Autonomous Navigation in Partially Observable Environments Using Hierarchical Q-Learning

Zhou, Ye; van Kampen, Erik-Jan; Chu, Qiping

Publication date
2016

Document Version
Accepted author manuscript

Published in
Proceedings of the International Micro Air Vehicles Conference and Competition 2016

Citation (APA)
Zhou, Y., van Kampen, E.-J., & Chu, Q. (2016). Autonomous Navigation in Partially Observable Environments Using Hierarchical Q-Learning. In *Proceedings of the International Micro Air Vehicles Conference and Competition 2016: Beijing, China* (pp. 70-76). IEEE.

Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control

Y. Zhou *, E. van Kampen, and Q. P. Chu
Delft University of Technology, 2629HS Delft, The Netherlands

ABSTRACT

This paper presents a new and effective approach, incremental model based heuristic dynamic programming, to design an adaptive near-optimal controller without a-prior knowledge of the dynamic model. Both traditional heuristic dynamic programming algorithm and incremental model based heuristic dynamic programming algorithm are provided and applied to an illustrative on-line learning task. The system dynamics are completely unknown at the beginning, and the agent learns the local system models and the control policies on-line to follow a reference signal. It was found that using incremental models in heuristic dynamic programming can avoid off-line learning of the system model and help to accelerate the on-line learning. This proposed method can potentially design a near-optimal controller for autonomous flight of unmanned aerial vehicles without a-prior knowledge of the system dynamics.

1 INTRODUCTION

Control of a complex, nonlinear flying vehicle without sufficient knowledge of the system dynamics is a challenging problem to maintain functionality and safety in aviation. Until recent decades, adaptive control methods allow certain levels of robustness and fault-tolerance to be achieved. These methods in some form or another rely on off-line or/and on-line identification of air vehicles' dynamics and adaptation of control laws when necessary. However, on-line identification of unknown dynamical systems is not a trivial task especially when the system is complex and highly nonlinear.

In recent years, Adaptive/Approximate Dynamic Programming (ADP), which obtains approximately optimal solutions of the Hamilton-Jacobi-Bellman (HJB) equations, has been actively researched to solve nonlinear, optimal, fault-tolerant control problems [1, 2, 3, 4, 5]. Different from traditional Reinforcement Learning (RL) methods, ADP applies a function approximator with parameters to approximate the value/cost function to solve optimality problems with large or continuous state spaces on-line and to tackle the 'curse of dimensionality', which traditional RL methods often confront with [6, 7]. Adaptive Critic Designs (ACDs), which are also known as actor-critic designs, constitute a class of ADP methods that separate evaluation and improvement using parametric structures [2].

The most basic form and widely used structure of ACD is Heuristic Dynamic Programming (HDP). An action independent heuristic dynamic programming controller consists of an actor, a critic and an approximated plant structure connected between the actor and the critic [2, 6, 7]. An alternative approach is Action Dependent Heuristic Dynamic Programming (ADHDP), which does not need plant approximation, but has a direct connection from the output of the actor network to the input of the critic network. However, from the theoretical perspective, the actor output is not necessarily an input to the critic for estimating the optimal value function. From the practical point of view, extra input will increase the complexity of the critic network. Furthermore, some research has investigated the difference between HDP and ADHDP, and found that HDP controller with the approximated plant dynamics can operate in a wider range of flight conditions and has a higher success learning ratio in controlling an F-16 model [8]. Therefore, in this paper, only HDP, which refers to action independent heuristic dynamic programming, is considered.

Neural networks are most widely used as function approximators to approximate plants. However, this method has two main drawbacks which may lead to failure when applied in practice. First, on-line identification of the plant using neural networks needs certain time to approximate feasible model, which may even need an off-line identification beforehand. Second, neural networks may add two sources of errors. One is lacking adequate computing power when neural networks are used to perform the least-square approximation of the desired cost-to-go function. Another is that the function approximator is trained from a simulation model which might not be correct due to the unknown system [9].

Incremental methods are able to deal with system nonlinearity. These methods compute the required control increment instead of the total control input. However, some parts of the system model are still required in order to complete the design process [10, 11, 12]. Incremental Approximate Dynamic Programming (iADP) was developed for the first time to control nonlinear unknown systems without using models. This control strategy uses a quadratic function to approximate the value function [13, 14].

In this paper, an action independent Heuristic Dynamic

*Email address(es): Y.Zhou-6@tudelft.nl

Programming controller using incremental models, which is named *Incremental model based Heuristic Dynamic Programming* (IHDP), is developed as a model-free adaptive control approach for nonlinear unknown systems. This is called a model-free approach, because it does not need any a priori model information at the beginning of the algorithm nor on-line identification of nonlinear systems, but only the on-line identified linear incremental model. The incremental form of a nonlinear dynamic system is actually a linear time-varying approximation of the original system assuming sufficiently high sample rate for discretization. As the plant to be controlled in this paper is nonlinear, the IHDP is therefore developed based on the linearized incremental model of the original nonlinear system. This algorithm can be seen as an extension to the algorithm developed in [13, 14] with more general value function approximaters.

The rest of the paper is structured as follows. An HDP algorithm with a widely used neural network plant approximator is introduced and designed in section 2. An IHDP algorithm using incremental approach is first presented in section 3. Then, in section 4, the two algorithms are applied to an illustrative application, and the results are compared and discussed, showing how much the IHDP method can improve the performance. The last part concludes the advantages and disadvantages of using the incremental approach with HDP, and addresses the challenges and possibilities of the future research.

