TU Delft

Delft University of Technology

Incremental model-based global dual heuristic programming with explicit analytical calculations applied to flight control

Sun, Bo; van Kampen, Erik Jan

DOI
10.1016/j.engappai.2019.103425

Publication date
2020

Document Version
Accepted author manuscript

Published in
Engineering Applications of Artificial Intelligence

Citation (APA)
Sun, B., & van Kampen, E. J. (2020). Incremental model-based global dual heuristic programming with explicit analytical calculations applied to flight control. *Engineering Applications of Artificial Intelligence*, *89*, Article 103425. https://doi.org/10.1016/j.engappai.2019.103425

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Incremental Model-Based Global Dual Heuristic Programming with Explicit Analytical Calculations Applied to Flight Control*

Bo Sun*, Erik-Jan van Kampen

Department of Control and Operations, Delft University of Technology, Delft, 2629HS, The Netherlands

ARTICLE INFO

Keywords:
Global dual heuristic programming
Flight control
Incremental technique
Analytical calculation
Artificial neural network

ABSTRACT

A novel adaptive dynamic programming method, called incremental model-based global dual heuristic programming, is proposed to generate a self-learning adaptive flight controller, in the absence of sufficient prior knowledge of system dynamics. An incremental technique is employed for online local dynamics identification, instead of the artificial neural networks commonly used in global dual heuristic programming, to enable a fast and precise learning. On the basis of the identified model, two neural networks are adopted to facilitate the implementation of the self-learning controller, by approximating the cost-to-go and the control policy, respectively. The required derivatives of cost-to-go are computed by explicit analytical calculations based on differential operations. Both methods are applied to an online attitude tracking control problem of a nonlinear aerospace system and the results show that the proposed method outperforms conventional global dual heuristic programming in tracking precision, online learning speed, robustness to different initial states and adaptability for fault-tolerant control problems.

1. Introduction

Controller design for aerospace systems, especially airplanes, is challenging for many reasons. One of the most challenging parts is the difficulty of modeling the dynamics of the system. Especially for complex, nonlinear vehicles, global plant information may be impossible to obtain. The aircraft can be susceptible to uncertainties, sudden faults and structural damages, which changes the real plant compared to the previously obtained model (Lungu and Lungu (2018)). To deal with these problems, one promising solution is to create learning or adaptive controllers.

Reinforcement learning (RL), for instance, which links several bio-inspired artificial intelligence techniques, can make a system learn desired policies without accurate models of its dynamics or environment and can adapt to changing situations (Sutton and Barto (2018)). For these reasons, there have been a number of RL methods developed to enable model-free flight control in various types of aerospace systems (Hwangbo et al. (2017), Coates et al. (2017), Zhou et al. (2018b), Zhou et al. (2016a), Sun and van Kampen (2019)). However, different from ground robots, it is difficult to employ end-to-end training approaches on aerospace systems because of their special working environments. Consequently, a combination of RL and traditional control theory is a promising strategy to improve adaptive flight control. The combination of an actor-critic structure, dynamic programming, and neural networks, results in the adaptive/ approximate dynamic programming (ADP) algorithm, which is regarded as an effective technique to design adaptive optimal controllers and can achieve certain levels of adaptiveness and fault-tolerance (Wang (2019a), Valadbeigi et al. (2019), Wang et al. (2017), Wang (2019b)). According to optimal control theory, for a nominal system, given a cost function, the analytical optimal solution can be obtained by solving the Hamilton-Jacobi-Bellman (HJB) equation. However, for complex or nonlinear systems, analytical solutions to the HJB equation can be difficult or even impossible to get, let alone in real time. To conquer the difficulty of directly solving the HJB equation online for general nonlinear systems, the adaptive critic framework and artificial neural networks (ANNs) are often involved in order to approximate the HJB solution (Wang et al. (2017), Ferreira et al. (2017), Wang et al. (2019a)).

As a class of ADP methods, adaptive critic designs (ACDs), which separate policy evaluation (critic) and policy improvement (actor), have shown great success in optimal adaptive control of nonlinear aerospace systems (Ferrari and Stengel (2004), Van Kampen et al. (2006), Zhou et al. (2016a), Zhou et al. (2018b), Sun and van Kampen (2019)). ACDs can generally be categorized into several groups (Prokhorov and Wunsch (1997)): heuristic dynamic programming (HDP), dual heuristic programming (DHP) and global dual heuristic programming (GDHP) and their action-dependent (AD) versions. HDP is the most basic form and most often used structure, which employs the critic to approximate the cost-to-go. The critic in DHP approximates the derivatives of the cost-to-go with respect to the critic inputs, and in many practical applications it outperforms HDP in success rate and precision (Venayagamoorthy et al. (2002)). GDHP, which approximates both the cost-to-go and its derivatives so as to take advantage of two kinds of information, has several different forms (Prokhorov and Wunsch (1997)). Among them, the straightforward form, where the critic approximates the cost-to-go and its deriva-

*No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work.

*Corresponding author

b.sun-1@tudelft.nl (B. Sun); E.vanKampen@tudelft.nl (E. van Kampen)

ORCID(s): 0000-0002-5229-8545 (B. Sun); 0000-0002-5593-4471 (E. van Kampen)

tives simultaneously (Sun and van Kampen (2019), Yi et al. (2019), Liu et al. (2012)), is most commonly used because of its simple structure. In this architecture, two kinds of outputs share the same inputs and hidden layers, making them strongly coupled. Although these outputs have explicit mathematical relationships, without analytical calculations, weight update processes can suffer from this coupling due to inconsistent errors, and sometimes it even leads to instability. In this paper, explicit analytical calculations of the mixed second-order derivatives of the critic network outputs with respect to its input vector and weight matrices are introduced for adaptive flight control. Prokhorov and Wunsch (1997) and Fairbank et al. (2012) illustrate how to calculate these derivatives in an element-wise way. However, Magnus and Neudecker (2019) clarifies that vectors and matrices more often appear as a whole in practical applications rather than multi-variable functions, and this holistic view is easier to obtain the chain rule. Therefore, one contribution of this paper is deriving a direct method based on differential operation (Magnus and Neudecker (2019)) to compute these second-order derivatives in a holistic way so as to tackle the inconsistent errors between approximated cost-to-go function and its derivatives.

ACDs can be model-free if the critic is an AD network, which means the control signals are also introduced as network inputs (Abouheaf et al. (2018), Vamvoudakis and Ferraz (2018)). Nevertheless, to achieve model free application, an alternative is building a third module to approximate the plant dynamics and ANNs are often regarded as the first choice (Van Kampen et al. (2006), Liu et al. (2012), Bhasin et al. (2013), Lin et al. (2017), Liu et al. (2013)). Van Kampen et al. (2006) illustrates how this three-network structure outperforms AD ones if rewards only depend on system states. Although ANNs can approximate the nonlinear function with arbitrary precision, many samples are required before the weights converge for online identification of complex plants dynamics like aerospace systems, which can be dangerous especially at the start of training because the critic and actor networks are then trained based on the incorrect model. For these complex systems, offline training is normally involved to obtain a primary model and it often remains constant in applications (Van Kampen et al. (2006), Liu et al. (2012), Bhasin et al. (2013)), which, however, cannot achieve adaptive control when facing unforeseen uncertainties and sudden disturbances in realistic application.

The main contribution of this paper is an incremental model-based GDHP (IGDHP) method that enables online model-free flight control based on our latest work (Sun and van Kampen (2019)). Different from conventional GDHP, an incremental model is involved for adaptive control to deal with the absence of full system information. Assuming sufficiently high sampling rate for discretization, incremental techniques are able to accurately identify system dynamics online, preventing the controllers from initial failure, and have been successfully applied to design adaptive flight controllers, such as incremental nonlinear dynamic inversion (INDI)(Wang et al. (2019b)), incremental back-stepping (IBS)(Wang and van Kampen (2019)), incremental sliding mode control (ISMC)(Wang et al. (2018), Wang and van Kampen (2019)) and IADP (Zhou et al. (2016b), Zhou et al. (2018a)), IACDs (Zhou et al. (2017), Zhou et al. (2018b), Sun and van Kampen (2019)). However, these existed incremental methods have some limitations. For instance, INDI, IBS and ISMC cannot deal with optimal control problems, and IADP requires linear quadratic reward and offline training process. Compared to existed methods, IGDHP develops current IACDs to achieve online adaptive optimal control. In summary, the primary advantages lie in that the novel algorithm speeds up the online policy learning without knowing system dynamics or offline training a model network, and the analytical calculations make use of the information of cost-to-go function and its derivatives without introducing inconsistent errors.

The remainder of this paper is structured as follows. Section 2 presents the basic formulation of three-network GDHP with explicit analytical calculations. Section 3 introduces the incremental method for online identification and uses it to simplify the weight update process of the actor and critic networks. Then Section 4 provides the necessary information for verification, where an F-16 Fighting Falcon model is built and possible noises, faults and damages during flight are explained. Section 5 verifies the approaches by applying both GDHP and IGDHP on a longitudinal attitude tracking task in various conditions and analyzing their results. Finally section 6 summarizes the paper and puts up possibilities for future research.

