Intelligent Adaptive Control Using LADP and IADP Applied to F-16 Aircraft with Imperfect Measurements

de Alvear Cardenas, J.I.; Sun, B.; van Kampen, E.

DOI
10.2514/6.2021-1119

Publication date
2021

Document Version
Final published version

Published in
AIAA Scitech 2021 Forum

AIAA SciTech Forum
11–15 & 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-1119

Intelligent Adaptive Control Using LADP and IADP Applied to F-16 Aircraft with Imperfect Measurements

José Ignacio de Alvear Cárdenas*, Bo Sun † and Erik-Jan van Kampen ‡

Linear Approximate Dynamic Programming (LADP) and Incremental Approximate Dynamic Programming (IADP) are Reinforcement Learning methods that seek to contribute to the field of Adaptive Flight Control. This paper assesses their performance and convergence, as well as the impact of sensor noise on policy convergence, online system identification, performance and control surface deflection. After summarising their theory and derivation with full state (FS) and output feedback (OPFB), they are implemented on the linearised longitudinal F-16 model. In order to establish an objective performance comparison, their hyper-parameters were tuned with an evolutionary algorithm: Particle Swarm Optimisation (PSO). Results show that LADP and IADP have the same performance in the presence of FS feedback, whereas LADP outperforms IADP when only OPFB is available. Output noise causes LADP based on OPFB to diverge. In the case of IADP based on OPFB, sensor noise improves the performance due to a better exploration of the solution space. The present research aims at bridging the gap between the discussed ADP algorithms and real world systems.

Nomenclature

α	=	Angle of attack, rad
γ	=	Forgetting factor, discounted rate or learning rate
γ_m^{RLS}	=	System identification forgetting factor
ϵ	=	Prediction error or innovation
ϵ_0	=	Probing noise
Θ	=	Parameter matrix
θ	=	Pitch angle, rad
μ	=	Policy
σ^2_{white}	=	White noise variance
A	=	State matrix
A_r	=	Reference state matrix
B	=	Input matrix
C	=	Output matrix
Cov	=	Estimation covariance matrix
C_r	=	Reference output matrix
c	=	One-step cost function
D	=	Throughput matrix
e	=	State (FS) or output (OPFB) tracking error
F	=	System matrix
F_t	=	Extended system matrix or system transition matrix
$\underline{\mathcal{F}}_t$	=	Extended system transition matrix
F^r	=	Reference output transition matrix
f	=	State function
f_s	=	Sampling frequency
f^r	=	Reference state function
G	=	Control effectiveness matrix
G_t	=	Extended control effectiveness matrix or input distribution matrix

*Graduate Student, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.
†PhD candidate, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.
‡Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology, AIAA Member.

$\underline{\mathscr{G}_t}$	=	Extended input distribution matrix
G^r	=	Reference output transition matrix
H	=	Observation matrix
h	=	Output function
J	=	Cost function
$k_x, k_u, k_{\Delta u}, k_e$	=	Uniformly distributed random number vectors
m	=	Number of inputs
N	=	Time horizon
n	=	Number of states
P	=	Value function or kernel matrix
p	=	Number of outputs
Q	=	Output/state cost matrix
q	=	Pitch rate, rad/s
R	=	Input cost matrix
R_N	=	Reference observability matrix
r	=	Number of reference states
t	=	Time step
U_N	=	Controllability matrix
u	=	Input vector
V	=	Velocity, m/s
$\overline{\mathscr{V}}_M, V_N, W_N$	=	Observability matrices
x	=	State vector
x^r	=	Reference state vector
y	=	Output vector
y^r	=	Reference output vector

I. Introduction

DURING the last decade, private and public institutions have turned their attention to the growing sector of urban air mobility. Multiple piloted and autonomous concepts have emerged, urging the need of robust flight control systems which guarantee safety in the envisioned crowded urban air space. The potential emergence of failures at low altitudes could be counteracted by Adaptive Flight Control systems which learn the new dynamics and bring the vehicle to safety.

At the moment, most flight control systems exploit gain scheduling [1], method which allows adaptation to different flight regimes but that can not cope with unexpected scenarios. Therefore, other nonlinear control approaches have been studied, such as Nonlinear Dynamic Inversion (NDI) [2, 3] and Backstepping (BS) [1]. The main idea behind NDI is to perform a state local coordinate transformation in order to cancel the nonlinear terms and linearize the dynamics of the aircraft. In contrast, BS is considered a selective NDI Lyapunov-based method which avoids the cancellation of useful nonlinearities. It is a recursive procedure which breaks the process into steps by designing intermediate control laws for "virtual control" states from the inner to the outer loops.

Incremental forms of these methods, such as Incremental NDI (INDI) [4–8] and Incremental BS (IBS) [9–12], rely on sensors and actuators to reduce the impact of model mismatch by computing the changes in the control inputs (e.g. control surface change in deflection) with respect to their current values. Besides that, adaptive frameworks, such as Adaptive BS [13, 14], count with dynamic feedback that constantly updates the static feedback control in order to deal with nonlinear systems with parametric uncertainties. For that purpose, online system identification is required which results in a difficult task when dealing with complex nonlinear aerodynamic aircraft models.

The main drawback of all the afore-mentioned methods is the need of full or partial knowledge of the system model and states. Therefore, there is a growing interest for intelligent control methods which do not require prior model knowledge, field which includes Reinforcement Learning (RL). In contrast with supervised learning in which input-output pairs are fed to the system, RL refers to an agent which learns from rewards/punishments received from its interaction with the environment, leading to the modification of its actions and control policy [15]. Given a certain goal, it is possible to control a system whose model is completely or partially unknown. Therefore, in the case of an unexpected situation, the flight control system is able to adapt its input to the system given the changed air vehicle dynamics.

Dynamic Programming is an optimisation method for solving, among others, RL problems by breaking them into sub-problems and solving them recursively. Therefore, the value of the larger problem is connected to the value of all its sub-problems, which can not be considered in isolation. It is based on Bellman's principle [16], which states that an optimal policy takes optimal decisions from any given state independently of the previous taken actions.

However, this framework requires the storing of the expected reward (value function) corresponding to all the states, which results in a high computational effort and the exponential growth of memory requirements with the addition of states. This phenomenon is known as the Curse of Dimensionality (CoD) [17]. Additionally, in order to determine the policy corresponding to the current vehicle state, information from the next time-step is required. Therefore, the Bellman's Principle results in a backwards-in-time procedure, common in offline planning methods.

Since online adaptive control is the final goal, Approximate Dynamic Programming was developed. This approach aims at solving the limitations of DP, namely the CoD, by combining it with Temporal Difference learning. The latter is a model-free online algorithm with an update rule for improving successive approximations of the desired value function (bootstrapping). This merge leads to two main concepts: Temporal Difference (TD) error and Value Function Approximation (VFA) [15]. TD allows a forward-in-time online solution method, whereas VFA solves the CoD by replacing the previous look-up table by an approximation of the desired value function.

Two model-free control approaches of ADP will be discussed, namely Linear Approximate Dynamic Programming (LADP) [18] and incremental Approximate Dynamic Programming (iADP) [19–21]. LADP deals with linear discrete-time systems whereas iADP exploits local operating point model linearizations in order to deal with nonlinear discrete-time systems; method inspired by already existing incremental approaches, such as the afore-mentioned INDI and IBS. iADP identifies directly the linearized system and control effectiveness matrices with Recursive Least Squares (RLS) [22] without exploiting a priori knowledge of the (unknown) system dynamics.

The implementation of noise plays a key role in LADP based on Output Feedback approaches (unknown states) and in all iADP agorithms, since it is required to excite the controlled system input in order to learn the system dynamics. However, most research in this field assumes the use of perfect sensors and ignores the potential impact of output noise in the performance of the methods. Hence the goal of this paper is to analyse the change in performance and convergence of LADP and iADP when non-ideal sensors are introduced. Its contribution is twofold. First, it provides an extensive literature review on both algorithms. Second, it compares their performance against each other with and without the presence of output noise; including their policy convergence and control surface deflection, as well as the identification of the linearised system in the iADP implementations. For that purpose, the algorithms are implemented in a linearised F-16 aircraft model [23].

The present paper is structured as follows. First, in Sections II and III, the theory and derivation is presented in detail for the LADP and iADP approaches, respectively. The regulation/stabilisation problem is laid out for each method in which the goal is to bring the system output to zero. The same is done for the tracking problem in which the system output or states should follow a reference signal as close as possible. In these problems, a cost function J will be used instead of the value function since, instead of rewards, penalties are given according to the error from the reference signal. Then, in Section IV, the implementations of the algorithms to the F-16 aircraft with and without sensor noise are discussed and their performance is compared. Finally, in Section V, conclusions will be drawn and some recommendations for future work will be given.

II. Linear ADP

Most of the content in this section is based on [18]. Linear Approximate Dynamic Programming (LADP) is an Approximate Dynamic Programming method which can be applied to the control of linear systems. For that purpose, the following discrete-time Linear Time Invariant (LTI) system notation is considered:

$$x_{t+1} = Ax_t + Bu_t \tag{1}$$

$$y_t = Cx_t, \tag{2}$$

where $x_t \in R^n$ is the state, $u_t \in R^m$ is the control input and $y_t \in R^p$ is the output. As can be seen, the control input is not present in the output equation (throughput matrix D = 0), and the A, B and C matrices are such that the system is controllable and observable.

Next, the application of LADP to two common problems in control will be discussed, namely regulation and tracking.

A. LADP for the regulation problem

The goal of the regulation problem is to bring the output or states to zero. Assuming that the states are known, in order to avoid the potential problems introduced by sensor noise on the system's output, this section aims at regulating the states. Therefore, with the aim of penalising the state error and minimising the control effort, the following one-step cost function is defined:

$$c_t = x_t^T Q x_t + u_t^T R u_t, \tag{3}$$

with Q=$Q^T \geq 0$ and R = $R^T > 0$. The values of the Q and R matrices are designed in order to prioritise the minimisation of the state error and the control effort, respectively. Furthermore, given a control policy $u_t = \mu(x_t)$, the Bellman equation can be described as follows:

$$J^\mu(x_t) = x_t^T Q x_t + u_t^T R u_t + \gamma J^\mu(x_{t+1}), \tag{4}$$

where $\gamma \in [0, 1]$ is known as the discounted rate or forgetting factor. An optimal policy can be derived from the Bellman's equation by selecting the policy which minimises its value, leading to the optimal control (u_t):

$$u_t = \mu^*(x_t) = \underset{u_t}{\operatorname{argmin}}(x_t^T Q x_t + u_t^T R u_t + \gamma J^*(x_{t+1})) \tag{5}$$

The true cost-to-go $J^\mu(x_t)$ should always be positive, given that it is the sum of quadratic values of the system inputs and outputs. In the past, it has been approximated successfully with Neural Networks [24] or multivariate splines [25]. However, in ADP, the goal is to capture the main features of the cost-to-go function instead of having an accurate and complex representation. Therefore, a surrogate function is used in order to approximate the true cost-to-go. With the aim of being able to carry out the policy evaluation step and making the problem tractable, the system cost function is reduced to a quadratic form in x_t with a symmetric, positive definite matrix P $\in R^{n \times n}$:

$$\hat{J}^\mu(x_t) = x_t^T P x_t \tag{6}$$

The optimal matrix P will be the "value function" which the algorithm will try to learn from its experience with the environment. After inserting Eq. (6) in Eq. (4):

$$x_t^T P x_t = x_t^T Q x_t + u_t^T R u_t + \gamma x_{t+1}^T P x_{t+1} \tag{7}$$

As observed in Eq. (5), the best policy corresponding to the learnt P in Eq. (7) can be computed by setting its derivative with respect to u_t to zero. This leads to the following control (u_t) as a function of the system state (x_t):

$$u_t = -(R/\gamma + B^T P B)^{-1} B^T P A x_t \tag{8}$$

Furthermore, in order to evaluate the policy, online real-time Temporal Difference methods are used. For that purpose, the temporal difference error is defined as:

$$\epsilon_t = -x_t^T P x_t + x_t^T Q x_t + u_t^T R u_t + \gamma x_{t+1}^T P x_{t+1} \tag{9}$$

In order to find the optimal cost for a given control policy, the error needs to be minimised:

$$0 = \underset{u_t}{min}(-x_t^T P x_t + x_t^T Q x_t + u_t^T R u_t + \gamma x_{t+1}^T P x_{t+1}) \tag{10}$$

This is a fixed point equation, which can be solved by applying a successive approximation with a contraction map. The resulting method is known as the Policy Iteration from which the optimal value and policy can be determined ([18]):

LADP Regulation Algorithm - PI using full state feedback

Select an initial stable control policy $u_t^0 = \mu^0(x_t)$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$0 = -x_t^T P^{j+1} x_t + x_t^T Q x_t + (u_t^j)^T R u_t^j + \gamma x_{t+1}^T P^{j+1} x_{t+1} \tag{11}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\begin{aligned} u_t^{j+1} &= \mu^{j+1}(x_t) \\ &= -(R/\gamma + B^T P^{j+1} B)^{-1} B^T P^{j+1} A x_t \end{aligned} \tag{12}$$

As can be observed, in PI it is required to start with an stable policy such that the solution is meaningful, as it has been shown by [26, 27] that certain conditions are required in order to converge to the optimal value and control policy.

Furthermore, the Policy Evaluation step consists of the Bellman equation, which is computed with LS. However, as an alternative, this algorithm could exploit the Lyapunov Iteration. For that purpose, the control input of Eq. (8) can be considered of the form $u_t = -Kx_t$, leading to the following state equation:

$$x_{t+1} = (A - BK)x_t, \tag{13}$$

where $K = (R/\gamma + B^T PB)^{-1} B^T PA$. Combining Eq. (13) with Eq. (7) leads to the Lyapunov equation required for the Policy Update:

LADP Regulation Algorithm - PI with Lyapunov Iteration using full state feedback

Select an initial stable control policy K^0.

1) **Policy Evaluation**: Solve for P^{j+1}:

$$0 = \gamma(A - BK^j)^T P^{j+1}(A - BK^j) + Q + (K^j)^T R K^j - P^{j+1} \tag{14}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$K^{j+1} = (R/\gamma + B^T P^{j+1} B)^{-1} B^T P^{j+1} A \tag{15}$$

As an alternative, the formulation of the optimal control policy in Eq. (10) can be modified by writing outside the minimisation function the component corresponding to the quadratic form of the current state. This leads to the following fixed-point equation:

$$x_t^T P x_t = \min_{u_t}(x_t^T Q x_t + (u_t)^T R u_t + \gamma x_{t+1}^T P x_{t+1}) \tag{16}$$

This new formulation leads to the Value Iteration algorithm:

LADP Regulation Algorithm - VI using full state feedback

Select an initial control policy $u_t^0 = \mu^0(x_t)$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$x_t^T P^{j+1} x_t = (x_t^T Q x_t + (u_t)^T R u_t + \gamma x_{t+1}^T P^j x_{t+1}) \tag{17}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\begin{aligned} u_t^{j+1} &= \mu^{j+1}(x_t) \\ &= -(R/\gamma + B^T P^{j+1} B)^{-1} B^T P^{j+1} A x_t \end{aligned} \tag{18}$$

As can be observed, whereas the first PI algorithm has the next step kernel matrix (P^{j+1}) as part of the current and next state quadratic forms, the VI algorithm uses the current kernel matrix (P^j) to compute P^{j+1}. Therefore, instead of the PI formulation in the form of a nonlinear Lyapunov equation, VI can provide a simpler recursion equation.

However, the PI and the VI algorithms have in common that both require knowledge of the system (A and B matrices) in order to use them. Therefore, they are not model-free approaches, constraining their use to a limited set of problems.

With the aim of achieving a model-free approach, the temporal difference error can be expressed in terms of the observed data (Input and Output) by employing the Output Feedback (OPFB) technique presented by [18]. Besides that, this technique has the added benefit that the complete measurement of the states is not required.

In order to exploit the OPFB technique, a time horizon of N+1 previously measured data points [t-N,t] is defined in order to represent the current state equation of the system:

$$x_t = A^N x_{t-N} + U_N \overline{u}_{t-1,t-N}, \tag{19}$$

where U_N represents the controllability matrix and $\overline{u}_{t-1,t-N} \in R^{mN}$ is a vertical vector of all the previous control inputs. Applying the same definitions to the output equation, this leads to:

$$y_t = CA^N x_{t-N} + CU_N \overline{u}_{t-1,t-N} \tag{20}$$

Based on the previous equation, the output vector for a given time frame [t-N,t-1] is:

$$\overline{y}_{t-1,t-N} = W_N x_{t-N} + D_N \overline{u}_{t-1,t-N}, \tag{21}$$

where W_N is the observability matrix:

$$W_N = \begin{bmatrix} CA^{N-1} \\ CA^{N-2} \\ \vdots \\ C \end{bmatrix}, \tag{22}$$

and D_N corresponds to a matrix of the form:

$$D_N = \begin{bmatrix} 0 & CB & CAB & \dots & CA^{N-2}B \\ 0 & 0 & CB & \dots & CA^{N-3}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & CB \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{23}$$

Simplifying Eq. (20) for x_{t-N} leads to:

$$x_{t-N} = W_N^+ (\overline{y}_{t-1,t-N} - D_N \overline{u}_{t-1,t-N}), \tag{24}$$

where $W_N^+ = (W_N^T W_N)^{-1} W_N^T$ is the pseudo-inverse. In this case, it is used to invert W_N, a vertical vector which, otherwise, would not be invertible due to its non-square matrix nature. Substituting Eq. (24) in Eq. (19) and rearranging the terms leads to:

$$\begin{aligned} x_t &= A^N W_N^+ \overline{y}_{t-1,t-N} + (U_N - A^N W_N^+ D_N) \overline{u}_{t-1,t-N} \\ &= \begin{bmatrix} U_N - A^N W_N^+ D_N & A^N W_N^+ \end{bmatrix} \begin{bmatrix} \overline{u}_{t-1,t-N} \\ \overline{y}_{t-1,t-N} \end{bmatrix} \\ &= M \overline{z}_{t-1,t-N}, \end{aligned} \tag{25}$$

where $\overline{z}_{t-1,t-N} \in R^{(m+p)N}$. This expression can be introduced in the quadratic form of the current and next step state, resulting in the following expression:

$$\begin{aligned} x_t^T P x_t &= \overline{z}_{t-1,t-N}^T M^T P M \overline{z}_{t-1,t-N} \\ &= \overline{z}_{t-1,t-N}^T \overline{P} \overline{z}_{t-1,t-N}, \end{aligned} \tag{26}$$

where $\overline{P} \in R^{(m+p)N \times (m+p)N}$ is the new kernel matrix that must be estimated. As can be observed, the dynamics of the system (A and B matrices) have been lumped into the new kernel matrix, resulting in that the knowledge of the system is not required beforehand.

Thanks to this new cost function, the Policy Evaluation of the new PI and VI can be modified accordingly. However, before defining the new versions of the algorithms, the changes to the Policy Improvement need to be investigated. In a similar fashion as with Eq. (5), the optimal policy is obtained by taking the derivative of the cost function with respect to the control input (u_t):

$$u_t = \underset{u_t}{\operatorname{argmin}}(y_t^T Q y_t + u_t^T R u_t + \gamma \overline{z}_{t,t-N+1}^T \overline{P} \overline{z}_{t,t-N+1}) \tag{27}$$

For that purpose, all the variables in Eq. (26) are decomposed as follows in order to discern which elements will be part of the policy update:

$$\overline{z}_{t-1,t-N}^T \overline{P} \overline{z}_{t-1,t-N} = \begin{bmatrix} u_t \\ \overline{u}_{t-1,t-N+1} \\ \overline{y}_{t,t-N+1} \end{bmatrix}^\intercal \begin{bmatrix} p_0 & p_u & p_y \\ p_u^T & P_{22} & P_{23} \\ p_y^T & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} u_t \\ \overline{u}_{t-1,t-N+1} \\ \overline{y}_{t,t-N+1} \end{bmatrix}, \tag{28}$$

with $p_0 \in R^{m \times m}$, $p_u \in R^{(m \times m)(N-1)}$ and $p_y \in R^{(m \times p)N}$. With this information, the new cost function presented in Eq. (27) can be differentiated with respect to u_t leading to:

$$u_t = -(R/\gamma + p_0)^{-1}(p_u \overline{u}_{t-1,t-N+1} + p_y \overline{y}_{t,t-N+1}) \tag{29}$$

Now, the PI and VI algorithms using output feedback (OPFB) can be presented for the case in which the states or the system dynamics are unknown ([18]):

LADP Regulation Algorithm - PI based on OPFB

Select an initial stable control policy $u_t^0 = \mu^0(\overline{z}_{t,t-N+1})$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$0 = -\overline{z}_{t-1,t-N}^T \overline{P}^{j+1} \overline{z}_{t-1,t-N} + y_t^T Q y_t + (u_t^j)^T R u_t^j + \gamma \overline{z}_{t,t-N+1}^T \overline{P}^{j+1} \overline{z}_{t,t-N+1} \tag{30}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$u_t^{j+1} = -(R/\gamma + p_0^{j+1})^{-1}(p_u^{j+1} \overline{u}_{t-1,t-N+1} + p_y^{j+1} \overline{y}_{t,t-N+1}) \tag{31}$$

LADP Regulation Algorithm - VI based on OPFB

Select an initial control policy $u_t^0 = \mu^0(\overline{z}_{t,t-N+1})$.

