Launch vehicle discrete-time optimal tracking control using global dual heuristic programming

Sun, Bo; Van Kampen, Erik Jan

DOI
10.1109/CCTA41146.2020.9206252

Publication date
2020

Document Version
Accepted author manuscript

Published in
CCTA 2020 - 4th IEEE Conference on Control Technology and Applications

Launch Vehicle Discrete-Time Optimal Tracking Control using Global Dual Heuristic Programming*

Bo Sun[1] and Erik-Jan van Kampen[1]

***Abstract*— Optimal tracking is a widely researched control problem, but the unavailability of sufficient information referring to system dynamics brings challenges. In this paper, an optimal tracking control method is proposed for an unknown launch vehicle based on the global dual heuristic programming technique. The nonlinear system dynamics is identified by an offline trained neural network and a feedforward neuro-controller is developed to obtain the desired system input and to facilitate the execution of the feedback controller. By transforming the tracking control problem into a regulation problem, an iterative adaptive dynamic programming algorithm, subject to global dual heuristic programming with explicit analytical calculations, is utilized to deal with the newly built regulation problem. The simulation results demonstrate that the developed method can learn an effective control law for the given optimal tracking control tasks.**

I. INTRODUCTION

It is generally recognised that the optimal tracking control problems have gained much attention in the control field, because the demands are closely linked with many real world applications [1], [2]. However, handling the optimal control problems for nonlinear systems generally involves solving the nonlinear Hamilton-Jacobi-Bellman (HJB) rather than the Riccati equation [3], which makes it intractable to be dealt with, in that nonlinear partial difference equations are required to be solved but analytical solutions are difficult to obtain [3], [4]. Although dynamic programming (DP) provides a useful implement to handle the optimal control problems for several decades, it often suffers from computational burden because of the widely known "curse of dimensionality". Moreover, sometimes even full information referring to the dynamics of the nonlinear systems is not available, which puts more challenges on controller design for complex nonlinear systems, such as the aerospace systems. Consequently, it is demanded to investigate effective optimal tracking control methods for unknown nonlinear systems.

In recent years, adaptive dynamic programming (ADP) and associated research have been paid much attention to because of its self-learning property and been widely applied to different tracking control problems in the aerospace applications, such as launch vehicle [5], [6], airplane [7], satellite [8], etc. Developed from DP, ADP is proposed to iteratively solve optimal control problems in a forward-in-time way. When combined with neural networks (NNs) and the actor-critic scheme, ADP has stronger generalization capability and thus can be applied nonlinear systems. According to the information used for policy evaluation, ADP approaches are generally categorized into several major structures: heuristic dynamic programming (HDP), dual heuristic programming (DHP) global dual heuristic programming (GDHP) as well as their action-dependent schemes [6]. Among them, GDHP combines the information used by HDP and DHP, so it takes advantage of the latter two schemes [7]. Most researches on GDHP employ the straight form, in which the critic approximates the cost-to-go and its derivatives simultaneously [3], [4], [6], [9]. However, two different outputs sharing the same input and hidden layers results in strong coupling. To tackle this limitation, [7] derives a direct method to analytically compute the mixed second-order derivatives, which has been successfully applied to an online flight control problem. Nevertheless, due to difficulty of satisfying the optimal persistence excitation (PE) condition for online applications, [7] does not prove the convergence of the method.

This paper aims to apply the model-free GDHP technique with analytical calculations to a discrete-time optimal tracking flight control problem. Because the model network is pretrained offline, the convergence property can be analyzed. The remainder of this paper is organized as follows. Section II gives the formulation of the discrete-time optimal tracking control task. In section III, we introduces the iterative ADP algorithm and discusses its convergence. Section IV presents the NN implementation of the GDHP technique. In Section V, an example about launch vehicle attitude tracking control is given to verify the proposed control scheme. Finally, we discuss the conclusions and future research in section VI.

II. PROBLEM STATEMENT

Consider a class of discrete-time affine systems which can be presented as:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + g(\mathbf{x}_k)\mathbf{u}_\mathrm{s}(\mathbf{x}_k) \tag{1}$$

in which $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector, $\mathbf{u}_\mathrm{s}(\mathbf{x}_k) \in \mathbb{R}^m$ is the control vector. $f(\cdot)$ and $g(\cdot)$ are differentiable in their argument with $f(0) = 0$. Assume that $f + g\mathbf{u}_\mathrm{s}$ is Lipschitz continuous on a set $\Omega_\mathrm{s} \subset \mathbb{R}^n$ containing the origin and that the system (1) is controllable on Ω_s. Assume that the generalised inverse of $g(\cdot)$ exists such that $g^{-1}(\cdot)g(\cdot) = \mathbf{I}_m \in \mathbb{R}^{m\times m}$, where $\mathbf{I}_m$ denotes the identity matrix and the subscript m gives the dimensionality. For simplicity, $\mathbf{u}(\mathbf{x}_k)$ is represented by $\mathbf{u}_k$ in the rest of this paper.

*The first author's Ph.D. is financially supported by China Scholarship Council, project reference number 201806290007.

[1]B. Sun and E. van Kampen are with the Department of Control and Operations, Delft University of Technology, 2629HS Delft, The Netherlands. Email:{B.Sun-1, E.vanKampen}@tudelft.nl.

