# Time-Varying Human-Operator Identification with Box-Jenkins Models

Master of Science Thesis

Graduation Report

Á. Ortiz Moya

Delft University of Technology, Faculty of Aerospace Engineering, Department Control & Operations, Section Control & Simulation

TU Delft

# Time-Varying Human-Operator Identification with Box-Jenkins Models

## Master of Science Thesis

by

Á. Ortiz Moya

to obtain the degree of
Master of Science in Aerospace Engineering at Delft University of Technology,
to be defended publicly on Wednesday December 6, 2023 at 12:45 PM.


| | |
|---|---|
| Student number | 5487862 |
| Supervisors: | Prof. Dr. Ir. M. Mulder |
| | Dr. Ir. M. M. van Paassen |
| | Dr. Ir. D. M. Pool |
| Date of submission: | November 2023 |

The work in this thesis was conducted in the:


Section Control & Simulation
Department Control & Operations
Faculty of Aerospace Engineering
Delft University of Technology


Cover: Commercial airplane cockpit by Rob Mark under Flying [39] (Modified)

TU Delft

Department of Control and Operations, Section Control and Simulation
Delft, The Netherlands

The undersigned hereby certify that they have read, and recommend to the Faculty of Aerospace Engineering at Delft University of Technology for acceptance, a thesis entitled: **Time-Varying Human-Operator Identification with Box-Jenkins Models**, submitted by **Á. Ortiz Moya** in partial fulfillment of the requirements for the award of the degree of **Master of Science**.

Dated: 6 December 2023

Assessment committee

Professor: _____

Prof. Dr. Ir. M. Mulder

Supervisors: _____

Dr. Ir. M. M. van Paassen

_____

Dr. Ir. D. M. Pool

External examiner: _____

Prof. Dr. Ir. J. C. F. de Winter

# Preface

*The work completed for my Master of Science (MSc) degree in Aerospace Engineering at Delft University of Technology is summarized in this thesis. This project was conducted at the Control and Simulation (C&S).*

*Part I includes a scientific paper that covers the main results and conclusions drawn from this project. In Part II, the preliminary graduation report is presented, which comprehends a literature study in time-varying identification methods and an analysis of the applicability of Box-Jenkins structures. Part III contains additional appendices that complement the work presented in Part I. Both Part I and III are meant for the Thesis Control and Operations (AE5310), while Part II was already graded for the Literature Study course (AE4020).*

*This graduation project has been a great opportunity to grow as a professional and acquire new skills. Learning more about system identification and how we can model human behaviour, while performing an extensive research project, has been a worthwhile experience. In addition, finding a mathematically feasible method by which the Box-Jenkins model can be applied, and deriving an expression to compute the remnant gain to be used, were the most exciting parts of this work.*

*I would like to thank Max, René and Daan who played a fundamental role in supervising this MSc Thesis. The goals proposed in this research project can be challenging and full of obstacles, so it was essential to rely on professors with adequate expertise to guide me as a student.*

*During these past two years, the best experience has been stepping out of my comfort zone and living in a different country, where I have had the great opportunity to meet friends from different parts of the world. Undoubtedly, discovering new cultures and ways of thinking not only enriches your mind, but also offers a new perspective on life. Thus, I want to thank them for their support and the great experiences we shared.*

*Á. Ortiz Moya*
*Delft, November 2023*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations and Symbols

## Abbreviations

| Abbreviation | Definition |
|---|---|
| ANN | Artificial Neural Network |
| AR | Auto-Regressive |
| ARX | Auto-Regressive-eXogeneous |
| ARMA | Auto-Regressive-Moving-Average |
| ARMAX | Auto-Regressive-Moving-Average-Exogenous |
| BJ | Box-Jenkins |
| CE | Controlled Element |
| EWP | Exponential-Weighting-into-the-Past |
| EKF | Extended Kalman Filter |
| FIR | Finite Impulse Response |
| FOH | First-Order Hold |
| GWN | Gaussian White Noise |
| HO | Human Operator |
| IC | Initial Conditions |
| LS | Least-Squares |
| LPV | Linear Parameter Variable |
| LTV | Linear-Time Variant |
| ML | Maximum Likelihood |
| MA | Moving-Average |
| NMS | Neuro-Muscular System |
| OL | Open Loop |
| OLS | Ordinary Least-Squares |
| OE | Output-Error |
| PBSID | Predictor-Based Subspace Identification |
| PEM | Prediction Error Method |
| RIV | Refined Instrumental Variable |
| RLS | Recursive Least-Squares |
| RPEM | Recursive Prediction Error Method |
| RRIV | Recursive Refined Instrumental Variable |
| RW | Rectangular Window |
| SLLANN | Single-Layer Linear Artificial Neural Network |
| TF | Transfer Function |
| UKF | Unscented Kalman Filter |
| VAF | Variance Accounted For |
| ZOH | Zero-Order Hold |

## Symbols

| Symbol | Definition | Unit |
|---|---|---|
| $A(z^{-1})$ | ARX model's output polynomial | - |
| $A_k$ | Sinusoid amplitude | $deg$ |
| $a_i^d$ | Coefficient $i$ of $A(z^{-1})$ | - |
| $B(z^{-1})$ | ARX or BJ model's HO input polynomial | - |
| $\overline{B}(z^{-1})$ | ARX or BJ model's HO input polynomial with time delay $\tau_e$ | - |
| $B_{r,i}$ | Relative bias for parameter $i$ | % |
| $b_i^d$ | Coefficient $i$ of $B(z^{-1})$ | - |

| Symbol | Definition | Unit |
|---|---|---|
| $C(z^{-1})$ | BJ model's remnant input polynomial | - |
| $c_i^d$ | Coefficient $i$ of $C(z^{-1})$ | - |
| $D(z^{-1})$ | BJ model's remnant output polynomial | - |
| $d_i^d$ | Coefficient $i$ of $D(z^{-1})$ | - |
| $e(t)$ | Error signal | $deg$ |
| $F(z^{-1})$ | BJ model's HO output polynomial | - |
| $f_i^d$ | Coefficient $i$ of $F(z^{-1})$ | - |
| $f_s$ | Sampling frequency | $Hz$ |
| $f_t$ | Forcing function | $deg$ |
| $G$ | Maximum rate of change in sigmoid | $s^{-1}$ |
| $g(t_k)$ | Gain vector | - |
| $H_{CE}(s,t)$ | Time-varying CE dynamics | - |
| $H_{HO}(s,t)$ | Time-varying HO dynamics | - |
| $H_{HO_e}(s,t)$ | Time-varying HO's linear response to $e(t)$ | - |
| $H_{nm}(s,t)$ | Neuromuscular dynamics | - |
| $H_n^m(s,t)$ | Time-varying remnant filter of order $m$ | - |
| $K_e(t)$ | Error gain | - |
| $K_n(t)$ | Remnant gain | - |
| $K_p(t)$ | Control gain | - |
| $M$ | Monte Carlo replications | - |
| $m$ | Order of remnant filter | - |
| $m^0$ | True order of remnant filter | - |
| $m^*$ | BJ model's order of remnant filter | - |
| $N$ | Number of samples | - |
| $N_i$ | Memory horizon in samples for $\lambda_i$ | - |
| $n(t)$ | Remnant signal | $deg$ |
| $n_a$ | Number of coefficients in $A(z^{-1})$ | - |
| $n_b$ | Number of coefficients in $B(z^{-1})$ | - |
| $n_c$ | Number of coefficients in $C(z^{-1})$ | - |
| $n_d$ | Number of coefficients in $D(z^{-1})$ | - |
| $n_f$ | Number of coefficients in $F(z^{-1})$ | - |
| $n_k$ | HO model's integer time delay | - |
| $n_k^*$ | ARX or BJ model's integer time delay | - |
| $n_k^0$ | True HO model's integer time delay | - |
| $n_t$ | Integer in sinusoid $k$ | - |
| $P(t_k)$ | Scaled covariance matrix | - |
| $P_0$ | Initial scaled covariance matrix | - |
| $P_n$ | Remnant intensity level | - |
| $p(t)$ | Parameter function | - |
| $p_i$ | Initial value of the parameter function | - |
| $p_f$ | Final value of the parameter function | - |
| $s$ | Laplace variable | - |
| $T_{e,i}$ | Memory horizon for $\lambda_i$ | s |
| $T_L(t)$ | Lead-time constant | s |
| $T_m$ | Total simulation time | s |
| $T_n$ | Remnant-time constant | s |
| $T_s(t)$ | Sampling time | s |
| $t$ | Continuous-time variable | s |
| $t_k$ | Discrete-time variable | s |
| $t_M$ | Time of maximum rate of change | s |
| $t_0$ | Transient time | s |
| $U$ | Control-output vector | $deg$ |
| $u(t)$ | Control-output signal | $deg$ |
| $V_N$ | Cost function | $deg^2$ |
| $V_N^0$ | Initial cost function | $deg^2$ |
| $VAF^{n_k^*,m^*}$ | Variance Accounted For at BJ model's $n_k$ and $m$ | % |
| $x(t)$ | System output signal | $deg$ |
| $z$ | Z-transform variable | - |

| Symbol | Definition | Unit |
|---|---|---|
| $Z^N$ | Data set with pairs $\{u(t_k), e(t_k)\}$ | $[deg, deg]$ |
| $Z_0^N$ | Initial data set with pairs $\{u(t_{k<1}), e(t_{k<1})\}$ | $[deg, deg]$ |
| $\varepsilon(t)$ | Remnant Gaussian White noise | $deg$ |
| $\epsilon(t)$ | Prediction error | $deg$ |
| $\zeta_{nm}(t)$ | Neuromuscular damping ratio | - |
| $\theta$ | Parameter vector | - |
| $\theta^0$ | Initial parameter estimate | - |
| $\theta_{ARX}$ | ARX model parameter vector | - |
| $\theta_{BJ}$ | BJ model parameter vector | - |
| $\Lambda$ | Forgetting matrix | - |
| $\lambda$ | Forgetting factor | - |
| $\pi(t_k)$ | Negative prediction error gradient | - |
| $\sigma_n^2$ | Variance of remnant noise signal | - |
| $\sigma_u^2$ | Variance of control-output signal | - |
| $\sigma_{u_n}^2$ | Variance of control-output signal due to remnant | - |
| $\sigma_\varepsilon^2$ | Variance of Gaussian White noise | - |
| $\tau_e$ | HO time delay | s |
| $\Phi$ | Regression matrix | - |
| $\phi_k$ | Sinusoid phase shift | $rad$ |
| $\varphi_m$ | Phase margin | $deg$ |
| $\varphi(t_k)$ | Regression vector | - |
| $\omega$ | Frequency | $rad/s$ |
| $\omega_b(t)$ | Break frequency | $rad/s$ |
| $\omega_c(t)$ | Crossover frequency | $rad/s$ |
| $\omega_k$ | Sinusoid frequency | $rad/s$ |
| $\omega_{k,0}$ | Sinusoid base frequency | $rad/s$ |
| $\omega_{nm}$ | Neuromuscular frequency | $rad/s$ |

# Part I

# Scientific Article

# Time-Varying Human-Operator Identification with Box-Jenkins Models

Á. Ortiz Moya*
*Delft University of Technology, Delft, South Holland, The Netherlands*

**The identification of time-varying, adaptive behaviour of a human operator in basic manual control tasks is undoubtedly under development since most methodologies only account for time-invariant systems. Previous authors have proved that estimation techniques based on ARX structures can generally identify the HO model parameters. Nonetheless, ARX methods present several problems, such as the persistent bias in estimates that may increase due to coupled noise and system models. Therefore, a novel identification technique based on Box-Jenkins models is proposed to achieve a more adequate match between the estimator structure and the HO model. The identification process can be conducted offline by the Ordinary Least Squares and Prediction Error Method, or online, when Recursive Least Squares and Recursive PEM are employed, respectively, in ARX and BJ models. The BJ estimator has excellent potential as an identification tool due to its bias reduction capabilities, as clearly shown in batch-fitting, although non-linear optimization processes decrease its convergence speed by 500%. An RPEM algorithm with forgetting factor $\lambda$ = 0.99609 and first-order remnant BJ structure is implemented and tested under Monte Carlo simulation and experimental data. Recursive BJ algorithms could help to achieve the ideal identification method by diminishing the Neuro-Muscular parameter bias in ARX.**

***Keywords*: ARX, Box-Jenkins, Equalization, Human Operator, Manual Control, Prediction Error Method, Recursive Prediction Error Minimization, Remnant, Time-varying Identification.**

## Nomenclature

| *Latin letters* | |
|---|---|
| $A(z^{-1})$ | BJ model's HO output polynomial |
| $A_k$ | Sinusoid amplitude, *deg* |
| $a_i$ | Coefficient $i$ of $A(z^{-1})$ |
| $B(z^{-1})$ | BJ model's HO input polynomial |
| $B_r(\vartheta)$ | Relative bias for parameter $\vartheta$, % |
| $b_i$ | Coefficient $i$ of $B(z^{-1})$ |
| $C(z^{-1})$ | BJ model's remnant input polynomial |
| $c_i$ | Coefficient $i$ of $C(z^{-1})$ |
| $D(z^{-1})$ | BJ model's remnant output polynomial |
| $d_i$ | Coefficient $i$ of $D(z^{-1})$ |
| $e(t)$ | Error signal, *deg* |
| $f_t(t)$ | Forcing function, *deg* |
| $G$ | Maximum rate of change in sigmoid, $s^{-1}$ |
| $g(t_k)$ | Gain vector |
| $H_{CE}(s,t)$ | Time-varying CE dynamics |
| $H_{HO}(s,t)$ | Time-varying HO dynamics |
| $H_{HO_e}(s,t)$ | Time-varying HO's linear response to $e(t)$ |
| $H_{nm}(s,t)$ | Neuromuscular dynamics |
| $H_n^m(s,t)$ | Time-varying remnant filter of order $m$ |
| $K_e(t)$ | Error gain |
| $K_n(t)$ | Remnant gain |
| $K_p(t)$ | Control gain |
| $M$ | Monte Carlo replications |
| $m$ | Order of remnant filter |
| $N$ | Number of samples |
| $N_e$ | Memory horizon in samples for $\lambda$ |
| $N_t$ | Number of sinusoids |
| $n(t)$ | Remnant signal, *deg* |
| $n_a$ | Number of coefficients in $A(z^{-1})$ |
| $n_b$ | Number of coefficients in $B(z^{-1})$ |
| $n_c$ | Number of coefficients in $C(z^{-1})$ |
| $n_d$ | Number of coefficients in $D(z^{-1})$ |
| $n_k$ | HO model's integer time delay |
| $n_t$ | Integer in sinusoid $k$ |
| $P(t_k)$ | Scaled covariance matrix |
| $P_n$ | Remnant intensity level |
| $p(t)$ | Parameter function |
| $s$ | Laplace variable |
| $T_e$ | Memory horizon for $\lambda$, s |
| $T_L(t)$ | Lead-time constant, s |
| $T_m$ | Total simulation time, s |

*MSc Student, Control and Simulation Section, Faculty of Aerospace Engineering, 2600 GB Delft, The Netherlands; a.ortizmoya@student.tudelft.nl. Student Member AIAA.

| | |
|---|---|
| $T_n$ | Remnant-time constant, s |
| $T_s$ | Sampling time, s |
| $t$ | Continuous-time variable, s |
| $t_k$ | Discrete-time variable, s |
| $t_M$ | Time of maximum rate of change, s |
| $u(t)$ | Control-output signal, $deg$ |
| $V^N$ | Cost function, $deg^2$ |
| $VAF$ | Variance Accounted For at BJ model |
| $x(t)$ | System output signal, $deg$ |
| $z$ | Z-transform variable |
| $Z^N$ | Data set with pairs $\{u(t_k), e(t_k)\}$, $deg$ |

*Greek letters*

| | |
|---|---|
| $\varepsilon(t)$ | Remnant Gaussian White noise, $deg$ |
| $\epsilon(t)$ | Prediction error, $deg$ |
| $\zeta_{nm}(t)$ | Neuromuscular damping ratio |
| $\theta$ | Parameter vector |
| $\vartheta$ | Global variable for BJ or HO parameter |
| $\lambda$ | Forgetting factor |
| $\pi(t_k)$ | Negative prediction error gradient |
| $\sigma^2$ | Variance |
| $\tau_e$ | HO time delay, s |
| $\phi_k$ | Sinusoid phase shift, $rad$ |
| $\varphi_m$ | Phase margin, $deg$ |
| $\omega$ | Frequency, $rad/s$ |
| $\omega_b(t)$ | Break frequency, $rad/s$ |
| $\omega_c(t)$ | Crossover frequency, $rad/s$ |
| $\omega_k$ | Sinusoid frequency, $rad/s$ |
| $\omega_{k,0}$ | Sinusoid base frequency, $rad/s$ |
| $\omega_{nm}$ | Neuromuscular frequency, $rad/s$ |

*Superscripts*

| | |
|---|---|
| 0 | True |
| $*$ | Simulated in BJ model |
| $d$ | Discrete time |
| $c$ | Continuous time |

*Subscripts*

| | |
|---|---|
| $i$ | Initial |
| $f$ | Final |

# I. Introduction

Human manual-control behaviour naturally changes over time, in various contexts, and across operators. In order to explain the dynamic features of human operators (HOs) in skill-based manual control tasks, identification methods have been developed [1–8]. However, they are often only applicable in situations when the control behaviour is sufficiently time-invariant [6]. The availability of control-theoretic models that can capture both the adaptive and learning aspects of manual-control behaviour has long been a goal [9–14]. Modern cybernetics is unable to fully explain how HOs modify their behaviour to deal with control-task changes. The continued development of time-varying identification techniques is necessary to make rapid progress in our knowledge of how people really interact with dynamic control systems [15].

The majority of research on identifying time-varying manual-control behaviour focuses on task variable changes, particularly those caused by variations in the dynamics of the controlled element (CE) [3, 9–13, 16]. Many research projects begin by examining single-axis compensatory control tasks and are based on the well-known crossover theory [3], assuming that those developed models are scaleable to scenarios with additional types of inputs to the HO or even multiple axes of control [13].

The purpose of this article is to study and develop a tool capable of identifying real-time (i.e. online) human control behaviour adaptation in a compensatory manual-control task. Real-time execution of these processes offers new possibilities in addition to advancing the earlier study objectives. For instance, pilot adaptation to anomalies in aircraft or controlled elements could be studied and predicted [11–14]. On the other hand, reduced attention or distraction in real-world control activities could be identified by continuously monitoring the operator through an update of a human operator model [17]. Additionally, this might allow for adaptive haptic feedback that matches the operator's present behaviour [18]. Online identification can assist in modifying experimental circumstances in real time to directly analyse adaptation behaviour or obtain desired haptic feedback characteristics in research with humans-in-the-loop.

Multiple authors have attempted to apply different techniques to achieve a successful online identification [7, 10, 18–23]. Nevertheless, most of the methodologies used are not able to provide ideal results due to the high difficulty of the problem. For instance, strategies based on recursive estimation of Auto-Regressive-eXogeneous (ARX) model structures [6, 10, 12, 24, 25] may fail in reducing the relative bias [10] of the predicted model, in spite of their outstanding qualities in terms of low computational effort and straightforward estimation. Consequently, it is necessary to find an identification method that is truly capable of assuming this task, so that a novel estimation technique based on Box-Jenkins (BJ) structures is proposed. This paper lays the foundation for the development of a recursive BJ algorithm, which aims to solve the previous issues in the human behaviour identification process. In parallel, the performance of such a novel identification technique is compared to the ARX estimation outcome. Both methods are evaluated under a time-varying scenario through Monte Carlo simulation and with experimental data obtained by Van Grootheest et al. [10].

This article is structured as follows. The compensatory manual-control task and previous identification methods are described in Section II. The Box-Jenkins identification approach is detailed in Section III. Afterwards, Section IV discusses the required simulation conditions and their setup, while Section V presents an overview of the Monte Carlo simulation results obtained. Identification results for experimental data are presented in Section VI. Section VII discusses the applicability, limitations and impact of the research project. Final conclusions are drawn in Section VIII.

## II. Compensatory Manual-Control Task

In a control task, the human operator is typically a multichannel, adaptive, learning, non-linear controller [6–8, 20, 24–26]. Although a general control-theoretic model of the human controller has not yet been discovered, validated models do exist for certain control tasks as shown in [27]. In particular, McRuer and Jex [3] proposed quasi-linear human operator models that still represent the state-of-the-art in HO modelling. These models are based on the Crossover Model theory [1, 3], which is only applicable to simple cases, i.e., single-channel tracking tasks with only a feedback path from visual perception (pure compensatory display). The quasi-linear models separate the additional unexplained behaviour by adding noise called 'remnant' $n$ and capture the linear behaviour of the human controller in a descriptive transfer function $H_{HO_e}(s,t)$.

Figure 1 depicts the quasi-linear operator model embedded in a compensating tracking task [11]. The HO, represented by the model $H_{HO}(s,t)$, monitors and responds to the error $e(t)$ between a goal $f_t(t)$ and the output $x(t)$ of the CE dynamics $H_{CE}(s,t)$. The HO dynamics are composed of the remnant $n(t)$ and deterministic responses from $H_{HO_e}(s,t)$. Regarding the remnant $n(t)$, it is generated by feeding a white-noise signal $\varepsilon(t)$ with a determined statistical distribution through a remnant filter $H_n^m(s,t)$. Additionally, the HO must alter its control strategy as the CE dynamics change over time (CE adaptation) [9].

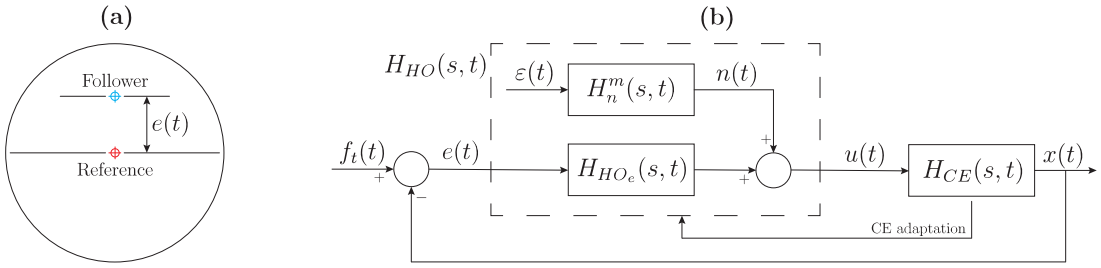


**Figure 1. Single-axis compensatory manual-control task with time-varying dynamics: (a) Compensatory display, where $e(t)$ acts as a stimulus, and (b) Block diagram.**

### A. Human-Operator Dynamics

In the Crossover model, McRuer and Jex [3] state that people modify their control behaviour to satisfy

$$H_{OL}(j\omega) = H_{HO_e}(s = j\omega)H_{CE}(j\omega) = \frac{\omega_c}{j\omega}e^{-j\omega\tau_e}, \omega \approx \omega_c \tag{1}$$

in the crossover zone when transitory behaviour is eliminated. Then adjustment rules define how the describing function $H_{HO_e}(s,t)$ behaves in relation to the controlled element $H_{CE}(s,t)$ and what impact it has on crossover frequency $\omega_c$ and time delay $\tau_e$ in the frequency domain [3, 9]. By modelling the neuromuscular system (NMS) as a second-order transfer function $H_{nm}(j\omega)$ and the operator equalization as a gain and a lead(L) [3–5, 11], while using the approach stated in [10–12], the general formulation of the describing function can be defined as follows:

$$H_{HO_e}(s,t) = K_e(t)\,[T_L(t)s + 1]\,e^{-s\tau_e}H_{nm}(j\omega) = \frac{K_e(t)\,[T_L(t)s + 1]\,e^{-s\tau_e}\omega_{nm}^2}{s^2 + 2\zeta_{nm}\omega_{nm}s + \omega_{nm}^2}. \tag{2}$$

The equalization parameters, i.e. $K_e(t)$ and $T_L(t)$, determine the action of the feedback controller in the pilot-vehicle loop. On the other hand, $\omega_{nm}$ and $\zeta_{nm}$ model the neuromuscular dynamics.

Regarding possible remnant signal models, the theoretical background is limited and there is no consensus on how to model and take into account this remnant in Monte Carlo simulations [11, 20, 28, 29]. In most cases, a remnant signal is obtained by passing zero-mean Gaussian white noise (GWN) through a filter.

The literature contains a variety of filter options, however, the most accepted one is the $m^{th}$-order remnant-filter proposed by Zaal [11] and later used in [10, 12]:

$$H_n^m(s,t) = \frac{K_n(t)}{(T_n s + 1)^m}. \tag{3}$$

Additionally, the noise level $P_n$ has to be set during Monte Carlo simulations to give a certain value for the remnant gain. The most adequate definition is provided by Van der El et al. [30], $P_n = \sigma_{u_n}^2/\sigma_u^2$, which compares the variance of $u(t)$ due to the remnant to the $u(t)$ signal's total variance $\sigma_u^2$.

### B. Controlled-Element Dynamics

The following second-order CE dynamics were taken into consideration by several authors [10–12], which serve as a general low-order approximation of typical vehicle dynamics [3]:

$$H_{CE}(s,t) = \frac{K_c(t)}{s(s + \omega_b(t))}. \tag{4}$$

The break frequency $\omega_b(t)$ and the control gain $K_c(t)$ can both change over time. The dynamics variation of the controlled element from single- to double-integrator dynamics (i.e., $1/s \leftrightarrow 1/s^2$) occurs at approximately $\omega_b(t)$. Furthermore, a sigmoid function is used in [10–12] to define the time variation of the operator equalization parameters in Equation (2) and CE coefficients in Equation (4):

$$p(t) = p_i + \frac{p_f - p_i}{1 + e^{-G(t - t_M)}}, \tag{5}$$

where $p(t)$ is the time-varying parameter, $p_i$ and $p_f$ are the initial and final parameter values, $G$ is the transition rate, and $t_M$ is the time when states transition occurs.

### C. Identification of Time-Varying Operator Behaviour

In HO system identification, there are two possible directions to estimate the $H_{HO_e}$ dynamics. The non-parametric method [7, 18, 19, 31, 32], only provides direct estimations of frequency response, and the parametric approach assumes a HO model structure and requires estimation of its parameters.

The parametric methodology provides a more tangible understanding of the human controller than non-parametric strategies. Five research lines are found: batch-fitting methods (maximum likelihood estimation [11, 20], fitting Linear Parameter Variable (LPV) state space systems [21]) and recursive fitting methods (Kalman filter estimation [16, 22, 33], fitting recursive ARX models [10, 12, 25], identification of Artificial Neural Networks (ANNs) [23]).

In batch fitting strategies, the fitting is applied on the whole dataset at once, and typically, the operation limitation parameters (neuromuscular dynamics and operator time delay) are deemed constant while only the operator equalization parameters may vary. On the other hand, in recursive fitting methods, the operation equalization parameters can vary, but also, the NMS and HO time delay can be set constant or assumed time-varying depending on the type of estimator used. The ARX model structure [34] is employed in [10, 12, 25] to estimate time-varying HO behaviour, extending the work done in [6, 24]. ARX parameters are computed by the Recursive Least Squares (RLS) [35], which minimizes a weighted linear least squares cost function relating to the input signals. Additionally, the RLS algorithm can be tuned by evaluating the ARX structure for constant HO models [10] through an Ordinary Least Squares (OLS) algorithm [34].

ARX identification methods represent a simple, efficient option since they only require a linear optimization process. In addition, convergence is fast in recursive estimation, which guarantees an acceptable adaptation to HO changes. However, permanent biases are found in estimated model parameters due to the mismatch in the remnant filter poles (mainly when the remnant order $m$ equals 1 [10]), seeing that both denominators of the HO linear component and remnant filter are considered as equivalent in ARX structures. Hence, an analysis of other model structures is recommended in order to reduce the persistent bias found in ARX.

## III. Box-Jenkins Model Identification

### A. Box-Jenkins Model Family

Ljung [34] presents a series of transfer-function models from a general family of model structures, which can be employed to identify the discrete-time TF corresponding to the HO model. Figure 2 shows the ARX and BJ structures:

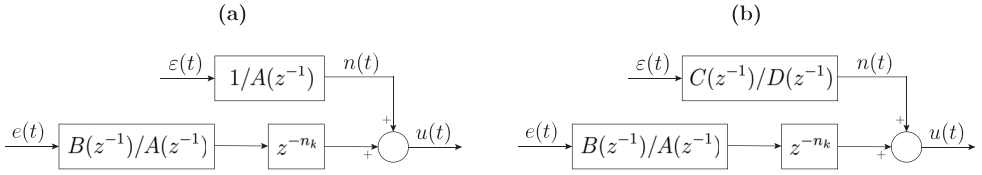


**Figure 2. Discrete-time model structures: (a) ARX, and (b) Box-Jenkins.**

For the present case study, the Box-Jenkins structure represents the most suitable option since it enables more freedom for zero and pole placement in both the remnant filter and the linear HO model, while ARX assumes a coupling between these two components. This BJ family can be represented by the following expression:

$$u(t_k) = \frac{B(z^{-1})}{A(z^{-1})} z^{-n_k} e(t_k) + \frac{C(z^{-1})}{D(z^{-1})} \varepsilon(t_k), \quad \varepsilon(t_k) \sim N(0, \sigma_\varepsilon), \tag{6}$$

where $u(t)$ is the output signal associated with the control-output, $e(t)$ is the input signal corresponding to the tracking error, and $\varepsilon(t)$ is the Gaussian White Noise (GWN) with a standard deviation $\sigma_\varepsilon$. The variable $z^{-1}$ acts as a discrete-time shift operator, while $t_k$ represents the discretized time. The general model structure depends on a total of 4 polynomials $\{A, B, C, D\}$ with orders defined by the integers $\{n_a, n_b, n_c, n_d\}$:

$$A(z^{-1}) = 1 + a_1^d z^{-1} + \ldots + a_{n_a}^d z^{-n_a}, \qquad B(z^{-1}) = b_0^d + b_1^d z^{-1} + \ldots + b_{n_b}^d z^{-n_b}, \tag{7a}$$

$$C(z^{-1}) = 1 + c_1^d z^{-1} + \ldots + c_{n_c}^d z^{-n_c}, \qquad D(z^{-1}) = 1 + d_1^d z^{-1} + \ldots + d_{n_d}^d z^{-n_d}. \tag{7b}$$

In addition, the predictor of the control-output signal, $\hat{u}(t)$, would present the following general expression [34]:

$$\hat{u}(t_k \mid \theta) = \frac{B(z^{-1})D(z^{-1})}{C(z^{-1})A(z^{-1})} z^{-n_k} e(t_k) + \left[1 - \frac{D(z^{-1})}{C(z^{-1})}\right] u(t_k), \tag{8}$$

where $\theta$ is the adjustable parameters vector, i.e.,

$$\theta = \left[a_1^d, a_2^d, \ldots, a_{n_a}^d, b_0^d, b_1^d, \ldots, b_{n_b}^d, c_1^d, c_2^d, \ldots, c_{n_c}^d, d_1^d, d_2^d, \ldots, d_{n_d}^d\right]^T. \tag{9}$$

By adjusting this previous expression, the prediction error is found:

$$\epsilon(t_k, \theta) = u(t_k) - \hat{u}(t_k \mid \theta) = \frac{D(z^{-1})}{C(z^{-1})} \left[u(t_k) - \frac{B(z^{-1})}{A(z^{-1})} z^{-n_k} e(t_k)\right]. \tag{10}$$

### B. HO Model Discretization and BJ Structure

Before setting batch-fitting or recursive estimations of BJ models, it is necessary to make sure those structures match the discretized HO and remnant filter models. Franklin et al. [36] present a variety of discretization techniques that can be applied to the continuous-time TF, and can follow a numerical integration, a Z-transform mapping strategy or try to model the sampled system. The discrete equivalents via numerical integration are based on the numerical method [37] used to find a solution of the differential equation associated to the continuous-time TF to be discretized. The principal methods are: Forward-Euler, Backward-Euler, Tustin (with/without pre-warp). Then, the zero-pole matching equivalents are obtained by mapping the continuous-time TF's poles and zeros from the relationship between s- and z-planes. On the other hand, Tangirala [38] proposes model-sampling techniques, based on the Zero-Order Hold (ZOH) or First-Order Hold (FOH), that reconstruct the measured signal and enable a two-step continuous-to-discrete time TF conversion. In the HO model, a combination between ZOH and Backward Euler is applied.

#### *1. HO Linear Component*

In the case of $H_{HO_e}(s, t)$, Hess and Mnich [39] proved that the only discretization method option is ZOH (there must be a single solution for HO coefficients), achieving a discrete model with $n_a = 2$ and $n_b = 1$:

$$H_{HO_{e,dis}}(z) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}} \cdot z^{-n_k}, \tag{11}$$

where an additional unit-sample delay $z^{-1}$ is obtained due to the discretization method. The integer $n_k$ represents the time delay in the discretized model. Thus, the estimated continuous-time TF of the HO model, $\hat{H}_{HO_e}(s,t_k)$, and the identified human operator coefficients are:

$$\hat{H}_{HO_e}(s,t_k) = \frac{b_0^c s + b_1^c}{s^2 + a_1^c s + a_2^c} \;\Rightarrow\; \left\{ \hat{K}_e = \frac{b_1^c}{a_2^c},\ \hat{T}_L = \frac{b_0^c}{b_1^c},\ \hat{K}_{\dot{e}} = \frac{b_0^c}{a_2^c},\ \hat{\omega}_{nm} = \sqrt{a_2^c},\ \hat{\zeta}_{nm} = \frac{a_1^c}{2\sqrt{a_2^c}}, \hat{\tau}_e = T_s(n_k - 1) \right\}. \tag{12}$$

*2. Remnant Filter*

By the Backward Euler discretization, $s = (1 - z^{-1})/T_s$, a proper discrete-time transfer function for the remnant filter is achieved:

$$H_n^m(s) = \frac{K_n}{(T_n s + 1)^m} \quad \rightarrow \quad H_{n,dis}^m(z) = \frac{K_n}{\left(T_n \frac{1-z^{-1}}{T_s} + 1\right)^m} = \frac{K_n \left(\frac{T_s}{T_n + T_s}\right)^m}{\left(1 - \frac{T_n}{T_n + T_s} z^{-1}\right)^m}. \tag{13}$$

Since the numerator of the estimated discrete-time remnant filter must be equal to 1 based on the BJ model structure, the resulting coefficient in the numerator of $H_{n,dis}^m(z)$ will be incorporated into the variance $\sigma_\varepsilon^2$, giving as a result a modified $\varepsilon'$ with standard deviation $\sigma_{\varepsilon'}$:

$$\sigma_{\varepsilon'} = \frac{K_n T_s^m}{(T_n + T_s)^m} \sigma_\varepsilon. \tag{14}$$

Hence, the discrete-time model of the remnant noise signal would be the following one:

$$n(t_k) = \frac{1}{\left(1 - \frac{T_n}{T_n + T_s} z^{-1}\right)^m} \varepsilon'(t_k), \qquad \varepsilon'(t_k) \sim N(0, \sigma_{\varepsilon'}). \tag{15}$$

Depending on the $m^{th}$-order of the remnant filter, the number of discrete-time parameters, $d_i^d$, to be estimated changes, while the time constant $T_n$ would need to be averaged from such parameters computed:

$$d_i^d = f_i(T_s, T_{n,d_i^d}) \quad \rightarrow \quad T_{n,d_i^d} \quad \rightarrow \quad \hat{T}_n = \frac{1}{m} \sum_{i=1}^{m} T_{n,d_i^d}. \tag{16}$$

*3. Applied BJ Model Expression*

Therefore, a BJ model structure $\{n_a = 2, n_b = 1, n_c = 0, n_d = m^*\}$ is chosen to address the identification problem:

$$u(t_k) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}} z^{-n_k^*} e(t_k) + \frac{1}{1 + d_1^d z^{-1} + \ldots + d_{m^*}^d z^{-m^*}} \varepsilon'(t_k), \quad \varepsilon'(t_k) \sim N(0, \frac{K_n T_s^{m^*}}{(T_n + T_s)^{m^*}} \sigma_\varepsilon), \tag{17}$$

where $m^*$ and $n_k^*$ are the remnant order and time delay considered in the estimation model, respectively. Thus, the $BJ(n_k^*, m^*)$ prediction error and adjustable parameter vector are given by the expressions:

$$\epsilon(t_k, \theta) = D(z^{-1}) \left[ u(t_k) - \frac{B(z^{-1})}{A(z^{-1})} z^{-n_k^*} e(t_k) \right], \quad \theta = \left[ a_1^d, a_2^d, b_0^d, b_1^d, d_1^d, \ldots, d_{m^*}^d \right]^T. \tag{18}$$

**C. Prediction Error Method**

The parameter estimation of each BJ discrete-time polynomial requires the optimization of a non-linear problem, hence, a Prediction Error Method (PEM) algorithm has to be evaluated as explained in [34, 38, 40, 41]. The PEM procedure consists of initial state estimation, definition of the optimization model, and variance estimation.

Other methods can also be applied to the discrete-time transfer function estimation problem, such as the Maximum Likelihood (ML) or the Refined Instrumental Variable (RIV) method [34, 40, 41]. The ML technique finds the full optimization model from a log-likelihood function, which is based on a Gaussian distribution of the noise $\varepsilon$. The RIV method is a pseudo-linear regression approach to ML estimation. Both methods could be in ARX and BJ structures, but they are only applicable to systems with Gaussian noise, thus, the PEM algorithm presents a more generalist alternative. The ML model is only used to find an expression to estimate the noise variance (see Equation (30)).

*1. The Algorithm*

Prediction error methods are based on the idea of minimising a cost function, $V^N$, that measures the level of prediction error, $\epsilon(t_k, \theta)$, to find a solution for $\hat{\theta}$ [34, 40]. Thus, from a batch of data $Z^N$,

$$Z^N = [u(t_1), e(t_1), u(t_2), e(t_2), \ldots, u(t_N), e(t_N)], \tag{19}$$

and the prediction error formula (see Eq. (18)), the cost function can be defined as follows:

$$V^N(\theta, Z^N) = \frac{1}{N}\sum_{i=1}^{i=N} l(\epsilon(t_i, \theta)). \tag{20}$$

In $V^N(\theta, Z^N)$, $l(\cdot)$ is a scalar-valued (typically positive) function. The quadratic norm is the most common in optimization problems:

$$l(\epsilon) = \frac{1}{2}\epsilon^2. \tag{21}$$

Therefore, the goal of a PEM algorithm is to find the vector of parameters $\hat{\theta}$ that minimizes the cost function:

$$\hat{\theta} = \arg\min_{\theta} V^N(\theta, Z^N) = \arg\min_{\theta} \left\{ \frac{1}{2N}\sum_{i=1}^{i=N} \left[ D(z^{-1})u(t_i) - \frac{B(z^{-1})D(z^{-1})}{A(z^{-1})} z^{-n_k} e(t_i) \right]^2 \right\}. \tag{22}$$

To optimize such cost function, the partial differentiation of $V^N(\theta, Z^N)$ with respect to all the parameters is made:

$$\nabla_\theta \left[ V^N(\theta, Z^N) \right] = \frac{1}{N}\sum_{i=1}^{i=N} \epsilon(t_i, \theta) \nabla_\theta \left[ \epsilon(t_i, \theta) \right]. \tag{23}$$

Consequently, the optimization model is defined by the following set of equations when equalled to zero:

$$\frac{\partial V^N}{\partial a^d_{j=1,2}} = \frac{1}{N}\sum_{i=i_0}^{i=N} \left[ D(z^{-1})u(t_i) - \frac{B(z^{-1})D(z^{-1})}{A(z^{-1})} z^{-n_k} e(t_i) \right] \times \frac{B(z^{-1})D(z^{-1})}{A^2(z^{-1})} z^{-j-n_k} e(t_i) = 0, \tag{24a}$$

$$\frac{\partial V^N}{\partial b^d_{j=0,1}} = \frac{1}{N}\sum_{i=i_0}^{i=N} \left[ D(z^{-1})u(t_i) - \frac{B(z^{-1})D(z^{-1})}{A(z^{-1})} z^{-n_k} e(t_i) \right] \times \frac{D(z^{-1})}{A(z^{-1})} z^{-j-n_k} e(t_i) = 0, \tag{24b}$$

$$\frac{\partial V^N}{\partial d^d_{j=1,2,\ldots,m^*}} = \frac{1}{N}\sum_{i=i_0}^{i=N} \left[ D(z^{-1})u(t_i) - \frac{B(z^{-1})D(z^{-1})}{A(z^{-1})} z^{-n_k} e(t_i) \right] \times \left[ z^{-j} u(t_i) - \frac{B(z^{-1}) z^{-j-n_k}}{A(z^{-1})} e(t_i) \right] = 0. \tag{24c}$$

At this point, the PEM and ML optimization problems are identical [41], since Equations (24) match the ones obtained in an ML scenario. In order to find the optimal solution to this minimization problem, BJ models require non-linear optimization methods based on gradient-based schemes [42], or more innovative techniques, such as the combination of different line search algorithms at each iteration [43], which is used in this case.

When calculating each component of Equations (24), there might be some difficulties in evaluating fractions of polynomials. To solve this issue, Maclaurin series [43, 44] can be used to convert a fraction into a pure polynomial:

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + \ldots \tag{25}$$

*2. Initial States Estimation*

In BJ estimation, the index $i_0$ in Eqs. (24) should be always equal to 1 due to the higher relevance of the initial states in the optimization process. Otherwise, if a first-order series is assumed, the minimum index should be:

$$i_0 = 3n_a + n_b + n_d + n_k + 1 = m^* + n_k^* + 8. \tag{26}$$

Therefore, an initial model needs to be estimated beforehand to find the pairs $\{u(t_{k<1}), e(t_{k<1})\}$, i.e.,

$$Z_i^N = [\ldots, u(t_{-1}), e(t_{-1}), u(t_0), e(t_0)], \tag{27}$$

based on the initial conditions $\theta_i$ (i.e., $A_i(z^{-1})$, $B_i(z^{-1})$, $C_i(z^{-1})$, $D_i(z^{-1})$) [43]:

$$\hat{Z}_i^N = \arg\min_{Z_i^N} V_i^N(\theta_i, Z_i^N) = \arg\min_{Z_i^N} \left\{ \frac{1}{2N}\sum_{i<1} \left[ D_i(z^{-1})u(t_i) - \frac{B_i(z^{-1})D_i(z^{-1})}{A_i(z^{-1})} z^{-n_k^*} e(t_i) \right]^2 \right\}. \tag{28}$$

*3. Variance Estimation*

Once, the optimization process is converged and a parameter vector solution is found, a partial differentiation of the log-likelihood function for $N$ observations shown by Young [41],

$$\mathscr{L}(\theta, \sigma^2_{\varepsilon'}, u(t), e(t)) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\left(\sigma^2_{\varepsilon'}\right) - \frac{1}{2\sigma^2_{\varepsilon'}}\sum_{i=1}^{i=N}\left[D(z^{-1})u(t_i) - \frac{B(z^{-1})D(z^{-1})}{A(z^{-1})}z^{-n_k}e(t_i)\right]^2 \tag{29}$$

is computed to find an estimation of the noise variance $\hat{\sigma}^2_{\varepsilon'}$ based on the predicted parameter vector:

$$\frac{\partial \mathscr{L}}{\partial \sigma^2_{\varepsilon'}} = 0 \quad \rightarrow \quad \hat{\sigma}^2_{\varepsilon'} = \frac{1}{N}\sum_{i=1}^{i=N}\left[\hat{D}(z^{-1})u(t_i) - \frac{\hat{B}(z^{-1})\hat{D}(z^{-1})}{\hat{A}(z^{-1})}z^{-n^*_k}e(t_i)\right]^2. \tag{30}$$

**D. Recursive Prediction Error Minimization**

The Recursive Prediction Error Minimization (RPEM) is one of the most extended online estimation techniques, which enables a recursive implementation of the PEM algorithm [34, 40]. Other methods can be applied to the online BJ structure estimation problem, such as the Real-Time Recursive Refined Instrumental Variable (RRIV) or the Extended Kalman Filter (EKF) [40]. The RRIV method is probably the most similar to the RPEM algorithm, although it presents some differences in terms of the covariance matrix definition (two sub-matrices with null off-diagonal blocks are used in RRIV), the robustness of the algorithm (the Instrumental Variable modifications ensure a stable estimation process), or the steps required (while RPEM is fully recursive, RRIV needs to iterate at each $k^{th}$ period).