1) **Policy Evaluation**: Solve for P^{j+1}:

$$\overline{z}_{t-1,t-N}^T \overline{P}^{j+1} \overline{z}_{t-1,t-N} = y_t^T Q y_t + (u_t^j)^T R u_t^j + \gamma \overline{z}_{t,t-N+1}^T \overline{P}^{j} \overline{z}_{t,t-N+1} \tag{32}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$u_t^{j+1} = -(R/\gamma + p_0^{j+1})^{-1}(p_u^{j+1} \overline{u}_{t-1,t-N+1} + p_y^{j+1} \overline{y}_{t,t-N+1}) \tag{33}$$

As can be seen, now the input to the policy function for both algorithms is the $\overline{z}_{t,t-N+1}$ vector which contains information regarding the input and output of the system for the current step and the previous N-1 steps. However, it can be noticed that the u_t element of the $\overline{z}_{t,t-N+1}$ vector is not used in the Policy Improvement.

B. LADP for the tracking problem

When looked closely, the regulation problem can be considered to be the tracking problem when the reference signal to be followed is zero. Therefore, the tracking problem can be considered a generalisation of the regulation counterpart in which the reference signal has the following dynamics:

$$x_{t+1}^r = A_r x_t^r \tag{34}$$

$$y_t^r = C_r x_t^r, \tag{35}$$

where $x_t^r \in R^l$ is the internal state and $y_t^r \in R^r$ is the reference output. Given that the system states are known, they can be set to track the reference system internal states or reference system output. As a result, the potential detrimental effects of sensor noise in the system output are circumvented. Therefore, the one-step cost function of the tracking problem is now given by:

$$c_t = (x_t - x_t^r)^T Q(x_t - x_t^r) + u_t^T R u_t = e_t^T Q e_t + u_t^T R u_t, \tag{36}$$

where $e_t = (x_t - x_t^r)$ is the error signal. In order to solve the tracking problem in a similar manner as the regulation problem, an augmented system is presented by [28] and [29] in which the system and reference dynamics are combined. However, this implementation has 2 disadvantages. First, it requires the knowledge of the reference state matrix (A_r) and assumes a reference system without input matrix. Second, it may be desired to feed the tracking algorithm with state and output reference signals without the use of a reference state-space system of the form shown in Eq. (34) and Eq. (35). Therefore, the cost function has the following final structure:

$$e_t^T P e_t = e_t^T Q e_t + u_t^T R u_t + \gamma e_{t+1}^T P e_{t+1} \tag{37}$$

With the new cost function, the PI and VI algorithms can be formulated as follows:

LADP Tracking Algorithm - PI using full state feedback

Select an initial stable control policy $u_t^0 = \mu^0(x_t, x_{t+1}^r)$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$0 = -e_t^T P^{j+1} e_t + e_t^T Q e_t + (u_t^j)^T R u_t^j + \gamma e_{t+1}^T P^{j+1} e_{t+1} \tag{38}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\begin{aligned} u_t^{j+1} &= \mu^{j+1}(x_t, x_{t+1}^r) \\ &= -(R/\gamma + B^T P^{j+1} B)^{-1} B^T P^{j+1} (A x_t - x_{t+1}^r) \end{aligned} \tag{39}$$

LADP Tracking Algorithm - VI using full state feedback

Select an initial control policy $u_t^0 = \mu^0(x_t, x_{t+1}^r)$.

1) **Policy Evaluation**: Solve for P^{j+1}:

$$e_t^T P^{j+1} e_t = e_t^T Q e_t + (u_t^j)^T R(u_t^j) + \gamma e_{t+1}^T P^j e_{t+1} \tag{40}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\begin{aligned} u_t^{j+1} &= \mu^{j+1}(x_t, x_{t+1}^r) \\ &= -(R/\gamma + B^T P^{j+1} B)^{-1} B^T P^{j+1} (A x_t - x_{t+1}^r) \end{aligned} \tag{41}$$

As with the regulation problem, knowledge of the system dynamics (A, B and C) and states are required. However, in this case, also knowledge of the reference system dynamics (A_r and C_r) may be required if the augmented system is used.

In order to take into account the case in which this information is not available, an alternative formulation will be provided based on the system and reference input and output signals in a time horizon of N+1 time points [t-N,t]. This results in the following state and input equations:

$$x_t^r = A_r^N x_{t-N}^r \tag{42}$$

$$\overline{y}^r_{t-1,t-N} = \begin{bmatrix} y^r_{t-1} \\ y^r_{t-2} \\ \vdots \\ y^r_{t-N} \end{bmatrix} = \begin{bmatrix} C_r A_r^{N-1} \\ C_r A_r^{N-2} \\ \vdots \\ C_r \end{bmatrix} x^r_{t-N} = R_N x^r_{t-N} \tag{43}$$

Solving with respect to x^r_{t-N} yields:

$$x^r_{t-N} = R^+_N \overline{y}^r_{t-1,t-N}, \tag{44}$$

where R^+_N is the pseudo-inverse of the reference signal observability matrix. When substituting Eq. (44) in Eq. (42) (the state equation) and combining it with Eq. (25), the current system and reference states can be represented by the previous system inputs, and system and reference outputs:

$$X_t = \begin{bmatrix} x_t \\ x^r_t \end{bmatrix} = \begin{bmatrix} U_N - A^N W^+_N D_N & A^N W^+_N & 0 \\ 0 & 0 & A^N_r R^+_N \end{bmatrix} \begin{bmatrix} \overline{u}_{t-1,t-N} \\ \overline{y}_{t-1,t-N} \\ \overline{y}^r_{t-1,t-N} \end{bmatrix} = M \overline{Z}_{t-1,t-N} \tag{45}$$

As with Eq. (26), the cost function can be modified by exploiting the final expression in Eq. (45), leading to the following expression:

$$\begin{aligned} X^T_t P X_t &= \overline{Z}^T_{t-1,t-N} M^T P M \overline{Z}_{t-1,t-N} \\ &= \overline{Z}^T_{t-1,t-N} \overline{PZ}_{t-1,t-N}, \end{aligned} \tag{46}$$

where $\overline{P} \in R^{(m+p+r)N \times (m+p+r)N}$ is the new kernel matrix which contains the system and reference signal dynamics. As a result, this information is not required, leading to a model-free controller. Again, in order to define the Policy Improvement step, the policy can be derived from:

$$u_t = \underset{u_t}{\operatorname{argmin}}((y_t - y^r_t)^T Q (y_t - y^r_t) + u^T_t R u_t + \gamma \overline{Z}^T_{t,t-N+1} \overline{PZ}_{t,t-N+1}) \tag{47}$$

In contrast with the full state feedback case, now the states are unknown and the output signals are tracked. In order to determine which parameters will be part of the policy update after differentiating with respect to the current control input, the elements of the cost function are decomposed as follows:

$$\overline{Z}^T_{t,t-N+1} \overline{PZ}_{t-N+1} = \begin{bmatrix} u_t \\ \overline{u}_{t-1,t-N+1} \\ \overline{y}_{t,t-N+1} \\ \overline{y}^r_{t,t-N+1} \end{bmatrix}^T \begin{bmatrix} p_0 & p_u & p_y & p_r \\ p^T_u & P_{22} & P_{23} & P_{24} \\ p^T_y & P_{32} & P_{33} & P_{34} \\ p^T_r & P_{42} & P_{43} & P_{44} \end{bmatrix} \begin{bmatrix} u_t \\ \overline{u}_{t-1,t-N+1} \\ \overline{y}_{t,t-N+1} \\ \overline{y}^r_{t,t-N+1} \end{bmatrix}, \tag{48}$$

with $p_0 \in R^{m \times m}$, $p_u \in R^{m \times m(N-1)}$, $p_y \in R^{m \times pN}$ and $p_r \in R^{m \times rN}$. After differentiating Eq. (47) with respect to u_t, the following control input is obtained:

$$u_t = -(R/\gamma + p_0)^{-1} (p_u \overline{u}_{t-1,t-N+1} + p_y \overline{y}_{t,t-N+1} + p_r y^r_{t-N+1}) \tag{49}$$

Given Eq. (46) and Eq. (49), the PI and VI algorithms can be defined ([28]):

LADP Tracking Algorithm - PI based on OPFB

Select an initial stable control policy $u^0_t = \mu^0(\overline{Z}_{t,t-N+1})$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$0 = -\overline{Z}^T_{t-1,t-N} \overline{P}^{j+1} \overline{Z}_{t-1,t-N} + (y_t - y^r_t)^T Q (y_t - y^r_t) + (u^j_t)^T R u^j_t + \gamma \overline{Z}^T_{t,t-N+1} \overline{P}^{j+1} \overline{z}_{t,t-N+1} \tag{50}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$u^{j+1}_t = -(R/\gamma + p^{j+1}_0)^{-1} (p^{j+1}_u \overline{u}_{t-1,t-N+1} + p^{j+1}_y \overline{y}_{t,t-N+1} + p^{j+1}_r y^r_{t-N+1}) \tag{51}$$

LADP Tracking Algorithm - VI based on OPFB

Select an initial control policy $u_t^0 = \mu^0(\overline{Z}_{t,t-N+1})$.

1) **Policy Evaluation**: Solve for P^{j+1}:

$$\overline{Z}_{t-1,t-N}^T \overline{P}^{j+1} \overline{Z}_{t-1,t-N} = (y_t - y_t^r)^T Q(y_t - y_t^r) + (u_t^j)^T R u_t^j + \gamma \overline{Z}_{t,t-N+1}^T \overline{P}^j \overline{Z}_{t,t-N+1} \tag{52}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$u_t^{j+1} = -(R/\gamma + p_0^{j+1})^{-1}(p_u^{j+1} \overline{u}_{t-1,t-N+1} + p_y^{j+1} \overline{y}_{t,t-N+1} + p_r^{j+1} y_{t-N+1}^r) \tag{53}$$

III. Incremental Approximate Dynamic Programming

Most of the content in this section is based on [19–21]. As discussed in section II, LADP can only be exploited with linear systems, excluding many problems in aerospace engineering which deal with nonlinearities. For that purpose, Incremental Approximate Dynamic Programming (iADP) was developed. It is able to deal with these problems by using an incremental form of the nonlinear system, which is obtained by linearizing it around an operating point. In this paper, at a certain time (t), the instant before (t-1) is taken as the operating point.

Although most physical systems run in continuous-time, data are collected by means of discrete samples. Therefore, if a high sampling rate is assumed, the nonlinear model can be rewritten in discrete form:

$$x_{t+1} = f(x_t, u_t) \tag{54}$$

$$y_t = h(x_t), \tag{55}$$

where $f(x_t,u_t) \in R^n$, $u_t \in R^m$ and $h(x_t) \in R^p$.

In this section, only the VI algorithms will be included due to its benefits over PI explained in section II. Besides that, they can be deduced by the reader given that their difference with respect to the VI algorithms is the requirement of an stable initial policy and the fact that in PI, the kernel matrix on the right-hand side of the Policy Evaluation equation corresponds to iteration j+1.

A. iADP for the regulation problem

By taking the first Taylor series expansion of Eq. (54) around x_0 and u_0:

$$x_{t+1} = f(x_t, u_t) \approx f(x_0, u_0) + \left.\frac{\delta f(x_t, u_t)}{\delta x_t}\right|_{x_0,u_0} (x_t - x_0) + \left.\frac{\delta f(x_t, u_t)}{\delta u_t}\right|_{x_0,u_0} (u_t - u_0) \tag{56}$$

As mentioned before, the operating point around which the linearization takes place is the previous time step. This is possible thanks to the high sample rate, leading to $x_0 = x_{t-1}$, $u_0 = u_{t-1}$ and $f(x_0,u_0) = f(x_{t-1},u_{t-1}) = x_t$:

$$x_{t+1} - x_t \approx F(x_{t-1}, u_{t-1})(x_t - x_{t-1}) + G(x_{t-1}, u_{t-1})(u_t - u_{t-1}) \tag{57}$$

$$\Delta x_{t+1} \approx F(x_{t-1}, u_{t-1})\Delta x_t + G(x_{t-1}, u_{t-1})\Delta u_t, \tag{58}$$

where $F(x_{t-1},u_{t-1}) \in R^{n\times n}$ is the system matrix and $G(x_{t-1}, u_{t-1}) \in R^{n\times m}$ is the control effectiveness matrix at time step t-1. This results in a Linear Time Variant (LTV) system.

In order to identify the F and G matrices, Recursive Least Squares (RLS) is used [22]. If $F(x_{t-1},u_{t-1}) = F_{t-1}$ and $G(x_{t-1}, u_{t-1}) = G_{t-1}$, then Eq. (58) can be rewritten as:

$$\Delta x_{r,t+1} \approx \begin{bmatrix} \Delta x_t^T & \Delta u_t^T \end{bmatrix} \cdot \begin{bmatrix} f_{r,t-1}^T \\ g_{r,t-1}^T \end{bmatrix}, \tag{59}$$

where $f_{r,t-1}$ and $g_{r,t-1}$ are the elements of the rth row vector of F_{t-1} and G_{t-1}. Since all the parameters share the same covariance matrix, they can be identified together with the parameter matrix $\in \Theta^{(n+m)\times n}$:

$$\Theta_{t-1} = \begin{bmatrix} F_{t-1}^T \\ G_{t-1}^T \end{bmatrix} \tag{60}$$

As a result, the state prediction can be computed with:

$$\Delta\hat{x}_{t+1}^T = X_t^T \hat{\Theta}_{t-1}, \tag{61}$$

where $X_t \in R^{(n+m)\times 1}$ contains the input information of the incremental model indentification, namely Δx_t and Δu_t which are the differences between the current state (x_t) and current input (u_t) with the previous ones (x_{t-1} and u_{t-1}):

$$X_t = \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} \tag{62}$$

Then, the approach used in order to identify the incremental model is:

$$\epsilon_t = \Delta x_{t+1}^T - \Delta\hat{x}_{t+1}^T \tag{63}$$

$$\hat{\Theta}_t = \hat{\Theta}_{t-1} + \frac{\mathrm{Cov}_{t-1} X_t}{\gamma_m^{\mathrm{RLS}} + X_t^T \mathrm{Cov}_{t-1} X_t} \epsilon_t \tag{64}$$

$$\mathrm{Cov}_t = \frac{1}{\gamma_m^{\mathrm{RLS}}} \left(\mathrm{Cov}_{t-1} - \frac{\mathrm{Cov}_{t-1} X_t^T X_t \mathrm{Cov}_{t-1}}{\gamma_m^{\mathrm{RLS}} + X_t^T \mathrm{Cov}_{t-1} X_t} \right), \tag{65}$$

where $\epsilon_t \in R^n$ is the prediction error also called innovation, $\mathrm{Cov}_t \in R^{(n+m)\times(n+m))}$ is the estimation covariance matrix, and $\gamma_m^{\mathrm{RLS}} \in [0,1]$ is the forgetting factor in the RLS incremental identification procedure.

With every time step, the system and effectiveness matrix of the previous time step ($\hat{\Theta}_{t-1}$) are used, as well as the change in states and input (X_t) in order to compute the predicted state change with Eq. (61). Then, with the actual change of states obtained with the state equation (Eq. (54)), the prediction error is computed with Eq. (63). Finally, the new parameter matrix and covariance estimation can be computed with Eq. (64) and Eq. (65), respectively. From the parameter matrix, the F_{t-1} and G_{t-1} matrices can be obtained and used in the next time step.

Once the linearized matrices are computed, the one-step cost function can be defined to be the same as for the LADP, namely Eq. (3). Then, the cost function is defined as:

$$J^\mu(x_t) = x_t^T Q x_t + (u_{t-1} + \Delta u_t)^T R (u_{t-1} + \Delta u_t) + \gamma J^\mu(x_{t+1}) \tag{66}$$

Making the same assumption as in Eq. (6), the cost function becomes:

$$\hat{J}^\mu(x_t) = x_t^T Q x_t + (u_{t-1} + \Delta u_t)^T R (u_{t-1} + \Delta u_t) + \gamma (x_t + F_{t-1}\Delta x_t + G_{t-1}\Delta u_t)^T P (x_t + F_{t-1}\Delta x_t + G_{t-1}\Delta u_t) \tag{67}$$

Since the optimal control can be found in a similar fashion as in Eq. (5), by setting the derivative with respect to Δu_t to zero, it results in:

$$\Delta u_t = -(R + \gamma G_{t-1}^T P G_{t-1})^{-1} [R u_{t-1} + \gamma G_{t-1}^T P x_t + \gamma G_{t-1}^T P F_{t-1} \Delta x_t] \tag{68}$$

As can be observed, the policy is in the form of system variables (u_{t-1}, x_t and Δx_t) feedback and the different gains depend on the cost function matrices (P and R), as well as on the linearized system and effectiveness matrices (F_{t-1}, G_{t-1}). In contrast with online identification methods, which seek to find the global nonlinear system, the iADP algorithm is a much simpler and less prone to errors approach which requires local linear models.

With the previously derived equations, the corresponding VI algorithm can be presented:

iADP Regulation Algorithm - VI using full state feedback

Select an initial control policy $u_t^0 = \mu^0(u_{t-1}, x_t, \Delta x_t)$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$x_t^T P^{j+1} x_t = (x_t^T Q x_t + u_t^T R u_t + \gamma x_{t+1}^T P^j x_{t+1}) \tag{69}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\Delta u_t = -(R + \gamma G_{t-1}^T P^{j+1} G_{t-1})^{-1} [R u_{t-1} + \gamma G_{t-1}^T P^{j+1} x_t + \gamma G_{t-1}^T P^{j+1} F_{t-1} \Delta x_t] \tag{70}$$

Given that the full state of the air vehicle is not always available, Partially Observable Markov Decision Process (POMDP) frameworks can be exploited. For that purpose, the output can also be linearized with a Taylor expansion around x_{t-1}:

$$\Delta y_t \approx H_{t-1} \Delta x_t, \tag{71}$$

where $H_{t-1} \in R^{p \times n}$ is the observation matrix at time step t-1. As with the LADP, a time horizon is defined of previously measured data [t-N,t], leading to the following nonlinear system incremental dynamics:

$$\Delta x_t \approx \tilde{F}_{t-2,t-N-1} \cdot \Delta x_{t-N} + U_N \cdot \overline{\Delta u}_{t-1,t-N} \tag{72}$$

$$\overline{\Delta y}_{t,t-N+1} \approx V_N \cdot \Delta x_{t-N} + T_N \cdot \overline{\Delta u}_{t-1,t-N}, \tag{73}$$

where $\tilde{F}_{t-a,t-b} = \prod_{i=t-a}^{t-b} F_i = F_{t-a} \cdot \ldots \cdot F_{t-b}$, $\overline{\Delta u}_{t-1,t-N} \in R^{mN}$ and $\overline{\Delta y}_{t,t-N+1} \in R^{pN}$. Besides that, $T_N \in R^{pN \times mN}$, $U_N \in R^{n \times mN}$ is the controllability matrix and $V_N \in R^{pN \times m}$ is the observability matrix.

