}_{t_{i+1}}. \tag{40}$$

4.2 Convergence analysis

Proposition 4.1: *Following the gradient method* (40)*, the adaptation parameter* $\hat{\boldsymbol{\theta}}$ *converges to the actual value* $\boldsymbol{\theta}$ *if the system is in a transient stage.*

Proof: According to Curry (1944), the gradient descent method stops only when a stationary point is reached, i.e. when

$$\mathrm{d}e_{t_{i+1}}/\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i} = \mathbf{0}. \tag{41}$$

Given the fact that the system state is affected by the uncertain parameter, $\mathrm{d}\tilde{\boldsymbol{x}}_{t_{i+1}}/\mathrm{d}\hat{\boldsymbol{\theta}}_{t_i} \neq \mathbf{0}$. Therefore, we know from (32) that the stopping condition (41) holds true only if $\tilde{\boldsymbol{x}}_{t_{i+1}} = \mathbf{0}$. For a system in transient stages, this condition can be elaborated as

$$\begin{aligned} \tilde{\boldsymbol{x}}_{t_{i+1}} &= [F(\boldsymbol{x}_{t_i}, \boldsymbol{\theta}) - F(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})] \\ &\quad + [G(\boldsymbol{x}_{t_i}, \boldsymbol{\theta})\boldsymbol{u}_{t_i} - G(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i})]\boldsymbol{u}_{t_i} = \mathbf{0}, \\ &\quad \forall\, \boldsymbol{x}_{t_i} \in \mathcal{S}^n, \end{aligned} \tag{42}$$

For a general system, we assume there are an infinite number of states in the state space, therefore, Equation (42) holds true only if $\hat{\boldsymbol{\theta}}_{t_i} = \boldsymbol{\theta}$. ■

4.3 Stability condition

To discuss the influence of parameter adaptation on the stability of the closed-loop system, $V(\cdot, \hat{\boldsymbol{\theta}})$ is used to denote a value function attained under the estimate $\hat{\boldsymbol{\theta}}$. Definition of $V_{\pi\upsilon}(\boldsymbol{x}, \hat{\boldsymbol{\theta}})$ follows (19).

Proposition 4.2: *Assuming that there exists a vector* $\boldsymbol{\kappa} \in \mathbb{R}^r$*, such that* $\max_{\boldsymbol{x}_{t_k} \in \mathcal{S}^n}[V_{\pi\upsilon}(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_i})] \leq \boldsymbol{\kappa}^T (\hat{\boldsymbol{\theta}}_{t_{i+1}} - \hat{\boldsymbol{\theta}}_{t_i})$*, $k \geq i$, then* $(\boldsymbol{x}_{d,t_i}, \boldsymbol{\theta})$ *is the asymptotically stable point of the system* (1) *under control* (18) *and parameter adaptation* (40) *if the parameter update step size* $\boldsymbol{\alpha}$ *satisfies*

$$\boldsymbol{\kappa}^T \boldsymbol{\alpha} \frac{\partial \boldsymbol{u}_{t_i}}{\partial \hat{\boldsymbol{\theta}}_{t_i}} \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}} \tilde{\boldsymbol{x}}_{t_{i+1}} + L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) \geq 0 \tag{43}$$

with the equality holds only when $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{d,t_i}$.

Proof: Proof of Proposition 4.2 follows a similar idea of Proposition 3.1. It is clear from Propositions 3.1 and 4.1 that $(\boldsymbol{x}_{d,t_i}, \boldsymbol{\theta})$ is an equilibrium point of the overall system.

From the definition of the step cost function following (3), $V_{\pi\upsilon}(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h}) \geq 0$, $\forall\ (\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h}) \in \{(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h}) | \boldsymbol{x}_{t_k} \in \mathcal{S}^n, \hat{\boldsymbol{\theta}}_{t_h} \in \mathcal{P}^r\}$, $k \geq i$, $h \geq i$. $V_{\pi\upsilon}(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h}) = 0$ only when $\boldsymbol{x}_{t_k} = \boldsymbol{x}_{d,t_i}$.

From (19), we have

$$\begin{aligned} V_{\pi\upsilon}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) &= L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) \\ &\quad + \sum_{p=1}^{l-1} L(\boldsymbol{x}_{t_{i+p}}, \pi(\boldsymbol{x}_{t_{i+p}}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) + V^*_{\pi\upsilon}(\upsilon^*_{t_{i+l}}) \\ &= L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) + V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i}). \end{aligned} \tag{44}$$

From (44) and the assumption of Proposition 4.2, we have

$$\begin{aligned} &V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) \\ &\quad = V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_i}) \\ &\qquad - L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) \\ &\quad \leq \boldsymbol{\kappa}^T (\hat{\boldsymbol{\theta}}_{t_{i+1}} - \hat{\boldsymbol{\theta}}_{t_i}) - L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})). \end{aligned} \tag{45}$$