*1. The Algorithm*

A Recursive PEM algorithm [40] is built, by analogy with the RLS method and fulfilling the theoretical requirements to achieve convergence. The RPEM algorithm is an online estimation method that consists of 3 steps, in which the parameter vector $\theta(t_k)$ is adjusted online by means of the gain vector $g(t_k)$ and the prediction error of $u(t_k)$. The gain vector is also tuned based on a forgetting factor $\lambda$, the negative gradient of the prediction error $\pi(t_k)$, and the scaled covariance matrix $P(t_k)$, which accounts for the certainty in the estimation conducted of each model parameter. Equations (31) [34, 40] present the formulas used in this iterative process:

$$\hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + P(t_k)\pi(t_k)\epsilon(t_k), \tag{31a}$$

$$g(t_k) = \frac{P(t_{k-1})\pi(t_k)}{\lambda + \pi^T(t_k)P(t_{k-1})\pi(t_k)}, \tag{31b}$$

$$P(t_k) = \frac{1}{\lambda}\left[P(t_{k-1}) - g(t_k)\pi^T(t_k)P(t_{k-1})\right]. \tag{31c}$$

Equation (31a) can be evaluated in two ways due to the relationship $g(t_k) = P(t_k)\pi(t_k)$, although the first option is usually preferred since it is more computationally efficient [40]. Each element of the forgetting matrix must belong to the interval $I = [0, 1]$, where a $\lambda$ near a null value provides a negligible memory horizon, and values close to 1 increase this horizon significantly. Hence, for a forgetting factor $\lambda$, a total number of $N_e = 1/(1-\lambda)$ samples are considered in the RLS algorithm for a time horizon of $T_e = T_s/(1-\lambda)$.

The definition of vector $\pi(t_k)$ is based on the minimization of the instantaneous part of the cost function in Equation (20), in which the one-step-ahead prediction error can be approximated by the employment of the model parameters estimation at the previous $(k-1)^{th}$ instant:

$$V^k(\theta(t_k)) = \frac{1}{2}\left[\epsilon^2(t_k, \theta(t_k))\right] \approx V^k(\hat{\theta}(t_{k-1})) = \frac{1}{2}\left[\epsilon^2(t_k, \hat{\theta}(t_{k-1}))\right]. \tag{32}$$

As a result, the gain $\pi(t_k)$ is computed as the negative gradient of the prediction error [40]:

$$\pi(t_k) = -\frac{\partial\epsilon(t_k, \hat{\theta}(t_{k-1}))}{\partial\hat{\theta}(t_{k-1}))}. \tag{33}$$

From Eq. (18), the prediction error derivatives can be defined as follows:

$$\frac{\partial \epsilon(t_k)}{\partial a^d_{i=1,2}} = \frac{D(z^{-1})}{A(z^{-1})}\frac{B(z^{-1})}{A(z^{-1})}e(t_{k-i-n_k}) = \frac{D(z^{-1})}{A(z^{-1})}y(t_{k-i-n_k}) = y_{f_1}(t_{k-i-n_k}), \tag{34a}$$

$$\frac{\partial \epsilon(t_k)}{\partial b^d_{i=0,1}} = -\frac{D(z^{-1})}{A(z^{-1})}e(t_{k-i-n_k}) = -e_{f_1}(t_{k-i-n_k}), \tag{34b}$$

$$\frac{\partial \epsilon(t_k)}{\partial d^d_{i=1,2,\ldots,m^*}} = -D(z^{-1})u(t_{k-i}) - \frac{D(z^{-1})B(z^{-1})}{A(z^{-1})}e(t_{k-i-n_k}) = -\epsilon(t_{k-i}). \tag{34c}$$

Where the subscript $f_1$ denotes that the variable is filtered by the transfer functions:

$$f_1 = \frac{D(z^{-1})}{A(z^{-1})}. \tag{35}$$

Therefore, the negative gradient $\pi(t_k)$ presents the following expression:

$$\pi(t_k) = -\frac{\partial \epsilon}{\partial \hat{\theta}} = \left[-\hat{y}_{\hat{f}_1}(t_{k-1-n_k}), -\hat{y}_{\hat{f}_1}(t_{k-2-n_k}), e_{\hat{f}_1}(t_{k-n_k}), e_{\hat{f}_1}(t_{k-1-n_k}), \hat{\epsilon}(t_{k-1}), \ldots, \hat{\epsilon}(t_{k-n_d})\right]^T. \tag{36}$$

The prefilter $\hat{f}_1$ and variables $\hat{y}(t_k)$ are calculated from the latest estimated polynomials, respectively:

$$\hat{f}_1 = \frac{\hat{D}(z^{-1})}{\hat{F}(z^{-1})}, \quad \hat{y}(t_k) = \frac{\hat{B}(z^{-1})}{\hat{F}(z^{-1})}e(t_{k-n_k}). \tag{37}$$

Thus, the classical RLS algorithm is adapted to non-linear cases by means of converting $\pi(t_k)$ into a vector composed of linear variables, as shown in Equation (36). A forgetting factor $\lambda = 0.99609$ is selected for the implementation of the recursive BJ method, based on previous results from Van Grootheest et al. [10]. In addition, the initial covariance matrix can be defined as:

$$P_i = diag\left(0.1, 0.1, 0.1, 0.1 \times m^*\right). \tag{38}$$

### E. Quality-of-Fit Metrics

In order to verify the quality of the estimation performed, several metrics need to be used. These metrics have to be relevant in terms of addressing how well the predicted model represents the dynamics of the real one. To achieve such a goal, the prediction error in estimated model parameters has to be analysed, but the prediction capabilities of such a model must be also studied.

Hence, two different quality-of-fit metrics are employed: the Variance Accounted For ($VAF$) [6], and the relative bias, $B_r(\vartheta)$. Both have been employed in previous works to verify the ARX results [10, 12], thus, these ones should be also applicable to the BJ results. The $VAF$ metric [45],

$$VAF^{n^*_k,m^*} = max\left\{0, \left(1 - \frac{\sum_{k=1}^{N} \mid u(t_k) - \hat{u}^{n^*_k,m^*}(t_k) \mid^2}{\sum_{k=1}^{N} \mid u(t_k) \mid^2}\right) \cdot 100\%\right\}, \tag{39}$$

evaluates the correctness of a model, by comparing the real output, $u(t_k)$, with the estimated output of the model, $\hat{u}^{n^*_k,m^*}(t_k)$. The $VAF$ of two signals that are the same is 100%, while it will be lower if they differ. For a certain combination of time delay and remnant filter order in the structure, the $VAF$ difference between ARX and BJ models can be used to compare the accuracy of both methods and analyze the effect of the BJ model remnant order on estimation results:

$$\Delta VAF^{n^*_k,m^*} = VAF^{n^*_k,m^*}_{BJ} - VAF^{n^*_k}_{ARX}. \tag{40}$$

On the other hand, the relative bias can be used to verify the accuracy of parameter estimation:

$$B_r(\vartheta) = \left(\frac{\hat{\vartheta} - \vartheta^0}{\vartheta^0}\right) \cdot 100\%, \tag{41}$$

where $\vartheta$ can be a discrete-time parameter from ARX or BJ models, or an HO coefficient.

# IV. Method

## A. Forcing Function

To accomplish adequate simulations, the forcing function needs to be defined properly. As proposed by Zaal [11], a summation of $N_t$ sinusoids with different amplitudes $A_k$, frequencies $\omega_k$ and phases $\phi_k$,

$$f_t(t) = \sum_{k=1}^{N_t} A_k \cdot \sin\left(\omega_k \cdot (t - t_0) + \phi_k\right), \tag{42}$$

is an effective option to excite the closed-loop system shown in Figure 1. To avoid leakage phenomena, the frequencies $\omega_k$ are multiples of the base frequency $\omega_{k,0} = 2\pi/T_m$, where $T_m$ is the total simulation time. Thus, the integer $n_t$ is defined as $n_t = \omega_k/\omega_{k,0}$. In addition, a transient interval time is introduced in the simulation, $t_0$, in order that control loop variables are stabilised, consequently, the sinusoids are initialized at $t = t_0$. Table 1 shows the coefficients of each sinusoid employed in simulations, while Figure 3 depicts its auto-Power Spectral Density (PSD) function, $S_{f_t f_t}(j\omega)$, and its peaks located at $\omega_k$.

**Table 1. Target function parameters for each component.**

| $k, -$ | $n_t, -$ | $\omega_k, rad/s$ | $A_k, deg$ | $\phi_k, rad$ |
|---|---|---|---|---|
| 1 | 3 | 0.230 | 1.186 | -0.753 |
| 2 | 5 | 0.384 | 1.121 | 1.564 |
| 3 | 8 | 0.614 | 0.991 | 0.588 |
| 4 | 13 | 0.997 | 0.756 | -0.546 |
| 5 | 22 | 1.687 | 0.447 | 0.674 |
| 6 | 34 | 2.608 | 0.245 | -1.724 |
| 7 | 53 | 4.065 | 0.123 | -1.963 |
| 8 | 86 | 6.596 | 0.061 | -2.189 |
| 9 | 139 | 10.661 | 0.036 | 0.875 |
| 10 | 229 | 17.564 | 0.025 | 0.604 |

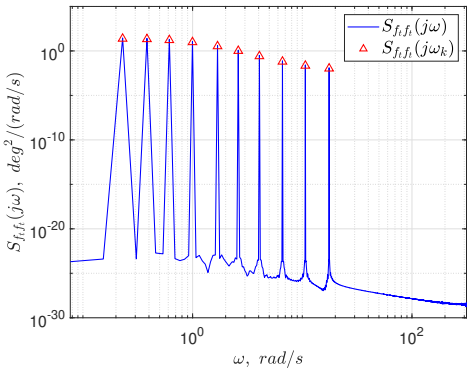


**Figure 3. Auto-PSD of forcing function $f_t(t)$.**

## B. Remnant Noise

Then, a remnant realisation is generated for every single simulation run by feeding the zero-mean GWN with unit variance through a filter, i.e., $\sigma_\varepsilon = 1$. Different $m^{th}$-order filters, $m \in \{1, 2, 3, 4\}$, and a remnant time constant, $T_n = 0.06$, can be simulated to address their influence on results (see [10, 11, 26]). Additionally, it is possible to introduce multiple noise levels $P_n \in [0.0, 1.0)$ in the system by modifying the remnant gain $K_n$. $P_n \in \{0.0, 0.1, 0.2, 0.3\}$ are considered in batch-fitting estimation, while $P_n \in \{0.01, 0.10, 0.20\}$ are used in recursive analysis. Consequently, a certain value for the $K_n$ parameter has to be selected to obtain a requested noise level at each simulation, for which the stochastics theory presented by Ljung [34] can be employed, so that a suitable formula is developed. From the noise level definition, $P_n = \sigma^2_{u_n}/\sigma^2_u$, based on the forcing function expression and HO and CE models:

$$K_n = \sqrt{\frac{P_n}{(1 - P_n) \cdot T_s} \frac{\frac{\pi}{2} \sum_{k=1}^{N_t} A_k^2 \left| \frac{H_{HO_e}(j\omega_k)}{1 + H_{HO_e}(j\omega_k) H_{CE}(j\omega_k)} \right|^2}{\int_0^{\pi/T_s} \frac{\mathrm{d}\omega}{\left| (T_n(j\omega)+1)^m (1 + H_{HO_e}(j\omega) H_{CE}(j\omega)) \right|^2}}}. \tag{43}$$

## C. Simulation Conditions

To create a time-varying simulation framework, two different states, $s_1$ and $s_2$, are defined on the basis of the CE dynamics variation and its effect on the human operator. Zaal [11] employs two sets of parameters for the CE dynamics and the assumed HO dynamics resulting from adaptation. In the state $s_1$, the HO×CE dynamics are given by a crossover frequency $\omega_c = 1.5\ rad/s$ and a phase margin $\phi_m = 77.0°$, while these two parameters are converted into $\omega_c = 2.8\ rad/s$ and $\phi_m = 22.7°$ in state $s_2$. When a ZOH discretization is applied, different discrete-time coefficients are obtained for each set. Furthermore, the corresponding remnant gain $K_n$ is defined for each state based on Equation (43). These parameters are recorded in Table 2:

**Table 2. CE, HO and ZOH-discretization parameters for states $s_1$ and $s_2$.**

| State | CE | | HO | | | | | | | ZOH | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $K_c, -$ | $\omega_b, rad/s$ | $K_e, -$ | $T_L, s$ | $K_{\dot{e}}, -$ | $\tau_e, -$ | $\omega_{nm}, rad/s$ | $\zeta_{nm}, -$ | $T_n, s$ | $a_1^{d,0}, -$ | $a_2^{d,0}, -$ | $b_0^{d,0}, -$ | $b_1^{d,0}, -$ |
| $s_1$ | 90 | 6.0 | 0.09 | 0.40 | 0.036 | 0.28 | 11.25 | 0.35 | 0.06 | -1.9121 | 0.9243 | 0.0443 | -0.0432 |
| $s_2$ | 30 | 0.2 | 0.07 | 1.20 | 0.084 | 0.28 | 11.25 | 0.35 | 0.06 | -1.9121 | 0.9243 | 0.1024 | -0.1016 |

Two types of simulation conditions are considered in this article based on the scenarios proposed by Zaal [11]: a constant set of parameters (C1-C2), or a time-varying scenario (C3-C6). In the first case, the state $s_1$ (C1) or $s_2$ (C2) is implemented during the entire simulation trial. The second case is defined by a change between states (performed by a sigmoid function in Equation (5)) $s_1 \rightarrow s_2$, in C3-C4 case, or $s_2 \rightarrow s_1$, in C5-C6. Two types of transitions are considered based on the sigmoid parameter $G$ (see Eq. (5)), i.e., a slow states change $G = 0.5\ s^{-1}$ in C3 and C5, and a fast transition $G = 100\ s^{-1}$ in C4 and C6. A transition from single- to double-integrator dynamics, together with an aggressive CE transition, i.e. $G = 100\ s^{-1}$, is applied in order to analyze the most challenging scenario, employing the simulation condition 'C4' in consequence. The HO time-delay and NMS parameters are assumed to remain constant during the simulation [10, 11]. Figure 4 shows the effect of each simulation condition on $K_e$ as an example:

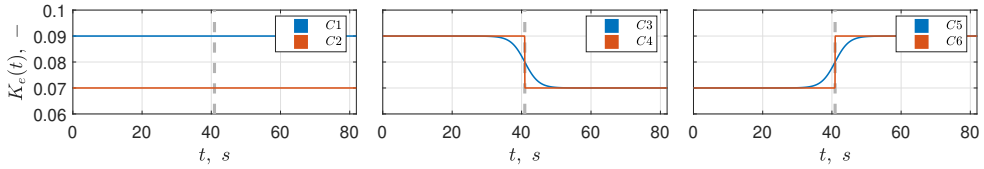


**Figure 4. HO gain for simulation conditions C1-C6 under simulation time $T_m = 81.92s$ and time of maximum rate of change $t_M = 40.96s$.**

## V. Simulation Results

Monte Carlo simulation results are addressed in this section. ARX and BJ estimators' performance is evaluated in a batch-fitting scenario under simulation condition C1 by OLS and PEM algorithms, respectively, so that the estimation accuracy and influence of model parameters, such as the remnant filter or time delay, and the noise level can be analyzed. Two cases of model remnant order $m^*$ are studied: an order that matches the simulation remnant order, i.e. $m^* = m^0 \in \{1, 2, 3, 4\}$, or a fixed first-order, i.e. $m^* = 1$. An optimal PEM algorithm with a BJ($m^* = 1$) structure is selected due to its efficiency and accuracy. Once the BJ estimator is studied, the RLS and RPEM algorithms are evaluated in a time-varying scenario (i.e. simulation condition C4) for ARX and BJ models, respectively. A total of $M = 100$ replications are evaluated in each case, which are shown in Box and Whiskers plots in batch-fitting scenario, or averaged in recursive estimation. A measurement time $T_m = 81.92\ s$ and transient time $t_0 = 10.00\ s$ are employed. Estimation processes are only conducted during the measurement time, after tracking error and control output signals are stabilized. The sample time is $T_s = 0.01\ s$.

### A. Batch-Fitting Estimation

*1. ARX and BJ Model with Remnant Order $m^* = m^0$*

Figure 5 shows that a large relative bias is obtained for $m^0 = 1$ due to poles mismatch in the ARX remnant filter structure, since it is modeled with the same discrete-time denominator used in the linear HO transfer function. Thus, ARX method sacrifices estimation accuracy to explain the remnant dynamics more precisely, by means of a filter structure unable to model a first-order one. However, when remnant filters with order $m^0 \geq 2$ are simulated, ARX bias is diminished to acceptable values because the number of model remnant filter poles is never higher than the number of poles in $H_n^m(s,t)$, although the estimation error in $b_0^d$ and $b_1^d$ is still around 50% in most cases.

Changes in model time delay (x-axis) produce a linear trend in computed relative bias since the ARX estimator tries to create or reduce lag. Hence, first-order integrator dynamics are achieved sooner for time delays $n_k^* < n_k^0 = 29$ by placing the NMS poles at a lower frequency, while additional lead is generated in cases when $n_k^* > n_k^0 = 29$.

Increments in noise level give as a result a greater bias in general, which affects the stability of the ARX estimator

severely. Since the model filter structure is wrong and imposes unrealistic poles, higher remnant levels will lead to an increase in estimation error. On the other hand, a null bias can be obtained in a remnant-free scenario if the ideal time delay is employed (this fact proves a discrete-time estimator is implemented properly).

As depicted in Figure 6, the BJ estimator is capable of improving on the ARX results considerably. The BJ model is initialized with coefficients close to the true simulated HO parameters to ensure convergence, although the PEM algorithm's flexibility with inaccurate initial conditions has been proven. The important bias found in first-order remnant order simulations is now highly reduced, while estimations with similar accuracy are achieved for different remnant orders.

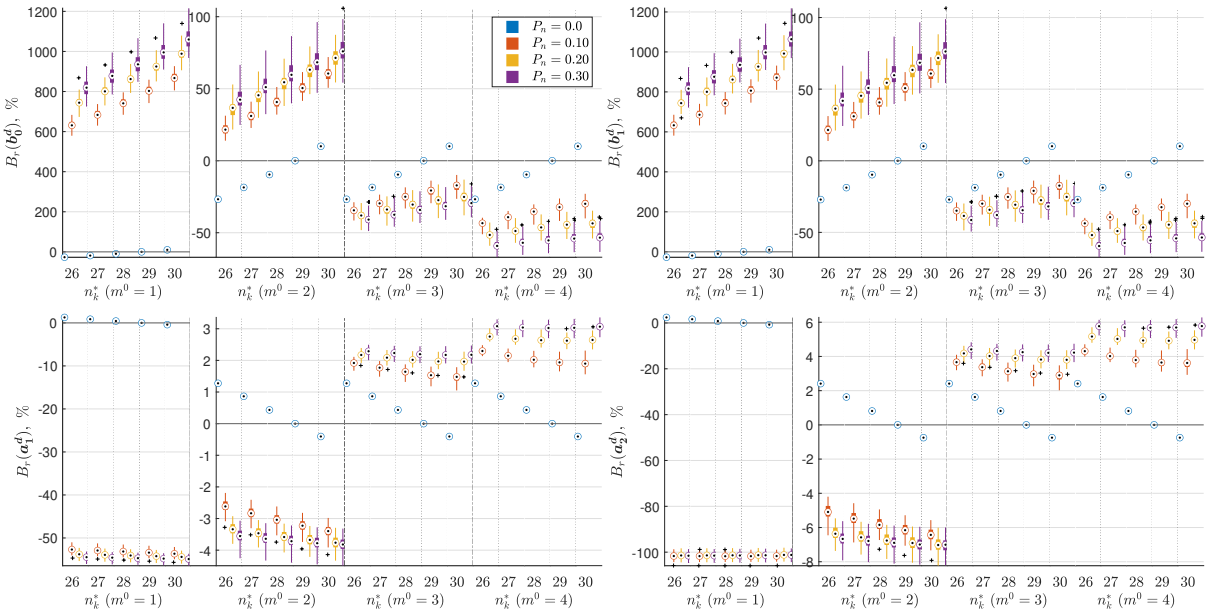


**Figure 5. Box and Whisker plots for relative bias of discrete-time parameters in ARX model.**

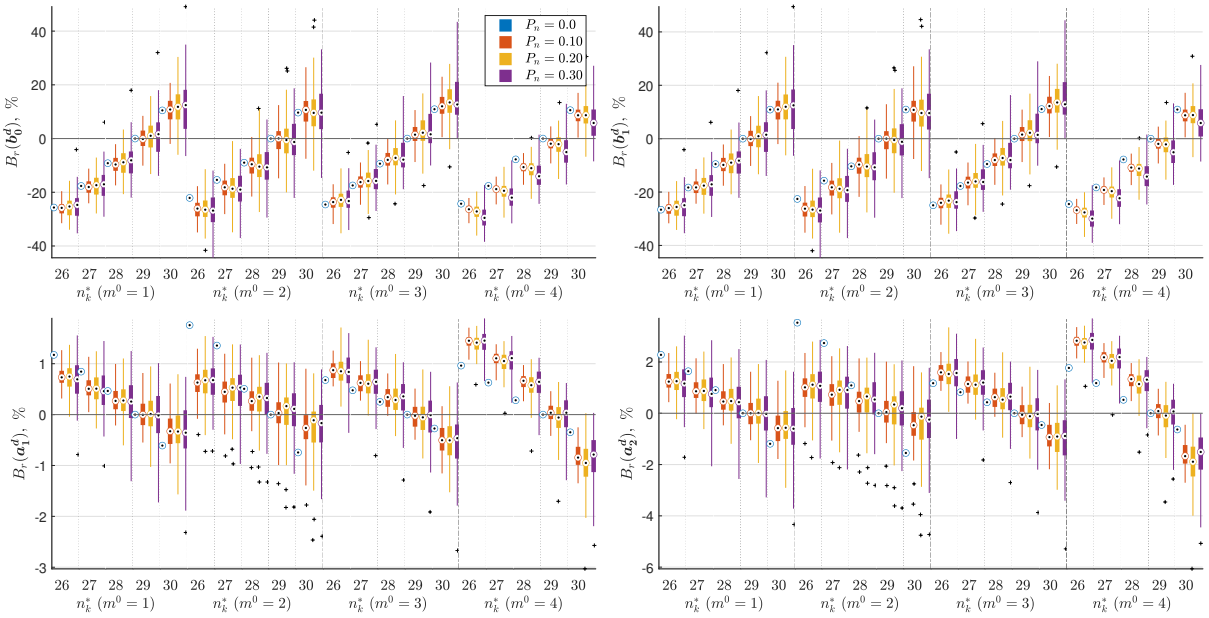


**Figure 6. Box and Whisker plots for relative bias of discrete-time parameters in BJ model with $m^* = m^0$.**

Since BJ structures allow different pole allocations in $H_{HO_e}(s,t)$ and $H_n^m(s,t)$, but also the same number of poles in simulated and model remnant filters, BJ estimators are quite adaptive to create discrete-time structures that fit each HO model adequately. For remnant orders $m^0 \geq 2$, ARX bias results present comparable values to BJ ones, nevertheless, BJ estimators still provide a response with around 50% less estimation error.

Similar to the ARX case, the BJ estimator creates lag or lead depending on the assumed model time delay, hence, a linear trend is also found. Nonetheless, relative bias values are close to zero for increasing values in noise level, which ensures an ideal HO identification. Only small changes in $b_0^d$ and $b_1^d$ are found for $m^0 = 4$. In contrast to ARX models, relative bias results oscillate around the value obtained for the remnant-free scenario in BJ estimators. Therefore, BJ structures represent a more consistent, robust alternative in batch-fitting cases, since the simulation remnant order and noise level do not affect the accuracy of estimation.

*2. ARX and BJ Model with Remnant Order $m^* = 1$*

Although the PEM algorithm for BJ model structures has been proven to be an adequate, precise estimation method in the discrete-time domain, more information about the effect of the model remnant order is required to develop a suitable, efficient estimator. Consequently, the $VAF$ metric is evaluated in Figure 7 for multiple BJ model time delays and remnant orders in a $m^0 = 3$ scenario, while it is also compared to the $VAF$ results for ARX models in Figure 8. The BJ estimator presents a homogeneous average $VAF$ value for all model remnant orders, and differences are only found for different time delays and noise levels. However, in scenarios with $m^0 = 1$, it has been observed a poorer performance when high model remnant orders are utilized. No significant differences are found between BJ and ARX for cases with a time delay close to $n_k^* = 29$. Hence, the use of BJ models with $m^* = 1$ is recommended to reduce the degrees of freedom in these structures without a cost in estimation accuracy.

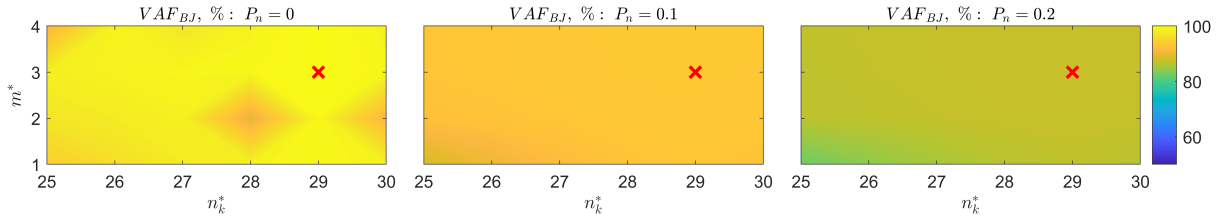


**Figure 7. $VAF$ results for BJ model with multiple model time delay $n_k^*$ and remnant order $m^*$. Simulation time delay $n_k^0 = 29$ and remnant order $m^0 = 3$ (red cross).**

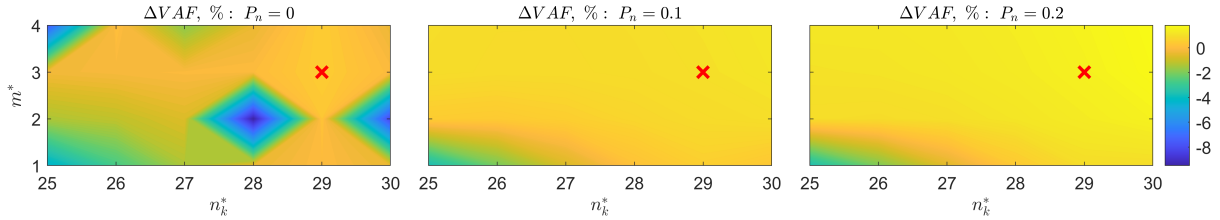


**Figure 8. Difference in $VAF$ results between BJ and ARX models with multiple model time delay $n_k^*$ and remnant order $m^*$. Simulation time delay $n_k^0 = 29$ and remnant order $m^0 = 3$ (red cross).**

Such a hypothesis about the use of a constant model remnant order $m^* = 1$ can be also validated by the relative bias analysis, shown in Figure 9. Equivalent results are found when compared to Figure 6, and the bias in $m^0 = 4$ is even slightly diminished. BJ structures with $m^* = 1$ present a discretized remnant filter directly obtained from the Backward-Euler method, while the number of parameters in $H_n^m(s,t)$ to be estimated differs from the number of $d^d$ coefficients, which makes the estimation more robust when $m^* = 1$.

Bode plots of ARX and BJ models for multiple noise levels and simulation remnant orders, when the ideal time delay is selected, are shown in Figure 10. The real $H_{HO_e}$ dynamics are represented by the ZOH discretization. There is a clear difference between both model structures, which can be explained by the relative bias results presented previously. The zero is located at lower frequencies for higher noise levels in ARX results, obtaining a higher magnitude peak. The ARX structure fails to capture the NMS poles in a first-order remnant filter, while these ones are moved to lower

frequencies when being estimated for higher remnant orders. ARX models attempt to explain remnant dynamics by generating lag when the remnant order is increased, giving as a result a wrong estimation. This issue is not present in the BJ case owing to the flexibility of its model structure, accomplishing almost 100% accurate estimations in all scenarios.

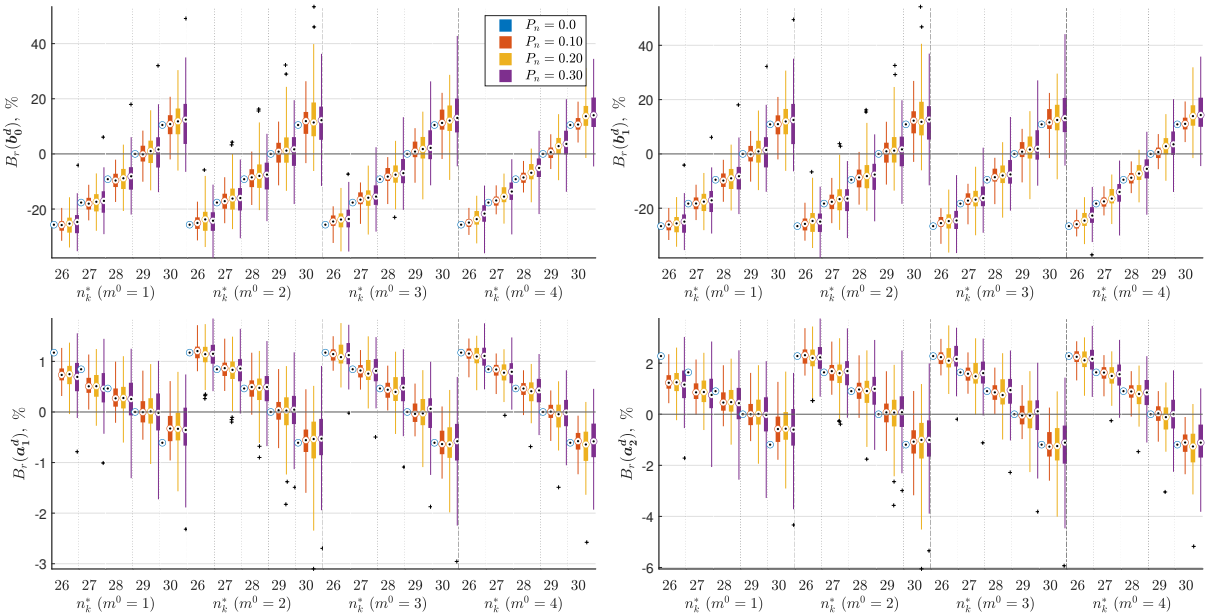


**Figure 9. Box and Whisker plots for relative bias of discrete-time parameters in BJ model with $m^* = 1$.**

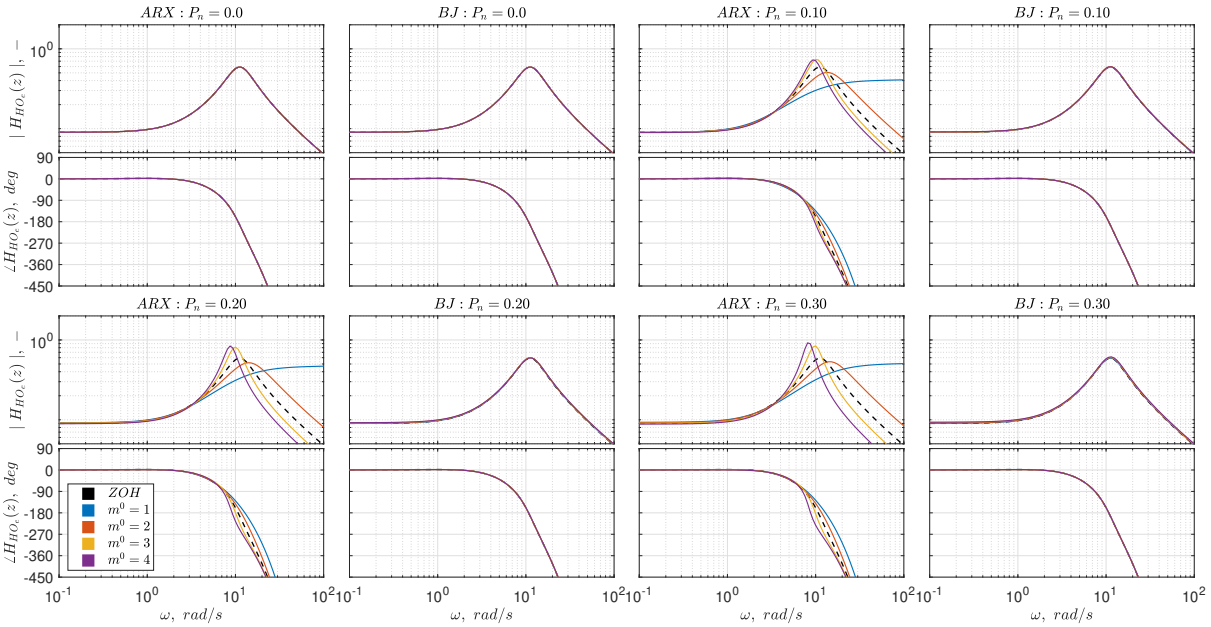


**Figure 10. Bode plots for ARX and BJ models with model time-delay $n_k^* = n_k^0 = 29$ and remnant filter order $m^* = 1$. Reference ZOH discretization (black, dashed line).**

## B. Recursive Estimation

To estimate ARX and BJ models recursively, RLS and RPEM algorithms with a forgetting factor $\lambda = 0.996609$ are employed, respectively. The covariance matrix is initialized as shown in Equation 38 in both methods. Two batch-fitting

estimations are performed for time intervals $[0, 30.72]$ s and $[51.20, 81.92]$ s to capture $s_1$ and $s_2$ dynamics, based on [10]. The first OLS batch-fitting outcome is utilized to initialize the RLS algorithm (i.e. OLS→RLS). Also, OLS estimations are used as initial conditions for the PEM algorithm, afterwards, these converged batch-fitting coefficients are employed in the RPEM initialization (i.e. OLS→PEM→RPEM). Through this strategy, the recursive BJ algorithm is independent of initial conditions set by the user.

Figure 11 depicts the estimation of the discrete-time parameters from RLS and RPEM for multiple remnant orders and noise levels. In the $m^0 = 1$ case, ARX models lead to significant relative bias while BJ provides a much more precise estimation with almost null bias after full convergence, similar to what is shown previously in Figures 5 and 9. For remnant orders $m^0 \geq 2$, ARX improves its results, but it still shows a considerable bias with respect to BJ outcome. Therefore, batch-fitting results observed in previous figures also represent the real functioning of RLS and RPEM algorithms in recursive estimation.

Nonetheless, the recursive BJ estimator presents more difficulties in achieving convergence for high noise levels. While the RLS algorithm adapts quite fast to changes in simulation parameters, and provides a stable estimation once those parameters are fixed, the BJ structure needs 500% more time to process, which may turn into divergence problems due to excessive oscillations in some cases. The RPEM algorithm denotes a slower adaptation capacity, making it more difficult for the recursive estimation to converge to each sub-batch-fitting result.

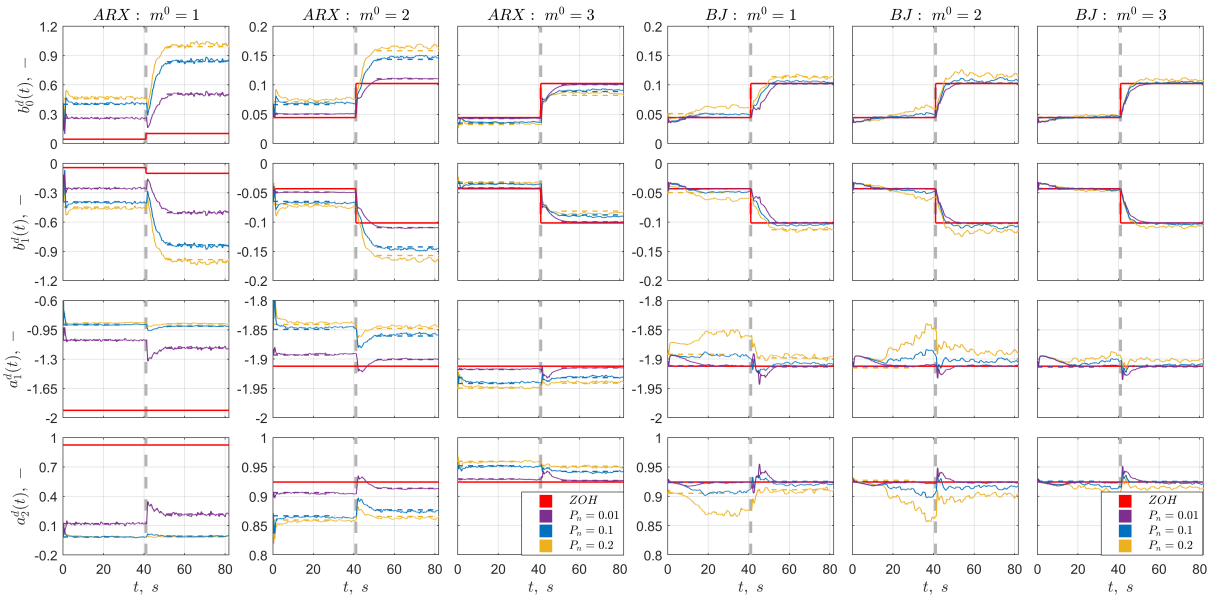


**Figure 11. Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line).**

Figure 12 shows the identified HO coefficients by ARX and BJ model structures. The RLS algorithm presents considerable errors in the NMS coefficients for remnant order $m^0 = 1$ simulations, which are mainly produced by highly biased discrete-time estimations. When the discrete-time coefficients are obtained, the discrete-time state-space system form is calculated for the transfer function $B(z^{-1})/A(z^{-1})$. Then, Gajic's procedure [46] is used reversely to get the continuous-time state-space system by means of the logarithm of its extended matrix. To obtain a successful conversion of the state-space system, its associated extended matrix must be invertible and with no negative real eigenvalues [46]. Hence, the following discrete-time parameter constraints must be respected to ensure a proper state-space conversion: $\{a_1^d < 0, a_2^d > 0\}$.

When simulations with $m^0 \geq 2$ are conducted, the bias in ARX estimations is reduced by almost 10 times, although it is still persistent in NMS coefficients. However, ARX offers more stable, precise results than BJ in $K_e$, $T_L$ and $K_{\dot{e}}$, while the BJ model provides a more adequate outcome in NMS parameters. Particularly, great oscillations are given in $T_L$ estimations from BJ structures. Thus, ARX is able to modify its discrete-time estimations to focus on gain and zero estimation mainly, at the cost of permanent NMS bias. On the other hand, BJ gives the same priority to all coefficients, based on its non-linear optimization process. In addition, an additional error is produced when $m^0 = 2$ in BJ estimations

since coupling can occur in poles of $H_{HO_e}(s,t)$ and $H_n^m(s,t)$.

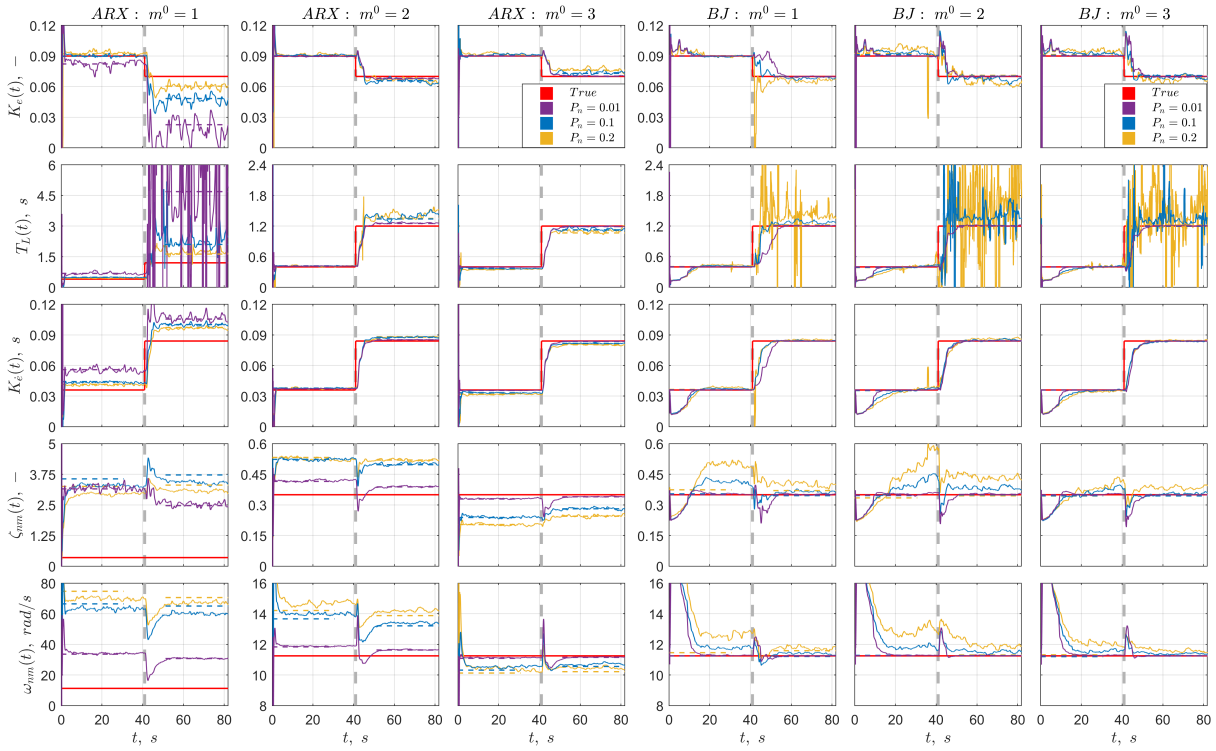


**Figure 12. Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line).**

## VI. Experimental Results

Experimental results obtained in an experiment in the SIMONA Research Simulator at Delft University of Technology by Van Grootheest et al. [10] are analyzed in this section, following a similar estimation strategy explained in Section V.B. Three subjects conducted the single-axis compensatory tracking task detailed in Figure 1. The run-in time is $t_0 = 8.08\ s$, while the measurement time $T_m = 81.92\ s$. A total of 7 runs were performed, from which the last 5 ones are evaluated. The sample time is $T_s = 0.01\ s$. Only the simulation condition C4 results are studied below, in which a fast transition from state $s_1$ to $s_2$ is given (see Section IV.C).

A model remnant order $m^* = 1$ is employed, while the model time delay $n_k^*$ and remnant time constant $T_n$ are computed a priori by means of the average between the best two sub-batch-fitting estimations made. Thus, specific $n_k^*$ and $T_n^*$ are used in each run and model structure, which are shown in Table 3.

**Table 3. Model parameters $n_k^*$ and $T_n^*$ used in each run and model structure.**

| Model | Subject 1 | | | | | Subject 2 | | | | | Subject 3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parameters | N1 | N2 | N3 | N4 | N5 | N1 | N2 | N3 | N4 | N5 | N1 | N2 | N3 | N4 | N5 |
| ARX: $n_k^*, -$ | 21 | 24 | 22 | 28 | 20 | 24 | 22 | 20 | 25 | 24 | 25 | 24 | 24 | 24 | 26 |
| BJ: $n_k^*, -$ | 30 | 30 | 31 | 32 | 31 | 26 | 28 | 26 | 29 | 27 | 28 | 30 | 26 | 27 | 27 |
| BJ: $T_n^*, s$ | 0.16 | 0.13 | 0.16 | 0.13 | 0.11 | 0.17 | 0.11 | 0.13 | 0.06 | 0.14 | 0.10 | 0.05 | 0.08 | 0.15 | 0.05 |

Figures 13 and 14 show the discrete-time and HO model coefficients estimation results for ARX and BJ, respectively. In all subjects, both recursive algorithms are capable of detecting HO adaptation, although subject 1 does not seem to generate as much lead in the state $s_2$ as subjects 2 and 3. As explained in Section V, the RPEM algorithm requires more

time to achieve convergence when facing changes in HO parameters, hence, this fact has a strong impact on the BJ performance in real life, seeing that the human operator is not a steady, perfect controller.