By left-multiplying Eq. (73) by the pseudo-inverse of the observability matrix (V_N^*) and substituting the equation of Δx_{t-N} into Eq. (72), the change of state can be computed as a function of previous inputs and outputs:

$$\begin{aligned} \Delta x_t &\approx \tilde{F}_{t-2,t-N-1} \cdot V_N^* \cdot \overline{\Delta y}_{t,t-N+1} + (U_N - \tilde{F}_{t-2,t-N-1} \cdot V_N^* \cdot T_N) \cdot \overline{\Delta u}_{t-,t-N} \\ &= \begin{bmatrix} M_{\Delta u} & M_{\Delta y} \end{bmatrix} \begin{bmatrix} \overline{\Delta u}_{t-1,t-N} \\ \overline{\Delta y}_{t,t-N+1} \end{bmatrix} \\ &= M_{t-1} \overline{\Delta z}_{t,t-N} \end{aligned} \tag{74}$$

Furthermore, the next goal is to be able to predict the output increment, namely Δy_{t+1}. For that purpose, Eq. (73) can be rewritten in a different manner in order to only take into account previous output increments:

$$\overline{\Delta y}_{t-1,t-N} \approx \overline{V}_N \cdot \Delta x_{t-N} + \overline{T}_N \cdot \overline{\Delta u}_{t-1,t-N}, \tag{75}$$

where $\overline{V}_N \in R^{pN \times n}$ and $\overline{T}_N \in R^{pN \times mN}$. By left-multiplying Eq. (75) with the pseudo-inverse of $\overline{V}_N$ ($\overline{V}_N^*$), substituting the resulted Δx_{t-N} into Eq. (72) and then the resulted Δx_t into Eq. (71), the increment of the output can be computed:

$$\begin{aligned} \Delta y_t &\approx (H_{t-1} U_N - H_{t-1} \tilde{F}_{t-2,t-N-1} \cdot \overline{V}_N^* \overline{T}_N) \cdot \overline{\Delta u}_{t-1,t-N} + H_{t-1} \tilde{F}_{t-2,t-N-1} V_N^* \cdot \overline{\Delta y}_{t-1,t-N} \\ &= \underline{G}_{t-1} \cdot \overline{\Delta u}_{t-1,t-N} + \underline{F}_{t-1} \cdot \overline{\Delta y}_{t-1,t-N} \end{aligned} \tag{76}$$

Translating this into Δy_{t+1}:

$$\Delta y_{t+1} = \underline{G}_{t,11} \cdot \Delta u_t + \underline{G}_{t,12} \cdot \overline{\Delta u}_{t-1,t-N+1} + \underline{F}_t \cdot \overline{\Delta y}_{t,t-N+1}, \tag{77}$$

where $\underline{G}_t \in R^{p \times Nm}$ is the extended control effectiveness matrix, $\underline{F}_t \in R^{p \times Np}$ is the extended system matrix, $\underline{G}_{t,11} \in R^{p \times m}$ and $\underline{G}_{t,12} \in R^{p \times (N-1)m}$ are partitioned matrices from $\underline{G}_t$. As can be observed, the system dynamics have been lumped into $\underline{G}_t$ and $\underline{F}_t$. These matrices are identifiable applying Recursive Least Squares (RLS) as done with iADP with full state feedback. For that purpose, Eq. (77) can be rewritten in the following form:

$$\Delta y_{r,t+1} \approx \begin{bmatrix} \overline{\Delta y}_{t,t-N+1}^T & \overline{\Delta u}_{t,t-N+1}^T \end{bmatrix} \cdot \begin{bmatrix} \underline{F}_{r,t}^T \\ \underline{G}_{r,t}^T \end{bmatrix}, \tag{78}$$

where $\Delta y_{r,t+1} = y_{r,t+1} - y_{r,t}$ is the increment of the r^{th} output, and $\underline{F}_{r,t}^T$ and $\underline{G}_{r,t}^T$ are the elements of the r^{th} row vector of $\underline{F}_t$ and $\underline{G}_t$. Since all the elements of the extended control effectiveness and extended system matrices share the same covariance matrix, they can be identified together in the parameter matrix $\Theta_t \in R^{(p+m)N \times p}$:

$$\Theta_t = \begin{bmatrix} \underline{F}_t^T \\ \underline{G}_t^T \end{bmatrix} \tag{79}$$

As a result, the output prediction can be computed with:

$$\Delta \hat{y}_{t+1}^T = \overline{X}_t^T \hat{\Theta}_t, \tag{80}$$

where $\overline{X}_t \in R^{(p+m)N \times 1}$ contains the input information to the extended incremental model identification, namely:

$$\overline{X}_t = \begin{bmatrix} \overline{\Delta y}_{t,t-N+1} \\ \overline{\Delta u}_{t,t-N+1} \end{bmatrix} \tag{81}$$

Then, the approach used in order to identify the incremental model is:

$$\epsilon_t = \Delta y_{t+1}^T - \Delta \hat{y}_{t+1}^T \tag{82}$$

$$\hat{\Theta}_{t+1} = \hat{\Theta}_t + \frac{\mathrm{Cov}_t \overline{X}_t}{\gamma_{em}^{\mathrm{RLS}} + \overline{X}_t^T \mathrm{Cov}_t \overline{X}_t} \epsilon_t \tag{83}$$

$$\mathrm{Cov}_{t+1} = \frac{1}{\gamma_{em}^{\mathrm{RLS}}} \left(\mathrm{Cov}_t - \frac{\mathrm{Cov}_t \overline{X}_t \overline{X}_t^T \mathrm{Cov}_t}{\gamma_{em}^{\mathrm{RLS}} + \overline{X}_t^T \mathrm{Cov}_t \overline{X}_t} \right), \tag{84}$$

where $\epsilon_t \in R^p$ is the prediction error also called innovation, $\mathrm{Cov}_t \in R^{(p+m)N \times (p+m)N)}$ is the estimation covariance matrix, and $\gamma_{em}^{\mathrm{RLS}} \in [0,1]$ is the forgetting factor in the RLS extended incremental identification procedure.

With every time step, the system and effectiveness matrix of the current time step ($\hat{\Theta}_t$) are used, as well as the change in states and input ($\overline{X}_t$) in order to compute the predicted output change with Eq. (80). Then, with the actual change of outputs obtained with the state equation (Eq. (55)), the prediction error is computed with Eq. (82). Finally, the new parameter matrix and covariance estimation can be computed with Eq. (83) and Eq. (84), respectively. From the parameter matrix, the $\underline{F}_t$ and $\underline{G}_t$ matrices can be obtained and used in the next time step.

Furthermore, the cost function of the system state at time t can be expressed as a function of the extended kernel matrix $\overline{P} \in R^{N(m+p) \times N(m+p)}$ in the quadratic form (as in Eq. (6)):

$$\hat{J}^{\mu}(z_{t,t-N}) = \overline{z}_{t,t-N}^T \overline{P} \overline{z}_{t,t-N}, \tag{85}$$

where $\overline{z}_{t,t-N}$ is the history of observation vectors:

$$\overline{z}_{t,t-N} = \begin{bmatrix} \overline{u}_{t-1,t-N}^T \\ \overline{y}_{t,t-N+1}^T \end{bmatrix} \tag{86}$$

The optimal policy can be written as:

$$\mu^* = \underset{\Delta u_t}{\mathrm{argmin}} (y_t^T Q y_t + u_t^T R u_t + \gamma \overline{z}_{t+1,t-N+1}^T \overline{P} \overline{z}_{t+1,t-N+1}) \tag{87}$$

where:

$$\overline{z}_{t+1,t-N+1}^{T}\overline{P}\overline{z}_{t+1,t-N+1} = \begin{bmatrix} u_{t-1}+\Delta u_t \\ \overline{u}_{t-1,t-N+1} \\ y_t+\Delta y_{t+1} \\ \overline{y}_{t,t-N+2} \end{bmatrix}^T \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{12}^T & P_{22} & P_{23} & P_{24} \\ P_{13}^T & P_{23}^T & P_{33} & P_{34} \\ P_{14}^T & P_{24}^T & P_{34}^T & P_{44} \end{bmatrix} \begin{bmatrix} u_{t-1}+\Delta u_t \\ \overline{u}_{t-1,t-N+1} \\ y_t+\Delta y_{t+1} \\ \overline{y}_{t,t-N+2} \end{bmatrix} \tag{88}$$

By taking the derivative with respect to Δu_t, the policy improvement step can be obtained in terms of the input and output measurements:

$$\begin{aligned} &-[R+\gamma P_{11}+\gamma G_{t,11}^T P_{33}\cdot G_{t,11}+\gamma P_{13}G_{t,11}+\gamma(P_{13}G_{t,11})^T]\cdot\Delta u_t \\ &\quad=[R+\gamma P_{11}+\gamma G_{t,11}^T P_{13}^T]u_{t-1}+\gamma[G_{t,11}^T P_{33}+P_{13}]y_t+\gamma[P_{12}+G_{t,11}^T P_{23}^T]\overline{u}_{t-1,t-N+1} \\ &\qquad+\gamma[P_{14}+G_{t,11}^T P_{34}]\overline{y}_{t,t-N+2}+\gamma[G_{t,11}^T P_{33}+P_{13}](G_{t,12}\cdot\overline{\Delta u}_{t-1,t-N+1}+F_t\cdot\overline{\Delta y}_{t,t-N+1}) \end{aligned} \tag{89}$$

Thanks to Eq. (85), Eq. (87) and Eq. (89), the VI algorithm can be defined for the iADP based on Output feedback:

iADP Regulation Algorithm - VI based on OPFB

Select an initial control policy $u_t^0 = \mu^0(u_{t-1}, y_t, \overline{u}_{t-1,t-N+1}, \overline{y}_{t,t-N+2}, \overline{\Delta u}_{t-1,t-N+1}, \overline{\Delta y}_{t,t-N+1})$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$\overline{z}_{t,t-N}^T\overline{P}^{j+1}\overline{z}_{t,t-N} = (y_t^T Q y_t + u_t^T R u_t + \gamma\overline{z}_{t+1,t-N+1}^T\overline{P}^j\overline{z}_{t+1,t-N+1}) \tag{90}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\begin{aligned} \Delta u_t = &-[R+\gamma P_{11}+\gamma G_{t,11}^T P_{33}\cdot G_{t,11}+\gamma P_{13}G_{t,11}+\gamma(P_{13}G_{t,11})^T]^{-1} \\ &\cdot\Big\{[R+\gamma P_{11}+\gamma G_{t,11}^T P_{13}^T]u_{t-1}+\gamma[G_{t,11}^T P_{33}+P_{13}]y_t+\gamma[P_{12}+G_{t,11}^T P_{23}^T]\overline{u}_{t-1,t-N+1} \\ &+\gamma[P_{14}+G_{t,11}^T P_{34}]\overline{y}_{t,t-N+2}+\gamma[G_{t,11}^T P_{33}+P_{13}](G_{t,12}\cdot\overline{\Delta u}_{t-1,t-N+1}+F_t\cdot\overline{\Delta y}_{t,t-N+1})\Big\} \end{aligned} \tag{91}$$

B. iADP for the tracking problem

This section will be a combination of the LADP tracking problem explained in Section II.B and the iADP regulation problem with full state feedback discussed in Section III.A.

As with Section III.A, the incremental model is defined by Eq. (58). Furthermore, the cost-to-go function is defined for the tracking problem with full state feedback as:

$$\begin{aligned} J^\mu(t) &= (x_t - x_t^r)^T Q(x_t - x_t^r) + u_t^T R u_t + \gamma J^\mu(t+1) \\ &= e_t^T P e_t \end{aligned} \tag{92}$$

As can be seen, the cost function adopts again a quadratic form in terms of the tracking error and the positive definite kernel P. Besides that, the tracking error is defined as:

$$\begin{aligned} e_{t+1} &= x_{t+1} - x_{t+1}^r \\ &\approx x_t + F_{t-1}\Delta x_t + G_{t-1}\Delta u_t - x_t^r - \Delta x_{t+1}^r \\ &= e_t + F_{t-1}\Delta x_t + G_{t-1}\Delta u_t - \Delta x_{t+1}^r \end{aligned} \tag{93}$$

The system transition matrix (F_{t-1}) and the input distribution matrix (G_{t-1}) are identified with RLS in a similar fashion as done in the iADP regulation problem (Eq. (59) to Eq. (65)). However, instead of using Δx_{t+1}^T in Eq. (59), Eq. (61) and Eq. (63) for the computation of the identification error, Δe_{t+1}^T is used for the tracking problem, as can be deduced from Equation 93.

Assuming a slow-variant reference signal and a sufficiently high sampling rate, Δx_{t+1}^r could be ignored; however, this parameter is kept in the rest of the derivation, leading to the following Bellman equation:

$$\hat{J}^\mu(e_t) = e_t^T Q e_t + u_t^T R u_t + \gamma(e_t + F_{t-1}\Delta x_t + G_{t-1}\Delta u_t - \Delta x_{t+1}^r)^T P(e_t + F_{t-1}\Delta x_t + G_{t-1}\Delta u_t - \Delta x_{t+1}^r) \tag{94}$$

Moreover, the optimal policy is defined as:

$$\mu^* = \underset{\Delta u_t}{\operatorname{argmin}}[e_t^T Q e_t + (u_{t-1} + \Delta u_t)^T R(u_{t-1} + \Delta u_t) + \gamma J^*(t+1)] \tag{95}$$

Therefore, by setting the derivative of the approximated cost function $\hat{J}^{\mu}(e_t)$ with respect to the change in control input (Δu_t) to zero, the optimal change in control input can be computed:

$$\Delta u_t = -(R + \gamma G_{t-1}^T P G_{t-1})^{-1}[R u_{t-1} + \gamma G_{t-1}^T P(e_t + F_{t-1}\Delta x_t - \Delta x_{t+1}^r)] \tag{96}$$

Thanks to the previous equations, the VI algorithm for the tracking problem with iADP with full state feedback can be defined:

iADP Tracking Algorithm - VI using full state feedback

Select an initial control policy $u_t^0 = \mu^0(u_{t-1}, e_t, \Delta x_t, \Delta x_{t+1}^r)$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$e_t^T P^{j+1} e_t = e_t^T Q e_t + u_t^T R u_t + \gamma e_{t+1}^T P^j e_{t+1} \tag{97}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\Delta u_t = -(R + \gamma G_{t-1}^T P^{j+1} G_{t-1})^{-1}[R u_{t-1} + \gamma G_{t-1}^T P^{j+1}(e_t + F_{t-1}\Delta x_t - \Delta x_{t+1}^r)] \tag{98}$$

However, in many real applications, the only measurement available is the tracking error and the control input provided to the system. It is therefore needed to analyse the system with partial observability. For that purpose, the output of the non-linear system can be linearised as in Eq. (71) with a Taylor expansion:

$$\Delta y_{t+1} = H_t \Delta x_{t+1} \tag{99}$$

In order to avoid assuming that reference signal is constant in the time horizon, the output reference signal is first-order continuous and can partially be a function of the system output. Therefore, the dynamics of the reference can be written in a discrete form with a high sampling frequency:

$$y_{t+1}^r = f^r(y_t^r, y_t) \tag{100}$$

By taking the first Taylor expansion, the incremental form of this discrete reference signal can be written as:

$$\Delta y_{t+1}^r \approx F_{t-1}^r \Delta y_t^r + G_{t-1}^r \Delta y_t, \tag{101}$$

where $F_{t-1}^r \in R^{pxp}$ and $G_{t-1}^r \in R^{pxp}$. The reference is stochastic and thus the F_{t-1}^r is time-varying. Besides that, if the reference is independent of the system, matrix G_{t-1}^r is a zero matrix. The system dynamics (Eq. (58) and Eq. (99)) can be combined with the reference dynamics (Eq. (101)), leading to the following extended system:

$$\begin{bmatrix} \Delta x_{t+1} \\ \Delta y_{t+1}^r \end{bmatrix} = \mathscr{F}_{t-1} \begin{bmatrix} \Delta x_t \\ \Delta y_t^r \end{bmatrix} + \mathscr{G}_{t-1} \Delta u_t \tag{102}$$

$$\Delta e_{t+1} = \mathscr{H}_t \begin{bmatrix} \Delta x_{t+1} \\ \Delta y_{t+1}^r \end{bmatrix}, \tag{103}$$

where:

$$\mathscr{F}_{t-1} = \begin{bmatrix} F_{t-1} & 0 \\ G_{t-1}^r H_{t-1} & F_{t-1}^r \end{bmatrix} \in R^{(n+p)\times(n+p)} \tag{104}$$

$$\mathscr{G}_{t-1} = \begin{bmatrix} G_{t-1} \\ 0 \end{bmatrix} \in R^{(n+p)\times m} \tag{105}$$

$$\mathscr{H}_t = \begin{bmatrix} H_t & 1 \end{bmatrix} \in R^{p\times(n+p)} \tag{106}$$

As with all the scenarios with partial observability, the system state and the tracking error equations can be expressed in terms of a time horizon [t-N,t]:

$$\begin{bmatrix} \Delta x_{t+1} \\ \Delta y^r_{t+1} \end{bmatrix} \approx \tilde{\mathscr{F}}_{t-1,t-M} \begin{bmatrix} \Delta x_{t-N+1} \\ \Delta y^r_{t-N+1} \end{bmatrix} + \mathscr{U}_M \overline{\Delta u}_{t,t-N+1} \tag{107}$$

$$\overline{\Delta e}_{t,t-N+1} \approx \mathscr{V}_M \begin{bmatrix} \Delta x_{t-N+1} \\ \Delta y^r_{t-N+1} \end{bmatrix} + \overline{\mathscr{T}}_M \cdot \overline{\Delta u}_{t,t-N+1}, \tag{108}$$

where $\tilde{\mathscr{F}}_{t-a,t-b} = \prod_{i=t-a}^{t-b} \mathscr{F}_i = \mathscr{F}_{t-a} \cdot \ldots \cdot \mathscr{F}_{t-b}$, $\mathscr{U}_M = \begin{bmatrix} \mathscr{G}_{t-1} & \mathscr{F}_{t-1}\mathscr{G}_{t-2} \ldots \tilde{\mathscr{F}}_{t-1.t-N+1} \cdot \mathscr{G}_{t-N} \end{bmatrix} \in R^{(n+p)\times nN}$

$$\overline{\Delta u}_{t,t-N+1} = \begin{bmatrix} \Delta u_t \\ \Delta u_{t-1} \\ \vdots \\ \Delta u_{t-N+1} \end{bmatrix} \in R^{mN\times 1} \tag{109}$$

$$\overline{\Delta e}_{t,t-N+1} = \begin{bmatrix} \Delta e_t \\ \Delta e_{t-1} \\ \vdots \\ \Delta e_{t-N+1} \end{bmatrix} \in R^{pN\times 1} \tag{110}$$

$$\overline{\mathscr{V}}_M = \begin{bmatrix} \mathscr{H}_{t-1}\tilde{\mathscr{F}}_{t-2,t-N} \\ \mathscr{H}_{t-2}\tilde{\mathscr{F}}_{t-3,t-N} \\ \vdots \\ \mathscr{H}_{t-N} \end{bmatrix} \in R^{pN\times(n+p)}, \tag{111}$$

is the observability matrix of this new system, and

$$\overline{\mathscr{T}}_M = \begin{bmatrix} 0 & \mathscr{H}_{t-1}\mathscr{G}_{t-2} & \mathscr{H}_{t-1}\mathscr{F}_{t-2}\mathscr{G}_{t-3} & \ldots & \mathscr{H}_{t-1}\tilde{\mathscr{F}}_{t-2,t-N+1} \cdot \mathscr{G}_{t-N} \\ 0 & 0 & \mathscr{H}_{t-2}\mathscr{G}_{t-3} & \ldots & \mathscr{H}_{t-2}\tilde{\mathscr{F}}_{t-3,t-N+1} \cdot \mathscr{G}_{t-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \mathscr{H}_{t-N+1} \cdot \mathscr{G}_{t-N} \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} \in R^{pN\times mN} \tag{112}$$

By left-multiplying Eq. (108) with the pseudo-inverse of $\overline{\mathscr{V}}_M$ ($\overline{\mathscr{V}}^{left}_M$), substituting the resulted $\begin{bmatrix} \Delta x_{t-N+1} \\ \Delta y^r_{t-N+1} \end{bmatrix}$ into Eq. (107) and then the resulted $\begin{bmatrix} \Delta x_{t+1} \\ \Delta y^r_{t+1} \end{bmatrix}$ into Eq. (103), the dynamics of the increment in the tracking error can be obtained:

$$\Delta e_{t+1} \approx (\mathscr{H}_t\mathscr{U}_M - \mathscr{H}_t\tilde{\mathscr{F}}_{t-1,t-M} \cdot \overline{\mathscr{V}}^{left}_M \overline{\mathscr{T}}_M) \cdot \overline{\Delta u}_{t,t-N+1} + \mathscr{H}_t\tilde{\mathscr{F}}_{t-1,t-N}\overline{\mathscr{V}}^{left}_M \cdot \overline{\Delta e}_{t,t-N+1} \tag{113}$$

As can be observed, the incremental form of the tracking error depends only on the previous observations of the tracking error and the control input in the time horizon [t-N,t]. Lumping all the system information into an extended input distribution matrix ($\underline{\mathscr{G}}_t \in R^{p\times Nm}$) and an extended transition matrix ($\underline{\mathscr{F}}_t \in R^{p\times Np}$), it is possible to express the tracking error increment as follows:

$$\Delta e_{t+1} \approx \underline{\mathscr{G}}_t \cdot \overline{\Delta u}_{t,t-N+1} + \underline{\mathscr{F}}_t \cdot \overline{\Delta e}_{t,t-N+1} \tag{114}$$

These new system matrices ($\underline{\mathcal{G}}_t$ and $\underline{\mathcal{F}}_t$) can be obtained using the same RLS procedure discussed in between Eq. (78) and Eq. (84). The only difference is the use of Δe_{t+1} instead of Δy_{t+1}^T in Eq. (78), Eq. (80) and Eq. (82) for the computation of the identification error. Then, with the identified incremental extended model, it is possible to predict the tracking error:

$$e_{t+1} = e_t + \underline{\mathcal{G}}_{t,11} \cdot \Delta u_t + \underline{\mathcal{G}}_{t,12} \cdot \overline{\Delta u}_{t-1,t-N+1} + \underline{\mathcal{F}}_t \cdot \overline{\Delta e}_{t,t-N+1}, \tag{115}$$

where $\underline{\mathcal{G}}_{t,11} \in R^{p\times m}$ and $\underline{\mathcal{G}}_{t,12} \in R^{p\times(N-1)m}$ are partitioned matrices from $\underline{\mathcal{G}}_t$.