Substituting (40) into (45), we have

$$\begin{aligned} &V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) \\ &\quad \leq -\boldsymbol{\kappa}^T \boldsymbol{\alpha} \frac{\partial \boldsymbol{u}_{t_i}}{\partial \hat{\boldsymbol{\theta}}_{t_i}} \frac{\mathrm{d}\boldsymbol{x}_{t_{i+1}}}{\mathrm{d}\boldsymbol{u}_{t_i}} \tilde{\boldsymbol{x}}_{t_{i+1}} - L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})), \end{aligned} \tag{46}$$

from which it is clear that (43) is the condition for $V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) \leq 0$. When $\boldsymbol{x}_{t_i} = \boldsymbol{x}_{d,t_i}$, $L(\boldsymbol{x}_{t_i}, \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}^{\upsilon *}_{t_{i+l}}, \hat{\boldsymbol{\theta}}_{t_i})) = 0$. Thus, based on Proposition 4.1, $V_{\pi\upsilon}(\boldsymbol{x}_{t_{i+1}}, \hat{\boldsymbol{\theta}}_{t_{i+1}}) - V_{\pi\upsilon}(\boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}) = 0$ only if $\hat{\boldsymbol{\theta}}_{t_i} = \boldsymbol{\theta}$. Therefore, $V_{\pi\upsilon}(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h})$, $\forall\ (\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h}) \in \{(\boldsymbol{x}_{t_k}, \hat{\boldsymbol{\theta}}_{t_h}) | \boldsymbol{x}_{t_k} \in \mathcal{S}^n, \hat{\boldsymbol{\theta}}_{t_h} \in \mathcal{P}^r\}$, $k \geq i$, $h \geq i$, is a Lyapunov function for the system (1) under control (18) and parameter adaptation (40), and the equilibrium point $(\boldsymbol{x}_{d,t_i}, \boldsymbol{\theta})$ is asymptotically stable (Luenberger, 1979; Sundarapandian, 2003). ■

5. Algorithm

The ID-VMC-VI algorithm is summarised in Algorithm 1. All related equations have been discussed before. It is worth mentioning that the algorithm needs to run at every time step t_i to take time-varying properties, such as state constraints, into consideration, as well as to update the estimate of uncertain parameter.

There are several guidelines to help efficiently implement the algorithm. (1) A small step size is preferred to achieve better

Algorithm 1 ID-VMC-VI Algorithm at t_i

1: Given $\boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{t_i}, \hat{\boldsymbol{\theta}}_{t_i}, \boldsymbol{\alpha}, \mathcal{V}^1$
2: Set $V_{\pi\upsilon,0}(\upsilon) = 0, \forall\, \upsilon \in \mathcal{V}^1, j = 1, flag = 0$
3: **while** $flag = 0$ **do**
4: **for** $\forall\, \upsilon_{t_k} \in \mathcal{V}^1 \setminus \{\upsilon_d\}$ **do**
5: Calculate $\boldsymbol{x}^{\upsilon}_{t_k} = \boldsymbol{x}(\upsilon_{t_k}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})$
6: Calculate $\boldsymbol{u}^{\upsilon}_{\pi,t_k} = \pi(\boldsymbol{x}^{\upsilon}_{t_k}, \boldsymbol{x}_{d,t_i}, \hat{\boldsymbol{\theta}}_{t_i})$
7: Propagate $\boldsymbol{x}^{\upsilon}_{t_{k+1}} = F(\boldsymbol{x}^{\upsilon}_{t_k}, \hat{\boldsymbol{\theta}}_{t_i}) + G(\boldsymbol{x}^{\upsilon}_{t_k}, \hat{\boldsymbol{\theta}}_{t_i})\boldsymbol{u}^{\upsilon}_{\pi,t_k}$
8: Calculate $\upsilon_{t_{k+1}} = \upsilon(\boldsymbol{x}^{\upsilon}_{t_{k+1}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})$
9: $V_{\pi\upsilon,j}(\upsilon_{t_k}) \leftarrow L(\boldsymbol{x}^{\upsilon}_{t_k}, \boldsymbol{u}^{\upsilon}_{\pi,t_k}) + V_{\pi\upsilon,j-1}(\upsilon_{t_{k+1}})$
10: **end for**
11: **if** $V_{\pi\upsilon,j}(\upsilon) = V_{\pi\upsilon,j-1}(\upsilon), \forall\, \upsilon \in \mathcal{V}^1$ **then**
12: Set $flag = 1$
13: **else**
14: $j \leftarrow j + 1$
15: **end if**
16: **end while**
17: Calculate $\upsilon^*_{t_{i+l}}$ following (16)
18: Calculate $\boldsymbol{x}^{\upsilon*}_{t_{i+l}} = \boldsymbol{x}(\upsilon^*_{t_{i+l}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i})$
19: Calculate $\boldsymbol{u}^{\upsilon*}_{t_i} = \pi(\boldsymbol{x}_{t_i}, \boldsymbol{x}(\upsilon^*_{t_{i+l}}, \boldsymbol{x}_{d,t_i}, \boldsymbol{x}_{ref,t_i}), \hat{\boldsymbol{\theta}}_{t_i})$
20: Observe state feedback $\boldsymbol{x}_{t_{i+1}}$
21: Calculate $\tilde{\boldsymbol{x}}_{t_{i+1}} = \boldsymbol{x}_{t_{i+1}} - \boldsymbol{x}^{\upsilon*}_{t_{i+1}}$
22: Update $\hat{\boldsymbol{\theta}}_{t_{i+1}} \leftarrow \hat{\boldsymbol{\theta}}_{t_i} - \boldsymbol{\alpha}\nabla_{\hat{\boldsymbol{\theta}}_{t_i}} e_{t_{i+1}}$

performance near the border of state and control constraints. (2) The solution optimality depends on the selection of reference point and prey motion (i.e. desired state $\boldsymbol{x}_{d,t_i}$). In general, the reference point and desired state $\boldsymbol{x}_{d,t_i}$ are chosen so that the VMC subspace does not violate the state constraints. However, in case of complicated state constraints, an intermediate desired state-to-reach can be adopted to break down the problem into sub-problems. (3) Control constraints should be accounted for in the state propagation in Line 9 of the above algorithm. (4) Both state and control constraints should be accounted for in the l-step lookahead in Line 14 of the algorithm.

