AIAA SciTech Forum
12-16 January 2026, Orlando, FL
AIAA SCITECH 2026 Forum

10.2514/6.2026-1585

A Multi-step and Eligibility Trace Approach to Incremental Dual Heuristic Programming for Flight Control

Wing Chan *, Erik-Jan van Kampen †
Delft University of Technology, P.O. Box 5058, 2600GB Delft, The Netherlands

Incremental Dual Heuristic Programming (IDHP) is a successor to the Dual Heuristic Programming (DHP) algorithm that uses an online identified incremental system model, this algorithm showed promising online learning and fault tolerance in simulated flights. This paper studies the potential for extending IDHP through augmenting the computation of agent updates and returns, more specifically, by using eligibility trace updates and multi-step temporal difference error. This results in the IDHP(λ), multi-step IDHP (MIDHP), and MIDHP(λ) variants, which are compared against IDHP in simulated flight scenarios with faults introduced mid-flight. The results demonstrate that flight controllers derived from the proposed variants have improved reference tracking & fault tolerance over the baseline IDHP, with the most improvement observed in MIDHP(λ).

Nomenclature

λ	=	Eligibility trace decay rate	[-]	k	=	Learning rate factor	[-]
$\mathcal{S}, \mathcal{A}$	=	MDP state & action space	[-]	$\mathbf{E}$	=	Eligibility trace	[-]
$\mathcal{P}()$	=	MDP transition function	[-]	L_{CAPS}	=	CAPS score	[-]
n, m	=	No. of MDP states, actions	[-]	λ_T, λ_S	=	CAPS temporal, spatial smoothness weight	[-]
s, s', a, r	=	MDP state, augmented state, action, reward	[-]	$\mathcal{N}$	=	Multivariate normal distribution	[-]
γ	=	MDP reward discount rate	[-]	p, q, r	=	Roll, pitch, yaw rate	[rad/s]
R_t	=	MDP Return	[-]	V_{TAS}	=	True airspeed	[m/s]
$J(), \pi()$	=	Value, policy function	[-]	α, β	=	Angle of attack, sideslip	[rad]
Λ	=	Value gradient function	[-]	ϕ, θ, ψ	=	Roll, pitch, yaw angle	[rad]
Λ'	=	Target value gradient function	[-]	H	=	Height	[m]
W_a, W_c	=	Actor & critic weights	[-]	X_e, Y_e	=	East, north-ward location	[m]
δ	=	Temporal difference error	[-]	θ_r	=	Reference pitch	[rad]
E_t	=	Quadratic δ	[-]	θ_e	=	Pitch error	[rad]
η_a, η_c	=	Actor, critic learning rate	[-]	$\delta_a, \delta_e, \delta_r$	=	Aileron, elevator, rudder deflection angle	[rad]
τ	=	Target critic update fraction	[-]	δ'_e	=	Elevator deflection from trim	[rad]
F, G	=	Incremental state transition, input matrix	[-]	tr_a, tr_e, tr_r	=	Aileron, elevator, rudder trim tab angle	[rad]
Θ	=	RLS parameter estimates	[-]	δ_f	=	Flaps deflection angle	[rad]
Σ	=	RLS estimate covariance	[-]	T_1, T_2	=	Left, right engine thrust setting	[-]
ρ	=	RLS forgetting factor	[-]	Sm	=	Action smoothness	[-]

*MSc., Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.
†Associate Professor, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.

I. Introduction

The aviation industry is undergoing many developments which will redefine what flight means, be it the advent of personal air vehicles, novel high performance aircraft designs, hydrogen fuelled concepts, or the increasing usage of drones [1–4]. Flight control on such novel systems, using model-based control system design methods, introduces a key conundrum: extensive dynamic models and therefore system identification campaigns will proof unavoidable, costing a vast amount of time and resources. An example is in the gain-scheduling approach to flight control system design [5]. Furthermore, since model-based controller synthesis techniques typically focus on flight control in nominal regions of the flight envelope, the occurrence of faults can result in a control system failure, as new dynamics are introduced or as the aircraft ventures into unmodelled regions of the flight envelope, thus risking loss of control [6]. Fault tolerant and adaptive flight control is a trend in flight control design which aims to address these issues, with promising techniques being actively developed [7, 8].

Simultaneously, Reinforcement Learning (RL) has been developing rapidly, from super human performance on games [9, 10], to autonomous cars and robots [11, 12]. RL is a method of Machine Learning (ML) that predicates on the machine learning through sovereign actions, as opposed to other ML methods such as supervised learning, where the machine is told what to do or what is correct. Actor-Critic Design (ACD) is an optimal control approach to the problem of RL [13]. In ACD, the RL agent comprises an actor, a critic, and a system model, the former two modeled using function approximators such as a neural network, and latter modelled typically as state derivatives. The actor and critic are updated through a form of policy improvement & policy evaluation respectively [14]. ACD algorithms are generally classified into three categories, all of which have identical architecture for the actor but not the critic: Heuristic Dynamic Programming (HDP), where the critic estimates the actors' value function; Dual Heuristic Programming (DHP), where the critic estimates the actors' value function gradient; and Global Dual Heuristic Programming (GDHP), whose critic estimates both simultaneously [15]. Incremental DHP (IDHP) is a successor to the DHP algorithm, which extends DHP with an online identified incremental model of the system dynamics, resulting in a sample efficient RL algorithm [16]. Parallel to ACD, there also exist Deep RL (DRL) algorithms where deep neural networks are used to model the agent.

The application of RL to flight control holds many promising potential developments. For instance, the learning emulation that RL methods offer has the potential of making flight controllers truly "intelligent", and offers an alternative reward-driven adaptive control method to methods such as Incremental Nonlinear Dynamic Inversion (INDI) or Incremental Back Stepping (IBS). Such a reward-driven adaptive control method can feasibly be directed to perform more than merely reference tracking, which traditonal controllers focus on, the reward could instead be formulated to incentivize a vast range of more abstract goals. Moreover, challenges in novel aircraft systems and fault tolerant control could be overcome with the successful application of RL based flight controllers. Specifically, mathematical models for novel aircrafts can be difficult to obtain, likewise for faulty aircrafts, which would not be an issue for model-free controllers such as RL. ACD algorithms are especially suited for this role due to their high adaptiveness & sample efficiency, allowing them to learn a stabilizing controller mid-flight [16]. Successfully applying RL to fault tolerant flight control has been demonstrated in attitude control of the Innovative Control Effectors (ICE) model with a pure ADP controller [17], an NDI hybrid ACD controller [18], as well as in attitude and velocity control using pure ACD [19].

ACD traditionally use information from one timestep per update. However, by using information from more time steps, it's possible to speed up agent learning. Learning with information from previous timesteps can be done in two ways: with eligibility traces, where past function parameter updates are recorded and subsequently reused [20]; or with multi-step policy evaluations, where rewards from multiple time steps are aggregated [21]. These ideas have been applied to the HDP algorithm which yielded improvements to the agents' learning rates [20–22].

Interestingly, such techniques have not yet been applied to IDHP, making the effects of such a combination unknown. Therefore, this papers' main contribution is in the extension of IDHP using eligibility trace updates, resulting in IDHP(λ), using the multi-step policy evaluations or Temporal Difference (TD) error, resulting in MIDHP, and a combination of the two, resulting in MIDHP(λ). These variants are evaluated both during nominal flight and with faults introduced to give an insight into these augmentations' effects.

This paper is outlined as follows: in Sec. II the background on IDHP, the application of IDHP to flight control, and the idea of multi-step updates and eligibility traces are presented; in Sec. III the proposed methodology for incorporating multi-step updates and eligibility traces to IDHP and the flight control task devised for testing are presented. The main

results are presented and discussed in Sec. IV, and finally the main conclusions are drawn in Sec. V.

II. Background

This section presents the background underlying the methodology of this paper. Beginning with introducing the sequential decision making framework used in all RL methods, the IDHP algorithm, the field of ACD, and finally eligibility traces and multi-step augmentations which exist in the TD learning framework.

A. MDP

Markov Decision Process (MDP) is the mathematical framework used in RL to model the sequential decision making process of some agent interacting with an environment. An MDP is described by the state space $\mathcal{S} \subset \mathbb{R}^n$, the action space $\mathcal{A} \subset \mathbb{R}^m$, a reward signal $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a state transition function $\mathcal{P}(s_{t+1}, r_{t+1}|s_t, a_t)$ which may deterministically or stochastically transition the state from s_t to s_{t+1} with reward r_{t+1} given an agent action a_t.

An MDP operates in discrete time. At each time step, an action is performed by the agent on the environment, which the environment responds to by producing the next time steps' state and reward, which are in turn observed by the agent to be processed into the next steps' actions. This notion of sequential decision making has important ramifications, it means that decisions made at an earlier time influence the current state, and therefore influencing what decisions should be made at a later time. In practice, this notion implies that taking suboptimal actions at an earlier time may lead to fewer rewards in the future. Building upon the MDP framework, it is possible to define a *Value function* $J(s)$, which describes the total future reward expected for starting in state s and acting according to the agents' current policy. One of the main goals of RL is to estimate such a function, which can be done by minimizing TD error, defined for every state transition as the difference between the agents' estimate of $J(s_t)$, and the sum of discounted $J(s_{t+1})$ for the state that the agent transitioned into s_{t+1} & the observed reward at s_t.