Furthermore, as in all the previous algorithms (e.g. Equation 92), it is possible to express the cost function in quadratic form; in this case, in terms of the tracking error and the kernel matrix $P \in R^{p\times p}$:

$$\begin{aligned} J^\mu(e_t) &= e_t^T Q e_t + (u_{t-1} + \Delta u_t)^T R(u_{t-1} + \Delta u_t) + \gamma J^\mu(t+1) \\ &= e_t^T P e_t \end{aligned} \tag{116}$$

From the expression of the cost function, it is possible to find the optimal control input by setting to zero its derivative with respect to Δu_t, leading to:

$$\Delta u_t = -(R + \gamma \underline{\mathcal{G}}_{t,11}^T P \underline{\mathcal{G}}_{t,11})^{-1} \cdot \left[R u_{t-1} + \gamma \underline{\mathcal{G}}_{t,11}^T P (e_t + \underline{\mathcal{G}}_{t,12} \cdot \overline{\Delta u}_{t-1,t-N+1} + \underline{\mathcal{F}}_t \cdot \overline{\Delta e}_{t,t-N+1}) \right] \tag{117}$$

Thanks to the definition of the cost function and the control input update, it is possible to define the iADP VI algorithm for the tracking problem based on OPFB:

iADP Tracking Algorithm - VI based on OPFB

Select an initial control policy $u_t^0 = \mu^0(u_{t-1}, e_t, \overline{\Delta u}_{t-1,t-N+1}, \overline{\Delta e}_{t,t-N+1})$.

1) **Policy Evaluation**: Solve for P^{j+1} (e.g. using Least Squares (LS)):

$$e_t^T P^{j+1} e_t = e_t^T Q e_t + u_t^T R u_t + \gamma e_{t+1}^T P^j e_{t+1} \tag{118}$$

2) **Policy Improvement**: Use the new kernel P^{j+1} update to improve the policy:

$$\Delta u_t = -(R + \gamma \underline{\mathcal{G}}_{t,11}^T P^{j+1} \underline{\mathcal{G}}_{t,11})^{-1} \cdot \left[R u_{t-1} + \gamma \underline{\mathcal{G}}_{t,11}^T P^{j+1} (e_t + \underline{\mathcal{G}}_{t,12} \cdot \overline{\Delta u}_{t-1,t-N+1} + \underline{\mathcal{F}}_t \cdot \overline{\Delta e}_{t,t-N+1}) \right] \tag{119}$$

IV. Performance and noise analysis

In order to assess the performance of the LADP and the iADP, the non-linear model of the F-16 aircraft obtained from the University of Minnesota [23] was used. The low-fidelity aerodynamic model was chosen since it does not include the effects of the leading edge flap and there is a complete decoupling between the longitudinal and lateral directions. Furthermore, it was trimmed in steady wings-level flight at 5000 [ft] and 300 [ft/s], and it was discretized with a discretization time of 0.5.

An overview of all the algorithms discussed in Sections II and III can be observed in Fig. 1. Including the possibility of adding sensor noise to each algorithm, there exist in total 32 branches: 16 for the longitudinal F-16 model and 16 for the lateral F-16 model. As a result, due to the large number of combinations, a few had to be selected for performance comparison. For that purpose, all the branches corresponding to the regulation problem are ignored, since they can be considered a simplification of their tracking counterparts in which the system states or outputs must track constant signals with zero value. Furthermore, this paper will focus on the SIMO F-16 longitudinal model. To sum up, this document will compare the branches shown in Fig. 2, progressing from the LADP algorithms (5-8) to the iADP (13-16).

The state-space matrices of the longitudinal F-16 model trimmed at the afore-mentioned altitude and speed conditions, before applying the discretization time, can be seen in Eq. (120) to Eq. (123). The state vector of the longitudinal model is composed of the total velocity, the angle of attach, the pitch angle and the pitch rate: $x = \begin{bmatrix} V & \alpha & \theta & q \end{bmatrix}^T$.

$$A = \begin{bmatrix} -0.0291 & 2.1300 & -32.1700 & -2.8952 \\ -0.0007 & -0.5447 & 0 & 0.9152 \\ 0 & 0 & 0 & 1.0000 \\ 0 & 0.3303 & 0 & -0.8169 \end{bmatrix} \tag{120}$$

$$B = \begin{bmatrix} -0.0045 \\ -0.0011 \\ 0 \\ -0.0570 \end{bmatrix} \tag{121}$$

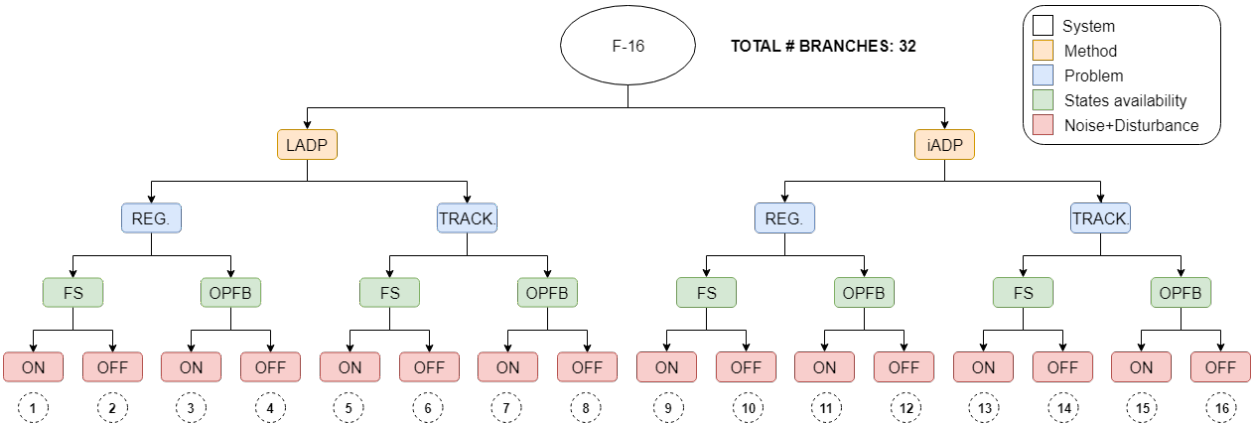

Fig. 1 Summary of algorithms.

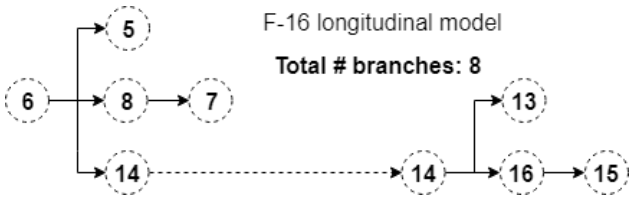

Fig. 2 Selected branches for comparison

$$C = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 57.2958 & 0 & 0 \\ 0 & 0 & 57.2958 & 0 \\ 0 & 0 & 0 & 57.2958 \end{bmatrix} \quad (122) \qquad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (123)$$

Furthermore, signals have to be generated for the algorithms to track. When it is desired that the tracking is performed by a single state or output, sinusoidal reference signals can be created with a function or a state equation of the following form:

$$\begin{bmatrix} \sin(t + \Delta t) \\ \cos(t + \Delta t) \end{bmatrix} = \begin{bmatrix} \cos \Delta t & \sin \Delta t \\ -\sin \Delta t & \cos \Delta t \end{bmatrix} \begin{bmatrix} \sin t \\ \cos t \end{bmatrix} \quad (124)$$

However, when multiple signals must be followed by different states or outputs (e.g. the side-slip and the roll angles), a different method must be employed since the system might not be able to follow random generated signals by its states, leading to an apparent low tracking performance.

With the aim of creating reference signals that the longitudinal system can track, a simple PD controller is implemented in which the angle of attack must follow a sinusoidal signal, namely $5\sin(0.04t)$, while the value of all the system outputs are being recorded. Given that the output matrix (Eq. (122)) only converts the state values from [rad] or [rad/s] to [deg] or [deg/s], the stored outputs can be multiplied by $\pi/180$ in order to obtain the corresponding states. At the end of this process, reference state and output signals are available for the system to track.

Since the goal is to generate tracking signals that the system should be able to follow, tight boundaries were provided in order to minimise strong oscillations in the output signals, specially for the derivative gain; smooth signals are preferred over perfectly following the side-slip angle reference sinusoid. Once the PD controllers are tuned, the output signals are discretised with a discretisation time of 0.5, the same value used for the discretisation of the F-16 longitudinal model, and fed to the algorithms. The discretisation time of the model is directly related to the sampling frequency or number of control inputs fed to the system per unit of time. The higher the sampling frequency, the better the performance. The impact on the performance by the choice of sampling frequency, as well as other implementation parameters, is discussed in detail in the Appendix.

Furthermore, for each algorithm some hyper-parameters have to be tuned for good performance, such as the learning rate (γ), the output/state cost matrix (Q) and the input cost matrix (R). Manually tuning is an option; however, this process can be biased and subjective, resulting in an unfair comparison between algorithms. Therefore, in order to establish a fair and objective comparison, PSO is exploited for hyper-parameter tuning. For that purpose, the ranges shown in Table 1 will be the common optimisation boundaries for all algorithms.

Table 1 Hyper-parameters optimisation boundaries.

Param.	Boundaries	Param.	Boundaries	Param.	Boundaries
Q	$[10^{-6}, 10^{8}]$	R	$[10^{-6}, 10^{8}]$	γ	$[10^{-6}, 0.9]$
		N	[3, 8]		

Previous research [30, 31] has shown that PSO is very stable and flexible in the presence of noise and when the landscape is continuously changing. However, the presence of output noise in some implementations can lead to different fitness values for identical hyper-parameters, a deception which can be mitigated by averaging over a number of fitness measurements (re-sampling) or by increasing the number of fitness evaluations (particles) [32]. In this research, the first option is applied in order to prevent overfitting to a specific output noise signal, running 10 simulations per particle with different seeds for the random number generator. As a result, whereas the performance of each simulation is the average RMSE of the last 30 iterations (those iterations in which it is highly likely that the P matrix will have converged), the fitness value of a particle is the average of the performances of its 10 simulations. 30 iterations were taken instead of only the last one in order to prevent favouring those solutions that happen to only perform well in the last iteration.

Additionally, in order to reject unfeasible control surface deflections, the elevator input has been constrained to (-25, 25) [deg] with a maximum deflection rate of 60 [deg/s], as provided in the Minnesota model. Also, those scenarios in which the policy function has not converged are not considered during the PSO optimisation process, except if stated otherwise in the implementation.

Furthermore, in order to carry out the Policy Evaluation, Ordinary Least Squares (OLS) is used in all the algorithms for the computation of the P matrix. For that purpose, the left-hand side of the equation is modified such that the state measurements are separated from the parameters of the kernel matrix which have to be estimated. In the case of the LADP tracking problem with full state feedback, this translates into:

$$e_t^T P^{j+1} e_t = (e_t \otimes e_t)^T \operatorname{vec}(P^{j+1}) = \overline{e}\Theta, \tag{125}$$

where $\otimes$ is the Kronecker product, $\operatorname{vec}(\cdot)$ is the columns stacking operator, $\overline{e} \in R^{1\times n^2}$ is the row vector containing the state errors and $\Theta \in R^{n^2\times 1}$ is the column vector containing the parameters of the P^{j+1} to be estimated. Furthermore, the right-hand side is defined under the output measurement y_{LS}:

$$y_{LS} = e_t^T Q e_t + (u_t^j)^T R(u_t^j) + \gamma e_{t+1}^T P^j e_{t+1} \tag{126}$$

As a result, after collecting all the state and output measurements of all the time-steps corresponding to a complete iteration, the parameters are estimated with the following equation:

$$\hat{\Theta}_{LS} = X_{LS}^{+}(\overline{e}) y_{LS}, \tag{127}$$

where X_{LS}^{+} is the pseudo-inverse of the regression matrix containing the state errors stored throughout an iteration.

Finally, in order to compare the performance of the different algorithms, the longitudinal algorithms are given the task of tracking an angle of attack and pitch rate reference signals. In order to avoid the influence of the random initialisation of some parameters, the mean is taken of the Root Mean Square Error (RMSE) of the last 30 iterations of a 100 trials for each algorithm for comparison. In the following sections, the initialisation and the optimised hyper-parameter values for each of the algorithms will be presented and each of the implementation results will be analysed.

A. LADP with FS feedback without sensor noise (Algo. 6)

First, the LADP algorithm with FS feedback applied to the tracking problem without sensor noise is studied. In Table 2, the initialisation of the different parameters is shown, including in ***bold italics*** the optimised values with PSO.

x_0 is the initialisation of the states, which means that the system starts with an angle of attack of 5°. With these parameters, Fig. 3 shows the convergence of the diagonal elements of the P matrix with respect to the iterations. Since

Table 2 Parameter initialisation and hyper-parameter tuning (Algo. 6).

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$4.64 \cdot 10^7$	$\boldsymbol{R}$	10^{-6}	γ	10^{-6}
P_0	$I_{2\times2}$	Iterations	50	Time steps	500
		x_0	$[0,5°,0,0]^T$		

there are 2 states to be tracked, the P matrix is a square matrix of dimension 2. As can be observed, convergence is achieved after 3 iterations with a final mean RMSE error of 0.0043 for 100 trials. This error was computed from the difference between the reference signals and the controlled system outputs in the last iteration.

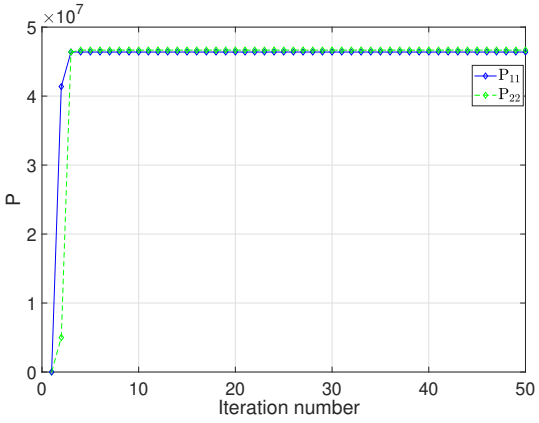

Fig. 3 Parameters of the kernel matrix P (Algo. 6).

In Fig. 4 and Fig. 5 the reference and the system output signals can be observed for the first and last iterations respectively. Besides that, in Fig. 6 it can be seen the elevator deflection for the last iteration. As can be observed, the signal is smooth and the deflection and rates limits are met.

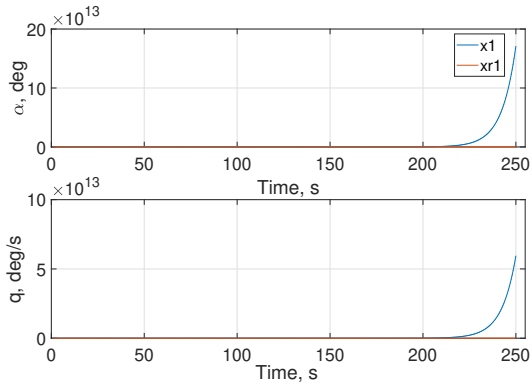

Fig. 4 Evolution of the reference and system states in the first iteration (Algo. 6).

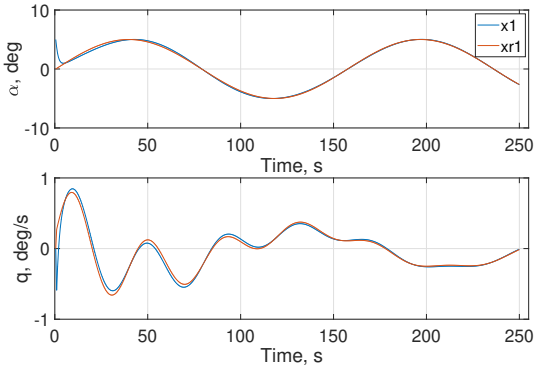

Fig. 5 Evolution of the reference and system states in the fiftieth iteration (Algo. 6).

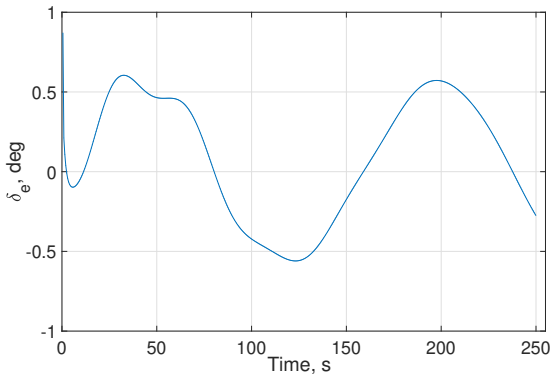

Fig. 6 Elevator deflection in the fiftieth iteration (Algo. 6).

B. LADP with FS feedback with sensor noise (Algo. 5)

Thanks to the full state feedback, it is possible to track the reference signals with the states. Therefore, the presence of noise in the output signals of the system does not influence the results, which are the same as discussed in subsection IV.A.

C. LADP based on OPFB without sensor noise (Algo. 8)

The following implementation makes use of the parameter initialisation shown in Table 3, including in ***bold italics*** the optimised values with PSO.

Table 3 Parameter initialisation and hyper-parameter tuning (Algo. 8).

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$3.08{\cdot}10^7$	$\boldsymbol{R}$	10^{-6}	$\boldsymbol{\gamma}$	10^{-6}
$\boldsymbol{N}$	5	P_0	$I_{25\times25}$	Iterations	50
Time steps	500	x_0	$[0{,}5°{,}0{,}0]^T$		

Given that the algorithm implicitly discovers the system dynamics by interaction with the environment, a good exploration of the state-space is required in order to maximise the possibility of obtaining the globally optimum kernel matrix. In order to generate Persistent Excitation (PE) in the system, probing noise is added to the control input:

$$u_t = \mu^j(\overline{z}_{t,t-N+1}) + \epsilon_0, \tag{128}$$

where ϵ_0 is a signal that is a sum of sine waves with different amplitudes, frequencies and phases. In this case, the following sum of sinusoid signals was used:

$$\begin{aligned}\epsilon_0 = \frac{0.006}{t}\Big(&\sin(100t)^2\cdot\cos(100t) + \sin(2t)^2\cdot\cos(0.1t) + \\ &\sin(-1.2t)^2\cdot\cos(0.5t) + \sin(t)^5 + \sin(1.12t)^2 + \cos(2.4t)\cdot\sin(2.4t)^3\Big),\end{aligned} \tag{129}$$

where t represents the time step at which an action is taken, leading to a different noise for each time step. However, the noise added to the input of the system is the same in each iteration. In this way, the probing signal serves it purpose by contributing to the exploration and the system learns to counteract it after a few iterations. Its impact on the performance is discussed in more detail in the Appendix.

As can be seen in Eq. (46), the $\overline{P}$ matrix has a size of 25×25 since there is 1 input (m), two outputs (p), 2 reference signals (r) and N has a value of 5. As in the previous implementation, the convergence of the $\overline{P}$ matrix can be observed

in a graph of its diagonal values with respect to the iteration number, as shown in Fig. 7. Only every 3 diagonal elements of the $\overline{P}$ are plotted and convergence can be observed after the 5^{th} iteration, 2 iterations later when compared to LADP with FS feedback.

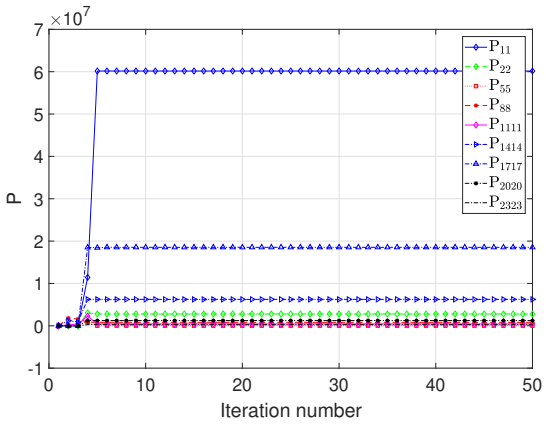

Fig. 7 Parameters of the kernel matrix P with Eq. (129) **as exploration signal (Algo. 8).**

As in Section IV.A, the mean RMSE over a 100 trials is computed, leading to a value of 0.2709. When compared to the full state feedback scenario, the RMSE is 63 times higher. This is due to the extra task of implicitly online identifying the system dynamics with the state-space exploration. As before, Fig. 8 show the output signal of the system and the reference for the last iteration. The first iteration is very similar to Fig. 4.