Remark 5.1: We reckon that the ID-VMC-VI algorithm can be implemented in real-time if the CPU time to execute the algorithm is less than the control time step size.

6. Simulation validation and discussion

Two examples are simulated to show the computational cost and memory usage advantages of the ID-VMC-VI method. Example 1 solves a regulating problem with control constraints. Example 2 solves an optimal trajectory planning problem with both state and control constraints. While the first example is adopted and refined from Li and Xu (2020), the improved algorithm and the new ID policy improvement method from this paper are applied. Furthermore, the simulation parameters and settings are tuned to achieve better performance. Since the estimates of uncertain parameters are updated at every time step, the VI algorithm needs to be implemented in real time. Theoretically, the classical DP based VI method (VI) will generate the optimal solutions, however, its high computational cost and a huge amount of memory requirement prohibit it from real-time application. Therefore, we show the comparison of VI, ID-VI, and ID-VMC-VI methods only in Example 1; while in Example 2, ID-VI and ID-VMC-VI are compared. The implementation of the ID-VI method has two differences from the ID-VMC-VI method. First, during the value iteration, the VMC rule is not followed. Second, (9) is followed instead of (16) to calculate the predicted optimal state-to-reach.

6.1 Example 1: angular velocity control

Simulation 1 is conducted on a laptop computer with a 3.7 GHz CPU and a 64 GB RAM. This example is adopted and refined from Li and Xu (2020). A UAV attitude dynamics is governed by Mahony et al. (2012)

$$\boldsymbol{\omega}_{t_{i+1}} = \boldsymbol{\omega}_{t_i} - T\mathbf{I}^{-1}\left(\boldsymbol{\omega}_{t_i} \times \mathbf{I}\boldsymbol{\omega}_{t_i}\right) + T\mathbf{I}^{-1}\boldsymbol{u}_{t_i} \tag{47}$$

where $\boldsymbol{\omega}_{t_i} \in \mathcal{S}^3$ is the UAV body frame angular velocity. $\boldsymbol{u}_{t_i} \in \mathcal{A}^3$ is the control torque imposed on the centre of mass of the UAV. Subscripts x, y and z are used to denote the components of $\boldsymbol{\omega}_{t_i}$ and $\boldsymbol{u}_{t_i}$ along the directions of roll, pitch, and yaw. $\mathbf{I}$ is the moment of inertia matrix. It is invertible, but not perfectly known. T is the time step size. We assume the initial guess of $\mathbf{I}$ to be $\hat{\mathbf{I}} = \text{diag}\{0.07, 0.07, 0.1\}\,\text{kg}\cdot\text{m}^2$ and the real values to be

$$\mathbf{I} = \begin{bmatrix} 0.09 & 0 & 0.01 \\ 0 & 0.05 & 0 \\ 0.01 & 0 & 0.13 \end{bmatrix} \text{kg}\cdot\text{m}^2. \tag{48}$$

The proposed ID-VMC-VI algorithm is tasked to follow the step angular velocity command. The step cost function is

$$L(\boldsymbol{\omega}_{t_i}, \boldsymbol{u}_{t_i}) = (\boldsymbol{\omega}_{d,t_i} - \boldsymbol{\omega}_{t_i})^{\mathrm{T}}\mathbf{Q}(\boldsymbol{\omega}_{d,t_i} - \boldsymbol{\omega}_{t_i}) + \boldsymbol{u}_{t_i}{}^{\mathrm{T}}\mathbf{R}\boldsymbol{u}_{t_i} \tag{49}$$

where the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ are picked to be $\text{diag}\{1, 1, 1\}$ and $\text{diag}\{0.01, 0.01, 0.01\}$. To test the ability of the proposed method in handling control constraints, the control limits are assumed to be ± 0.8 Nm in all three directions.

The time step size is set to be 0.1 s in the simulation. For the purpose of comparison, the 6-dimensional state and action spaces of VI, 3-dimensional state space of ID-VI, and the 1-dimensional PCP space of ID-VMC-VI are discretised into 41^6 nodes, 41^3 nodes, and 41 nodes, respectively.

The step signals last 20 s in all three directions. In the roll and pitch directions, the signal magnitudes are 2 rad/s. In the yaw direction, the magnitude is 1 rad/s. The simulation results show that all three methods can stabilise the angular velocities (shown in Figure 1), without violating the control constraints (shown in Figure 2). The benefit of ID-VI and VI being able to search the state space and state/action spaces yields a performance boost over ID-VMC-VI at the rising phase of the step response (shown in the zoomed plots in Figure 1). As the errors in $\hat{\mathbf{I}}$ converge to zeros (shown in Figure 3), the steady state errors in the step responses gradually decrease (shown in Figure 1), indicating that the control performance is being improved.