2. GDHP Implementation

GDHP, which combines the advantages of HDP and DHP, can be implemented as a model free technique with three ANNs, namely model, critic and actor. The variables or pathways corresponding to these ANNs are denoted by the subscripts m, c and a, respectively. The architecture of GDHP with explicit analytical calculations is illustrated in Fig. 1. Based on current states, the actor network generates an action to control both real system and plant model. The model network estimates the states at the next time step, which are connected to the critic network to approximate cost-to-go, whose derivatives are computed analytically. All weights of the ANNs are updated in a back-propagation way according to the gradient-descent algorithm (Prokhorov and Wunsch (1997)).

2.1. Global Model

For a full-state feedback system, the inputs of the system model are the current state vector $\mathbf{x}_t \in \mathbb{R}^n$ and current control vector $\mathbf{u}_t \in \mathbb{R}^m$, while the output is the estimated next state vector $\hat{\mathbf{x}}_{t+1} \in \mathbb{R}^n$. However, because the future true value of state vector $\mathbf{x}_{t+1}$ is unavailable at time step t, the update is implemented with current and previous data, i.e. the network weights are updated by minimizing the difference between the current measured state vector $\mathbf{x}_t$ and the

estimated state vector $\hat{\mathbf{x}}_t$:

$$E_m(t) = \frac{1}{2}\mathrm{e}_m(t)^{\mathrm{T}} Q_m \mathrm{e}_m(t) \tag{1}$$

where

$$\mathrm{e}_m(t) = \hat{\mathbf{x}}_t - \mathbf{x}_t \tag{2}$$

and $Q_m \in \mathbb{R}^{n\times n}$ is a positive definite matrix. For simplicity, Q_m is usually defined as a diagonal matrix, i.e. $Q_m = \mathrm{diag}\{\zeta_1, \zeta_2, \cdots, \zeta_n\}$, where the elements respectively select and weigh the approximating errors.

The weights are updated by a gradient-descent algorithm:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) - \eta_m \cdot \frac{\partial E_m(t)}{\partial \mathbf{w}_m(t)} \tag{3}$$

where η_m is the learning rate, and

$$\frac{\partial E_m(t)}{\partial \mathbf{w}_m(t)} = \frac{\partial E_m(t)}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \mathbf{w}_m(t)} = \mathrm{e}_m^{\mathrm{T}}(t) \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \mathbf{w}_m(t)} \tag{4}$$

2.2. The Critic

GDHP combines HDP and DHP and requires the information of both the cost-to-go $J(\tilde{\mathbf{x}}_t)$ and its derivatives with respect to the network inputs $\tilde{\mathbf{x}}_t$, where $\tilde{\mathbf{x}}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t^{ref}$ stands for tracking error vector. The critic network only employs $\hat{J}(\tilde{\mathbf{x}}_t)$ to approximate the true cost-to-go $J(\mathbf{x}_t, \mathbf{x}_t^{ref})$, which is the cumulative sum of future rewards r_t from any initial state $\tilde{\mathbf{x}}_t$:

$$J(\mathbf{x}_t, \mathbf{x}_t^{ref}) = \sum_{l=t}^{\infty} \gamma^{l-t} r_l \tag{5}$$

where $\gamma \in (0, 1)$ is discount factor, used to control the extent to which the short-term cost or long-term cost is concerned. The derivative of the cost-to-go with respect to the input vector $\hat{\lambda}(\tilde{\mathbf{x}}_t)$ is shown in Appendix A .

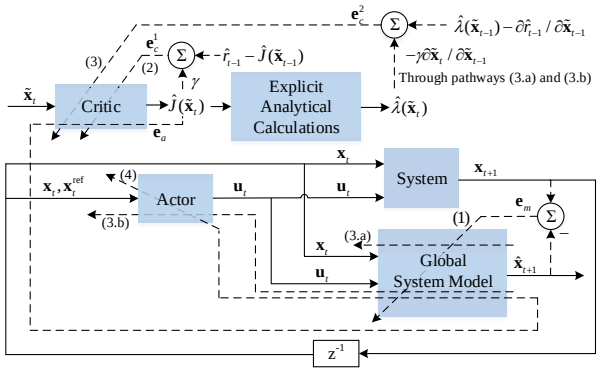

Figure 1: The architecture of GDHP with explicit analytical calculations, where solid lines represent the feedforward flow of signals, and dashed lines represent the adaptation pathways.

The goal of the experimental setup is to track the reference states contained in $\mathbf{x}_t^{ref}$, so a one-step cost function with a quadratic form is designed:

$$r_t = r(\mathbf{x}_t, \mathbf{x}_t^{ref}) = (\mathbf{x}_t - \mathbf{x}_t^{ref})^{\mathrm{T}} Q_c (\mathbf{x}_t - \mathbf{x}_t^{ref}) = \tilde{\mathbf{x}}_t^{\mathrm{T}} Q_c \tilde{\mathbf{x}}_t \tag{6}$$

where $Q_c \in \mathbb{R}^{n\times n}$ is a non-negative definite matrix.

Because future rewards are required, a temporal difference (TD) method is introduced to iteratively update the critic network (Sutton and Barto (2018)). The principle is to minimize the temporal difference error, the error between the current and successive estimates of the state value. Similar to the model network, the weights of the critic network are updated with current and previous data. The critic errors are as follows:

$$e_{c1}(t) = \hat{J}(\tilde{\mathbf{x}}_{t-1}) - \hat{r}_{t-1} - \gamma \hat{J}(\tilde{\mathbf{x}}_t) \tag{7}$$

and

$$\begin{aligned} \mathrm{e}_{c2}(t) &= \frac{\partial[\hat{J}(\tilde{\mathbf{x}}_{t-1}) - \hat{r}_{t-1} - \gamma \hat{J}(\tilde{\mathbf{x}}_t)]}{\partial \tilde{\mathbf{x}}_{t-1}} \\ &= \hat{\lambda}(\tilde{\mathbf{x}}_{t-1}) - \frac{\partial \hat{r}_{t-1}}{\partial \tilde{\mathbf{x}}_{t-1}} - \gamma \hat{\lambda}(\tilde{\mathbf{x}}_t) \frac{\partial \tilde{\mathbf{x}}_t}{\partial \tilde{\mathbf{x}}_{t-1}} \end{aligned} \tag{8}$$

where $e_{c1}(t)$ is the TD error of the estimated cost-to-go $\hat{J}(\tilde{\mathbf{x}}_{t-1})$ with current network weights, while $\mathrm{e}_{c2}(t)$ is the TD error of the computed derivatives $\hat{\lambda}(\tilde{\mathbf{x}}_{t-1})$ with current network weights. $\hat{r}$ denotes the estimated reward, because true states are unavailable to evaluate updated control policy. GDHP combines both of them in an overall error function $E_c(t)$:

$$E_c(t) = \beta \frac{1}{2} e_{c1}^2(t) + (1-\beta)\frac{1}{2}\mathrm{e}_{c2}^{\mathrm{T}}(t)\mathrm{e}_{c2}(t) \tag{9}$$

where β is a scalar indicating the importance within a range of $[0, 1]$. If $\beta = 1$, then it becomes pure HDP. If $\beta = 0$, then the tuning of weights merely depends on the TD error of computed derivatives $\hat{\lambda}(\tilde{\mathbf{x}}_t)$, and consequently it is equivalent to DHP, which is different from the straight form in Liu et al. (2012), Yi et al. (2019) and Sun and van Kampen (2019), where if $\beta = 0$, the back-propagation channel of the actor is cut.

The critic weights are updated using a gradient-descent algorithm with a learning rate η_c to minimize the overall error $E_c(t)$:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) - \eta_c \cdot \frac{\partial E_c(t)}{\partial \mathbf{w}_c(t)} \tag{10}$$

where

$$\begin{aligned} \frac{\partial E_c(t)}{\partial \mathbf{w}_c(t)} &= \frac{\partial E_c(t)}{\partial \hat{J}(\tilde{\mathbf{x}}_{t-1})} \cdot \frac{\partial \hat{J}(\tilde{\mathbf{x}}_{t-1})}{\partial \mathbf{w}_c(t)} + \frac{\partial E_c(t)}{\partial \hat{\lambda}(\tilde{\mathbf{x}}_{t-1})} \cdot \frac{\partial \hat{\lambda}(\tilde{\mathbf{x}}_{t-1})}{\partial \mathbf{w}_c(t)} \\ &= \beta e_{c1}(t) \cdot \frac{\partial \hat{J}(\tilde{\mathbf{x}}_{t-1})}{\partial \mathbf{w}_c(t)} + (1-\beta)\mathrm{e}_{c2}^{\mathrm{T}}(t) \cdot \frac{\partial \hat{\lambda}(\tilde{\mathbf{x}}_{t-1})}{\partial \mathbf{w}_c(t)} \end{aligned} \tag{11}$$

where $\partial \hat{\lambda}(\tilde{\mathbf{x}}_{t-1})/\partial \mathbf{w}_c(t)$ represents the second-order mixed gradient of the estimated cost-to-go $\hat{J}(\tilde{\mathbf{x}}_{t-1})$. An example of how to obtain it is illustrated in Appendix A .