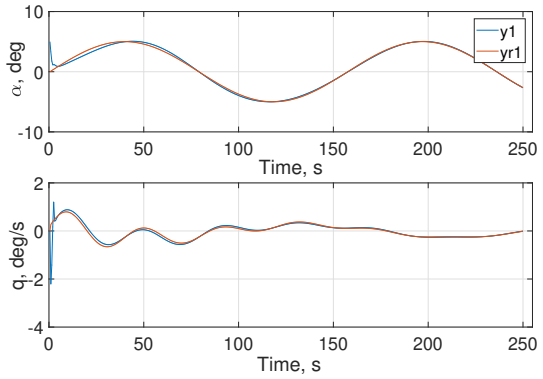

Fig. 8 Evolution of the reference and system outputs in the fiftieth iteration with Eq. (129) **as exploration signal (Algo. 8).**

D. LADP based on OPFB with sensor noise (Algo. 7)

In contrast with all the previous cases, when performing the parameter optimisation of the LADP based on OPFB with sensor noise, the PSO algorithm was not able to converge when having to track the angle of attack and q signals with an initial angle of attack deviation of 5 [deg]. In order to verify whether it was possible to obtain decent results with LADP based on OPFB with sensor noise, it was decided that the algorithm should only track the angle of attack with no initial deviation. Besides that, those solutions in which the policy function (the P matrix) did not converge were still considered in the optimisation process, otherwise no solution could be found. The parameter initialisation for this implementation can then be observed in Table 4, including in ***bold italics*** the optimised values with PSO.

As in subsection IV.C, excitation by means of a probing signal is included in order to explore the solution space, using Eq. (129). The behaviour of the $\bar{P}$ matrix can be observed in a graph of it diagonal values with respect to the iteration number, as shown in Fig. 9.

Table 4 Parameter initialisation and hyper-parameter tuning (Algo. 7).

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$9.37{\cdot}10^7$	$\boldsymbol{R}$	$5{\cdot}10^7$	$\boldsymbol{\gamma}$	1
$\boldsymbol{N}$	2	P_0	$I_{6\times6}$	Iterations	50
Time steps	500	x_0	$[0,0,0,0]^T$	σ^2_{white}	10^{-9}

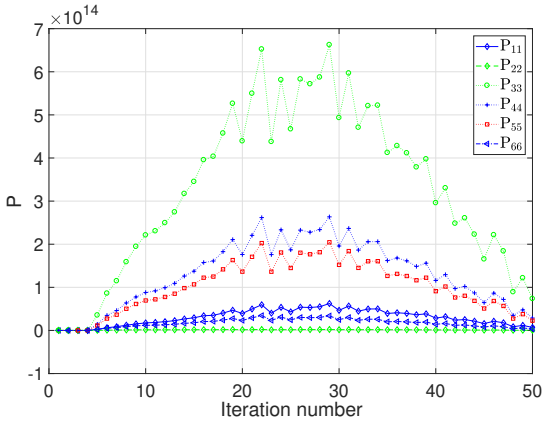

Fig. 9 Parameters of the kernel matrix P with Eq. (129) as exploration signal (Algo. 7).

In contrast with the previous implementations, the values of the P matrix do not converge to a fix value with the iterations, but constantly adapt to the different noise instances. The RMSE over a 100 trials equals 0.1492. When compared to the LADP OPFB without noise, the RMSE is 1.82 times smaller. However, in this case, the RMSE does not show which implementation has a better performance given that the scenario with noise was simplified, namely only one signal has to be followed and there is no initial deviation in the angle of attack. In Fig. 10, the output signal of the system and the reference for the last iteration are shown. The first iteration, as in the previous cases, is very similar and shows a diverging behaviour.

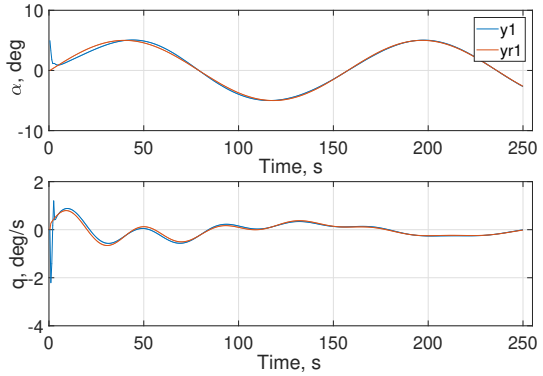

Fig. 10 Evolution of the reference and system outputs in the fiftieth iteration with Eq. (129) as exploration signal (Algo. 7).

Unfortunately, running the algorithm with more than 50 iterations, which is the number used for the optimisation, causes the algorithm to diverge. The evolution of the P matrix for 70 iterations can be observed in Fig. 11 and the evolution of the states in the last iteration is shown in Fig. 12. This means that the optimisation parameters have been overfitted to the number of iterations. The use of different numbers of iterations during the optimisation process with the aim of obtaining robust results to modifications to this parameter did not lead to the convergence of the optimisation

to a feasible solution. Although robustness is desired, in principle there is no problem if the system is trained with the number of iterations used during the optimisation. However, similar diverging behaviour was observed when altering other simulation parameters, such as the reference signal.

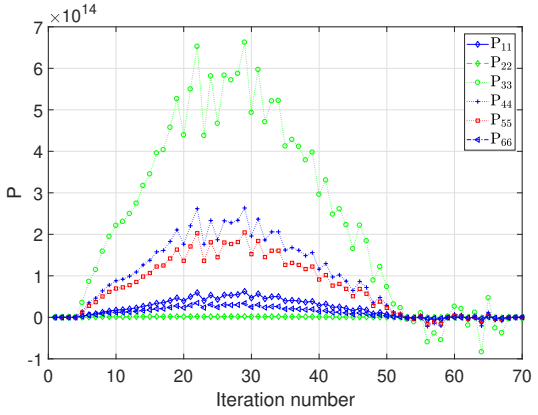

Fig. 11 Parameters of the kernel matrix P with Eq. (129) as exploration signal (Algo. 7).

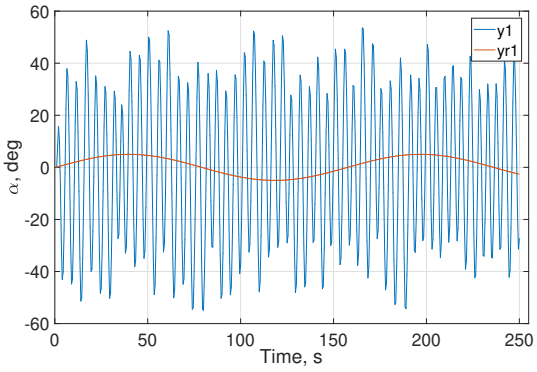

Fig. 12 Evolution of the reference and system outputs in the seventieth iteration with Eq. (129) as exploration signal (Algo. 7).

Given that the LADP OPFB with noise implementation can only follow the angle of attack with no initial deviation, with a low white noise variance and it is not robust to changes in the number of iterations, it is considered that the LADP OPFB algorithm alone can not be used in the presence of output white noise.

E. IADP with FS without sensor noise (Algo. 14)

The following implementation makes use of the parameter initialisation shown in Table 5, including in ***bold italics*** the optimised values with PSO. As can be observed in Eq. (96), the change of the states between their value in the current time step and the previous one is required (Δx_t), as well as the previous input (u_{t-1}). Both were initialised to 0 vectors. However, if the P matrix is initialised to the zero matrix, it is recommended to initialise Δx_t and u_{t-1} to random small values. It was observed that $\Delta x_0 = 0.5 \cdot \mathrm{k}_x + 0.6$ and $u_{-1} = 0.03 \cdot \mathrm{k}_u + 0.03$, where $\mathrm{k}_x \in R^{n\times 1}$ and $\mathrm{k}_u \in R^{m\times 1}$ correspond to vectors with uniformly distributed random numbers between 0 and 1, were values that lead to the convergence of the implementation.

Table 5 Parameter initialisation and hyper-parameter tuning (Algo. 14).

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$3.17 \cdot 10^7$	$\boldsymbol{R}$	10^{-6}	$\boldsymbol{\gamma}$	0.51
P_0	$\mathrm{I}_{2\times 2}$	Iterations	50	Time steps	500
x_0	$[0{,}5°,0,0]^T$	Δx_0	[0;0;0;0]	u_{-1}	0

In contrast with the previous implementation, LADP based on OPFB with noise, the algorithm tracks the angle of attack and pitch rate signals. As can be observed, the angle of attack counts with an initial deviation of 5°. Additionally, in Table 6 it is shown the initialisation of the parameters corresponding to the online RLS system identification.

Table 6 RLS parameter initialisation (Algo. 14).

Param.	Value	Param.	Value	Param.	Value
F_0	$\mathrm{I}_{4\times 4}$	G_0	[0;0;0;0]	Cov_0	$1000 \cdot \mathrm{I}_{5\times 5}$
		γ_m^{RLS}	0.8		

Furthermore, the probing signal described in Eq. (129) was used in order to explore the system; probing noise which

is learnt and later is counteracted by the system. As in all previous implementations, the convergence of the P matrix can be observed in a graph of its diagonal values with respect to the iteration number, as shown in Fig. 13. For the current implementation, this plot shows that after 5 iterations the P matrix is very close to its final value and after 20 iterations the oscillatory behaviour of P_{22} has ceased.

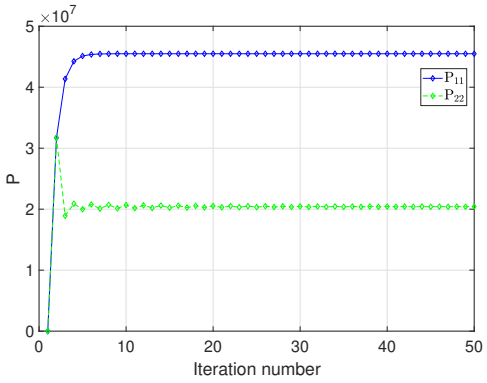

Fig. 13 Parameters of the kernel matrix P with Eq. (129) as exploration signal (Algo. 14).

In contrast with all previous implementations, IADP with FS is the first algorithm that requires an online system identification method. Given that the linearised F-16 model has been used, it is desired that with the increasing time steps the F and G matrix converge to the values of the A and B matrices. In Fig. 14 and Fig. 15 it can be observed the values of the F-A and G-B matrices with respect to the time steps in the fiftieth iteration. The values of the A and B matrices are shown with dotted horizontal lines without a marker and with the same colour as their corresponding F and G values, as can be observed in Fig. 16; an augmented figure of plot Fig. 15. It can be seen that all the values of the G matrix start at 0, as they were initialised. Around the 4^{th} time step, the F and G matrices have converged to the correct A and B matrix values.

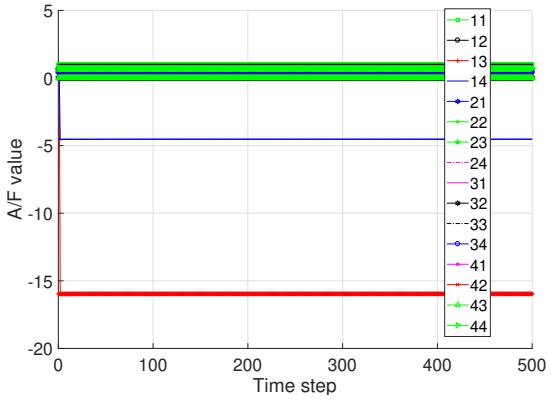

Fig. 14 Evolution of the components of the F and A matrices with the number of time steps in the fiftieth iteration (Algo. 14).

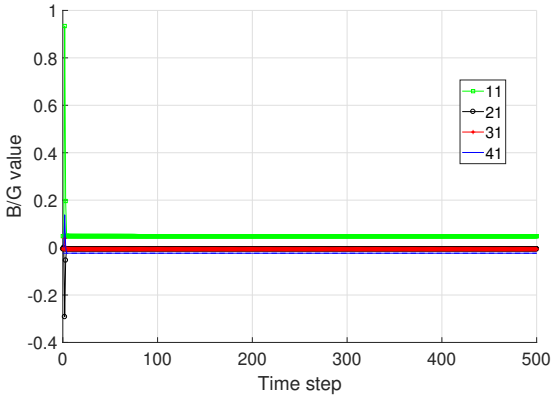

Fig. 15 Evolution of the components of the G and B matrices with the number of time steps in the fiftieth iteration (Algo. 14).

As in Section IV.A, the mean RMSE over 100 trials is computed, leading to a value of 0.0043. When compared to the LADP with FS without noise, the RMSE is exactly the same, showing that the RLS convergence time does not have a considerable effect on the algorithm's performance. In Fig. 17, the reference and the system states for the last iteration are shown. Additionally, in Fig. 18 it can be seen the elevator deflection (longitudinal input) of the F-16 which does not surpass the deflection and deflection rate limits presented in Section IV.

Finally, Fig. 19 shows the elevator deflection of LADP with FS and IADP with FS, both without noise. Both inputs are very similar in shape and with time they come closer to each other. Besides that, the greatest difference between both signals takes place during the convergence of the RLS of the IADP, namely the first 2 seconds.

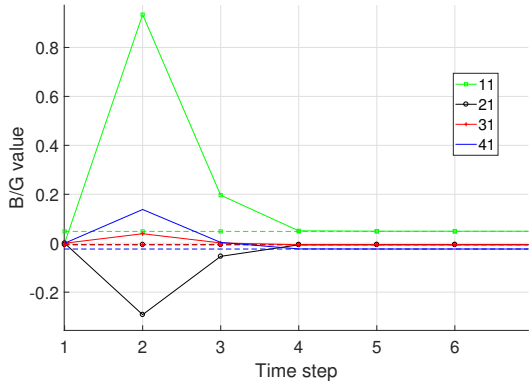

Fig. 16 Zoom in of the evolution of the components of the G and B matrices with the number of time steps in the fiftieth iteration (Algo. 14).

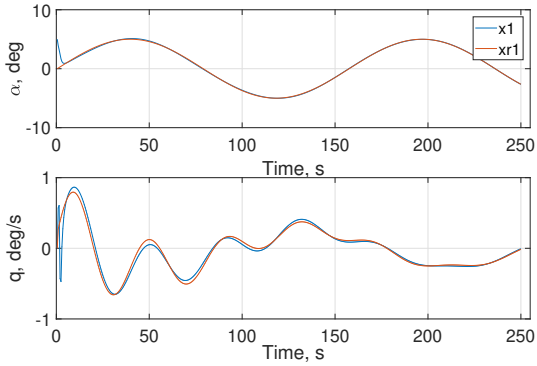

Fig. 17 Evolution of the reference and system states in the fiftieth iteration with Eq. (129) as exploration signal (Algo. 14).

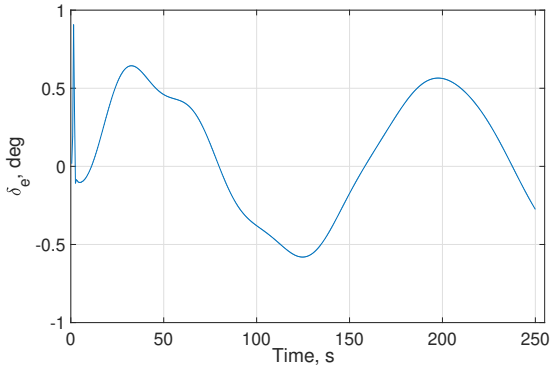

Fig. 18 Elevator deflection in the fiftieth iteration (Algo. 14).

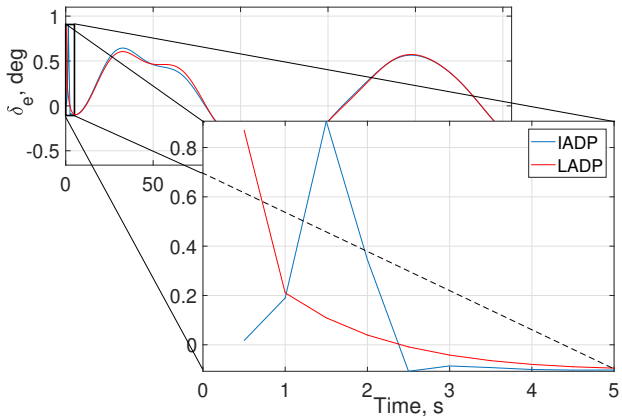

Fig. 19 Elevator deflection in the fiftieth iteration for LADP and IADP with FS and without noise, including zoom in to the first 5 seconds (Algo. 14).

F. IADP with FS with sensor noise (Algo. 13)

As in Section IV.B, thanks to the full state feedback, the system does not have to track the reference signals with its outputs but with its states. As a result, the presence of noise in the system's output signals can not influence the results, leading to the same performance as described in Section IV.E.

G. IADP based on OPFB without sensor noise (Algo. 16)

The parameter initialisation for this implementation can be observed in Table 7, including in ***bold italics*** the optimised values with PSO. As can be observed in Eq. (117), the input to the system at the previous time step (u_{t-1}), the N-2 previous changes of inputs in the time frame [t-N+1, t-1] ($\overline{\Delta u}_{t-1,t-N+1}$) and the N-1 previous changes of output errors ($\overline{\Delta e}_{0,-N+1}$) are required. All of them were initialised to 0 vectors. However, if the P matrix is initialised to the zero matrix, it is recommended to initialise these parameters to random small values. It was observed that $u_{-1} = k_u \cdot 0.03 + 1$, $\overline{\Delta u}_{-1,-N+1} = k_{\Delta u} \cdot 0.5$ and $\overline{\Delta e}_{t,-N+1} = k_e \cdot 0.7$; where $k_u \in R^{m\times 1}$, $k_{\Delta u} \in R^{m\cdot(N-1)\times 1}$ and $k_e \in R^{p\cdot N\times 1}$ correspond to uniformly distributed random numbers between 0 and 1; were values that lead to the convergence of the implementation. Additionally, Table 8 shows the initialisation of the parameters corresponding to the online RLS system identification.

Table 7 Parameter initialisation and hyper-parameter tuning (Algo. 16).

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$7.32\cdot10^{7}$	$\boldsymbol{R}$	$1\cdot10^{-6}$	$\boldsymbol{\gamma}$	$1\cdot10^{-6}$
$\boldsymbol{N}$	4	P_0	$I_{2\times2}$	Iterations	50
Time steps	500	x_0	$[0{,}5°,0,0]^T$	u_{-1}	0
$\overline{\Delta u}_{-1,-N+1}$	$0_{3,1}$	$\overline{\Delta e}_{0,-N+1}$	$0_{8,1}$		

Table 8 RLS parameter initialisation IADP based on OPFB without sensor noise (Algo. 16).

Param.	Value	Param.	Value	Param.	Value
F_0	$I_{2\times8}$	G_0	$0_{2,4}$	Cov_0	$1000 \cdot I_{12\times12}$
		γ_m^{RLS}	0.8		

As in the previous implementation (subsection IV.E), the algorithm tracks the pitch rate and the angle of attack with an initial deviation of 5°. Furthermore, the exploration signal described in Eq. (129) was used in order to explore the system; probing noise which is learnt and later is counteracted by the system. As in all previous implementations, the convergence of the P matrix can be observed in a graph of its diagonal values with respect to the iteration number, as shown in Fig. 13. For the current implementation, this plot shows that after 3 iterations the P matrix has already converged.

The results of the online system identification can be observed in Fig. 21 and Fig. 22. It can be seen that the system identification does not converge to a fixed value and it shows an irregular behaviour with multiple spikes. These observations can be attributed to the parameter γ_m^{RLS} in the system identification. In RLS, this parameter is known as the forgetting factor with a range of [0,1] and the lower its value, the higher the weight given to the latest data points fed to the online system identification algorithm. Since γ_m^{RLS} does not have a value of 1 in this analysis, then, with every time step, the latest data point has a higher weight in the identification of the $\underline{\mathcal{F}}_t$ and $\underline{\mathcal{G}}_t$ matrices. As a result, if the input or error gradients of the latest data point are very different when compared to the previous ones, then the SI signals will show an irregular behaviour until they converge again.

If a value of 0.99 would be used for γ_m^{RLS}, then the SI shown in Fig. 23 and Fig. 24 would be observed. In contrast with the lower γ_m^{RLS} value, the SI signals do not show an irregular pattern or spikes and they have much slower dynamics. At the 32^{th} time step, all the SI signals stop showing an oscillatory behaviour. A more clear explanation of the influence of γ_m^{RLS} is presented in the Appendix.

The mean RMSE over 100 trials when $\gamma_m^{RLS} = 0.8$ is computed, leading to a value of 6.2959. When compared to its LADP counterpart, which had an RMSE of 0.2709, the performance is 23 times worse; and when compared to IADP with FS without noise, which had an RMSE 0.0043, in order to assess the impact of the lack of direct information about the states, the performance is 1464 times worse. In Fig. 25, the reference and the system states for the last iteration

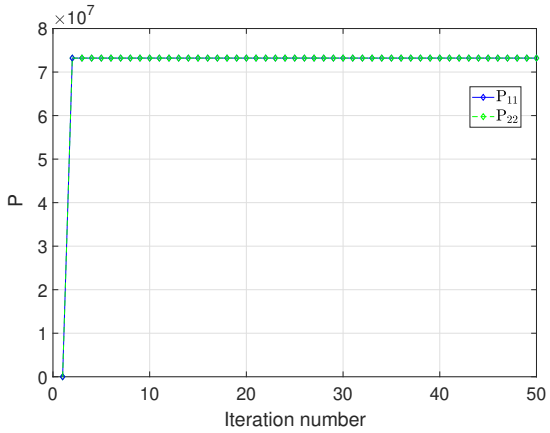

Fig. 20 Parameters of the kernel matrix P with Eq. (129) as exploration signal (Algo. 16).