Table 1 shows the CPU time, memory usage, and performance indices for all three algorithms. On average, ID-VMC-VI is 654 times and 38,080 times faster than ID-VI and VI, respectively. It also uses 1607 times less memory with respect to ID-VI,

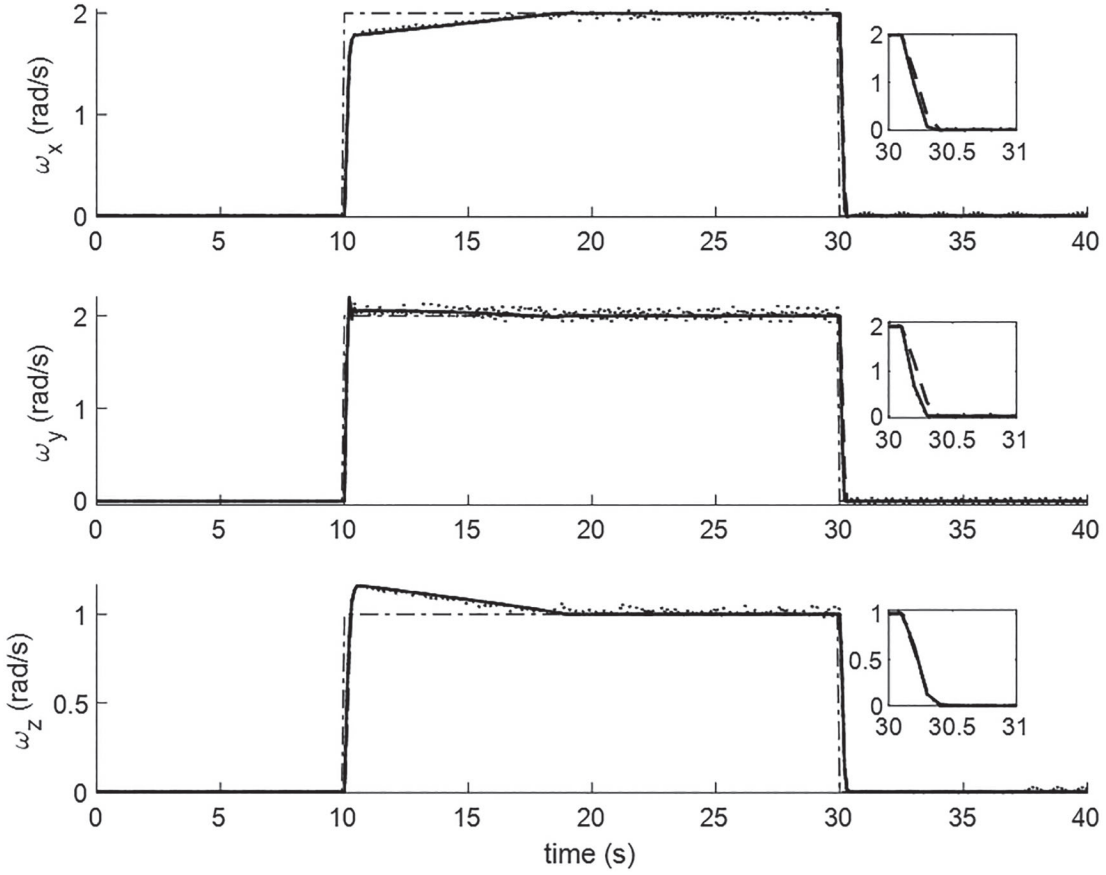

Figure 1. Comparison of the angular velocity step responses. (Dash-dotted lines: step signals; solid lines: ID-VI resulted step responses; dashed lines: ID-VMC-VI resulted step responses; and dotted lines: VI resulted step responses.)

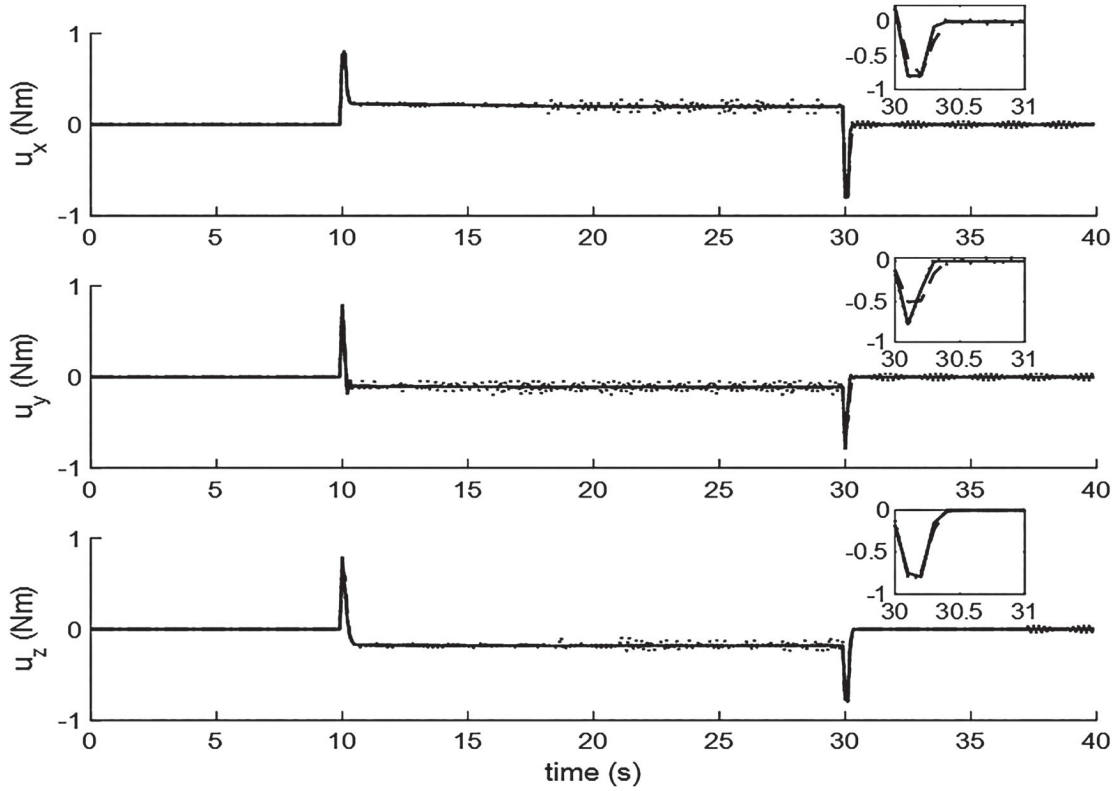

Figure 2. Comparison of the control commands in response to the step signals. (Solid lines: ID-VI control commands; dashed lines: ID-VMC-VI control commands; and dotted lines: VI control commands.)