2.3. The Actor

The actor network outputs control action $\mathbf{u}_t$, which is an input of the model network, and thus it will affect the critic outputs at the next time-step. The goal of the actor network is to produce an optimal control policy by minimizing the error between the current approximated cost-to-go $\hat{J}(\tilde{\mathbf{x}}_t)$ and the ideal one $J^*(t)$, which depends on the given reward function and is set to be zero in this paper:

$$\mathbf{u}_t^* = \arg\min_{\mathbf{u}_t} E_a(t) \tag{12}$$

where $E_a(t)$ is the overall actor error function and is defined as:

$$E_a(t) = \frac{1}{2}\mathrm{e}_a^2(t) \tag{13}$$

where

$$e_a(t) = \hat{J}(\tilde{\mathbf{x}}_t) - J^*(t) \tag{14}$$

Different from the straight form in Liu et al. (2012), Yi et al. (2019) and Sun and van Kampen (2019) where the actor network can be trained through the pathways either leading from $\hat{J}(\tilde{\mathbf{x}}_t)$ or carried out by $\hat{\lambda}(\tilde{\mathbf{x}}_t)$, there is only one back-propagation way for GDHP with explicit analytical calculations to update the actor weights and the information from both $\hat{J}(\tilde{\mathbf{x}}_t)$ and $\hat{\lambda}(\tilde{\mathbf{x}}_t)$ can be utilized. As illustrated in Fig. 1, the actor weights are updated along the 4th back-propagation direction with a learning rate η_a:

$$\mathbf{w}_a(t+1) = \mathbf{w}_a(t) - \eta_c \cdot \frac{\partial E_a(t)}{\partial \mathbf{w}_a(t)} \tag{15}$$

where

$$\begin{aligned}\frac{\partial E_a(t)}{\partial \mathbf{w}_a(t)} &= \frac{\partial E_a(t)}{\partial \hat{J}(\tilde{\mathbf{x}}_t)} \cdot \frac{\partial \hat{J}(\tilde{\mathbf{x}}_t)}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \mathbf{u}_{t-1}} \cdot \frac{\partial \mathbf{u}_{t-1}}{\partial \mathbf{w}_a(t)} \\ &= \hat{J}(\tilde{\mathbf{x}}_t) \cdot \frac{\partial \hat{J}(\tilde{\mathbf{x}}_t)}{\partial \hat{\mathbf{x}}_t} \cdot \frac{\partial \hat{\mathbf{x}}_t}{\partial \mathbf{u}_{t-1}} \cdot \frac{\partial \mathbf{u}_{t-1}}{\partial \mathbf{w}_a(t)}\end{aligned} \tag{16}$$

3. IGDHP Implementation

Aerospace systems have complex nonlinear dynamics for which ANNs can fail to achieve accurate online identification fast enough. In this section, an incremental technique is introduced to ensure a quick and accurate approximation using locally linearized models (Fig. 2). In addition to online learning and quick adaptation, it also reduces computational burden of the network weight update processes.

3.1. Incremental Model

Although most physical systems are continuous, modern processors work in a discrete way, leading to discrete measurements and computations. With the assumption of sufficiently high sampling frequency and relatively slow time-varying dynamics, one can represent a continuous nonlinear plant with a discrete incremental model and retain high enough precision. The derivation (Zhou et al. (2018b)) can be generally given as follows:

Consider a nonlinear continuous system described by:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)] \tag{17}$$

where $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathbb{R}^n$ provides the dynamics of the state vector over time. The general form can be used to describe dynamic and kinematic equations of complicated aerospace systems.

By taking the first order Taylor series expansion of (17) around time t_0 and omitting higher-order terms, the system is linearized approximately as follows:

$$\begin{aligned}\dot{\mathbf{x}}(t) \approx \dot{\mathbf{x}}(t_0) &+ \mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{x}(t) - \mathbf{x}(t_0)] \\ &+ \mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{u}(t) - \mathbf{u}(t_0)]\end{aligned} \tag{18}$$

where

$$\mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{x}(t)}\Big|_{\mathbf{x}(t_0), \mathbf{u}(t_0)} \tag{19}$$

$$\mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{u}(t)}\Big|_{\mathbf{x}(t_0), \mathbf{u}(t_0)} \tag{20}$$

$\mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)] \in \mathbb{R}^{n\times n}$ is the system matrix and $\mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)] \in \mathbb{R}^{n\times m}$ is the control effectiveness matrix. Assuming the states and state derivatives of the system are measurable, i.e. $\Delta\dot{\mathbf{x}}(t)$, $\Delta\mathbf{x}(t)$ and $\Delta\mathbf{u}(t)$ are measurable, an incremental model can be used to describe the above system:

$$\Delta\dot{\mathbf{x}}(t) \simeq \mathbf{F}[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{x}(t) + \mathbf{G}[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{u}(t) \tag{21}$$

With a constant, high sampling frequency, i.e. the sampling time Δt is sufficiently small, the plant model can be written approximately in a discrete form:

$$\frac{\mathbf{x}_{t+1} - \mathbf{x}_t}{\Delta t} \approx \mathbf{F}_{t-1} \cdot (\mathbf{x}_t - \mathbf{x}_{t-1}) + \mathbf{G}_{t-1} \cdot (\mathbf{u}_t - \mathbf{u}_{t-1}) \tag{22}$$

where $\mathbf{F}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathbb{R}^{n\times n}$ is the system transition matrix and $\mathbf{G}_{t-1} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathbb{R}^{n\times m}$ is the input distribution matrix at time step $t-1$ for the discretized

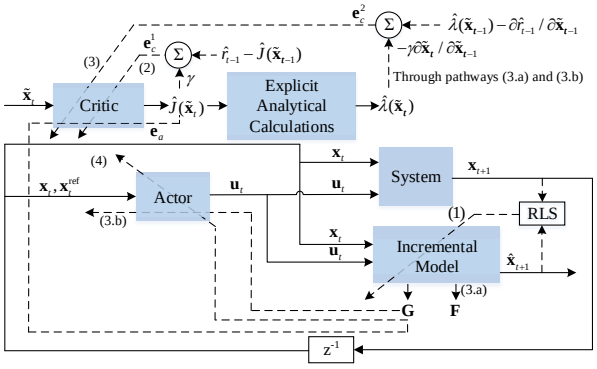

Figure 2: The architecture of IGDHP with explicit analytical calculations, where solid lines represent the feedforward flow of signals, and dashed lines represent the adaptation pathways.

systems. From (22), following incremental form of the new discrete nonlinear system can be obtained:

$$\Delta \mathbf{x}_{t+1} \approx \mathbf{F}_{t-1}\Delta t \cdot \Delta \mathbf{x}_t + \mathbf{G}_{t-1} \cdot \Delta t \cdot \Delta \mathbf{u}_t \tag{23}$$

In this way, the continuous nonlinear global plant is simplified into a linear incremental dynamic equation. The obtained local plant model can be identified online with recursive least squares (RLS) technique, to take advantage of its adaptability to cope with time variations in the regression parameters and fast convergence speed (Ferreira et al. (2017)), so as to avoid training a complex ANN. Although some information is omitted, such as state variation related nonlinear terms and higher-order terms in their Taylor series expansion, with the identified $\hat{\mathbf{F}}_{t-1}$ and $\hat{\mathbf{G}}_{t-1}$ matrix, the next system state can be predicted:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \hat{\mathbf{F}}_{t-1} \cdot \Delta t \cdot \Delta \mathbf{x}_t + \hat{\mathbf{G}}_{t-1} \cdot \Delta t \cdot \Delta \mathbf{u}_t \tag{24}$$

3.2. Online Identification Using RLS

A RLS approach is applied to identify the system transition matrix $\mathbf{F}_{t-1}$ and the input distribution matrix $\mathbf{G}_{t-1}$ online with the assumption of full-state feedback. The incremental form of the states in (23) can be rewritten in a row by row form as follows:

$$\Delta \mathbf{x}_{t+1} \approx \begin{bmatrix}\Delta \mathbf{x}_t^{\mathrm{T}} & \Delta \mathbf{u}_t^{\mathrm{T}}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{F}_{t-1}^{\mathrm{T}} \\ \mathbf{G}_{t-1}^{\mathrm{T}}\end{bmatrix} \cdot \Delta t \tag{25}$$

Since all increments of the states share the same covariance matrix, the parameters can be identified together as $\Theta_{t-1} = \begin{bmatrix}\mathbf{F}_{t-1}^{\mathrm{T}} \\ \mathbf{G}_{t-1}^{\mathrm{T}}\end{bmatrix} \in \mathbb{R}^{(n+m)\times n}$ (Zhou et al. (2018b)). Therefore, the state prediction equation (24) can be rewritten as follows:

$$\Delta \hat{\mathbf{x}}_{t+1} = \mathbf{X}_t^{\mathrm{T}} \cdot \hat{\Theta}_{t-1} \cdot \Delta t \tag{26}$$

where $\mathbf{X}_t = \begin{bmatrix}\Delta \mathbf{x}_t \\ \Delta \mathbf{u}_t\end{bmatrix} \in \mathbb{R}^{(n+m)\times 1}$ is the input information of the incremental model, and it is assumed to be measured directly.

The main procedure of the RLS approach is presented as follows:

$$\epsilon_t = \Delta \mathbf{x}_{t+1}^{\mathrm{T}} - \Delta \hat{\mathbf{x}}_{t+1}^{\mathrm{T}} \tag{27}$$

$$\hat{\Theta}_t = \hat{\Theta}_{t-1} + \frac{Cov_{t-1}\mathbf{X}_t}{\gamma_{RLS} + \mathbf{X}_t^{\mathrm{T}} Cov_{t-1}\mathbf{X}_t}\frac{\epsilon_t}{\Delta t} \tag{28}$$

$$Cov_t = \frac{1}{\gamma_{RLS}}\left(Cov_{t-1} - \frac{Cov_{t-1}\mathbf{X}_t\mathbf{X}_t^{\mathrm{T}}Cov_{t-1}}{\gamma_{RLS} + \mathbf{X}_t^{\mathrm{T}}Cov_{t-1}\mathbf{X}_t}\right) \tag{29}$$

where $\epsilon_t \in \mathbb{R}^{1\times n}$ stands for the prediction error, also called *innovation*, $Cov_t \in \mathbb{R}^{(n+m)\times(n+m)}$ is the estimation covariance matrix and it is symmetric and semi-positive definite, and γ_{RLS} is the forgetting factor for this RLS approach.