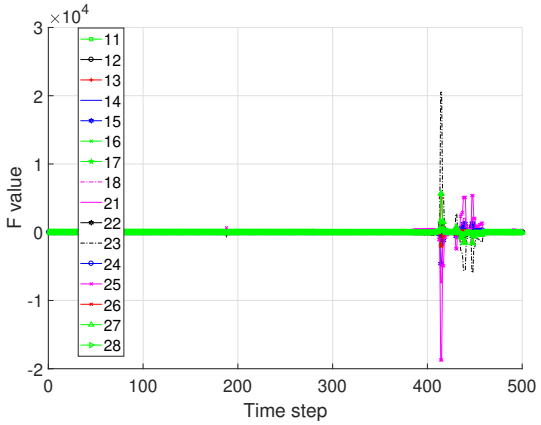

Fig. 21 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration (Algo. 16).

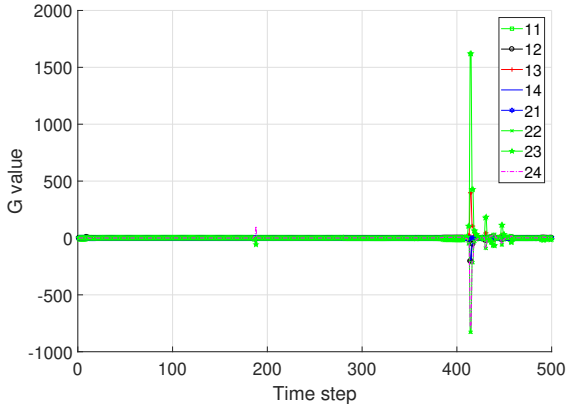

Fig. 22 Evolution of the components of the $\underline{\mathcal{G}}_t$ matrix with the number of time steps in the fiftieth iteration (Algo. 16).

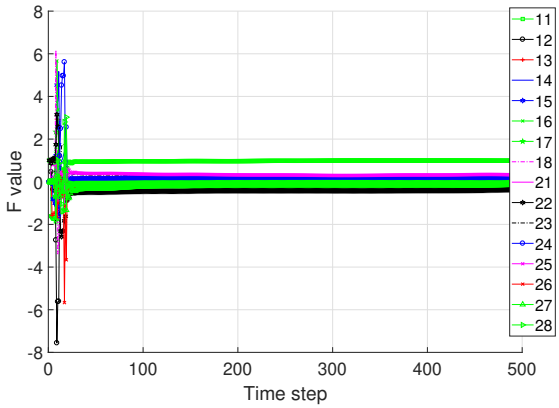

Fig. 23 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration $\gamma_m^{\text{RLS}} = 0.99$ (Algo. 16).

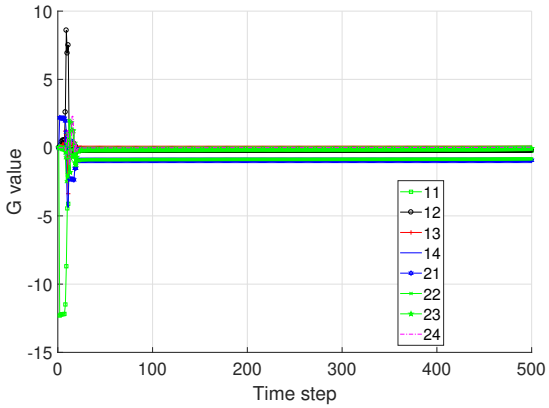

Fig. 24 Evolution of the components of the $\underline{\mathcal{G}}_t$ matrix with the number of time steps in the fiftieth iteration and $\gamma_m^{\text{RLS}} = 0.99$ (Algo. 16).

are shown. Additionally, the elevator deflection in the last iteration is shown in Fig. 26. It can be observed that the input signal reaches the 25° absolute magnitude limit multiple times in the first 10 seconds and it ceases its oscillatory behaviour at the 16^{th} second. Given the 2 Hz sampling frequency, this coincides with the 32^{nd} time step at which the system identification shown in Fig. 23 and Fig. 24 stops oscillating, even though $\gamma_m^{\text{RLS}} = 0.99$ was used. This shows that the system identification convergence time is independent of γ_m^{RLS} and that the irregular behaviour in later time steps presented in Fig. 21 and Fig. 22 does not mean that the SI has not converged.

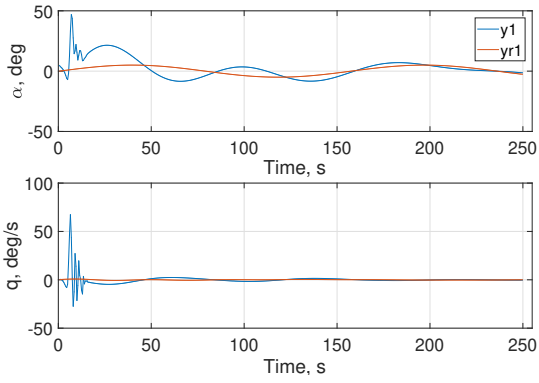

Fig. 25 Evolution of the reference and system states in the fiftieth iteration with Eq. (129) as exploration signal (Algo. 16).

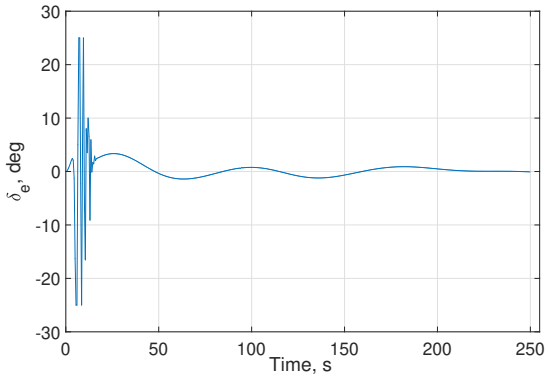

Fig. 26 Elevator deflection in the fiftieth iteration (Algo. 16).

H. IADP based on OPFB with sensor noise (Algo. 15)

The parameter initialisation for this implementation can be observed in Table 9, including in ***bold italics*** the optimised values with PSO. The system identification initialisation can be found in Table 10. The meaning of the parameters u_{t-1}, $\overline{\Delta u}_{t-1,t-N+1}$ and $\overline{\Delta e}_{0,-N+1}$ is explained in subsection IV.G, as well as what value they should obtain when the P matrix is zero, which is still applicable in the presence of noise.

Table 9 Parameter initialisation and hyper-parameter tuning IADP based on OPFB with sensor noise (Algo. 15).

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$2.03{\cdot}10^{7}$	$\boldsymbol{R}$	$1{\cdot}10^{-6}$	$\boldsymbol{\gamma}$	$8.85{\cdot}10^{-2}$
$\boldsymbol{N}$	3	P_0	$I_{2\times2}$	Iterations	50
Time steps	500	x_0	$[0{,}5°,0,0]^T$	u_{-1}	0
$\overline{\Delta u}_{-1,-N+1}$	$0_{2,1}$	$\overline{\Delta e}_{0,-N+1}$	$0_{6,1}$	σ^2_{white}	10^{-5}

Table 10 RLS parameter initialisation IADP based on OPFB with sensor noise (Algo. 15).

Param.	Value	Param.	Value	Param.	Value
F_0	$I_{2\times6}$	G_0	$0_{2,3}$	Cov_0	$1000 \cdot I_{9\times9}$
		γ_{em}^{RLS}	0.8		

The algorithm is given the task of tracking the pitch rate and the angle of attack with an initial deviation of 5° and an output noise with a variance of 10^{-5}. Furthermore, the exploration signal described in Eq. (129) was used in order to explore the system; probing noise which is learnt and later is counteracted by the system. As in all previous implementations, the convergence of the P matrix can be observed in a graph of its diagonal values with respect to

the iteration number, as shown in Fig. 27. For the current implementation, this plot shows that the P matrix does not converge. In order to verify that this is due to the constant adaptation of the system to the changing output noise signal, the system was simulated with an output noise signal that was the same between iterations. The result can be observed in Fig. 28, where the P matrix values converge after 5 iterations; the system learns a fixed policy capable of counteracting the output noise signal.

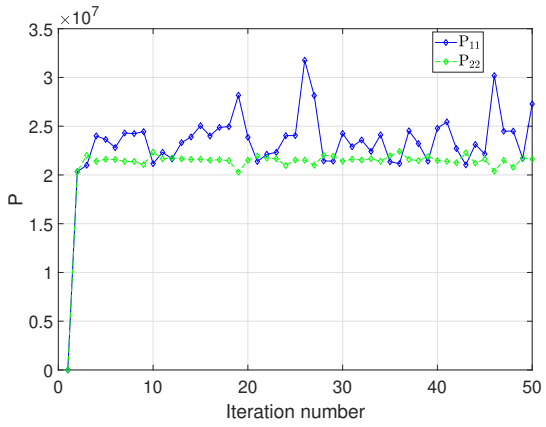

Fig. 27 Parameters of the kernel matrix P with Eq. (129) as exploration signal (Algo. 15).

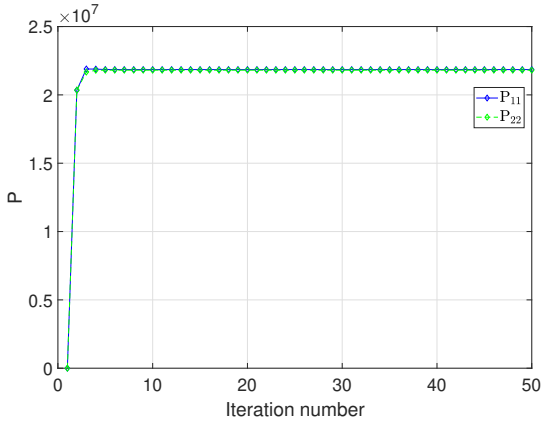

Fig. 28 Parameters of the kernel matrix P with Eq. (129) as exploration signal and constant output noise (Algo. 15).

The results of the system identification can be observed in Fig. 29 and Fig. 30 for the last iteration. It can be seen that the system identification does not converge to a fix value but it shows an irregular pattern. As explained in subsection IV.G, the forgetting factor can be increased in order to obtain a smoother SI. In contrast with the implementation without output noise, there are not large spikes.

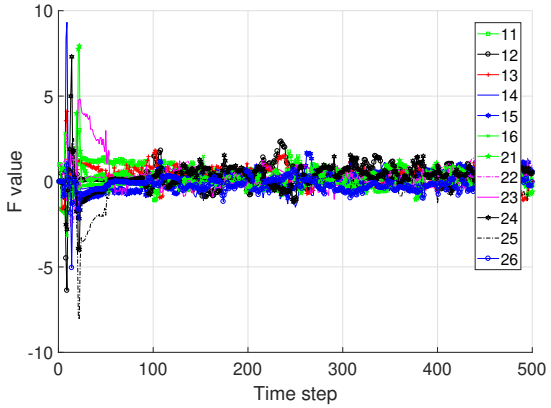

Fig. 29 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration (Algo. 15).

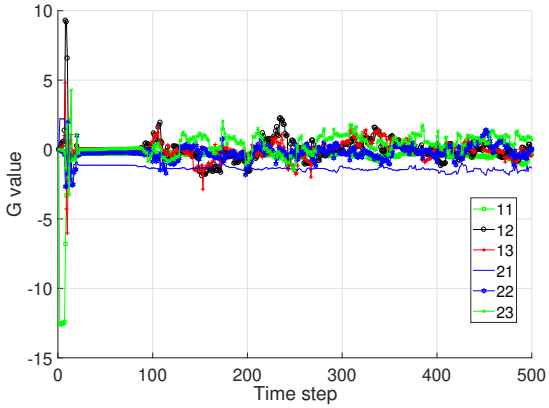

Fig. 30 Evolution of the components of the $\underline{\mathcal{G}}_t$ matrix with the number of time steps in the fiftieth iteration (Algo. 15).

The mean RMSE over 100 trials is computed, leading to a value of 3.2559. This result is better than the implementation optimised without noise; the RMSE is 1.93 times higher when there is no output noise present. This might be due to the output noise incentivizing the exploration of the solution space.

Although the P matrix does not converge, the performance of all the iterations is very similar. This can be observed in Fig. 31 and Fig. 32 which show the 43^{nd} and 50^{th} iteration reference and system states. The forty-third iteration was chosen since in Fig. 27 it can be seen that the P matrix values are very different compared to the last iteration. Furthermore, Fig. 33 shows the elevator input signal whose amplitude is saturated multiple times in the first 10 seconds.

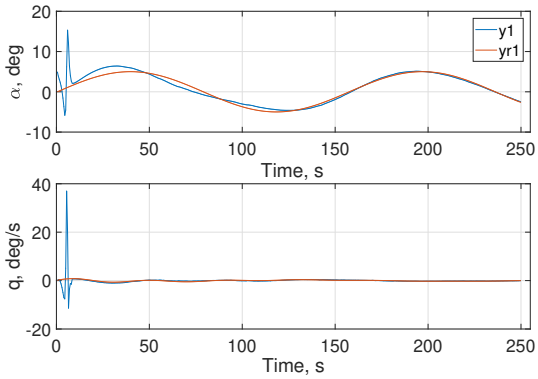

Fig. 31 Evolution of the reference and system states in the forty-third iteration with Eq. (129) as exploration signal (Algo. 15).

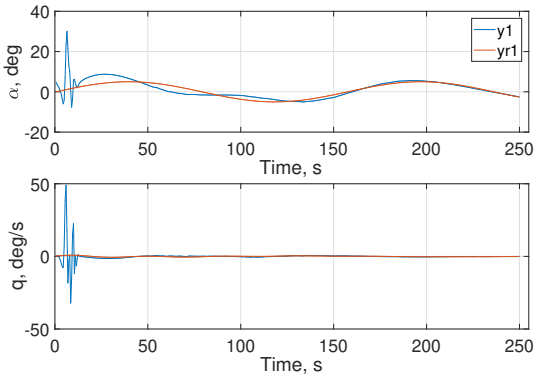

Fig. 32 Evolution of the reference and system states in the fiftieth iteration with Eq. (129) as exploration signal (Algo. 15).

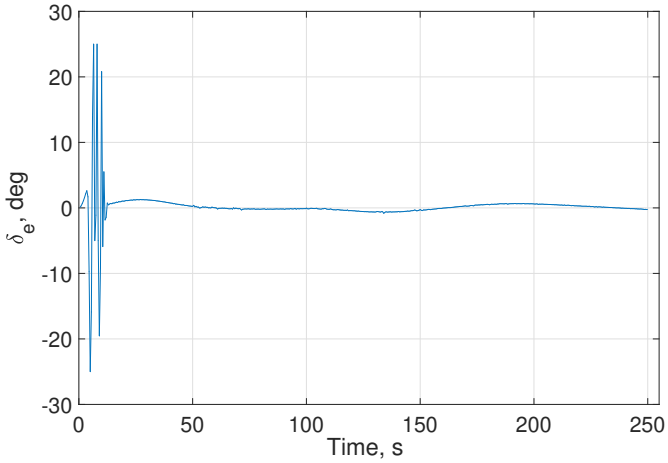

Fig. 33 Elevator deflection in the fiftieth iteration (Algo. 15).

V. Conclusion

In this paper, the LADP and IADP algorithms were discussed and implemented on a linearised F-16 model while assessing their policy convergence, online system identification, performance and control surface deflection. In order to objectively compare the results and avoid human subjective parameter tuning, which could favour some algorithms over others, Particle Swarm Optimisation (PSO) was exploited. Table 11 contains the final RMSE for the different implementations with (YES) and without (NO) output noise.

Table 11 RMSE for LADP and IADP implementations on F-16 longitudinal model.

	FS-NO	**FS-YES**	**OPFB-NO**	**OPFB-YES**
LADP	0.0043	0.0043	0.2709	*Divergence*
IADP	0.0043	0.0043	6.2959	3.2559

From Table 11 it can be observed that the LADP and IADP with FS and no output noise lead to the same performance (0.0043), thanks to the fast convergence of the IADP system identification. The main difference between the control surface deflections of both algorithms takes place during those initial time steps that the system identification of IADP

requires to converge, as it could be seen in Fig. 19. During that time, the behaviour of the system is highly dependant on the Persistent Excitation. This highlights the importance of carefully choosing the properties of those signals such that they do not cause large deviations at the beginning and that, at the same time, are capable of exciting the system in order to achieve quick convergence of the SI. Once the SI has converged, IADP corrects the error observed during the initial seconds and quickly tracks the reference.

Although both algorithms show the same performance when full state feedback is available, choosing one over the other depends on the application. LADP is a simpler and faster implementation only for linear systems, whereas IADP can handle non-linear systems and is more robust; it does not need any information of the F-16 model.

The FS implementations with output noise are trivial cases since the states are available and it is not required to rely on the contaminated output. This leads to the same performance as when there is no noise present.

When no state feedback is available, the performance of IADP is more affected than the performance of LADP. In IADP based on OPFB, the SI does not fully converge because it shows multiple spikes throughout the iteration. This is attributed to the influence of the forgetting factor in RLS which provides a higher weight to the latest data points. Besides that, IADP based on OPFB is able to recover from actuator saturation, as it was shown in Fig. 26 and Fig. 33. Although the elevator reaches the absolute deflection limit of 25 degrees, once the SI has converged it is able to quickly bring the input magnitude and change rate to lower levels.

LADP based on OPFB is not suitable in the presence of output noise since there does not exist a hyper-parameter combination that prevents divergence. Even though the implementation was also tested in a simplified scenario, the policy matrix did not converge. The good tracking performance shown in the last iteration (Fig. 10) could be seen as a mirage, since the result is very sensitive to the simulation parameters, such as the number of iterations or the reference signal.

In contrast to LADP, IADP based on OPFB converges in the presence of output noise, showing almost twice as good performance than when the output noise is removed. This tracking improvement is attributed to a better exploration of the solution space. Even though the policy matrix does not show convergence, it is constantly adapting to the changing output noise signals such that similar performance is observed throughout the iterations. This was validated by making the output noise constant between iterations, which led to a converging policy matrix.

The next steps of this work would be to apply different filters to mitigate the negative effects of the output noise, potentially leading to the convergence and performance improvement of the algorithms. This would result in a trade-off between the negative effects of output noise and the delays introduced by the filters. Also, it is necessary to assess their behaviour when there is bias in the output noise and in the presence of atmospheric disturbances.

Finally, LADP and IADP can provide a boost to the concept of Smart Cities by bringing a model-free adaptive flight control that ensures safety in the future crowded urban airspace, populated by (autonomous) urban air mobility and UAVs. In particular, this paper has shown the ability of IADP to withstand the presence of output noise when implemented in the longitudinal (SIMO, fast dynamics) aircraft model without prior information of the system and with Output Feedback (OPFB); no state information available. The current and mentioned further research are the building blocks for a better understanding of the behaviour of these control algorithms in outdoor environments with non-ideal sensors, contributing to a more informed decision when having to implement them in future flying vehicles.

Appendix

In the following sections it is assessed the impact of modifying some implementation parameters in the performance and the system identification of the IADP based on OPFB with and without noise. In order to assess better the impact of the different parameters, in the reference scenario the algorithm will track only the angle of attack with no deviation during 1300 time steps and 50 iterations, except if mentioned otherwise. The corresponding initialisation and PSO optimised parameters are shown in Table 12, whereas the system identification initialisation can be found in Table 13.

Table 12 Parameter initialisation and hyper-parameter tuning IADP based on OPFB without sensor noise.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$2.92{\cdot}10^{7}$	$\boldsymbol{R}$	$1{\cdot}10^{-6}$	$\boldsymbol{\gamma}$	0.833
$\boldsymbol{N}$	3	P_0	$I_{1\times1}$	Iterations	50
Time steps	1300	x_0	$[0,0,0,0]^T$	u_{-1}	0
$\overline{\Delta u}_{-1,-N+1}$	$0_{2,1}$	$\overline{\Delta e}_{0,-N+1}$	$0_{3,1}$		

Table 13 RLS parameter initialisation IADP based on OPFB without sensor noise.

Param.	Value	Param.	Value	Param.	Value
F_0	$I_{1\times3}$	G_0	$0_{1,3}$	Cov_0	$1000 \cdot I_{6\times6}$
		γ_m^{RLS}	0.8		

A. Impact of the SI forgetting factor (γ_m^{RLS})

In RLS, γ_m^{RLS} is known as the forgetting factor with a range of [0,1] and the lower its value, the higher the weight given to the latest data points fed to the online system identification algorithm. When γ_m^{RLS} does not have a value of 1, with every time step the latest data point has a higher weight in the identification of the $\underline{\mathcal{F}}_t$ and $\underline{\mathcal{G}}_t$ matrices. As a result, if the input or error gradients of the latest data point are very different when compared to the previous ones, then the SI signals will show an irregular behaviour until convergence is shown again.