2 HEURISTIC DYNAMIC PROGRAMMING

Similar to other ADP methods, action independent Heuristic Dynamic Programming (HDP) algorithms operate by alternating between two steps: policy evaluation, implemented by the critic, and policy improvement, implemented by the actor [14, 9]. Fig.1 is a schematic diagram of an HDP controller, which uses 3 Neural Networks to approximate actor, critic, and system dynamics with weights w_a, w_c, and w_m, respectively.

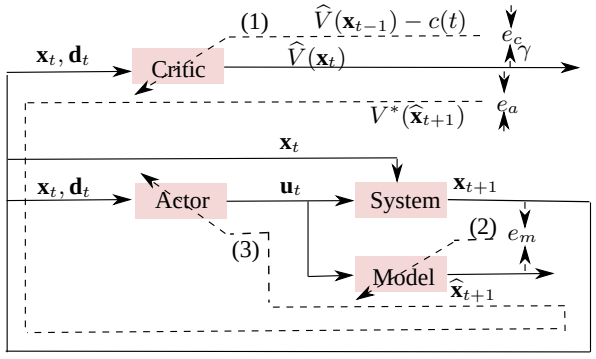

Figure 1: Architecture of HDP using Neural Networks to approximate system model

2.1 HDP using Neural Networks to approximate system model

2.1.1 Critic

The critic network is used to approximate the state-value function $V(x_t)$, which is the cumulative future rewards from any initial state $\mathbf{x}_t$:

$$V^{\mu}(\mathbf{x}_t) = \sum_{i=t}^{\infty} \gamma^{i-t} c_t, \tag{1}$$

where μ is the *current policy* for this algorithm, γ is called *discount factor* or *forgetting factor*, which is a scalar with $0 < \gamma < 1$, and c_t is the *one-step cost function*. The discount factor ensures that the cost for any state is finite and provides a reasonable evaluation and approximation to infinite-horizon problems as well as problems involving a finite but very large number of stages. By adjusting γ, it is able to control the extent to which the short-term cost or long-term cost is concerned [9].

To minimize the cost of the system approaching its goal, the one-step cost function is defined quadratically as a function of the difference between the current state and the desired state, as follows:

$$c_t = c(\mathbf{x}_t, \mathbf{d}_t) = (\mathbf{x}_t - \mathbf{d}_t)^T Q (\mathbf{x}_t - \mathbf{d}_t), \tag{2}$$

where $\mathbf{d}_t$ is the reference track, Q is a positive definite matrix. To normalize the effect of each state, we usually use normalization factors in the Q matrix. Thus, we let Q be a diagonal matrix, and Eq. 2 can be rewritten as follows:

$$c_t = \sum_{i=1}^{n} (\zeta_i)^2 \cdot \left(\frac{x_{t,i} - d_{t,i}}{x_{max,i}} \right)^2, \tag{3}$$

where ζ_i is a given weight to indicate the importance of the cost for the i-th state approaching the desired track.

Actor-Critic methods are on-policy Temporal Difference (TD) methods, which continually estimate the cost-to-go for the current policy by updating the critic, and change the policy towards greediness by updating the actor at the same time [15]. The evaluation of the critic is the TD error:

$$e_c(t) = c_{t-1} + \gamma \widehat{V}(\mathbf{x}_t) - \widehat{V}(\mathbf{x}_{t-1}), \tag{4}$$

where $\widehat{V}(\mathbf{x}_t)$ is the approximated cost-to-go from state $\mathbf{x}_t$ under current policy. Note that $\widehat{V}$ is a function of $\mathbf{x}_t$ and $\mathbf{w}_c(t)$ with a static neural network structure. The target for the critic update is $c_{t-1} + \gamma \widehat{V}(\mathbf{x}_t)$.

The critic network tries to minimize the defined error function:

$$E_c(t) = \frac{1}{2} e_c^2(t). \tag{5}$$

Therefore, the weights of the critic network are updated according to a gradient-descent algorithm with a learning rate η_c:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \Delta \mathbf{w}_c(t), \tag{6}$$

where

$$\begin{aligned}\Delta \mathbf{w}_c(t) &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \mathbf{w}_c(t)} \\ &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \widehat{V}(\mathbf{x}_t)} \cdot \frac{\partial \widehat{V}(\mathbf{x}_t)}{\partial \mathbf{w}_c(t)}.\end{aligned} \tag{7}$$

With a fixed policy and converged critic, the neural network parameters $\mathbf{w}_c(t)$ will be constant.

2.1.2 Actor

The actor is used to find the policy which minimizes the difference between the defined cost-to-go function $\widehat{V}(\mathbf{x}_t)$ and the goal $V^*(t)$:

$$E_a(t) = \frac{1}{2} e_a^2(t), \tag{8}$$

$$e_a(t) = \widehat{V}(\mathbf{x}_t) - V^*(t), \tag{9}$$

where the goal $V^*(t)$ is set to 0.