B. ACD and IDHP

RL agents seek to maximize the total expected reward in a given MDP, that is to not only obtain the maximum reward available immediately, but also the maximum cumulative reward of future states: the *return*. There are several methods of creating such an agent, among which lie the ACD algorithms: the focus of this paper. ACD algorithms are composed of three components; the actor, which observes the state of the MDP and chooses an action it thinks is most optimal; the critic, which observes the state of the MDP and outputs either the value function, value functions' gradient, or a combination of the two, all capturing information regarding the expected rewards from the current state; and the model, which models the dynamics of the MDP states. Function approximators are used to approximate the true functions which underly each of these components, for example, the true value function. Common options for such approximators are linear functions such as polynomials, radial basis functions, and neural networks.

In traditional ACD's, either an offline phase for learning the MDP dynamics, or providing the agents with the dynamics a-priori are required [19, 23, 24]. This means it is not possible for ACD agents to initialize entirely from scratch. Therefore, for pure online learning, there is a need to remove the burden of knowing system dynamics a-priori. In answer to this, an incremental model identified using Recursive Least Squares (RLS) was proposed, resulting in IDHP [16], which bridges this gap. Such a model came at the cost of only being able to model local dynamics as linear, as opposed to identifying potentially nonlinear dynamics across the entire state space, which placed new constraints on a high enough model update rate to overcome nonlinearities in system dynamics. However, a linear model also has the added benefit of significantly reducing the complexity of model identification, as a linear least squares approach can be adopted.

The IDHP actor, also known as the policy, is a function denoted as $\pi : \mathcal{S} \to \mathcal{A}$. To act, the agent evaluates the policy $\pi()$ and obtains the appropriate a, which is then performed by the agent on the environment. Any function approximator with parameters W_a may be used to represent $\pi()$. To train the agent to maximize future expected reward, the parameters W_a are updated to maximize the return $R_t = r_t + \gamma J(s_t)$ which is an estimate of the total reward or return to be expected in the future, with $J()$ being the expected return or value function of s_t, discounted by $\gamma \in [0, 1]$. In ACD, updating W_a to maximize R is done using gradient ascent with the following equations:

$$W_{a,t} = W_{a,t-1} + \eta_a \frac{\partial R_{t-1}}{\partial W_a} \tag{1}$$

$$\frac{\partial R_t}{\partial W_a} = \left(\frac{\partial r_t}{\partial s_t} + \Lambda_t \right) \frac{\partial s_t}{\partial a_{t-1}} \frac{\partial a_{t-1}}{\partial W_a} \tag{2}$$

$$\Lambda_t = \frac{\partial J(s_t)}{\partial s_t} \tag{3}$$

Where η_a is the gradient descent step size, also referred to as the actors' learning rate.

The IDHP critic is a value gradient function denoted as $\Lambda : \mathcal{S} \rightarrow \mathbb{R}^n$, it outputs the gradient of the value function with respect to the MDP states s as expressed in Eq. 3. This value gradient function is internal to the agent and serves only to aid in improving the actors' actions. Similar to the actor, any function approximator with parameters W_c may be used to represent Λ. To stabilize learning, an optional target critic Λ' is used. This Λ' is identical in everyway to Λ except for the function parameter values, which are updated with a moving average filter towards the latest Λ weights [25], see Eq. 7. The objective of the critic is to accurately estimate the value functions' gradient. This is done by updating the critics' parameters to minimize E_t, a quadratic of the TD error δ_t, where $E_t = \frac{1}{2}\delta_t \cdot \delta_t$ with $\cdot$ being the dot product and δ_t defined as follows:

$$\delta_t = \Lambda_{t-1} - \frac{\partial r_{t-1}}{\partial s_{t-1}} - \gamma \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}} \tag{4}$$

Note that the target critic output is used in constructing δ_t, which helps smoothen the variation of δ_t over time, as the target critic is updated slowly. In ACD, W_c is updated to minimize E through gradient descent using Eq. 5.

$$W_{c,t} = W_{c,t-1} - \eta_c \frac{\partial E_{t-1}}{\partial W_c} \tag{5}$$

$$\begin{aligned} \frac{\partial E_t}{\partial W_c} &= \frac{\partial E_t}{\partial \delta_t} \frac{\partial \delta_t}{\partial \Lambda_{t-1}} \frac{\partial \Lambda_{t-1}}{\partial W_c} \\ &= \delta_t \frac{\partial \Lambda_{t-1}}{\partial W_c} \end{aligned} \tag{6}$$

$$W_{c',t+1} = \tau W_{c,t} + (1-\tau) W_{c',t} \tag{7}$$

The system model of IDHP is then used to identify the $\frac{\partial s_{t+1}}{\partial a_t}$ term from Eq. 2 and the $\frac{\partial s_t}{\partial s_{t-1}}$ term from Eq. 4. This model is also internal to the agent, it is used to improve the actions of the actor and to improve the value estimation of the critic. Treating the system in discrete time, an incremental discrete model of the states can be written as $\Delta s_{t+1} = F_t \Delta s_t + G_t \Delta a_t$. This results in the following definition for the two partial derivatives

$$\begin{aligned} \frac{\partial s_{t+1}}{\partial a_t} &\approx G_t \\ \frac{\partial s_{t+1}}{\partial s_t} &\approx F_t + G_t \frac{\partial a_t}{\partial s_t} \end{aligned} \tag{8}$$

A linear model can be constructed by first defining the state increment $\delta s_{t+1} = s_{t+1} - s_t$ and action increment $\delta a_{t+1} = a_{t+1} - a_t$, then defining the linear model parameter matrix $\Theta_t = \begin{bmatrix} F_t & G_t \end{bmatrix}^\top \in \mathbb{R}^{(n+m)\times n}$ and variable vector $X_t = \begin{bmatrix} \delta s_{t+1} & \delta a_{t+1} \end{bmatrix} \in \mathbb{R}^{n+m}$. With the RLS algorithm, this linear model Θ_t can then be identified online [16].

Combining the actor, critic, and RLS model, the IDHP algorithm can then be summarized in the diagram presented in Fig. 1.

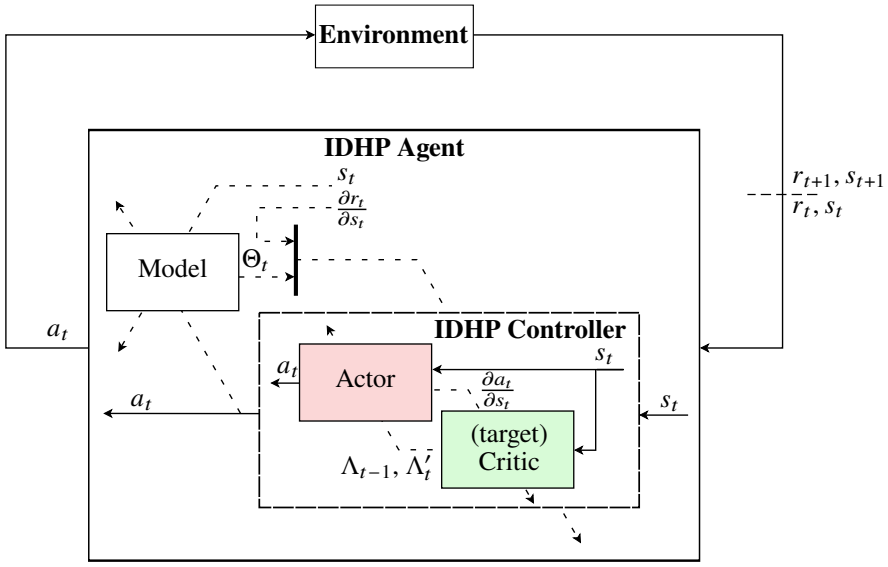


Fig. 1 MDP flow diagram representing the signal flows internal to the IDHP agent, dashed signals represent variables used to update the blocks which they cross.

C. Multi-step Updates and Eligibility Traces

The TD error computes the error between the value which the critic estimated and the value which is observed, this observed value can also be referred to as the TD target. In the case of Eq. 4, the TD target is composed of the terms $\frac{\partial r_{t-1}}{\partial s_{t-1}} + \gamma \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}}$. It can be seen that the TD target is composed of an observed reward signal and an estimate made by the target critic. The idea of multi-step TD error is to use more observed reward signals to construct a more accurate TD target [26, 27].