In Fig. 34, Fig. 35, Fig. 36 and Fig. 37 it can be observed the identification of the $\underline{\mathcal{F}}_t$ matrix for 3 different values of the forgetting factor, namely 0.5, 0.8, 0.95 and 0.99, respectively. In this case, the angle of attack tracking scenario defined by the parameters in Table 12 and Table 13 was exploited with the original PE signal of Eq. (129). γ_m^{RLS} was the only altered value.

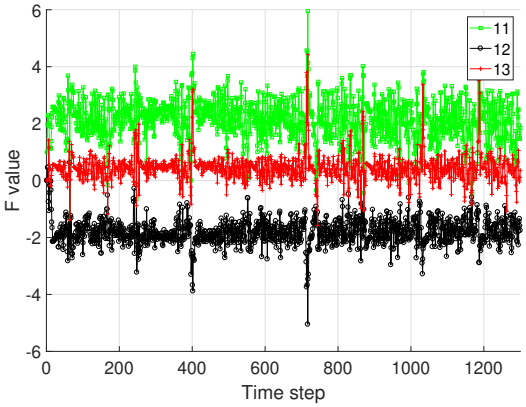

Fig. 34 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with γ_m^{RLS} = 0.5.

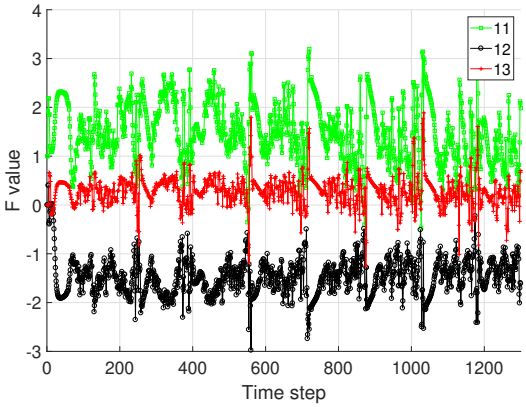

Fig. 35 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with γ_m^{RLS} = 0.8.

From these figures it can be seen that SI smoothens with increasing forgetting factor and the larger γ_m^{RLS}, the longer it takes to converge. Besides that, the system identification takes longer to converge in the presence of an initial deviation. This can be observed comparing Fig. 37 with Fig. 38, which has an initial deviation in the angle of attack of 5°.

In terms of performance, plotting in Fig. 39 the RMSE with respect to the forgetting factor for different initial deviation values and normalising it such that the sum of RMSE for a specific deviation is not greater than 1, we observe that the RMSE is not constantly decreasing or increasing but it shows a different irregular pattern for each initial deviation. If the normalised RMSE's for all the initial deviations are added and the result is normalised again (red curve), we observe a much flatter normalised RMSE-curve. This shows that the overall performance is not strongly dependant on the forgetting factor choice. Different forgetting factors will lead to the best performance with different initial deviations; however, when the different initial deviation curves are averaged, all the forgetting factors show similar performance.

Although all the forgetting factors show a similar performance, it can be observed that the lowest overall normalised RMSE values can be found with high values of γ_m^{RLS}, namely in the range [0.86, 0.96]. It must be borne in mind that the hyper-parameters were PSO tuned with a forgetting factor equal to 0.8.

Finally, this RMSE performance analysis is carried out on the same system tracking the angle of attack and pitch rate reference signals with the initialisation and hyper parameters shown in Table 14, as well as the RLS parameter initialisation shown in Table 15.

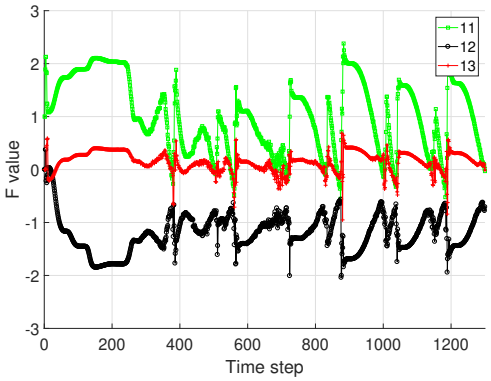

Fig. 36 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with $\gamma_m^{\text{RLS}} = 0.95$.

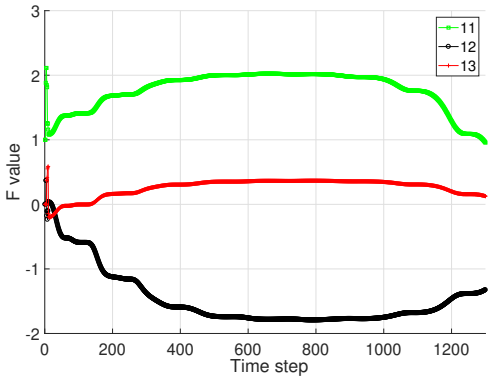

Fig. 37 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with $\gamma_m^{\text{RLS}} = 0.99$.

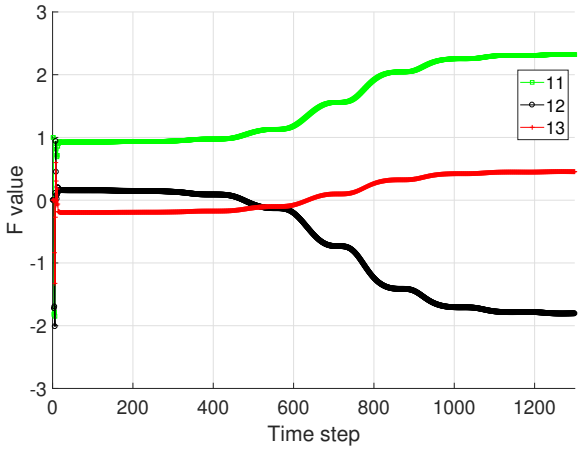

Fig. 38 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with $\gamma_m^{\text{RLS}} = 0.99$ and an initial angle of attack deviation of 5°.

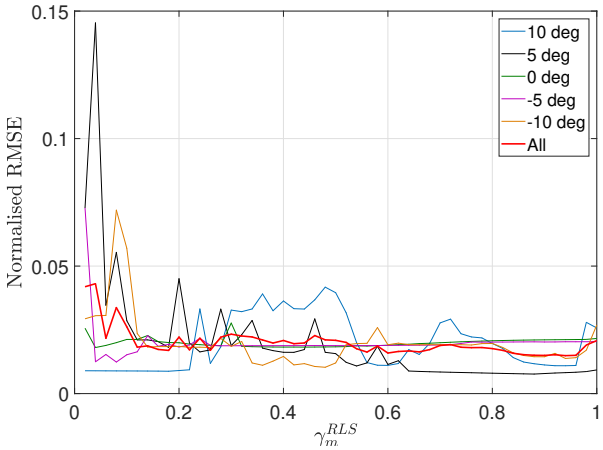

Fig. 39 Normalised RMSE with respect to γ_m^{RLS} for different initial deviations and overall normalised RMSE.

Table 14 Parameter initialisation and hyper-parameter tuning IADP based on OPFB without sensor noise tracking α and q.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$7.32{\cdot}10^{7}$	$\boldsymbol{R}$	$1{\cdot}10^{-6}$	$\boldsymbol{\gamma}$	$1{\cdot}10^{-6}$
$\boldsymbol{N}$	4	P_0	$I_{2\times2}$	Iterations	50
Time steps	1300	x_0	$[0,0,0,0]^T$	u_{-1}	0
$\overline{\Delta u}_{-1,-N+1}$	$0_{3,1}$	$\overline{\Delta e}_{0,-N+1}$	$0_{8,1}$		

Table 15 RLS parameter initialisation IADP based on OPFB without sensor noise tracking α and q.

Param.	Value	Param.	Value	Param.	Value
F_0	$I_{2\times8}$	G_0	$0_{2,4}$	Cov_0	$1000 \cdot I_{12\times12}$
		γ_m^{RLS}	0.8		

The resulting RMSE performance plot is presented in Fig. 40. In contrast with tracking only the angle of attack, it can be observed that the performance increases with increasing forgetting factor when tracking multiple signals. After $\gamma_m^{\text{RLS}} = 0.7$, the normalised RMSE is lower than 10^{-26}. This shows that the more complicated the task, the more the system relies on previous input-output information in order to correctly identify the system, instead of blindly trusting more the latest data points. A forgetting factor lower than 0.7 will lead to the controller not able to identify the system and diverge. Again, it must be borne in mind that the hyper-parameters were PSO optimised with a forgetting factor of 0.8.

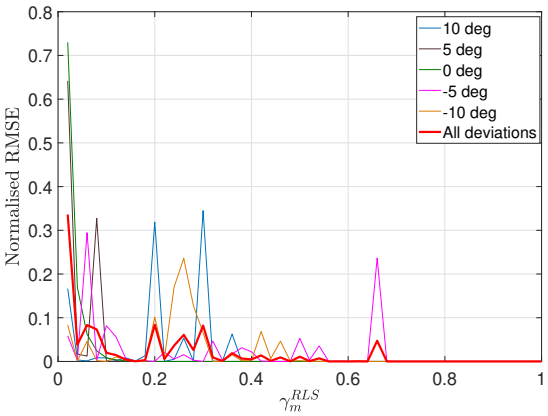

Fig. 40 Normalised RMSE with respect to γ_m^{RLS} for different initial deviations and overall normalised RMSE. The system tracks α and q.

B. Impact of the Persistent Excitation signal

1. PE selection

The Persistent Excitation used in the analysis is described by Eq. (129) and it has the shape shown in Fig. 41. As can be observed, the signal spikes for the first time steps and after the 60^{th} time step its magnitude is lower than 1/100 its initial value.

As a result, the PE disappears with time and its initial value determines in part the performance of the whole iteration. In Fig. 42 and Fig. 43, the output and reference signals can be observed for the fiftieth iteration with an RMSE of 0.0146. The greatest deviation of the output from the reference signal takes place in the first 6 time steps that the signal oscillates.

However, if the signal to track is multiplied by -1, an RMSE of 0.437 is obtained with the tracking shown in Fig. 44

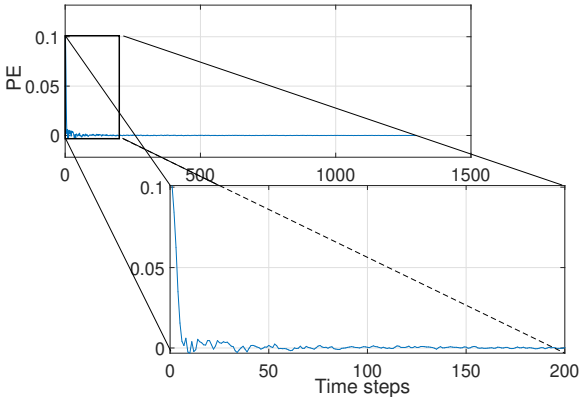

Fig. 41 Persistent Excitation signal (Eq. (129)).

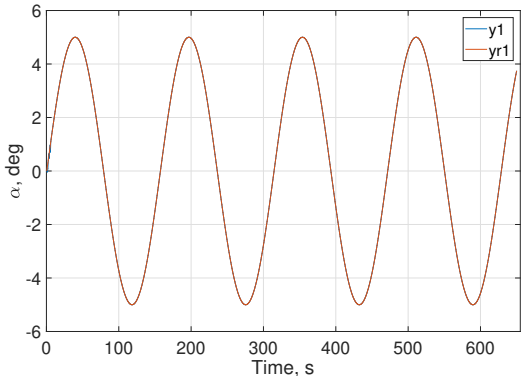

Fig. 42 Evolution of the reference and system states in the fiftieth iteration with Eq. (129) as PE.

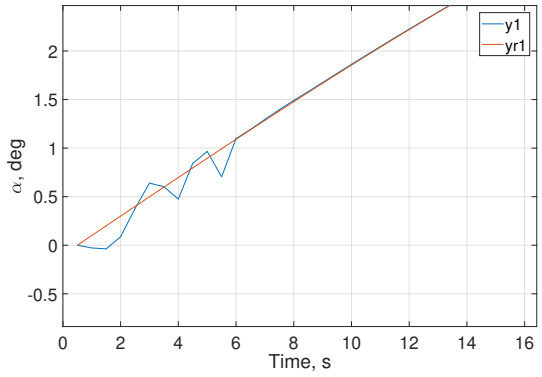

Fig. 43 Zoom in of the evolution of the reference and system states in the fiftieth iteration with Eq. (129) as PE.

and Fig. 45. This is 41 times worse performance when compared to the default tracking signal. As can be seen, there is an initial deviation in the upwards direction which the system must compensate. This behaviour is caused by the initial positive spike of the PE acting while the system identification has not converged. If the PE signal is also multiplied by -1, the tracking shown in Fig. 46 can be observed, with an RMSE of 0.0146. This performance is identical to the scenario in which neither the reference signal nor the PE are multiplied by -1. It can be concluded that the selection of the PE signal has great importance for performance. The first seconds of an iteration are strongly dominated by the PE, since the system identification is still oscillating.

2. PE magnitude

At the end of Eq. (129), we can observe that the default PE signal is divided by 10. In this section, it is assessed the impact of the magnitude of the PE signal in the system identification by modifying this factor of the PE equation. For that purpose, it will be given the following four values: 0.01, 1, 100 and 10000.
In Figs. 47, 48, 49 and 50 it is possible to observe the values of the components of the $\underline{\mathscr{F}}_t$ matrix with the number of time steps in the fiftieth iteration.

In general, two phases can be distinguished: an initial oscillatory part during which the system identification is converging and a second part during which the SI is constantly re-adapting due to the higher weight given by γ_m^{RLS} to the latest data points. With the increasing PE magnitude, the second part of the SI signals is much smoother and they show a repetitive pattern. This periodic part of the signal is mostly due to the sinusoid reference signal which must

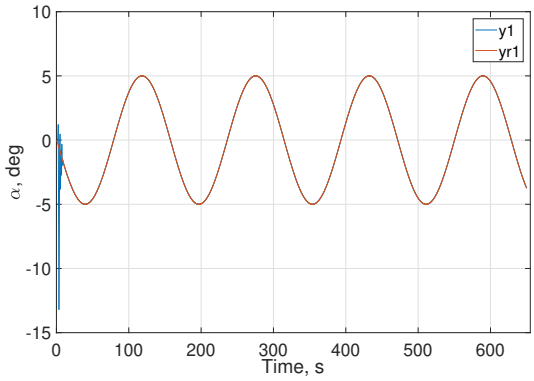

Fig. 44 Evolution of the negative reference and system states in the fiftieth iteration with Eq. (129) as PE.

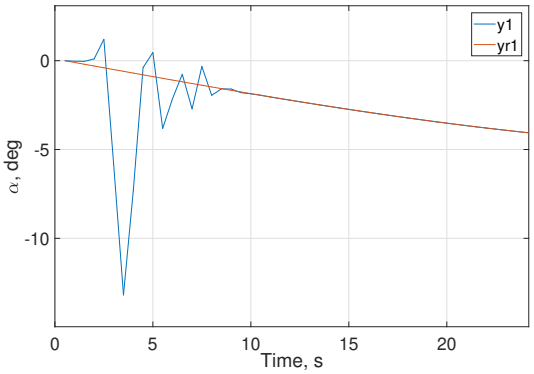

Fig. 45 Zoom in of the evolution of the negative reference and system states in the fiftieth iteration with Eq. (129) as PE.

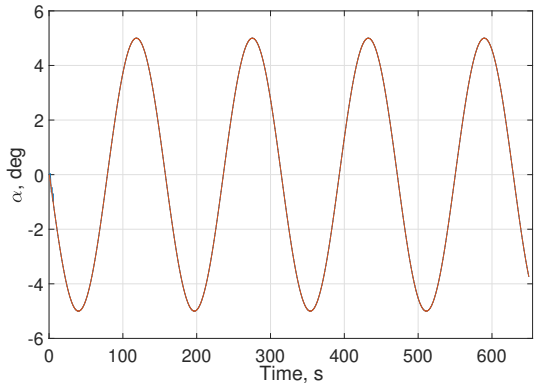

Fig. 46 Zoom in of the evolution of the negative reference and system states in the fiftieth iteration with negative Eq. (129) as PE.

be tracked. The sudden changes in the SI signals are separated by half the period of the sinusoid and all the signals of the components of the $\underline{\mathscr{F}}_t$ and $\underline{\mathscr{G}}_t$ matrices experience this sudden changes at the same time steps. In contrast, if the magnitude is decreased, more oscillations are present and it is more complicated to observe the afore-mentioned periodic behaviour. Besides that, the higher the magnitude of the PE signals the larger the magnitude of the SI signals in the initial phase.

Finally, in Table 16 it can be observed that increasing the PE amplitude, although it may improve the SI, it tends to worsen the performance when there is no deviation present (ND) and when there is a deviation, in this case of 5°. Also, the lowest RMSE values are observed for the factors 10 and 100. This was expected since the hyper-parameters were PSO optimised with a PE divided by a factor of 10.

Table 16 IADP based on OPFB without noise RMSE for different PE amplitudes in the case of no initial deviation and 5° deviation.

	0.01	**1**	**10**	**100**	**10000**
RMSE (ND)	0.8181	0.0225	0.0146	**0.0081**	0.3302
RMSE (5°)	1.3073	1.6399	**0.5436**	0.7095	0.4998

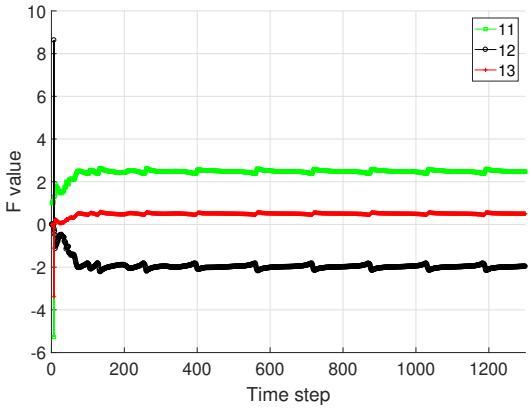

Fig. 47 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with Eq. (129) multiplied by 1000 as PE.

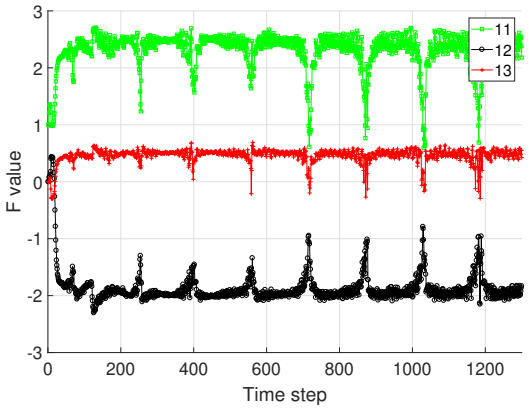

Fig. 48 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with Eq. (129) multiplied by 10 as PE.

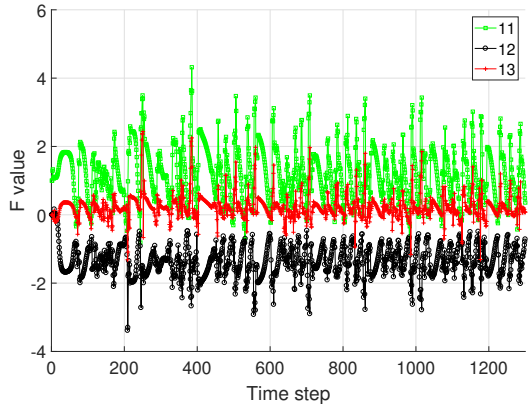

Fig. 49 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with Eq. (129) divided by 10 as PE.

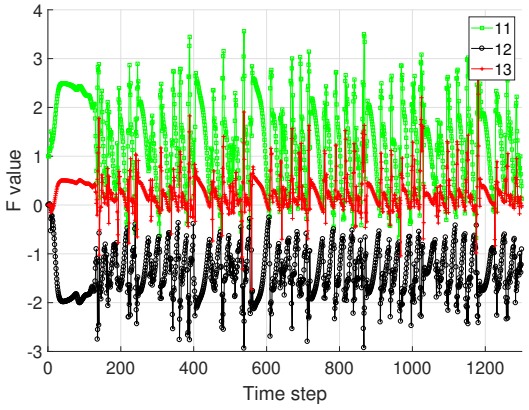

Fig. 50 Evolution of the components of the $\underline{\mathcal{F}}_t$ matrix with the number of time steps in the fiftieth iteration with Eq. (129) divided by 1000 as PE.

3. PE signal dissipation

As can be observed in Eq. (129), the PE signal is divided by t such that it has a strong effect at the beginning of the iteration and rapidly is dissipated with time. In this section it is assessed the effect of removing the t which divides the PE signal on the performance, resulting in the signal shown in Fig. 51. For this analysis, the most complete scenario is chosen, meaning that the angle of attack and pitch rate are tracked, white output noise is present and there are different initial deviations for the angle of attack.