For most ACD designs, sufficient exploration of the state space guarantees good performance. Although RLS depends less on the global exploration, it is better to satisfy the persistent excitation (PE) condition (Zhou et al. (2018b)) for identifying incremental model. A 3211 disturbance signal is introduced to excite the system modes at the start of training.

3.3. Network Update Simplification

Considering (8), the last term $-\gamma\hat{\lambda}(\tilde{\mathbf{x}}_t)\frac{\partial \tilde{\mathbf{x}}_t}{\partial \tilde{\mathbf{x}}_{t-1}}$ needs to be dealt with carefully, because there are two pathways for $\tilde{\mathbf{x}}_{t-1}$ to affect $\tilde{\mathbf{x}}_t$. One is through the model network directly (pathway 3.a), and another one firstly goes through the actor network and then through the model network (pathway 3.b), as shown in both Figs. 1 and 2:

$$\frac{\partial \tilde{\mathbf{x}}_t}{\partial \tilde{\mathbf{x}}_{t-1}} = \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}} = \underbrace{\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}|_m}_{pathway\ (3.a)} + \underbrace{\frac{\partial \mathbf{x}_t}{\partial \mathbf{u}_{t-1}}|_m \cdot \frac{\partial \mathbf{u}_{t-1}}{\partial \mathbf{x}_{t-1}}|_a}_{pathway\ (3.b)} \tag{30}$$

In conventional GDHP, the two system model derivative terms in (30) are calculated back through the global system model, while IGDHP introduces the identified incremental model information directly to approximate them, whose computation burden is decreased compared to GDHP:

$$\frac{\partial \tilde{\mathbf{x}}_t}{\partial \tilde{\mathbf{x}}_{t-1}} \approx \hat{\mathbf{F}}_{t-1} \cdot \Delta t + \hat{\mathbf{G}}_{t-1} \cdot \Delta t \cdot \frac{\partial \mathbf{u}_{t-1}}{\partial \mathbf{x}_{t-1}}|_a \tag{31}$$

Similarly, the actor weight update process can also be simplified by the incremental information. Specifically, the term $\frac{\partial \hat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t}$ in (16) can be approximated by the identified input distribution matrix $\hat{G}_{t-1}$ directly:

$$\frac{\partial \hat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}_t} = \hat{G}_{t-1} \cdot \Delta t \tag{32}$$

Therefore, with the identified system transition matrix $\hat{\mathbf{F}}_{t-1}$ and input distribution matrix $\hat{\mathbf{G}}_{t-1}$, one can simplify the update processes of the critic network and actor network and thus accelerate the learning.

Wang et al. (2018), Wang et al. (2019b) and Wang and van Kampen (2019) demonstrate that under the assumption that the sampling rate is sufficiently high, in other words, Δt is small enough, the errors due to linearization and discretization can be ignored. In this paper, Δt is set to be 1 ms, which is realistic and accurate enough. The stability analysis of the GDHP method is investigated in Liu et al. (2012) and Yi et al. (2019). However, to the best of our knowledge, the theoretical assurance for the closed-loop convergence of online model-free control algorithms is still an open problem. The parameter convergence of control policy requires accurate and stable model information, which in turn depends on the parameter convergence of the control policy, making a circular argument.

4. Numerical Experiments Setup

The first part in this section introduces a nonlinear longitudinal model of F-16 Fighting Falcon for evaluation of

the proposed algorithms. The second part briefly introduces some potential uncertainties in practical flight. The third part discusses some related issues of the network structures for implementation of the aforementioned algorithms, including the activation function, the hierarchical actor network, etc.

4.1. Aerospace System Model

IGDHP can be applied to nonlinear aerospace systems, whose dynamic and kinematic state equations can be generally represented as:

$$\dot{\mathbf{x}}(t) = f\left[\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)\right] \tag{33}$$

where $\mathbf{d}(t)$ represents the disturbances and noises.

To verify the proposed method and compare the effect of these differences for a practical case, a nonlinear longitudinal model of F-16 Fighting Falcon (Abdullah et al. (2004), Nguyen et al. (1979)) is introduced. The model consists of the longitudinal force and moment equations, and it is a specific example of (33):

$$\begin{aligned}
\dot{V} &= g\sin(\alpha-\theta) + \frac{\cos\alpha}{m}T + \frac{\bar{q}S\cos\alpha}{m}C_{x,v} + \frac{\bar{q}S\sin\alpha}{m}C_{x,v} \\
\dot{\alpha} &= \frac{g}{V}\cos(\alpha-\theta) - \frac{\sin\alpha}{mV}T + (1 + \frac{\bar{q}S\bar{c}}{2mV^2}C_{x,\alpha})q + \frac{\bar{q}S}{mV}C_{z,\alpha} \\
\dot{q} &= \frac{\bar{q}S\bar{c}}{2I_{yy}V}C_{x,q}q + \frac{\bar{q}S\bar{c}}{I_{yy}}C_{z,q} \\
\dot{\theta} &= q \\
\dot{h} &= V\sin(\alpha-\theta)
\end{aligned} \tag{34}$$

where

$$\begin{aligned}
C_{x,v} &= C_x(\alpha,\delta_e) + \frac{\bar{c}}{2V}C_{xq}(\alpha)q \\
C_{z,v} &= C_z(\alpha,\delta_e) + \frac{\bar{c}}{2V}C_{zq}(\alpha)q \\
C_{x,\alpha} &= C_{zq}(\alpha)\cos\alpha - C_{xq}(\alpha)\sin\alpha \\
C_{z,\alpha} &= C_z(\alpha,\delta_e)\cos\alpha - C_x(\alpha,\delta_e)\sin\alpha \\
C_{x,q} &= \bar{c}C_{mq}(\alpha) + \bar{c}(X_{cgr} - X_{cg})C_{zq}(\alpha) \\
C_{z,q} &= C_m(\alpha,\delta_e) + (X_{cgr} - X_{cg})C_x(\alpha) \\
\bar{q} &= \frac{1}{2}\rho V^2
\end{aligned} \tag{35}$$

where V denotes the velocity, α and θ denote angle of attack and pitch angle, q denotes pitch rate, T denotes engine thrust, δ_e denotes elevator deflection, m denotes aircraft mass, $\bar{q}$ denotes dynamic pressure, ρ denotes air density, $\bar{c}$ denotes mean aerodynamic chord, S denotes wing planform area, Iyy denotes pitch moment of inertia, X_{cg} and X_{cgr} denote center of gravity (CG) location and reference CG location, g denotes gravitational constant, C_x, C_z, C_{xq}, C_{zq} are aerodynamic force coefficients and C_m, C_{mq} are aerodynamic moment coefficients. All parameters are determined by simulating this model around a steady wings-level flight condition at an altitude of approximately $15000ft$ with the speed of $600ft/s$ based on Nguyen et al. (1979).

Although the model has multiple states, two main states are selected as identified system states, which are angle of attack, that is to be controlled and pitch rate q, the basic inner state. In this paper, only one control input, elevator deflection δe, is considered, and engine thrust T is set be constant. Before elevator deflection is practically adjusted, the control signal that is generated by the actor $\mathbf{u}$, or δ_e^c in this attitude tracking problem, has to go through the actuator, which consists of a command saturation and a first-order filter with rate saturation, as shown in Fig. 3. δ_e^c is bounded in the range of $[-25°, 25°]$ and changing rate of elevator deflection is limited in the range of $[-60°/s, 60°/s]$ (Nguyen et al. (1979)).

4.2. Uncertainties

In flight control system design, system uncertainties, such as measurement uncertainties, unexpected changes of system dynamics or even sudden failures, need to be taken into account (Wang et al. (2018), Zhou et al. (2018b)). This section will introduce the uncertainties the system has to deal with.

In practice sensors have measurement uncertainties, and the magnitude of real-world phenomena used in this paper is illustrated in Table 1 (Van't Veld et al. (2018)). The bias acting on the feedback signals is based on the mean of the disturbances, while the noise acting on the signals is based on the standard deviation of the disturbances.

Sudden partial damages might be encountered during flight. The effects of the aircraft structural damages have been investigated in (Wang et al. (2018)). It has been found that actuators, as moving components, can be affected by unforeseen faults. The first actuator fault considered in this paper is the sudden decrease of elevator bandwidth, which is initially set to be 20.2 rad/s (Nguyen et al. (1979), Wang and van Kampen (2019)). The bandwidth, which numerically equals to the reciprocal of the time constant of the first-order filter, denotes the maximum frequency of sinusoidal command that the elevator can follow. A smaller bandwidth may lead to a high gain control policy, which can results in oscillation or divergence. Another actuator fault scenario considered is the reduction of control effectiveness, which can be caused directly by actuator damages or indirectly by structural damages that change aerodynamics.