The policy is determined by the weights of the actor network. However, updating the actor network is more complicated, since it involves the critic network and the model network. Fig. 1 shows that, through the 3rd back-propagation direction, the actor weights affect cost-to-go function $V(\mathbf{x}_{t+1})$ through affecting $\mathbf{x}_{t+1}$ and $\mathbf{u}_t$. Thus, the actor network weights can be updated according to the gradient-descent algorithm with a learning rate η_a:

$$\mathbf{w}_a(t+1) = \mathbf{w}_a(t) + \Delta \mathbf{w}_a(t), \tag{10}$$

where

$$\begin{aligned}\Delta \mathbf{w}_a(t) &= -\eta_a \cdot \frac{\partial E_a(t+1)}{\partial \mathbf{w}_a(t)} \\ &= -\eta_a \cdot \frac{\partial E_a(t+1)}{\partial \widehat{V}(\mathbf{x}_{t+1})} \frac{\partial \widehat{V}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}} \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t} \frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}.\end{aligned} \tag{11}$$

The approximated system model can be used to estimate the next state $\mathbf{x}_{t+1}$ with an input $\mathbf{u}_t$. This helps to get the useful term $\frac{\partial \widehat{\mathbf{x}}(t+1)}{\partial \mathbf{u}_t}$ approximating $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$ in updating the actor network [8]. Therefore, Eq. 11 can be rewritten as follows:

$$\Delta \mathbf{w}_a(t) = -\eta_a \frac{\partial E_a(t+1)}{\partial \widehat{V}(\widehat{\mathbf{x}}_{t+1})} \frac{\partial \widehat{V}(\widehat{\mathbf{x}}_{t+1})}{\partial \widehat{\mathbf{x}}_{t+1}} \frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}(t)} \frac{\partial \mathbf{u}(t)}{\partial \mathbf{w}_a(t)}. \tag{12}$$

2.1.3 Model

The model network approximates the system dynamics and gives the estimated next state $\widehat{\mathbf{x}}_{t+1}$ as output. The next state is a function of input $\mathbf{u}_t$ and the network parameters $\mathbf{w}_m(t)$ with a fixed neural network structure: $\widehat{\mathbf{x}}_{t+1}(\mathbf{u}_t, \mathbf{w}_m(t))$. The update of the model network is by minimizing the difference between the measured state $\mathbf{x}_t$ and the estimated state $\widehat{\mathbf{x}}_t$:

$$E_m(t) = \frac{1}{2} e_m^2(t), \tag{13}$$

where

$$e_m(t) = \mathbf{x}_t - \widehat{\mathbf{x}}_t. \tag{14}$$

The model network weights are updated according to the gradient-descent algorithm with a learning rate η_m:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) + \Delta \mathbf{w}_m(t), \tag{15}$$

where

$$\begin{aligned}\Delta \mathbf{w}_m(t) &= -\eta_m \cdot \frac{\partial E_m(t+1)}{\partial \mathbf{w}_m(t)} \\ &= -\eta_m \cdot \frac{\partial E_m(t+1)}{\partial \widehat{\mathbf{x}}_{t+1}} \frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{w}_m(t)}.\end{aligned} \tag{16}$$

2.2 HDP training by back-propagation

The actor, critic and model neural networks in this paper are all Multilayer Perceptrons (MLP), which consist of multiple, fully connected, and feedforward layers of nodes. Each neural network has an input layer, a hidden layer and an output layer. Each node in hidden layers is a neuron with a continuous, nonlinear hyperbolic tangent activation function σ:

$$\sigma(y) = \frac{1 - e^{-y}}{1 + e^{-y}}. \tag{17}$$

At each point, it has a positive derivative:

$$\frac{\partial \sigma(y)}{\partial y} = \frac{1}{2}(1 - \sigma(y)^2). \tag{18}$$

In fully connected multilayer neural networks, the input of the $(n+1)$-th layer consists of the outputs of the n-th layer and sometimes also a bias term b_n. When the neural network has I inputs, J hidden neurons, and K outputs, the neural network weight from i-th input layer neuron to j-th hidden layer neuron is $w_{ji}(i = 1, ..., I+1, j = 1, ..., J)$, and the weight from j-th hidden layer neuron to k-th output layer neuron is $w_{kj}(j = 1, ..., J+1, k = 1, ..., K)$. Thus, the feedforward neural networks from i-th input layer (noted as superscript in) neuron to j-th neuron of the hidden layer (noted as superscript hi) can be described as follows:

$$\sigma_j^{hi}(t) = \frac{1 - e^{-y_j^{hi}(t)}}{1 + e^{-y_j^{hi}(t)}}, \tag{19}$$

$$y_j^{hi}(t) = \sum_{i=1}^{I+1} w_{ji}^{hi}(t) x_i^{in}(t), \tag{20}$$

where, $\sigma_j^{hi}(t)$ is the output of the j-th hidden layer neuron, $y_j^{hi}(t)$ is the network input of the j-th hidden layer neuron, $w_{ji}^{hi}(t)$ is the weight from the i-th input neuron to j-th hidden layer neuron at time t, and $x_i^{in}(t)$ is the i-th input of the hidden layer, which consists I inputs of the system and the bias term b^{in}. The feedforward neural networks from j-th hidden

layer neuron to k-th output layer (noted as superscript out) neuron can be described as follows:

$$y_k^{out}(t) = \sum_{j=1}^{J+1} w_{kj}^{out}(t) x_j^{hi}(t), \tag{21}$$

where, $y_k^{out}(t)$ is the output of the k-th output layer neuron, $w_{kj}^{out}(t)$ is the weight from the j-th hidden layer neuron to k-th output layer neuron at time t, and $x_j^{hi}(t)$ is the jth input of output layer, which consists J outputs of the hidden layer neurons and a bias term b^{hi}.