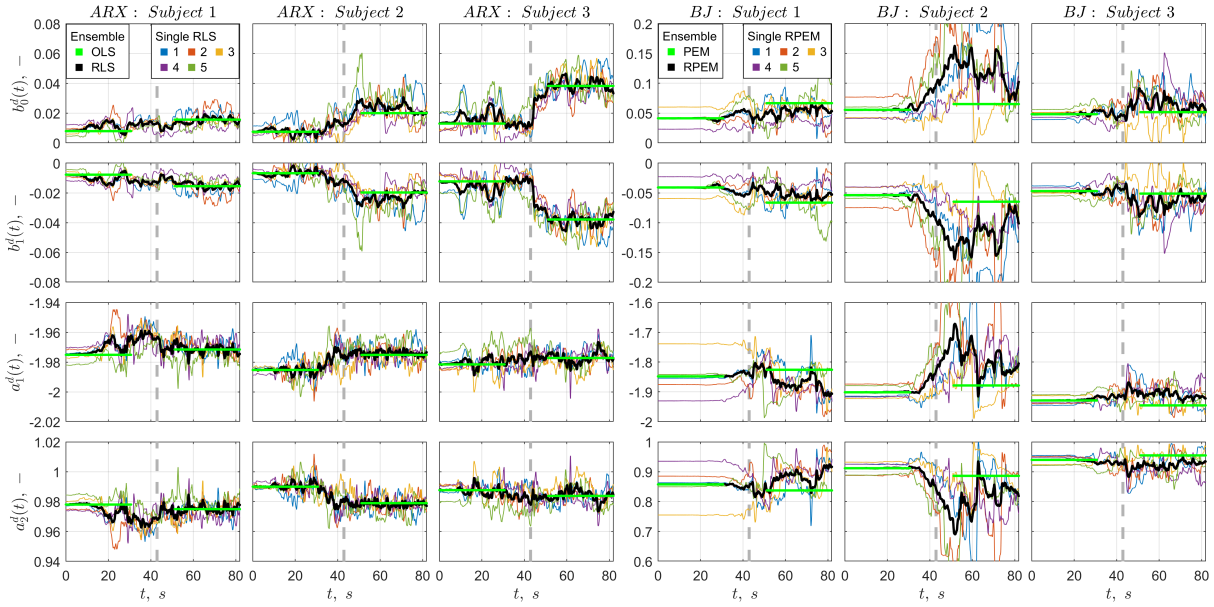


**Figure 13. Experimental results of recursive ARX and BJ algorithms in discrete-time parameters.**

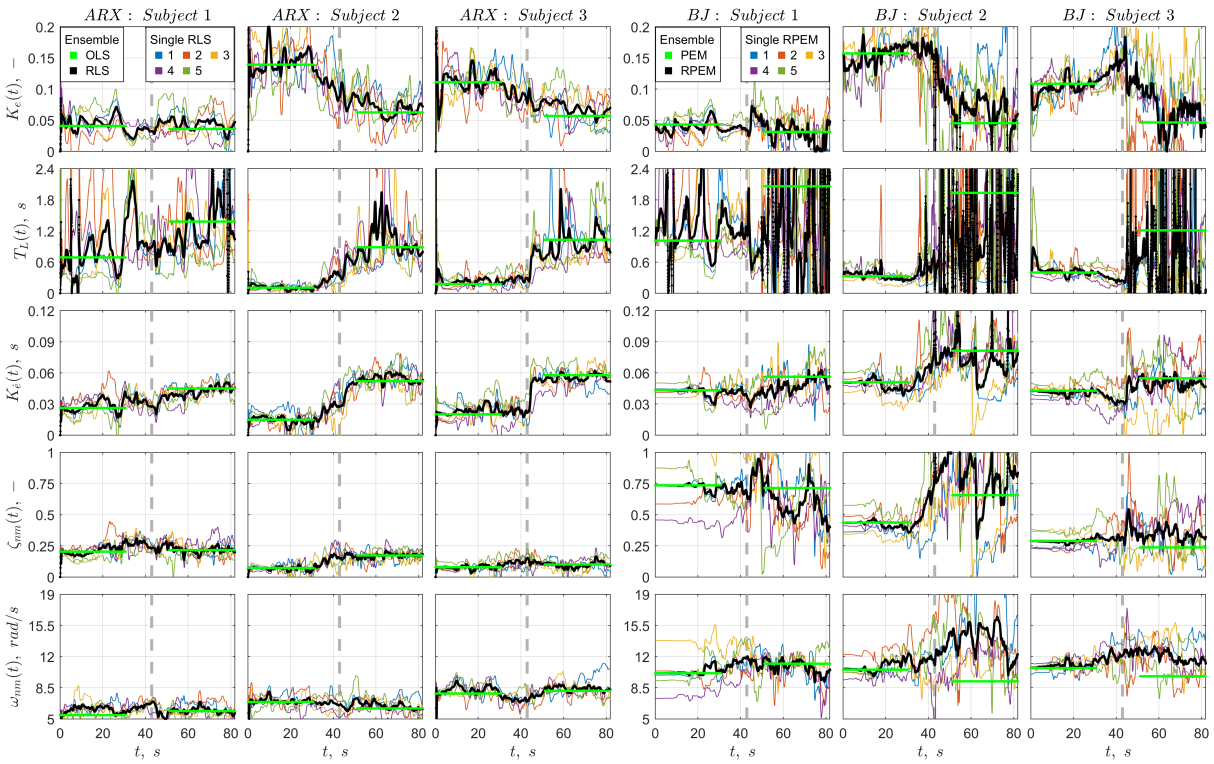


**Figure 14. Experimental results of recursive ARX and BJ algorithms in HO coefficients.**

Magnitudes of discrete-time parameter estimations are different for ARX and BJ, as can be observed in Figure 13. This fact can be due to the effort of ARX structures in focusing more on HO gain and zero estimation, while RPEM

follows an optimization for all model coefficients equally. Modifications in $a_1^d$ and $a_2^d$ coefficients magnitudes directly imply an adjustment in $b_0^d$ and $b_1^d$, and vice versa. Additionally, incorrect estimations in $n_k^*$ in ARX and BJ models also affect the discrete-time parameter magnitudes.

Oscillations in estimated parameters are more noticeable in BJ than ARX. Since the RPEM algorithm follows a non-linear optimization process, more noise is found in BJ estimations than whether the RLS was used, particularly in the $T_L$ coefficient. BJ model covariance matrix starts to increase from the start of estimation, since more uncertainty is given about the accuracy of predicted parameters, which turns into convergence issues when the state $s_2$ is being evaluated.

In addition, estimated NMS coefficients are supposed not to show high oscillations during the entire simulation (relatively small changes are given in real life), but the RPEM algorithm provides an outcome with greater fluctuations. Nevertheless, the average NMS estimations in BJ structures show higher values than in ARX models, which denotes that a considerable bias may be generated by the RLS algorithm.

## VII. Discussion

A novel time-varying HO identification method based on recursive BJ model structures is developed, which tries to make a precise fit of the HO coefficients that define its dynamics in single-axis manual-control tasks. An optimal version of the algorithm is found for a structure with a model remnant order $m^* = 1$, based on batch-fitting results. The RPEM algorithm, implemented with a constant forgetting factor $\lambda$, shows acceptable results when being tested on simulation data, since it is able to correctly track HO adaptations to CE dynamics transitions proposed by Zaal [11]. Through testing on experimental data, the recursive BJ is capable of detecting adaptation in subjects, while providing HO coefficient estimations whose magnitudes appear feasible. BJ results in batch and recursive fitting are compared to ARX ones in order to analyze the advantages and drawbacks of estimating HO coefficients by PEM or RPEM algorithms instead of OLS and RLS.

BJ structures have great potential as identification method, however, they present several problems that can affect their applicability. Firstly, the PEM algorithm is dependent on initial conditions, and although it offers adequate flexibility, initial conditions with high deviations from real values may lead to convergence failure. This fact is quite relevant regarding PEM initialization by a prior OLS estimation, since it is proven that ARX batch-fitting could provide a highly biased output. Furthermore, despite the BJ capability of detecting HO adaptation, the convergence time required can be even 5 times higher due to the non-linear optimization procedure it follows, which can cause identification problems in real life due to the fast changes in HO dynamics. Noticeable fluctuations can be found in recursive BJ estimations, mainly in the parameter $T_L$, and in NMS coefficients that should be steady during the entire test.

When comparing ARX and BJ model structures, BJ offers a 5000% more accurate estimation when the remnant order is $m^* = 1$, owing to the mismatch of the ARX model in the remnant filter poles. The BJ model can also prevent the persistent bias found in ARX estimations, because of the versatility of BJ structures when facing different remnant dynamics. Thus, RPEM can provide accurate results in NMS coefficients once the algorithm is converged, while RLS reaches its final estimation fast, but it will always have a bias. Nevertheless, ARX is able to modify the discrete-time parameters in order to still reliably estimate $K_e$, $T_L$ and $K_{\dot{e}}$, achieving a more adequate, steady fit than BJ for considerably high noise levels.

Definitely, there is room for improvement in the RPEM algorithm implementation. For instance, the covariance matrix could be updated based on $VAF$ from a validation data set, in order to reduce the strong oscillations that are given in state $s_2$. A forgetting matrix RPEM algorithm could also be implemented, although no MATLAB functions are available for this setup, so that the algorithm should be built based on the theory explained in Section III. Such a forgetting matrix could weigh all samples in discrete-time poles estimation, while applying a certain time horizon for $b_0^d$ and $b_1^d$, as conducted in [10]. Another strategy would consist of applying the PEM algorithm to sub-batches of data recursively, since this method has shown really high accuracy in all HO coefficient estimations. However, divergence problems may occur due to wrong initialization, or whether the sub-batches are too small. Additionally, other recursive BJ algorithms could be explored, such as the RRIV, which can be implemented through the Captain Toolbox [41]. In particular, this could solve the issues of dependency on initial conditions and convergence rate.

Once the recursive BJ algorithm is refined, conducting a decimation analysis is important to verify the BJ applicability when different sample times are used, as shown by Van Grootheest et al. [10]. When $n_k$ can not be a multiple of the sample time $T_s$, or a time-varying delay is given, it is necessary to implement a Padé approximation [47] that represents the effect of time delay by a transfer function. This approximation increases the number of discrete-time parameters to be estimated, thus, it may affect the accuracy and convergence of recursive BJ algorithms. Moreover, other models for

remnant dynamics should be evaluated to test the performance of the RPEM algorithm in a real-life scenario.

Experimental results are also affected by the time delay estimation, which may not be optimal in the calculations performed in this article. It is more convenient to have an independent time delay estimation algorithm [12, 22] that provides an acceptable recursive identification for each subject. This way, the predicted time delay can be introduced into the BJ structure in the RPEM algorithm. In addition, the model remnant time constant $T_n$ could be predicted by means of other methods, although it does not have much effect on the estimation accuracy.

Based on the comparison of RLS and RPEM performance as shown in this paper, an 'identification method fusion' could be an interesting alternative. ARX has been proven to give adequate results for the HO gain and lead contribution, while BJ can remove the bias in NMS coefficients. An ideal identification technique should have the precision of BJ, but also the adaptation velocity of ARX. Therefore, RPEM could be implemented with a forgetting matrix, so NMS estimations performed by RLS can be adjusted based on recursive BJ calculations, diminishing the persistent bias in these coefficients.

Moreover, since high oscillations are frequently given in the identified coefficients, the utilization of a moving average would be advisable, and particularly, when the RPEM is applied to experimental data. Thus, important fluctuations that affect the stability of the estimation could be filtered out. This method could be quite useful in the time constant $T_L$ specially.

## VIII. Conclusion

This article lays the foundation for the development of an identification method based on Box-Jenkins structures that is able to provide an adequate estimation of the time-varying adaptation of a human operator in compensatory tracking tasks. A Monte Carlo simulation, based on the conditions presented by Zaal [11], is conducted to evaluate the performance of the Prediction Error Method (PEM) algorithm, in batch-fitting estimation, and the Recursive Prediction Error Minimization (RPEM) in recursive. The recursive Box-Jenkins (BJ) estimator employs a constant forgetting factor $\lambda = 0.99609$, based on Van Grootheest's research [10], and a model remnant order $m^* = 1$. Both Recursive Least Squares (RLS) and RPEM are tested with experimental data from three subjects. All results are compared to ARX outcome.

In batch-fitting, BJ fixes the bias problem in all parameters found in ARX results. The remnant filter order no longer meaningfully affects the bias and accuracy in BJ estimation, while it clearly does for ARX. Furthermore, the use of a fixed model remnant order $m^* = 1$ does not change the bias obtained by BJ, which makes it a suitable model structure configuration. Additionally, BJ presents more robustness than ARX when facing high noise levels.

Regarding recursive estimation, BJ offers a reduction in persistent bias found in identified coefficients, while ARX converges to its final estimation 5 times faster, reaching a quicker adaptation to changes in HO parameters. Except for a first-order remnant, the RLS algorithm usually captures the HO gain and zero with 0-5% less bias, but BJ is able to approximate the real Neuro-Muscular (NMS) dynamics with 25% less error. When a $m^0 = 1$ simulation is conducted, the BJ model clearly outperforms ARX. Through experimental testing, it is proven that the slow adaptation of the recursive BJ can lead to excessive fluctuations in the identified HO coefficients. In addition, BJ shows the expected adaptation of $K_e$, $T_L$ and $K_{\dot{e}}$ less clearly than the ARX outcomes.

The RPEM algorithm contributes to the human-machine systems research field of study by providing novel non-linear procedures that can improve the current estimation methodologies in certain aspects, such as batch-fitting or recursive estimation in first-order remnant scenarios. For instance, an advanced identification technique could be based on an RLS algorithm powered by an improved version (i.e. with optimized convergence) of the recursive BJ method, which provides a more accurate, stable NMS estimation.

## References


[1] McRuer, D. T., "Human Pilot Dynamics in Compensatory Systems," *Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, OH*, 1965. https://doi.org/10.21236/ad0470337.

[2] McRuer, D. T., and Weir, D. H., "Theory of Manual Vehicular Control," *IEEE Transactions on Man-Machine Systems*, Vol. 10, No. 4, 1969, pp. 257–291. https://doi.org/10.1109/TMMS.1969.299930.

[3] McRuer, D. T., and Jex, H., "A Review of Quasi-Linear Pilot Models," *IEEE Transactions on Human Factors in Electronics*, Vol. HFE-8, No. 3, 1967, pp. 231–249. https://doi.org/10.1109/THFE.1967.234304.

[4] McRuer, D. T., Graham, D., and Krendel, E. S., "Manual control of single-loop systems: Part I," *Journal of the Franklin Institute*, Vol. 283, No. 1, 1967, pp. 1–29. https://doi.org/10.1016/0016-0032(67)90112-3.

[5] McRuer, D. T., Graham, D., and Krendel, E. S., "Manual control of single-loop systems: Part II," *Journal of the Franklin Institute*, Vol. 283, No. 2, 1967, pp. 145–168. https://doi.org/10.1016/0016-0032(67)90231-1.

[6] Nieuwenhuizen, F. M., Zaal Pmt, M., Mulder, Zaal, P. M. T., Mulder, M., van Paassen, M., and Mulder, J., "Modeling Human Multichannel Perception and Control Using Linear Time-Invariant Models," *Journal of Guidance Control and Dynamics*, 2008. https://doi.org/10.2514/1.32307.

[7] van Paassen, M., and Mulder, M., "Identification of Human Operator Control Behaviour in Multiple-Loop Tracking Tasks," *IFAC Proceedings Volumes*, Vol. 31, No. 26, 1998, pp. 455–460. https://doi.org/10.1016/S1474-6670(17)40135-2, 7th IFAC Symposium on Analysis, Design and Evaluation of Man-Machine Systems (MMS'98), Kyoto, Japan, 16-18 September 1998.

[8] Stapleford, R., McRuer, D., and Magdaleno, R., "Pilot Describing Function Measurements in a Multiloop Task," *IEEE Transactions on Human Factors in Electronics*, Vol. HFE-8, No. 2, 1967, pp. 113–125. https://doi.org/10.1109/THFE.1967.233628.

[9] Young, L. R., "On Adaptive Manual Control," *Ergonomics*, Vol. 12, No. 4, 1969, pp. 635–674. https://doi.org/10.1080/00140136908931083.

[10] van Grootheest, A., Pool, D. M., van Paassen, M., and Mulder, M., "Identification of Time-Varying Manual-Control Adaptations with Recursive ARX Models," *AIAA*, 2018. https://doi.org/10.2514/6.2018-0118.

[11] Zaal, P. M., "Manual Control Adaptation to Changing Vehicle Dynamics in Roll-Pitch Control Tasks," *Journal of Guidance Control and Dynamics*, 2016. https://doi.org/10.2514/1.g001592.

[12] Plaetinck, W., Pool, D. M., van Paassen, M., and Mulder, M., "Online Identification of Pilot Adaptation to Sudden Degradations in Vehicle Stability," *IFAC-PapersOnLine*, Vol. 51, No. 34, 2019, pp. 347–352. https://doi.org/10.1016/j.ifacol.2019.01.020, 2nd IFAC Conference on Cyber-Physical and Human Systems CPHS 2018.

[13] Hess, R. A., "Modeling Human Pilot Adaptation to Flight Control Anomalies and Changing Task Demands," *Journal of Guidance Control and Dynamics*, 2016. https://doi.org/10.2514/1.g001303.

[14] Hess, R. A., "Modeling Pilot Control Behavior with Sudden Changes in Vehicle Dynamics," *Journal of Aircraft*, 2009. https://doi.org/10.2514/1.41215.

[15] Mulder, M., Pool, D. M., Abbink, D. A., Boer, E. R., Zaal, P. M. T., Drop, F. M., van der El, K., and van Paassen, M. M., "Manual Control Cybernetics: State-of-the-Art and Current Trends," *IEEE Transactions on Human-Machine Systems*, Vol. 48, No. 5, 2018, pp. 468–485. https://doi.org/10.1109/THMS.2017.2761342.

[16] Popovici, A., Zaal, P. M. T., and Pool, D. M., "Dual Extended Kalman Filter for the Identification of Time-Varying Human Manual Control Behavior," *AIAA*, 2017. https://doi.org/10.2514/6.2017-3666.

[17] Ameyoe, A., Chevrel, P., Le-Carpentier, E., Mars, F., and Illy, H., "Identification of a Linear Parameter Varying Driver Model for the Detection of Distraction," *IFAC-PapersOnLine*, Vol. 48, No. 26, 2015, pp. 37–42. https://doi.org/10.1016/j.ifacol.2015.11.110, 1st IFAC Workshop on Linear Parameter Varying Systems LPVS 2015.

[18] Olivari, M., Nieuwenhuizen, F. M., Bülthoff, H. H., and Pollini, L., "Identifying time-varying neuromuscular system with a recursive least-squares algorithm: a Monte-Carlo simulation study," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014, pp. 3573–3578. https://doi.org/10.1109/SMC.2014.6974484.

[19] Zaal, P. M. T., and Sweet, B. T., "Estimation of Time-Varying Pilot Model Parameters," *AIAA*, 2011. https://doi.org/10.2514/6.2011-6474.

[20] Zaal, P. M. T., Pool, D. M., Chu, Q., van Paassen, M., Mulder, M., and Mulder, J., "Modeling Human Multimodal Perception and Control Using Genetic Maximum Likelihood Estimation," *Journal of Guidance Control and Dynamics*, 2009. https://doi.org/10.2514/1.42843.

[21] Duarte, R., Pool, D. M., van Paassen, M., and Mulder, M., "Experimental Scheduling Functions for Global LPV Human Controller Modeling," *IFAC-PapersOnLine*, Vol. 50, No. 1, 2017, pp. 15853–15858. https://doi.org/10.1016/j.ifacol.2017.08.2329.

[22] Boer, E., and Kenyon, R., "Estimation of time-varying delay time in nonstationary linear systems: an approach to monitor human operator adaptation in manual tracking tasks," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol. 28, No. 1, 1998, pp. 89–99. https://doi.org/10.1109/3468.650325.

[23] Jiao, J., Sun, L., Tan, W., Xu, S., and Liu, X., "Identifying Pilot Control Adaptations to Sudden Changes in Aircraft Dynamics," *Journal of Guidance Control and Dynamics*, 2023. https://doi.org/10.2514/1.g007358.

[24] Drop, F. M., Pool, D. M., Mulder, M., and Bulthoff, H. H., "Constraints in Identification of Multi-Loop Feedforward Human Control Models," *IFAC-PapersOnLine*, Vol. 49, No. 19, 2016, pp. 7–12. https://doi.org/10.1016/j.ifacol.2016.10.444, 13th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems HMS 2016.

[25] Linssen, M., *Identifying Time-Varying Multimodal Manual Control Using Recursive ARX Model Techniques*, TU Delft Library, 2020. URL http://resolver.tudelft.nl/uuid:442f4308-0ea2-41a5-b38c-ee6b1a289f78.

[26] Pool, D. M., Pais, A. V., Vroome, A. D., van Paassen, M., and Mulder, M., "Identification of Nonlinear Motion Perception Dynamics Using Time-Domain Pilot Modeling," *Journal of Guidance Control and Dynamics*, 2012. https://doi.org/10.2514/1.56236.

[27] Lone, M., and Cooke, A., "Review of pilot models used in aircraft flight dynamics," *Aerospace Science and Technology*, Vol. 34, 2014, pp. 55–74. https://doi.org/10.1016/j.ast.2014.02.003.

[28] Metz, L. D., "A Time-Varying Approach to the Modeling of Human Control Remnant," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 12, No. 1, 1982, pp. 24–35. https://doi.org/10.1109/TSMC.1982.4308772.

[29] Levison, W. H., Baron, S., and Kleinman, D. L., "A Model for Human Controller Remnant," *IEEE Transactions on Man-Machine Systems*, Vol. 10, No. 4, 1969, pp. 101–108. https://doi.org/10.1109/TMMS.1969.299906.

[30] van der El, K., Pool, D. M., and Mulder, M., "Analysis of Human Remnant in Pursuit and Preview Tracking Tasks," *IFAC-PapersOnLine*, Vol. 52, No. 19, 2019, pp. 145–150. https://doi.org/10.1016/j.ifacol.2019.12.165, 14th IFAC Symposium on Analysis, Design, and Evaluation of Human Machine Systems HMS 2019.

[31] Olivari, M., Venrooij, J., Nieuwenhuizen, F. M., Pollini, L., and Bulthoff, H. H., "Identifying Time-Varying Pilot Responses: A Regularized Recursive Least-Squares Algorithm," *AIAA*, 2016. https://doi.org/10.2514/6.2016-1182.

[32] Klyde, D. H., Brenner, M. J., and Thompson, P., "Wavelet-based time-varying human operator models." *AIAA*, 2001. https://doi.org/10.2514/6.2001-4009.

[33] Mandal, T., and Gu, Y., "Online Pilot Model Parameter Estimation Using Sub-Scale Aircraft Flight Data," *AIAA*, 2016. https://doi.org/10.2514/6.2016-0636.

[34] Ljung, L., *System Identification: Theory for the User, 2nd Edition*, Prentice Hall Information and System Sciences Series, 2012. https://doi.org/10.1109/MRA.2012.2192817.

[35] Islam, S. A. U., and Bernstein, D. S., "Recursive Least Squares for Real-Time Implementation [Lecture Notes]," *IEEE Control Systems Magazine*, Vol. 39, No. 3, 2019, pp. 82–85. https://doi.org/10.1109/MCS.2019.2900788.

[36] Franklin, G. F., Powell, J., and Workman, M. L., *Digital Control Of Dynamic Systems 3rd Edition*, Ellis-Kagle Press, 1998.

[37] Butcher, J. C., *Numerical methods for ordinary differential equations*, John Wiley & Sons, Ltd, 2003. https://doi.org/10.1002/9781119121534.

[38] Tangirala, A. K., *Principles of System Identification: Theory and Practice*, Taylor & Francis Group, 2017. https://doi.org/10.1201/9781315222509.

[39] Hess, R. A., and Mnich, M. A., "Identification of pilot-vehicle dynamics from in-flight tracking data," *Journal of Guidance Control and Dynamics*, 1986. https://doi.org/10.2514/3.20129.

[40] Young, P. C., *Recursive Estimation and Time-Series Analysis, 2nd Edition*, Springer, 2011. https://doi.org/10.1007/978-3-642-21981-8.

[41] Young, P. C., "Refined instrumental variable estimation: Maximum likelihood optimization of a unified Box–Jenkins model," *Automatica*, Vol. 52, 2015, pp. 35–46. https://doi.org/10.1016/j.automatica.2014.10.126.

[42] Box, G. E. P., and Jenkins, G. M., *Time series analysis forecasting and control*, 5$^{th}$ ed., John Wiley & Sons, Inc., 1970.

[43] Ljung, L., *System Identification Toolbox: User's Guide*, Mathworks, 2023.

[44] Brabenec, R. L., *Resources for the Study of Real Analysis*, Mathematical Association of America, 2004.

[45] Verhaegen, M., and Verdult, V., *Filtering and System Identification: A Least Squares Approach*, Cambridge University Press, 2007.

[46] Gajic, Z., *Linear Dynamic Systems and Signals*, Prentice Hall, 2003.

[47] Vajta, M., *Some remarks on Padé-approximations*, 2000, pp. 53–58. 3rd TEMPUS-INTCOM Symposium on Intelligent Systems in Control and Measurements 2000 ; Conference date: 09-09-2000 Through 14-09-2000.

# Part II

# Preliminary Report

# Summary

The identification of time-varying, adaptive behaviour of a human operator (HO) in basic manual control tasks is undoubtedly under development since most methodologies only account for time-invariant systems. Previous authors have proved that estimation techniques based on Auto-Regressive-eXogeneous (ARX) structures can generally identify the HO model parameters. Nonetheless, ARX methods present several problems, such as the permanent bias in estimates that may increase depending on the HO model. Therefore, a novel identification technique based on Box-Jenkins (BJ) models is proposed to solve the issues found in ARX results, by achieving a more adequate match between the estimator structure and the HO model. BJ estimators not only offer the possibility of creating different poles for the linear and remnant components of the HO model, but also the adaptation to the remnant filter order found in the human. The identification process can be conducted offline by means of Prediction Error Method (PEM) algorithms, applicable for both ARX and BJ structures, or online, when the Recursive Least Squares (RLS) and the Recursive PEM (RPEM) are employed in ARX and BJ models, respectively. The BJ estimator has excellent potential as an identification tool, although non-linear optimization processes are involved, which increases the problem difficulty and the computational effort.

***Keywords:*** ARX; Box-Jenkins; Equalization; Gaussian White Noise; Human Operator; Prediction Error Method; Recursive Least Squares; Recursive Prediction Error Minimization; Remnant.

# 1 Introduction

Human manual-control behaviour naturally changes over time, in various contexts, and across operators. In order to explain the dynamic features of human operators (HOs) in skill-based manual control tasks, identification methods have been developed [22, 26, 25, 23, 24, 30, 34, 40]. However, they are often only applicable in situations when the control behaviour is thought to be time-invariant [30]. The availability of control-theoretic models that can capture both the adaptive and learning aspects of manual-control behaviour has long been a goal [45, 10, 50, 36, 11, 12]. Modern cybernetics is unable to fully explain how HOs modify their behaviour to deal with control-task changes. The continued development of time-varying identification techniques is necessary to make rapid progress in our knowledge of how people really interact with dynamic control systems [29].

The majority of research on identifying time-varying manual-control behaviour focuses on task variable changes that are particularly caused by variations in the dynamics of the controlled element (CE) [25, 45, 11, 50, 38, 10, 36]. Numerous research begin by examining single-axis compensatory control tasks and are based on the well-known crossover model [25], keeping in mind that the use of additional types of inputs to the HO or even extensions to multiple axes of control should not provide any significant challenges [11].

Thus, the purpose of this report is to propose a research plan, whose ultimate goal is to identify real-time (i.e. online) human control behaviour adaptation in a compensatory manual-control task. Real-time execution of these processes offers new possibilities in addition to advancing the earlier study objectives. For instance, pilot adaptation to anomalies in aircraft or controlled elements could be studied and modelled [50, 36, 11, 12]. On the other hand, reduced attention or distraction in real-world control activities could be identified by continuously monitoring the operator through an update of a human operator model [1]. Additionally, this might allow for adaptive haptic feedback that monitors the operator's present behaviour [32]. Online identification can assist in modifying experimental circumstances in real time to directly analyse adaptation behaviour or obtain desired haptic feedback characteristics in research with humans-in-the-loop.

Multiple authors have attempted to apply different techniques to achieve a successful online identification [32, 34, 48, 49, 6, 2, 10, 14], nevertheless, most of the methodologies used are not able to provide ideal results due to the high difficulty of the problem. For instance, strategies based on recursive estimation of Auto-Regressive-eXogeneous (ARX) model structures [30, 5, 10, 36, 17] may fail in reducing the relative bias [10] of the predicted model, in spite of their outstanding qualities in terms of low computational effort and straightforward estimation. Consequently, it is necessary to find an optimal identification method that is truly capable of assuming this task, so that a novel estimation technique based on Box-Jenkins (BJ) structures is proposed. Hence, this report lays the foundation for the development of a recursive BJ algorithm, which aims to solve the previous issues in the human behaviour identification process.

This preliminary report is structured as follows. A review of the literature on time-varying system identification techniques for manual control is included in Section 2. The principal research goal and sub-questions are then offered in Section 3. Afterwards, Section 4 discusses the required simulation conditions and their setup, while Section 5 presents an overview of the identification methods to be employed and their application. An extensive analysis of the results obtained is presented in Section 6. Finally, Section 7 provides conclusions and closing remarks regarding the applicability and impact of the research project, together with an analysis of future works to do in the BJ algorithm development.

# 2

# Literature survey

A literature review is done to acquire an overview of the current state of time-varying system identification in manual control. It addresses the various human operator models and related techniques for recognising them.

Pilot's decisions are extremely complex to model in most control tasks, however, the Crossover theory allows for the application of the quasi-linear HO model with acceptable accuracy. This model is mainly validated in compensatory manual-control tasks, in which the tracking error is only introduced as an input to the human operator.

In order to estimate the HO model parameters, multiple techniques have been employed. These can be divided into parametric and non-parametric methodologies. Additionally, estimators can be evaluated offline (batch-fitting) or recursively.

## 2.1. Compensatory manual-control task

In a control task, the human operator is typically a multichannel, adaptive, learning, non-linear controller [30, 34, 40, 49, 5, 37, 17]. In the loop, the pilot makes decisions based on feedback and feedforward paths, which provide information by means of visual and motion perception. The action generated is converted into a control-output through the neuromuscular system of the pilot, applying a force on the control device. Then, the final control-output is introduced into the controlled element dynamics. These mentioned paths can be adapted by the mental representation of a learned task, updating such knowledge based on new experiences. Figure 2.1 shows this pilot-vehicle loop [28]:

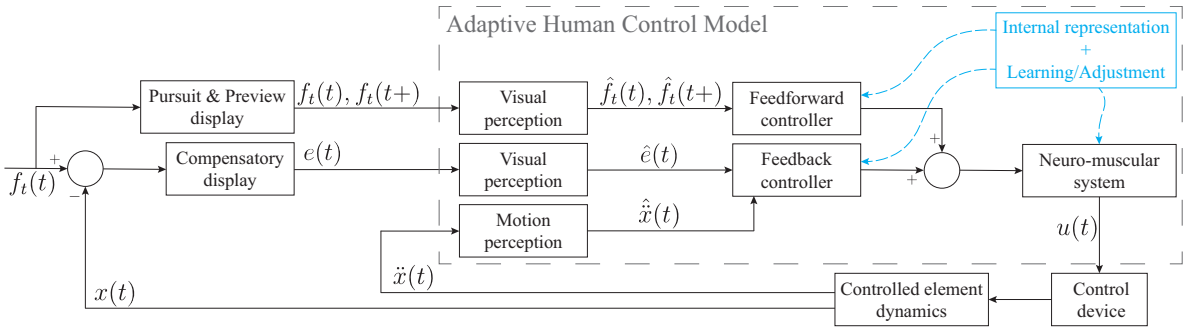


**Figure 2.1:** The multi-channel, learning, adaptive human controller. Adapted from Mulder et al. [28].

Although a general control-theoretic model of the human controller has not yet been discovered, validated models do exist for certain control tasks as shown in [20]. In particular, McRuer and Jex [25] proposed quasi-linear human operator models that present a satisfactory performance in HO modelling. These models are based on the Crossover theory, which is only applicable to simple cases, i.e., single-channel tracking tasks with only a feedback path from visual perception (pure compensatory display). In addition, no adaptation is supposed to be given since the learning process is finished. The Crossover model separates the additional unexplained behaviour by adding noise called 'remnant' $n$ and captures the linear behaviour of the human controller in a descriptive transfer function $H_P(j\omega)$.

Figure 2.2 depicts the quasi-linear operator model embedded in a compensating tracking task [50]. The HO, represented by the model $H_{HO}(s,t)$, monitors and responds to the error $e(t)$ between a goal $f_t(t)$ and the

output $x(t)$ of the CE dynamics $H_{CE}(s,t)$. The HO dynamics are composed of the remnant $n(t)$ and deterministic responses from $H_{HO_e}(s,t)$ (i.e. $H_P(s,t)$). Regarding the remnant $n(t)$, it is generated by feeding a signal $\varepsilon(t)$ with a determined statistical distribution through a remnant filter $H_n^m(s,t)$. Additionally, the HO must alter its control strategy as the CE dynamics change over time (CE adaptation) [45].

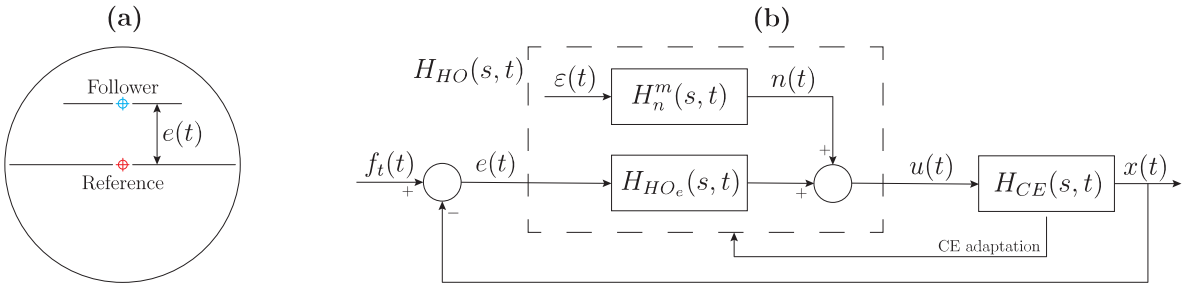


**Figure 2.2:** Single-axis compensatory manual-control task with time-varying dynamics: (a) Compensatory display, where $e(t)$ acts as stimulus, and (b) Block diagram.

## 2.2. Human operator models

In the Crossover model, McRuer and Jex [25] state that people modify their control behaviour to satisfy

$$H_{OL}(j\omega) = H_P(j\omega)H_{CE}(j\omega) = \frac{\omega_c}{j\omega}e^{-j\omega\tau_e}, \omega \approx \omega_c \tag{2.1}$$

in the crossover zone when transitory behaviour is eliminated. Then adjustment rules define how the describing function $H_{HO_e}(s,t)$ behaves in relation to the controlled element $H_{CE}(s,t)$ and what impact it has on crossover frequency $\omega_c$ and time delay $\tau_e$ in the frequency domain [25, 45]. By modelling the neuromuscular system (NMS) as a second-order transfer function $H_{nm}(j\omega)$ and the operator equalization as a gain and a lead(L)/lag(l) [25, 23, 24], while using the approach stated in [50, 10, 36], the general formulation of the describing function can be defined as follows:

$$H_P(j\omega) = K_p\frac{1+\tau_L j\omega}{1+\tau_l j\omega}e^{-j\omega\tau_e}H_{nm}(j\omega) \;\rightarrow\; H_{HO_e}(s,t) = \frac{K_e(t)\left[T_L(t)s+1\right]e^{-s\tau_e}\omega_{nm}^2}{s^2+2\zeta_{nm}\omega_{nm}s+\omega_{nm}^2}. \tag{2.2}$$

The equalization parameters, i.e. $K_e(t)$ and $T_L(t)$, determine the action of the feedback controller in the pilot-vehicle loop. On the other hand, $\omega_{nm}$ and $\zeta_{nm}$ model the neuromuscular dynamics.

Regarding possible remnant signal models, the theoretical background is limited and there is no consensus on how to model and take into account this remnant in Monte Carlo simulations [49, 50, 27, 16]. In most cases, a remnant signal is obtained by passing zero-mean Gaussian white noise (GWN) through a filter.

The literature contains a variety of filter options, however, the most accepted one is the $m$th-order remnant-filter proposed by Zaal [50] and later used in [10, 36]:

$$H_n^m(s,t) = \frac{K_n(t)}{(T_n s+1)^m}. \tag{2.3}$$

Additionally, the noise level $P_n$ has to be set during Monte Carlo simulations to give a certain value for the remnant gain. The definition provided by Van der El. [43], $P_n = \sigma_{u_n}^2/\sigma_u^2$, compares the variance of $u(t)$ due to the remnant to such variance $\sigma_u^2$. Other authors also use the definition $P_n = \sigma_n^2/\sigma_u^2$ [10, 36], directly measuring the ratio between the variances of remnant and control-output.

## 2.3. Controlled-element dynamics

The following second-order CE dynamics were taken into consideration by several authors [50, 10, 36], which serve as a general low-order approximation of typical vehicle dynamics [25]:

$$H_{CE}(s,t) = \frac{K_c(t)}{s(s+\omega_b(t))}. \tag{2.4}$$

The break frequency $\omega_b(t)$ and the control gain $K_c(t)$ can both change over time. The dynamics variation of the controlled element from single- to double-integrator dynamics (i.e., $1/s \leftrightarrow 1/s^2$) occurs at approximately $\omega_b(t)$. Furthermore, a sigmoid function is used in [50, 10, 36] to define the time variation of the operator equalization parameters in Equation 2.2 and CE coefficients in Equation 2.4:

$$p(t) = p_i + \frac{p_f - p_i}{1+e^{-G(t-t_M)}}, \tag{2.5}$$

where $p(t)$ is the time-varying parameter, $p_i$ and $p_f$ are the initial and final parameter values, $G$ is the transition rate, and $t_M$ is the time when states transition occurs.

## 2.4. Identification of time-varying operator behaviour

In system identification, there are two possible directions to take. The non-parametric method, only provides direct estimations of frequency response, and the parametric one, assumes a HO model structure and requires estimation of its parameters.

In relation to non-parametric methodologies, Olivari [32, 33] uses a Fourier Transform to convert a discrete Finite Impulse Response (FIR) into frequency response. Paassen [34] employs the method of Fourier coefficients, in which analytically generated transfer functions are only calculated at the forcing function frequency. In addition, wavelet-based analysis has been used in [48, 15], which reveals the frequency components of signals, but it also identifies where a certain frequency exists in the temporal or spatial domain.

Every other study taken into consideration employs a parametric strategy, which provides a more tangible understanding of the human controller than non-parametric strategies. Five research lines are found: batch fitting methods (maximum likelihood estimation, fitting Linear Parameter Variable (LPV) state space systems) and recursive fitting methods (Kalman filter estimation, fitting recursive ARX models, identification of Artificial Neural Networks (ANNs) structures).

In batch fitting strategies, the fitting is applied on the whole dataset at once, and typically, the operation limitation parameters (neuromuscular dynamics and operator time delay) are deemed constant while only the operator equalization parameters may vary. In [49, 50], a genetic maximum likelihood estimation is utilised as a time-domain technique to estimate the model parameters by assuming that time variation in the operator model has a sigmoid form that needs to be fitted. In [6], an LPV system is identified by applying the Predictor-Based Subspace Identification (PBSID) technique from [44]. In order to reflect the human operator adaptability for the LPV model, scheduling functions that have been derived analytically and experimentally are compared.

In recursive fitting methods, the operation equalization parameters can vary, but also, the NMS and HO time delay can be set constant or assumed time-varying depending on the type of estimator used. Regarding Kalman filters, a dual Extended Kalman Filter (EKF) [2, 38] and an Unscented Kalman Filter (UKF) [21] are used, in which a state-space form of the describing transfer function is used in the observation equations with an $n$th order Padé approximation for the time delay component. Also, settings are adjusted every time step based on forecasts and measurements. The ARX model structure [19] is employed in [10, 36, 17] to estimate time-varying HO behaviour, extending the work done in [30, 5]. ARX parameters are computed by the Recursive Least Squares (RLS) [13], which minimises a weighted linear least squares cost function relating to the input signals. Lastly, Jiao [14] uses a Single-Layer Linear Artificial Neural Network (SLLANN) model structure based on the discretized form of the pilot model in order to identify the equalization parameters.

# 3

# Research Objective and Questions

As shown in Section 2, several studies have attempted to apply recursive ARX model structures in manual-control cybernetics [30, 5, 10, 36, 17]. This method allows parameter estimation by direct linear regression, which makes the process more straightforward and reduces the computational effort, nevertheless, it leads to a relative bias [10] in the estimated model due to the fact that the ARX structure imposes the existence of unrealistic NMS poles in the remnant filter.

Therefore, different model structures should be addressed in order to achieve a more accurate parameter estimation. Franklin et al. [7] propose some non-linear model architectures, such as the Output-Error (OE), Box-Jenkins, or Auto-Regressive-Moving-Average-Exogenous (ARMAX). In particular, the BJ model perfectly matches the selected HO discrete-time transfer functions, hence, this could be the most adequate option to reduce the relative bias from recursive ARX methods and increase the robustness of the estimation process.

As a result, a new non-linear estimation process has to be defined accordingly for the time-varying human behaviour problem. The primary research question is:

**Is it possible to implement an online system identification method based on a recursive BJ algorithm for detecting and modelling time-varying manual control behaviour without explicitly presuming how human operator parameters would change over time?**

In order to answer the main research question, several sub-questions are required:

1. In relation to implementing the estimator in simulation:
   (a) How is the forcing function $f_t(t)$ defined?
   (b) What are the values used for the HO and CE parameters?
   (c) What ranges of remnant filter $m$th-order, time constant $T_n$ and noise level $P_n$ should be simulated?
   (d) What is the ideal remnant filter constant $K_n(t)$ for a required noise level $P_n$?
   (e) How can the BJ coefficients be converted into the HO parameters after fitting? What estimations of these coefficients might lead to problems during conversion?
2. In relation to testing the estimator in simulation:
   (a) What quality-of-fit metrics can be employed to determine how well a BJ structure fits the model?
   (b) What is the accuracy of the BJ estimation for each remnant filter?
   (c) What is the influence of variations in BJ model time-delay and remnant filter on each HO parameter estimation?
   (d) How is the performance of the BJ estimator with simulation data in comparison with ARX results?
3. In relation to evaluating the estimator experimentally:
   (a) What experiment will provide more information about the estimation performance and limitations of the BJ method?
   (b) Which experiment types require the human operator to modify the HO parameters significantly?
   (c) What steps are necessary to integrate the selected methodology into the research simulator framework?
   (d) How is the performance of the BJ estimator with experimental data in comparison with ARX results?

Since the development of an adequate, effective online identification algorithm based on BJ models can be quite complex, an extensive analysis of possible settings of the offline estimation method is made in Section 6. In this way, it is possible to know which is the best configuration for a BJ structure and how to properly tune the recursive algorithm to improve estimation.

# 4

# Human-controller simulation setup

In order to perform an adequate Monte Carlo simulation and evaluate the performance of new identification techniques, it is necessary to define a proper simulation framework to address all possible scenarios in an efficient and correct manner. Since Linear-Time Variant (LTV) models are used to define human behaviour, frequent tools such as Simulink are not valid, so that the iterative method to be used for simulation must be analysed more precisely, avoiding wrong results that may affect the veracity of the identification method's performance.

The forcing function and remnant noise characteristics are defined in this section, together with all simulation conditions to be employed in the Monte Carlo simulation, both for time-varying and constant HO model scenarios. Finally, an explanation of the connection between the simulation and parameter identification loops is included.