For the IDHPs' TD error, a multi-step version is expressed as follows:

$$\delta_{n,t} = \Lambda_{t-n} - \frac{\partial(\gamma^n J'_t + \sum_{m=0}^{n-1} \gamma^m r_{t-n+m})}{\partial s_{t-n}} \tag{9}$$

Instead of using the reward at one single time step to compute the TD error, it is possible to extend the error to include rewards from multiple time steps. This results in a new gradient descent term for the critic parameters update equation:

$$\frac{\partial E_t}{\partial W_c} = \delta_t \frac{\partial \Lambda_{t-n}}{\partial W_c} \tag{10}$$

An alternative and more general method for incorporating additional information to agent updates is the eligibility trace $\mathbf{E}$, which keeps track of how eligible each parameter for updating. This is done by storing previous function gradients and reusing them in the future at decaying magnitudes [28]. The function gradient in question is the derivative of the function output with respect to each function parameter, in the actors' case $\mathbf{E}$ would accumulate the $\frac{\partial a_t}{\partial W_a}$ term from Eq. 2, and in the critics' case $\mathbf{E}$ would accumulate $\frac{\partial \Lambda_t}{\partial W_c}$ from Eq. 6. In implementation, the eligibility trace results in a similar algorithm as momentum gradient descent, which accumulates past function updates and also reuses them in the future at decaying magnitude. The difference between these two methods is in the variable being accumulated. Whereas momentum gradient descent accumulates the total parameter update, eligibility traces only accumulate the functions' gradient.

To incorporate eligibility traces, the parameter update equations are changed to the form shown in the following equations:

$$W_t = W_{t-1} + \eta \frac{\partial M}{\partial O} \mathbf{E}_{t-1} \tag{11}$$

$$\mathbf{E}_t = \lambda\gamma \mathbf{E}_{t-1} + \nabla W_{t-1}, \quad \mathbf{E}_0 = \mathbf{0} \tag{12}$$

Where ∇W is the gradient of the network output with respect to network weights, $\frac{\partial M}{\partial O}$ is the gradient of the relevant metric, M, with respect to network output, O. For the critic, this partial derivative would be δ_t and for the actor this would be $\left(\frac{\partial r_t}{\partial s_{t+1}} + \Lambda_{t+1}\right)$; $\lambda \in [0, 1)$ is the trace decay rate which controls how quickly do prior gradients decay in the eligibility trace, $\lambda = 0$ decays previous gradients completely and results in the original update rules, while $\lambda \to 1$ means previous gradients decay according to γ.

A multi-step policy evaluation has been successfully augmented to HDP yielding increased sample efficiency [21], an improvement further enhanced by an adaptive multi-step scheme [22]. An eligibility trace style update has also been applied successfully to ACD algorithms on several works [20, 29], such an augmentation similarly improved the agents' sample efficiency.

III. Methodology

Taking the idea of multi-step updates and eligibility traces from TD learning, it becomes possible to refine the method of updating the actor and the critic in the ACD algorithms. The methods for incorporating such ideas into the IDHP algorithm, and how the augmentations are to be evaluated, are outlined in the methodology presented in this section.

A. IDHP Augmentations

Multi-step updates and eligibility traces can be incorporated individually or together into IDHP, with multi-step TD applied to the critic and eligibility traces applied to either the critic or the actor. However, from preliminary empirical testing, the option of applying eligibility traces to the critic was shown to negatively impact the learning performance of IDHP. Thus the three options present are to: apply multi-step updates to the critic, apply eligibility traces to the actor, or the two options simultaneously.

For the critic, which is only augmented with the multi-step TD error, the gradient descent term of Eq. 6 used for updating the critic is reworked to include a multi-step TD error. While each algorithm or MDP being solved typically has a certain n which is optimal, the present paper only considers a 2-step TD for the sake of simplicity, which can be obtained by substituting $n = 2$ to Eq. 9, which yields $\delta_{2,t}$ expressed in Eq. 13. To incorporate the 2 step TD error into the critic update is to simply substitute δ_t with $\delta_{2,t}$ in Eq. 6.

$$\begin{aligned}
\delta_{2,t} &= \Lambda_{t-2} - \frac{\partial(\gamma^2 J'_t + \sum_{m=0}^{1} \gamma^m r_{t-2+m})}{\partial s_{t-2}} \\
&= \Lambda_{t-2} - \frac{\partial(\gamma^2 J'_t + r_{t-2} + \gamma r_{t-1})}{\partial s_{t-2}} \\
&= \Lambda_{t-2} - \gamma^2 \Lambda'_t \frac{\partial s_t}{\partial s_{t-1}}\bigg|_t \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1} - \frac{\partial r_{t-2}}{\partial s_{t-2}} - \gamma \frac{\partial r_{t-1}}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1}
\end{aligned} \tag{13}$$

Regarding the actor, the primary augmentation is in incorporating eligibility traces. To do so, the gradient descent term $\frac{\partial a_{t-1}}{\partial W_a}$ of Eq. 2 is replaced by $\mathbf{E}$, which is updated according to Eq. 14. The actors' update equation are also experimentally augmented with a Conditioning for Action Policy Smoothness (CAPS) parameter [30], which has been observed to mitigate noisey actions in RL agents. This is done by adding L_{CAPS}, as defined in Eq. 15, to $\frac{\partial R_t}{\partial W_a}$. This term penalizes large variations in the actor policy and large temporal changes in the actors' actions, the weights of these penalties can be adjusted through the factors λ_S and λ_T respectively. Combining these two augmentations results in a new expression shown in Eq. 16 for the gradient descent term $\frac{\partial R_t}{\partial W_a}$ used in Eq. 1.

$$\mathbf{E}_t = \lambda\gamma\mathbf{E}_{t-1} + \frac{\partial a_t}{\partial W_a} \tag{14}$$

$$L_{CAPS} = \lambda_T \|\pi(s_t) - \pi(\tilde{s})\|_2 + \lambda_S \|\pi(s_t) - \pi(s_{t-1})\|_2 \,, \quad \tilde{s} \sim \mathcal{N}(s_t, diag(\sigma)) \,, \quad \sigma \in \mathcal{R}^n \tag{15}$$

$$\frac{\partial R_t}{\partial W_a} = \left(\left[\frac{\partial r_t}{\partial s_t} + \Lambda_t \right] G_{t-1} - L_{CAPS} \right) \mathbf{E}_t \tag{16}$$

Ultimately, this results in 3 variants of the IDHP algorithms. The first is IDHP(λ), which has the actor augmented with an eligibility trace update; the second is MIDHP, which has the critic augmented with a multi-step update; the third is MIDHP(λ), which has both the two augmentations simultaneously.

B. Aircraft Pitch Control as an MDP

The four algorithms are evaluated in their ability to control the pitching motion of an aircraft in a flight control task. This is done in simulation using CitAST, a high fidelity nonlinear 6 degrees of freedom dynamics model of a fixed-wing business jet running at 100 Hz or $dt = 0.01$ s is used [31]. The aircraft is modelled with engine and control surface dynamics, such as inertia in deflecting the control surfaces. Airspeed control is delegated to an auto-throttle controlling engine thrust settings. The aircraft model has been validated by flight test data onboard a Cessna Citation II research aircraft [32]. During the experiments conducted in this paper, the aircraft models' state and inputs are trimmed for an airspeed of 90 m/s at an altitude of 2000 m.

The pitch control task needs to first be framed in terms of an MDP before the IDHP algorithm is applied as a flight controller. The control scheme which the MDP will emulate is a simple feedback control scheme, which can be represented in the form of Figure 2.

The dynamics model of the aircraft already fulfils the role of the state transition function $\mathcal{P}(s_{t+1}, r_{t+1}|s_t, a_t)$, this model is the plant used in Fig 2. Thus, the only remaining variables to fully define the MDP are s, a, and r. In addition to these variables, $\frac{\partial r}{\partial s}$ also needs to be defined for the IDHP algorithm,

The aircraft model used has a state vector x as shown in Eq. 17, using the inertial and body frames as shown in Fig 3. To create s, only a subset of x states will be used, resulting in a Partially Observable MDP (POMDP). Specifically, s only includes a subset of longitudinal aircraft states, see Eq. 18. In addition, to make the control task more explicit to the agent, the actor and critic functions are provided with the pitching error θ_e, equal to the aircrafts' pitch θ minus the refernce pitch θ_r. This addition requires a slight abuse of notation where the actor and critic inputs will be the augmented MDP state s' Eq. 19, instead of only s.

$$x = \begin{bmatrix} p & q & r & V & \alpha & \beta & \theta & \phi & \psi & H & X_e & Y_e \end{bmatrix}^\top \tag{17}$$

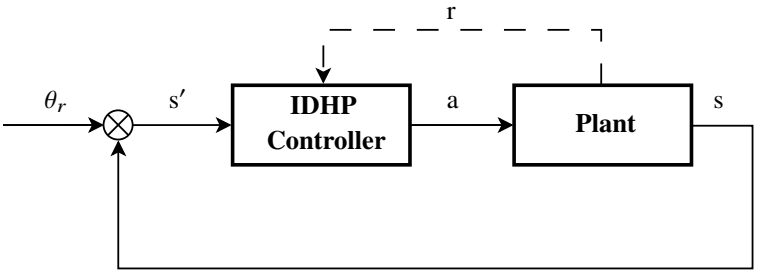


Fig. 2 Control diagram of the pitch control MDP.

$$s = \begin{bmatrix} \alpha & \theta & q \end{bmatrix}^\top \qquad s' = \begin{bmatrix} \alpha & \theta & q & \theta_e \end{bmatrix}^\top \tag{18, 19}$$

The full set of actuators modelled u is shown in Eq. 20. Just like for s, not all of the modelled variables will be used