Because the output of a hyperbolic tangent function is bounded with $(-1, 1)$, and the outputs of the neural network is a summation with parameters, the neural network with bias items can approximate any value theoretically. Thus, the output of the neural networks is written as $\mathbf{O}(t)$:

$$\begin{aligned} \mathbf{O}(t) &= \mathbf{y}^{out}(t) \\ &= [y_1^{out}(t), y_2^{out}(t), ..., y_K^{out}(t)]^T. \end{aligned} \tag{22}$$

2.2.1 Critic and Model

To update the critic and model network weights (through the 1st and 2nd back-propagation directions in Fig. 1) according to Eq. 7 and Eq. 16, the partial derivative of the network output with respect to the network weights is needed:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial \mathbf{O}(t)} \frac{\partial \mathbf{O}(t)}{\partial w(t)}. \tag{23}$$

To be more specific, the partial derivative of each network output $y_k^{out}(t)$ with respect to the network weights from the hidden layer to the output layer $w_{kj}^{out}(t)$ and the weights from the input layer to the hidden layer $w_{ji}^{hi}(t)$ is shown below, respectively:

$$\frac{\partial y_k^{out}(t)}{\partial w_{kj}^{out}(t)} = x_j^{hi}(t), \tag{24}$$

$$\begin{aligned} \frac{\partial y_k^{out}(t)}{\partial w_{ji}^{hi}(t)} &= \frac{\partial y_k^{out}(t)}{\partial \sigma_j^{hi}(t)} \cdot \frac{\partial \sigma_j^{hi}(t)}{\partial y_j^{hi}(t)} \cdot \frac{\partial y_j^{hi}(t)}{\partial w_{ji}^{hi}(t)} \\ &= w_{kj}^{out}(t) \cdot \frac{1}{2}(1 - \sigma_j^{hi}(t)^2) \cdot x_i^{in}(t). \end{aligned} \tag{25}$$

2.2.2 Actor

To update the actor network weights, the network error E_a go through the critic network, model network, and finally the actor network along the 3rd back-propagation directions in Fig. 1. Thus, the partial derivative of the network output with respect to the network input $x_i^{in}(i = 1, ..., I)$ is also needed for the term $\frac{\partial \widehat{V}(\widehat{\mathbf{x}}_{t+1})}{\partial \widehat{\mathbf{x}}_{t+1}}$ and $\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}(t)}$ in Eq. 12 to update the weights:

$$\frac{\partial \mathbf{O}(t)}{\partial \mathbf{x}^{in}} = \frac{\partial \mathbf{O}(t)}{\mathbf{y}^{out}(t)} \frac{\mathbf{y}^{out}(t)}{\partial \mathbf{x}^{in}}. \tag{26}$$

The partial derivative of the k-th output with respect to the i-th input can be calculated as follows:

$$\begin{aligned} \frac{\partial y_k^{out}(t)}{\partial x_i^{in}} &= \sum_{j=1}^{J} \left[\frac{\partial y_k^{out}(t)}{\partial \sigma_j^{hi}(t)} \cdot \frac{\partial \sigma_j^{hi}(t)}{\partial x_i^{in}} \right] \\ &= \sum_{j=1}^{J} \left[w_{kj}^{out}(t) \cdot \frac{1}{2}(1 - \sigma_j^{hi}(t)^2) w_{ji}^{hi}(t) \right]. \end{aligned} \tag{27}$$

3 Incremental Heuristic Dynamic Programming

Fig.2 is the diagram for implementation of an Incremental model based Heuristic Dynamic Programming (IHDP) controller. It uses 2 Neural Networks to approximate the actor and critic with weights w_a and w_c, and an incremental model to find the system dynamics at a certain moment. The updating of the critic weights is the same as updating critic in HDP algorithm, which will not be reiterated in this section. This section focus on the new idea of using the incremental approach to approximate $\frac{\partial \mathbf{x}(t+1)}{\partial u(t)}$ in Eq. 11 as part of updating actor.

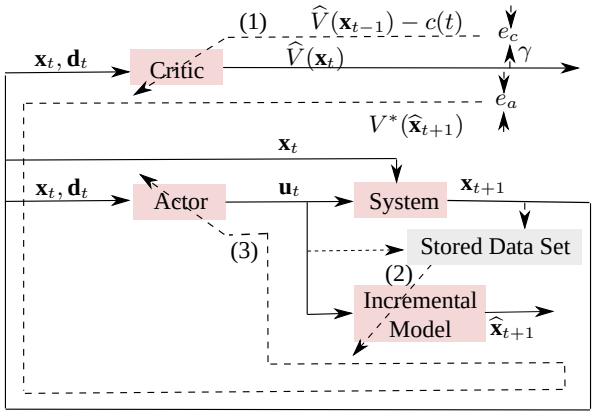

Figure 2: Architecture of HDP using incremental approach to approximate system model

3.1 HDP using incremental approach

3.1.1 Incremental Model

Many physical system, such as aircraft, are highly non-linear and can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)], \tag{28}$$

$$\mathbf{y}(t) = h[\mathbf{x}(t)], \tag{29}$$

where Eq. 28 is the *kinematic state equation*, in which $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathcal{R}^n$ provides the physical evaluation of the state vector over time, Eq. 29 is the *output (observation) equation*, which can be measured using sensors, $h[\mathbf{x}(t)] \in \mathcal{R}^p$ is a vector denoting the measured output.