This paper aims to tackle optimal tracking control problems, so the reference trajectory $\mathbf{r}_k$ is assumed to be bounded and satisfies:

$$\mathbf{r}_{k+1} = \phi(\mathbf{r}_k) \tag{2}$$

where $\phi(\cdot)$ is a Lipschitz continuous vector function with $\phi(0) = 0$. Therefore the tracking error can be defined as $\mathbf{e}_k = \mathbf{x}_k - \mathbf{r}_k$.

There exists a sequence of controls corresponding to the reference trajectory $\mathbf{r}_k$, namely desired control. Inspired by the work of [1], [2], [10], the desired control is as follows:

$$\mathbf{u}_{\mathrm{d}k} = g^{-1}(\mathbf{r}_k)(\phi(\mathbf{r}_k) - f(\mathbf{r}_k)) \tag{3}$$

Then, define the feedback control $\mathbf{u}_k$ as $\mathbf{u}_k = \mathbf{u}_{\mathrm{s}k} - \mathbf{u}_{\mathrm{d}k}$, and a new system is built:

$$\begin{cases} \mathbf{e}_{k+1} = & f(\mathbf{e}_k + \mathbf{r}_k) + g(\mathbf{e}_k + \mathbf{r}_k)g^{-1}(\mathbf{r}_k)\times \\ & (\phi(\mathbf{r}_k) - f(\mathbf{r}_k)) - \phi(\mathbf{r}_k) + g(\mathbf{e}_k + \mathbf{r}_k)\mathbf{u}_k \\ \mathbf{r}_{k+1} = & \phi(\mathbf{r}_k) \end{cases} \tag{4}$$

In the new system (4), $\mathbf{e}_k$ and $\mathbf{r}_k$ are regarded as the system states and $\mathbf{u}_k$ is regarded as system input. Note that the second equation only provides the evolution of the reference trajectory and is independent from the system input. Hence, (4) can be concisely represented as:

$$\mathbf{e}_{k+1} = F(\mathbf{e}_k, \mathbf{u}_k) \tag{5}$$

where $F(\mathbf{e}_k, \mathbf{u}_k)$ is a Lipschitz continuous on a set $\Omega_{\mathrm{e}} \subset \mathbb{R}^n$ with $F(0, 0)$. Therefore, $\mathbf{e} = 0$ is an equilibrium state of system (5) with $\mathbf{u} = 0$.

Let $\mathbf{v}(\mathbf{e}_k)$ denote an arbitrary feedback control law and $\underline{\mathbf{u}}_k = \{\mathbf{u}_k, \mathbf{u}_{k+1}, \mathbf{u}_{k+2}, ...\}$ be the control chain from k to ∞ produced by $\mathbf{v}(\mathbf{e}_k)$. The performance index function is given as:

$$J(\mathbf{e}_0, \underline{\mathbf{u}}_0) = \sum_{k=0}^{\infty} \gamma^k U(\mathbf{e}_k, \mathbf{u}_k) \tag{6}$$

where $\mathbf{e}_0$ denotes the initial state, γ denotes the forgetting factor, and $0 < \gamma \leq 1$. γ is set to be 1 in this paper. U is the utility function and can be defined in a classical quadratic form:

$$U(\mathbf{e}_k, \mathbf{u}_k) = \mathbf{e}_k^{\mathrm{T}}\mathbf{Q}\mathbf{e}_k + \mathbf{u}_k^{\mathrm{T}}\mathbf{R}\mathbf{u}_k \tag{7}$$

in which $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ are positive semi-definite and positive definite matrices, respectively.

In this way, the original tracking problem is transformed into a regular optimal control task. The designed feedback control is supposed not only to stabilize the system (5) but also to prevent (6) from being infinite, i.e. the control is admissible.

Definition 1. *[3] A control $\mathbf{v}(\mathbf{e})$ is said to be admissible w.r.t. (6) on Ω_e if $\mathbf{v}(\mathbf{e})$ is continuous on a compact set $\Omega_v \subset \mathbb{R}^m$ including the origin, $\mathbf{v}$ stabilizes (5) on Ω_e, and $\forall \mathbf{e}_0 \in \Omega_e$, J is finite.*

As claimed by Bellman's principle of optimality, the optimal performance index function satisfies the discrete-time HJB (DTHJB) equation:

$$J^*(\mathbf{e}_k) = \min_{\underline{\mathbf{u}}_k} J(\mathbf{e}_k, \underline{\mathbf{u}}_k) = \min_{\mathbf{u}_k}\{U(\mathbf{e}_k, \mathbf{u}_k) + J^*(\mathbf{e}_{k+1})\} \tag{8}$$

where $\bullet^*$ stands for the optimal value of $\bullet$. Therefore, the optimal control law can be expressed as:

$$\mathbf{v}^*(\mathbf{e}_k) = \arg\min_{\mathbf{u}_k}\{U(\mathbf{e}_k, \mathbf{u}_k) + J^*(\mathbf{e}_{k+1})\} \tag{9}$$

From (7) and (9), the optimal control law is given as:

$$\mathbf{v}^*(\mathbf{e}_k) = -\frac{1}{2}\mathbf{R}^{-1}g^{\mathrm{T}}(\mathbf{e}_k + \mathbf{r}_k)\frac{\partial J^*(\mathbf{e}_{k+1})}{\partial \mathbf{e}_{k+1}} \tag{10}$$

According to (8) and (10), the DTHJB equation (8) can be represented as:

$$J^*(\mathbf{e}_k) = J(\mathbf{e}_k, \mathbf{v}_k^*) = U(\mathbf{e}_k, \mathbf{v}_k^*) + J^*(\mathbf{e}_{k+1}) \tag{11}$$

Launch vehicles are generally nonlinear and the partial derivative of $J^*(\mathbf{e}_{k+1})$ is intractable to compute analytically. Consequently, an iterative ADP algorithm is employed to iteratively solve the DTHJB equation in the next section.