For longitudinal dynamics, damage of the horizontal stabilizer needs to be taken into consideration, which leads to significant loss in both static and dynamic stability on the directional axis with an approximately linear relationship with the scale of tip loss, whose effectiveness is reflected by the changes of the aerodynamic moment coefficients C_m and C_{mq}. Besides, these structural damages are usually accompanied with mass loss, instantaneously shifting the CG to a new location.

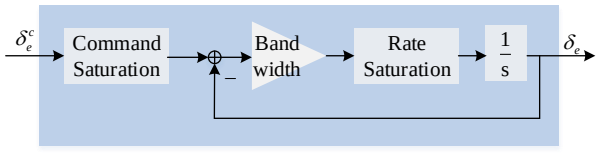

Figure 3: The dynamics of actuator, which is a first-order filter with command and rate saturation.

Table 1
Sensor uncertainties of F16 aircraft (adapted from Van't Veld et al. (2018)).

	Bias	Noise
q [°/s]	$1.7 \cdot 10^{-3}$	$3 \cdot 10^{-2}$
α [°]	$2.2 \cdot 10^{-1}$	$1.8 \cdot 10^{-3}$
δ_e [°]	$2.6 \cdot 10^{-1}$	$4 \cdot 10^{-2}$

4.3. Network Structure

ANNs, or more specifically multilayer perceptions (MLPs), are utilized to approximate the actor, critic and global model. For simplicity, the introduced ANNs are fully connected and consist of only three layers of nodes: an input layer, a hidden layer and an output layer. The activation function σ in the nodes of the hidden layer is a sigmoid function:

$$\varsigma(o) = \frac{1 - e^{-o}}{1 + e^{-o}} \tag{36}$$

It is anti-symmetric, zero-center and differentiable, with output bounded between -1 and 1. Its derivative is continuous, differentiable and positive at every point:

$$\frac{\partial \varsigma(o)}{\partial o} = \frac{1}{2}(1 - \varsigma(o))^2 \tag{37}$$

The actor is implemented as a hierarchical structure, or specifically a cascaded actor network (Van Kampen et al. (2006), Zhou et al. (2018b), Sun and van Kampen (2019)), as shown in Fig. 4. The first sub-network outputs a virtual reference signal of the pitch rate q, which is one input of the second sub-network, and the second sub-network produces the control command. Compared to the "flat" actor with only one end-to-end network, the virtual reference signal q^{ref} provides a more direct instruction. This hierarchical structure takes advantage of the physical properties of the system, by putting some prior knowledge into the design of the controller, which in theory will reduce the complexity of the problem. To improve stability, the output layers of the actor sub-networks adopt a sigmoid function as activation function, to add restrictions to the pitch rate reference and the control action, which is different from the global model and the critic, where linear functions are employed. The pitch rate and the elevator deflection commands are bounded in the range of $[-20°/s, 20°/s]$ and $[-25°, 25°]$ respectively.

The critic and actor networks in both GDHP and IGDHP have the same settings. The DHP technique generally surpasses HDP in tracking precision, convergence speed and success rate because the costate function is employed (Wang et al. (2019a) and Zhou et al. (2018b)). Therefore, to take advantage of the information of derivatives, β is set to be 0.01. More neurons will improve approximation precision, but can also increase computational burden or even lead to overfitting, which will decrease the robustness of the controller. As a trade off, the number of hidden layer neurons in the actor is 15, while in both the critic and the global system it is 25. Initial weights of the neural networks can have a great influence on the learning. In this paper, all weights are randomly initialized within a small range of $[-0.01, 0.01]$ to reduce the impact of initialization, and bounded within the range of $[-20, 20]$ to prevent sudden failure in the learning process. To guarantee effective learning, learning rates have to be chosen carefully. A descending method is applied, which means that the initial learning rates are set to be large numbers which gradually decrease as the weights are updated.

5. Results and Discussion

Both the GDHP and the IGDHP algorithms are applied to a simulation of controlling the F16 aircraft longitudinal model. First, the flight controller learns to track a changing reference at different initial conditions. Then, these two methods are compared in the failure cases where actuator faults and structural damages take place. All numerical experiments are implemented in the presence of sensor uncertainties.

5.1. Different Initial Conditions

Figures 5-7 compare the performance of the IGDHP and GDHP approaches, when applied to control the F16 aircraft to track a given reference signal online at different initial states. To be more specific, the controllers are required to control the angle of attack α to track the reference signal α^{ref}, which is a sinusoidal wave with the amplitude of 10 degrees and the period of 4π seconds. The sub-figures on the top present how the angle of attack α tracks the reference signal α^{ref} using these two approaches respectively, while the sub-figures on the bottom provide the tracking errors during these tasks.

Take every two degrees as a scale to examine the influence of the initial states. When the initial state α_0 is $\pm 2°$, both methods perform similarly to the simulation results with zero initial states, as shown in Fig. 5 and Fig. 6. When the initial state is 4°, although tracking precision decreases for GDHP method, both methods can successfully complete the tracking task, as shown in Fig. 7. Nevertheless, compared to GDHP, IGDHP spends less time to find a feasible actor, leading to a smaller settling time. These results imply the incremental techniques can accelerate the learning process of the actor and critic networks. Furthermore, when the initial state α_0 is beyond the range of $[-2°, 4°]$,

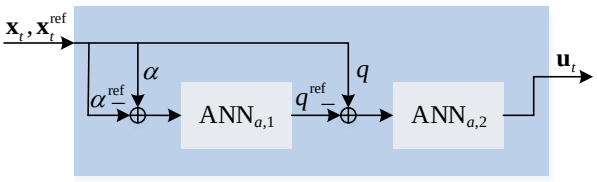

Figure 4: The architecture of the cascaded actor network (Sun and van Kampen (2019)), where the physical properties of aircraft dynamics are utilized.

the GDHP method cannot track the reference signal without oscillation. On the other side, the IGDHP method can deal with a wider range of initial states within $[-10°, 10°]$ without the loss of precision. As presented in Fig. 8, the angle of attack α can follow the given reference signal α^{ref} in less than 1 second in all initial conditions using the IGDHP approach, which shows that IGDHP is more robust than GDHP to various initial states.

However, Fig. 5-8 only present the nominal results when the task is successfully performed. Random factors, including initial weights of the neural networks and sensor noises, can affect the performance and sometimes can even lead to failure. To compare how robust the proposed algorithms are to these random factors, a concept of success ratio is introduced to indicate their performance, which has been widely used in Zhou et al. (2018b), Van Kampen et al. (2006), Sun and van Kampen (2019). The success ratio used in this paper is defined as that the angle of attack α can track the given reference signal α^{ref} after one period of α^{ref}, 4π seconds, and the tracking errors will not exceed $\pm 2°$ hereafter. Without adjustment of parameters, 1000 times of Monte Carlo simulation are implemented to evaluate the performance of algorithms.

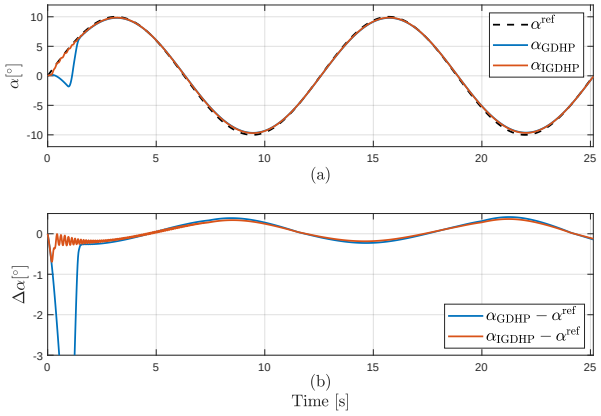

Figure 5: Online attitude tracking control with the zero initial state, $\alpha_0 = 0°$ using GDHP and IGDHP approaches.

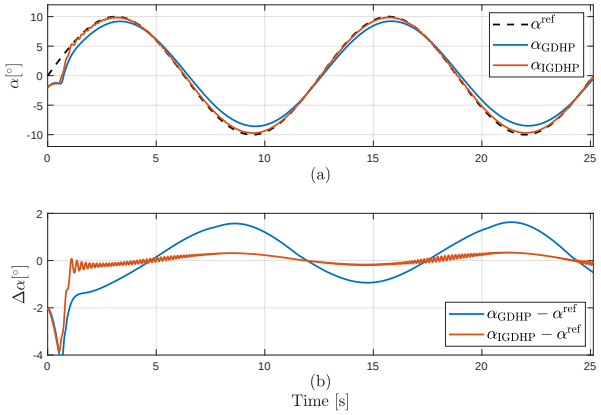

Figure 6: Online attitude tracking control with a negative initial state, $\alpha_0 = -2°$ using GDHP and IGDHP approaches.

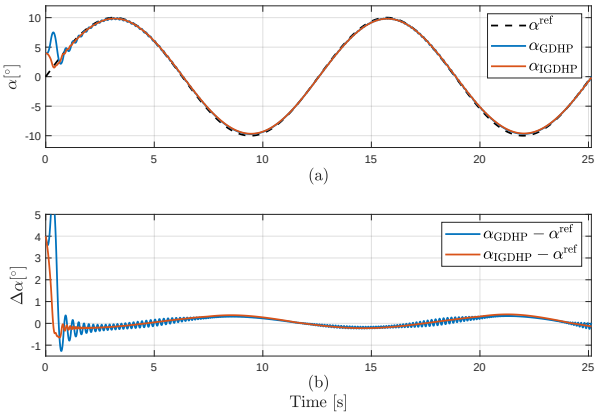

Figure 7: Online attitude tracking control with positive initial state, $\alpha_0 = 4°$ using GDHP and IGDHP approaches.