The system dynamics around the condition of the system at time t_0 can be linearized approximately by using the first-order Taylor series expansion:

$$\begin{aligned}\dot{\mathbf{x}}(t) \simeq\ & f[\mathbf{x}(t_0), \mathbf{u}(t_0)] \\ & + \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{x}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)}[\mathbf{x}(t) - \mathbf{x}(t_0)] \\ & + \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{u}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)}[\mathbf{u}(t) - \mathbf{u}(t_0)] \\ =\ & \dot{\mathbf{x}}(t_0) + F[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{x}(t) - \mathbf{x}(t_0)] \\ & + G[\mathbf{x}(t_0), \mathbf{u}(t_0)][\mathbf{u}(t) - \mathbf{u}(t_0)],\end{aligned} \tag{30}$$

where $F[\mathbf{x}(t), \mathbf{u}(t)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{x}(t)} \in \mathcal{R}^{n\times n}$ is the *system matrix* at time t, and $G[\mathbf{x}(t), \mathbf{u}(t)] = \frac{\partial f[\mathbf{x}(t), \mathbf{u}(t)]}{\partial \mathbf{u}(t)} \in \mathcal{R}^{n\times m}$ is the *control effectiveness matrix* at time t.

We assume that the states and state derivatives of the system are measurable, which means $\Delta\dot{\mathbf{x}}(t), \Delta\mathbf{x}(t), \Delta\mathbf{u}(t)$ are measurable. Under this assumption, the model around time t_0 can be written in the incremental form:

$$\begin{aligned}\Delta\dot{\mathbf{x}}(t) \simeq\ & F[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{x}(t) \\ & + G[\mathbf{x}(t_0), \mathbf{u}(t_0)]\Delta\mathbf{u}(t).\end{aligned} \tag{31}$$

This current incremental model can be identified using least squares (LS) techniques and can be used to obtain an approximated value of $\frac{\partial \mathbf{x}(t+1)}{\partial u(t)}$ without using model networks in previous section.

The physical systems are generally continuous, but the data we collect are discrete samples. We assume that the control system has a constant, sufficiently high sampling frequency. With this constant data sampling rate, the non-linear system can be written in a discrete form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{32}$$

$$\mathbf{y}_t = h(\mathbf{x}_t), \tag{33}$$

where $f(\mathbf{x}_t, \mathbf{u}_t) \in \mathcal{R}^n$ provides the *system dynamics*, and $h(\mathbf{x}_t) \in \mathcal{R}^p$ is a vector denoting the measuring system.

By taking the Taylor expansion, we can get the system dynamics linearized around x_0:

$$\begin{aligned}\mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t) \\ &\simeq f(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_0,\mathbf{u}_0}(\mathbf{x}_t - \mathbf{x}_0) \\ &\quad + \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_0,\mathbf{u}_0}(\mathbf{u}_t - \mathbf{u}_0).\end{aligned} \tag{34}$$

When Δt is very small, $\mathbf{x}_{t-1}$ approximates $\mathbf{x}_t$. Thus, $\mathbf{x}_0, \mathbf{u}_0$ in Eq. 34 can be replaced by $\mathbf{x}_0 = \mathbf{x}_{t-1}$ and $\mathbf{u}_0 = \mathbf{u}_{t-1}$, and we obtain the discrete incremental form of this non-linear system:

$$\begin{aligned}\mathbf{x}_{t+1} - \mathbf{x}_t \simeq\ & F(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})(\mathbf{x}_t - \mathbf{x}_{t-1}) \\ & + G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})(\mathbf{u}_t - \mathbf{u}_{t-1}),\end{aligned} \tag{35}$$

$$\Delta\mathbf{x}_{t+1} \simeq F(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\Delta\mathbf{x}_t + G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\Delta\mathbf{u}_t, \tag{36}$$

where $F(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathcal{R}^{n\times n}$ is the *system transition matrix*, and $G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathcal{R}^{n\times m}$ is the *control effectiveness matrix* at time step $t-1$. Because of the high frequency sample data and the relatively slow-variant system assumption, the current linearized model can be identified by using the measured data in previous M steps.

3.1.2 Actor

The structure of the actor is the same as the one in HDP controller. It is used to minimize the difference between the cost-to-go function $\widehat{V}(\mathbf{x}_t)$ and the goal $V^*(t)$. However, updating this actor network is easier and faster than the one in HDP controller, since it involves a critic network and an incremental model.

Through the 3rd back propagation direction in Fig. 2, the actor weights affect cost-to-go function $V(\mathbf{x}_{t+1})$ also through affecting $\mathbf{x}_{t+1}$ and $\mathbf{u}_t$. The actor network weights can be updated according to the gradient-descent algorithm as shown in Eq. 10 and Eq. 11. The incremental model of the system can be used to approximate the derivative of the next state with respect to the input, $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$.