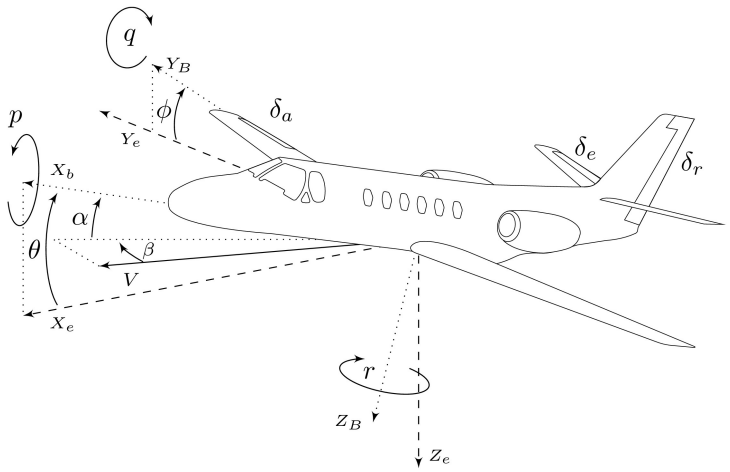


Fig. 3 Reference frames used for the definition of the various states of the Cessna-Citation II model, adopted from Seres [33].

in creating a. The agents in this control task will only be in control over the elevator, specifically the elevator deflection from the trim angle δ'_e, see Eq. 21. In addition, a symmetric deflection limit close to the realistic saturation limits of the PH-LAB is set on the elevator δ_e.

$$u = \begin{bmatrix} \delta_e & \delta_a & \delta_r & \text{tr}_e & \text{tr}_a & \text{tr}_r & \delta_f & \text{gear} & T_1 & T_2 \end{bmatrix}^\top \tag{20}$$

$$a = \begin{bmatrix} \delta'_e \end{bmatrix}, \quad -15\frac{\pi}{180} < \delta'_e < 15\frac{\pi}{180} \text{ [rad]} \tag{21}$$

The reward r is defined as the negative square of the pitch tracking error, as shown in Eq. 22, and its' derivative $\frac{\partial r}{\partial s}$ shown in the following:

$$r = -(\theta - \theta_r)^2 \qquad \frac{\partial r}{\partial s} = \begin{bmatrix} 0 & -2(\theta - \theta_r) & 0 \end{bmatrix} \tag{22, 23}$$

At initialization, the states and inputs of the dynamical model are set to trim.

C. IDHP Algorithm Configuration

The constants and variables which need to be initialized for the IDHP algorithm, including the dynamic model state and inputs of the MDP, are tabulated in Tab. 1. The version of IDHP implemented uses varying learning rates and eligibility trace decay rates, starting at higher values to aid network weights convergence, and then decreasing to a lower value after 4 s. These high and low values are denoted by an h and l in the subscript respectively: $\eta_{\cdot,h}$, $\eta_{\cdot,l}$, λ_h, and λ_l. Starting from 4 s, the rates smoothly transition from higher to lower values, this is done by multiplying with the factor k_t defined at every t as follows:

$$k_t = c + (1 - c)\frac{v_l}{v_t}, \quad c = 0.998 \tag{24}$$

Where v_t is the rates' value at t, and v_l is the lower value of the rate. The constant c changes how rapidly the rate drops, and is set to be 0.998 for the present study. Multiplying v_t by the factor k_t brings v_t asymptotically towards v_l, that is to say $\lim_{t\to\infty} v_h \prod_{i=0}^{t} k_i = v_l$.

For the function approximators of the actor and the critic, a single hidden layer neural network with 10 nodes in the hidden layer is used. This network size is a compromise struck between learning complexity and agent performance, as

it was observed that this size is just before agent performance degrades, but not too big that the number of parameters to be optimized slows down learning. In both networks, there are no biases in any of the layers, the input layer takes s' and has four nodes, and the hidden layer uses a tanh activation function. The output layer of the two networks are different. For the actor, its output is elevator deflection from trim δ_e' and so only a single node in the output layer is required, a tanh activation function is also used in this layer with the upper and lower asymptotes being $15\pi/180$, the limits stated in Eq. 21. For the critic, its output layer estimates value function gradients over s and thus has 3 nodes, a simple linear activation function is used.

The IDHP algorithm is created by combining the three modules and defining the flow of variables from one module to another. This is depicted graphically in Figure 1, in addition to the r and s signals shown, the inputs of the IDHP agent are augmented by θ_r to allow for the construction of s', while the output is a. IDHP uses the inputs for two categories of computation, the first of which is to compute the control action, which is done simply by the actor, and the second category is to compute all the necessary internal updates, done by the critic and the RLS model. The order of update computations is clarified further in Alg. 1.

Table 1 IDHP initialization variables and hyperparameters, η_a, η_c, and λ values are shown in Tab 2.

Hyperparameters		Variable initialization			
Actor learning rates:	$\eta_{a,h}$,$\eta_{a,l}$	Eligibility trace:	$\mathbf{E}_0$	=	$\mathbf{0}$
Critic learning rates:	$\eta_{c,h}$,$\eta_{c,l}$	RLS model parameter:	Θ_0	=	$\mathbf{0}$
Eligibility trace decay rates:	λ_h,λ_l	RLS model covariance:	Σ_0	=	$10^6 \cdot \mathbb{I}$
MDP reward discount factor:	$\gamma = 0.6$	Actor-network weights:	$W_{a,0}$	$\sim$	$\mathcal{N}(\mathbf{0}, 0.1^2\mathbb{I})$
RLS forgetting factor:	$\rho = 1$	Critic-network weights:	$W_{c,0}$	$\sim$	$\mathcal{N}(\mathbf{0}, 0.1^2\mathbb{I})$
Target critic mixing factor:	$\tau = 0.02$				
CAPS weights:	$\lambda_T, \lambda_S = 0.012, 0.001$				

Implementation of the ∇W term in Eq. 12 differs based on the actor and critic function form used. In the case of the present IDHP, where eligibility traces are used only on the actor, whose function is a neural network, ∇W represents the derivative of the actor network with respect to its weights. Such a derivative is called the Jacobian matrix of the actor network, where the rows are the outputs of the network, and columns are the derivatives of the outputs with respect to each weight. For the actor, the Jacobian matrix is a 1 by 40 matrix, since in total the actor network has 1 output and $3 \cdot 10 + 10 \cdot 1$ weights, therefore $\nabla W = \frac{\partial \pi(s)}{\partial W_a} \in \mathcal{R}^{1\times 40}$.

Since the augmented IDHP algorithms are either algorithmically different from the baseline, have new hyperparameters, or both, each algorithms' hyperparameters are tuned individually; doing so yields us the best performance of each variant. These parameters are reported in Tab. 2.

Table 2 Hyperparameters used for each of the four algorithms.

		IDHP	IDHP(λ)	MIDHP	MIDHP(λ)
$\eta_{a,h}, \eta_{a,l}$	:	40, 10	25, 7.5	43, 6.5	30, 5.0
$\eta_{c,h}, \eta_{c,l}$	:	0.5, 0.25	0.5, 0.25	0.9, 0.05	0.7, 0.1
λ_h, λ_l	:	N/A	0.99, 0.8	N/A	0.99, 0.4

D. Experiment Setup

A simulated flight manoeuvre lasting 90 s is used to evaluate the performance of the various algorithms. During which, several aircraft faults will be introduced suddenly to evaluate each agents' fault tolerance. As the agents are initialized randomly, the first 55 s is designed to be a *warmup* phase, so as to expose agents to the aircrafts' dynamics. Thus, the reference signal is designed similarly to system identification manoeuvres. The remainder of the flight is

Algorithm 1 IDHP algorithm.

Initialize:
Set initial variable values and hyperparameters listed in Tab. 1.
Online loop:
For $t = 0$ **to** T/dt **do**

$a_t \leftarrow \pi(s'_t; W_{a,t})$ ▷ sample action from policy

$u_t = u_0 + a_t$ ▷ add commanded action to trim input

$s_{t+1}, r_{t+1}, \theta_{r,t+1} \leftarrow \text{Environment}(u_t)$ ▷ perform action to propagate dynamics

$$\eta_{a,t}, \eta_{c,t}, \lambda_t \leftarrow \begin{cases} \eta_{\cdot,h}, \lambda_h & \text{if} \quad t < 4s \\ \eta_{\cdot,t-1} k_t, \lambda_{t-1} k_t & \text{otherwise} \end{cases}$$ ▷ $k_t \leftarrow$ Eq. 24

If using multi-step IDHP **and** $t > 2$ **then**

$$\delta_t = \Lambda(s'_{t-2}) - \frac{\partial r_{t-2}}{\partial s_{t-2}} - \gamma \frac{\partial r_{t-1}}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1} - \gamma^2 \Lambda'(s'_t) \frac{\partial s_t}{\partial s_{t-1}}\bigg|_t \frac{\partial s_{t-1}}{\partial s_{t-2}}\bigg|_{t-1}$$ ▷ Eq. 13, $\frac{\partial s_t}{\partial s_{t-1}} \leftarrow$ Eq. 8