The results are illustrated in Table 2. The reason why the highest success ratio is not 100% lies in the difficulty to achieve optimal PE condition due to the circular argument between PE condition, accurate system information and stable control policy. Improving several factors can increase success ratio and improve robustness, such as the performance of sensors, exploration noise, parameter initialization and learning rates. However, the improvement of these factors still remain to be open problems and therefore this paper only concentrates on the comparison of robustness between different methods. As presented in Table 2, the success ratios of IGDHP are higher than those of GDHP at any initial state. When applied to non-zero initial states, success ratios decrease dramatically for GDHP, which means that the global model is not robust enough for various initial conditions. However, the success ratios of both IGDHP and GDHP degrade heavily due to measurement uncertainties, and the impacts on IGDHP are even more severe. That is because, to achieve the quick and high-precision identified model, the incremental method adapts quickly to locally acquired data. Nevertheless, IGDHP still shows better performance.

Table 2
Success ratio comparison for different initial states and measurement uncertainties situations with 1000 times of Monte Carlo simulation.

	α_0/[°]	−2	0	2	4
Without uncertainties	GDHP	1.1%	50.3%	1.9%	1.4%
	IGDHP	32.4%	91.6%	41.3%	36.6%
With uncertainties	GDHP	0.7%	43.9%	1.6%	1.0%
	IGDHP	19.5%	54.3%	25.1%	19.8%

5.2. Fault-Tolerant Examination

The capability to adapt is one of the most important advantages for ACDs compared to other traditional control

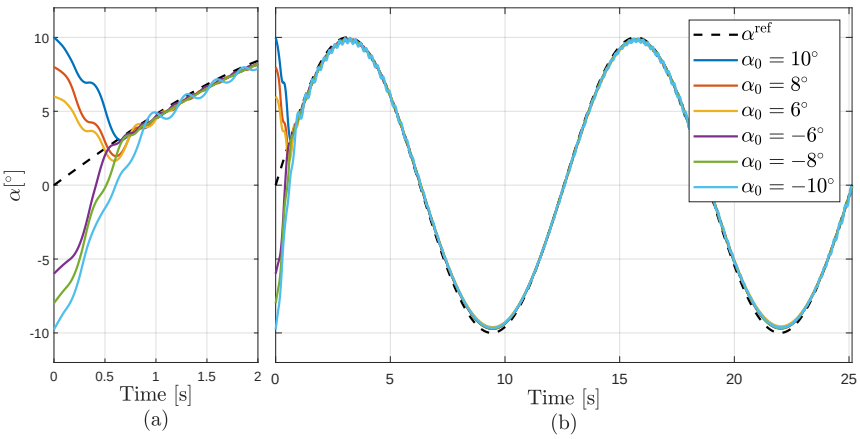

Figure 8: Online attitude tracking control with different initial states using the IGDHP approach.

techniques. It allows the controller to learn sound polices automatically by tuning the weights of the networks and this merit makes ACDs suitable for fault-tolerant control (FTC). Fault diagnosis is a significant part in FTC area, but will not be discussed in this paper. It is assumed that when sudden fault occurs, the controller can recognize it immediately. One challenge of FTC is that the controller is unable to change its policy quickly enough when faced with sudden changes in plant dynamics, while in this situation the originally learned control policy may even increase the instability of the closed loop plant. Therefore, in this paper, the way the controller adapt to new situation is to reset the actor weights to small random numbers within the range of $[-0.01, 0.01]$ and to increase the corresponding learning rate as long as the fault is detected.

Figure 9 compares the online adaptability of the GDHP method and the IGDHP method in the presence of the sudden decrease of elevator bandwidth to 18 rad/s. This change is introduced after the convergence of the policy for original system at 4π, 4.5π and 5π seconds, respectively, which corresponds to different values of the reference signal. As shown in Fig. 9, GDHP shows poor adaptation performance and it results in divergence in sub-figure (c). Although the GDHP method can adapt in sub-figures (a) and (b), its tracking performance degrades and the adapted controller leads to unexpected oscillations due to a higher gain control policy. On the contrary, IGDHP is able to adapt to elevator faults, and continues to track the commands precisely. The adaptation of the actor weights during this online FTC task is presented in Fig. 10. There are in total 60 weights belonging to two cascaded networks. Figure 10 demonstrates how the controller achieves a convergent policy by adapting actor weights and sub-figures (b) and (c) take a closer look at the main learning processes at the beginning and after the elevator fault happens, respectively.

The second fault scenario considered is that the elevator suddenly loses 30% of its effectiveness during flight at 4π, 4.5π and 5π seconds, respectively. As can be seen from Fig. 11, only when this sudden fault happens at 4π

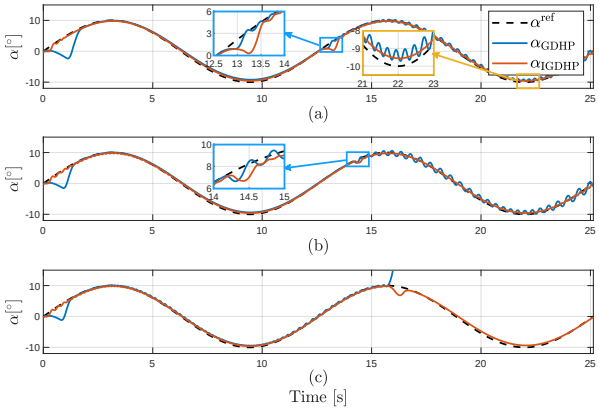

Figure 9: Online fault-tolerant tracking control using GDHP and IGDHP approaches in the presence of a sudden decrease of elevator bandwidth at 3 different times.

seconds, GDHP can recover from this situation, but it encounters slightly growing oscillations thereafter, making the tracking errors have larger root mean square value. If the sudden damage occurs at the points where α^{ref} has non-zero value, GDHP will suffer from divergence. On the other hand, IGDHP is able to rapidly adapt to the elevator fault with smaller tracking errors.

The last fault scenario considered is that at the three different times mentioned above, the left stabilator is damaged, while the right stabilator is still working normally. Accompanying with the left stabilator damage, the CG shifts forwards and to the right, producing both rolling and pitching moment increments. However, only the effects of pitching moment increments are considered for this longitudinal model and rolling effects are omitted. The reduced longitudinal damping and stability margin are also influencing the closed-loop system responses. The results are illustrated in Fig. 12, where at all three times, GDHP fails to adapt to the new dynamics while IGDHP shows satisfying performance.

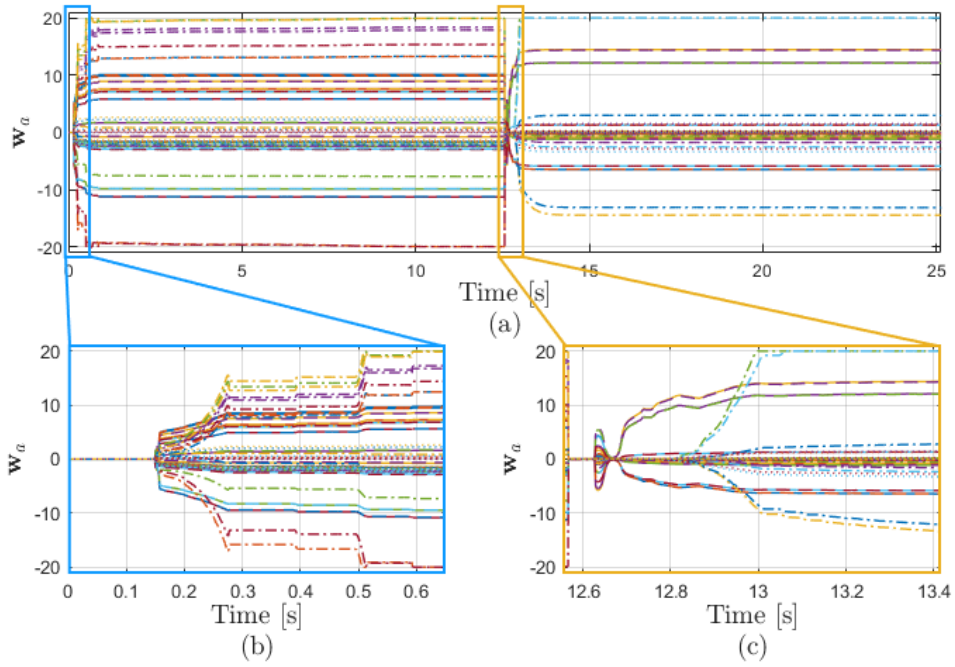

Figure 10: Convergence of the actor weights using the IGDHP approach when faced with a sudden decrease of elevator bandwidth at 4π seconds.

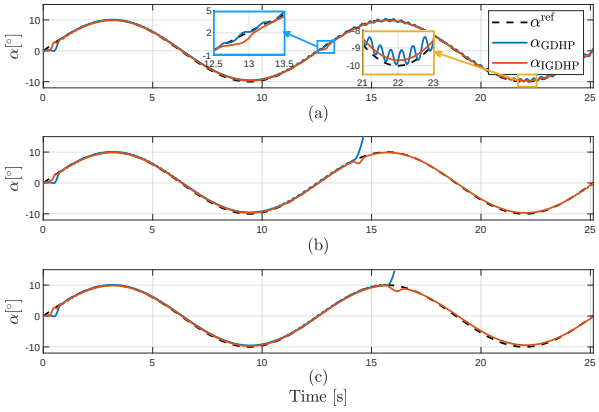

Figure 11: Online fault-tolerant tracking control using GDHP and IGDHP approaches in the presence of sudden reduction of control effectiveness at 3 different times.