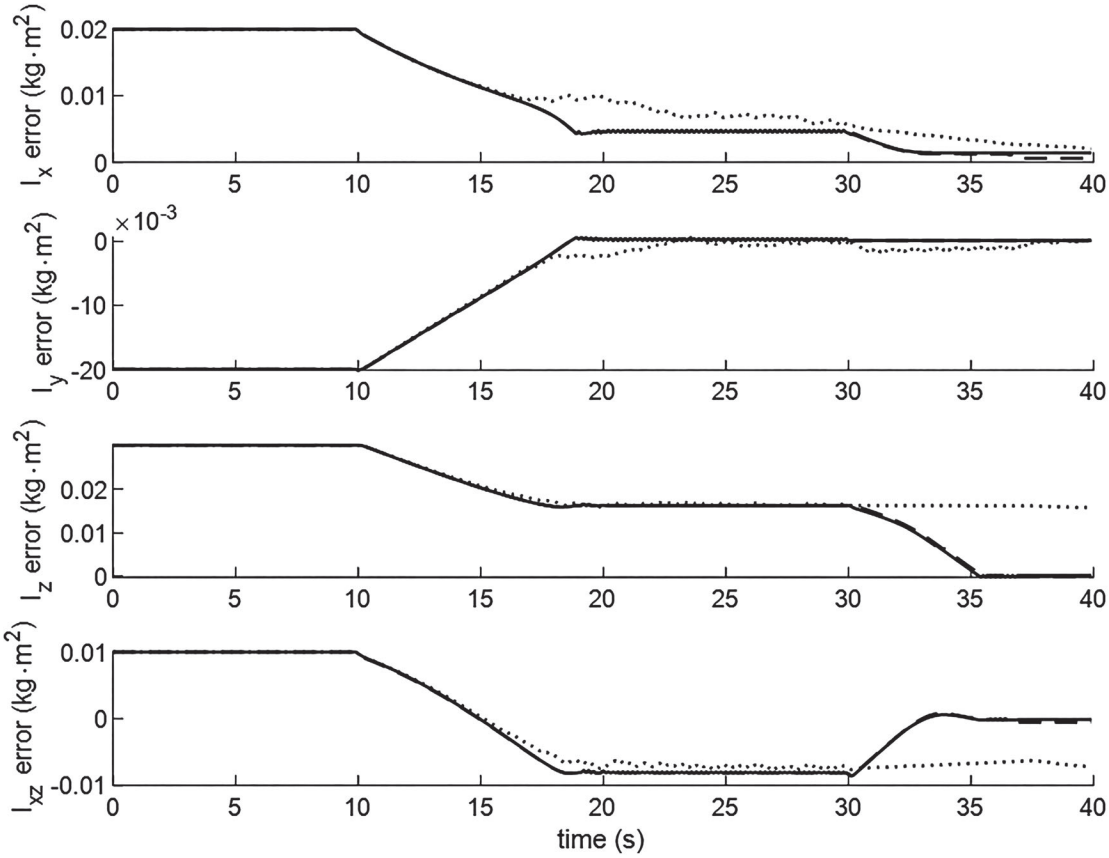

Figure 3. Comparison of the errors in estimation. (Solid lines: ID-VI; dashed lines: ID-VMC-VI; and dotted lines: VI)

Table 1. CPU time, memory usage and performance indices in Example 1.

Method	VI	ID-VI	ID-VMC-VI
Average CPU Time per simulation step (0.1 s)	23.8 s	0.409 s	0.625 ms
Memory usage	35.4 GB	3.15 MB	1.96 KB
Performance index	2.516	2.447	2.631

which is consistent with the analysis in Section 3.5. The memory usage of VI is huge, and is the main reason for a computer with a 64 GB RAM to be involved. This type of RAM requirement makes the VI algorithm practically infeasible for most mobile robotic platforms. In terms of performance index, the lesser the better. The performance index of ID-VMC-VI is 7.5% higher than that of ID-VI. Theroetically, VI should achieve the minimum performance index. However, practically more optimisable parameters may lead to challenges in convergence and the performance index might be higher. As we argued in Remark 5.1, it can be implemented in real time. One way to improve the performance of both ID-VI and ID-VMC-VI is to increase the number of discretised nodes in the search spaces. In contrast to the memory usage of ID-VI that grows exponentially with respect to the number of nodes in the search space, the memory usage of ID-VMC-VI is proportional to that number, which means that ID-VMC-VI is much more scalable. Thus, it can be implemented on platforms with limited memory resources and can be easily scaled up to provide a better performance.

6.2 Example 2: skid-steer vehicle minimum time-to-reach problem

Simulation 2 is conducted on a laptop computer (CPU: 2.6 GHz; memory: 16 GB). The point mass model of the vehicle is modified from Laumond et al. (1998) as

$$x_{t_{i+1}} = x_{t_i} + TV_{t_i}\cos\theta_{t_i}; \quad y_{t_{i+1}} = y_{t_i} + TV_{t_i}\sin\theta_{t_i};$$
$$\theta_{t_{i+1}} = \theta_{t_i} + T(\omega_{t_i} + b_c) \tag{50}$$

where (x_{t_i}, y_{t_i}) is the vehicle location in the x-y plane of an inertial north-east-down frame, of which the origin is on the ground surface. V_{t_i} is the velocity, while θ_{t_i} and ω_{t_i} are the heading angle and angular velocity around the z axis. $T = 0.1$ s is the time step size. With the vehicle initially facing north, $\theta_0 = 0^\circ$. A constant control input bias $b_c = -3^\circ/\text{s}$ is considered and estimated following the gradient based adaptation rule discussed in Section 4. The following constraints are used: $0 \le V \le 2$ m/s and $|\omega| \le 30^\circ/\text{s}$.