Else

$$\delta_t = \Lambda(s'_{t-1}) - \frac{\partial r_{t-1}}{\partial s_{t-1}} - \gamma \Lambda'(s'_t) \frac{\partial s_t}{\partial s_{t-1}}\bigg|_t$$ ▷ Eq. 4, $\frac{\partial s_t}{\partial s_{t-1}} \leftarrow$ Eq. 8

End If

$\frac{\partial E_t}{\partial W_c} = \delta_t \frac{\partial \Lambda_t}{\partial W_c}$ ▷ Eq. 6

$W_{c,t+1} = W_{c,t} - \eta_{c,t} \frac{\partial E_t}{\partial W_c}$ ▷ Eq. 5

$W_{c',t+1} = \tau W_{c,t} + (1 - \tau) W_{c',t}$

$\mathbf{E}_{t+1} = \lambda_t \gamma \mathbf{E}_t + \frac{\partial \pi(s'_t)}{\partial W_a}$ ▷ Eq. 14, $\lambda = 0$ if not using eligibility traces

$\frac{\partial R_t}{\partial W_a} = \left(\left[\frac{\partial r_t}{\partial s_t} + \Lambda(s_t) \right] G_{t-1} - L_{CAPS} \right) \mathbf{E}_t$ ▷ Eq. 16

$W_{a,t+1} = W_{a,t} + \eta_{a,t} \frac{\partial R_t}{\partial W_a}$ ▷ Eq. 1

$F_{t+1}, G_{t+1} \leftarrow RLS$

referred to as the *manoeuvring* phase, where a more typical pitch-up pitch-down flight manoeuvre is used.

In summary, θ_r takes a sinusoidal form during warmup, a trapezoidal form during the pitch-up pitch-down manoeuvre, and for all other times remains equal to the θ_{trim} which is approximately 4 deg. This is summarized by Eq. 25.

$$\theta_{r,t} = \theta_{trim} + \frac{\pi}{180} \cdot \begin{cases} (10 \sin(\frac{2\pi t}{15}) + 8 \sin(\frac{2\pi t}{30}))(1 - \frac{t}{75}) & t < 45 \\ 1.5(t - 55) & 55 \leq t < 65 \\ 15 & 65 \leq t < 75 \\ -1.5(t - 85) & 75 \leq t < 85 \\ 0 & \text{otherwise} \end{cases} \quad \text{[rad]} \tag{25}$$

To test the agent for fault tolerance, four scenarios are designed. The first scenario serves as a baseline, and no fault is introduced to the aircraft during the flight. In the second scenario, the aircrafts' Centre of Gravity (CG) is suddenly shifted forward by 0.5 m 60 s into the flight. In the third, the elevator control effectiveness is reduced by 70% at 60 s to model elevator damage. The fourth scenario repeats the 70% effectiveness reduction in addition to lowering the saturation on δ'_e to ±5 deg at 60 s thus modelling a more severe elevator damage.

The full time trace of the reference signal, along with at which point a fault will be introduced, is shown in Figure 4.

Monte Carlo experiments are conducted to gather the agents' performances, with each agent repeating each of the scenarios 100 times. Upon each repetition, the agent is re-initialized according to Tab. 2, a new Random Number Generator (RNG) seed distinct from the previous repetition is used to initialize the network weights, the same 100 RNG seeds are used across the 4 scenarios for all agents. Based upon these experiments, five tests are compiled to compare the agents, an overview of which is tabulated in Tab. 3. Test one focuses on how the four variants performed during the warmup phase, while tests two to four study how the agents handle the nominal and faulty aircraft during the

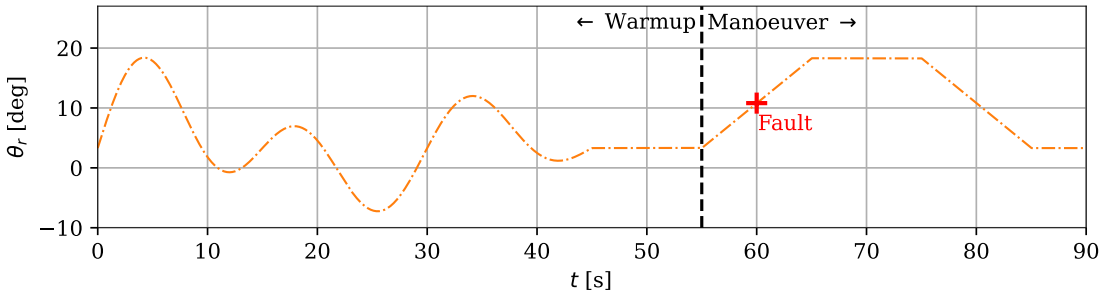


Fig. 4 Reference pitch signal θ_r, vertical black line at 55 s demarcates the end of the *warmup* phase and the start of the *manoeuvering* phase, red cross at 60 marks introduction of fault if any.

manoeuvering phase.

All tests use the same two metrics for evaluation, the first metric is $|E|$ Eq. 26, the absolute error in pitch tracking summed over time, measuring the controllers' tracking accuracy. The second metric is Sm, a smoothness metric, which measures controller action smoothness. Its' definition is taken from [30] and is written in Eq. 27.

$$|E| = \sum_{t=0}^{T/dt} \left|\theta_{e,t}\right| \qquad Sm = \frac{2}{n f_s} \sum_{i=1}^{n} M_i f_i \tag{26, 27}$$

Where M_i is the amplitude of the i-th frequency component f_i where $i \in [1, n]$, with n the number of frequency components sampled, and $f_s = 100$ being the sampling frequency in the time domain. Calculating Sm is done by first taking the Fourier transform of δ_e from the concerned flight phase, and then using the obtained spectrum in Eq. 27.

Table 3 The five tests used in evaluating the proposed augmentations on IDHP.

		Test 1	Test 2	Test 3	Test 4	Test 5
Phase	:	Warmup (0 to 55 s)	Manoeuvering (55 to 90 s)	Manoeuvering (55 to 90 s)	Manoeuvering (55 to 90 s)	Manoeuvering (55 to 90 s)
Faults	:	N/A	None	Shifted CG	Damaged Elevator	Damaged and saturated elevator
Metrics	:	$\|E\|$, Sm				

With the first test, the question of how the proposed augmentations affect the ability of IDHP to learn from tabular rasa may be answered. As the first test is concerned only with the performance of the controllers when learning from a randomly initialized state. The second to fifth tests will tell of the fault tolerance behaviours of the four controllers as well as how they may perform in flight once trained.

E. Statistical Interpretation of Results

The five tests in Tab. 3 will generate samples of metrics for the four algorithms. To systematically decide which algorithms' metrics were better than another, two statistical measures must be used to interpret these results: first is the common t test, and another measure to quantify the *substantive significance* of such observed differences, the A test [34]. The second test is more formally referred to as Vargha and Delaneys' (VD) A-test, which uses the A-value to ordinally ranks probability distributions. The A-value is bound to $[0, 1]$ and measures the relative difference between two sample groups, $A = 0.5$ means that there are no differences, < 0.5 means that the first sample is smaller than the second, and vice versa for > 0.5. The gathered metrics are cleaned of outliers before any statistical testing, which are used to answer two questions: whether an augmented algorithms' metrics are significantly different from IDHPs' metrics; and by what magnitude this augmented algorithms' metrics are different from IDHPs' metrics.

IV. Results and Discussion

IDHP and the three proposed variants of IDHP(λ), MIDHP, and MIDHP(λ) are evaluated through the Monte Carlo experiments designed in Sec. III.D, the results of which are presented & discussed in this section.

Before diving into the metrics, a summary of the 100 simulated flights flown by IDHP with no faults introduced is presented in Figure 5, to give an initial frame of reference for the results shown hereafter. To improve the interpretability of the time traces shown, runs with $|E| > 2000$ during the manoeuvring phase are omitted when plotting all time traces hereafter. For transparency's sake, the number of runs omitted are written in the caption.

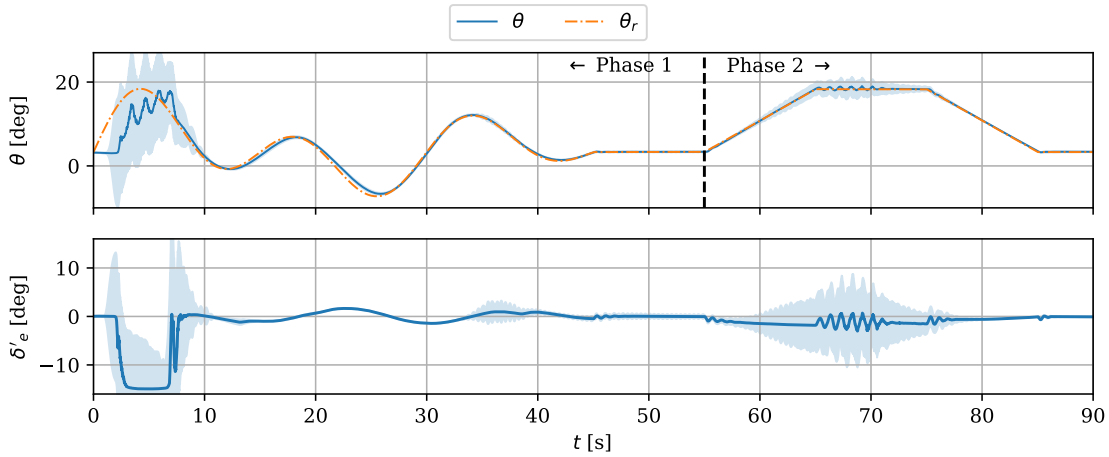


Fig. 5 Time trace of converged IDHP flights with no faults introduced to the aircraft, pitch & elevator deflection commanded by IDHP in the upper and lower subplot respectively, mean shown in solid blue line, 2σ bounds shown in light blue shaded region (No. $|E| > 2000 = 1$).