III. The Iterative ADP Algorithm

A. Algorithm Procedure

By convention, value function $V(\cdot)$ and reward function $v(\cdot)$ are introduced for derivation, which are compatible with the perform index function $J(\cdot)$ and utility function $U(\cdot)$, respectively.

The procedure starts from a initial value function $V_0(\cdot) = 0$ and solving the following equation:

$$\mathbf{v}_0(\mathbf{e}_k) = \arg\min_{\mathbf{u}_k}\{U(\mathbf{e}_k, \mathbf{u}_k) + V_0(\mathbf{e}_{k+1})\} \tag{12}$$

After that, the cost value function can be updated by:

$$V_1(\mathbf{e}_k) = \mathbf{e}_k^{\mathrm{T}}\mathbf{Q}\mathbf{e}_k + \mathbf{v}_0^{\mathrm{T}}(\mathbf{e}_k)\mathbf{R}\mathbf{v}_0(\mathbf{e}_k) \tag{13}$$

Next, for $i = 1, 2, \cdots$, the ADP algorithm iterates between policy improvement

$$\begin{aligned} \mathbf{v}_i(\mathbf{e}_k) &= \arg\min_{\mathbf{u}_k}\{U(\mathbf{e}_k, \mathbf{u}_k) + V_i(\mathbf{e}_{k+1})\} \\ &= -\frac{1}{2}\mathbf{R}^{-1}g^{\mathrm{T}}(\mathbf{e}_k + \mathbf{r}_k)\frac{\partial V_i(\mathbf{e}_{k+1})}{\partial \mathbf{e}_{k+1}} \end{aligned} \tag{14}$$

and policy evaluation

$$V_{i+1}(\mathbf{e}_k) = \mathbf{e}_k^{\mathrm{T}}\mathbf{Q}\mathbf{e}_k + \mathbf{v}_i^{\mathrm{T}}(\mathbf{e}_k)\mathbf{R}\mathbf{v}_i(\mathbf{e}_k) + V_i(\mathbf{e}_{k+1}) \tag{15}$$

where $\mathbf{e}_{k+1} = F(\mathbf{e}_k, \mathbf{v}_i(\mathbf{e}_k))$, i and k denote iterative and time indices, respectively. The iteration will not stop until the value function and control law converge to the optimal ones, i.e. $V_i \to J^*$ and $\mathbf{v}_i \to \mathbf{u}^*$ as $i \to \infty$.

B. Convergence analysis

The convergence analysis of the iteration between (14) and (15) is presented based on the work of [1], [2], [11] as follows.

Lemma 1. *Let $\{\mathbf{v}_i\}$ be the control sequence expressed in (14) and $\{\mu_i\}$ be an arbitrary sequence of control laws. Define $V_i(\mathbf{e}_k)$ as in (15) and Λ_i as:*

$$\Lambda_{i+1}(\mathbf{e}_k) = \mathbf{e}_k^T \mathbf{Q} \mathbf{e}_k + \mu_i^T \mathbf{R} \mu_i + \Lambda_i(\mathbf{e}_{k+1}) \tag{16}$$

If $V_0(\mathbf{e}_k) = \Lambda_0(\mathbf{e}_k) = 0$, then $V_i(\mathbf{e}_k) \leq \Lambda_i(\mathbf{e}_k)$, $\forall i$.

Proof. The proof is omitted here for space reasons. □

Lemma 2. *Let the value function sequence $\{V_i\}$ be defined in (15). Given the system is controllable, there exists an upper bound Y which satisfies $0 \leq V_i(\mathbf{e}_k) \leq Y$, $\forall i$.*

Proof. The proof is omitted here for space reasons. □

Based on Lemma 1 and 2, the following theorem can be deduced.

Theorem 1. *Define the control law sequence and the value function sequence $\{V_i\}$ as in (14) and (15), respectively, with $V_0(\cdot) = 0$. Then, $V_i(\mathbf{e}_k)$ is a non decreasing sequence satisfying $V_i(\mathbf{e}_k) \leq V_{i+1}(\mathbf{e}_k)$, $\forall i$. Moreover, as $i \to \infty$, $V_i \to J^*$, i.e.,*

$$\lim_{i \to \infty} V_i = \min_{\mathbf{u}_k} \{U(\mathbf{e}_k, \mathbf{u}_k) + V_0(\mathbf{e}_{k+1})\} \tag{17}$$

Proof. The proof is omitted here for space reasons. □

Consequently, it can be concluded that the value sequence $\{V_i\}$ converges to the optimal one J^*. Moreover, according to (14), the corresponding control law sequence $\{\mathbf{v}_i\}$ converges to the optimal one $\mathbf{v}^*$, i.e., $\mathbf{v}_i \to \mathbf{v}^*$ as $i \to \infty$.