The proposed ID-VMC-VI algorithm is used to control the vehicle from the starting point $(0, 0)$ to the target point $(10, 10)$, while avoiding two circular obstacle regions O_1 and O_2, which are respectively centred at $(7, 7)$ and $(4, 3)$ with a radius of 2 m. The step cost function is $L = T$, where $T > 0$ is the constant time step size.

For the purpose of comparison, the 2-dimensional state space of ID-VI and the 1-dimensional PCP space of ID-VMC-VI are discretised into 100×100 nodes and 100 nodes. The

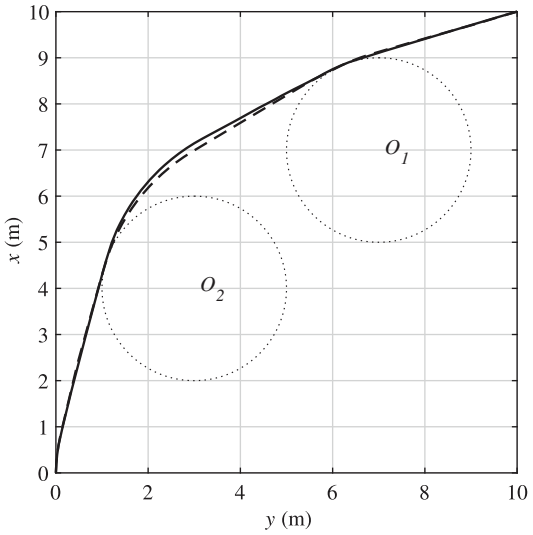

Figure 4. Comparison of the optimised paths. (Dotted line: obstacle; solid line: ID-VI optimised path; dashed line: ID-VMC-VI optimised path.)

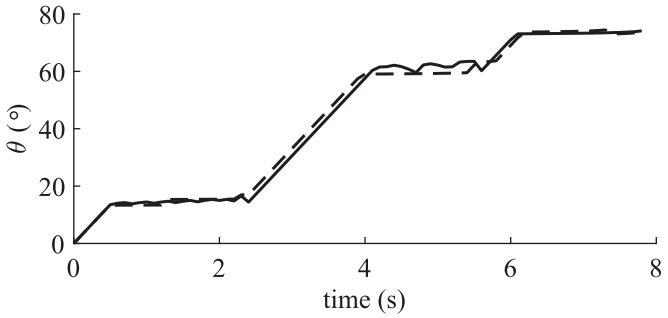

Figure 5. Comparison of the heading angles of the vehicle. (Solid line: ID-VI resulted heading angle; dashed line: ID-VMC-VI resulted heading angle.)

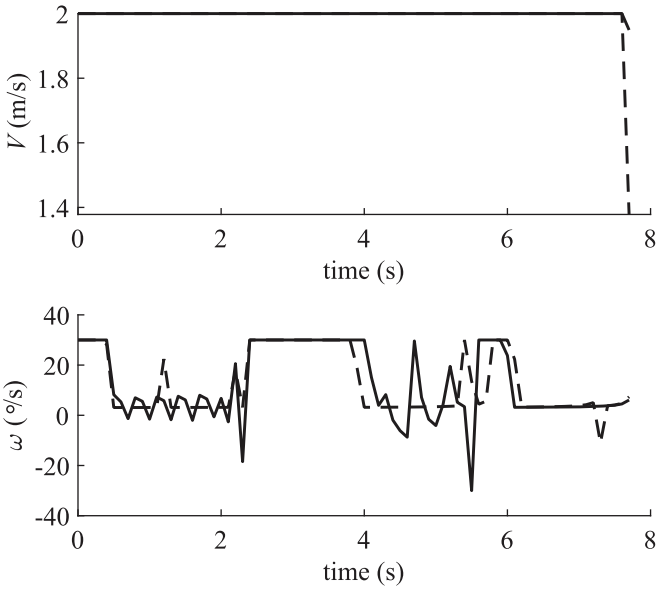

Figure 6. Comparison of the control variables. (Solid lines: ID-VI control commands; dashed lines: ID-VMC-VI control commands.)

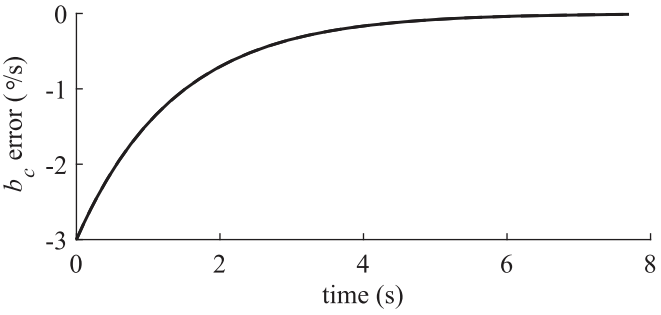

Figure 7. Comparison of the errors in the input bias estimation. (Solid lines: ID-VI resulted estimation error changes; dashed lines: ID-VMC-VI resulted estimation error changes. The solid line and the dashed line are almost identical.)

Table 2. CPU time, memory usage and performance indices in Example 2.