3.2 IHDP training by back-propagation and incremental model identification

3.2.1 Incremental Model

Since $\Delta\mathbf{x}(t), \Delta\mathbf{u}(t)$ are measurable as assumed, F_{t-1}, G_{t-1} are identifiable by using the simple equation error method:

$$\begin{aligned}\Delta x_{i,t-k+1} &= \mathbf{f}_i\Delta\mathbf{x}_{t-k} + \mathbf{g}_i\Delta\mathbf{u}_{t-k} \\ &= \begin{bmatrix}\Delta\mathbf{x}_{t-k}^T & \Delta\mathbf{u}_{t-k}^T\end{bmatrix}\begin{bmatrix}\mathbf{f}_i^T \\ \mathbf{g}_i^T\end{bmatrix},\end{aligned} \tag{37}$$

where $\Delta x_{i,t-k+1} = x_{i,t-k+1} - x_{i,t-k}$ is the increment of ith state element, $\mathbf{f}_i$ and $\mathbf{g}_i$ are the elements of ith row vector of F_{t-1}, G_{t-1}, and $k = 1, 2...M$ denotes at which time the historic information is available. Because there are $n+m$ parameters in the ith row, M needs to satisfy $M \geq (n+m)$. By using the Ordinary Least Squares (OLS) method, the linearized system dynamics (ith row) can be identified from M different data points:

$$\begin{bmatrix}\mathbf{f}_i^T \\ \mathbf{g}_i^T\end{bmatrix} = (\mathbf{A}_t^T\mathbf{A}_t)^{-1}\mathbf{A}_t^T\mathbf{y}_t, \tag{38}$$

where

$$\mathbf{A}_t = \begin{bmatrix}\Delta\mathbf{x}_{t-1}^T & \Delta\mathbf{u}_{t-1}^T \\ \vdots & \vdots \\ \Delta\mathbf{x}_{t-M}^T & \Delta\mathbf{u}_{t-M}^T\end{bmatrix},\ \mathbf{y}_t = \begin{bmatrix}\Delta x_{i,t} \\ \vdots \\ \Delta x_{i,t-M+1}\end{bmatrix}. \tag{39}$$

Choosing a suitable number of data M is also important. The identified incremental model will be used and is only capable of describing the system behaviour within a small time

range. When M is very large, the identified model may not represent the local linearized model of the nonlinear system. However, when M is too small, the linear system might be ill-conditioned, especially at which point the excitation is not sufficient. Thus, choosing M depends not only on the sampling frequency and non-linearity, but also the intensity of the excitation. In this paper, M is chosen to be $2 \cdot (n+m)$.

3.2.2 Actor

To update the actor network weights, the network error E_a go through the critic network, incremental model, and the actor network consequently along the 3rd back-propagation directions in Fig. 2. The partial derivative of the network output with respect to the network weights (Eq. 23 to 25) and the partial derivative of the network output with respect to the network input (Eq. 26, 27) are needed for the term $\frac{\partial \widehat{V}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}}$ and term $\frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}$ in Eq. 11, respectively.

With the linearized incremental model, the term $\frac{\partial \mathbf{x}(t+1)}{\partial u(t)}$ can be easily approximated:

$$\begin{aligned}\frac{\partial \mathbf{x}(t+1)}{\partial u(t)} &\simeq \partial[\mathbf{x}_t + F(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})(\mathbf{x}_t - \mathbf{x}_{t-1}) \\ &\quad + G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})(\mathbf{u}_t - \mathbf{u}_{t-1})]/\partial u(t) \\ &= G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}).\end{aligned} \tag{40}$$

This method simplified the approach of updating actor network weights and accelerate the learning with direct on-line identification of the incremental model.

4 Application

This section will present an illustrative application of both the HDP and the IHDP algorithms on a simulation model for validation. The flight control task is to track a changing reference, when there is input disturbances, which is a most basic and important control task for air vehicles.

4.1 Air vehicle model

A nonlinear air vehicle simulation model will be used in this section. Air vehicle models are highly nonlinear and can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t) + \mathbf{w}(t)], \tag{41}$$

$$\mathbf{y}(t) = h[\mathbf{x}(t)], \tag{42}$$

where Eq. 41 is the kinematic state equation, $\mathbf{w}(t)$ is the external disturbace, which is set to be caused only by the input noise, and Eq. 42 is the output equation.

As an application for these control algorithms, only elevator deflection will be regulated as pitch control to stabilize the air vehicles. Thus, we are interested in two longitudinal states, *angle of attack* α and *pitch rate* q (i.e. the system variables are $\mathbf{x} = [\alpha \;\; q]$), and one control input, *elevator deflection angle* δ_e.

The nonlinear model in the pitch plane is simulated around a steady wings-level flight condition:

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

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

where $\bar{q}$ is dynamic pressure, S is reference area, m_a is mass, V_T is speed, d is reference length, I_{yy} is pitching moment of inertia, C_z is the aerodynamic force coefficient, and C_m is the aerodynamic moment coefficient. C_z and C_m are highly nonlinear functions of angle of attack α, pitch rate q, Mach number M_a and elevator deflection δ_e.