IV. NN Implementation of the ADP Algorithm using the GDHP Technique

To carry out the iterative ADP algorithm, function approximation structures, such as NNs, are utilized to provide the system information, and to approximate $\mathbf{v}_i(\mathbf{e}_k)$, $V_i(\mathbf{e}_k)$ and $\mathbf{u}_{dk}$. The architecture of the algorithm is shown in Fig. 1.

A. The Model Network

The iterative ADP algorithm requires the state information at the next time instant. Therefore, a NN identifier with single hidden layer is employed to approximate the dynamics of the unknown nonlinear system. According to the universal approximation property of NN, there is a NN representation of the system dynamics (1) on a compact set Ω_{m}, and it can generally be given as:

$$\mathbf{x}_{k+1} = \mathbf{w}_{m2}^{*\mathrm{T}} \sigma(\mathbf{w}_{m1}^{*\mathrm{T}} \mathbf{z}_k) + \epsilon_{mk} \tag{18}$$

In (18), $\mathbf{z}_k = [\mathbf{x}_k \ \ \mathbf{u}_k]^{\mathrm{T}}$ is the input, $\mathbf{w}_{m1}^*$ and $\mathbf{w}_{m2}^*$ are the ideal weight matrices between the input layer and hidden layer, and between the hidden layer and output layer, respectively, $\sigma(\cdot)$ is the element-wise activation function of the hidden layer, and ϵ_{mk} is the bounded approximation error. The activation function is set as a sigmoid function [6] in this paper such that it is bounded and $||\sigma(\cdot)|| \leq 1$.

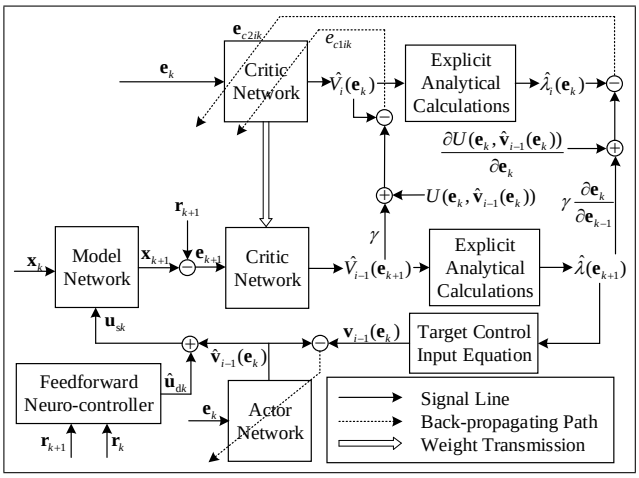

Fig. 1. The architecture diagram of the iterative GDHP algorithm (adapted from [9]).

The system identification scheme with the model network is presented as:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{w}_{m2}^{\mathrm{T}} \sigma(\mathbf{w}_{m1}^{\mathrm{T}} \mathbf{z}_k) \tag{19}$$

where $\hat{\cdot}$ stands for the estimated or predicted value of $\cdot$. The approximation error to be minimized is defined as follows:

$$E_{mk} = \frac{1}{2} \mathbf{e}_{mk}^{\mathrm{T}} \mathbf{e}_{mk} \tag{20}$$

where

$$\mathbf{e}_{mk} = \hat{\mathbf{x}}_{k+1} - \mathbf{x}_{k+1} \tag{21}$$

is the error between the measured state vector $\mathbf{x}_{k+1}$ and the estimated state vector $\hat{\mathbf{x}}_{k+1}$. Then, a gradient-descent algorithm is used to tune the weights:

$$\mathbf{w}_{m2}(j+1) = \mathbf{w}_{m2}(j) - \eta_m \cdot \frac{\partial E_{mk}}{\partial \mathbf{w}_{m2}(j)} \tag{22}$$

$$\mathbf{w}_{m1}(j+1) = \mathbf{w}_{m1}(j) - \eta_m \cdot \frac{\partial E_{mk}}{\partial \mathbf{w}_{m1}(j)} \tag{23}$$

where η_m denotes the learning rate and j denotes the iterative step for updating.

Remark 1. *The model network are trained offline and its weights should be kept unchanged for controller design processes.*

B. The Critic Network

Combining HDP and DHP, GDHP approximates the information of both $V_i(\mathbf{e}_k)$ and its derivative w.r.t. the input of the critic network $\mathbf{e}_k$, i.e. $\partial V_i(\mathbf{e}_k)/\partial \mathbf{e}_k$, which is denoted by $\lambda_i(\mathbf{e}_k)$. Different from the straight form presented in [3], [4], [6], [9], the output of the critic network in this paper is only the approximated value function $\hat{V}_i(\mathbf{e}_k)$:

$$\hat{V}_i(\mathbf{e}_k) = \mathbf{w}_{c2}^{\mathrm{T}} \sigma(\mathbf{w}_{c1}^{\mathrm{T}} \mathbf{e}_k) \tag{24}$$

Then analytical calculation [7] is carried out to compute $\hat{\lambda}_i(\mathbf{e}_k)$ directly using $\hat{V}_i(\mathbf{e}_k)$:

$$\hat{\lambda}_i(\mathbf{e}_k) = \frac{\partial \hat{V}_i(\mathbf{e}_k)}{\partial \mathbf{e}_k} = \mathbf{w}_{c1}(\mathbf{w}_{c2} \odot \sigma'(\mathbf{w}_{c1}^{\mathrm{T}}\mathbf{e}_k)) \tag{25}$$

where $\odot$ is the Hadamard product, and $\sigma'(\bullet)$ is the first order derivative of σ.