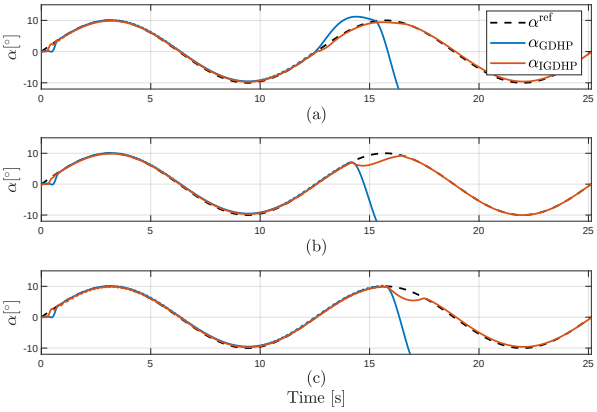

Figure 12: Online fault-tolerant tracking control using GDHP and IGDHP approaches when horizontal stabilizers are partially damaged at 3 different times.

6. Conclusion

This paper develops a novel approach, called incremental model based global dual heuristic programming (IGDHP), to generate an adaptive model-free flight controller. Different from traditional global dual heuristic programming (GDHP), which often employs an artificial neural network to approximate the global system dynamics, IGDHP adopts incremental approaches instead to identify the local plant model online and to speed up policy convergence. Besides, this paper derives a direct method from a holistic viewpoint based on differential operation, to explicitly analytically compute derivatives of cost-to-go function with respect to the critic inputs, rather than utilizes conventional neural network approximation, so as to eliminate the inconsistent errors due to coupling.

Both methods are applied to an online longitudinal attitude tracking task of a nonlinear F-16 Fighting Falcon system, whose dynamics are unknown to the controller. The numerical experiment results uniformly illustrate that, in comparison to conventional GDHP, IGDHP improves tracking precision, accelerates the online learning process, has advantages in robustness to different initial states and mea-

surement uncertainties, and has increased capability to adapt when faced with unforeseen sudden faults.

This study generalizes the basic form of the IGDHP but still has limitations for realistic applications. Further research should, therefore, concentrate on the investigation of various types of function approximators, the improvement of stability and success ratio, and expansion to other application scenarios.

Acknowledgement

The authors would like to thank the Chinese Scholarship Council for financial support for B. Sun with project number of the project reference number of 201806290007.

A. Vector and Matrix Derivation

The following derivation process to calculate derivatives are based on differential operation (Magnus and Neudecker (2019)).Consider a critic network with a single hidden layer architecture the numbers of network inputs, neurons and outputs are n, p and 1, respectively. For convenience, define I_r as an identity matrix with a dimension of r, where r can be n, p and 1, respectively.

A.1. $\hat{\lambda}(\tilde{\mathbf{x}}_t)$

The feed-forward process of the three-layer critic network is given as:

$$\hat{J} = \mathrm{w}_{c2}^{\mathrm{T}}\sigma(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}) \tag{38}$$

where w_{c1} denotes the weight matrix connecting input layer and hidden layer, w_{c2} denotes the weight matrix connecting hidden layer and output layer, and $\sigma(\cdot)$ is an element-wise operation applied to the transfer function in the hidden layer. Compute the differentials of both sides:

$$\begin{aligned}&\mathrm{d}\hat{J} = \mathrm{dw}_{c2}^{\mathrm{T}}\cdot\sigma(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}) + \mathrm{w}_{c2}^{\mathrm{T}}\cdot\mathrm{d}\sigma(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})\\&= \mathrm{w}_{c2}^{\mathrm{T}}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})\odot(\mathrm{w}_{c1}^{\mathrm{T}}\cdot\mathrm{d}\tilde{\mathbf{x}}))\end{aligned} \tag{39}$$

where $\odot$ denotes Hadamard product, and $\sigma'(\cdot)$ denotes the first order derivative of function σ. Perform trace operation on both sides:

$$\begin{aligned}&\mathrm{tr}(\mathrm{d}\hat{J}) = \mathrm{d}\hat{J} = \mathrm{tr}(\mathrm{w}_{c2}^{\mathrm{T}}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})\odot(\mathrm{w}_{c1}^{\mathrm{T}}\cdot\mathrm{d}\tilde{\mathbf{x}})))\\&= \mathrm{tr}((\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))^{\mathrm{T}}(\mathrm{w}_{c1}^{\mathrm{T}}\mathrm{d}\tilde{\mathbf{x}}))\\&= \mathrm{tr}((\mathrm{w}_{c1}(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})))^{\mathrm{T}}\mathrm{d}\tilde{\mathbf{x}})\end{aligned} \tag{40}$$

The relationship between derivative of scalar with respect to vector and their differentials, the first order derivative of critic network output with respect to its inputs, is:

$$\hat{\lambda}(\tilde{\mathbf{x}}_t) = \frac{\partial\hat{J}}{\partial\tilde{\mathbf{x}}} = \mathrm{w}_{c1}(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})) \tag{41}$$

A.2. $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial\mathrm{w}_c$

Since the critic network has two weights matrices, the second-order derivatives have to be calculated separately. Perform the differential operation on both sides of (41):

$$\mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t) = \mathrm{dw}_{c1}(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})) + \mathrm{w}_{c1}\cdot\mathrm{d}(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})) \tag{42}$$

Firstly consider $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial\mathrm{w}_{c2}$ and regard w_{c1} as a constant matrix, then (42) can be modified as:

$$\begin{aligned}&\mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t) = \mathrm{w}_{c1}\cdot\mathrm{d}(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\\&= \mathrm{w}_{c1}(\mathrm{dw}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\mathrm{I}_1\end{aligned} \tag{43}$$

Reshape both sides to column vectors:

$$\begin{aligned}&\mathrm{vec}(\mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t)) = \mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t)\\&= \mathrm{vec}(\mathrm{w}_{c1}\cdot\mathrm{d}(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\mathrm{I}_1)\\&= (\mathrm{I}_1\otimes\mathrm{w}_{c1})\cdot\mathrm{vec}(\mathrm{dw}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\\&= (\mathrm{I}_1\otimes\mathrm{w}_{c1})\cdot\mathrm{diag}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\cdot\mathrm{vec}(\mathrm{dw}_{c2})\end{aligned} \tag{44}$$

where $\mathrm{vec}(\cdot)$ is vector reshaping function, $\mathrm{diag}(\cdot)$ reshapes the vector to a diagonal matrix, and $\otimes$ is Kronecker product. $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial\mathrm{w}_{c2}$ is the derivative of the vector $\hat{\lambda}(\tilde{\mathbf{x}}_t)$ with respect to the matrix w_{c2}. Based on the relationship between $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial\mathrm{w}_{c2}$ and the differentials of $\hat{\lambda}(\tilde{\mathbf{x}}_t)$ and w_{c2} (Magnus and Neudecker (2019)), $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial\mathrm{w}_{c2}$ can be obtained:

$$\begin{aligned}&\tfrac{\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)}{\partial\mathrm{w}_{c2}} = ((\mathrm{I}_1\otimes\mathrm{w}_{c1})\cdot\mathrm{diag}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})))^{\mathrm{T}}\\&= \mathrm{diag}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))(\mathrm{I}_1\otimes\mathrm{w}_{c1}^{\mathrm{T}})\end{aligned} \tag{45}$$

Then consider $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial\mathrm{w}_{c1}$ and regard w_{c2} as a constant matrix. Perform the differential operation on both sides of (41):

$$\begin{aligned}&\mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t) = \mathrm{dw}_{c1}(\mathrm{w}_2\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))+\\&\quad\mathrm{w}_{c1}(\mathrm{dw}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}) + \mathrm{w}_{c2}\odot\mathrm{d}\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\\&= \mathrm{I}_n\cdot\mathrm{dw}_{c1}\cdot(\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))+\\&\quad\mathrm{w}_{c1}(\mathrm{w}_{c2}\odot\mathrm{d}\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\mathrm{I}_1\end{aligned} \tag{46}$$