Figure 5 shows how the IDHP agent starts out not knowing how to control the aircraft, as is obvious by the lack of elevator deflection. However, over the first 10 s, the agents come to learn a stable controller that is capable of following the reference pitch signal with ease. The behaviour of this controller can be seen to stabilize over the remainder of the warmup phase, indicating that the actor-network weights have stabilized. Having learned a usable actor network in the warmup phase, the aircraft is then commanded to trim for a short time before commencing the pitch-up pitch-down manoeuvre.

The agents in these runs track the reference signal with good accuracy through most of the flight. However, at various phases during the flight, many controllers command a very oscillatory or noisy action; such controllers can be said to be marginally stable, to draw a parallel with linear control theory. For example around 65 s when the high-pitch hold begins, the elevator deflections start becoming very oscillatory, but dampens out towards the end of the high-pitch hold. Moreover, there are 3 runs that exhibit oscillatory actions towards the end of the warmup phase around 35 s.

These oscillations may have been mitigated if CAPS was added to the actor update equation in another manner. Specifically, instead of directly adding L_{CAPS} to $\frac{\partial R_t}{\partial W_a}$ as done in Equation 16, it should have been first been added to the return equation to yield $R_t = r_t + \gamma J(s_t) - L_{CAPS}$ and thereafter take the derivative of R_t w.r.t. W_a.

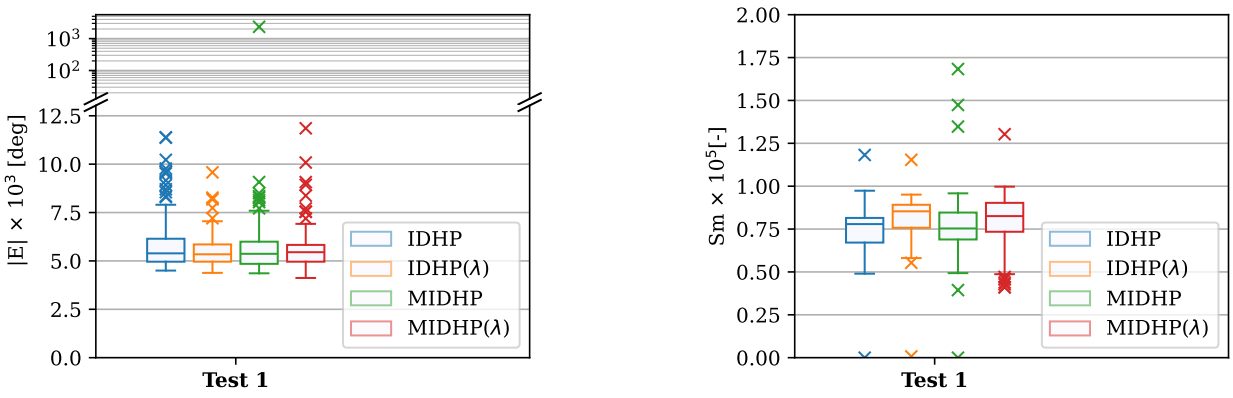
Taking a step back, the performances of the four variants over the five tests are presented in the following subsections using $|E|$ and Sm metrics from the Monte Carlo experiments, giving way for an empirical yet concrete comparison amongst the variants.

A. Warmup Phase

The four variants' $|E|$ and Sm metrics in Test 1, the test evaluating a controllers' warmup performance, are shown in Figure 6. The statistical test results on the outlier-free or cleaned metric samples are summarized and presented in Tab. 4. From the figure, it appears that the tracking error and smoothness of all controllers are comparable, with the Inter-Quartile Range (IQR) being at similar values of around 5-6 $\times 10^3$ deg for $|E|$ and 0.7-0.9 $\times 10^5$ for Sm.

The statistical testing results of Tab. 4 also tell a similar story. When it comes to $|E|$, there is no statistically significant difference between the four variants. However, a slightly smaller A-value is reported in all the comparisons between the augmented IDHP algorithms and the baseline IDHP, signifying a marginal albeit statistically insignificant improvement in tracking performance by the variants during the warmup phase.

The Sm metric statistical tests show that there are statistically significant differences between the smoothness of the augmented algorithms of IDHP(λ), and MIDHP(λ), versus the baseline IDHP. Specifically, both these two augmented algorithms demonstrated a less smooth control action during warmup, indicated by $A > 0.5$, meaning the Sm of these variants were higher than IDHP. For the MIDHP Sm, however, while the actions are less smooth than those of IDHP according to the A-value, this finding was not statistically significant.



(a) $|E|$. **(b)** Sm.

Fig. 6 Test 1 $|E|$ & Sm result boxplots.

Table 4 A-values on the first test, red A-value indicates statistical insignificance (p-value > 0.05).

	Test 1	
	$\|E\|$	Sm
IDHP(λ) vs IDHP :	0.478	0.756
MIDHP vs IDHP :	0.475	0.540
MIDHP(λ) vs IDHP :	0.471	0.653

B. Manoeuvering Phase

In tests 2 to 5, the control performance of the four algorithms on the nominal aircraft and one which has some fault introduced, is evaluated. Figure 7 presents the boxplots that capture the distribution of $|E|$ and Sm metrics, while Tab. 5 tabulates the corresponding statistical test results on the cleaned samples. The distribution of metrics in the manoeuvring phase was more diverse than in the warmup phase, according to Figure 7. Unexpectedly, it is interesting to note that for all algorithms the introduction of faults seems to have a smoothening effect on the control action, this can be seen in sub-figure 7b, where the Sm values are generally lower than that of sub-figure 6b. It is as if the change in aircraft dynamics pressured or allowed the agents to learn a smoother policy.

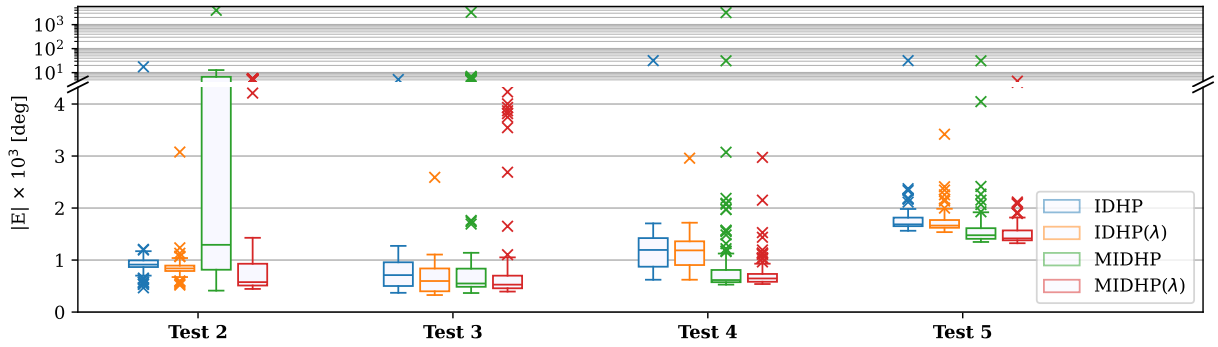
Table 5 *A*-values on the second to fifth tests, red *A*-value indicates statistical insignificance (p-value = 0.05).

(a) $|E|$.

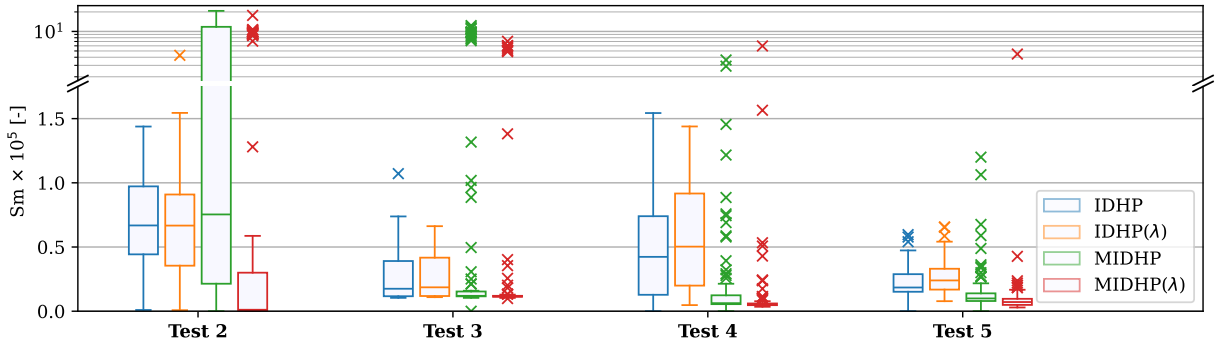
		Test 2	Test 3	Test 4	Test 5
IDHP(λ) vs IDHP	:	0.190	0.346	0.471	0.399
MIDHP vs IDHP	:	0.537	0.283	0.051	0.102
MIDHP(λ) vs IDHP	:	0.042	0.303	0.042	0.080

(b) Sm.