Remark 2. *(25) is the derivative form of (24) , so they are equivalent for convergence in principle. Since $V_i \to J^*$ as $i \to \infty$, it can be concluded that the sequence $\{\lambda_i\}$ also converges, i.e. $\lambda_i \to \lambda^*$ as $i \to \infty$.*

Similarly, the critic network is also trained by minimizing the errors between the estimated value and the optimal one. The critic errors are given as follows:

$$e_{c1ik} = \hat{V}_i(\mathbf{e}_k) - V_{i+1}(\mathbf{e}_k) \tag{26}$$

$$\mathbf{e}_{c2ik} = \hat{\lambda}_i(\mathbf{e}_k) - \lambda_{i+1}(\mathbf{e}_k) \tag{27}$$

GDHP merges them into an overall error function E_{cik}:

$$E_{cik} = \beta\frac{1}{2}e_{c1ik}^2 + (1-\beta)\frac{1}{2}\mathbf{e}_{c2ik}^{\mathrm{T}}(t)\mathbf{e}_{c2ik} \tag{28}$$

in which β denotes the importance scalar within a range of $[0, 1]$. If $\beta = 1$, then it becomes pure HDP, and if $\beta = 0$, it is equivalent to DHP. Then the critic network is also trained by the gradient-descent algorithm with a learning rate η_c:

$$\mathbf{w}_{c2i}(j+1) = \mathbf{w}_{c2i}(j) - \eta_c \cdot \frac{\partial E_{cik}}{\partial \mathbf{w}_{c2i}(j)} \tag{29}$$

$$\mathbf{w}_{c1i}(j+1) = \mathbf{w}_{c1i}(j) - \eta_c \cdot \frac{\partial E_{cik}}{\partial \mathbf{w}_{c1i}(j)} \tag{30}$$

C. The Actor Network

The actor network approximates the control law using the tracking error information:

$$\hat{\mathbf{v}}_i(\mathbf{e}_k) = \mathbf{w}_{a2}^{\mathrm{T}}\sigma(\mathbf{w}_{a1}^{\mathrm{T}}\mathbf{e}_k) \tag{31}$$

Considering (14), the target control policy $\mathbf{v}_i(\mathbf{e}_k)$ can be obtained by:

$$\mathbf{v}_i(\mathbf{e}_k) = -\frac{1}{2}\mathbf{R}^{-1}g^{\mathrm{T}}(\mathbf{e}_k + \mathbf{r}_k)\frac{\partial \hat{V}_i(\mathbf{e}_{k+1})}{\partial \mathbf{e}_{k+1}} \tag{32}$$

Therefore the overall error function of the actor network can is as follows:

$$E_{aik} = \frac{1}{2}\mathbf{e}_{aik}^{\mathrm{T}}\mathbf{e}_{aik} \tag{33}$$

where

$$\mathbf{e}_{aik} = \hat{\mathbf{v}}_i(\mathbf{e}_k) - \mathbf{v}_i(\mathbf{e}_k) \tag{34}$$

Similar to the model and critic networks, the actor network is trained in a back-propagation way with a learning rate η_a:

$$\mathbf{w}_{a2i}(j+1) = \mathbf{w}_{a2i}(j) - \eta_a \cdot \frac{\partial E_{aik}}{\partial \mathbf{w}_{a2i}(j)} \tag{35}$$

$$\mathbf{w}_{a1i}(j+1) = \mathbf{w}_{a1i}(j) - \eta_a \cdot \frac{\partial E_{aik}}{\partial \mathbf{w}_{a1i}(j)} \tag{36}$$

According to (32), the computation of the target control policy $\mathbf{v}_i(\mathbf{e}_k)$ needs the information of the control coefficient matrix $g(\mathbf{e}_k + \mathbf{r}_k) = g(\mathbf{x}_k)$, which, however, is not available directly. Therefore, the model network is used to estimate $g(\mathbf{x}_k)$:

$$\hat{g}(\mathbf{x}_k) = \frac{\partial \mathbf{w}_{m2}^{\mathrm{T}}\sigma(\mathbf{w}_{m1}^{\mathrm{T}}\mathbf{z}_k)}{\partial \mathbf{u}_{sk}} \tag{37}$$