## 4.1. Simulation iterative approach

A simulation of a single-axis, single-channel compensatory tracking task [50] is performed (see Figure 2.2). The quasi-linear pilot behaviour is defined by the model shown in Equation 2.2, while the non-linear dynamics is explained by the $m$th-order remnant filter, given by Equation 2.3. The unfiltered remnant noise signal can be modelled as a GWN variable, so that zero mean and a unitary variance are chosen for simplicity, which can be modified by means of the remnant gain $K_n$. Hence, the final control-output from the pilot is computed as follows:

$$u(s) = \frac{K_e(t)\,[T_L(t)s+1]\,e^{-s\tau_e}\omega_{nm}^2}{s^2+2\zeta_{nm}\omega_{nm}s+\omega_{nm}^2}e(s) + \frac{K_n(t)}{(T_n s+1)^m}\varepsilon(s), \qquad \varepsilon(s)\sim N(0,\sigma_\varepsilon=1). \tag{4.1}$$

The controlled element dynamics, given by Equation 2.4, switch from a single integrator to a double integrator by means of the sigmoid function (see Equation 2.5) in order to compel time-varying human operator behaviour, giving as a result the corresponding change in human operator dynamics.

Discrete-time simulation is required owing to its time-varying nature. Thus, in order to simulate the control loop, all continuous-time transfer functions presented in Section 2 (i.e., human operator model, $H_{HO_e}(s,t)$, remnant filter, $H_n^m(s,t)$, and controlled element, $H_{CE}(s,t)$) must be converted into time-varying state-space systems [31],

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) + D(t)u(t) \end{cases}, \tag{4.2}$$

where $x(t)$, $u(t)$ and $y(t)$ are the state, input and output vectors, and $A(t)$, $B(t)$ and $C(t)$ are the state, input and output matrices, respectively. The direct transmission matrix $D(t)$ will be null in all cases. These matrices will be presented in controllable canonical form, as explained by Ogata [31]. Then, for each realisation, continuous-time matrices are converted into discrete-time ones. To achieve such a goal, the transformation procedure detailed by Gajic [8] is employed. This process is applied to the transfer functions of the human operator model, $H_{HO_e}(s,t)$, remnant filter, $H_n^m(s,t)$, and controlled element, $H_{CE}(s,t)$.

Once the discrete-time state-space systems are generated, the control-output signal $u(t)$ is delayed $\tau_e$ seconds before using it in the CE dynamics block. This means that the index in the discrete-time $u(t_k)$ vector is shifted $int(-\tau_e/T_s)$ positions.

If system dynamics are kept constant, the discrete-time data obtained from this process can be validated by a continuous-time simulation in Simulink, so that the relative error is low enough for the given time step.

## 4.2. Forcing function

To accomplish adequate simulations, the forcing function needs to be defined properly. As proposed by Zaal [50], a summation of $N_t$ sinusoids with different amplitudes $A_k$, frequencies $\omega_k$ and phases $\phi_k$,

$$f_t(t) = \sum_{k=1}^{N_t} A_k \cdot \sin\left(\omega_k \cdot (t - t_0) + \phi_k\right), \tag{4.3}$$

is an adequate option to excite the closed-loop system shown in Figure 2.2. To avoid the leakage phenomena in simulations, the frequencies $\omega_k$ are multiples of the base frequency $\omega_{k,0} = 2\pi/T_m$, where $T_m$ is the total simulation time. Thus, the integer $n_t$ is defined as $n_t = \omega_k/\omega_{k,0}$. In addition, a transient interval time is introduced in the simulation, $t_0$, in order that control loop variables are stabilised, consequently, the sinusoids are initialized at $t = t_0$. Table 4.1 shows the coefficients of each sinusoid employed in simulations.

| $k\,[-]$ | $n_t\,[-]$ | $\omega_k\,[\frac{rad}{s}]$ | $A_k\,[deg]$ | $\phi_k\,[rad]$ |
|---|---|---|---|---|
| 1 | 3 | 0.230 | 1.186 | -0.753 |
| 2 | 5 | 0.384 | 1.121 | 1.564 |
| 3 | 8 | 0.614 | 0.991 | 0.588 |
| 4 | 13 | 0.997 | 0.756 | -0.546 |
| 5 | 22 | 1.687 | 0.447 | 0.674 |
| 6 | 34 | 2.608 | 0.245 | -1.724 |
| 7 | 53 | 4.065 | 0.123 | -1.963 |
| 8 | 86 | 6.596 | 0.061 | -2.189 |
| 9 | 139 | 10.661 | 0.036 | 0.875 |
| 10 | 229 | 17.564 | 0.025 | 0.604 |

**Table 4.1:** Target function parameters for each component.

## 4.3. Remnant noise

Then, a remnant realisation is generated for every single simulation run by feeding the zero-mean GWN with unit variance through a filter. Different $m$th-order filters, $m \in [1, 4]$ (see [10]), and time remnant constants, $T_n \in \{0.06, 0.2\}$ (see [50, 10]), can be simulated to address their influence on results. Additionally, it is possible to introduce multiple noise levels $P_n \in [0.0, 0.99]$ in the system by modifying the remnant gain $K_n$. Consequently, a certain value for the $K_n$ parameter has to be selected to obtain a requested noise level at each simulation, for which the stochastics theory presented by Ljung [19] can be employed, so that a suitable formula is developed.

Appendix A shows the derivation process to obtain such remnant gain definition and the theoretical background used for this purpose. The $K_n$ formula found after the analysis depends on the forcing function $f_t$, filtered GWN characteristics, sampling frequency, transfer functions' coefficients, and how the noise level is defined. Based on previous works [43, 9], two possible definitions of $P_n$ are found, i.e., $P_n = \sigma^2_{\bar{u}_n\bar{u}_n}/\sigma^2_{\bar{u}\bar{u}}$ and $P_n = \sigma^2_{\bar{n}\bar{n}}/\sigma^2_{\bar{u}\bar{u}}$, therefore, the calculation of the remnant gain can follow two different formulas shown in Equations A.14 and A.17, respectively. In this case, the first option is preferred due to its advantages, such as the higher clearness in the $P_n$ domain (is always $[0, 1)$). Thus, Equation A.14 with unitary $\varepsilon(t)$ variance,

$$K_n = \sqrt{\frac{P_n}{(1 - P_n) \cdot T_s} \frac{\frac{\pi}{2} \sum_{k=1}^{N_t} A_k^2 \left| \frac{H_{HO}(j\omega_k)}{1 + H_{HO}(j\omega_k) H_{CE}(j\omega_k)} \right|^2}{\int_0^{\pi/T_s} \frac{1}{\left|(T_n(j\omega)+1)^m (1 + H_{HO}(j\omega) H_{CE}(j\omega))\right|^2} \mathrm{d}\omega}}, \tag{4.4}$$

is utilised to compute the remnant gain of each set of simulations (generation of new remnants do not require a new $K_n$, unless some simulation parameter is modified). Additionally, this formula is validated in Section A.4.

## 4.4. Simulation conditions

To create a time-varying simulation framework, two different states, $s_1$ and $s_2$, are defined on the basis of the CE dynamics variation and its effect on the human operator. Zaal [50] and van Grootheest [10] employ two sets of parameters for the CE dynamics and the assumed HO dynamics. Furthermore, the corresponding remnant gain $K_n$ is defined for each state based on Equation A.14. These parameters are recorded in Table 4.2.

| State | CE | | HO | | | | | | HO× CE | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $K_c[-]$ | $\omega_b[rad/s]$ | $K_e[-]$ | $T_L[s]$ | $K_{\dot{e}}[-]$ | $\tau_e[-]$ | $\omega_{nm}[rad/s]$ | $\zeta_{nm}[-]$ | $\omega_c[rad/s]$ | $\phi_m[deg]$ |
| $s_1$ | 90 | 6.0 | 0.09 | 0.40 | 0.036 | 0.28 | 11.25 | 0.35 | 1.5 | 77.0 |
| $s_2$ | 30 | 0.2 | 0.07 | 1.20 | 0.084 | 0.28 | 11.25 | 0.35 | 2.8 | 22.7 |

**Table 4.2:** CE, HO and HO×$CE$ parameters for states $s_1$ and $s_2$.

In addition, each state has its associated discrete-time parameters, which are obtained from the discrete-to-continuous conversion of the HO transfer function (see Section 5). These values are presented in Table 4.3.

| States | $a_1^{d,0}[-]$ | $a_2^{d,0}[-]$ | $b_0^{d,0}[-]$ | $b_1^{d,0}[-]$ | $n_k^0[-]$ |
|---|---|---|---|---|---|
| $s_1$ | -1.9121 | 0.9243 | 0.0443 | -0.0432 | 29 |
| $s_2$ | -1.9121 | 0.9243 | 0.1024 | -0.1016 | 29 |

**Table 4.3:** Discrete-time parameters values for ZOH discretization: states $s_1$ and $s_2$.

In the change between states, a transition (performed by a sigmoid function in Equation 2.5) from single- to double-integrator dynamics, or vice-versa, is evaluated. A total of six simulation conditions are proposed, where the first two of them consider constant sets of parameters, and the remaining four propose a time-varying scenario with a soft or aggressive state transition (determined by the parameter $G$). The HO time-delay and NMS parameters are assumed to remain constant during the simulation [50, 9]. The conditions C1-C6 are recorded in Table 4.4.

| Condition | $H_{CE}(s)$ | $G\left[s^{-1}\right]$ |
|---|---|---|
| C1 | $s_1$ | - |
| C2 | $s_2$ | - |
| C3 | $s_1 \rightarrow s_2$ | 0.5 |
| C4 | $s_1 \rightarrow s_2$ | 100 |
| C5 | $s_2 \rightarrow s_1$ | 0.5 |
| C6 | $s_2 \rightarrow s_1$ | 100 |

**Table 4.4:** Simulation conditions.

## 4.5. Identification process

As explained in Section 5, offline and online identification approaches, based on ARX and BJ structures, are proposed for the HO parameters estimation problem. In batch-fitting identification methods, the estimation is performed once all simulation data is obtained, while new parameter predictions are conducted at each iteration in the case of recursive methods. In addition, the run-in time data, which is created in the region $t \in [0, t_0]$, is discarded during estimation to remove the transients from the measured signals.

Since the goal of this project is to develop a possible identification method based on BJ models and compare its performance in general terms with ARX results from previous works, no special datasets (or part of training data) are used for validation or verification, so that the entire simulation data is only used for estimation purposes. Therefore, the online estimation algorithms are applied directly to the simulation data and no revision of the identification procedure (such as detection of overfitting) is possible, which offers a clearer comparison of the real performance of techniques based on ARX or BJ.

The ARX and BJ models are set up by defining the order of its polynomials and the integer values for the estimated HO time delay, $n_k^* \in \left[n_{k,min}^*, n_{k,max}^*\right]$, and the $m$th-order of the remnant filter, $m^* \in [m_{min}, m_{max}]$, in the BJ scenario. Additionally, a time-invariant time-delay is assumed in ARX or BJ structures to be modelled, so that the value of $n_k^*$ is varied in Section 6 with the only purpose of analysing estimators' performance for different model time-delays.

Once the estimation process is finished, the discrete-time parameters of the estimated ARX and BJ models are converted into continuous-time ones to obtain the identified human operator parameters.

ARX estimators can be easily implemented, however, identification procedures based on BJ structures involve a more difficult optimization process due to its non-linear characteristics. Hence, the system identification library in Matlab [18] can be used for the estimation and evaluation of batch and recursive BJ models. In particular, the functions *bj.m* and *recursiveBJ.m* are utilised to achieve this goal.

# 5

# Human-controller identification setup

In this section, two methods are proposed for the human-controller identification problem. The first one is based on an Auto-Regressive-eXogeneous (ARX) structure and has been used in previous works [10, 36], while the second one represents a novel estimation technique that consists of Box-Jenkins (BJ) models. The purpose of the latter will be to solve the issues present in the ARX method, in which the parameter estimation is permanently biased due to a mismatch in the remnant filter poles between the real model and the ARX structure. Appendix B presents the derivation of the process to set up these two identification methods and the constraints to be fulfilled to achieve an adequate HO parameter retrieval.

HO parameters can be estimated by means of batch-fitting and online identification. For the first scenario, algorithms based on the Prediction Error Method (PEM) are applied for ARX and BJ structures. In the second case, Recursive Least-Squares (RLS) or Recursive Prediction Error Minimization (RPEM) algorithms are employed. Regarding the online ARX identification, the corresponding algorithms are tuned based on the solutions found by [10].

Finally, a set of quality-of-fit metrics is proposed to evaluate the performance of the ARX and BJ estimators.

## 5.1. Transfer-function models

Ljung [19] presents a series of transfer-function models from a general family of model structures, which can be employed to identify the discrete-time TF corresponding to the HO model. This family can be represented by the expression adapted to the case study:

$$u(t_k) = \frac{B(z^{-1})}{A(z^{-1})F(z^{-1})} z^{-n_k} e(t_k) + \frac{C(z^{-1})}{A(z^{-1})D(z^{-1})} \varepsilon(t_k), \quad \varepsilon'(t_k) \sim N(0, \sigma_\varepsilon), \tag{5.1}$$

where $u(t)$ is the output signal associated with the control-output, $e(t)$ is the input signal corresponding to the tracking error, and $\varepsilon(t)$ is the Gaussian white noise with a standard deviation $\sigma_\varepsilon$. The general model structure depends on a total of 5 polynomials $\{A, B, C, D, F\}$:

$$A(z^{-1}) = 1 + a_1^d z^{-1} + \ldots + a_{n_f}^d z^{-n_a} \tag{5.2a}$$

$$B(z^{-1}) = b_0^d + b_1^d z^{-1} + \ldots + b_{n_b}^d z^{-n_b}, \qquad C(z^{-1}) = 1 + c_1^d z^{-1} + \ldots + c_{n_c}^d z^{-n_c} \tag{5.2b}$$

$$D(z^{-1}) = 1 + d_1^d z^{-1} + \ldots + d_{n_d}^d z^{-n_d}, \qquad F(z^{-1}) = 1 + f_1^d z^{-1} + \ldots + f_{n_f}^d z^{-n_f}. \tag{5.2c}$$

In addition, the predictor of the control-output signal, $\hat{u}(t)$, would present the following general expression [19]:

$$\hat{u}(t_k \mid \theta) = \frac{B(z^{-1})D(z^{-1})}{C(z^{-1})F(z^{-1})} z^{-n_k} e(t_k) + \left[1 - \frac{D(z^{-1})A(z^{-1})}{C(z^{-1})}\right] u(t_k), \tag{5.3}$$

where $\theta$ is the adjustable parameters vector. By adjusting this previous expression, the prediction error is found:

$$\epsilon(t_k, \theta) = u(t_k) - \hat{u}(t_k \mid \theta) = \frac{D(z^{-1})}{C(z^{-1})} \left[A(z^{-1})u(t_k) - \frac{B(z^{-1})}{F(z^{-1})} z^{-n_k} e(t_k)\right]. \tag{5.4}$$

From the general expression in Equation 5.1, a total of 32 models can be found based on [19], however, only 10 are the most commonly used in discrete-time estimation problems. These structures are shown in Table 5.1.

In this project, the ARX model is selected together with an additional structure able to solve the issues shown in [10, 36]. Although ARX structures represent an adequate tool for HO identification, they include a permanent bias in the control loop that cannot be removed due to the poles mismatch in the remnant filter TF. Therefore, a Box-Jenkins model is the only alternative by which completely different poles can be introduced in the structure.

The ARX structure allows the application of linear regression, hence, the parameters vector $\theta$ can be estimated by Ordinary Least-Squares (OLS) or Recursive Least-Squares (RLS) methods. On the other hand, the parameter estimation of each BJ discrete-time polynomial requires the optimization of a non-linear problem, so that a Prediction Error Method (PEM) algorithm has to be evaluated as explained in [41, 19, 46, 47]. When an online BJ estimation is desired, a recursive PEM algorithm is implemented based on Young [46]. In an identification procedure, the control-output and tracking error are known at each time step, however, the noise signal is unknown, so the density function of a normal distribution is always assumed in the estimation process.

| Name of model structure | Polynomials used in Eq. 5.1 | Model formula |
|---|---|---|
| Finite-Impulse-Response (FIR) | B | $u(t_k) = B(z^{-1})z^{-n_k}e(t_k) + \varepsilon(t_k)$ |
| Auto-Regressive (AR) | A | $u(t_k) = \frac{1}{A(z^{-1})}\varepsilon(t_k)$ |
| Moving-Average (MA) | C | $u(t_k) = C(z^{-1})\varepsilon(t_k)$ |
| Auto-Regressive-eXogeneous (ARX) | A, B | $u(t_k) = \frac{B(z^{-1})}{A(z^{-1})}z^{-n_k}e(t_k) + \frac{1}{A(z^{-1})}\varepsilon(t_k)$ |
| Auto-Regressive-Moving-Average (ARMA) | A, C | $u(t_k) = \frac{C(z^{-1})}{A(z^{-1})}\varepsilon(t_k)$ |
| Auto-Regressive-Moving-Average-eXogeneous (ARMAX) | A, B, C | $u(t_k) = \frac{B(z^{-1})}{A(z^{-1})}z^{-n_k}e(t_k) + \frac{C(z^{-1})}{A(z^{-1})}\varepsilon(t_k)$ |
| ARARX | A, B, D | $u(t_k) = \frac{B(z^{-1})}{A(z^{-1})}z^{-n_k}e(t_k) + \frac{1}{A(z^{-1})D(z^{-1})}\varepsilon(t_k)$ |
| ARARMAX | A, B, C, D | $u(t_k) = \frac{B(z^{-1})}{A(z^{-1})}z^{-n_k}e(t_k) + \frac{C(z^{-1})}{A(z^{-1})D(z^{-1})}\varepsilon(t_k)$ |
| Output-Error (OE) | B, F | $u(t_k) = \frac{B(z^{-1})}{F(z^{-1})}z^{-n_k}e(t_k) + \varepsilon(t_k)$ |
| Box-Jenkins (BJ) | B, C, D, F | $u(t_k) = \frac{B(z^{-1})}{F(z^{-1})}z^{-n_k}e(t_k) + \frac{C(z^{-1})}{D(z^{-1})}\varepsilon(t_k)$ |

**Table 5.1:** Principal model structures based on general family of discrete-time transfer functions.

Before estimating the ARX or BJ models online or offline, it is necessary to make sure those structures match the discretized HO and remnant filter models. Franklin et al. [7] present a variety of discretization techniques that can be applied to the continuous-time transfer function, and can follow a numerical integration, a Z-transform mapping strategy or try to model the sampled system. The discrete equivalents via numerical integration are based on the numerical method [4] used to find a solution of the differential equation associated to the continuous-time TF to be discretized. The principal methods are: Forward-Euler, Backward-Euler, Tustin (with/without pre-warp). Then, the zero-pole matching equivalents are obtained by mapping the continuous-time TF's poles and zeros from the relationship between s- and z-planes. On the other hand, Tangirala [41] proposes model-sampling techniques, based on the Zero-order hold (ZOH) or First-order hold (FOH), that reconstruct the measured signal and enable a two-step continuous-to-discrete time TF conversion. Table 5.2 records the mentioned techniques.

Nevertheless, not all discretization methods are applicable to this case, since the order of each polynomial is constrained by the number of continuous-time parameters to be estimated. Regarding the HO model, $H_{HO_e}(s,t)$, Tangirala's ZOH method is the best option for both ARX and BJ scenarios. Attending to the remnant filter, $H_n^m(s,t)$, which is only addressed in a BJ structure, the backward-Euler strategy is needed. Thus, the obtained ARX model structure is defined by $\{n_a = 2, n_b = 1, n_k = int(\tau_e/T_s) + 1\}$, while the BJ one is represented by $\left\{n_b = 1, n_c = 0, n_d = m, n_f = 2, n_k = int(\tau_e/T_s) + 1\right\}$, where $m$ is the remnant filter order, $\tau_e$ is the HO time delay, and $T_s$ is the time step. The discrete-time transfer functions of the HO model and remnant filter are shown in Equations B.5 and B.11, respectively. Consequently, the ARX and BJ models will be defined by the following structures:

$$ARX : u(t_k) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}} z^{-n_k} e(t_k) + \frac{1}{1 + a_1^d z^{-1} + a_2^d z^{-2}} \varepsilon'(t_k), \quad \varepsilon'(t_k) \sim N\left(0, K_n \cdot \sigma_\varepsilon\right) \tag{5.5a}$$

$$BJ(m) : u(t_k) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}} z^{-n_k} e(t_k) + \frac{1}{1 + d_1^d z^{-1} + \ldots + d_m^d z^{-m}} \varepsilon'(t_k), \quad \varepsilon'(t_k) \sim N\left(0, K_n \left(\frac{T_s}{T_n + T_s}\right)^m \cdot \sigma_\varepsilon\right), \tag{5.5b}$$

where $\varepsilon'(t)$ is equivalent to a standard GWN signal $\varepsilon(t)$ scaled with $K_n$ and parameters from the discretization process. For the sake of simplicity, coefficients of the polynomial $F(z^{-1})$ are evaluated as if they were from $A(z^{-1})$, hence, terms $f_i^d$ will be named as $a_i^d$ from now on in BJ structures.

| Method | Description |
|---|---|
| Forward-Euler | Based on an integration rule in which the amplitude of the integrated area (for a time-domain $[kT_s - T_s, kT_s]$) is equal to the integrand value at $t = kT_s - T_s$. The relationship obtained is: $s = \frac{z-1}{T_s}$. |
| Backward-Euler | Based on an integration rule in which the amplitude of the integrated area (for a time-domain $[kT_s - T_s, kT_s]$) is equal to the integrand value at $t = kT_s$. The relationship obtained is: $s = \frac{1-z^{-1}}{T_s}$. |
| Trapezoid, bilinear, or Tustin | Based on an integration rule in which the amplitude of the integrated area (for a time-domain $[kT_s - T_s, kT_s]$) is equal to the average of integrand values at $t = kT_s$ and $t = kT_s - T_s$. The relationship obtained is: $s = \frac{2(z-1)}{T_s(z+1)}$. |
| Bilinear transformation with prewarping | Trapezoidal discretization that keeps relevant dynamics information at the prewarp frequency, $\omega_p$, obtaining a more accurate match. The relationship obtained is: $s = \frac{\omega_p}{\tan \omega_p T_s/2} \frac{(z-1)}{(z+1)}$. |
| Zero-Order Hold (ZOH) | The control-output is defined as piecewise constant over the sample interval $T_s$. From a continuous-time TF, $H_c(s)$, its discrete-time TF, $H_d(z)$, is obtained by: $H_d(z) = (1 - z^{-1})\mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{H_c(s)}{s}\right\}\Big\|_{t=kT_s}\right\}$. |
| First-Order Hold (FOH) | Based on a triangle approximation, the control-output is defined as piecewise linear over the sample interval $T_s$. From a continuous-time TF, $H_c(s)$, its discrete-time TF, $H_d(z)$, is obtained by: $H_d(z) = \frac{(z-1)^2}{T_s z}\mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{H_c(s)}{s^2}\right\}\Big\|_{t=kT_s}\right\}$. |
| Zero-Pole matching | Poles and zeros are found by extrapolation between the s- and z-planes according to the relationship $z = e^{sT_s}$. This method consists of a set of heuristical rules (see [7]) for setting the gain of the z-transform and locating the poles and zeros that will define an equivalent discrete-time TF, $H_d(z)$, of the continuous-time TF, $H_c(s)$. |

**Table 5.2:** Discretization methods of continuous-time transfer functions.

Once the discrete-time coefficients are obtained, the discrete-time state-space system (in controllable canonical [31]) form is calculated for the transfer function $B(z^{-1})/A(z^{-1})$, in ARX, or $B(z^{-1})/F(z^{-1})$, in BJ. Then, Gajic's procedure [8] is used reversely to get the continuous-time state-space system by means of the logarithm of its extended matrix. To obtain a successful conversion of the state-space system, its associated extended matrix must be invertible and with no negative real eigenvalues [8] (see Equations B.17, B.18 and B.19). After, the HO model can be computed from the matrices realisation as $\hat{H}_{HO_e}(s, t_k) = C(t_k)(sI - A(t_k))^{-1}B(t_k)$. In relation to the transfer function $C(z^{-1})/D(z^{-1})$ in BJ models, a direct discrete-to-continuous time conversion is possible (see Equations B.14 and B.15). In Appendix B, Sections B.1 and B.2 show the mentioned procedures to obtain the continuous-time HO parameters and identified coefficients in the remnant filter, while Section B.3 presents the derivation process to define the discrete-time parameters constraints (i.e., $\left\{a_1^d < 0, a_2^d > 0\right\}$) that ensure a proper discrete-to-continuous time conversion.

Thus, from the estimated continuous-time transfer function of the HO model, $\hat{H}_{HO_e}(s, t_k)$, whose parameters are defined in Equations B.9 and B.10, the identified human operator coefficients are:

$$\hat{H}_{HO_e}(s, t_k) = \frac{b_0^c s + b_1^c}{s^2 + a_1^c s + a_2^c} \Rightarrow \left\{\hat{K}_e = \frac{b_1^c}{a_2^c},\ \hat{T}_L = \frac{b_0^c}{b_1^c},\ \hat{\omega}_{nm} = \sqrt{a_2^c},\ \hat{\zeta}_{nm} = \frac{a_1^c}{2\sqrt{a_2^c}}\right\}. \tag{5.6}$$

Based on previous experience [10, 36, 50], these discretization methods and posterior discrete-to-continuous time conversion should not lose any system identification information, providing acceptable estimations.

## 5.2. Prediction Error Method

In order to estimate the set of model parameters offline, PEM techniques represent an adequate option for all structures, both linear and non-linear. The PEM procedure consists of initial states estimation, definition of the optimization model, and variance estimation.

Other methods can also be applied to the discrete-time transfer function estimation problem, such as the Maximum Likelihood (ML) or the Refined Instrumental Variable (RIV) [19, 46, 47]. The ML technique finds the full optimization model from a log-likelihood function, which is based on a Gaussian distribution of the noise $\varepsilon$. The RIV method is a pseudo-linear regression approach to ML estimation. Both methods could be in ARX and BJ structures, but they are only applicable to systems with Gaussian noise, hence, the PEM algorithm presents a more generalist alternative. The ML model is only used to find an expression to estimate the noise variance.

### 5.2.1. The PEM algorithm

Prediction error methods are based on the idea of minimising a cost function, $V_N$, that measures the level of prediction error, $\epsilon(t_k, \theta)$, to find a solution for $\hat{\theta}$ [19, 46]. Thus, from a batch of data $Z^N$,

$$Z^N = [u(t_1), e(t_1), u(t_2), e(t_2), \dots, u(t_N), e(t_N)], \tag{5.7}$$

and the prediction error formula (see Equation 5.4), the cost function can be defined as follows:

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{i=1}^{i=N} l(\epsilon(t_i, \theta)). \tag{5.8}$$

In $V_N(\theta, Z^N)$, $l(\cdot)$ is a scalar-valued (typically positive) function. The quadratic norm is the most common in optimization problems:

$$l(\epsilon) = \frac{1}{2}\epsilon^2. \tag{5.9}$$

In order to reduce equation sizes, the HO time delay can be integrated into the $B(z^{-1})$ polynomial, such that

$$\overline{B}(z^{-1}) = B(z^{-1})z^{-n_k}. \tag{5.10}$$

Therefore, the goal of a PEM algorithm is to find the vector of parameters $\hat{\theta}$ that minimizes the cost function:

$$\hat{\theta} = \arg\min_{\theta} V_N(\theta, Z^N) = \arg\min_{\theta} \left\{ \frac{1}{2N} \sum_{i=1}^{i=N} \left[ \frac{D(z^{-1})}{C(z^{-1})} A(z^{-1}) u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})} e(t_i) \right]^2 \right\}. \tag{5.11}$$

In order to optimize such cost function, the partial differentiation of $V_N(\theta, Z^N)$ with respect to all the parameters is made, i.e.:

$$\nabla_\theta \left[ V_N(\theta, Z^N) \right] = \frac{1}{N} \sum_{i=1}^{i=N} \epsilon(t_i, \theta) \nabla_\theta \left[ \epsilon(t_i, \theta) \right]. \tag{5.12}$$

Hence, the derivative of the prediction error for each model parameter needs to be computed before defining the complete optimization problem:

$$\frac{\partial \epsilon(t_k, \theta)}{\partial a_j^d} = \frac{D(z^{-1})}{C(z^{-1})} z^{-j} u(t_k), \qquad j = 1, 2, \dots, n_a, \tag{5.13a}$$

$$\frac{\partial \epsilon(t_k, \theta)}{\partial b_j^d} = -\frac{D(z^{-1})}{F(z^{-1})C(z^{-1})} z^{-j-n_k} e(t_k), \qquad j = 0, 1, \dots, n_b, \tag{5.13b}$$

$$\frac{\partial \epsilon(t_k, \theta)}{\partial c_j^d} = -\frac{D(z^{-1})}{C^2(z^{-1})} A(z^{-1}) z^{-j} u(t_k) + \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C^2(z^{-1})} z^{-j} e(t_k), \qquad j = 1, 2, \dots, n_c, \tag{5.13c}$$

$$\frac{\partial \epsilon(t_k, \theta)}{\partial d_j^d} = \frac{A(z^{-1})}{C(z^{-1})} z^{-j} u(t_k) - \frac{\overline{B}(z^{-1})}{F(z^{-1})C(z^{-1})} z^{-j} e(t_k), \qquad j = 1, 2, \dots, n_d, \tag{5.13d}$$

$$\frac{\partial \epsilon(t_k, \theta)}{\partial f_j^d} = \frac{\overline{B}(z^{-1})D(z^{-1})}{F^2(z^{-1})C(z^{-1})} z^{-j} e(t_k), \qquad j = 1, 2, \dots, n_f. \tag{5.13e}$$

Consequently, the optimization model is defined by the following set of equations when equalled to zero:

$$\frac{\partial V_N}{\partial a^d_{j=1,2,\dots,n_a}} = \frac{1}{N} \sum_{i=1}^{i=N} \left[ \frac{D(z^{-1})}{C(z^{-1})} A(z^{-1}) u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})} e(t_i) \right] \times \frac{D(z^{-1})}{C(z^{-1})} z^{-j} u(t_i), \tag{5.14a}$$

$$\frac{\partial V_N}{\partial b^d_{j=0,1,\dots,n_b}} = \frac{1}{N} \sum_{i=1}^{i=N} \left[ \frac{D(z^{-1})}{C(z^{-1})} A(z^{-1}) u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})} e(t_i) \right] \times \frac{-D(z^{-1})}{F(z^{-1})C(z^{-1})} z^{-j-n_k} e(t_i), \tag{5.14b}$$

$$\frac{\partial V_N}{\partial c^d_{j=1,2,\dots,n_c}} = \frac{1}{N} \sum_{i=1}^{i=N} \left[ \frac{D(z^{-1})}{C(z^{-1})} A(z^{-1}) u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})} e(t_i) \right] \times z^{-j} \left[ -\frac{D(z^{-1})}{C^2(z^{-1})} A(z^{-1}) u(t_i) + \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C^2(z^{-1})} e(t_i) \right], \tag{5.14c}$$

$$\frac{\partial V_N}{\partial d^d_{j=1,2,\ldots,n_d}} = \frac{1}{N}\sum_{i=1}^{i=N}\left[\frac{D(z^{-1})}{C(z^{-1})}A(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})}e(t_i)\right]\times z^{-j}\left[\frac{A(z^{-1})}{C(z^{-1})}u(t_i) - \frac{\overline{B}(z^{-1})}{F(z^{-1})C(z^{-1})}e(t_i)\right], \tag{5.14d}$$

$$\frac{\partial V_N}{\partial f^d_{j=1,2,\ldots,n_f}} = \frac{1}{N}\sum_{i=1}^{i=N}\left[\frac{D(z^{-1})}{C(z^{-1})}A(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})}e(t_i)\right]\times \frac{\overline{B}(z^{-1})D(z^{-1})}{F^2(z^{-1})C(z^{-1})}z^{-j}e(t_i). \tag{5.14e}$$

At this point, the PEM and ML optimization problems are identical [47], since Equations 5.14 match the ones obtained in an ML scenario. In order to find the optimal solution to this minimization problem, a linear or non-linear procedure has to be used in the function of the type of model to be estimated. In the case of ARX structures, the estimation process is quite straightforward since linear regression can be applied [19], while BJ models require non-linear optimization methods based on gradient-based schemes [3] or more innovative techniques [18].

Other options for cost function can be employed in the PEM algorithm, such as the regularised $V_N(\theta, Z^N)$ [19], which also weighs the parameter vector values or its deviations with respect to a pre-estimated $\hat{\theta}^0$. However, only a standard cost function is considered in this case. Additionally, other definitions of $\epsilon(t_k, \theta)$ can be used, since it can represent the one-step-ahead prediction error (Equation 5.4), or the simulation error. The type of error to be employed in the cost function can be selected in the Matlab function *pem.m* by means of the option *'Focus'*: *'prediction'* (prediction error) and *'simulation'* (simulation error) [18]. The first option is preferred.

When calculating each component of Equations 5.14, there might be some difficulties in evaluating fractions of polynomials, such as $D(z^{-1})/C(z^{-1})$. To solve this issue, Maclaurin series can be used to convert the fraction into a pure polynomial:

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 + \ldots \tag{5.15}$$

### 5.2.2. Initial states estimation

Matlab functions based on *pem.m*, such as *arx.m* or *bj.m*, use the whole data set $Z^N$ to optimize the cost function. Therefore, it is necessary to determine the values of pairs $\{u(t_k), e(t_k)\}$ when $k < 1$. Two possible scenarios are given: setting all $\{u(t_{k<1}), e(t_{k<1})\}$ to zero, or estimating all those pairs required in the $V_N$ minimization on the basis of prediction error reduction. The first case can lead to inaccuracies in the estimation (especially, in remnant-free cases) seeing that the actual signal may not be null, hence, a prediction of $k < 1$ conditions is performed (Matlab option *'InitialCondition'* is set to *'estimate'*), analysing the initial states as independent estimation variables [18].

Since the prediction error does not depend on the Gaussian noise, this variable is not taken into account during the initial state estimation. Only the control-output and tracking error are employed in the cost function minimization to obtain the initial states. Initial values for the discrete-time parameters, $\theta^0$ (i.e., $A_0(z^{-1})$, $B_0(z^{-1})$, $C_0(z^{-1})$, $D_0(z^{-1})$ and $F_0(z^{-1})$), are utilised in the optimization process. Thus, the initial conditions,

$$Z_0^N = [\ldots, u(t_{-1}), e(t_{-1}), u(t_0), e(t_0)], \tag{5.16}$$

can be predicted from the non-linear optimization of the function $V_N^0(\theta^0, Z_0^N)$:

$$\hat{Z}_0^N = \arg\min_{Z_0^N} V_N^0(\theta^0, Z_0^N) = \arg\min_{Z_0^N}\left\{\frac{1}{2N}\sum_{i<1}\left[\frac{D_0(z^{-1})}{C_0(z^{-1})}A_0(z^{-1})u(t_i) - \frac{B_0(z^{-1})D_0(z^{-1})}{F_0(z^{-1})C_0(z^{-1})}z^{-n_k}e(t_i)\right]^2\right\} \tag{5.17}$$

### 5.2.3. Variance estimation

Once, the optimization process is converged and a parameter vector solution is found, a partial differentiation of the log-likelihood function for $N$ observations shown by Young [47],

$$\mathscr{L}(\theta, \sigma_\varepsilon^2, u(t), e(t)) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\left(\sigma_\varepsilon^2\right) - \frac{1}{2\sigma_\varepsilon^2}\sum_{i=1}^{i=N}\left[\frac{D(z^{-1})}{C(z^{-1})}A(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})}e(t_i)\right]^2 \tag{5.18}$$

is computed to find an estimation of the noise variance $\hat{\sigma}_\varepsilon^2$ (provided that the noise signal presents a Gaussian distribution) based on the predicted parameter vector:

$$\frac{\partial \mathscr{L}}{\partial \sigma_\varepsilon^2} = -\frac{N}{\sigma_\varepsilon^2} + \frac{1}{\sigma_\varepsilon^4}\sum_{i=1}^{i=N}\left[\frac{D(z^{-1})}{C(z^{-1})}A(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})C(z^{-1})}e(t_i)\right]^2 = 0. \tag{5.19}$$

This partial derivative represents the difference between the PEM and ML algorithms, since the first one is only focused on the polynomial coefficients. Therefore, the estimation of the noise variance is given by the following equation:

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{N}\sum_{i=1}^{i=N}\left[\frac{\hat{D}(z^{-1})}{\hat{C}(z^{-1})}\hat{A}(z^{-1})u(t_i) - \frac{\hat{B}(z^{-1})\hat{D}(z^{-1})}{\hat{F}(z^{-1})\hat{C}(z^{-1})}z^{-n_k}e(t_i)\right]^2. \tag{5.20}$$

### 5.2.4. Summary

Hence, the PEM algorithm consists of the following steps:

1. Define the initial batch of data $Z_0^N$ to be estimated based on the pairs $\{u(t_{k<1}), e(t_{k<1})\}$ needed in the partial derivatives.
2. Minimize the initial cost function $V_N^0(\theta^0, Z_0^N)$ to find an estimation of the initial pairs based on the starting parameter vector values.
3. Define the partial derivatives of the cost function $V_N(\theta, \hat{Z}_0^N \cup Z^N)$ with respect to the polynomial coefficients.
4. Equal such derivatives to zero to set the optimization model equations (Eqs. 5.14). Find the parameter vector that minimizes the cost function through the search method selected.
5. Compute the estimation of the system noise variance based on the log-likelihood partial derivation (Equation 5.20).

## 5.3. Recursive Least-Squares

RLS techniques are quite useful methods for online estimation, by which acceptable predictions of the model parameter values can be achieved for different system dynamics. The Recursive Least-Squares algorithm is applicable for linear structures, although it can be extended to non-linear scenarios as explained by Young [46].

### 5.3.1. The deterministic RLS algorithm

Depending on the assumptions about the statistical nature of the signals or the remnant noise, and statistical information about the estimates, two types of algorithms can be found: deterministic or stochastic [46]. In this case, the deterministic RLS methodology is considered due to the lower number of assumptions it makes.

The RLS algorithm is an online estimation method that consists of 3 steps, in which the parameter vector $\theta(t_k)$ is adjusted online by means of the gain vector $g(t_k)$ and the prediction error of $u(t_k)$ [19, 46]. The gain vector is also tuned based on the scaled covariance matrix $P(t_k)$, which accounts for the certainty in the estimation conducted of each model parameter. Equations 5.21 present the formulas used in this iterative process:

$$\hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + g(t_k)\epsilon(t_k), \qquad or, \qquad \hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + P(t_k)\pi(t_k)\epsilon(t_k), \tag{5.21a}$$

$$g(t_k) = \frac{P(t_{k-1})\pi(t_k)}{1 + \pi^T(t_k)P(t_{k-1})\pi(t_k)}, \tag{5.21b}$$

$$P(t_k) = P(t_{k-1}) - g(t_k)\pi^T(t_k)P(t_{k-1}). \tag{5.21c}$$

Equation 5.21a can be evaluated in two ways due to the relationship $g(t_k) = P(t_k)\pi(t_k)$, although the first option is usually preferred since it is more computationally efficient [46].

The definition of vector $\pi(t_k)$ is based on the minimization of the instantaneous part of the cost function in Equation 5.8, in which the one-step-ahead prediction error can be approximated by the employment of the model parameters estimation at the previous $(k-1)^{th}$ instant:

$$V_k(\theta(t_k)) = \frac{1}{2}\left[\epsilon^2(t_k, \theta(t_k))\right] \approx V_k(\hat{\theta}(t_{k-1})) = \frac{1}{2}\left[\epsilon^2(t_k, \hat{\theta}(t_{k-1}))\right]. \tag{5.22}$$

As a result, the gain $\pi(t_k)$ is computed as the negative gradient of the prediction error [46]:

$$\pi(t_k) = -\frac{\partial\epsilon(t_k, \hat{\theta}(t_{k-1}))}{\partial\hat{\theta}(t_{k-1}))}. \tag{5.23}$$

### 5.3.2. Forgetting factor

In the deterministic RLS algorithm shown in Eqs. 5.21, all data until the $k^{th}$ instant, $Z^k$, are equally weighted over the observation time, which contributes to the implicit assumption that the model parameters remain constant. To solve this issue, the memory of the estimator can be shaped by using rectangular or exponential data weighting [46]. In the moving Rectangular Window (RW) RLS algorithm, a certain number of samples within an observation period are weighted equally, while in an Exponential-Weighting-into-the-Past (EWP) window each prediction error is taken into account by an exponential forgetting factor, $\lambda(t_k) \in [0, 1]$, in the cost function,

$$V_{EWP}(\hat{\theta}(t_{k-1})) = \frac{1}{2k}\sum_{i=1}^{i=k}\left[\epsilon^2(t_i, \hat{\theta}(t_{i-1}))\lambda(t_{k-i})\right]. \tag{5.24}$$

This forgetting factor is given by the expression $\lambda = exp(-T_s/T_e)$, where $T_s$ is the sampling time and $T_e$ is the memory horizon. The parameter $\lambda$ can be modified during the estimation process, or be maintained constant.

The EWP-RLS algorithm is more adequate in cases when model parameters change fast, and therefore, more recent data are more relevant for prediction. In addition, the forgetting factor offers the possibility of selecting a suitable value that works better for the system dynamics, which is not possible when a Rectangular Window is used. For a constant $\lambda$ parameter, these new RLS equations are defined as follows [19, 46]:

$$\hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + g(t_k)\epsilon(t_k), \tag{5.25a}$$

$$g(t_k) = \frac{P(t_{k-1})\pi(t_k)}{\lambda + \pi^T(t_k)P(t_{k-1})\pi(t_k)}, \tag{5.25b}$$

$$P(t_k) = \frac{1}{\lambda}\left[P(t_{k-1}) - g(t_k)\pi^T(t_k)P(t_{k-1})\right]. \tag{5.25c}$$

### 5.3.3. Forgetting matrix

The gain vector is tuned based on the scaled covariance matrix $P(t_k)$ and a factor $\lambda$, which can be defined as a real variable that affects all adjustable parameters equally, or as a diagonal matrix that imposes a different action on each adjustable parameter (forgetting matrix, $\Lambda$). Equations 5.26 present the formulas used in this iterative process [19, 46]:

$$\hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + g(t_k)\epsilon(t_k), \tag{5.26a}$$

$$g(t_k) = \frac{\Lambda P(t_{k-1})\Lambda\pi(t_k)}{1 + \pi^T(t_k)\Lambda P(t_{k-1})\Lambda\pi(t_k)}, \tag{5.26b}$$

$$P(t_k) = \Lambda P(t_{k-1})\Lambda - g(t_k)\pi^T(t_k)\Lambda P(t_{k-1})\Lambda. \tag{5.26c}$$

Where the forgetting matrix $\Lambda$ is a diagonal matrix composed of a sub-forgetting factor for each model parameter,

$$\Lambda = diag\left(\lambda_{a_1}, \ldots, \lambda_{a_{n_a}}, \lambda_{b_0}, \ldots, \lambda_{b_{n_b}}, \lambda_{c_1}, \ldots, \lambda_{c_{n_c}}, \lambda_{d_1}, \ldots, \lambda_{d_{n_d}}, \lambda_{f_1}, \ldots, \lambda_{f_{n_f}}\right)^{-1/2}. \tag{5.27}$$

Each element of the forgetting matrix must belong to the interval $I = [0, 1]$, where a $\lambda_i$ near a null value provides a negligible memory horizon, and values close to 1 increase this horizon significantly. Hence, for a forgetting factor $\lambda_i$, a total number of $N_i = 1/(1 - \lambda_i)$ samples are considered in the RLS algorithm for a time horizon of $T_{e,i} = T_s/(1 - \lambda_i)$.

## 5.4. ARX model estimation

ARX structures can be estimated by linear regression techniques [19], achieving a high computational efficiency. From the general expression for ARX models in Equation 5.5a, the vector of adjustable parameters $\theta_{ARX}$ is defined as follows:

$$\theta_{ARX} = \left[a_1^d, a_2^d, \ldots, a_{n_a}^d, b_0^d, b_1^d, \ldots, b_{n_b}^d\right]^T = \left[a_1^d, a_2^d, b_0^d, b_1^d\right]^T. \tag{5.28}$$

This vector $\theta_{ARX}$ can be found through two different estimation methods: Ordinary Least-Squares (OLS) and Recursive Least-Squares (RLS). In the first one, a standard least-squares technique is employed for all samples in a unique batch, obtaining a set of values of adjustable parameters for the entire simulation time. In the second one, such LS technique is applied recursively, updating the parameter estimation for each new measurement sample, giving as a result a different $\theta_{ARX}$ for each data point.