		Test 2	Test 3	Test 4	Test 5
IDHP(λ) vs IDHP	:	0.447	0.521	0.559	0.615
MIDHP vs IDHP	:	0.544	0.217	0.078	0.074
MIDHP(λ) vs IDHP	:	0.013	0.143	0.027	0.022



(a) $|E|$.



(b) Sm.

Fig. 7 Test 2, 3, 4, 5 $|E|$ & Sm result boxplots.

Generally, the Multi-step augmented algorithms have improved tracking performance and much-improved action smoothness. However, this is not consistently observed, as the MIDHP algorithm had a dramatically larger spread of $|E|$ and *Sm* during test 2: where no fault is introduced. From Figure 7, it can be seen that the MIDHP(λ) controller has the lowest median and general $|E|$ in all four tests, where 75 % of the runs had lower $|E|$ than approximately 75 % of the runs from IDHP for three out of the four tests. For example, the $|E|$ lower quartile for IDHP on test 4 was roughly 0.8×10^3 deg, while the $|E|$ upper quartile for MIDHP(λ) was roughly 0.7×10^3 deg. The smallest improvement in $|E|$ by MIDHP(λ) was found in test 3, which had a 30 % drop in median $|E|$. On the matter of action smoothness,

MIDHP(λ) also consistently had some of the lowest overall Sm of the four algorithms. However, there were still a handful of runs from other algorithms which achieved lower Sm than any MIDHP(λ) runs.

Aside from analysis of the metrics, the timetraces of all IDHP and MIDHP(λ) flights during the manoeuvering segment with faults are also presented in Figure 8, to allow the reader to qualitatively compare the performance of MIDHP(λ) against IDHP. Comparing the flights of IDHP to MIDHP(λ), the contrasts evident from the boxplots can be re-observed in the time traces. First, the action of MIDHP(λ) is generally smoother than IDHP, especially once the aircraft θ has reached up to the flat segment of θ_r in the manoeuvering segment, recovering better from the overshoot around 70 s. Second, the θ of MIDHP(λ) follows θ_r more closely throughout the entire phase, from pitching up, maintaining the high pitch, to pitching down, this improvement is largely a consequence of smoother control actions.

However, one interesting observation which could not have been made in the boxplots, can be made in sub-figure 8e and 8f. The elevator deflection can be seen to not go below -5 deg from the inception of the damaged and saturated elevator fault, up to until approximately 68 s, so not go beyond the saturated deflection as expected. What is less expected, however, is how all plotted flights of both algorithms continue deflecting the elevator at -5 deg even when θ crosses above θ_{ref}, which happens around 67 s. Contrast this with a purely gain based PID error feedback controller, which would likely have immediately flipped elevator deflection to the other direction.

Moreover, it can be seen in Figure 8 that during the high pitch phase, from 65 to 75 s, is where agents control a noticeably oscillatory action. While RL agents are, on occasion, known for such behaviour [30], another possible explanation could be the lack of integral action in the IDHP controllers. As the aircraft was trimmed for horizontal flight, dynamically its' pitch will tend to remain around low angles; e.g. at high pitch, airspeed drops, α increases, eventually θ drops as lift falls assuming a negative slope of the pitching moment in positive α. Therefore, attempts at maintaining a high pitch may be difficult without the presence of integral action; something that is missing in the here studied agents.

C. Summary

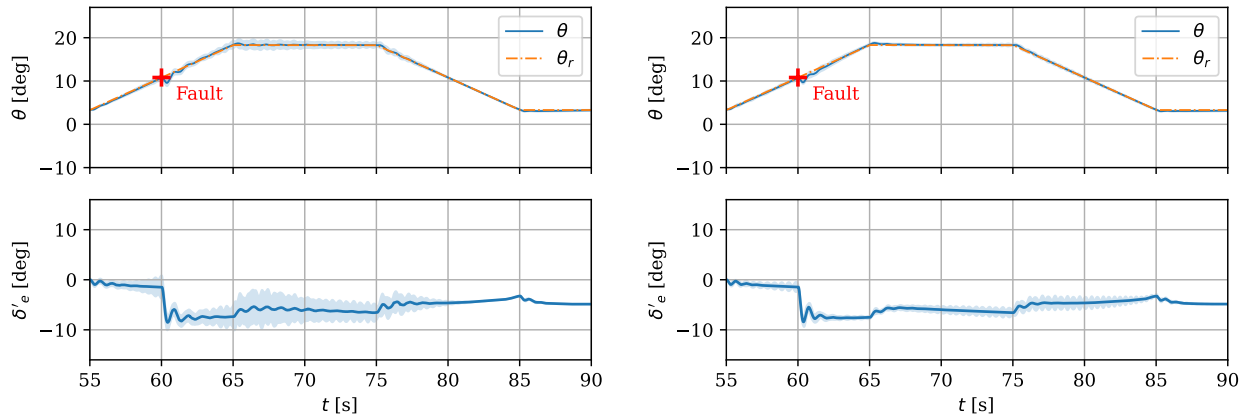
Observing the statistical test results, it can be seen that most tests demonstrate a statistically significant improvement in $|E|$ and Sm by the variants. The two notable outliers of these improvements are first the MIDHP performance in test 1, which is deemed significantly worse than that of IDHP; second, the smoothness of the IDHP(λ) controller versus IDHP, where IDHP(λ) was deemed in general noisier but with three out of four tests showing an insignificant change. Finally, considering the results of Tab. 4 and 5, the $|E|$ and Sm of MIDHP(λ) ranks best of all algorithms in three tests, ties for best in test 1, and ranks second in test 3. While there was little improvement in $|E|$ during warmup, MIDHP(λ)s' improvement in median $|E|$ during the manoeuvering phase ranged between 26% on Test 3 to 46% on Test 4. MIDHP(λ)s' Sm metrics similarly rank very high in all the tests, being the best from tests 2 to 5, and third in test 1.

V. Conclusion

Eligibility traces and Multi-step temporal difference errors are well known means of improving the sample efficiency and, on occasion, the asymptotic performance of RL agents in solving an MDP.

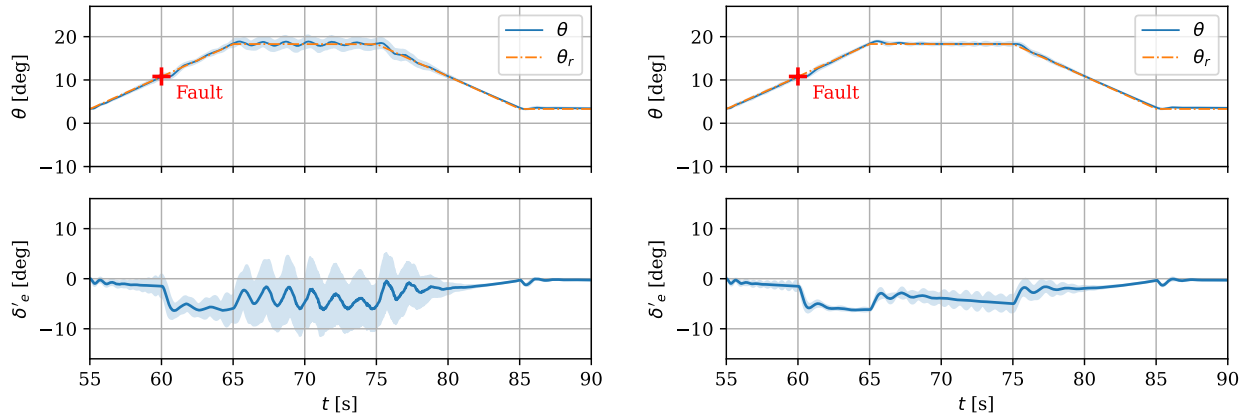
In this paper, these means have been incorporated into the ACD algorithm, IDHP, resulting in three variants. Simulated flight tests were then conducted, where a flight controller created from these RL algorithms was used to control the pitch attitude of an aircraft with various faults introduced. Through these tests, it was empirically shown that the proposed variants can improve the fault tolerance of a controller based on IDHP. In faults such as an elevator control effectiveness reduction of 70% and its deflection angles limited to a third of the original range, these variants not only improved tracking performance, but also in action smoothness: a pertinent issue in RL-based controllers. Out of all the augmentations, MIDHP(λ) showed the best tracking performance and the smoothest control actions.

The issue of success rate stands as an obstacle to the real-world online deployment of such controllers. Seeing as several runs result in highly oscillatory control actions in all the here-studied variants. Overcoming this issue will be an important step towards these algorithms' real-world deployment as intelligent and adaptive flight controllers. It is therefore worthwhile to investigate the dependence of this oscillation to a lack of integral control action and a dominance of proportional gain action, or how methods such as CAPS might be exploited more effectively to mitigate oscillations.