D. The Feedforward Neuro-Controller

In addition, the obtained control law $\hat{\mathbf{v}}_i(\mathbf{e}_k)$ cannot directly be introduced into the model network and the real system before added with $\mathbf{u}_{dk}$. However, according to (3), computing $\mathbf{u}_{dk}$ requires the information of $g^{-1}(\mathbf{r}_k)$, $\phi(\mathbf{r}_k)$ and $f(\mathbf{r}_k)$, which is not available directly. Inspired by [1], the desired control $\mathbf{u}_{dk}$ is approximated by a feedforward neuro-controller using the trajectory reference at current and next time step:

$$\hat{\mathbf{u}}_{dk} = \mathbf{w}_{g2}^{\mathrm{T}}\sigma(\mathbf{w}_{g1}^{\mathrm{T}}[\mathbf{r}_k^{\mathrm{T}}\ \mathbf{r}_{k+1}^{\mathrm{T}}]^{\mathrm{T}}) \tag{38}$$

Since training the controller usually cannot be carried out with the real system due to safety reasons, the pretrained model network is involved, as shown in Fig. 2. $\hat{\mathbf{u}}_{dk}$ and $\mathbf{r}_k$ work as the inputs of the model network for outputting the estimated reference trajectory $\hat{\mathbf{r}}_{k+1}$. Therefore the overall error function of the feedforward neuro-controller is given as:

$$E_{gk} = \mathbf{e}_{gk}^{\mathrm{T}}\mathbf{Q}_g\mathbf{e}_{gk} \tag{39}$$

where

$$\mathbf{e}_{gk} = \hat{\mathbf{r}}_{k+1} - \mathbf{r}_{k+1} \tag{40}$$

and $\mathbf{Q}_g$ is the weights matrix in that perhaps not all state references are available and in these cases the corresponding weight of the absent reference is set to be 0.

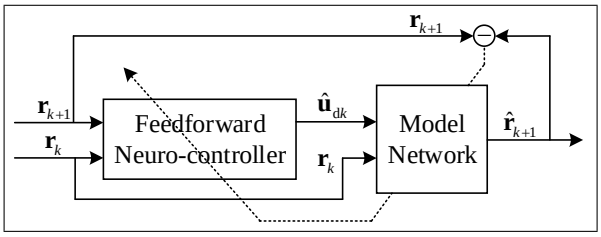

Fig. 2. The architecture diagram of the feedforward neuro-controller (adatpted from [1]).

Similarly, the network is trained in a back-propagation way with a learning rate η_g:

$$\mathbf{w}_{g2}(j+1) = \mathbf{w}_{g2}(j) - \eta_g \cdot \frac{\partial E_{gk}}{\partial \hat{\mathbf{r}}_{k+1}} \cdot \frac{\partial \hat{\mathbf{r}}_{k+1}}{\partial \hat{\mathbf{u}}_{dk}} \cdot \frac{\partial \hat{\mathbf{u}}_{dk}}{\partial \mathbf{w}_{g2}(j)} \tag{41}$$

$$\mathbf{w}_{g1}(j+1) = \mathbf{w}_{g1}(j) - \eta_g \cdot \frac{\partial E_{gk}}{\partial \hat{\mathbf{r}}_{k+1}} \cdot \frac{\partial \hat{\mathbf{r}}_{k+1}}{\partial \hat{\mathbf{u}}_{dk}} \cdot \frac{\partial \hat{\mathbf{u}}_{dk}}{\partial \mathbf{w}_{g1}(j)} \tag{42}$$

Note that $\partial \hat{\mathbf{r}}_{k+1}/\partial \hat{\mathbf{u}}_{dk}$ can be obtained by backpropagation from the model network output $\hat{\mathbf{r}}_{k+1}$ to its input $\hat{\mathbf{u}}_{dk}$.

V. Simulation Study

A. Aerospace System Model

The proposed method is applied to control a generic surface-to-air missile [5], [6] for verification. The nonlinear system is simplified to a second order model including the longitudinal force and moment equations, and the model has two system states: angle of attack α and pitch rate q, and one input: elevator deflection δ_e. At an altitude of approximately 6000 meters, the nonlinear model in the steady wings-level flight condition can be presented as:

$$\dot{\alpha} = q + \frac{\bar{q}S}{mV_T} C_z(\alpha, q, M_a, \delta_e) \tag{43}$$

$$\dot{q} = \frac{\bar{q}Sd_l}{I_{yy}} C_m(\alpha, q, M_a, \delta_e) \tag{44}$$

where $\bar{q}$, S, m, V_T, d_l, I_{yy}, M_a are dynamic pressure, reference area, mass, speed, reference length, pitching moment of inertia, and Mach number, respectively. C_z and C_m denote the aerodynamic force and moment coefficients, and are approximated by:

$$\begin{aligned}
C_z(\alpha, q, M_a, \delta_e) &= C_{z1}(\alpha, M_a) + B_z\delta_e \\
C_m(\alpha, q, M_a, \delta_e) &= C_{m1}(\alpha, M_a) + B_m\delta_e \\
C_{z1}(\alpha, M_a) &= \phi_{z1}(\alpha) + \phi_{z2}M_a \\
C_{m1}(\alpha, M_a) &= \phi_{m1}(\alpha) + \phi_{m2}M_a \\
\phi_{z1}(\alpha) &= h_1\alpha^3 + h_2\alpha|\alpha| + h_3\alpha \\
\phi_{m1}(\alpha) &= h_4\alpha^3 + h_5\alpha|\alpha| + h_6\alpha \\
\phi_{z2} &= h_7\alpha|\alpha| + h_8\alpha \\
\phi_{m2} &= h_9\alpha|\alpha| + h_{10}\alpha \\
B_z &= b_1M_a + b_2 \\
B_m &= b_3M_a + b_4
\end{aligned} \tag{45}$$

where $h_1, \cdots, h_{10}$, $b_1, \cdots, b_4$ are identified constant coefficients in a valid flight envelope of $\alpha \in (-10°, 10°)$ and $M_a \in (1.8, 2.6)$, and $M_a = 2.0$ thereafter.