### 5.4.1. Ordinary Least-Squares

The derivatives (Eqs. 5.13) of the prediction error for an ARX model,

$$\epsilon(t_k, \theta) = A(z^{-1})u(t_k) - B(z^{-1})z^{-n_k}e(t_k), \tag{5.29}$$

are given by the following expressions:

$$\frac{\partial\epsilon(t_k,\theta)}{\partial a^d_{j=1,2}} = \frac{D(z^{-1})}{C(z^{-1})}z^{-j}u(t_k) = u(t_{k-j}), \qquad \frac{\partial\epsilon(t_k,\theta)}{\partial b^d_{j=0,1,\ldots,n_b}} = -\frac{D(z^{-1})}{F(z^{-1})C(z^{-1})}z^{-j-n_k}e(t_k) = -e(t_{k-j-n_k}). \tag{5.30}$$

Thus, the negative gradient of the prediction error would be equal to the regression vector $\varphi(t_k)$:

$$\begin{aligned}\pi(t_k) &= -\frac{\partial\epsilon(t_k,\hat{\theta}(t_{k-1}))}{\partial\hat{\theta}(t_{k-1})} = \left[-u(t_{k-1}), -u(t_{k-2}), \ldots, -u(t_{k-n_a}), e(t_{k-n_k}), e(t_{k-n_k-1}), \ldots, e(t_{k-n_k-n_b})\right]^T = \\ &= \left[-u(t_{k-1}), -u(t_{k-2}), e(t_{k-n_k}), e(t_{k-n_k-1})\right]^T = \varphi(t_k).\end{aligned} \tag{5.31}$$

Therefore, the cost function (Eq. 5.8) can be redefined by means of the regression vector after modifying the expression:

$$\begin{aligned}\hat{\theta}_{ARX} = \underset{\theta_{ARX}}{\arg\min}\left[\frac{1}{2N}\sum_{i=1}^{i=N}\epsilon^2(t_i,\theta)\right] &= \frac{1}{2N}\underset{\theta_{ARX}}{\arg\min}\left[\sum_{i=1}^{i=N}\left(A(z^{-1})u(t_k) - B(z^{-1})z^{-n_k}e(t_k)\right)^2\right] \\ &= \frac{1}{2N}\underset{\theta_{ARX}}{\arg\min}\left[\sum_{i=i_0}^{i=N}\left(u(t_i) - \varphi^T(t_i)\theta_{ARX}\right)^2\right]\end{aligned} \tag{5.32}$$

This loss function matches the one used in the OLS estimator [19], which means that the OLS method can be also considered as a type of prediction error method. Hence, ARX structures can be estimated by means of linear regression since the PEM optimization problem has an analytical solution.

In Equation 5.32, $i_0$ represents the index related to the start of the dataset employed in the OLS estimation. It is not necessary for all samples to be considered, since initial data points might be affected by transients, and then, it would be recommendable to discard them. Nevertheless, a prediction of the initial pairs $\{u(t_{k<1}), e(t_{k<1})\}$ (see Eq. 5.17) is usually required in order to employ the whole dataset (i.e., $i_0 = 1$). If such initial pairs estimation is not performed, then the minimum $i_0$ allowed is:

$$i_0 = max\left\{n_a + 1, n_b + n_k + 1\right\} = max\left\{3, n_k + 2\right\}. \tag{5.33}$$

Before finding the analytical solution to the minimization problem, the regression matrix and control-output vector must be defined. For an input-output dataset that consists of $N$ samples, the regression matrix is built based on the regression vector $\varphi(t_k)$ and an integer $i_0$:

$$\Phi = \begin{bmatrix}\varphi^T(t_{i_0}) \\ \varphi^T(t_{i_0+1}) \\ \vdots \\ \varphi^T(t_N)\end{bmatrix} = \begin{bmatrix}-u(t_{i_0-1}) & -u(t_{i_0-2}) & e(t_{i_0-n_k}) & e(t_{i_0-n_k-1}) \\ -u(t_{i_0}) & -u(t_{i_0}) & e(t_{i_0+1-n_k}) & e(t_{i_0-n_k}) \\ \vdots & \vdots & \vdots & \vdots \\ -u(t_{N-1}) & -u(t_{N-2}) & e(t_{N-n_k}) & e(t_{N-n_k-1})\end{bmatrix}, \tag{5.34}$$

while the control-output vector is:

$$U = \begin{bmatrix}u(t_{i_0}) & u(t_{i_0+1}) & \cdots & u(t_N)\end{bmatrix}^T. \tag{5.35}$$

The parameter vector $\theta_{ARX}$ can be computed by means of a QR-factorization solver, provided that the regression matrix is full rank. Hence, the final expression to calculate $\theta_{ARX}$ is:

$$\hat{\theta}_{ARX} = \left(\Phi^T\Phi\right)^{-1}\Phi^T U. \tag{5.36}$$

After computing the model parameter estimation, the noise variance is calculated based on Equation 5.20:

$$\hat{\sigma}^2_\varepsilon = \frac{1}{N}\left[\sum_{i=i_0}^{i=N}\left(u(t_i) - \varphi^T(t_i)\hat{\theta}_{ARX}\right)^2\right]. \tag{5.37}$$

### 5.4.2. RLS application

Based on results shown in [10], the selection of a forgetting matrix is more recommendable, thus, the RLS algorithm to be employed in this report is configured accordingly. Therefore, Equations 5.26 are used in the online ARX estimation process, where the forgetting matrix $\Lambda$ is a diagonal matrix of 4 elements,

$$\Lambda = diag\left(\lambda_{a_1}, \lambda_{a_2}, \ldots, \lambda_{a_{n_a}}, \lambda_{b_0}, \lambda_{b_1}, \ldots, \lambda_{b_{n_b}}\right)^{-1/2} = diag\left(\lambda_{a_1}, \lambda_{a_2}, \lambda_{b_0}, \lambda_{b_1}\right)^{-1/2}, \tag{5.38}$$

and the term $\Lambda P(t_{k-1})\Lambda$ is defined as follows:

$$\Lambda P(t_{k-1})\Lambda = \begin{bmatrix}\frac{P_{a_1}(t_{k-1})}{\lambda_{a_1}} & \frac{P_{a_1,a_2}(t_{k-1})}{\sqrt{\lambda_{a_1}\lambda_{a_2}}} & \frac{P_{a_1,b_0}(t_{k-1})}{\sqrt{\lambda_{a_1}\lambda_{b_0}}} & \frac{P_{a_1,b_1}(t_{k-1})}{\sqrt{\lambda_{a_1}\lambda_{b_1}}} \\ \frac{P_{a_1,a_2}(t_{k-1})}{\sqrt{\lambda_{a_1}\lambda_{a_2}}} & \frac{P_{a_2}(t_{k-1})}{\lambda_{a_2}} & \frac{P_{a_2,b_0}(t_{k-1})}{\sqrt{\lambda_{a_2}\lambda_{b_0}}} & \frac{P_{a_2,b_1}(t_{k-1})}{\sqrt{\lambda_{a_2}\lambda_{b_1}}} \\ \frac{P_{a_1,b_0}(t_{k-1})}{\sqrt{\lambda_{a_1}\lambda_{b_0}}} & \frac{P_{a_2,b_0}(t_{k-1})}{\sqrt{\lambda_{a_2}\lambda_{b_0}}} & \frac{P_{b_0}(t_{k-1})}{\lambda_{b_0}} & \frac{P_{b_0,b_1}(t_{k-1})}{\sqrt{\lambda_{b_0}\lambda_{b_1}}} \\ \frac{P_{a_1,b_1}(t_{k-1})}{\sqrt{\lambda_{a_1}\lambda_{b_1}}} & \frac{P_{a_2,b_1}(t_{k-1})}{\sqrt{\lambda_{a_2}\lambda_{b_1}}} & \frac{P_{b_0,b_1}(t_{k-1})}{\sqrt{\lambda_{b_0}\lambda_{b_1}}} & \frac{P_{b_1}(t_{k-1})}{\lambda_{b_1}}\end{bmatrix}. \tag{5.39}$$

In this case, the optimal forgetting matrix found by van Grootheest [10] is chosen, in which an infinite memory horizon is applied for poles ($a_1^d$ and $a_2^d$) and $N_i = 256$ samples are taken into account for zeros estimation ($b_0^d$ and $b_1^d$). Elements in the forgetting matrix are kept constant during the entire simulation time, which gives as a result an exponential discount of old measurements in the estimation of $b_0^d$ and $b_1^d$. Therefore, the matrix used is:

$$\Lambda = diag\left(1, 1, \frac{N_{b_0}-1}{N_{b_0}}, \frac{N_{b_1}-1}{N_{b_1}}\right)^{-1/2} = diag(1, 1, 0.99609, 0.99609)^{-1/2}. \tag{5.40}$$

Also, the covariance matrix can be initialized as a diagonal matrix whose components are equal to 0.1, i.e.:

$$P_0 = diag\,(0.1, 0.1, 0.1, 0.1)\,. \tag{5.41}$$

## 5.5. BJ model estimation

BJ structures can not be estimated by linear regression techniques used in ARX models due to their non-linear nature, thus, more adequate optimization methods are required to find the model parameters that minimize the cost function. From the general expression for BJ models in Equation 5.5b, the vector of adjustable parameters $\theta_{BJ}$ is defined as follows:

$$\theta_{BJ} = \left[a_1^d, a_2^d, \ldots, a_{n_f}^d, b_0^d, b_1^d, \ldots, b_{n_b}^d, c_1^d, c_2^d, \ldots, c_{n_c}^d, d_1^d, d_2^d, \ldots, d_{n_d}^d\right]^T = \left[a_1^d, a_2^d, b_0^d, b_1^d, d_1^d, \ldots, d_m^d\right]^T \tag{5.42}$$

As mentioned before, discrete-time parameters $f_i^d$ are called $a_i^d$ in BJ structures, so that the equations for the polynomial $F(z^{-1})$ will be used under the name of $A(z^{-1})$. Additionally, one less equation is to be required since the order of $C(z^{-1})$ is null. This vector $\theta_{BJ}$ can be found through two different algorithms: Prediction Error Method and Recursive Prediction Error minimization.

### 5.5.1. PEM application

From the prediction error (Eq. 5.4),

$$\epsilon(t_k, \theta_{BJ}) = D(z^{-1})\left[u(t_k) - \frac{B(z^{-1})}{F(z^{-1})}z^{-n_k}e(t_k)\right], \tag{5.43}$$

the minimization problem can be defined as follows:

$$\hat{\theta}_{BJ} = \underset{\theta_{BJ}}{\arg\min}\, V_N(\theta_{BJ}, Z^N) = \underset{\theta_{BJ}}{\arg\min}\left\{\frac{1}{2N}\sum_{i=i_0}^{i=N}\left[D(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})}e(t_i)\right]^2\right\}. \tag{5.44}$$

Hence, the optimization model is represented by the following equations:

$$\frac{\partial V_N}{\partial a_{j=1,2}^d} = \frac{1}{N}\sum_{i=i_0}^{i=N}\left[D(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})}e(t_i)\right] \times \frac{\overline{B}(z^{-1})D(z^{-1})}{F^2(z^{-1})}z^{-j}e(t_i) = 0, \tag{5.45a}$$

$$\frac{\partial V_N}{\partial b_{j=0,1}^d} = \frac{1}{N}\sum_{i=i_0}^{i=N}\left[D(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})}e(t_i)\right] \times \frac{D(z^{-1})}{F(z^{-1})}z^{-j-n_k}e(t_i) = 0, \tag{5.45b}$$

$$\frac{\partial V_N}{\partial d_{j=1,2,\ldots,n_d}^d} = \frac{1}{N}\sum_{i=i_0}^{i=N}\left[D(z^{-1})u(t_i) - \frac{\overline{B}(z^{-1})D(z^{-1})}{F(z^{-1})}e(t_i)\right] \times \left[z^{-j}u(t_i) - \frac{\overline{B}(z^{-1})z^{-j}}{F(z^{-1})}e(t_i)\right] = 0. \tag{5.45c}$$

In BJ model estimation, the index $i_0$ should be always equal to 1 due to the higher relevance of the initial states in the optimization process, so that the whole dataset is utilised in the PEM algorithm. In this scenario, an initial model needs to be estimated beforehand to find the pairs $\{u(t_{k<1}), e(t_{k<1})\}$ based on the initial conditions $\theta_{BJ}^0$:

$$\hat{Z}_0^N = \underset{Z_0^N}{\arg\min}\, V_N^0(\theta_{BJ}^0, Z_0^N) = \underset{Z_0^N}{\arg\min}\left\{\frac{1}{2N}\sum_{i<1}\left[D_0(z^{-1})u(t_i) - \frac{B_0(z^{-1})D_0(z^{-1})}{F_0(z^{-1})}z^{-n_k}e(t_i)\right]^2\right\}. \tag{5.46}$$

Nevertheless, initial data can also be discarded (only if the Matlab function *pem.m* is not been employed), but the minimum $i_0$ allowed must be defined based on the Maclaurin series of the most delayed term, i.e., $\frac{\overline{B}(z^{-1})D(z^{-1})}{F^2(z^{-1})}z^{-j}$. For instance, if a first-order series is assumed, the minimum index should be:

$$i_0 = n_b + n_d + 3n_f + n_k + 1 = n_d + n_k + 8. \tag{5.47}$$

In order to find the optimal solution to the non-linear problem shown in Equations 5.45, the most adequate and computationally efficient approach is to use gradient-based schemes based on the ones initially employed by Box and Jenkins [3], although many other useful methodologies are being used currently as well. For instance, Matlab function *pem.m* offers the possibility of combining a set of line search algorithms (*'gn'* - Subspace Gauss-Newton Least-Squares, *'lm'* - Levenberg-Marquardt Least Squares, *'gna'* - Adaptive Subspace Gauss-Newton, and *'grad'* - Steepest Descent Least-Squares) to make the estimation process more robust and efficient [18]. This option can be selected by setting *'SearchMethod'* to *'auto'*

Finally, the noise variance is computed from the model parameter estimates:

$$\hat{\sigma}_{\varepsilon}^2 = \frac{1}{N}\sum_{i=1}^{i=N}\left[D(z^{-1})u(t_i) - \frac{B(z^{-1})D(z^{-1})}{F(z^{-1})}z^{-n_k}e(t_i)\right]^2. \tag{5.48}$$

## 5.5.2. Recursive Prediction Error minimization

From Equations 5.13, the prediction errors derivatives can be defined as follows:

$$\frac{\partial\epsilon(t_k)}{\partial a_i^d} = \frac{D(z^{-1})}{F(z^{-1})C(z^{-1})}\frac{B(z^{-1})}{F(z^{-1})}e(t_{k-i-n_k}) = \frac{D(z^{-1})}{F(z^{-1})C(z^{-1})}y(t_{k-i-n_k}) = y_{f_1}(t_{k-i-n_k}), \tag{5.49a}$$

$$\frac{\partial\epsilon(t_k)}{\partial b_i^d} = -\frac{D(z^{-1})}{F(z^{-1})C(z^{-1})}e(t_{k-i-n_k}) = -e_{f_1}(t_{k-i-n_k}), \tag{5.49b}$$

$$\frac{\partial\epsilon(t_k)}{\partial c_i^d} = \frac{1}{C(z^{-1})}\left[u(t_{k-i}) - \frac{B(z^{-1})}{F(z^{-1})}e(t_{k-i-n_k})\right] = \frac{1}{C(z^{-1})}\xi(t_{k-i}) = \xi_{f_2}(t_{k-i}), \tag{5.49c}$$

$$\frac{\partial\epsilon(t_k)}{\partial d_i^d} = -\frac{1}{C(z^{-1})}\left[\frac{D(z^{-1})}{C(z^{-1})}u(t_{k-i}) - \frac{D(z^{-1})B(z^{-1})}{C(z^{-1})F(z^{-1})}e(t_{k-i-n_k})\right] = -\epsilon_{f_2}(t_{k-i}). \tag{5.49d}$$

Where the subscripts $f_1$ and $f_2$ denote that the variable is filtered by the transfer functions:

$$f_1 = \frac{D(z^{-1})}{C(z^{-1})F(z^{-1})}, \qquad f_2 = \frac{1}{C(z^{-1})}. \tag{5.50}$$

Therefore, the negative gradient $\pi(t_k)$ presents the following expression:

$$\begin{aligned}\pi(t_k) = -\frac{\partial\epsilon}{\partial\hat{\theta}} = \big[&-\hat{y}_{\hat{f}_1}(t_{k-1-n_k}),\ldots,-\hat{y}_{\hat{f}_1}(t_{k-n_f-n_k}),e_{\hat{f}_1}(t_{k-n_k}),\ldots,e_{\hat{f}_1}(t_{k-n_b-n_k}),-\hat{\xi}_{\hat{f}_2}(t_{k-1}),\ldots,-\hat{\xi}_{\hat{f}_2}(t_{k-n_c}),\\ &\hat{\epsilon}_{\hat{f}_2}(t_{k-1}),\ldots,\hat{\epsilon}_{\hat{f}_2}(t_{k-n_d})\big]^T = \left[-\hat{y}_{\hat{f}_1}(t_{k-1-n_k}),-\hat{y}_{\hat{f}_1}(t_{k-2-n_k}),e_{\hat{f}_1}(t_{k-n_k}),e_{\hat{f}_1}(t_{k-1-n_k}),\hat{\epsilon}(t_{k-1}),\ldots,\hat{\epsilon}(t_{k-n_d})\right]^T,\end{aligned} \tag{5.51}$$

where the prefilters $\hat{f}_1$ and $\hat{f}_2$ are the latest estimated prefilters,

$$\hat{f}_1 = \frac{\hat{D}(z^{-1})}{\hat{F}(z^{-1})}, \qquad \hat{f}_2 = 1, \tag{5.52}$$

and variables $\hat{y}(t_k)$ are calculated from the latest estimated polynomials,

$$\hat{y}(t_k) = \frac{\hat{B}(z^{-1})}{\hat{F}(z^{-1})}e(t_{k-n_k}). \tag{5.53}$$

A Recursive PEM algorithm [46] is built, by analogy with the RLS method defined in Equations 5.21 and fulfilling the theoretical requirements to achieve convergence. Since recursive function *rpem.m* in Matlab only admits a unique forgetting factor (there is no option for forgetting matrix), the RPEM algorithm to be used is given by the following steps:

$$\hat{\theta}_{BJ}(t_k) = \hat{\theta}_{BJ}(t_{k-1}) + P(t_k)\pi(t_k)\epsilon(t_k), \tag{5.54a}$$

$$g(t_k) = \frac{P(t_{k-1})\pi(t_k)}{\lambda + \pi^T(t_k)P(t_{k-1})\pi(t_k)}, \tag{5.54b}$$

$$P(t_k) = \frac{1}{\lambda}\left[P(t_{k-1}) - g(t_k)\pi^T(t_k)P(t_{k-1})\right]. \tag{5.54c}$$

Thus, the classical RLS algorithm is adapted to non-linear cases by means of converting $\pi(t_k)$ into a vector composed of linear variables, as shown in Equation 5.51. A forgetting factor $\lambda = 0.99609$ is recommended for

the implementation of the recursive BJ method, based on previous results from Van Grootheest [10]. In addition, following the RLS initialization in ARX estimation, the initial covariance matrix can be defined as:

$$P_0 = diag\,(0.1, 0.1, 0.1, 0.1 \times n_d)\,. \tag{5.55}$$

In addition, other methods can be applied to the online BJ structure estimation problem, such as the Real-Time Recursive Refined Instrumental Variable (RRIV) or the Extended Kalman Filter (EKF) [46]. The RRIV method is probably the most similar to the RPEM algorithm, although it presents some differences in terms of the covariance matrix definition (two sub-matrices with null off-diagonal blocks are used in RRIV), the robustness of the algorithm (the Instrumental Variable modifications ensure a stable estimation process), or the steps required (while RPEM is fully recursive, RRIV needs to iterate at each $k^{th}$ period).

## 5.6. Quality-of-fit metrics

In order to verify the quality of the estimation performed, several metrics need to be used. These metrics have to be relevant in terms of addressing how well the predicted model represents the dynamics of the real one. To achieve such a goal, the prediction error in estimated model parameters has to be analysed, but the prediction capabilities of such a model must be also studied.

Hence, two different quality-of-fit metrics are employed: the Variance Accounted For (VAF) and the relative bias ($B_{r,i}$). Both have been employed in previous works to verify the ARX results [10, 36], thus, these ones should be also applicable to the BJ results. The VAF metric,

$$VAF^{n_k^*,m^*} = max\left\{0, \left(1 - \frac{\sum_{k=1}^{N} \mid \tilde{u}(t_k) - \hat{u}^{n_k^*,m^*}(t_k) \mid^2}{\sum_{k=1}^{N} \mid \tilde{u}(t_k) \mid^2}\right) \cdot 100\%\right\}, \tag{5.56}$$

evaluates the correctness of a model, by comparing the real output, $\tilde{u}(t_k)$, with the estimated output of the model, $\hat{u}^{n_k^*,m^*}(t_k)$, for a certain combination of time delay and remnant filter order in the model structure. The VAF of two signals that are the same is 100%, while it will be lower if they differ. On the other hand, the relative bias can be used to verify the accuracy of model parameter estimation,

$$B_{r,i} = \left(\frac{\hat{\theta}_i - \theta_i^0}{\theta_i^0}\right) \cdot 100\%. \tag{5.57}$$

This bias is applied to each coefficient in order that errors in estimation can be found more easily. In addition, the absolute relative bias is addressed in special scenarios when the standard $B_{r,i}$ does not offer enough clarity:

$$\mid B_{r,i} \mid = \left|\frac{\hat{\theta}_i - \theta_i^0}{\theta_i^0}\right| \cdot 100\%. \tag{5.58}$$

# 6

# Preliminary simulation analysis

In this chapter, the different results obtained for ARX and BJ models are shown for batch-fitting estimation. In addition, the results from the recursive ARX algorithm are shown, together with the future perspectives for the BJ algorithm in online estimation.

Previous investigations [30, 49, 5] confirm that it is quite important to assess a novel identification method's accuracy using Monte Carlo simulations before applying it to real-world activities or laboratory tests. Therefore, such a simulation is performed for $M = 100$ realizations of the remnant noise, in which all possible scenarios are covered for each case. Simulation conditions C1 and C2 are addressed in batch-fitting estimation, while condition C3 is evaluated by the recursive algorithm. Remnant filter orders $m^0 \in \{1, 2, 3, 4\}$ are simulated, so that their effect on the HO parameters estimation quality can be properly analysed. In addition, ARX and BJ models are built for multiple HO time delays, $n_k^* \in \{25, 26, 27, 28, 29, 30\}$, and noise levels, $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Only a remnant time constant $T_n = 0.06s$ is considered due to its low relevance in the estimators' capabilities.

Since the configuration of BJ model structures and their optimization processes are more complex than ARX ones, a more detailed study of the model remnant orders and initial conditions to be employed is required. In relation to the model remnant filter structure, two possible strategies are analysed: the model $m - th$ order matches the simulated one, or the model $m - th$ is always equal to 1. Attending to changes in initial conditions of the estimated discrete-time BJ parameters, a total of 8 different cases are evaluated with respect to the reference I.C.

Moreover, no decimation analysis has been conducted in this report, since the goals of this research are only to develop a novel estimator based on BJ model structures and compare its performance with results from the ARX models. When a decimation analysis is performed, a Padé approximation [42] must be implemented whether $n_k^0$ is not an integer due to the variation in time step $T_s$. This approximation introduces additional zeros and poles in the discrete-time transfer function of the BJ model, which will interfere with the estimation of the rest of the parameters. Thus, such an analysis is considered to be out of the scope of this MSc Thesis.

## 6.1. Batch-fitting

### 6.1.1. ARX results

For simulation condition C1, Figures C.1 and C.2 show the relative bias in discrete- and continuous-time parameters estimated through ARX models, respectively. High errors are encountered for first order remnant filter, and particularly in $b_0^d$ and $b_1^d$, since the number of poles in the simulated filter is lower than the one corresponding to the poles introduced by the ARX model. Thus, the ARX model sacrifices quality of fit in HO parameters to explain the remnant noise dynamics more precisely, by means of a filter structure unable to model a first-order one. On the other hand, acceptable relative bias results are found for remnant orders greater than 1 (due to the fact that the remnant filter presents at least 2 poles), although significant errors can still be observed for discrete-time zeros. Hence, no consistency in results is possible through ARX models whether the remnant filter order is modified in the human operator.

Model time delay changes produce a linear trend in computed relative bias, however, these ones do not oscillate around zero and no optimal results are found for actual HO time delay (i.e., $n_k^* = 29$). Only the HO gain $K_e$ results in Figure C.2 achieve the ideal performance. Furthermore, null biases are achieved for remnant-free cases when the model time delay is the correct one, which means that the ARX structure is adequately implemented.

Additionally, increments in noise level lead to a bigger bias in general, which affects the stability of the ARX estimator. Since the model filter structure is wrong and fixed for changes in $m^0$, HO parameters results will be affected by stronger remnant dynamics. Therefore, the ARX model can only be considered as a robust estimation technique when really small noise level remnant noises are introduced in the system.

For simulation condition C2 (see Figures C.9 and C.10), similar results in discrete-time parameters can be observed. In relation to the HO coefficients, $K_e$ and $T_L$ present a greater bias when compared to the results for C1, while estimations of $\omega_{nm}$ and $\eta_{nm}$ are slightly improved. This can be explained by the fact that the location of the zero for C2 is moved to a lower frequency, so that the estimation of HO parameters in the numerator is going to miss a valuable part of the information from the HO dynamics generated by the forcing function (i.e., the frequencies of some sinusoidals in the forcing function will be further from the frequency corresponding to the HO's zero).

Figures D.1 to D.4 show the Bode plots of the computed discrete-time model for multiple simulation remnant orders, HO time delays, and noise levels. For the remnant-free case, the computed discrete-time transfer functions from ARX models offer satisfactory results for all remnant orders in terms of magnitude and phase, but also in the location of the zero and poles. Nevertheless, in non-null remnant noise cases, poles are not correctly captured and the heights of their associated peaks present a considerable error with respect to the reference value. Phase angle deviations with respect to the reference case are relatively small, except for the frequency region near the peak at poles, since ARX model fit is not accurate enough for discrete-time parameters $a_1^d$ and $a_2^d$. Also, the correct location of poles is completely missed when a first-order remnant filter is simulated, which leads to huge errors in phase angle.

For higher remnant orders in simulation data, the location of poles moves towards the left of the graph. Seeing that the HO transfer function and the remnant filter share poles in the ARX model, the optimal solution found by the estimator to properly approximate the HO neuromuscular dynamics and remnant filter structure is to locate such poles at lower frequencies. Additionally, this re-location towards the left of Bode plots produces higher peaks at poles due to the weighting function in the bias minimization process, as shown by van Grootheest [10].

Regarding the modification in HO time delay, ARX estimators provide poles at lower frequencies when smaller $n_k^*$ are chosen, and vice versa. Therefore, the ARX model 'generates lag' to compensate for smaller model time delays, while it 'creates lead' for greater time delays, in order that the ARX model phase captures more precisely the real delay in the system.

Figures D.10 to D.13 present the resultant Bode plots for simulation condition C2. In this case, a higher peak is generated due to the location of the zero at a lower frequency, which creates lead earlier. Similar results are obtained when compared to the previous simulation condition, however, the HO zero is captured by the ARX model less accurately since it is in less direct contact with HO dynamics information coming from sinusoidals' frequencies of the forcing function.

### 6.1.2. BJ results

For simulation condition C1, Figures C.3 and C.4 show the relative bias in discrete- and continuous-time HO parameters, while results for remnant parameters can be appreciated in Figure C.5.

In this case, the high errors for first-order filter simulations are diminished, since the number of poles placed in the BJ model filter matches the simulation scenario. In fact, results obtained for all model remnant orders are quite consistent, so that the BJ estimator could be considered quite robust when facing different remnant orders.

Furthermore, relative bias results oscillate around zero when the model time delay is modified from the reference value $n_k^* = n_k^0 = 29$. Previous ARX results showed that relative bias increased or decreased in a linear tendency with respect to the reference time delay, but it did not fluctuate with respect to a zero value. Thus, the optimization process of the BJ estimator is able to converge to solutions really close to the real HO parameters, while the ARX estimator presents a clear bias in its results due to the unreal remnant filter model structure. Moreover, relative biases can still be considerable for $b_0^d$ and $b_1^d$ in scenarios with model time delays far from the reference one, but those ones are a bit smaller than in the ARX case.

An increase in the noise level does not lead to a greater relative bias in general, but results fluctuate around the bias of the remnant-free case at each model time delay. Hence, the BJ estimator can adapt well enough to simulation data in which the remnant dynamics are quite relevant.

Regarding the remnant parameters in Figure C.5, unsatisfactory results are found for remnant orders higher than 1. Since the coefficient $T_n$ is obtained from the discrete-time remnant parameters such as $d_1^d$, as explained in Section B.2, its estimation is contingent upon a high variability and bias when model remnant orders greater than 1 are chosen. In this way, relative biases around 100% are obtained in the results. The modified remnant noise variance $\sigma_\epsilon^2$, presents an important deviation with respect to the reference value as well, which increases for higher remnant orders. This wrong prediction of the variance could be originated from the effort of the BJ estimator in explaining the HO remnant dynamics by means of adjusting the discrete-time remnant parameters instead of fitting $\sigma_\epsilon^2$, more precisely. Finally, the estimation of the remnant gain also presents important errors for $m - th$ orders greater than 1 since it depends on how accurately the parameters $T_n$ and $\sigma_\epsilon^2$, are predicted (see Equation B.15).

Figures C.11 to C.13 show the relative bias results for simulation condition C2. Similar results are obtained, however, the parameters $K_e$ and $T_L$ associated with the HO zero present a less precise estimation, while the neuromuscular dynamics are better captured. Attending to the remnant parameters, the results in the modified noise variance (and the remnant gain in consequence) have improved, although the variability of the obtained biases is much higher than for simulation condition C1.

Figures D.5 to D.8 present the plots from the obtained BJ discrete-time models. In remnant-free cases, BJ models present higher deviations than ARX models with respect to the reference transfer function. The BJ estimator is trying to explain a fictional filtered remnant noise with part of HO dynamics, which leads to errors in parameter prediction when the correct model time delay is not selected. In particular, a remnant order equal to 2 produces the highest deviations, since it matches the order of the neuromuscular dynamics (estimations of neuromuscular and remnant poles overlap). Regarding non-null noise level scenarios, the results obtained are excellent since the approximation of the HO poles and zero, together with the HO gain, are really accurate. In addition, a similar effect in relation to lead/lag generation is given in the BJ estimator in comparison with the ARX one, seeing that BJ poles are placed at lower frequencies for smaller model time delays to create additional lag, and vice versa.

In Figure D.9, the Bode plots from ARX and BJ models can be observed for different noise levels and remnant orders, while fixing the model time delay as $n_k^* = n_k^0 = 29$, so that it matches the simulation one perfectly. Except for the remnant-free case, in which ARX and BJ offer comparable results, the BJ model always outperforms the ARX one, both in magnitude and phase estimation. When the ARX estimator has difficulties locating the system poles and peak magnitudes, the estimation method based on BJ structures provides an accurate solution (no additional lead/lag is required by the BJ estimator since the model time delay is the real one).

Figures D.14 to D.17 show the BJ Bode plots for simulation condition C2. Similar to the ARX case, the placement of the zero is estimated less accurately than for simulation condition C1, since less information from HO dynamics due to the forcing function is available for this issue. Also, the HO gain estimation is poorer than in the previous simulation condition C1, which can be observed clearly in the initial magnitude values of the Bode plot. Nevertheless, the pole estimation still remains acceptable.

In Figure D.18, the comparison between ARX and BJ Bode plots is shown for simulation condition C2. Almost the same results are obtained as for simulation condition C1. The BJ estimator is capable of capturing the HO dynamics perfectly when the model time delay chosen is the real one.

### 6.1.3. Implementation of $m^* = 1$ in BJ model

When recursive algorithms based on BJ are applied, the model remnant order $m^*$ can be a variable to predict based on simulation data, or can be fixed as long as it does not affect the quality of estimation results. The ideal BJ structure should incorporate a model remnant order such that it is equal to the simulation one (i.e., $m^* = m^0$), however, this is not always possible in real cases when the estimator is applied online for pilot data, since the implemented algorithm might not be capable of determining which remnant order is the real one (or even the pilot's remnant filter order could be in transition between two integer values). Therefore, fixing the model remnant order may be a recommendable choice to simplify the complexity of the model parameters estimation problem, which implies a decrease in computational cost that can be invested in a higher emphasis on the HO time delay identification.

In order to verify whether a fixed model remnant order is an acceptable option, the VAF and relative bias metrics [1] can be employed to analyse the performance of the BJ estimator for multiple combinations of model remnant order $m^*$ and time delay $\tau_e^*$. Hence, a set of colour contour maps (see Appendix E) has been generated for simulation conditions C1 and C2 and simulation remnant orders $m^0 \in \{1, 2, 3, 4\}$, while the model time delay and remnant order are selected from the range $\tau_e^* \in \{25, 26, 27, 28, 29, 30\}$ and $m^* \in \{1, 2, 3, 4\}$ at each case, respectively.

For simulation condition C1, Figures E.1, E.3, E.5 and E.7 show the VAF metric results for $m^0 \in \{1, 2, 3, 4\}$ and multiple noise levels $P_n \in \{0.0, 0.1, 0.2, 0.3\}$. In all these graphs, the VAF is similar for each combination $\{m^*, \tau_e^*\}$, although Figure E.1 reveals that a model remnant order equal to 4 (the simulation one is 1 for this case) provides lower VAF results. For simulation condition C2, Figures E.9, E.11, E.13 and E.15, same metrics are obtained at each combination of BJ model parameters, nevertheless, a more intense contrast can be observed in Figure E.9 when the model remnant order does not match the real one. Hence, the VAF metrics could only be useful to find whether a first-order remnant filter is present in the human operator, but no conclusions can be extracted from results when $m^0 > 1$.

On the other hand, absolute bias results for the HO discrete-time, $\left\{b_0^d, b_1^d, a_1^d, a_2^d\right\}$, and remnant parameters, $\left\{\sigma_{\epsilon'}^2, d_1^d\right\}$, are more informative about the estimated BJ model at each combination $\{m^*, \tau_e^*\}$. For simulation condition C1, Figures E.2, E.4, E.6 and E.8 show a clearer minimum relative bias in the area surrounding the simulation configuration $\left\{m^0, \tau_e^0\right\}$ for HO discrete-time parameters results. However, as can be appreciated from these figures, the time delay is a more determinant factor than the remnant order in terms of relative bias reduction. Only in the case $m^0 = 1$ (see E.2), there is clear evidence that a selection of a model remnant order that matches the simulation one reduces the bias significantly.

In relation to the remnant parameters, $\left\{\sigma_{\epsilon'}^2, d_1^d\right\}$, the opposite conclusion can be drawn. The time delay becomes irrelevant in terms of bias reduction, while the remnant order is fundamental. This is due to the fact that such parameters are much more influenced by the remnant dynamics than the HO discrete-time parameters.

[1] In this subsection, absolute relative bias results are presented instead of standard bias, since an absolute value permits a better visual finding of minimum bias region on the colour contour map.

For simulation condition C2, Figures E.10, E.12, E.14 and E.16 confirm the mentioned statements. Figure E.10 presents an even clearer contrast between results for model remnant orders closer to 1 and higher ones, while Figures E.12, E.14 and E.16 show the great relevance of correct estimation of the HO time delay in the BJ model structure.

When comparing relative bias results between BJ estimators with $m^* = m^0$ and $m^* = 1$ (see Appendix C), it can be concluded that the alternative of fixed model remnant order is a valid option. From Figures C.6 to C.8 for simulation condition C1, and Figures C.14 to C.16 for C2, show that similar relative bias values are achieved through a BJ model structure with $m^* = 1$. The medians of the computed biases in the discrete-time parameters are almost the same as the values obtained by a BJ model with $m^* = m^0$, however, the interquartile range may increase slightly owing to the variability increment. Regarding the HO continuous-time parameters, very similar values are achieved in the neuromuscular dynamics, although bias in $K_e$ and $T_L$ become noticeably higher when the model time delay differs from the simulation one. Nevertheless, these increments in bias are still assumable and close to a zero value for $\tau_e^* = \tau_e^0$. Finally, the remnant parameters are obviously wrongly estimated for the $m^* = 1$ case, since these ones are quite sensitive to a possible mismatch between the model remnant filter structure and the simulation one.

Thus, based on the results shown in Appendix E, a fixed model remnant order $m^* = 1$ would not only simplify the BJ structure to be computed, but it would also improve the results obtained for $m^0 = 1$ simulations when high model remnant orders are chosen due to possible failures during the optimization process. However, the implementation of such a BJ model structure could imply a small decrease in accuracy estimation for high simulation remnant orders.

### 6.1.4. Modification of initial conditions in BJ model parameters

The optimization of the BJ model parameters is a non-linear process, and consequently, the initial conditions (IC) of such parameters are quite relevant for the convergence and quality of the estimations. Optimizations from non-ideal initial conditions can provide convergence to local minima of the cost function, but maybe not the global one. Furthermore, ICs with important deviations from the reference parameters could even produce divergence in the estimation process. Therefore, it is fundamental to have a proper understanding of the effects of changes in the initial value of each HO parameter.

In Appendix F, Figures F.1 to F.8 show the differences (for simulation condition C1) between the computed relative biases for the new IC ($B_r^{IC_{new}}$) and the values obtained from the base case in Appendix C ($B_r^0$). The initial conditions employed for this section are recorded in Table 6.1:

| I.C. Number | $b_0^d\,[-]$ | $b_1^d\,[-]$ | $a_1^d\,[-]$ | $a_2^d\,[-]$ |
|---|---|---|---|---|
| Base | 0.075 | −0.075 | −1.912 | 0.924 |
| $IC_1$ | 0.075 | −0.075 | −1.812 | 0.824 |
| $IC_2$ | 0.075 | −0.075 | −1.962 | 0.974 |
| $IC_3$ | 0.050 | −0.050 | −1.912 | 0.924 |
| $IC_4$ | 0.050 | −0.050 | −1.812 | 0.824 |
| $IC_5$ | 0.050 | −0.050 | −1.962 | 0.974 |
| $IC_6$ | 0.100 | −0.100 | −1.912 | 0.924 |
| $IC_7$ | 0.100 | −0.100 | −1.812 | 0.824 |
| $IC_8$ | 0.100 | −0.100 | −1.962 | 0.974 |

**Table 6.1:** Initial conditions of discrete-time parameters in BJ models.

In general, the medians of relative bias differences are null or close to zero, which means that all ICs from Table 6.1 provide a similar output on average. However, the variability can be noticeable in some cases, while several particular simulations can lead to considerable differences with respect to the base case. That is the effect of the local minimum finding of the cost function instead of localising the global one. In addition, it can be observed that for those ICs in which there is a higher deviation of the initial $a_1^d$ and $a_2^d$ values with respect to the reference ones, there is a bigger difference in bias for $b_0^d$ and $b_1^d$, while for greater deviations in $b_0^d$ and $b_1^d$ with respect to the reference ones, less bias differences are encountered. This can be explained by the frequency range of the forcing function, since it provides more information to the BJ estimator for zero prediction than for poles, as explained previously. Therefore, the BJ model optimization is more sensitive to high initial deviations for the poles than for the zero location, i.e., small initial errors in $a_1^d$ or $a_2^d$ can make a more negative impact on the estimation quality than great initial deviations in $b_0^d$ and $b_1^d$.

IC 3 (see Figure F.3) contains the set of initial discrete-time parameters closest to the reference ones. In this case, since both the poles and the zero do not contain high initial deviations from the simulated data, relative bias results are even improved when compared to the base case.

## 6.2. Recursive-fitting

### 6.2.1. ARX results

For simulation condition C3, remnant orders $m^0 \in \{1, 2, 3, 4\}$ and noise levels $P_n \in \{0.0, 0.1, 0.2, 0.3\}$, the discrete- and continuous-time parameters of a human operator are estimated online by means of a recursive ARX algorithm. Figures G.1 to G.4 present the predicted HO discrete-time parameters, while Figures G.5 to G.8 show the continuous-time ones.

As explained before, first-order remnant simulations are quite difficult for an ARX estimator to model properly, because a persistent bias is accumulated in the feedback loop due to the mismatch of the quasi-linear transfer function and remnant filter poles. For instance, Figure G.1 depicts a huge bias in $a_1^d$ and $a_2^d$ parameters, which even does not respect the relationship from Appendix B.3 in the coefficient $a_2^d$, giving, as a result, many divergences in the computation of HO continuous-time parameters. In the case of $b_0^d$ and $b_1^d$, the estimation errors are more intense, although the transition from first- to second-order dynamics is perceived by the recursive algorithm at least.

When the simulation remnant order is increased, the bias is heavily reduced, as shown in batch-fitting results. In addition, an increment in noise level clearly produces a bigger error in $b_0^d$ and $b_1^d$ parameters with respect to their reference values.

Regarding continuous-time results, the estimator provides a wrong prediction of the neuromuscular parameters for a first-order remnant due to the huge biases in $a_1^d$ and $a_2^d$, together with important divergences in the computation of $T_L$ (condition $a_2^d > 0$ is not fulfilled at multiple occasions). Nonetheless, the estimation of the HO gain is always accurate. In relation to higher remnant orders, estimation biases decrease in all parameters, achieving a more adequate parameter prediction.

### 6.2.2. Perspectives for BJ results

Based on the great advantages that the usage of BJ models has shown for batch-fitting, the recursive BJ estimator is expected to:

- Diminish the huge bias encountered in recursive ARX results for a first-order remnant filter.
- Solve the instabilities in $T_L$ online estimation when the controlled element acts as a second-order integrator.
- Improve the stability of the transitions between estimations from controlled elements with first- to second-order integrator dynamics.
- Provide online estimation results with a low error when high remnant filter orders are implemented in the simulation.
- Be quite dependent on the initial conditions of the discrete-time parameters.

# 7

# Conclusions and Future Works

It is been proved that estimators based on BJ structures can provide more adequate results than the ones based on ARX models. BJ techniques are more robust when facing different remnant order filters, and particularly, for first-order ones. The flexibility of BJ structures in introducing a certain number of poles in the remnant filter, which also can differ from the NMS ones, allows a more accurate match between the real and estimated HO model.

In addition, more stable results are obtained for high noise level scenarios, since BJ biases are kept close to the one achieved in the remnant-free case, while bias in ARX estimations presents greater oscillations. Therefore, higher noise levels will lead to worse estimation results in ARX models, while BJ ones are still capable of reducing the amount of bias produced.

Regarding the Bode plots of estimated models, BJ structures are also able to achieve a more satisfactory identification of the real HO zeros and poles. The difference in the accuracy of pole identification can be observed for both simulation conditions, while the effect of correct zeros estimation is especially noticeable for condition C2[1]. The errors in phase estimation are significantly improved with BJ structures, and also, BJ models are capable of finding the magnitude peak more precisely. For the ideal model time delay (see Figures D.9 and D.18), ARX models show a poorer identification at all remnant orders in comparison to BJ results (particularly in first-order scenarios).

Moreover, the analysis conducted in Section 6 shows that BJ structures with unitary remnant order can achieve a similar performance than whether correct model remnant orders were used in each case. This implies that BJ estimation procedures can be considerably simplified, because all the attention can be put on the HO time delay estimation at real identification tasks while keeping a unitary remnant order. To determine such real HO time delay, bigger intervals for $n_k^*$ are recommended due to the limitations in the VAF metric to find the actual $n_k^0$, since BJ estimators still show quite acceptable converged results for model time delays close to $n_k^0$.

Nevertheless, BJ models need to be initialised in order that the non-linear estimation process can converge. ARX estimators present a greater advantage in this way since convergence is guaranteed as long as the regression matrix is full rank. On the other hand, batch-fitting BJ estimators can accept inaccurate initial conditions, provided that starting NMS pole values are relatively close to the real ones. Hence, it is true that BJ techniques are dependent on ICs, but these estimators are flexible with the initial values of HO zeros.