Method	ID-VI	ID-VMC-VI
Average CPU Time (s) per simulation step (0.1 s)	5.652	0.029
Memory usage (kilobyte)	234.375	2.344
Performance index	7.8	7.8

point where the upper tangent line of O_1 through the target point intersects $y = 0$ is used as the reference point.

As shown in Figures 4 and 5, the ID-VI and ID-VMC-VI methods generate very similar paths that can avoid the obstacles and reach the target location. The ID-VI method introduces oscillations in the heading angle control due to the discretization of 2-dimensional state space. On the other hand, the ID-VMC-VI method generates much smoother heading angle control commands as the benefit of searching the 1-dimensional PCP space.

On the other hand, as shown in Figure 6, both methods generate control signals that satisfy the control constraints. Figure 7 shows that the adopted gradient based adaptation rule can efficiently reduce the estimation error in the input bias following both methods.

Table 2 shows the comparison of CPU time, memory usage and performance between ID-VI and ID-VMC-VI. On average, the ID-VMC-VI algorithm is 194 times faster to compute and consumes 100 times less memory than ID-VI, while delivering about the same performance index in this example.

7. Conclusion

In this study, a dynamic programming search space dimension reduction strategy is proposed for a class of discrete-time dynamic systems. Two reduction steps are involved in the proposed value iteration algorithm: (1) eliminating the action search space using inverse dynamics; and (2) projecting the state search space into a 1-dimensional parameter space by the virtual motion camouflage rule. A gradient based uncertain parameter adaptation rule is developed to improve the inverse dynamics policy at each time step. Compared with typical dynamic programming based optimal controls, the proposed algorithm can dramatically reduce memory usage and computational time, and is thus feasible for real-time applications. Stability and optimality of the algorithm are analysed and its performance lower

bounds are given. Two simulation examples are used to illustrate the aforementioned salient features of the proposed algorithm.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the National Science Foundation [grant number 1924622].

References

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983, September–October). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*(5), 834–846. https://doi.org/10.1109/TSMC.1983.6313077

Bellman, R. (1954). Dynamic programming and a new formalism in the calculus of variations. *Proceedings of the National Academy of Sciences of the United States of America*, *40*(4), 231–235. https://doi.org/10.1073/pnas.40.4.231

Bellman, R. (1961). *Adaptive control processes: A guided tour*. Princeton University Press.

Bertsekas, D. P. (2013). Rollout algorithms for discrete optimization: A survey. In P. M. Pardalos, D. Du, & R. L. Graham (Eds.), *Handbook of combinatorial optimization* (pp. 2989–3013). Springer.

Bertsekas, D. P. (2020). *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Bertsekas, D. P., Tsitsiklis, J. N., & Wu, C. (1997). Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, *3*(3), 245–262. https://doi.org/10.1023/A:1009635226865

Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming*. Society for Industrial and Applied Mathematics.

Bhatnagar, S. (2010, December). An actor-critic algorithm with function approximation for discounted cost constrained Markov decision processes. *Systems & Control Letters*, *59*(12), 760–766. https://doi.org/10.1016/j.sysconle.2010.08.013

Borghese, N. A., & Arbib, M. A. (1995). Generation of temporal sequences using local dynamic programming. *Neural Networks*, *8*(1), 39–54. https://doi.org/10.1016/0893-6080(94)00053-O

Borkar, V. S. (2005, March). An actor-critic algorithm for constrained Markov decision processes. *Systems & Control Letters*, *54*(3), 207–213. https://doi.org/10.1016/j.sysconle.2004.08.007

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems* (pp. 369–376). MIT Press.

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*. CRC Press.

Chisci, L., Rossiter, J. A., & Rice, M. J. (1998, April). Stable GPC by dynamic programming. *Systems & Control Letters*, *33*(5), 291–300. https://doi.org/10.1016/S0167-6911(97)00116-3

Clarke, D. W., Mohtadi, C., & Tuffs, P. S. (1987, March). Generalized predictive control–Part I. The basic algorithm. *Automatica*, *23*(2), 137–148. https://doi.org/10.1016/0005-1098(87)90087-2

Curry, H. B. (1944). The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, *2*(3), 258–261. https://doi.org/10.1090/qam/1944-02-03

Forrest-Barlach, M., & Babcock, S. (1987). Inverse dynamics position control of a compliant manipulator. *IEEE Journal on Robotics and Automation*, *3*(1), 75–83. https://doi.org/10.1109/JRA.1987.1087072

Gros, S., & Schild, A. (2017, January). Real-time economic nonlinear model predictive control for wind turbine control. *International Journal of Control*, *90*(12), 2799–2812. https://doi.org/10.1080/00207179.2016.1266514

Howard, R. A. (1960). *Dynamic programming and markov processes*. MIT Press.

Kumar, R. R., & Seywald, H. (1996). Should controls be eliminated while solving optimal control problems via direct methods? *Journal of Guidance, Control, and Dynamics*, *19*(2), 418–423. https://doi.org/10.2514/3.21634

Laumond, J., Sekhavat, S., & Lamiraux, F. (1998). Guidelines in nonholonomic motion planning for mobile robots. In J. P. Laumond (Ed.), *Robot motion planning and control* (pp. 1–53). Springer.

Lendaris, G., Cox, C., Saeks, R., & Murray, J. (2002, August 4–7). A radial basis function implementation of the adaptive dynamic programming algorithm. In *The 2002 45th midwest symposium on circuits and systems,* Tulsa, Oklahoma (pp. II–II). IEEE.