As a preliminary test, an air vehicle model (parameter data) is taken in the pitch plane for $-10^o < \alpha < 10^o$ [16, 17]:

$$\begin{aligned}
&C_z(\alpha, q, M_a, \delta_e) = C_{z1}(\alpha, M_a) + B_z \delta_e, \\
&C_m(\alpha, q, M_a, \delta_e) = C_{m1}(\alpha, M_a) + B_m \delta_e, \\
&B_z = b_1 M_a + b_2, \\
&B_m = b_3 M_a + b_4, \\
&C_{z1}(\alpha, M_a) = \phi_{z1}(\alpha) + \phi_{z2} M_a, \\
&C_{z2}(\alpha, M_a) = \phi_{m1}(\alpha) + \phi_{m2} M_a, \\
&\phi_{z1}(\alpha) = h_1 \alpha^3 + h_2 \alpha|\alpha| + h_3 \alpha, \\
&\phi_{m1}(\alpha) = h_4 \alpha^3 + h_5 \alpha|\alpha| + h_6 \alpha, \\
&\phi_{z2} = h_7 \alpha|\alpha| + h_8 \alpha, \\
&\phi_{m2} = h_9 \alpha|\alpha| + h_{10} \alpha,
\end{aligned} \tag{45}$$

where $b_1, ..., b_4, h_1, ..., h_{10}$ are validated constant coefficients in the flight envelop [17].

To accomplish the reference tracking task, an adaptive controller with the actor need to be found out by minimizing the cost-to-go function $V(\mathbf{x}_t)$ with a feasible critic and model.

4.2 Results and Discussions

Two algorithms are applied to this problem: traditional HDP uses a neural network to approximate the plant model and IHDP uses the incremental approach. The identified models are used 1) to predict the next states, which is used to estimate the cost-to-go of the next state and its difference from the minimal cost, and 2) to estimate the control effective matrix, which is used to update the actor during the error back-propagation.

Fig.3 shows the one-step prediction of α and q, when there is a sine input excitation, using the on-line identified neural network model and the incremental model. As illustrated in Fig.3 (a), the one-step state predictions using both methods are feasible. However, the prediction using the neural network needs more time to learn at the beginning. When having a close look at the prediction errors with Fig.3 (b) and (c), the prediction using incremental approach has significantly higher precision. Fig.4 presents the identification

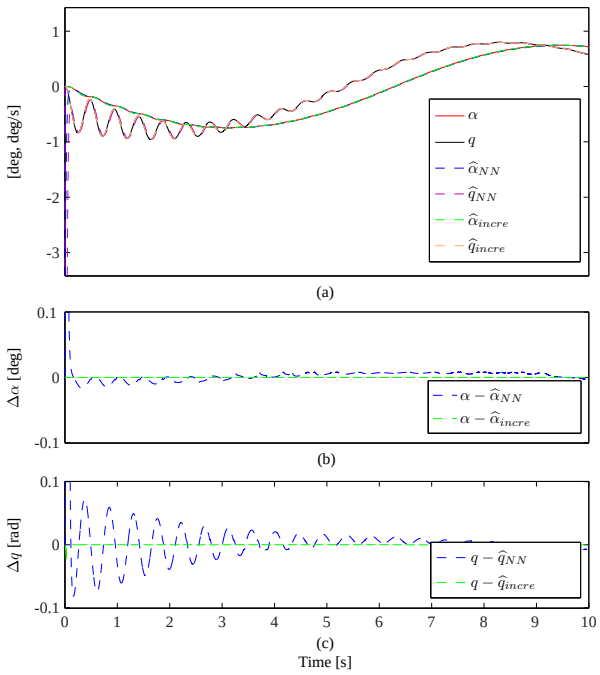

Figure 3: One-step prediction with on-line identified model using neural networks and incremental approach.

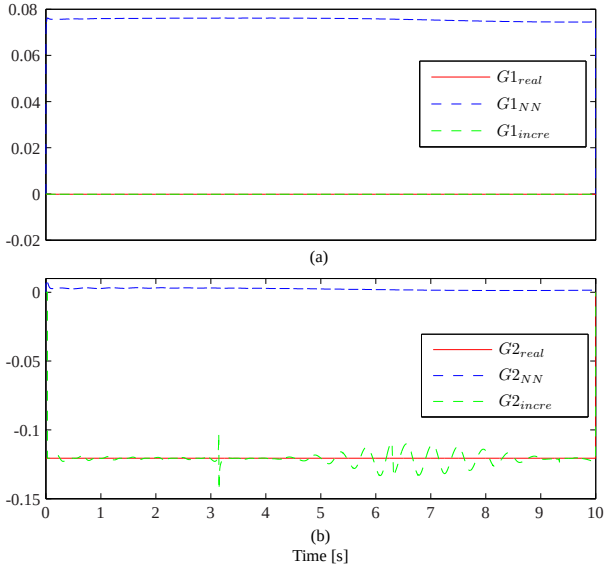

Figure 4: On-line identified control effective matrix using neural networks and incremental approach.

results of the control effective matrix, $G = [G1, G2]'$, using the two methods. It is apparent that the incremental method has a substantially improved identified control effective matrix, which is to approximate an important term $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$ in updating Actors.

Fig.5 illustrates the performance of the traditional HDP method and IHDP method when applied to an on-line tracking problem. Comparing to traditional HDP method, IHDP method can identify the local model and reject the disturbance much quicker at the beginning, and follow the reference signal more precisely. A slow on-line training of model neural network of traditional HDP may lead to a large overshoot and lost control at the initial stage. In realistic cases, traditional HDP needs an off-line training of the model before on-line training of the controller to prevent failures. On the other hand, IHDP does not need off-line learning because of a quick linearized local model identification at the beginning, which is very fast and accurate.