In future works, a decimation analysis should be conducted in order to study the effect of additional model parameters to be estimated due to the Padé approximation. The BJ algorithm performance does not have to necessarily diminish in this case, since as observed for high remnant orders, the bias was still acceptable in spite of the increase in model remnant parameters.

In addition, BJ estimators could be initialised with the model parameters obtained from a prior ARX computation (OLS), above all when HO remnant orders are higher than one. This strategy could be quite interesting given the sensitivity of BJ models to inaccurate initial conditions.

In the case of a recursive identification, BJ algorithms might face some instabilities due to unacceptable discrete-time parameter estimations (the conditions $a_1^d < 0$ and $a_2^d > 0$ must be fulfilled), and could find difficulties in achieving an optimal estimation of the HO time delay cause of the poor information provided by VAF. It is crucial to explore different modifications of the main recursive BJ algorithm to avoid these possible issues in real-life identification tasks.

[1]The human operator needs to generate lead at a lower frequency in C2, so the effect of a wrong zero identification is more significative in HO magnitudes at higher frequencies.

# References


[1] A. Ameyoe et al. "Identification of a Linear Parameter Varying Driver Model for the Detection of Distraction". In: *IFAC-PapersOnLine* 48.26 (2015). 1st IFAC Workshop on Linear Parameter Varying Systems LPVS 2015, pp. 37–42. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2015.11.110.

[2] E.R. Boer and R.V. Kenyon. "Estimation of time-varying delay time in nonstationary linear systems: an approach to monitor human operator adaptation in manual tracking tasks". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28.1 (1998), pp. 89–99. DOI: 10.1109/3468.650325.

[3] G. E. P. Box and G. M. Jenkins. *Time series analysis forecasting and control*. 5th ed. John Wiley & Sons, Inc., 1970.

[4] John C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, Ltd, 2003, pp. 55–142. DOI: 10.1002/9781119121534.

[5] Frank M. Drop et al. "Constraints in Identification of Multi-Loop Feedforward Human Control Models". In: *IFAC-PapersOnLine* 49.19 (2016). 13th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems HMS 2016, pp. 7–12. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.10.444.

[6] R.F.M. Duarte et al. "Experimental Scheduling Functions for Global LPV Human Controller Modeling". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 15853–15858. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2017.08.2329.

[7] Gene F. Franklin, J.D. Powell, and Michael L. Workman. *Digital Control Of Dynamic Systems 3rd Edition*. Ellis-Kagle Press, 1998, pp. 146–164.

[8] Z. Gajic. *Linear Dynamic Systems and Signals*. Prentice Hall, 2003, pp. 407–418.

[9] A. van Grootheest. "Human-Operator Identification with Time-Varying ARX Models". 2017. URL: http://resolver.tudelft.nl/uuid:da69d1cf-3274-466f-bbc2-573f571d154e.

[10] A. van Grootheest et al. "Identification of Time-Varying Manual-Control Adaptations with Recursive ARX Models". In: *AIAA* (2018). DOI: 10.2514/6.2018-0118.

[11] Ronald A. Hess. "Modeling Human Pilot Adaptation to Flight Control Anomalies and Changing Task Demands". In: *Journal of Guidance Control and Dynamics* (2016). DOI: 10.2514/1.g001303.

[12] Ronald A. Hess. "Modeling Pilot Control Behavior with Sudden Changes in Vehicle Dynamics". In: *Journal of Aircraft* (2009). DOI: 10.2514/1.41215.

[13] Syed Aseem Ul Islam and Dennis S. Bernstein. "Recursive Least Squares for Real-Time Implementation [Lecture Notes]". In: *IEEE Control Systems Magazine* 39.3 (2019), pp. 82–85. DOI: 10.1109/MCS.2019.2900788.

[14] J. Jiao et al. "Identifying Pilot Control Adaptations to Sudden Changes in Aircraft Dynamics". In: *Journal of Guidance Control and Dynamics* (2023). DOI: 10.2514/1.g007358.

[15] David H. Klyde, Martin J. Brenner, and P. Thompson. "Wavelet-based time-varying human operator models." In: *AIAA* (2001). DOI: 10.2514/6.2001-4009.

[16] William H. Levison, S. Baron, and David L. Kleinman. "A Model for Human Controller Remnant". In: *IEEE Transactions on Man-Machine Systems* 10.4 (1969), pp. 101–108. DOI: 10.1109/TMMS.1969.299906.

[17] M. Linssen. "Identifying Time-Varying Multimodal Manual Control Using Recursive ARX Model Techniques". 2020. URL: http://resolver.tudelft.nl/uuid:442f4308-0ea2-41a5-b38c-ee6b1a289f78.

[18] L. Ljung. *System Identification Toolbox: User's Guide*. Mathworks, 2023.

[19] L. Ljung. *System Identification: Theory for the User, 2nd Edition*. Ed. by Thomas Kailath. Prentice Hall Information and System Sciences Series, 2012.

[20] M. Lone and A. Cooke. "Review of pilot models used in aircraft flight dynamics". In: *Aerospace Science and Technology* 34 (2014), pp. 55–74. ISSN: 1270-9638. DOI: 10.1016/j.ast.2014.02.003.

[21] T. Mandal and Y. Gu. "Online Pilot Model Parameter Estimation Using Sub-Scale Aircraft Flight Data". In: *AIAA* (2016). DOI: 10.2514/6.2016-0636.

[22] Duane T. McRuer. "Human Pilot Dynamics in Compensatory Systems". In: *Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, OH* (1965). DOI: 10.21236/ad0470337.

[23] Duane T. McRuer, Dunstan Graham, and Ezra S. Krendel. "Manual control of single-loop systems: Part I". In: *Journal of the Franklin Institute* 283.1 (1967), pp. 1–29. ISSN: 0016-0032. DOI: 10.1016/0016-0032(67)90112-3.

[24] Duane T. McRuer, Dunstan Graham, and Ezra S. Krendel. "Manual control of single-loop systems: Part II". In: *Journal of the Franklin Institute* 283.2 (1967), pp. 145–168. ISSN: 0016-0032. DOI: 10.1016/0016-0032(67)90231-1.

[25] Duane T. McRuer and H.R. Jex. "A Review of Quasi-Linear Pilot Models". In: *IEEE Transactions on Human Factors in Electronics* HFE-8.3 (1967), pp. 231–249. DOI: 10.1109/THFE.1967.234304.

[26] Duane T. McRuer and D. H. Weir. "Theory of Manual Vehicular Control". In: *IEEE Transactions on Man-Machine Systems* 10.4 (1969), pp. 257–291. DOI: 10.1109/TMMS.1969.299930.

[27] L. D. Metz. "A Time-Varying Approach to the Modeling of Human Control Remnant". In: *IEEE Transactions on Systems, Man, and Cybernetics* 12.1 (1982), pp. 24–35. DOI: 10.1109/TSMC.1982.4308772.

[28] M. Mulder et al. "Fundamental Issues in Manual Control Cybernetics". In: *IFAC-PapersOnLine* 49.19 (2016). 13th IFAC Symposium on Analysis, Design, and Evaluation ofHuman-Machine Systems HMS 2016, pp. 1–6. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.10.429.

[29] M. Mulder et al. "Manual Control Cybernetics: State-of-the-Art and Current Trends". In: *IEEE Transactions on Human-Machine Systems* 48.5 (2018), pp. 468–485. DOI: 10.1109/THMS.2017.2761342.

[30] Frank M. Nieuwenhuizen et al. "Modeling Human Multichannel Perception and Control Using Linear Time-Invariant Models". In: *Journal of Guidance Control and Dynamics* (2008). DOI: 10.2514/1.32307.

[31] K. Ogata. *Discrete-time control systems, 2nd Edition*. Prentice Hall International, 1995, pp. 293–377.

[32] M. Olivari et al. "Identifying time-varying neuromuscular system with a recursive least-squares algorithm: a Monte-Carlo simulation study". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2014, pp. 3573–3578. DOI: 10.1109/SMC.2014.6974484.

[33] M. Olivari et al. "Identifying Time-Varying Pilot Responses: A Regularized Recursive Least-Squares Algorithm". In: *AIAA* (2016). DOI: 10.2514/6.2016-1182.

[34] M.M. van Paassen and M. Mulder. "Identification of Human Operator Control Behaviour in Multiple-Loop Tracking Tasks". In: *IFAC Proceedings Volumes* 31.26 (1998). 7th IFAC Symposium on Analysis, Design and Evaluation of Man-Machine Systems (MMS'98), Kyoto, Japan, 16-18 September 1998, pp. 455–460. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)40135-2.

[35] R. Pintelon, J. Schoukens, and P. Guillaume. "Continuous-Time Noise Modeling From Sampled Data". In: *IEEE Transactions on Instrumentation and Measurement* 55.6 (2006), pp. 2253–2258. DOI: 10.1109/TIM.2006.884131.

[36] W. Plaetinck et al. "Online Identification of Pilot Adaptation to Sudden Degradations in Vehicle Stability". In: *IFAC-PapersOnLine* 51.34 (2019). 2nd IFAC Conference on Cyber-Physical and Human Systems CPHS 2018, pp. 347–352. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.01.020.

[37] Daan M. Pool et al. "Identification of Nonlinear Motion Perception Dynamics Using Time-Domain Pilot Modeling". In: *Journal of Guidance Control and Dynamics* (2012). DOI: 10.2514/1.56236.

[38] A. Popovici, Peter M. T. Zaal, and Daan M. Pool. "Dual Extended Kalman Filter for the Identification of Time-Varying Human Manual Control Behavior". In: *AIAA* (2017). DOI: 10.2514/6.2017-3666.

[39] Mark R. *What are the Different Types of Pilot Licenses?* 2021. URL: https://www.flyingmag.com/types-of-pilot-licenses/ (visited on 11/06/2023).

[40] R.L. Stapleford, D.T. McRuer, and R.E. Magdaleno. "Pilot Describing Function Measurements in a Multiloop Task". In: *IEEE Transactions on Human Factors in Electronics* HFE-8.2 (1967), pp. 113–125. DOI: 10.1109/THFE.1967.233628.

[41] Arun K. Tangirala. *Principles of System Identification: Theory and Practice*. Taylor & Francis Group, 2017, pp. 129–147. DOI: 10.1201/9781315222509.

[42] M. Vajta. "Some remarks on Padé-approximations". English. In: 3rd TEMPUS-INTCOM Symposium on Intelligent Systems in Control and Measurements 2000 ; Conference date: 09-09-2000 Through 14-09-2000. 2000, pp. 53–58.

[43] K. van der El, Daan M. Pool, and M. Mulder. "Analysis of Human Remnant in Pursuit and Preview Tracking Tasks". In: *IFAC-PapersOnLine* 52.19 (2019). 14th IFAC Symposium on Analysis, Design, and Evaluation of Human Machine Systems HMS 2019, pp. 145–150. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.12.165.

[44] J.W. van Wingerden and M. Verhaegen. "Subspace identification of Bilinear and LPV systems for open- and closed-loop data". In: *Automatica* 45.2 (2009), pp. 372–381. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2008.08.015.

[45] L. R. Young. "On Adaptive Manual Control". In: *Ergonomics* 12.4 (1969), pp. 635–674. DOI: 10.1080/00140136908931083.

[46] Peter C. Young. *Recursive Estimation and Time-Series Analysis, 2nd Edition*. Springer, 2011, pp. 289–315. DOI: 10.1007/978-3-642-21981-8.

[47] Peter C. Young. "Refined instrumental variable estimation: Maximum likelihood optimization of a unified Box–Jenkins model". In: *Automatica* 52 (2015), pp. 35–46. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2014.10.126.

[48] Peter M. T. Zaal and Barbara T. Sweet. "Estimation of Time-Varying Pilot Model Parameters". In: *AIAA* (2011). DOI: 10.2514/6.2011-6474.

[49] Peter M. T. Zaal et al. "Modeling Human Multimodal Perception and Control Using Genetic Maximum Likelihood Estimation". In: *Journal of Guidance Control and Dynamics* (2009). DOI: 10.2514/1.42843.

[50] Peter M.T. Zaal. "Manual Control Adaptation to Changing Vehicle Dynamics in Roll-Pitch Control Tasks". In: *Journal of Guidance Control and Dynamics* (2016). DOI: 10.2514/1.g001592.

# A

# Remnant gain definition

## A.1. Theoretical background

In order to obtain a valid formula for the remnant filter gain $K_n$ in the function of the requested noise level, it is necessary to first define the covariance function for two null-mean stochastic variables (SV) $\{\bar{x}(t), \bar{y}(t)\}$,

$$C_{\bar{x}\bar{y}}(\tau) = E\left\{\bar{x}(t)\bar{y}(t+\tau)\right\} = \lim_{T\to\infty}\left(\frac{1}{2T}\int_{-T}^{T}\bar{x}(t)\bar{y}(t+\tau)\mathrm{d}t\right), \tag{A.1}$$

from which the Power Spectral Density (PSD) function can be obtained as its Fourier transform:

$$S_{\bar{x}\bar{y}}(\omega) = \mathcal{F}\left\{C_{\bar{x}\bar{y}}(\tau)\right\} = \int_{-\infty}^{\infty} C_{\bar{x}\bar{y}}(\tau)e^{-j\omega\tau}\mathrm{d}\tau. \tag{A.2}$$

For two different stochastic processes, $\{\bar{x}(t), \bar{y}(t)\}$, the Power Spectral Density function is called cross-PSD, while for the same stochastic process, $\bar{x}(t)$, it is named as auto-PSD:

$$S_{\bar{x}\bar{y}}(\omega) = \lim_{T\to\infty}\left(\frac{1}{2T}\bar{Y}(\omega)\bar{X}(-\omega)\right), \qquad S_{\bar{x}\bar{x}}(\omega) = \lim_{T\to\infty}\left(\frac{1}{2T}\bar{X}(\omega)\bar{X}(-\omega)\right) = \lim_{T\to\infty}\left(\frac{1}{2T}\mid\bar{X}(\omega)\mid^2\right). \tag{A.3}$$

Furthermore, the variance of a SV $\bar{x}(t)$ can be computed by integrating its auto-PSD function:

$$\sigma^2_{\bar{x}\bar{x}} = C_{\bar{x}\bar{x}}(\tau = 0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\bar{x}\bar{x}}(\omega)\mathrm{d}\omega. \tag{A.4}$$

From the diagram in Figure 2.2, the control-output signal is defined by $\bar{u}(t) = \bar{u}_e(t) + \bar{n}(t)$. The forcing function $\bar{f}_t(t)$ and unfiltered remnant noise $n(t)$ signals are assumed to be uncorrelated (i.e., $S_{\bar{\varepsilon}\bar{f}_t}(\omega) = S_{\bar{f}_t\bar{\varepsilon}}(\omega) = 0$). Hence, applying Fourier transforms and arranging the expression to obtain the relation between auto-PSD functions:

$$\begin{aligned}
&\mathcal{F}\left\{\bar{u}(t)\right\} = \mathcal{F}\left\{\bar{u}_e(t)\right\} + \mathcal{F}\left\{\bar{n}(t)\right\} \;\rightarrow\; \bar{U}(\omega) = \bar{U}_e(\omega) + \bar{N}(\omega) \;\rightarrow\; \bar{U}(\omega)\bar{U}(-\omega) = (\bar{U}_e(\omega) + \bar{N}(\omega))\cdot\\
&\cdot(\bar{U}_e(-\omega) + \bar{N}(-\omega)) \;\rightarrow\; \bar{U}(\omega)\bar{U}(-\omega) = \left(\frac{H_{HO}(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\bar{F}(\omega) + \frac{H_n(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\bar{\varepsilon}(\omega)\right)\cdot\\
&\cdot\left(\frac{H_{HO}(-\omega)}{1+H_{HO}(-\omega)H_{CE}(-\omega)}\bar{F}(-\omega) + \frac{H_n(-\omega)}{1+H_{HO}(-\omega)H_{CE}(-\omega)}\bar{\varepsilon}(-\omega)\right) \;\rightarrow\\
&\rightarrow\; S_{\bar{u}\bar{u}}(\omega) = \left|\frac{H_n(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega) + \left|\frac{H_{HO}(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{f}_t\bar{f}_t}(\omega) = S_{\bar{u}_n\bar{u}_n}(\omega) + S_{\bar{u}_f\bar{u}_f}(\omega).
\end{aligned} \tag{A.5}$$

In the final expression, there are two terms that contribute to the auto-PSD function of the control-output signal, the first one ($S_{\bar{u}_n\bar{u}_n}(\omega)$), which corresponds to the unfiltered remnant noise, and the second one ($S_{\bar{u}_f\bar{u}_f}(\omega)$), owing to the forcing function. When integrating the equation, the corresponding variances for each component are obtained:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\bar{u}\bar{u}}(\omega)\mathrm{d}\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\bar{u}_n\bar{u}_n}(\omega)\mathrm{d}\omega + \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\bar{u}_f\bar{u}_f}(\omega)\mathrm{d}\omega \;\rightarrow\; \sigma^2_{\bar{u}\bar{u}} = \sigma^2_{\bar{u}_n\bar{u}_n} + \sigma^2_{\bar{u}_f\bar{u}_f}. \tag{A.6}$$

## A.2. PSD function of the forcing function

To compute an expression of $S_{\bar{f}_t\bar{f}_t}(\omega)$, the covariance function of the forcing function is obtained before:

$$C_{\bar{f}_t\bar{f}_t}(\tau) = E\left\{\bar{f}_t(t)\bar{f}_t(t+\tau)\right\} = \sum_{k=1}^{N_t} \lim_{T\to\infty}\left(\frac{A_k^2}{2T}\int_{-T}^{T}\sin\big(\omega_k t+\phi_k\big)\sin\big(\omega_k t+\omega_k\tau+\phi_k\big)\mathrm{d}t\right). \tag{A.7}$$

The integrand can be decomposed in a sum of products of sines and cosines:

$$\sin\big(\omega_k t+\phi_k\big)\sin\big(\omega_k t+\omega_k\tau+\phi_k\big) = \frac{1}{2}\sin(2\omega_k t)\sin\big(2\phi_k+\omega_k\tau\big)+\sin\big(\phi_k\big)\sin\big(\phi_k+\omega_k\tau\big)+\sin^2(\omega_k t)\cos\big(2\phi_k+\omega_k\tau\big). \tag{A.8}$$

Thus, when integrating on time-domain, applying the limits, and rearranging the trigonometrical expression:

$$\begin{aligned}
C_{\bar{f}_t\bar{f}_t}(\tau) &= \sum_{k=1}^{N_t}\left(\lim_{T\to\infty}\frac{A_k^2}{2T}\int_{-T}^{T}\frac{1}{2}\sin(2\omega_k t)\sin\big(2\phi_k+\omega_k\tau\big)\mathrm{d}t + \lim_{T\to\infty}\frac{A_k^2}{2T}\int_{-T}^{T}\sin\big(\phi_k\big)\sin\big(\phi_k+\omega_k\tau\big)\mathrm{d}t+\right.\\
&\left.+\lim_{T\to\infty}\frac{A_k^2}{2T}\int_{-T}^{T}\sin^2(\omega_k t)\cos\big(2\phi_k+\omega_k\tau\big)\mathrm{d}t\right) = \sum_{k=1}^{N_t}A_k^2\left(\sin\big(\phi_k\big)\sin\big(\phi_k+\omega_k\tau\big)+\frac{1}{2}\cos\big(2\phi_k+\omega_k\tau\big)\right) =\\
&\sum_{k=1}^{N_t}A_k^2\left(\sin\big(\phi_k\big)\sin\big(\phi_k+\omega_k\tau\big)+\frac{1}{2}\cos\big(2\phi_k+\omega_k\tau\big)\right) = \sum_{k=1}^{N_t}A_k^2\left(\sin(\omega_k\tau)\left(\sin\big(\phi_k\big)\cos\big(\phi_k\big)-\frac{1}{2}\sin\big(2\phi_k\big)\right)+\right.\\
&\left.+\cos(\omega_k\tau)\left(\sin^2(\phi_k)+\frac{1}{2}\cos\big(2\phi_k\big)\right)\right) = \sum_{k=1}^{N_t}A_k^2\left(0+\cos(\omega_k\tau)\left(\sin^2(\phi_k)+\cos^2(\phi_k)-\frac{1}{2}\right)\right) = \sum_{k=1}^{N_t}\frac{A_k^2\cos(\omega_k\tau)}{2}.
\end{aligned} \tag{A.9}$$

Then, by integrating the covariance function, based on Equation A.2, the expression of $S_{\bar{f}_t\bar{f}_t}(\omega)$:

$$S_{\bar{f}_t\bar{f}_t}(\omega) = \mathcal{F}\left\{C_{\bar{f}_t\bar{f}_t}(\tau)\right\} = \sum_{k=1}^{N_t}\frac{A_k^2}{2}\mathcal{F}\left\{\cos(\omega_k\tau)\right\} = \frac{\pi}{2}\sum_{k=1}^{N_t}A_k^2\left(\delta(\omega-\omega_k)+\delta(\omega+\omega_k)\right). \tag{A.10}$$

## A.3. Remnant gain definition

Based on the literature, the noise level is usually defined as $P_n = \sigma^2_{\bar{u}_n\bar{u}_n}/\sigma^2_{\bar{u}\bar{u}}$ (Van der El. [43]), or $P_n = \sigma^2_{\bar{n}\bar{n}}/\sigma^2_{\bar{u}\bar{u}}$ (Van Grootheest [9]).

### A.3.1. First option

By the definition of $P_n = \sigma^2_{\bar{u}_n\bar{u}_n}/\sigma^2_{\bar{u}\bar{u}}$, all noise level values which belong to the interval $[0,1)$ can be achieved. Hence, when the remnant gain tends to infinity, it will provide a noise level value that converges to 1. From the final expression in Equation A.5:

$$P_n = \frac{\sigma^2_{\bar{u}_n\bar{u}_n}}{\sigma^2_{\bar{u}\bar{u}}} = \frac{\frac{1}{2\pi}\int_{-\infty}^{\infty}S_{\bar{u}_n\bar{u}_n}(\omega)\mathrm{d}\omega}{\frac{1}{2\pi}\int_{-\infty}^{\infty}S_{\bar{u}\bar{u}}(\omega)\mathrm{d}\omega} = \frac{\frac{1}{2\pi}\int_{-\infty}^{\infty}\left|\frac{H_n(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)\mathrm{d}\omega}{\frac{1}{2\pi}\int_{-\infty}^{\infty}\left(\left|\frac{H_n(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)+\left|\frac{H_{HO}(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{f}_t\bar{f}_t}(\omega)\right)\mathrm{d}\omega}. \tag{A.11}$$

Then, an expression of the remnant gain is obtained:

$$K_n = \sqrt{\frac{P_n}{1-P_n}\cdot\frac{\int_0^{\infty}\left|\frac{H_{HO}(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{f}_t\bar{f}_t}(\omega)\mathrm{d}\omega}{\int_0^{\infty}\frac{S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)}{\left|(T_n(j\omega)+1)^m(1+H_{HO}(\omega)H_{CE}(\omega))\right|^2}\mathrm{d}\omega}}. \tag{A.12}$$

The PSD function $S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)$ must be defined before obtaining the final expression of $K_n$. When a White Gaussian noise signal $\bar{\varepsilon}(t)$ is created in discrete-time, a Band-Limited noise is actually generated in the frequency domain [35], which is constrained by the sampling frequency $\omega_s$. This implies that if the variance of $\bar{\varepsilon}(t)$ is computed by the Equation A.4, the final result would be $\sigma^2_{\bar{\varepsilon}\bar{\varepsilon}}/T_s$, since the PSD function is only defined in the interval $I = [-\omega_s/2, \omega_s/2]$:

$$S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega) = \begin{cases}\sigma^2_{\bar{\varepsilon}\bar{\varepsilon}} & |\ \omega\ | \leq \frac{\pi}{T_s}\\ 0 & |\ \omega\ | > \frac{\pi}{T_s}\end{cases} \quad\rightarrow\quad \frac{1}{2\pi}\int_{-\infty}^{\infty}S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)\mathrm{d}\omega = \frac{\sigma^2_{\bar{\varepsilon}\bar{\varepsilon}}}{T_s}. \tag{A.13}$$

Therefore, the terms affected by $S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)$ must be multiplied by $T_s$ in order to scale them properly, so the real analytical variance is achieved. Moreover, the integration limits of such terms will be modified as well, since the PSD function is zero for frequency values out of the interval $I = [-\pi/T_s, \pi/T_s]$.

Finally, by means of the $S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)$ scaling and PSD function of $\bar{f}_t(t)$ (Eq. A.10), the final formula of the first-option remnant gain is as follows:

$$K_n = \sqrt{\frac{P_n}{(1-P_n)\sigma^2_{\bar{\varepsilon}\bar{\varepsilon}} \cdot T_s} \frac{\frac{\pi}{2}\sum_{k=1}^{N_t} A_k^2 \left|\frac{H_{HO}(j\omega_k)}{1+H_{HO}(j\omega_k)H_{CE}(j\omega_k)}\right|^2}{\int_0^{\pi/T_s} \frac{1}{\left|(T_n(j\omega)+1)^m(1+H_{HO}(j\omega)H_{CE}(j\omega))\right|^2}\mathrm{d}\omega}}. \tag{A.14}$$

### A.3.2. Second option

In the case of $P_n = \sigma^2_{\bar{n}\bar{n}}/\sigma^2_{\bar{u}\bar{u}}$, not all values of noise level can be obtained since some of them are physically impossible due to the dependency of $\bar{u}(t)$ on $\bar{n}(t)$. Thus, when the remnant gain tends to infinity, it will provide the maximum possible noise level. From the final expression in Equation A.5:

$$P_n = \frac{\sigma^2_{\bar{n}\bar{n}}}{\sigma^2_{\bar{u}\bar{u}}} = \frac{\frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\bar{n}\bar{n}}(\omega)\mathrm{d}\omega}{\frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\bar{u}\bar{u}}(\omega)\mathrm{d}\omega} = \frac{\frac{1}{2\pi}\int_{-\infty}^{\infty} |H_n(\omega)|^2 S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)\mathrm{d}\omega}{\frac{1}{2\pi}\int_{-\infty}^{\infty}\left(\left|\frac{H_n(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega) + \left|\frac{H_{HO}(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{f}_t\bar{f}_t}(\omega)\right)\mathrm{d}\omega}. \tag{A.15}$$

Then, an expression of the remnant gain is obtained:

$$K_n = \sqrt{\frac{P_n \cdot \int_0^{\infty}\left|\frac{H_{HO}(\omega)}{1+H_{HO}(\omega)H_{CE}(\omega)}\right|^2 S_{\bar{f}_t\bar{f}_t}(\omega)\mathrm{d}\omega}{\int_0^{\infty}\frac{S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)}{\left|(T_n(j\omega)+1)^m\right|^2}\mathrm{d}\omega - P_n \cdot \int_0^{\infty}\frac{S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)}{\left|(T_n(j\omega)+1)^m(1+H_{HO}(\omega)H_{CE}(\omega))\right|^2}\mathrm{d}\omega}}. \tag{A.16}$$

Finally, by means of the definitions for the Band-limited White Gaussian noise (Equation A.13) and PSD function of $\bar{f}_t(t)$ (Eq. A.10), the final formula of the second-option remnant gain is as follows:

$$K_n = \sqrt{\frac{P_n}{\sigma^2_{\bar{\varepsilon}\bar{\varepsilon}} \cdot T_s} \frac{\frac{\pi}{2}\sum_{k=1}^{N_t} A_k^2 \left|\frac{H_{HO}(j\omega_k)}{1+H_{HO}(j\omega_k)H_{CE}(j\omega_k)}\right|^2}{\int_0^{\pi/T_s}\frac{1}{\left|(T_n(j\omega)+1)^m\right|^2}\mathrm{d}\omega - P_n \cdot \int_0^{\pi/T_s}\frac{1}{\left|(T_n(j\omega)+1)^m(1+H_{HO}(j\omega)H_{CE}(j\omega))\right|^2}\mathrm{d}\omega}}. \tag{A.17}$$

## A.4. Validation of the remnant gain formula

The PSD function $S_{\bar{\varepsilon}\bar{\varepsilon}}(\omega)$ is not homogeneous at every realization, since it provides different results at each frequency value in the range $I = [-\omega_s/2, \omega_s/2]$, which interfere with the spectrum of the control-output $\bar{u}(t)$. Only its integration on the frequency domain gives the same output. Hence, obtained noise levels will vary for each realization, although these ones will present only small deviations with respect to the requested $P_n$.

Consequently, the average and variance of the computed noise levels must be addressed in order to verify whether the remnant gain formula is valid and offers consistent results. There are two possible methods to estimate the calculated noise level depending on its definition:

- First option ($P_n = \sigma^2_{\bar{u}_n\bar{u}_n}/\sigma^2_{\bar{u}\bar{u}}$): since the variance component $\sigma^2_{\bar{u}_n\bar{u}_n}$ is invariant for each realization and requested noise level, its value can be computed from a null-$P_n$ realization as $\sigma^2_{\bar{u}_f\bar{u}_f} = \sigma^2_{\bar{u}\bar{u}}(P_n = 0)$. Thus, the resultant noise level can be calculated as $P_n = \sigma^2_{\bar{u}_n\bar{u}_n}/\sigma^2_{\bar{u}\bar{u}} = \left(\sigma^2_{\bar{u}\bar{u}} - \sigma^2_{\bar{u}\bar{u}}(P_n = 0)\right)/\sigma^2_{\bar{u}\bar{u}}$.
- Second option ($P_n = \sigma^2_{\bar{n}\bar{n}}/\sigma^2_{\bar{u}\bar{u}}$): in this case, the noise level can be easily computed, since both variances can be calculated after storing the filtered White Gaussian noise and the control-output signals.

In Figure A.1, the obtained noise levels are shown for simulation condition C1. The variability of the resultant $P_n$ is reduced for the entire domain, while the average obtained noise level only presents small deviations with respect to the demanded ones. Thus, the accuracy of the remnant gain formula is acceptable.

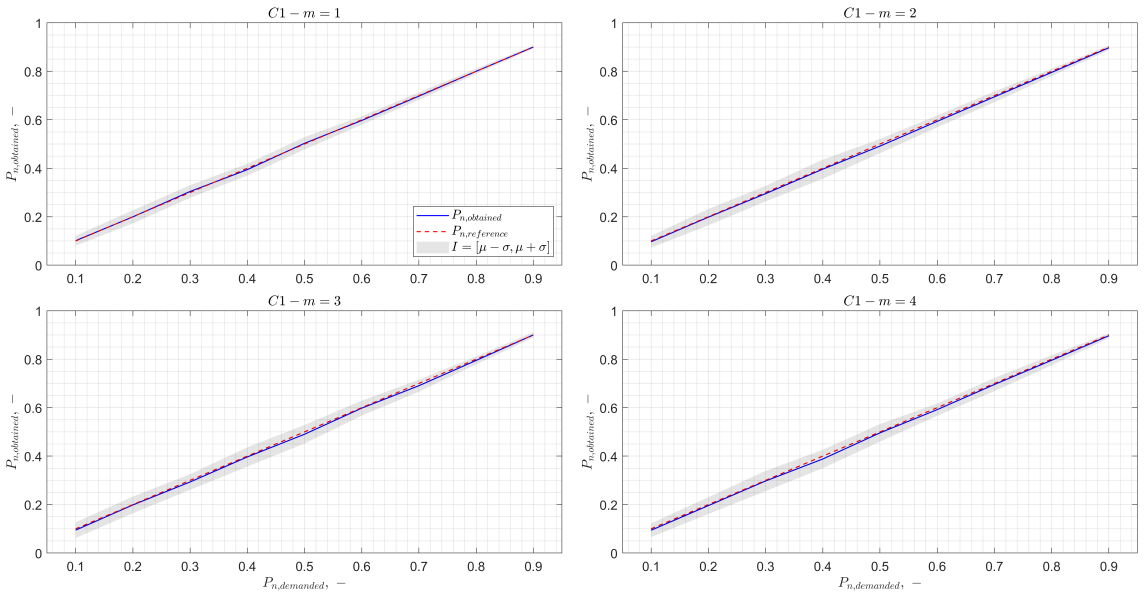


**Figure A.1:** Obtained noise levels for simulation condition C1 and remnant filter order $m^0 \in \{1, 2, 3, 4\}$, averaged for $M = 100$ realizations: demanded $P_n$ (red), obtained $\mu_{P_n}$ (blue), confidence interval $I = [\mu_{P_n} - \sigma_{P_n}, \mu_{P_n} + \sigma_{P_n}]$ (shaded area).

# B

# Continuous-time parameter retrieval

## B.1. Human operator

From the cybernetics theory, the following human operator model is employed:

$$H_{HO_e}(j\omega) = K_e\left[T_L j\omega + 1\right]\frac{\omega_{nm}^2}{(j\omega)^2 + 2\zeta_{nm}\omega_{nm}j\omega + \omega_{nm}^2}e^{-s\tau_e}. \tag{B.1}$$

Without taking into account the time delay $\tau_e$, the resultant model is:

$$H(j\omega) = \frac{\omega_{nm}^2 K_e T_L j\omega + \omega_{nm}^2 K_e}{(j\omega)^2 + 2\zeta_{nm}\omega_{nm}j\omega + \omega_{nm}^2}. \tag{B.2}$$

In order to obtain the discrete-time transfer function of the human operator model, a zero-order hold (ZOH) discretization is applied based on the method proposed by Tangirala [41]:

$$H_d(z) = (1 - z^{-1})\mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{H_c(s)}{s}\right\}\Big|_{t=kT_s}\right\}. \tag{B.3}$$

In this way, the obtained HO transfer function is defined by the discrete-time variable $z$, in ascending powers of $z^{-1}$:

$$H(z) = z^{-1}\frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}}. \tag{B.4}$$

As can be observed, a unit-sample input-output delay appears due to discretization. Finally, the term corresponding to the time delay is introduced in the previous expression:

$$H_{HO_{e,dis}}(z) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}} \cdot z^{-n_k}, \tag{B.5}$$

where $n_k$ is an integer defined as $n_k = int(\tau_e/T_s) + 1$. This number must always be an integer, so that a Padé approximation [42] has to be used when this is no longer possible. Nevertheless, this is not needed for this project since no decimation process has been conducted and the chosen measurement time and time delay have been properly selected in order that $int(\tau_e/T_s)$ is an integer.

Once the discrete-time HO parameters are estimated through ARX/BJ structures, the resultant transfer function is converted into a state-space system in controllable canonical form [31]. The state-space matrices without time delay defined by the HO parameters are:

$$A_d = \begin{bmatrix} 0 & 1 \\ -a_2^d & -a_1^d \end{bmatrix}, \quad B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C_d = \begin{bmatrix} b_1^d & b_0^d \end{bmatrix}, \quad D_d = \begin{bmatrix} 0 \end{bmatrix}. \tag{B.6}$$

After, the discrete-time state-space system is converted into a continuous-time one as explained by Gajic [8]. The new states-space system is obtained from the logarithm of the extended matrix, while $C_c$ is equivalent to $C_d$:

$$\left[\begin{array}{c|c} A_c & B_c \\ \hline 0 & 0 \end{array}\right] = \frac{1}{T_s}\ln\left(\left[\begin{array}{c|c} A_d & B_d \\ \hline 0 & I \end{array}\right]\right), \quad C_c = C_d = \begin{bmatrix} b_1^d & b_0^d \end{bmatrix}. \tag{B.7}$$

The realization obtained for this continuous-time state-space system may not be in the controllable canonical form, so the matrices are defined as

$$A_c = \begin{bmatrix} a'^c_{11} & a'^c_{12} \\ a'^c_{21} & a'^c_{22} \end{bmatrix}, \quad B_c = \begin{bmatrix} b'^c_{11} \\ b'^c_{21} \end{bmatrix}, \quad C_c = \begin{bmatrix} c'^c_{11} & a'^c_{12} \end{bmatrix}, \quad D_c = \begin{bmatrix} 0 \end{bmatrix}, \tag{B.8}$$

and then, the estimated continuous-time transfer function is computed as

$$\hat{H}(s) = C_c(sI - A_c)^{-1}B_c = \frac{(b'^c_{11}c'^c_{11} + b'^c_{21}c'^c_{12})s + (a'^c_{21}b'^c_{11}c'^c_{12} + a'^c_{12}b'^c_{21}c'^c_{11} - a'^c_{22}b'^c_{11}c'^c_{11} - a'^c_{11}b'^c_{21}c'^c_{12})}{s^2 + (-a'^c_{11} - a'^c_{22})s + (a'^c_{11}a'^c_{22} - a'^c_{12}a'^c_{21})}. \tag{B.9}$$

From this last transfer function, the following continuous-time parameters can be extracted:

$$b^c_0 = b'^c_{11}c'^c_{11} + b'^c_{21}c'^c_{12}, \qquad b^c_1 = a'^c_{21}b'^c_{11}c'^c_{12} + a'^c_{12}b'^c_{21}c'^c_{11} - a'^c_{22}b'^c_{11}c'^c_{11} - a'^c_{11}b'^c_{21}c'^c_{12} \tag{B.10a}$$

$$a^c_1 = -a'^c_{11} - a'^c_{22}, \qquad a^c_2 = a'^c_{11}a'^c_{22} - a'^c_{12}a'^c_{21}. \tag{B.10b}$$

From these estimated coefficients, the final HO parameters can be identified by means of Equation 5.6.

## B.2. Remnant filter and noise

By the Backward Euler discretization, $s = (1 - z^{-1})/T_s$, a proper discrete-time transfer function for the remnant filter is achieved:

$$H^m_n(s) = \frac{K_n}{(T_n s + 1)^m} \quad \rightarrow \quad H^m_{n,dis}(z) = \frac{K_n}{\left(T_n \frac{1-z^{-1}}{T_s} + 1\right)^m} = \frac{K_n \left(\frac{T_s}{T_n + T_s}\right)^m}{\left(1 - \frac{T_n}{T_n + T_s} z^{-1}\right)^m}. \tag{B.11}$$

Since the numerator of the estimated discrete-time remnant filter must be equal to 1, the resultant coefficient in the numerator of $H^m_{n,dis}(z)$ will be incorporated into the unitary variance $\sigma^2_\varepsilon$ of the White Gaussian noise $\varepsilon$, giving as a result, a modified $\varepsilon'$ with standard deviation $\sigma_{\varepsilon'}$:

$$\sigma_{\varepsilon'} = \frac{K_n T_s^m}{(T_n + T_s)^m} \sigma_\varepsilon. \tag{B.12}$$

Hence, the discrete-time model of the remnant noise signal would be the following one:

$$n(t_k) = \frac{1}{\left(1 - \frac{T_n}{T_n + T_s} z^{-1}\right)^m} \varepsilon'(t_k), \qquad \varepsilon'(t_k) \sim N(0, \sigma_{\varepsilon'}). \tag{B.13}$$

Depending on the $mth$-order of the remnant filter, the number of discrete-time parameters, $d^d_i$, to be estimated changes, while the time constant $T_n$ would need to be averaged from such parameters computed:

$$d^d_i = f_i(T_s, T_{n,d^d_i}) \quad \rightarrow \quad T_{n,d^d_i} \quad \rightarrow \quad \hat{T}_n = \frac{1}{m} \sum_{i=1}^{m} T_{n,d^d_i}. \tag{B.14}$$

Therefore, these are the possible combinations:

- Remnant order $m = 1$: The remnant filter $H^1_{n,dis}(z) = \frac{K_n \left(\frac{T_s}{T_n + T_s}\right)}{\left(1 - \frac{T_n}{T_n + T_s} z^{-1}\right)}$ can be estimated by the discrete-time transfer function $H_d(z) = \frac{1}{1 + d^d_1 z^{-1}}$. The time constant is computed as $\hat{T}_n = T_{n,d^d_1}$.
- Remnant order $m = 2$: The remnant filter $H^2_{n,dis}(z) = \frac{K_n \left(\frac{T_s}{T_n + T_s}\right)^2}{1 - \frac{2T_n}{T_n + T_s} z^{-1} + \frac{T_n^2}{(T_n + T_s)^2} z^{-2}}$ can be estimated by the discrete-time transfer function $H_d(z) = \frac{1}{1 + d^d_1 z^{-1} + d^d_2 z^{-2}}$. The time constant is computed as $\hat{T}_n = \frac{1}{2}\left(T_{n,d^d_1} + T_{n,d^d_2}\right)$.
- Remnant order $m = 3$: The remnant filter $H^3_{n,dis}(z) = \frac{K_n \left(\frac{T_s}{T_n + T_s}\right)^3}{1 - \frac{3T_n}{T_n + T_s} z^{-1} + \frac{3T_n^2}{(T_n + T_s)^2} z^{-2} - \frac{T_n^3}{(T_n + T_s)^3} z^{-3}}$ can be estimated by the discrete-time transfer function $H_d(z) = \frac{1}{1 + d^d_1 z^{-1} + d^d_2 z^{-2} + d^d_3 z^{-3}}$. The time constant is computed as $\hat{T}_n = \frac{1}{3}\left(T_{n,d^d_1} + T_{n,d^d_2} + T_{n,d^d_3}\right)$.

- Remnant order $m = 4$: The remnant filter $H^4_{n,dis}(z) = \frac{K_n\left(\frac{T_s}{T_n+T_s}\right)^4}{1-\frac{4T_n}{T_n+T_s}z^{-1}+\frac{6T_n^2}{(T_n+T_s)^2}z^{-2}-\frac{4T_n^3}{(T_n+T_s)^3}z^{-3}+\frac{T_n^4}{(T_n+T_s)^4}z^{-4}}$ can be estimated by the discrete-time transfer function $H_d(z) = \frac{1}{1+d_1^d z^{-1}+d_2^d z^{-2}+d_3^d z^{-3}+d_4^d z^{-4}}$. The time constant is computed as $\hat{T}_n = \frac{1}{4}\left(T_{n,d_1^d} + T_{n,d_2^d} + T_{n,d_3^d} + T_{n,d_4^d}\right)$.

Finally, after obtaining $\hat{\sigma}_{\varepsilon'}$ from the discrete-time model and calculating $\hat{T}_n$, the remnant gain $K_n$ is estimated based on the formula in Equation B.12:

$$\hat{K}_n = \frac{\left(\hat{T}_n + T_s\right)^m}{T_s^m}\frac{\hat{\sigma}_{\varepsilon'}}{\sigma_\varepsilon}. \tag{B.15}$$

And considering a unitary variance of the original remnant noise:

$$\hat{K}_n = \frac{\left(\hat{T}_n + T_s\right)^m}{T_s^m}\hat{\sigma}_{\varepsilon'}. \tag{B.16}$$

## B.3. Requirements for discrete-time parameters

In order to ensure the convertibility of the discrete-time state-space system shown in Equation B.6 into a continuous-time one, the logarithm of the extended matrix must provide a unique solution. Gajic [8] makes two statements:

- The logarithm of a matrix exists if and only if such a matrix is invertible.
- If a matrix has no negative real eigenvalues, then there is a unique logarithm of such a matrix. This logarithm is known as the principal logarithm.

Therefore, the eigenvalues of the discrete-time extended matrix are computed to check whether both statements are fulfilled:

$$A_{E,d} = \left[\begin{array}{c|c} A_d & B_d \\ \hline 0 & I \end{array}\right] = \begin{bmatrix} 0 & 1 & 0 \\ -a_2^d & -a_1^d & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \rightarrow \quad |\, A_{E,d} - \lambda I \,| = \begin{vmatrix} -\lambda & 1 & 0 \\ -a_2^d & -a_1^d - \lambda & 1 \\ 0 & 0 & 1-\lambda \end{vmatrix} = (1-\lambda)\begin{vmatrix} -\lambda & 1 \\ -a_2^d & -a_1^d - \lambda \end{vmatrix} =$$
$$= (1-\lambda)(\lambda^2 + a_1^d\lambda + a_2^d). \tag{B.17}$$

Three eigenvalues are obtained from this determinant:

$$\lambda_1 = 1, \qquad \lambda_2 = \frac{-a_1^d + \sqrt{\left(a_1^d\right)^2 - 4a_2^d}}{2}, \qquad \lambda_3 = \frac{-a_1^d - \sqrt{\left(a_1^d\right)^2 - 4a_2^d}}{2}. \tag{B.18}$$

To avoid any negative real eigenvalue, two inequalities can be extracted from the previous expressions:

$$\begin{cases} a_1^d < 0 \\ a_1^d < \sqrt{\left(a_1^d\right)^2 - 4a_2^d} \end{cases} \quad \rightarrow \quad \left(a_1^d\right)^2 > \left(a_1^d\right)^2 - 4a_2^d \quad \rightarrow \quad a_2^d > 0 \tag{B.19}$$

Thus, it can be concluded that the extended matrix is invertible, so its logarithm exists, and also it would be unique if the conditions $\left\{a_1^d < 0, a_2^d > 0\right\}$ are satisfied.