Although the given system model is continuous, the computation and simulation in the computer are all discrete, and therefore the proposed discrete method can be utilized. The sampling time for discretizing the system is set to be 0.001 seconds.

B. Simulation Results

Model network, critic network, action network and feedforward neuro-controller all take a feedforward structure, with three layers of 3–20–2, 2–20–1, 2–30–1 and 4-50-1, respectively. The algorithm is carried out from the time instant $k = 0$. The initial weights of these NNs are randomly initialized in the range of $[-0.01, 0.01]$. Firstly, random samples are taken within $\alpha \in (-10°, 10°)$, $q \in (-20°/\text{s}, 20°/\text{s})$ and $\delta_e \in (-15°, 15°)$, and the model network is trained for 200 time steps by 5000 samples with $\eta_m = 0.002$. As shown in Fig. 3, the mean values and standard deviations of the identification errors converge to approximately 0 as the training progresses. After training the model network, its weights remain unchanged.

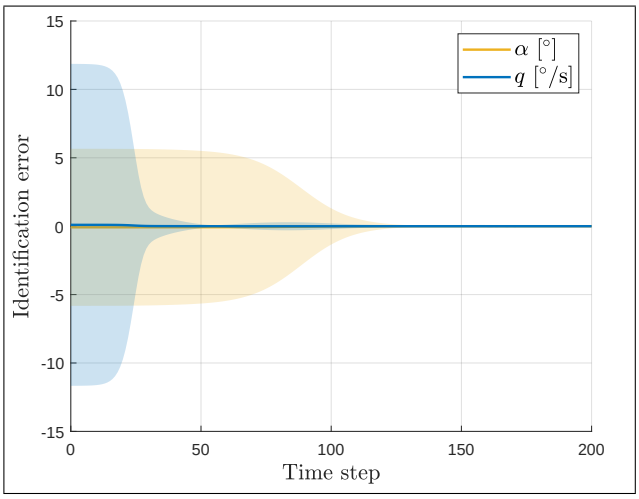

Fig. 3. The system identification errors.

Then the feedforward neuro-controller is trained with $\eta_g = 0.3$ and $\mathbf{Q}_g = \text{diag}([100, 1])$ for 500 time steps by 5000 data samples using the same sampling method as the model network. The weights are also kept unchanged after training and the trained controller is tested with a desired control from initial zero states. The results are illustrated in Fig. 4, from which it can be seen that the feedforward neuro-controller can learn the desired control $\mathbf{u}_{dk}$.

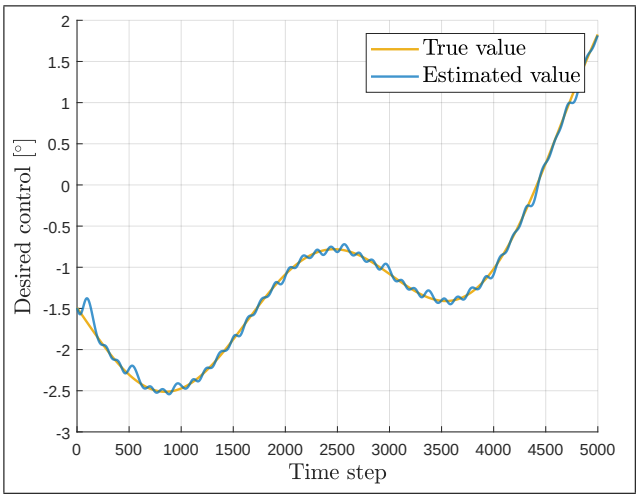

Fig. 4. Comparison between the estimated desired control produced by the feedforward neuro-controller and the true value.

Next, let $\gamma_c = 1$, $\beta = 0.5$, $\mathbf{Q} = \text{diag}([100, 1])$, $\mathbf{R} = 1$, $\eta_a = 0.001$ and $\eta_c = 0.01$, and the actor and critic networks are trained for 100 iterations (i.e., for $i = 1, 2, ..., 100$) with every iteration containing 10 epochs. After training, the proposed ADP algorithm is applied to an optimal tracking control problem of the aforementioned launch vehicle. The launch vehicle is supposed to track a given angle of attack reference α^{ref}, which is a sinusoidal function with respect to time step, i.e. $\alpha^{\text{ref}} = 8\sin(1000k)$ degrees. Because the pitch rate reference is not available directly, it is set to be 0 all the time. Then the relevant simulation results of the

first 10000 time steps (10 seconds for real world), with different initial angles of attack α_0 and zero initial pitch rates, are presented in Figs. 5-6. Fig. 5 presents the tracking performance when beginning from different initial angles of attack α_0, and Fig. 6 shows the corresponding elevator control signals outputted by the controller in these cases. It can be found that the launch vehicles can track the reference well in all cases. In both figures, the subfigure (a) presents detailed curves at the beginning of the control task, and all curves converge fast to similar values.