Recall (37) and reshape both sides to column vectors:

$$\begin{aligned}&\mathrm{vec}(\mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t)) = \mathrm{d}\hat{\lambda}(\tilde{\mathbf{x}}_t)\\&= ((\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))^{\mathrm{T}}\otimes\mathrm{I}_n)\cdot\mathrm{vec}(\mathrm{dw}_{c1})+\\&\quad(\mathrm{I}_1\otimes\mathrm{w}_{c1})\cdot\mathrm{vec}(\mathrm{w}_{c2}\odot\mathrm{d}(\tfrac{1}{2}(1-\sigma^{\circ 2}(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))))\\&= ((\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))^{\mathrm{T}}\otimes\mathrm{I}_n)\cdot\mathrm{vec}(\mathrm{dw}_{c1})+\\&\quad(\mathrm{I}_1\otimes\mathrm{w}_{c1})\cdot\mathrm{vec}(\mathrm{w}_{c2}\odot(-\sigma(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})\odot\\&\quad(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})\odot((\mathrm{dw}_{c1})^{\mathrm{T}}\tilde{\mathbf{x}}))))\\&= ((\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))^{\mathrm{T}}\otimes\mathrm{I}_n)\cdot\mathrm{vec}(\mathrm{dw}_{c1})-\\&\quad(\mathrm{I}_1\otimes\mathrm{w}_{c1})\mathrm{diag}(\mathrm{vec}(\mathrm{w}_{c2}))\cdot\\&\quad\mathrm{diag}(\sigma(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\mathrm{diag}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}})))\mathrm{vec}(\mathrm{I}_p(\mathrm{dw}_{c1})^{\mathrm{T}}\tilde{\mathbf{x}})\\&= ((\mathrm{w}_{c2}\odot\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))^{\mathrm{T}}\otimes\mathrm{I}_n - (\mathrm{I}_1\otimes\mathrm{w}_{c1})\cdot\\&\quad\mathrm{diag}(\mathrm{vec}(\mathrm{w}_{c2}))\mathrm{diag}(\sigma(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\mathrm{diag}(\sigma'(\mathrm{w}_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\cdot\\&\quad(\tilde{\mathbf{x}}^{\mathrm{T}}\otimes\mathrm{I}_p)\mathrm{K})\cdot\mathrm{vec}(\mathrm{dw}_{c1})\end{aligned} \tag{47}$$

where $\cdot^{\circ 2}$ is an element-wise square operation, and K is a commutation matrix of w_{c1}. Similar to $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial w_{c2}$, based on the relationship between the derivative of the vector $\hat{\lambda}(\tilde{\mathbf{x}}_t)$ with respect to the matrix w_{c1}, and the differentials of $\hat{\lambda}(\tilde{\mathbf{x}}_t)$ and w_{c1}, $\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)/\partial w_{c1}$ is obtained as follows:

$$\begin{aligned}\frac{\partial\hat{\lambda}(\tilde{\mathbf{x}}_t)}{\partial w_{c1}} &= ((w_{c2}\odot\sigma'(w_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))^{\mathrm{T}}\otimes I_n-(I_1\otimes w_{c1})\cdot\\ &\quad \mathrm{diag}(\mathrm{vec}(w_{c2}))\mathrm{diag}(\sigma(w_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\cdot\\ &\quad \mathrm{diag}(\sigma'(w_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))(\tilde{\mathbf{x}}^{\mathrm{T}}\otimes I_p)K)^{\mathrm{T}}\\ &= (w_{c2}\odot\sigma'(w_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\otimes I_n-K^{\mathrm{T}}(\tilde{\mathbf{x}}\otimes I_p)\cdot\\ &\quad \mathrm{diag}(\sigma'(w_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\mathrm{diag}(\sigma(w_{c1}^{\mathrm{T}}\tilde{\mathbf{x}}))\cdot\\ &\quad \mathrm{diag}(\mathrm{vec}(w_{c2}))(I_1\otimes w_{c1}^{\mathrm{T}})\end{aligned}\tag{48}$$

Note that in order to obtain the derivative of a vector with respect to a matrix, tensor operation is involved, which, however, reduces the dimensionality of the original matrix. Therefore, after calculating the gradients of error with respect to weights (referring to (16) by chain rule), dimensionality analysis is required to reshape the results.

References

Abdullah, A.A., Ioannou, P.A., Samaras, G., 2004. Control design for f-16 longitudinal motion. IFAC Proceedings Volumes 37, 529–534.

Abouheaf, M., Gueaieb, W., Lewis, F., 2018. Model-free gradient-based adaptive learning controller for an unmanned flexible wing aircraft. Robotics 7, 66.

Bhasin, S., Kamalapurkar, R., Johnson, M., Vamvoudakis, K.G., Lewis, F.L., Dixon, W.E., 2013. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. Automatica 49, 82–92.

Coates, A., Abbeel, P., Ng, A.Y., 2017. Autonomous helicopter flight using reinforcement learning. Encyclopedia of Machine Learning and Data Mining , 75–85.

Fairbank, M., Alonso, E., Prokhorov, D., 2012. Simple and fast calculation of the second-order gradients for globalized dual heuristic dynamic programming in neural networks. IEEE transactions on neural networks and learning systems 23, 1671–1676.

Ferrari, S., Stengel, R.F., 2004. Online adaptive critic flight control. Journal of Guidance, Control, and Dynamics 27, 777–786.

Ferreira, E.F., Rêgo, P.H., Neto, J.V., 2017. Numerical stability improvements of state-value function approximations based on rls learning for online hdp-dlqr control system design. Engineering Applications of Artificial Intelligence 63, 1–19.

Hwangbo, J., Sa, I., Siegwart, R., Hutter, M., 2017. Control of a quadrotor with reinforcement learning. IEEE Robotics and Automation Letters 2, 2096–2103.

Lin, H., Wei, Q., Liu, D., 2017. Online identifier–actor–critic algorithm for optimal control of nonlinear systems. Optimal Control Applications and Methods 38, 317–335.

Liu, D., Huang, Y., Wang, D., Wei, Q., 2013. Neural-network-observer-based optimal control for unknown nonlinear systems using adaptive dynamic programming. International Journal of Control 86, 1554–1566.

Liu, D., Wang, D., Zhao, D., Wei, Q., Jin, N., 2012. Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming. IEEE Transactions on Automation Science and Engineering 9, 628–634.

Lungu, M., Lungu, R., 2018. Neural network based adaptive control of airplane's lateral-directional motion during final approach phase of landing. Engineering Applications of Artificial Intelligence 74, 322–335.

Magnus, J.R., Neudecker, H., 2019. Matrix differential calculus with applications in statistics and econometrics. John Wiley & Sons.

Nguyen, L., Ogburn, M., Gilbert, W., Kibler, K., Brown, P., Deal, P., 1979. Nasa technical paper 1538-simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability. Tech. rep., NASA .

Prokhorov, D.V., Wunsch, D.C., 1997. Adaptive critic designs. IEEE transactions on Neural Networks 8, 997–1007.

Sun, B., van Kampen, E., 2019. Incremental model-based global dual heuristic programming for flight control. IFAC-PapersOnLine In press.

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Valadbeigi, A.P., Sedigh, A.K., Lewis, F.L., 2019. H∞ static output-feedback control design for discrete-time systems using reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems , 1–11.

Vamvoudakis, K.G., Ferraz, H., 2018. Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. Automatica 87, 412–420.

Van Kampen, E., Chu, Q.P., Mulder, J., 2006. Online adaptive critic flight control using approximated plant dynamics, in: 2006 International Conference on Machine Learning and Cybernetics, IEEE. pp. 256–261.

Van't Veld, R., van Kampen, E., Chu, Q.P., 2018. Stability and robustness analysis and improvements for incremental nonlinear dynamic inversion control, in: 2018 AIAA Guidance, Navigation, and Control Conference, p. 1127.

Venayagamoorthy, G.K., Harley, R.G., Wunsch, D.C., 2002. Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator. IEEE Transactions on Neural Networks 13, 764–773.

Wang, D., 2019a. Intelligent critic control with robustness guarantee of disturbed nonlinear plants. IEEE transactions on cybernetics Early Access.

Wang, D., 2019b. Robust policy learning control of nonlinear plants with case studies for a power system application. IEEE Transactions on Industrial Informatics Early Access.

Wang, D., Ha, M., Qiao, J., 2019a. Self-learning optimal regulation for discrete-time nonlinear systems under event-driven formulation. IEEE Transactions on Automatic Control Early Access.

Wang, D., He, H., Liu, D., 2017. Adaptive critic nonlinear robust control: A survey. IEEE transactions on cybernetics 47, 3429–3451.

Wang, X., van Kampen, E., 2019. Incremental backstepping sliding mode fault-tolerant flight control, in: AIAA Scitech 2019 Forum, p. 0110.

Wang, X., van Kampen, E., Chu, Q.P., Lu, P., 2018. Incremental sliding-mode fault-tolerant flight control. Journal of Guidance, Control, and Dynamics 42, 244–259.

Wang, X., van Kampen, E., Chu, Q.P., Lu, P., 2019b. Stability analysis for incremental nonlinear dynamic inversion control. Journal of Guidance, Control, and Dynamics 42, 1116–1129.

Yi, J., Chen, S., Zhong, X., Zhou, W., He, H., 2019. Event-triggered globalized dual heuristic programming and its application to networked control systems. IEEE Transactions on Industrial Informatics 15, 1383–1392.

Zhou, Y., van Kampen, E., Chu, Q.P., 2016a. Incremental model based heuristic dynamic programming for nonlinear adaptive flight control, in: Proceedings of the International Micro Air Vehicles Conference and Competition 2016, Beijing, China.

Zhou, Y., van Kampen, E., Chu, Q.P., 2016b. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. Journal of Guidance, Control, and Dynamics 40, 493–496.

Zhou, Y., van Kampen, E., Chu, Q.P., 2017. Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming, in: 68th International Astronautical Congress (IAC), Adelaide, Australia.

Zhou, Y., van Kampen, E., Chu, Q.P., 2018a. Incremental approximate dynamic programming for nonlinear adaptive tracking control with partial observability. Journal of Guidance, Control, and Dynamics 41, 2554–2567.

Zhou, Y., van Kampen, E., Chu, Q.P., 2018b. Incremental model based online dual heuristic programming for nonlinear adaptive control. Control Engineering Practice 73, 13–25.