# C

# Relative Bias Results

## C.1. Simulation Condition C1

### C.1.1. ARX

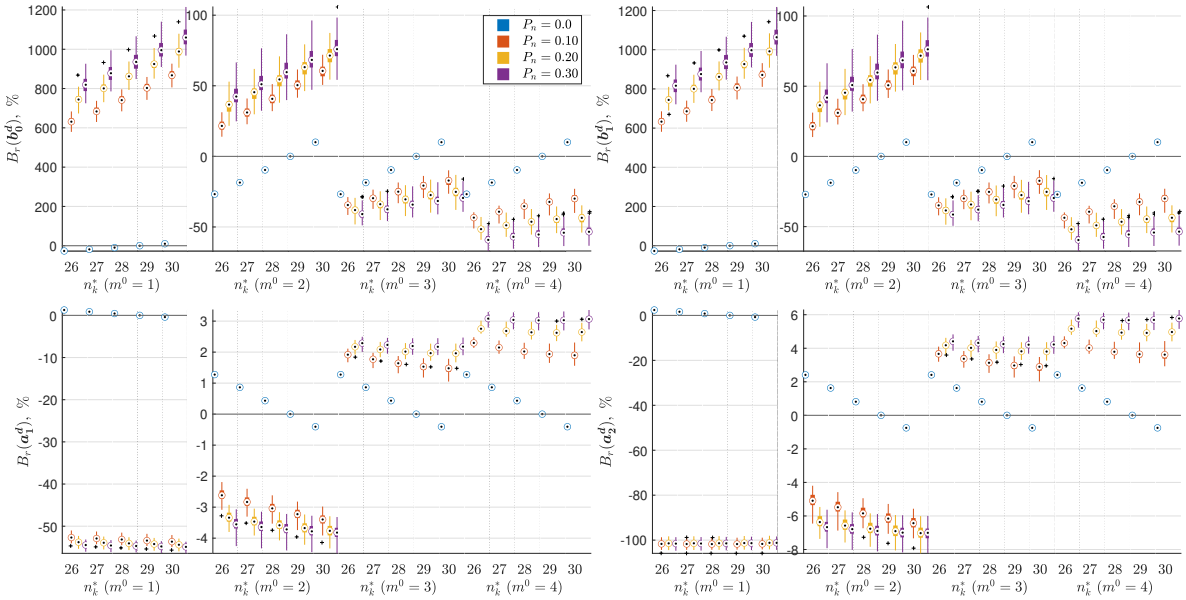


**Figure C.1:** Relative bias results in discrete-time parameters for ARX model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

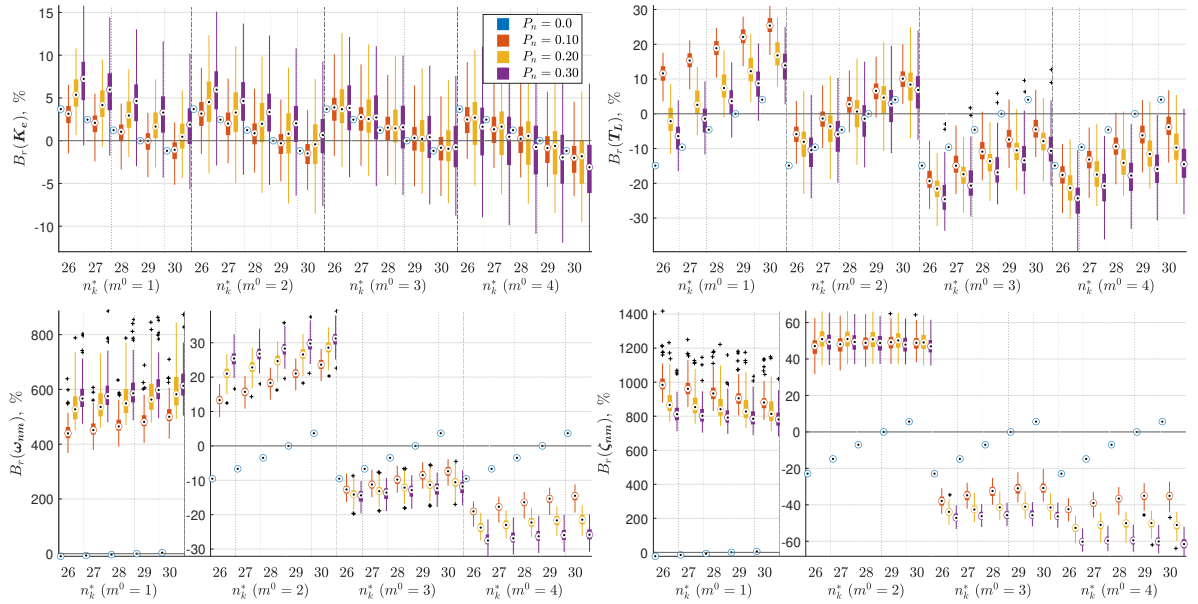


**Figure C.2:** Relative bias results in continuous-time parameters for ARX model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

## C.1.2. BJ: $m^* \in \{1, 2, 3, 4\}$

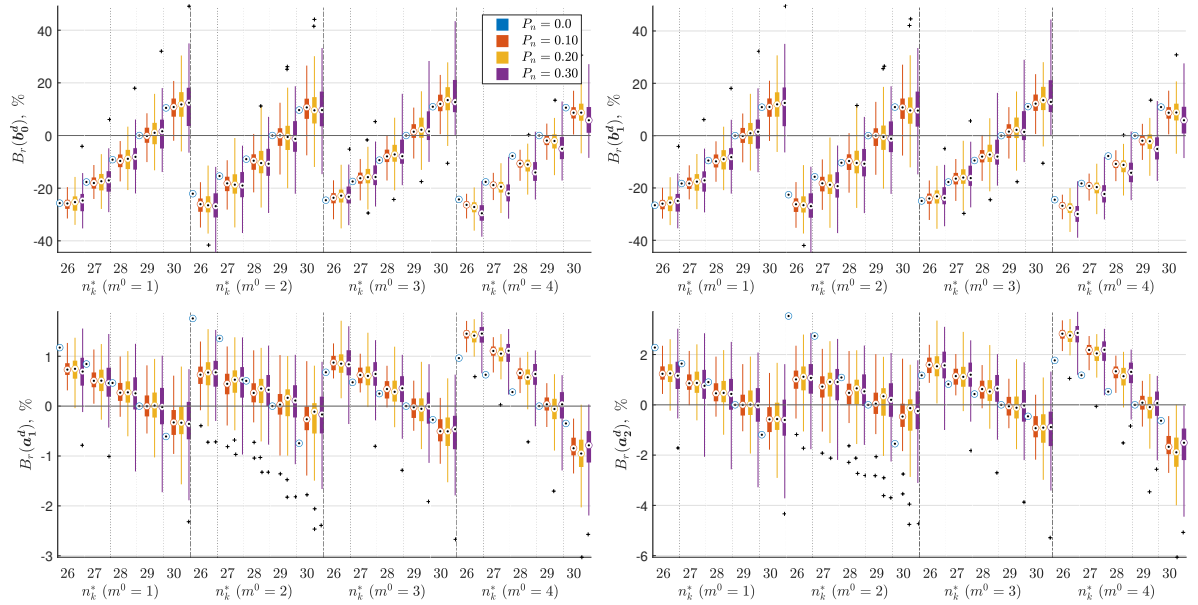


**Figure C.3:** Relative bias results in discrete-time parameters for BJ model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = m^0$. Box and Whisker plots made from $M = 100$ realizations.

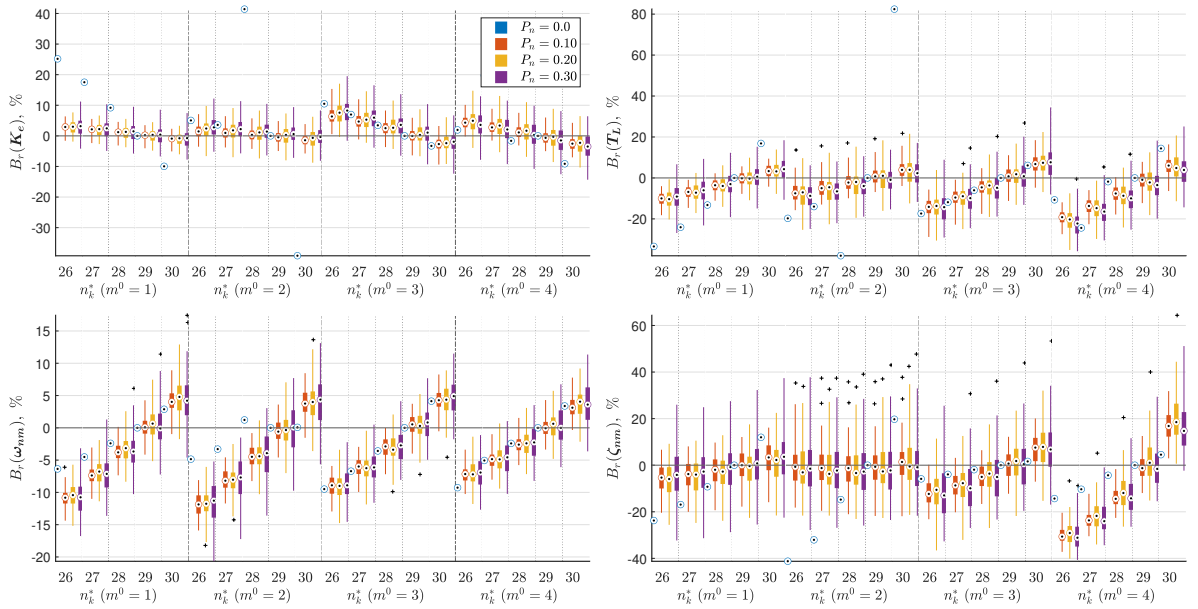


**Figure C.4:** Relative bias results in continuous-time parameters for BJ model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = m^0$. Box and Whisker plots made from $M = 100$ realizations.

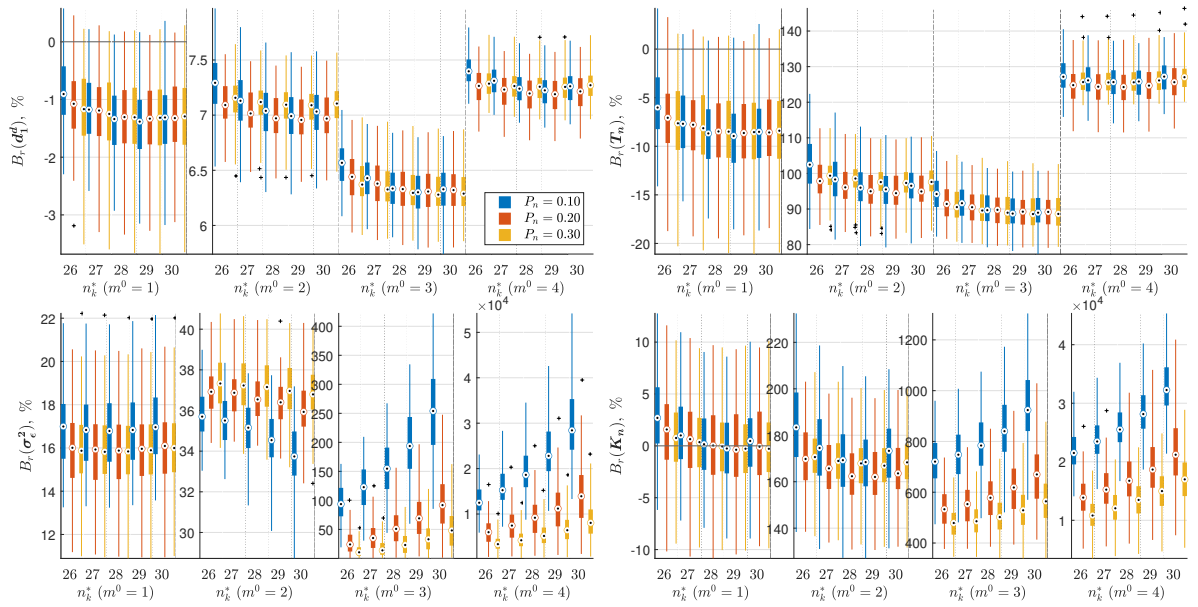


**Figure C.5:** Relative bias results in remnant filter parameters for BJ model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = m^0$. Box and Whisker plots made from $M = 100$ realizations.

### C.1.3. BJ: $m^* = 1$

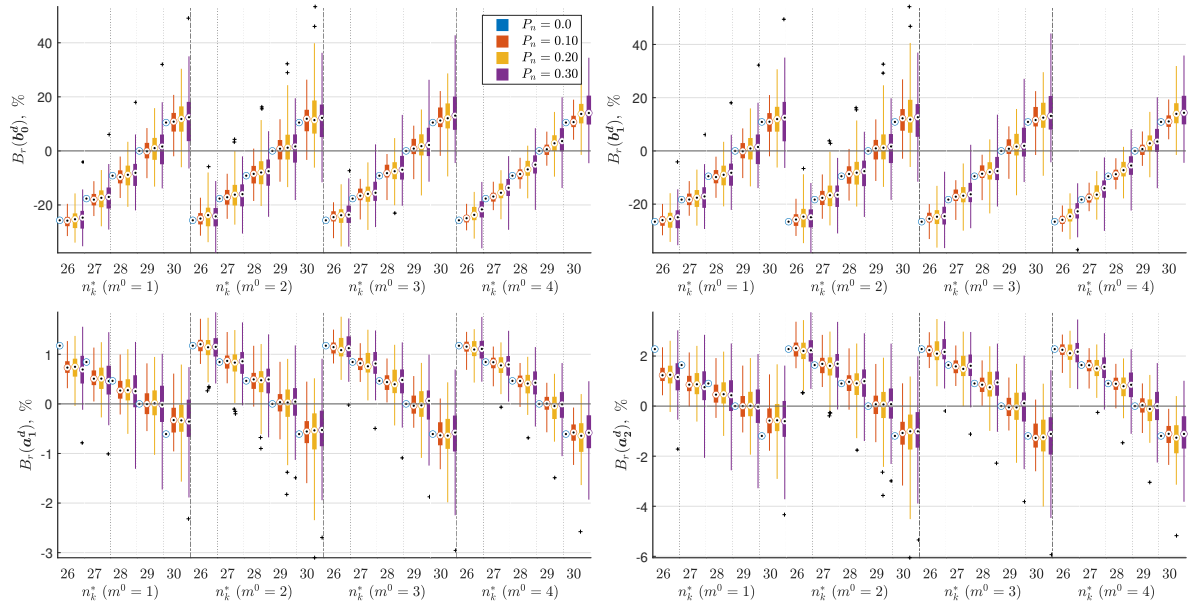


**Figure C.6:** Relative bias results in discrete-time parameters for BJ model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = 1$. Box and Whisker plots made from $M = 100$ realizations.

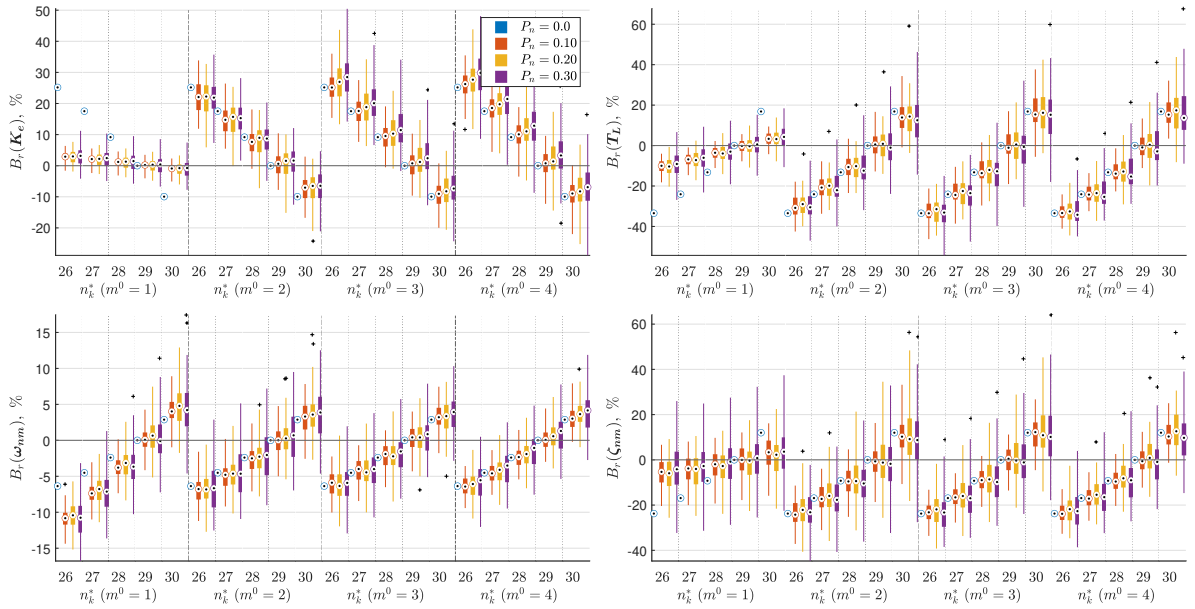


**Figure C.7:** Relative bias results in continuous-time parameters for BJ model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = 1$. Box and Whisker plots made from $M = 100$ realizations.

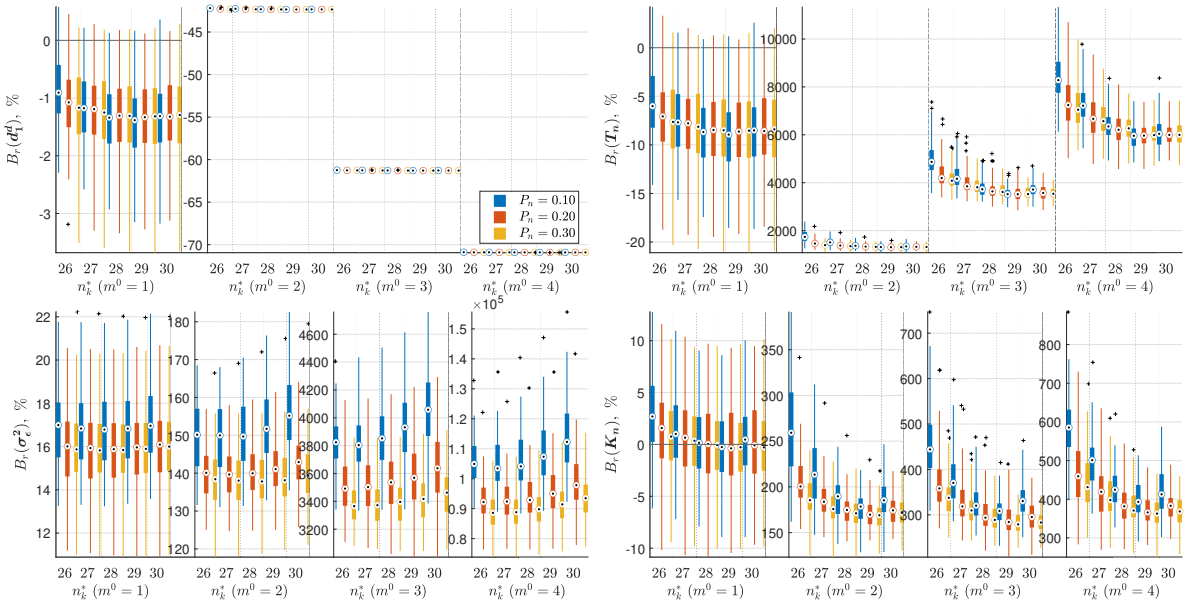


**Figure C.8:** Relative bias results in remnant filter parameters for BJ model: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = 1$. Box and Whisker plots made from $M = 100$ realizations.

# C.2. Simulation Condition C2

## C.2.1. ARX

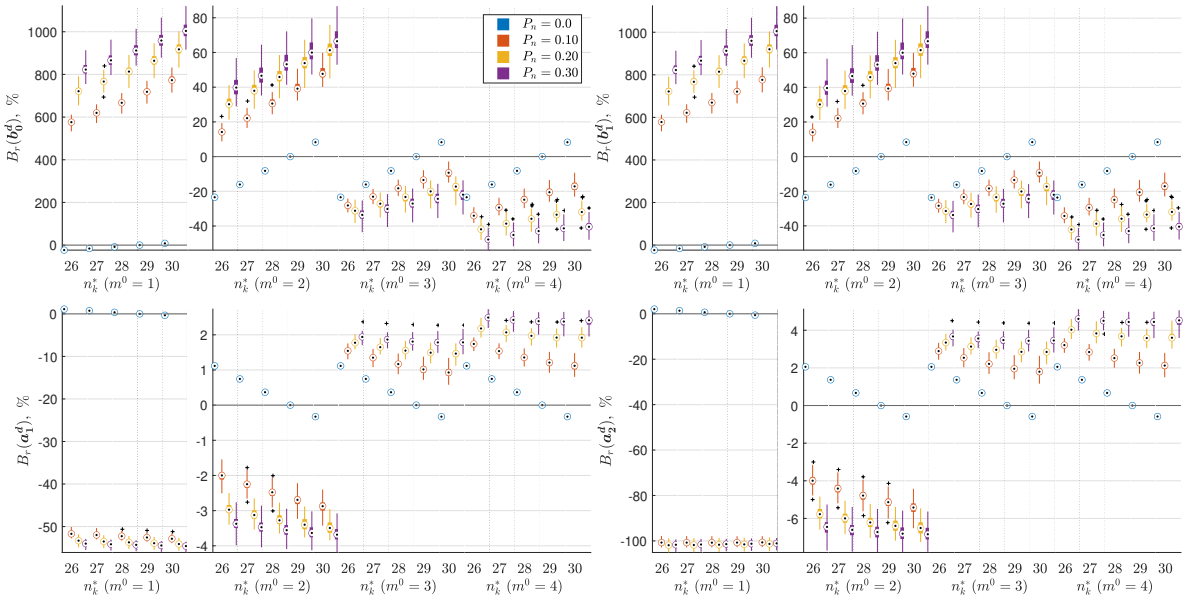


**Figure C.9:** Relative bias results in discrete-time parameters for ARX model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

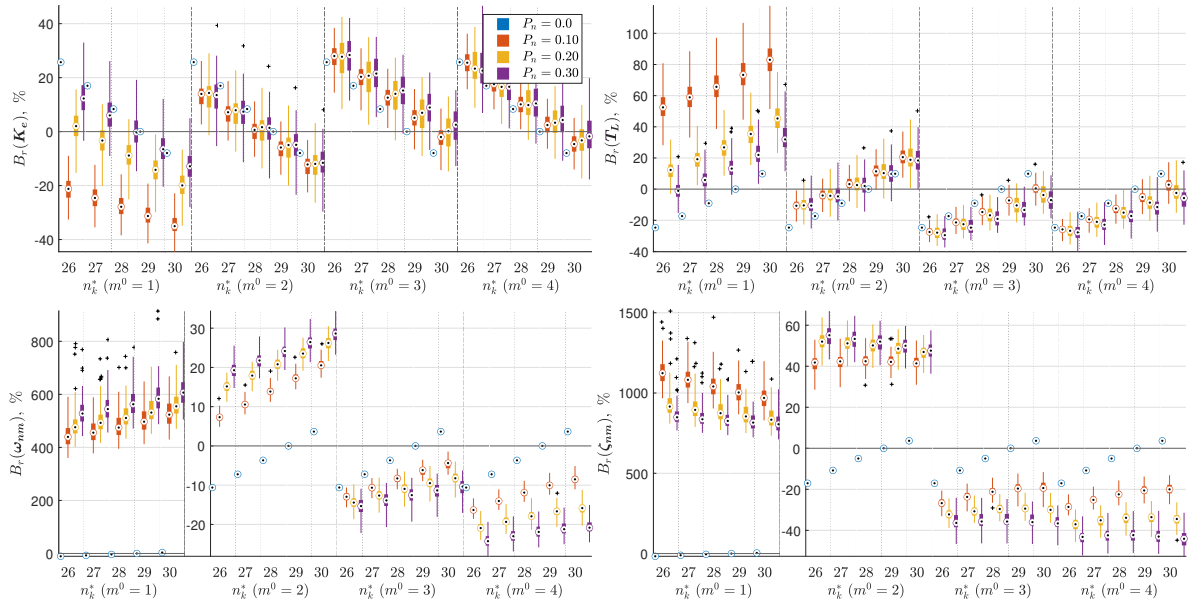


**Figure C.10:** Relative bias results in continuous-time parameters for ARX model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

## C.2.2. BJ: $m^* \in \{1, 2, 3, 4\}$

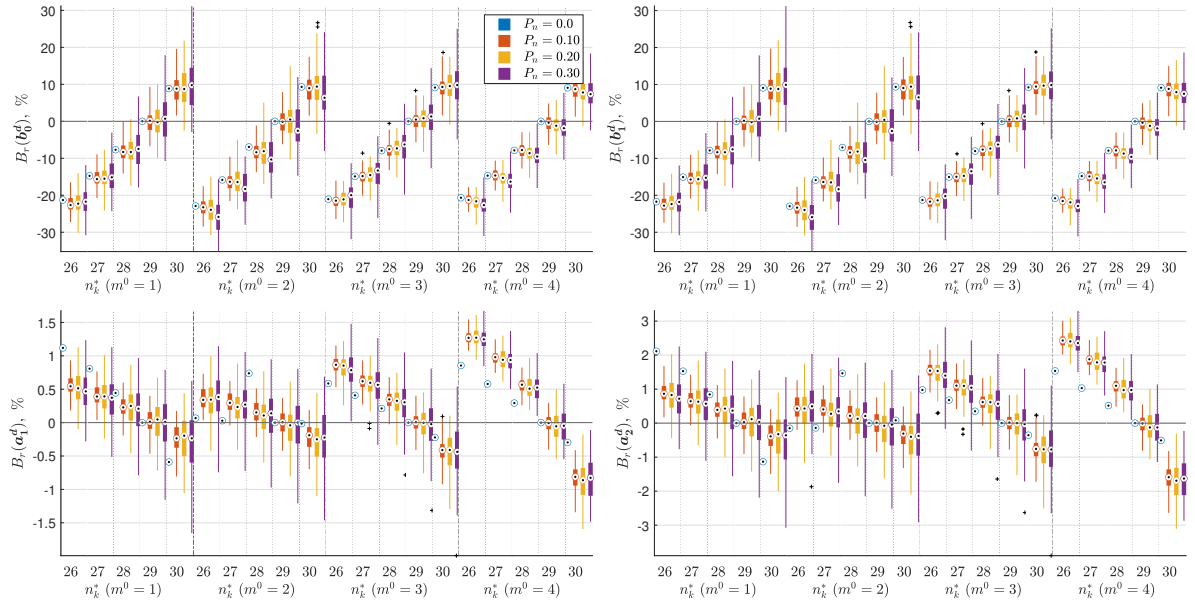


**Figure C.11:** Relative bias results in discrete-time parameters for BJ model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = m^0$. Box and Whisker plots made from $M = 100$ realizations.

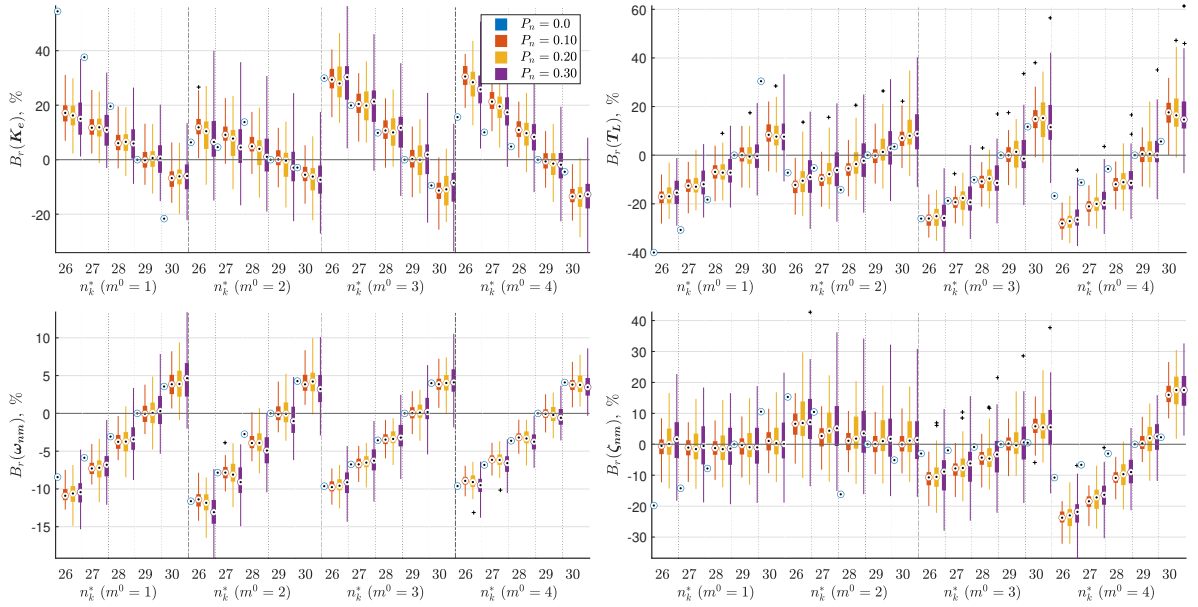


**Figure C.12:** Relative bias results in continuous-time parameters for BJ model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = m^0$. Box and Whisker plots made from $M = 100$ realizations.

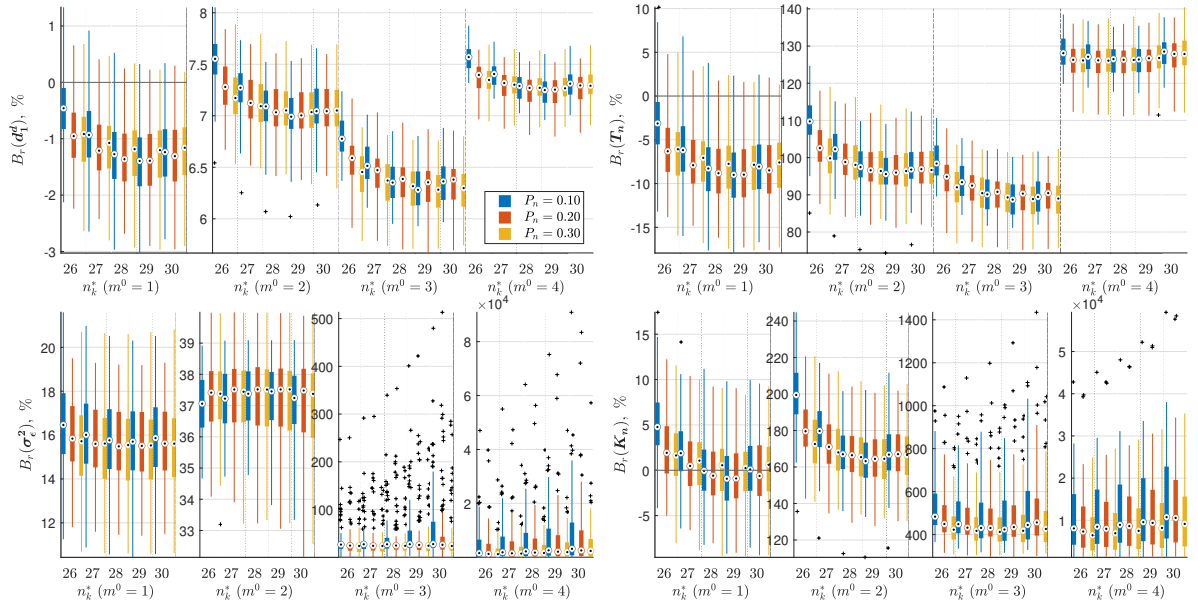


**Figure C.13:** Relative bias results in remnant filter parameters for BJ model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = m^0$. Box and Whisker plots made from $M = 100$ realizations.

### C.2.3. BJ: $m^* = 1$

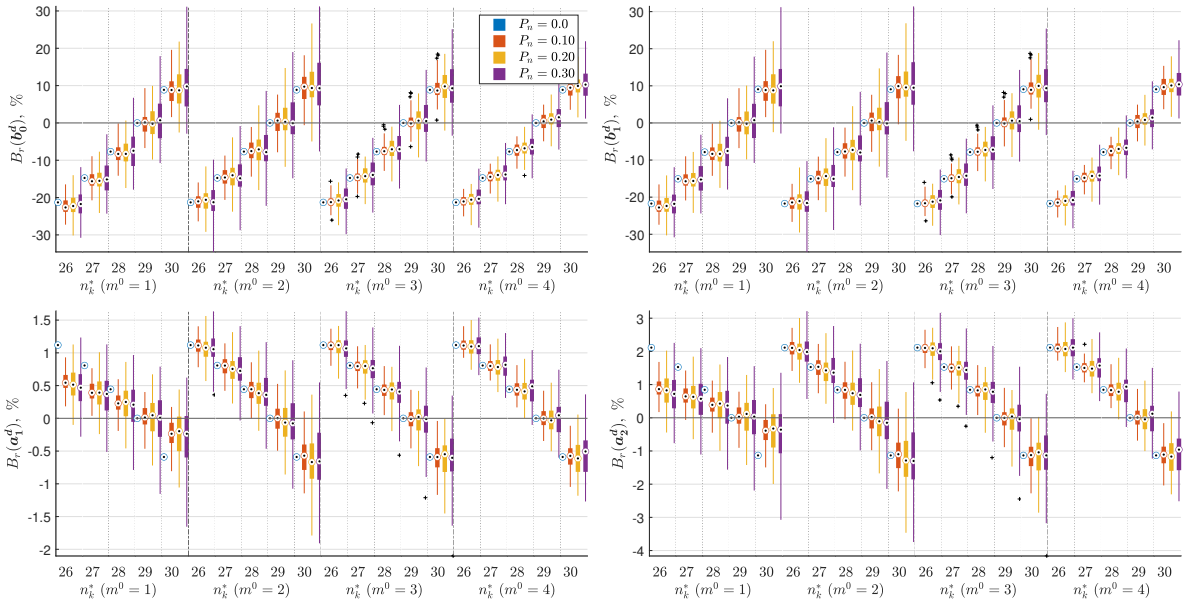


**Figure C.14:** Relative bias results in discrete-time parameters for BJ model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = 1$. Box and Whisker plots made from $M = 100$ realizations.

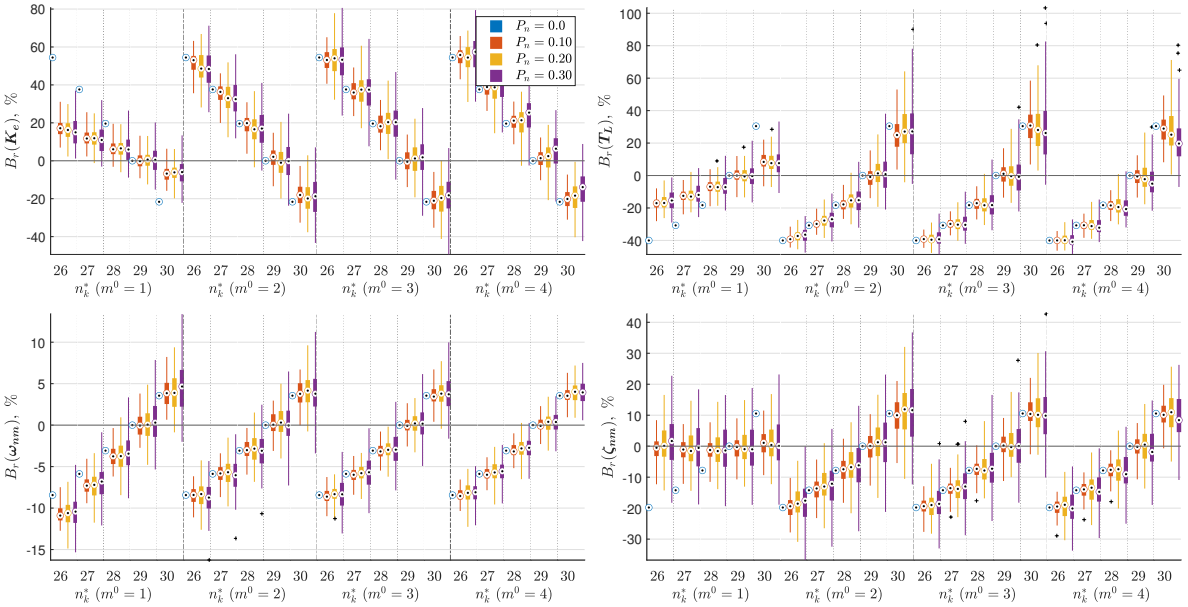


**Figure C.15:** Relative bias results in continuous-time parameters for BJ model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = 1$. Box and Whisker plots made from $M = 100$ realizations.

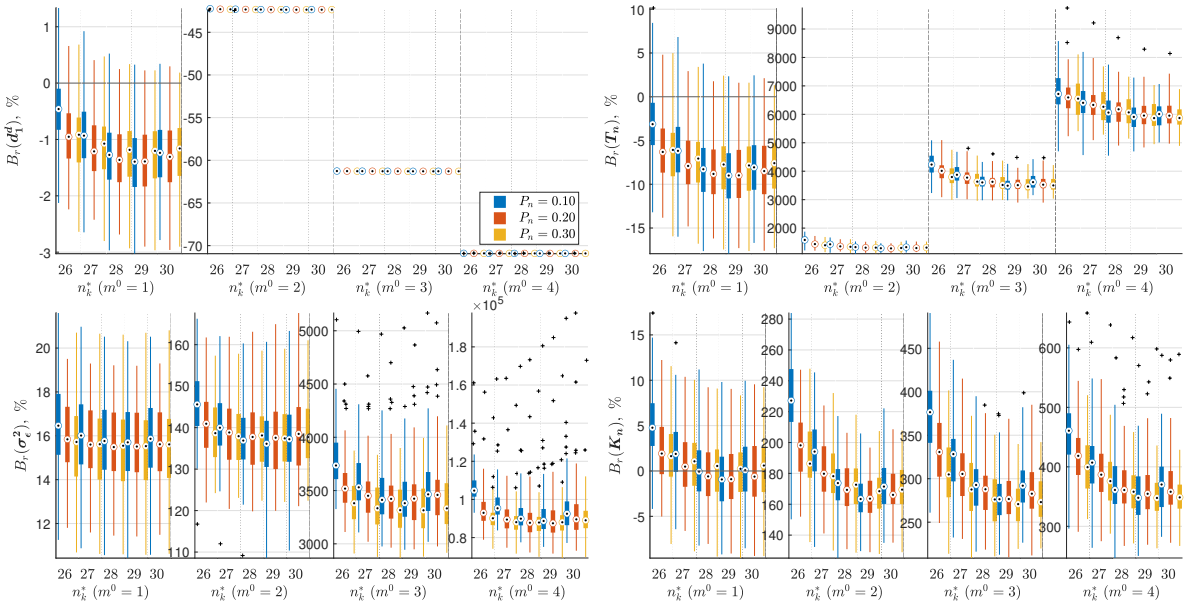


**Figure C.16:** Relative bias results in remnant filter parameters for BJ model: simulation condition C2, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$, model remnant filter orders $m^* = 1$. Box and Whisker plots made from $M = 100$ realizations.