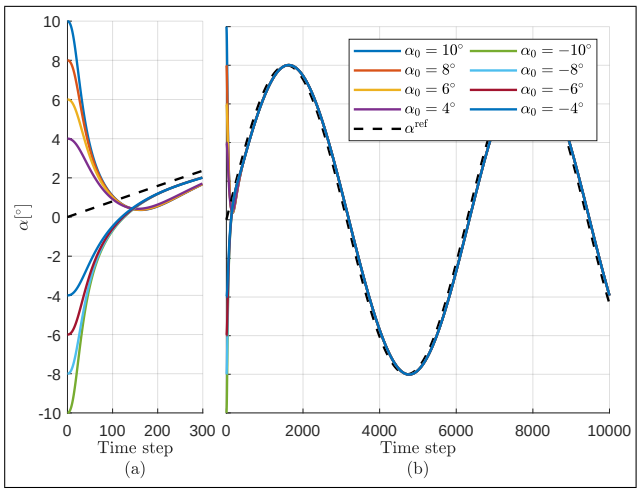

Fig. 5. The tracking performance using the GDHP algorithm with different initial angle of attack.

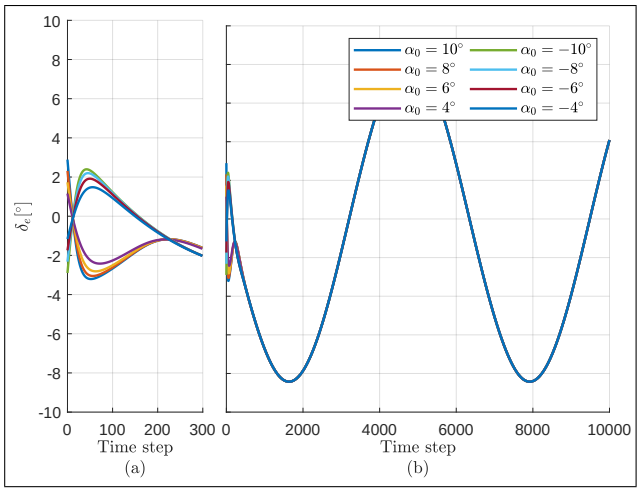

Fig. 6. The control input performance using the GDHP algorithm with different initial angle of attack.

VI. CONCLUSIONS

This paper proposes an iterative global dual heuristic programming (GDHP) algorithm in the discrete-time condition to cope with the optimal tracking control problems. The GDHP algorithm can effectively solve the discrete-time Hamilton-Jacobi-Bellman equation (DTHJB) with the facilitation of the neural networks (NNs), under the framework of the scheme of the iterative adaptive dynamic programming (ADP) with convergence analysis. Explicit analytical calculations are introduced to obtain the derivatives of the value function, which can eliminate the inconsistent errors of the traditional straight form of GDHP. Behaving as a model-free method, the algorithm is facilitated by a model network and a feedforward neuro-controller, which are utilized to learn the system dynamics and the inverse dynamics for approximating the desired control, respectively. The simulation results demonstrate that both the model network and the feedforward neuro-controller can effectively learn the objective dynamics and the proposed GDHP method can be successfully applied to the optimal tracking control task for the given launch vehicle.

Nevertheless, some problems need to be considered before realistic applications. In this launch vehicle control problem, the only useful reference is actually the angle of the attack while the pitch rate can be an interference term. Besides, the offline trained NNs cannot deal with the uncertainties and sudden changes in the practical scenarios. Therefore further study should concentrate on these topics.

REFERENCES

[1] Y. Huang and D. Liu, "Neural-network-based optimal tracking control scheme for a class of unknown discrete-time nonlinear systems using iterative adp algorithm," *Neurocomputing*, vol. 125, pp. 46–56, 2014.

[2] Q. Lin, Q. Wei, and D. Liu, "A novel optimal tracking control scheme for a class of discrete-time nonlinear systems using generalised policy iteration adaptive dynamic programming algorithm," *International Journal of Systems Science*, vol. 48, no. 3, pp. 525–534, 2017.

[3] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[4] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "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*, vol. 9, no. 3, pp. 628–634, 2012.

[5] Y. Zhou, E. van Kampen, and Q. P. Chu, "Launch vehicle adaptive flight control with incremental model based heuristic dynamic programming," in *Proceedings of the IAC 2017, Adelaide, Australia*, 2017.

[6] B. Sun and E.-J. van Kampen, "Incremental model-based global dual heuristic programming for flight control," *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 7–12, 2019.

[7] B. Sun and E. van Kampen, "Incremental model-based global dual heuristic programming with explicit analytical calculations applied to flight control," *Engineering Applications of Artificial Intelligence*, vol. 89, p. 103425, 2020.

[8] Y. Zhou, E. van Kampen, and Q. P. Chu, "Incremental approximate dynamic programming for nonlinear adaptive tracking control with partial observability," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 12, pp. 2554–2567, 2018.

[9] D. Liu, D. Wang, and X. Yang, "An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs," *Information Sciences*, vol. 220, pp. 331–342, 2013.

[10] D. Wang, D. Liu, and Q. Wei, "Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach," *Neurocomputing*, vol. 78, no. 1, pp. 14–22, 2012.

[11] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 621–634, 2014.