First, the RMSE performance is assessed when tracking the angle of attack and pitch rate for different variances of output white noise and for two initial deviations: no deviation (ND) and a 5° deviation (D) in the angle of attack. For that purpose, Table 17 and Table 18 show the used PSO optimised parameters when the PE is divided by t (Y) and when the PE is not divided by t (N), respectively. The maximum variance of white output noise that the algorithm can withstand with P matrix convergence with the current parameters is 10^{-3}.

In Table 19 the RMSE errors for the different scenarios are given. On the left side are located those tests in which the PE function is not divided by t and on the right side are located those tests in which the PE function is divided by t. In red are those values in which there was no initial deviation and in blue are those values in which there was an initial deviation in the angle of attack of 5°.

As can be observed, when there is no deviation, it is best to use the PE signal that is divided by t and when there is a deviation, it is best to use the PE signal that is not divided by t. This behaviour can be seen for all the instances

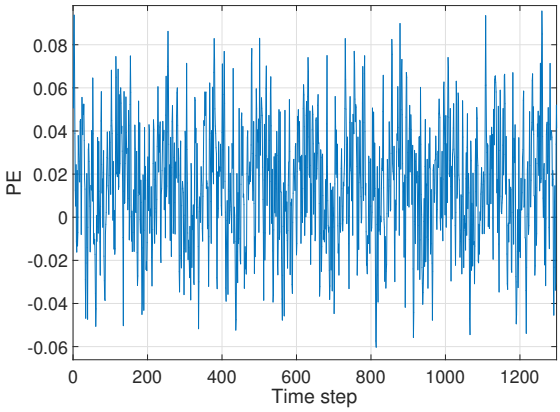

Fig. 51 Persistent Excitation signal (Eq. (129)) multiplied by t.

Table 17 Hyper-parameter tuning IADP based on OPFB with sensor noise and with Eq. (129) divided by t.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$7.32{\cdot}10^{7}$	$\boldsymbol{R}$	$1{\cdot}10^{-6}$	$\boldsymbol{\gamma}$	$1{\cdot}10^{-6}$
$\boldsymbol{N}$	4				

Table 18 Hyper-parameter tuning IADP based on OPFB without sensor noise and with Eq. (129) not divided by t.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$1{\cdot}10^{8}$	$\boldsymbol{R}$	$1{\cdot}10^{-6}$	$\boldsymbol{\gamma}$	$1{\cdot}10^{-6}$
$\boldsymbol{N}$	3				

Table 19 RMSE for the different variances of white noise, with and without an initial deviation (D, ND) of 5° and dividing or not the PE (Eq. (129)) by t (Y, N).

	ND/N	D/N	ND/Y	D/Y
No noise	**0.0772**	**1.7304**	0.3604	3.9113
$\sigma^2_{white} = 10^{-9}$	0.0772	**1.7298**	**0.0291**	3.9018
$\sigma^2_{white} = 10^{-7}$	0.0774	**1.7507**	**0.0285**	3.9635
$\sigma^2_{white} = 10^{-5}$	0.0806	**1.9656**	**0.0309**	5.1208
$\sigma^2_{white} = 10^{-3}$	0.3372	**2.7328**	**0.1439**	5.0397

with output noise. In the case of no output noise, not dividing by t provides better performance when there is no initial deviation.

Not dividing by t contributes towards exploration, that is why the results when there is a deviation are better. The results are worse when there is no deviation because no much exploration is required and the extra PE simply acts as noise that contributes to error. Since in most of the scenarios, a deviation is present (no deviation could be seen as simply one scenario of deviation with 0 degrees out of the infinite number of potential deviations), it is best to use a PE signal that is not divided by t such that there is the same input noise variance throughout the complete iteration. Most of the encountered scenarios will have an initial deviation, except if the tracking signal was designed to start with the same value as the output signal of the F-16 at the first time step.

Performing the same analysis with different initial deviations when there is no output noise present, results in Table 20 with the RMSE for the different scenarios. As can be seen, independently of the initial deviation, the performance of the scenario when the PE is not divided by t is higher.

Table 20 RMSE for the simulations with and without an initial deviation in the range of [-10°, 10°] and dividing or not the PE (Eq. (129)) by t (Y-N).

	N	Y
-10°	**3.5600**	6.3104
-5°	**1.5504**	2.4421
ND	**0.0772**	0.3604
5°	**1.7304**	3.9113
10°	**1.2726**	1.3173

C. Impact of the sampling frequency

The sampling frequency defines how many times a control input is defined per second. In theory, given that IADP consists of a local linearisation around the operating point, the higher the sampling frequency the more accurate the results. In order to verify it with the implementation, the RMSE is compared with two different sampling frequencies, namely 2 Hz and 10 Hz. In the case of a sampling frequency of 2 Hz, the initialisation and PSO parameters are shown in Table 12. In the case of a sampling frequency of 10 Hz, these values are shown in Table 21. As can be seen, the number of time steps for the higher sampling frequency has been increased such that the simulation is run for the same number of seconds as the lower sampling frequency. In both cases, only the angle of attack is tracked and the original PE signal of Eq. (129) is used.

Table 21 Parameter initialisation and hyper-parameter tuning IADP based on OPFB without sensor noise.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$1{\cdot}10^{8}$	$\boldsymbol{R}$	$8.6{\cdot}10^{6}$	γ	0.964
$\boldsymbol{N}$	4	P_0	$I_{1\times1}$	Iterations	50
Time steps	6500	u_{-1}	0	$\overline{\Delta u}_{-1,-N+1}$	$0_{3,1}$
$\overline{\Delta e}_{0,-N+1}$	$0_{4,1}$				

In Table 22 it is shown the RMSE for the different angle of attack initial deviations and sampling frequencies. As can be observed, a higher sampling frequency leads to better performance, as it was expected from theory.

Table 22 RMSE for the simulations with and without an initial deviation in the range of [-10°, 10°] and different sampling frequencies (2 Hz and 10 Hz).

	f_s = 2 Hz	f_s = 10 Hz
-10°	1.4336	**1.3012**
-5°	1.2859	**1.0583**
ND	0.0146	**0.0049**
5°	0.5436	**0.2830**
10°	1.0796	**0.6364**

D. Impact of the time horizon

The time horizon (N) determines the number of previously measured data points used for the derivation of the OPFB policy evaluation and improvement equations. In order to assess the impact of this parameter on the performance, its value is changed in the range of [3,8] for the scenario described in Table 12. In Fig. 52, it can be observed the normalised RMSE with respect to N for the different initial deviations of the angle of attack, as well as the overall performance independently of the initial deviation. Although a larger time horizon would mean more information for the computation of the policy and the input increment, Fig. 52 shows that increasing N corresponds to an increment in the RMSE. This means that the information stored in the previous 3 points is enough. The use of more previous data

points only causes a larger initial deviation since N initial data points are required in order to fill in vectors such as $\overline{\Delta e}_{t,t-N+1}$ and $\overline{\Delta u}_{t,t-N+1}$. As an example, if N equals 8 and it is the fifth time step in the simulation, those vectors will have 3 entries equal to zero, those for which there has been no information obtained yet.

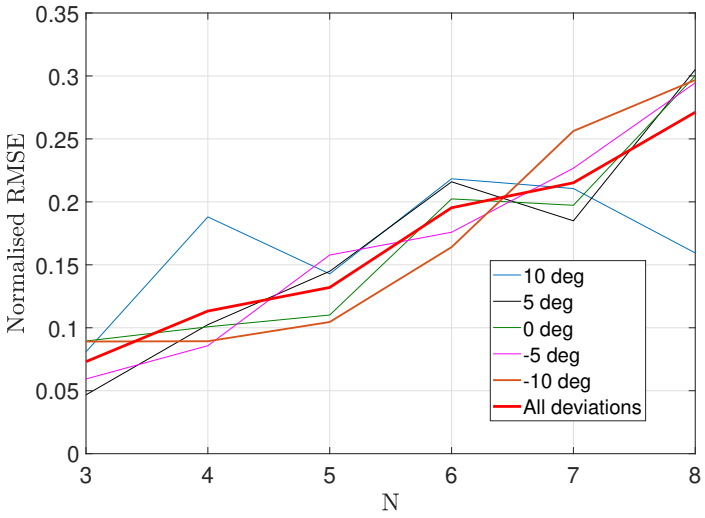

Fig. 52 Normalised RMSE with respect to N for different initial deviations and overall normalised RMSE.

The better performance could be attributed to the optimised parameters for N equal to 3. In order to challenge this premise, the PSO optimisation was carried out with the time horizon limited to the range [5,8], leading to the hyper-parameters shown in Table 23.

Table 23 Parameter initialisation and hyper-parameter tuning IADP based on OPFB without sensor noise.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$5.97 \cdot 10^7$	$\boldsymbol{R}$	$4.93 \cdot 10^7$	$\boldsymbol{\gamma}$	1
$\boldsymbol{N}$	5	P_0	$I_{1\times1}$	Iterations	50
Time steps	1300	u_{-1}	0	$\overline{\Delta u}_{-1,-N+1}$	$0_{2,1}$
$\overline{\Delta e}_{0,-N+1}$	$0_{3,1}$				

Again, the lowest time horizon possible (N=5) was found to be the best option for the PSO. However, when obtaining the RMSE performance for different time horizons in the range [3,8], it was observed that N=3 lead to the best performance even though the hyper-parameters were optimised for N=5, as it is shown in Fig. 53. Besides that, although the PSO hyper-parameters of Table 12 and Table 23 are very different, the normalised RMSE values are very similar.

Finally, when output noise is present, it is expected that the use of previous data points leads to better results since the algorithm can leverage more information to filter out the noise. In order to test this behaviour, the system was tested with the hyper-parameters shown in Table 24.

Table 24 Parameter initialisation and hyper-parameter tuning IADP based on OPFB with sensor noise.

Param.	Value	Param.	Value	Param.	Value
$\boldsymbol{Q}$	$5.06 \cdot 10^7$	$\boldsymbol{R}$	$3.65 \cdot 10^7$	$\boldsymbol{\gamma}$	0.85
$\boldsymbol{N}$	3	P_0	$I_{1\times1}$	Iterations	50
Time steps	1300	u_{-1}	0	$\overline{\Delta u}_{-1,-N+1}$	$0_{2,1}$
$\overline{\Delta e}_{0,-N+1}$	$0_{3,1}$	σ^2_{white}	10^{-5}		

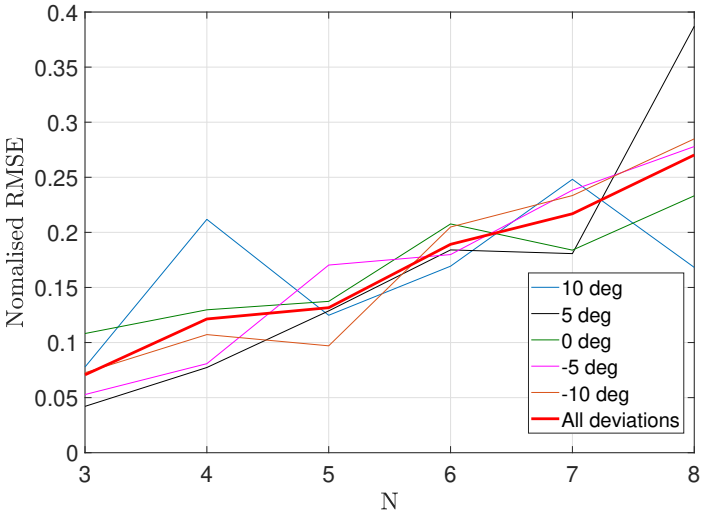

Fig. 53 Normalised RMSE with respect to N for different initial deviations and overall normalised RMSE. Hyper-parameters tuned for N equal to 5.

In contrast with Fig. 52, Fig. 54 shows that the RMSE initially decreases to a minimum at N equals 6. This means that using N previous data points in the range [3,6] contributes to reducing the effect of the output noise on performance; whereas exploiting an N higher than 6 leads to a higher initial deviation that counteracts the benefits of the reduced output noise effects. This effect is most notable when there is no initial deviation in the angle of attack. Removing the line corresponding to no initial α deviation, would lead to an almost flat overall normalised RMSE.

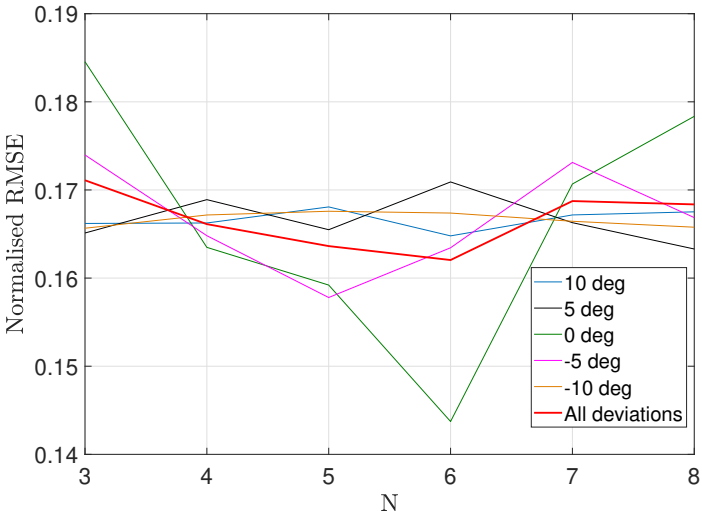

Fig. 54 Normalised RMSE with respect to N for different initial deviations and overall normalised RMSE. The system shows output noise with a variance of 10^{-5}.

As a result, it can be concluded that in the absence of output noise, the lower the time horizon, the better the performance independently of the PSO optimised time horizon. However, in the presence of output noise, the performance initially slightly improves with the time horizon or remains unaffected.

References

[1] Ioannou, P., and Fidan, B., *Adaptive Control Tutorial*, Advances in Design and Control, Society for Industrial and Applied Mathematics, SIAM, Philadelphia, 2006.

[2] Lane, S. H., and Stengel, R. F., "Flight control design using non-linear inverse dynamics," *Automatica*, Vol. 24, No. 4, 1988, pp. 471 – 483. https://doi.org/10.1016/0005-1098(88)90092-1.

[3] Acquatella, P., Briese, L. E., and Schnepper, K., "Guidance command generation and nonlinear dynamic inversion control for reusable launch vehicles," *Acta Astronautica*, Vol. 174, 2020, pp. 334 – 346. https://doi.org/10.1016/j.actaastro.2020.04.002.

[4] Bacon, B., and Ostroff, A., "Reconfigurable flight control using nonlinear dynamic inversion with a special accelerometer implementation," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000. https://doi.org/10.2514/6.2000-4565.

[5] da Costa, R. R., Chu, Q. P., and Mulder, J. A., "Reentry Flight Controller Design Using Nonlinear Dynamic Inversion," *Journal of Spacecraft and Rockets*, Vol. 40, No. 1, 2003, pp. 64–71. https://doi.org/10.2514/2.3916.

[6] Sieberling, S., Chu, Q. P., and Mulder, J. A., "Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 6, 2010, pp. 1732–1742. https://doi.org/10.2514/1.49978.

[7] Acquatella, P., Falkena, W., van Kampen, E.-J., and Chu, Q. P., "Robust Nonlinear Spacecraft Attitude Control using Incremental Nonlinear Dynamic Inversion," *AIAA Guidance, Navigation, and Control Conference*, 2012. https://doi.org/10.2514/6.2012-4623.

[8] Grondman, F., Looye, G., Kuchar, R. O., Chu, Q. P., and van Kampen, E.-J., "Design and Flight Testing of Incremental Nonlinear Dynamic Inversion-based Control Laws for a Passenger Aircraft," *2018 AIAA Guidance, Navigation, and Control Conference*, 2018. https://doi.org/10.2514/6.2018-0385.

[9] Acquatella, P., van Kampen, E.-J., and Chu, Q. P., "Incremental backstepping for robust nonlinear flight control," *EuroGNC 2013, 2nd CEAS Specialist Conference on Guidance, Navigation & Control*, 2013.

[10] Ali, A. A. H., Chu, Q. P., van Kampen, E.-J., and de Visser, C. C., "Exploring Adaptive Incremental Backstepping using Immersion and Invariance for an F-16 aircraft," *AIAA Guidance, Navigation, and Control Conference*, 2014. https://doi.org/10.2514/6.2014-0084.

[11] Wang, X., van Kampen, E.-J., Chu, Q. P., and De Breuker, R., "Flexible Aircraft Gust Load Alleviation with Incremental Nonlinear Dynamic Inversion," *Journal of Guidance, Control, and Dynamics*, 2019, pp. 1–18. https://doi.org/10.2514/1.G003980.

[12] Keijzer, T., Looye, G., Chu, Q. P., and van Kampen, E.-J., "Design and Flight Testing of Incremental Backstepping based Control Laws with Angular Accelerometer Feedback," *AIAA Scitech 2019 Forum*, 2019. https://doi.org/10.2514/6.2019-0129.

[13] Sonneveldt, L., Chu, Q. P., and Mulder, J. A., "Nonlinear Flight Control Design Using Constrained Adaptive Backstepping," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 322–336. https://doi.org/10.2514/1.25834.

[14] van Oort, E., Sonneveldt, L., Chu, Q. P., and Mulder, J., "Modular Adaptive Input-to-State Stable Backstepping of a Nonlinear Missile Model," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007. https://doi.org/10.2514/6.2007-6676.

[15] Lewis, F. L., and Vrabie, D., "Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control," *IEEE Circuits and Systems Magazine*, Vol. 09, No. 3, 2009, p. 32–50. https://doi.org/10.1109/MCAS.2009.933854.

[16] Bellman, R., Bellman, R., and Corporation, R., *Dynamic Programming*, Rand Corporation research study, Princeton University Press, 1957.

[17] Bellman, R., "Dynamic Programming," *Science*, Vol. 153, No. 3731, 1966, pp. 34–37. https://doi.org/10.1126/science.153.3731.34.

[18] Lewis, F. L., and Vamvoudakis, K. G., "Reinforcement Learning for Partially Observable Dynamic Processes: Adaptive Dynamic Programming Using Measured Output Data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 41, No. 1, 2011, pp. 14–25.

[19] Zhou, Y., van Kampen, E.-J., and Chu, Q. P., "Nonlinear Adaptive Flight Control Using Incremental Approximate Dynamic Programming and Output Feedback," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 493–496. https://doi.org/10.2514/1.G001762.

[20] Zhou, Y., van Kampen, E.-J., and Chu, Q. P., "Incremental Approximate Dynamic Programming for Nonlinear Adaptive Tracking Control with Partial Observability," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 12, 2018, pp. 2554–2567. https://doi.org/10.2514/1.G003472.

[21] Dias, P. M., Zhou, Y., and van Kampen, E.-J., "Intelligent Nonlinear Adaptive Flight Control using Incremental Approximate Dynamic Programming," *AIAA Scitech 2019 Forum*, 2019. https://doi.org/10.2514/6.2019-2339.

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

[23] Nguyen, L., Ogburn, M., Gilbert, W., Kibler, K., Brown, P., and Deal, P., "NASA Technical Paper 1538-Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with relaxed Longitudinal Static Stability," *Tech. rep., NASA*, 1979.

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

[25] Eerland, W., de Visser, C. C., and van Kampen, E.-J., "On Approximate Dynamic Programming with Multivariate Splines for Adaptive Control," *ArXiv*, Vol. abs/1606.09383, 2016.

[26] Leake, R. J., and Liu, R.-W., "Construction of Suboptimal Control Sequences," *SIAM Journal on Control*, Vol. 5, No. 1, 1967, pp. 54–63. https://doi.org/10.1137/0305004.

[27] Howard, R., *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, 1960.

[28] Kiumarsi, B., Lewis, F. L., Naghibi-Sistani, M., and Karimpour, A., "Optimal Tracking Control of Unknown Discrete-Time Linear Systems Using Input-Output Measured Data," *IEEE Transactions on Cybernetics*, Vol. 45, No. 12, 2015, pp. 2770–2779.

[29] Vamvoudakis, K. G., "Optimal trajectory Output Tracking control with a Q-learning algorithm," *2016 American Control Conference (ACC)*, 2016, pp. 5752–5757.

[30] Shirazi, M. J., Vatankhah, R., Boroushaki, M., Salarieh, H., and Alasty, A., "Application of particle swarm optimization in chaos synchronization in noisy environment in presence of unknown parameter uncertainty," *Communications in Nonlinear Science and Numerical Simulation*, Vol. 17, No. 2, 2012, pp. 742 – 753. https://doi.org/10.1016/j.cnsns.2011.05.032.

[31] Parsopoulos, K. E., "PARTICLE SWARM OPTIMIZER IN NOISY AND CONTINUOUSLY CHANGING ENVIRONMENTS," 2001.

[32] Beyer, H.-G., "Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2, 2000, pp. 239 – 267. https://doi.org/10.1016/S0045-7825(99)00386-2.