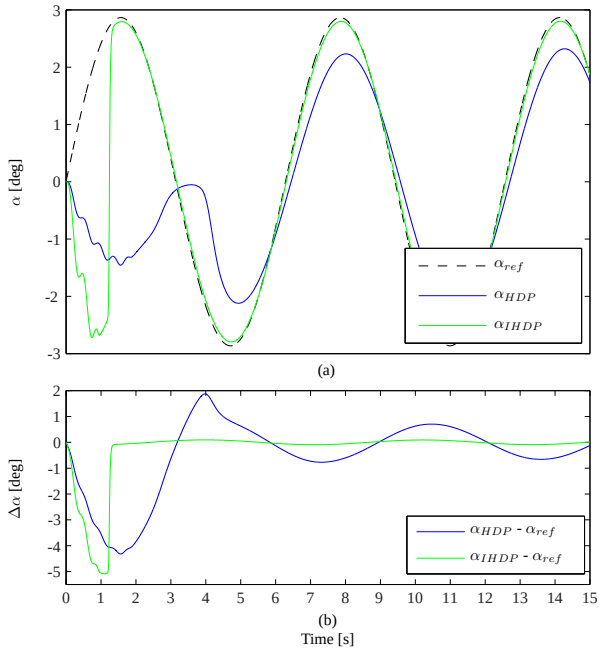

Figure 5: On-line tracking problem using neural networks and incremental approach.

5 Conclusion

This paper proposed a new approach, incremental model based heuristic dynamic programming, to design an adaptive flight control method without sufficient a-prior knowledge of the system dynamics. This method combines the advantages of HDP methods, which are adaptive and use a more general function approximator, and of incremental approaches, which do not need off-line learning and accelerate the on-line learning efficiently. The HDP method using a neural network to approximate the system dynamics and the IHDP method using incremental models are applied to a simple and illustrative application. By comparing the results, it is apparent that the presented IHDP method speeds up the on-line learning at the beginning and has a significantly higher precision than traditional HDP methods. To accelerate the on-line learning when a-prior knowledge is unknown or the system dynamics are changed suddenly is of great practical value.

As an extension of iADP method using a quadratic cost-to-go function, the IHDP method presented in this paper uses neural network functions with greater approximation ability and separates the policy evaluation and improvement with two approximators. This study generalized the use and applications of the iADP methods. Further investigation into different type of approximator and experimentation into more complex and realistic applications are strongly recommended.

REFERENCES

[1] Russell Enns and Jennie Si. Helicopter trimming and tracking control using direct neural dynamic programming. *Neural Networks, IEEE Transactions on*, 14(4):929–939, 2003.

[2] Silvia Ferrari and Robert F Stengel. Online adaptive critic flight control. *Journal of Guidance, Control, and Dynamics*, 27(5):777–786, 2004.

[3] Thomas Hanselmann, Lyle Noakes, and Anthony Zaknich. Continuous-time adaptive critics. *Neural Networks, IEEE Transactions on*, 18(3):631–647, 2007.

[4] Vivek Yadav, Radhakant Padhi, and SN Balakrishnan. Robust/optimal temperature profile control of a high-speed aerospace vehicle using neural networks. *Neural Networks, IEEE Transactions on*, 18(4):1115–1128, 2007.

[5] Fei-Yue Wang, Huaguang Zhang, and Derong Liu. Adaptive dynamic programming: an introduction. *Computational Intelligence Magazine, IEEE*, 4(2):39–47, 2009.

[6] Said G Khan, Guido Herrmann, Frank L Lewis, Tony Pipe, and Chris Melhuish. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, 36(1):42–59, 2012.

[7] Jennie Si. *Handbook of learning and approximate dynamic programming*, volume 2. John Wiley & Sons, 2004.

[8] E. van Kampen, Q. P. Chu, and J. A. Mulder. Continuous adaptive critic flight control aided with approximated plant dynamics. In *Proc AIAA Guidance Navig Control Conf*, volume 5, pages 2989–3016, 2006.

[9] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall International, Inc., 1999.

[10] P. Simplício, M. D. Pavel, E. van Kampen, and Q. P. Chu. An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion. *Control Engineering Practice*, 21(8):1065–1077, 2013.

[11] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. *Journal of guidance, control, and dynamics*, 33(6):1732–1742, 2010.

[12] P. J. ACQUATELLA, E. van Kampen, and Qi Ping Chu. Incremental backstepping for robust nonlienar flight control. In *Proceedings of the EuroGNC 2013*, 2013.

[13] Y. Zhou, E. van Kampen, and Qi Ping Chu. Incremental approximate dynamic programming for nonlinear flight control design. In *Proceedings of the EuroGNC 2015*, 2015.

[14] Y. Zhou, E. van Kampen, and Qi Ping Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. In *AIAA Guidance, Navigation and Control Conference*, 2016.

[15] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[16] Lars Sonneveldt. *Adaptive backstepping flight control for modern fighter aircraft*. TU Delft, Delft University of Technology, 2010.

[17] Seung-Hwan Kim, Yoon-Sik Kim, and Chanho Song. A robust adaptive nonlinear control approach to missile autopilot design. *Control engineering practice*, 12(2):149–154, 2004.