Li, N., Remeikas, C., Xu, Y., Jayasuriya, S., & Ehsani, R. (2015, May). Task assignment and trajectory planning algorithm for a class of cooperative agricultural robots. *Journal of Dynamic Systems, Measurement, and Control*, *137*(5), Article 051004. https://doi.org/10.1115/1.4028849

Li, Q., & Xu, Y. (2020, July 1–3). Unmanned aerial vehicle angular velocity control via reinforcement learning in dimension reduced search spaces. In *2020 American control conference,* Denver, CO (pp. 4926–4931). IEEE.

Liu, D., Huang, Y., Wang, D., & Wei, Q. (2013, May). Neural-network-observer-based optimal control for unknown nonlinear systems using adaptive dynamic programming. *International Journal of Control*, *86*(9), 1554–1566. https://doi.org/10.1080/00207179.2013.790562

Luenberger, D. G. (1979). *Introduction to dynamic systems*. John Wiley & Sons.

Luo, B., Yang, Y., & Liu, D. (2018). Adaptive Q-learning for data-based optimal output regulation with experience replay. *IEEE Transactions on Cybernetics*, *48*(12), 3337–3348. https://doi.org/10.1109/TCYB.2018.2821369

Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation Magazine*, *19*(3), 20–32. https://doi.org/10.1109/MRA.2012.2206474

Mayne, D. Q., & Michalsha, H. (1988, Dec 7–9). Receding horizon control of nonlinear systems. In *Proceedings of the 27th IEEE conference on decision and control,* Austin, TX (Vol. 1, pp. 464–465). IEEE.

Michalsha, H., & Mayne, D. Q. (1993, November). Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, *38*(11), 1623–1633. https://doi.org/10.1109/9.262032

Miller, W. T., Werbos, P. J., & Sutton, R. S. (1995). *Neural networks for control*. MIT Press.

Ogata, K. (2010). *Modern control engineering*. Prentice Hall.

Penrose, R. (1954). A generalized inverse for matrices. In *Mathematical proceedings of the cambridge philosophical society* (Vol. 51, pp. 406–413). Cambridge University Press.

Pi, C., Hu, K., Cheng, S., & Wu, I. (2020, February). Low-level autonomous control and tracking of quadrotor using reinforcement learning. *Control Engineering Practice*, *95*, Article 104222. https://doi.org/10.1016/j.conengprac.2019.104222

Polak, E. (1973). An historical survey of computational methods in optimal control. *SIAM Review*, *15*(2), 553–584. https://doi.org/10.1137/1015071

Pontryagin, L. S. (1986). *Mathematical theory of optimal processes*. CRC Press.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley & Sons.

Prokhorov, D. V., & Wunsch, D. C. (1997). Adaptive critic designs. *IEEE Transactions on Neural Networks*, *8*(5), 997–1007. https://doi.org/10.1109/TNN.72

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, June 22–24). Deterministic policy gradient algorithms. In *Proceedings of the 31st international conference on machine learning*, Beijing, China (pp. 387–395). JMIR.

Srinivasan, M. V., & Davey, M. (1995). Strategies for active camouflage of motion. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, *259*(1354), 19–25. https://doi.org/10.1098/rspb.1995.0004

Sundarapandian, V. (2003, January). An invariance principle for discrete-time nonlinear systems. *Applied Mathematics Letters*, *16*(1), 85–91. https://doi.org/10.1016/S0893-9659(02)00148-9

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. https://doi.org/10.1007/BF00115009.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Tesauro, G. (1992). Practical issues in temporal difference learning. In *Advances in neural information processing systems* (pp. 259–266). Morgan-Kaufmann.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* [Unpublished doctoral dissertation]. UKKing's College.

Watkins, C. J. C. H., & Dayan, P. (1992, May). Q-learning. *Machine Learning*, *8*(3-4), 279–292. https://doi.org/10.1007/BF00992698

Wei, Q., Liu, D., Lin, Q., & Song, R. (2017). Discrete-time optimal control via local policy iteration adaptive dynamic programming. *IEEE Transactions on Cybernetics*, *47*(10), 3367–3379. https://doi.org/10.1109/TCYB.2016.2586082

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In F. K. R. F. Drenick (Ed.), *System modeling and optimization* (pp. 762–770). Springer.

Werbos, P. J. (1989a, June). Back-propagation and neurocontrol: A review and prospectus. In *International 1989 joint conference on neural networks,* Washington, DC(pp. 209–216). IEEE.

Werbos, P. J. (1989b). Neural networks for control and system identification. In *Proceedings of the 28th IEEE conference on decision and control* (pp. 260–265). IEEE.

Werbos, P. J. (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, *3*(2), 179–189. https://doi.org/10.1016/0893-6080(90)90088-3

Werbos, P. J. (1992). Approximate dynamic programming for realtime control and neural modelling. In D. A. White & D. A. Sofge (Eds.), *Handbook of intelligent control: Neural, fuzzy and adaptive approaches* (pp. 493–525). Van Nostrand Reinhold.

Xin, M., & Pan, H. (2009, November). Integrated nonlinear optimal control of spacecraft in proximity operations. *International Journal of Control*, *83*(2), 347–363. https://doi.org/10.1080/00207170903171314

Xu, Y., & Basset, G. (2012). Sequential virtual motion camouflage method for nonlinear constrained optimal trajectory control. *Automatica*, *48*(7), 1273–1285. https://doi.org/10.1016/j.automatica.2012.05.017

Yaghmaie, F. A., & Braun, D. J. (2019). Reinforcement learning for a class of continuous-time input constrained optimal control problems. *Automatica*, *99*, 221–227. https://doi.org/10.1016/j.automatica.2018.10.038