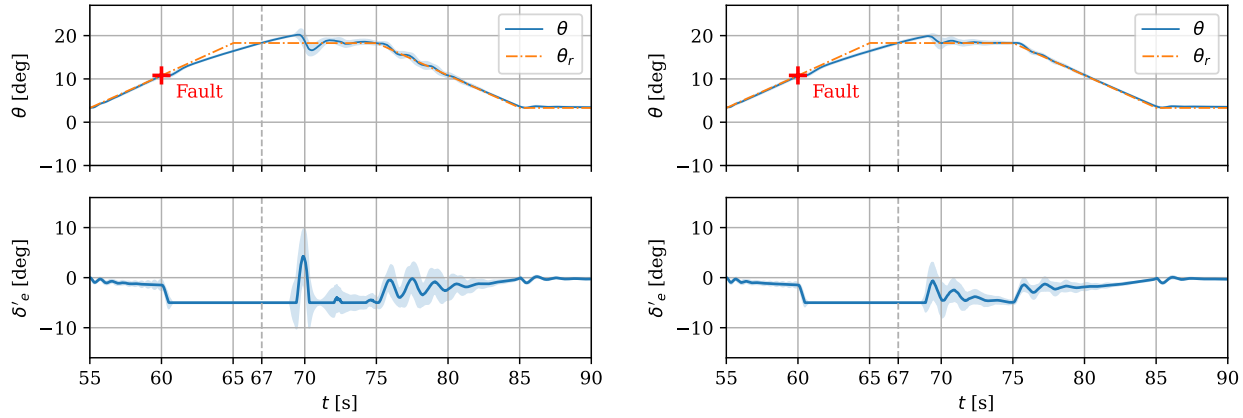
(a) IDHP shifted CG (No. $|E| > 2000 = 1$).

(b) MIDHP(λ) shifted CG (No. $|E| > 2000 = 8$).



(c) IDHP damaged elevator (No. $|E| > 2000 = 1$).

(d) MIDHP(λ) damaged elevator (No. $|E| > 2000 = 2$).



(e) IDHP damaged and saturated elevator (No. $|E| > 2000 = 7$).

(f) MIDHP(λ) damaged and saturated elevator (No. $|E| > 2000 = 4$).

Fig. 8 Comparing the Monte Carlo flights of IDHP and MIDHP(λ) under 3 faults, IDHP in 1st column, MIDHP(λ) in 2nd column, 2σ bounds shown in light blue shaded region.

References

[1] McDonald, R. A., German, B. J., Takahashi, T., Bil, C., Anemaat, W., Chaput, A., Vos, R., and Harrison, N., "Future aircraft concepts and design methods," *The Aeronautical Journal*, Vol. 126, No. 1295, 2022, pp. 92–124.

[2] Hodgkinson, D., and Johnston, R., *Aviation law and drones: Unmanned aircraft and the future of aviation*, Routledge, 2018.

[3] Yusaf, T., Fernandes, L., Abu Talib, A. R., Altarazi, Y. S., Alrefae, W., Kadirgama, K., Ramasamy, D., Jayasuriya, A., Brown, G., Mamat, R., et al., "Sustainable aviation—Hydrogen is the future," *Sustainability*, Vol. 14, No. 1, 2022, p. 548.

[4] Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Penicka, R., Song, Y., Cioffi, G., Kaufmann, E., and Scaramuzza, D., "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, 2024.

[5] Balas, G. J., "Flight control law design: An industry perspective," *European Journal of Control*, Vol. 9, No. 2-3, 2003, pp. 207–226.

[6] Kwatny, H., Dongmo, J.-E., Chang, B.-C., Bajpai, G., Yasar, M., and Belcastro, C., "Aircraft accident prevention: Loss-of-control analysis," *AIAA guidance, navigation, and control conference*, 2009, p. 6256.

[7] Sonneveldt, L., Van Oort, E., Chu, Q., and Mulder, J., "Nonlinear adaptive trajectory control applied to an F-16 model," *Journal of Guidance, control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 25–39.

[8] Johnson, E., Calise, A., and De Blauwe, H., "In flight validation of adaptive flight control methods," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6989.

[9] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D., "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *CoRR*, Vol. abs/1712.01815, 2017. URL `http://arxiv.org/abs/1712.01815`.

[10] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533. https://doi.org/10.1038/nature14236.

[11] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 6, 2021, pp. 4909–4926.

[12] Kober, J., Bagnell, J. A., and Peters, J., "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, Vol. 32, No. 11, 2013, pp. 1238–1274.

[13] Khan, S. G., Herrmann, G., Lewis, F. L., Pipe, T., and Melhuish, C., "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual reviews in control*, Vol. 36, No. 1, 2012, pp. 42–59.

[14] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., The MIT Press, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[15] Prokhorov, D. V., and Wunsch, D. C., "Adaptive critic designs," *IEEE transactions on Neural Networks*, Vol. 8, No. 5, 1997, pp. 997–1007.

[16] Zhou, Y., Van Kampen, E.-J., and Chu, Q., "Incremental model based online dual heuristic programming for nonlinear adaptive control," *Control Engineering Practice*, Vol. 73, 2018, pp. 13–25. https://doi.org/10.1016/j.conengprac.2017.12.011.

[17] Shayan, K., and Van Kampen, E.-J., "Online actor-critic-based adaptive control for a tailless aircraft with innovative control effectors," *AIAA Scitech 2021 Forum*, 2021, p. 0884.

[18] Li, H., Sun, L., Tan, W., Liu, X., and Dang, W., "Incremental dual heuristic dynamic programming based hybrid approach for multi-channel control of unstable tailless aircraft," *IEEE Access*, Vol. 10, 2022, pp. 31677–31691.

[19] Ferrari, S., and Stengel, R. F., "Online adaptive critic flight control," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 777–786.

[20] Li, T., Zhao, D., and Yi, J., "Heuristic Dynamic Programming strategy with eligibility traces," *2008 American Control Conference*, 2008, pp. 4535–4540. https://doi.org/10.1109/ACC.2008.4587210.

[21] Luo, B., Liu, D., Huang, T., Yang, X., and Ma, H., "Multi-step heuristic dynamic programming for optimal control of nonlinear discrete-time systems," *Information Sciences*, Vol. 411, 2017, pp. 66–83. https://doi.org/https://doi.org/10.1016/j.ins.2017.05.005.

[22] Wang, D., Wang, J., Zhao, M., Xin, P., and Qiao, J., "Adaptive Multi-Step Evaluation Design With Stability Guarantee for Discrete-Time Optimal Learning Control," *IEEE/CAA Journal of Automatica Sinica*, Vol. 10, No. 9, 2023, pp. 1797–1809. https://doi.org/10.1109/JAS.2023.123684.

[23] Enns, R., and Si, J., "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Transactions on Neural networks*, Vol. 14, No. 4, 2003, pp. 929–939.

[24] van Kampen, E.-J., Chu, Q., and Mulder, J., "Continuous adaptive critic flight control aided with approximated plant dynamics," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6429.

[25] Heyer, S., Kroezen, D., and Van Kampen, E.-J., "Online Adaptive Incremental Reinforcement Learning Flight Control for a CS-25 Class Aircraft," *AIAA SciTech 2022 Forum*, 2020, p. 1844. https://doi.org/10.2514/6.2020-1844.

[26] Watkins, C. J. C. H., "Learning from delayed rewards," Ph.D. thesis, King's College, Cambridge United Kingdom, 1989.

[27] Cichosz, P., "Truncating temporal differences: On the efficient implementation of TD (lambda) for reinforcement learning," *Journal of Artificial Intelligence Research*, Vol. 2, 1994, pp. 287–318.

[28] Singh, S. P., and Sutton, R. S., "Reinforcement learning with replacing eligibility traces," *Machine learning*, Vol. 22, No. 1, 1996, pp. 123–158.

[29] Ye, J., Bian, Y., Xu, B., Qin, Z., and Hu, M., "Online Optimal Control of Discrete-Time Systems Based on Globalized Dual Heuristic Programming with Eligibility Traces," *2021 3rd International Conference on Industrial Artificial Intelligence (IAI)*, 2021, pp. 1–6. https://doi.org/10.1109/IAI53119.2021.9619346.

[30] Mysore, S., Mabsout, B., Mancuso, R., and Saenko, K., "Regularizing action policies for smooth control with reinforcement learning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 1810–1816.

[31] Van Der Linden, C., "DASMAT-Delft University aircraft simulation model and analysis tool: A Matlab/Simulink environment for flight dynamics and control analysis," *Series 03: Control and Simulation 03*, 1998.

[32] Van den Hoek, M., de Visser, C., and Pool, D., "Identification of a Cessna Citation II model based on flight test data," *Advances in Aerospace Guidance, Navigation and Control: Selected Papers of the Fourth CEAS Specialist Conference on Guidance, Navigation and Control Held in Warsaw, Poland, April 2017*, Springer, 2018, pp. 259–277.

[33] Seres, P., "Distributional Reinforcement Learning for Flight Control," MSc thesis, Faculty of Aerospace Engineering, TU Delft, 2022.

[34] Vargha, A., and Delaney, H. D., "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, Vol. 25, No. 2, 2000, pp. 101–132.