# D

# Bode Plots

## D.1. Simulation Condition C1

### D.1.1. ARX: $n_k^* \in \{26, 27, 28, 29, 30\}$

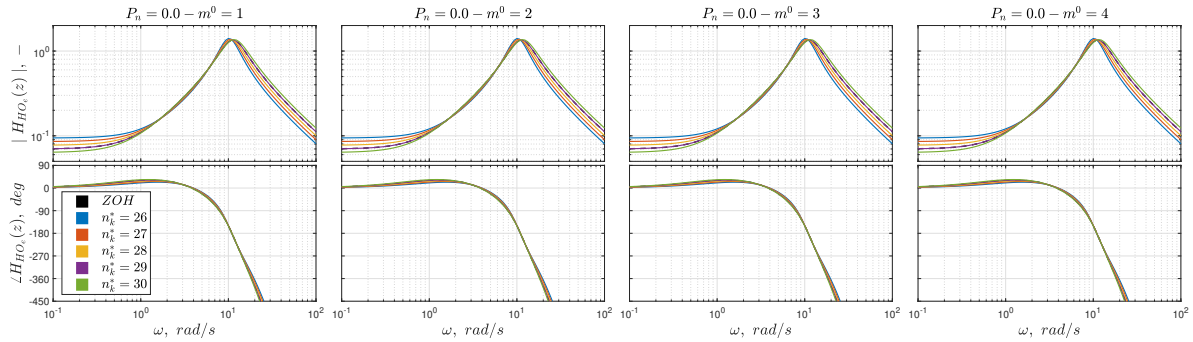


**Figure D.1:** Bode plots for ARX model: simulation condition C1, noise level $P_n = 0.0$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

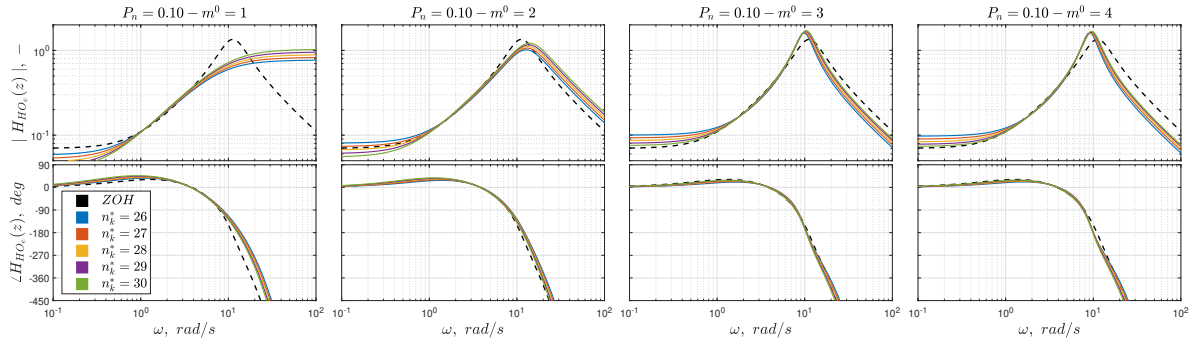


**Figure D.2:** Bode plots for ARX model: simulation condition C1, noise level $P_n = 0.10$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

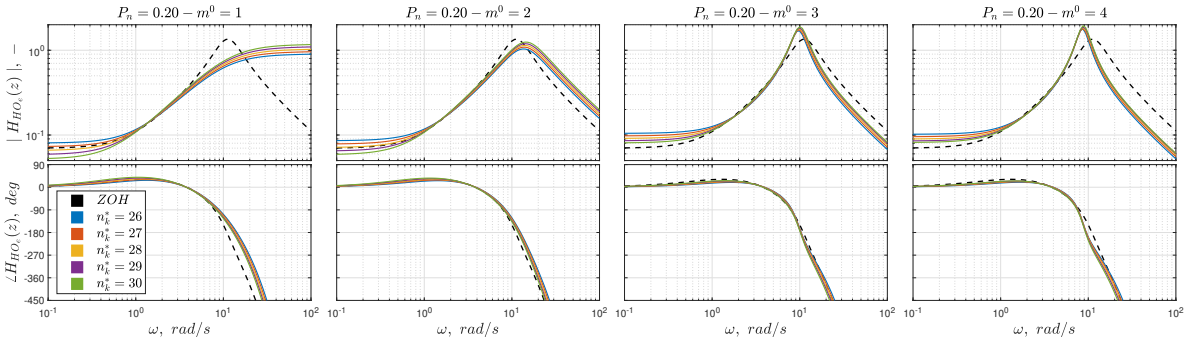


**Figure D.3:** Bode plots for ARX model: simulation condition C1, noise level $P_n = 0.20$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

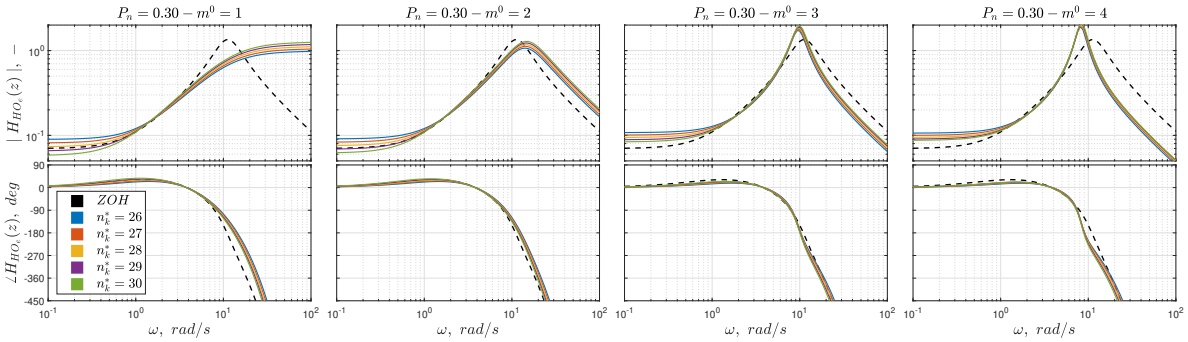


**Figure D.4:** Bode plots for ARX model: simulation condition C1, noise level $P_n = 0.30$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

### D.1.2. BJ: $n_k^* \in \{26, 27, 28, 29, 30\}$

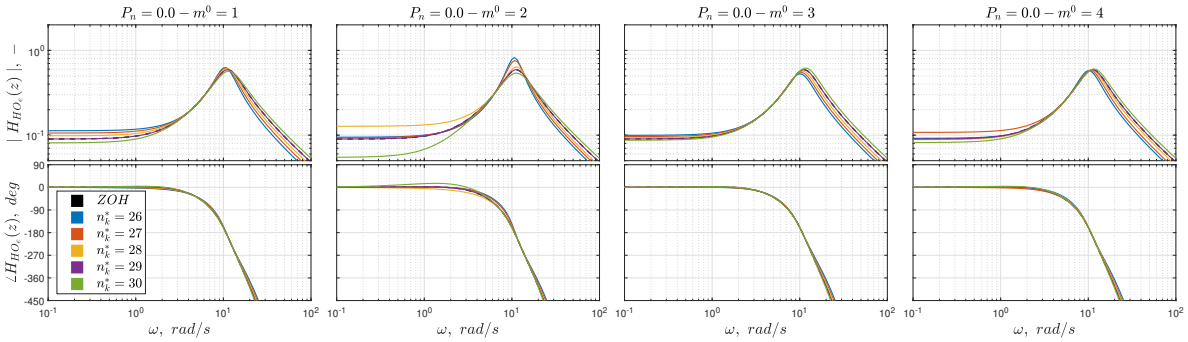


**Figure D.5:** Bode plots for BJ model: simulation condition C1, noise level $P_n = 0.0$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

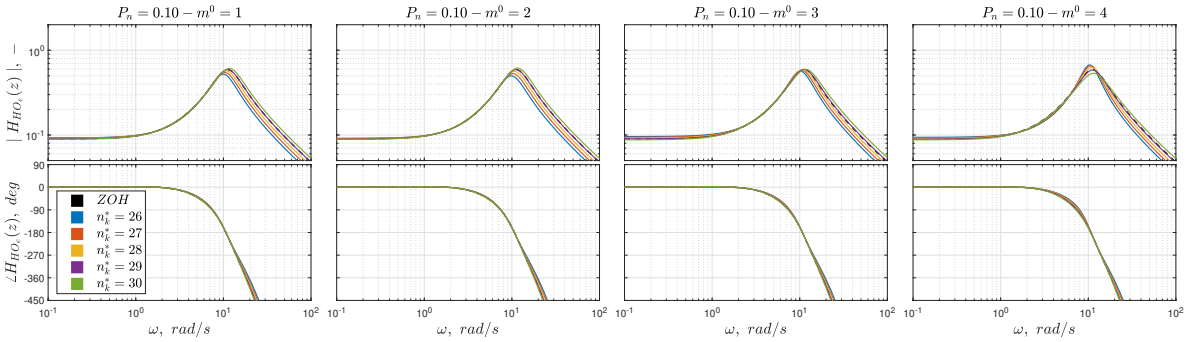


**Figure D.6:** Bode plots for BJ model: simulation condition C1, noise level $P_n = 0.10$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

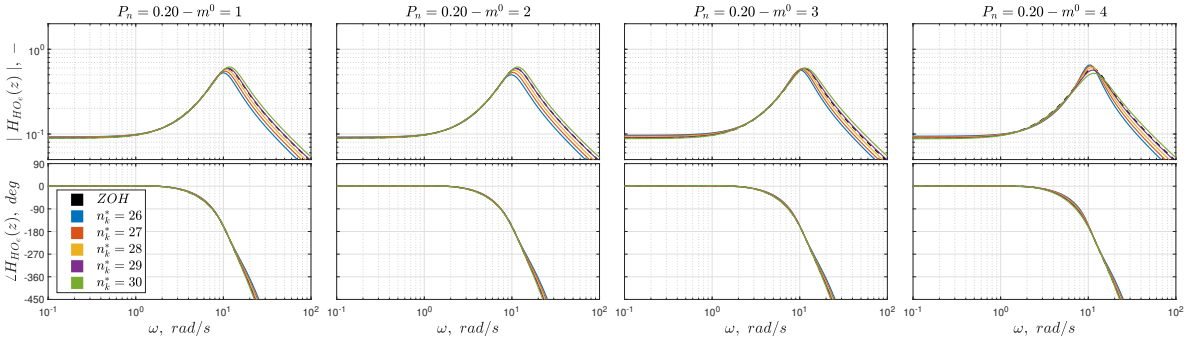


**Figure D.7:** Bode plots for BJ model: simulation condition C1, noise level $P_n = 0.20$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

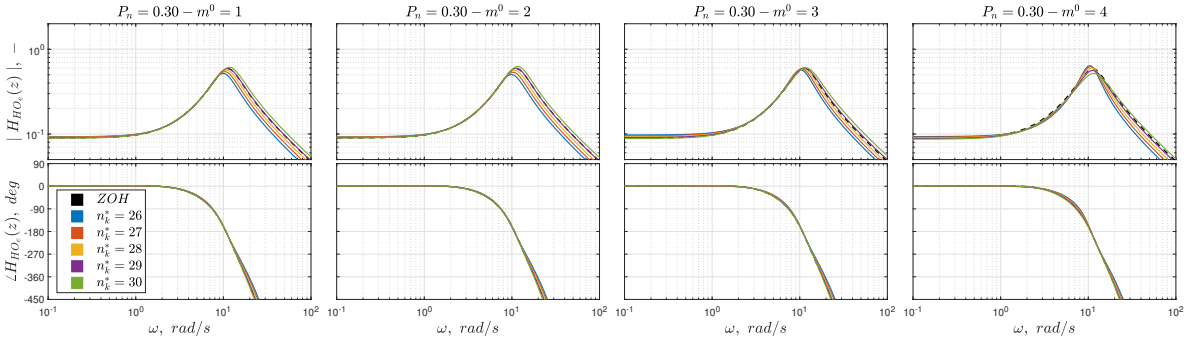


**Figure D.8:** Bode plots for BJ model: simulation condition C1, noise level $P_n = 0.30$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

### D.1.3. ARX vs. BJ: $n_k^* = 29$

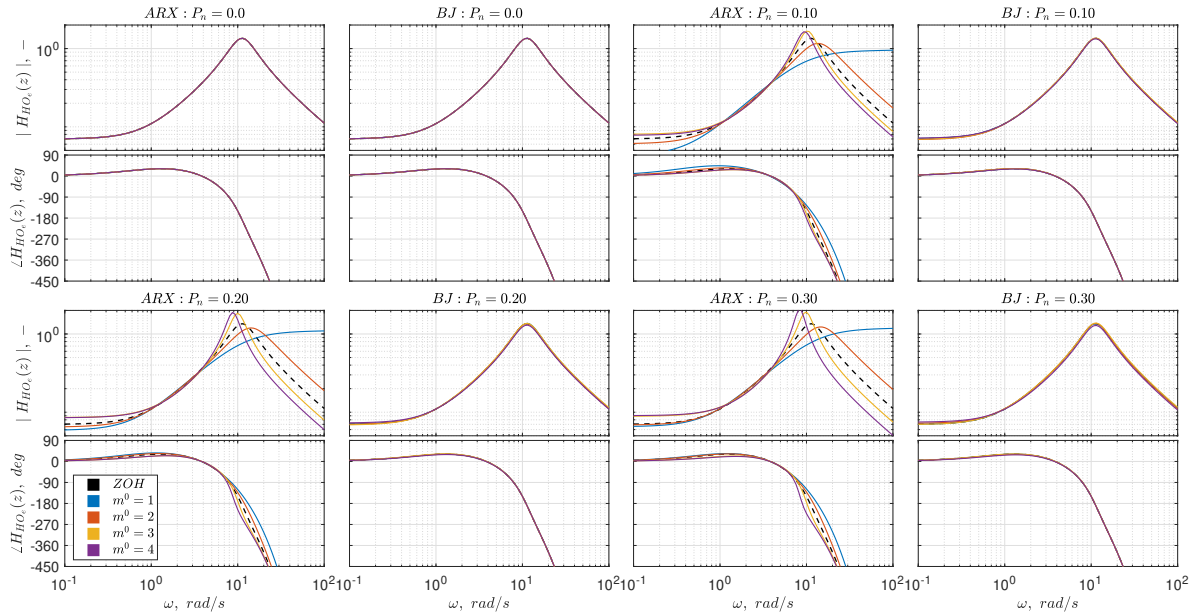


**Figure D.9:** Bode plots for ARX and BJ models: simulation condition C1, noise level $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delay $n_k^* = 29$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

## D.2. Simulation Condition C2

### D.2.1. ARX: $n_k^* \in \{26, 27, 28, 29, 30\}$

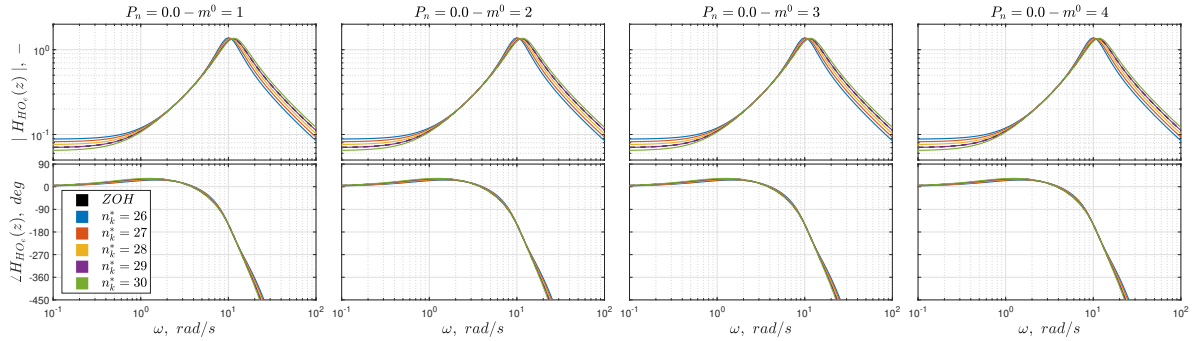


**Figure D.10:** Bode plots for ARX model: simulation condition C2, noise level $P_n = 0.0$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

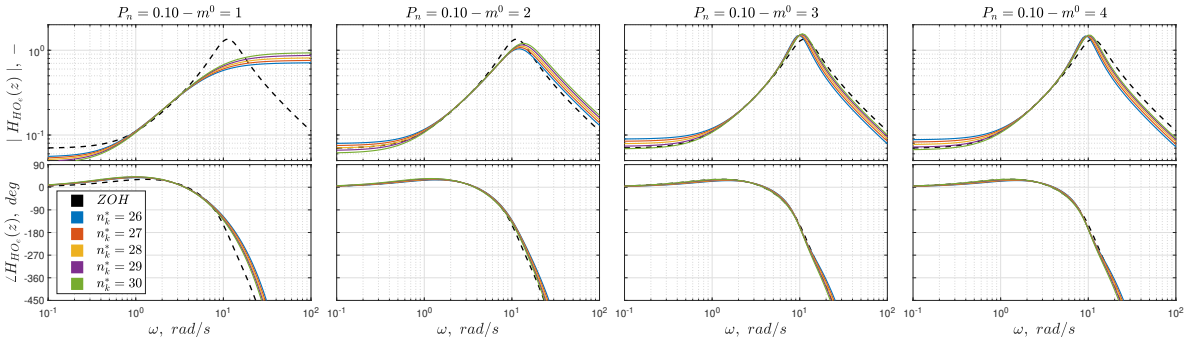


**Figure D.11:** Bode plots for ARX model: simulation condition C2, noise level $P_n = 0.10$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

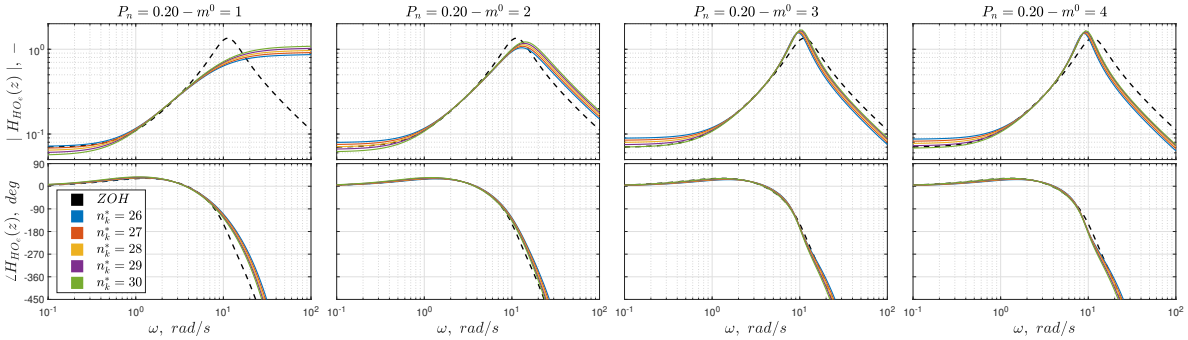


**Figure D.12:** Bode plots for ARX model: simulation condition C2, noise level $P_n = 0.20$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

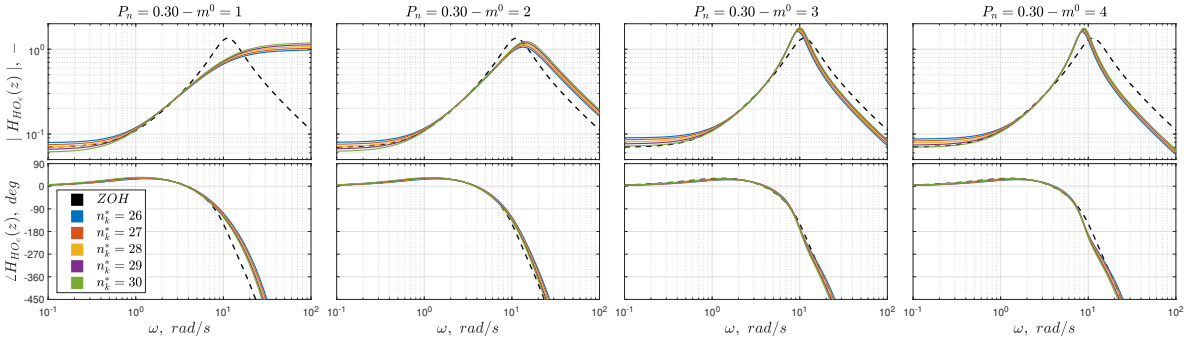


**Figure D.13:** Bode plots for ARX model: simulation condition C2, noise level $P_n = 0.30$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

### D.2.2. BJ: $n_k^* \in \{26, 27, 28, 29, 30\}$

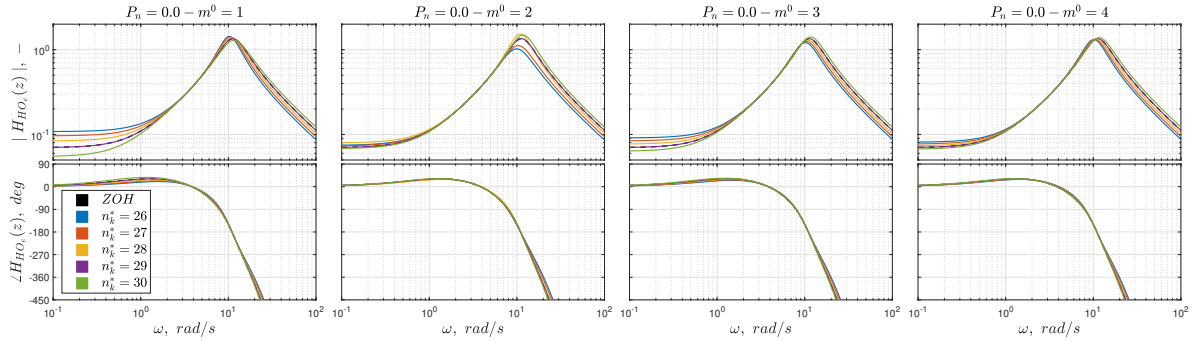


**Figure D.14:** Bode plots for BJ model: simulation condition C2, noise level $P_n = 0.0$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

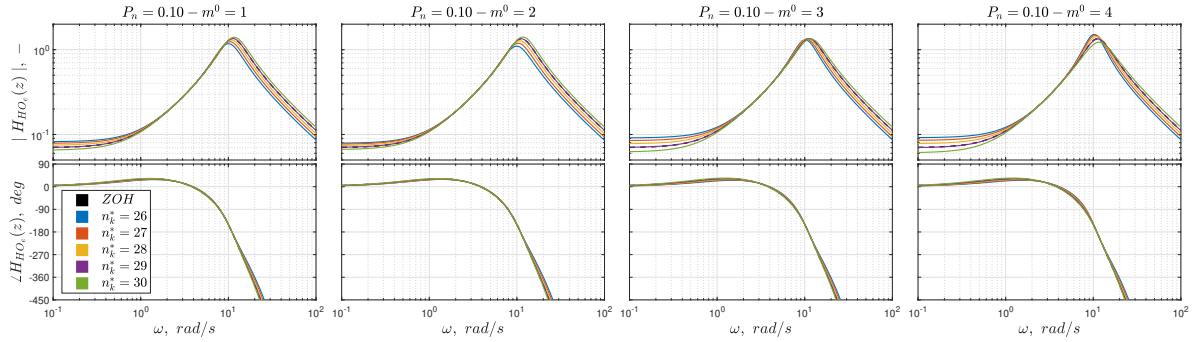


**Figure D.15:** Bode plots for BJ model: simulation condition C2, noise level $P_n = 0.10$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

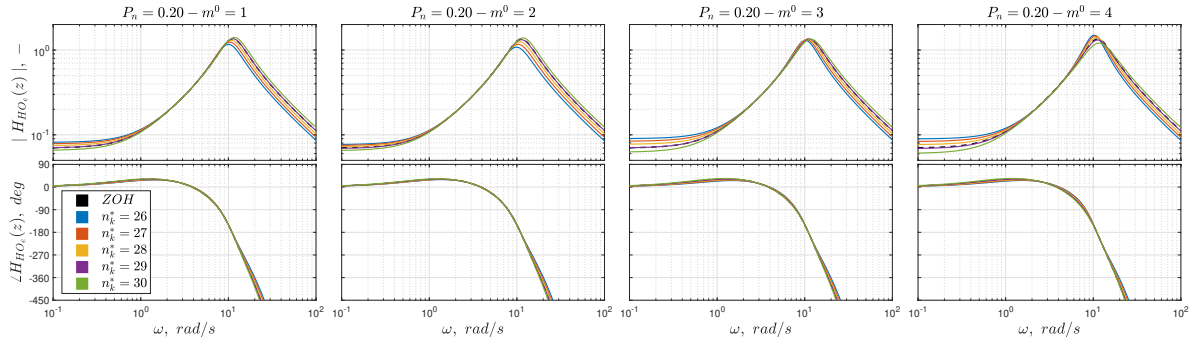


**Figure D.16:** Bode plots for BJ model: simulation condition C2, noise level $P_n = 0.20$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

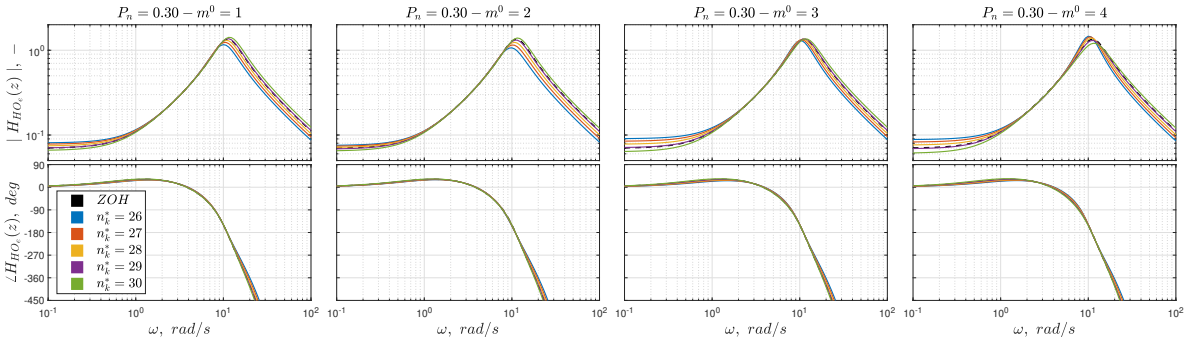


**Figure D.17:** Bode plots for BJ model: simulation condition C2, noise level $P_n = 0.30$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

## D.2.3. ARX vs. BJ: $n_k^* = 29$

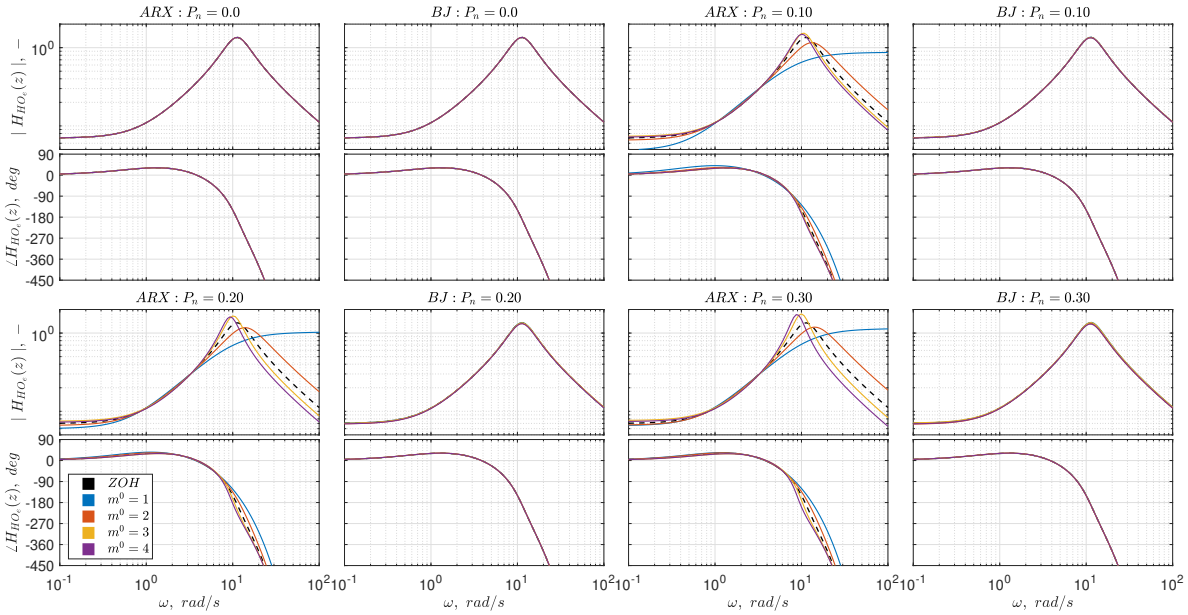


**Figure D.18:** Bode plots for ARX and BJ models: simulation condition C2, noise level $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model remnant filter order $m^* = m^0$, model time-delay $n_k^* = 29$. Simulated HO model (black, dashed line): ZOH discretization. Discrete-time parameters in transfer functions averaged from $M = 100$ realizations.

# E

# VAF and Relative Bias for Multiple Remnant Orders in BJ structure

## E.1. Simulation Condition C1

### E.1.1. Simulation data from $m^0 = 1$

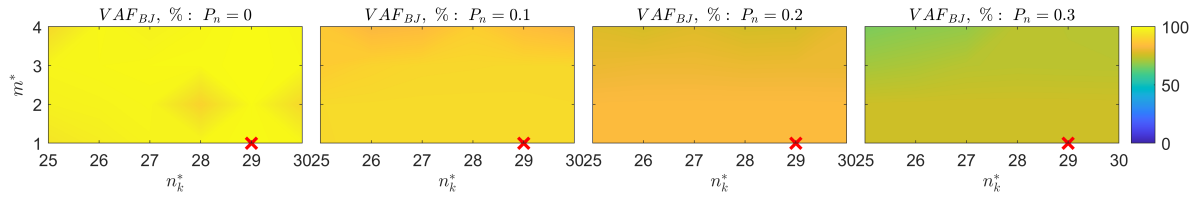


**Figure E.1:** VAF results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 1$ (black, symbol $\times$), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

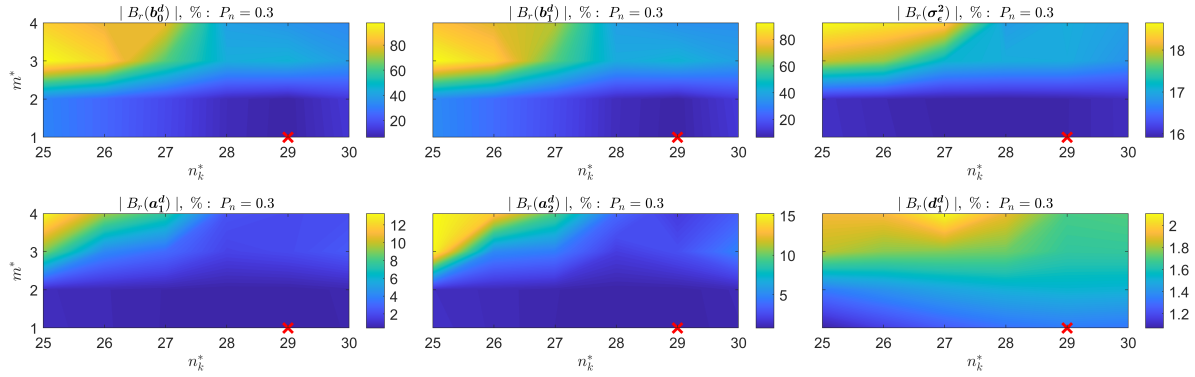


**Figure E.2:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 1$ (black, symbol $\times$), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

## E.1.2. Simulation data from $m^0 = 2$

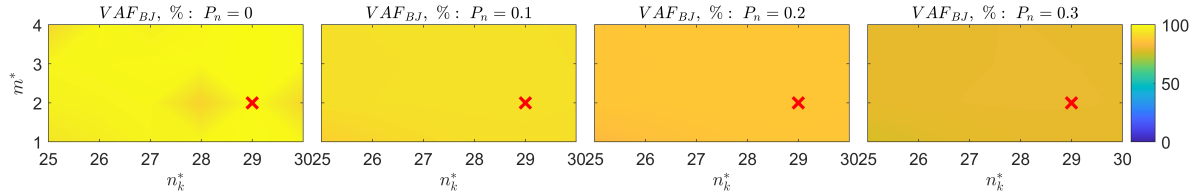


**Figure E.3:** VAF results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 2$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

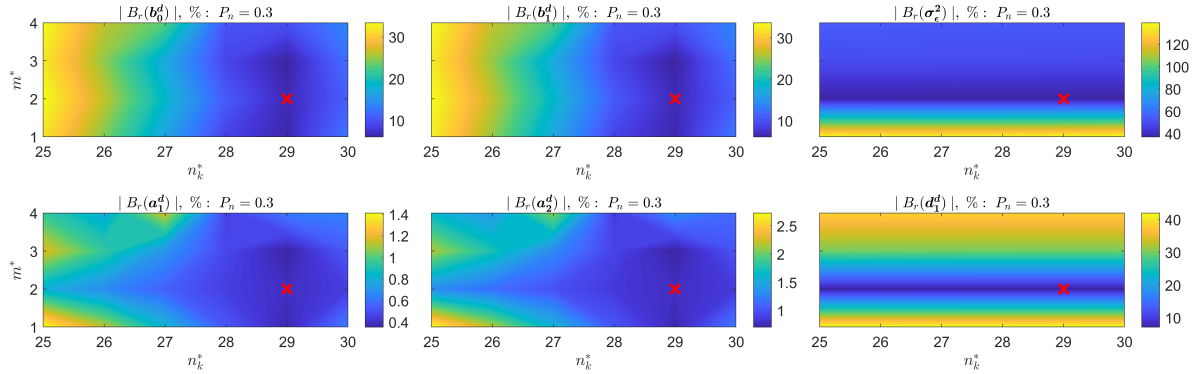


**Figure E.4:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 2$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

## E.1.3. Simulation data from $m^0 = 3$

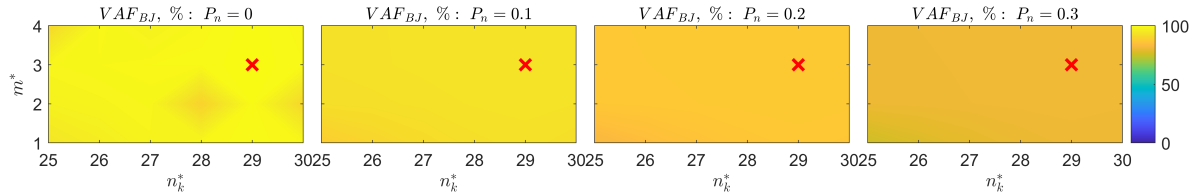


**Figure E.5:** VAF results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 3$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

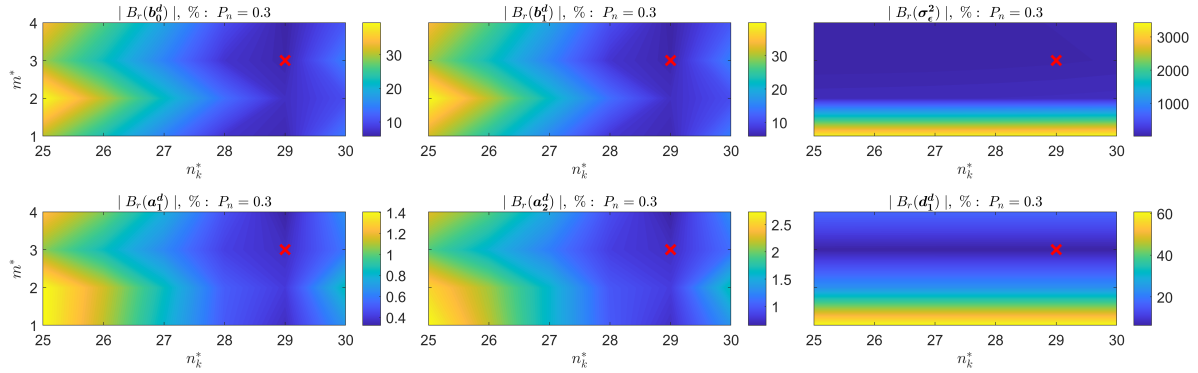


**Figure E.6:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 3$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

## E.1.4. Simulation data from $m^0 = 4$

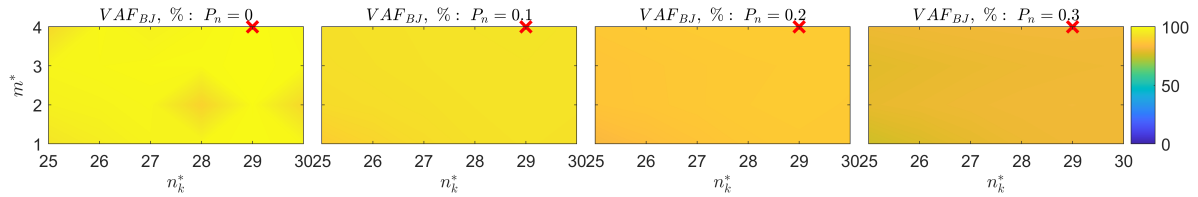


**Figure E.7:** VAF results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 4$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

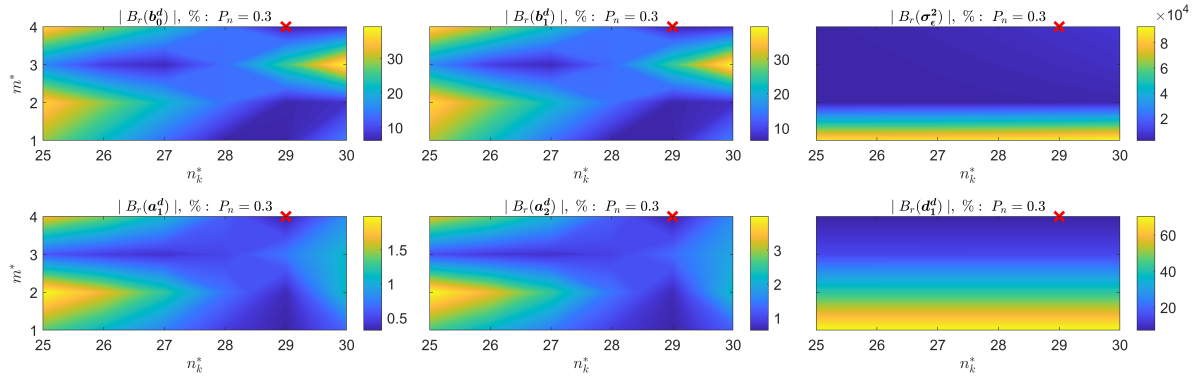


**Figure E.8:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C1, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 4$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

# E.2. Simulation Condition C2

## E.2.1. Simulation data from $m^0 = 1$

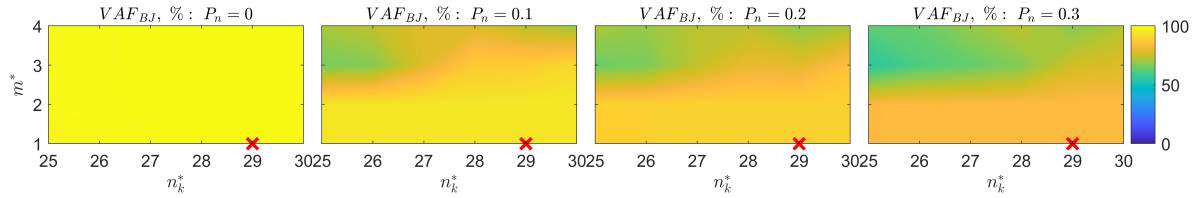


**Figure E.9:** VAF results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 1$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

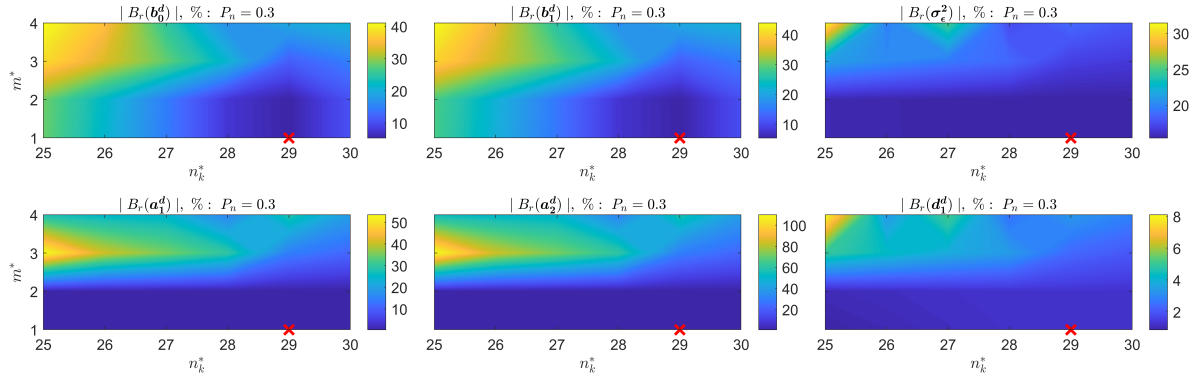


**Figure E.10:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 1$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

## E.2.2. Simulation data from $m^0 = 2$

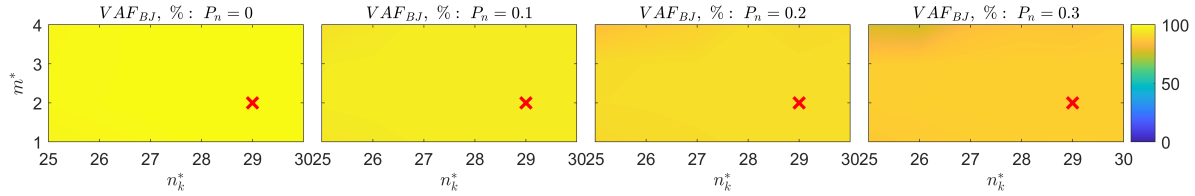


**Figure E.11:** VAF results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 2$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

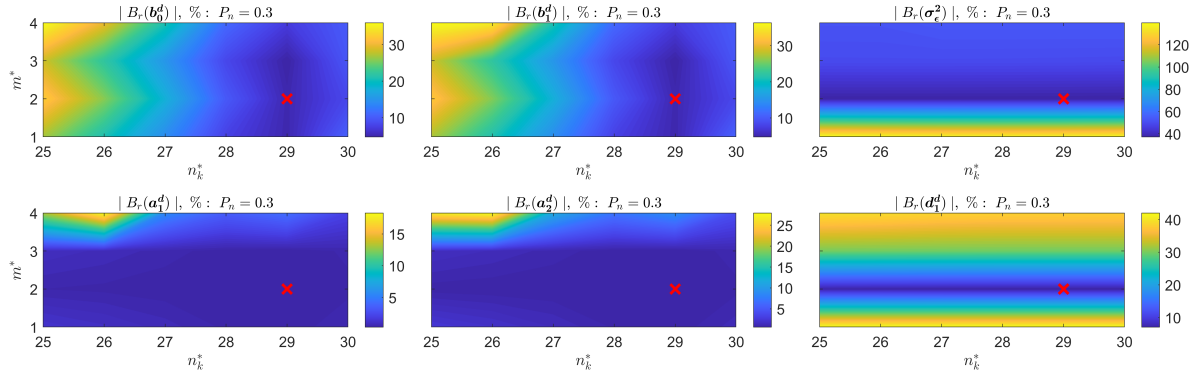


**Figure E.12:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 2$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

## E.2.3. Simulation data from $m^0 = 3$

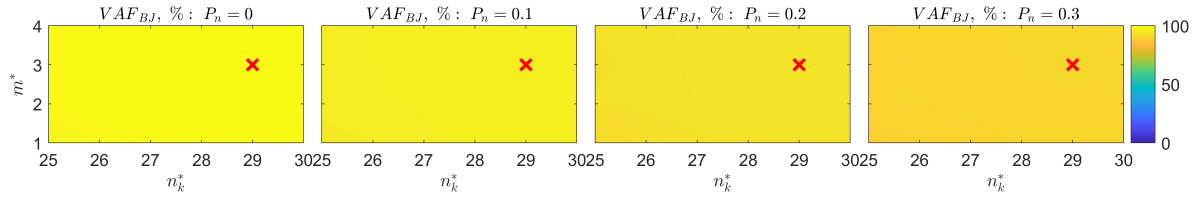


**Figure E.13:** VAF results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 3$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

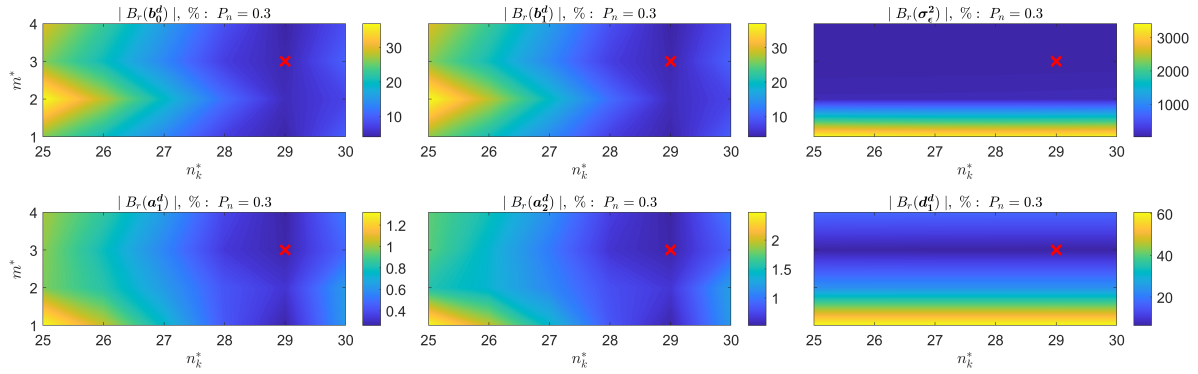


**Figure E.14:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 3$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

## E.2.4. Simulation data from $m^0 = 4$

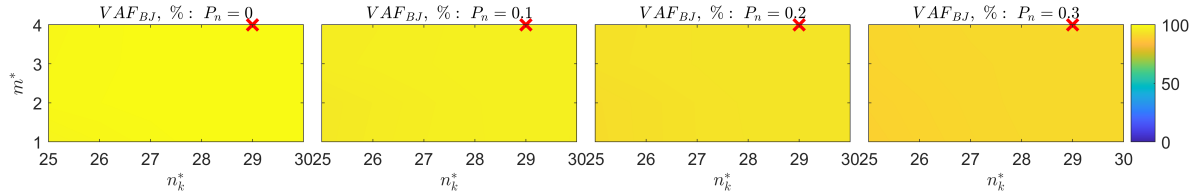


**Figure E.15:** VAF results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 4$ (black, symbol ×), noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

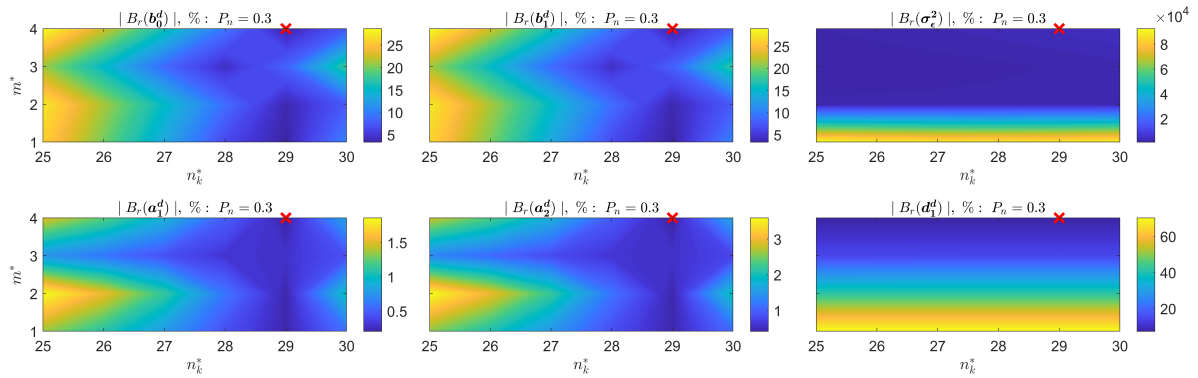


**Figure E.16:** Absolute relative bias results of discrete-time parameters for BJ model: simulation condition C2, model remnant filter orders $m^* \in \{1, 2, 3, 4\}$, simulation remnant filter order $m^0 = 4$ (black, symbol ×), noise levels $P_n = 0.30$, model time-delays $n_k^* \in \{25, 26, 27, 28, 29, 30\}$. Obtained values averaged from $M = 100$ realizations.

# F

# Multiple Initial Conditions in BJ structure

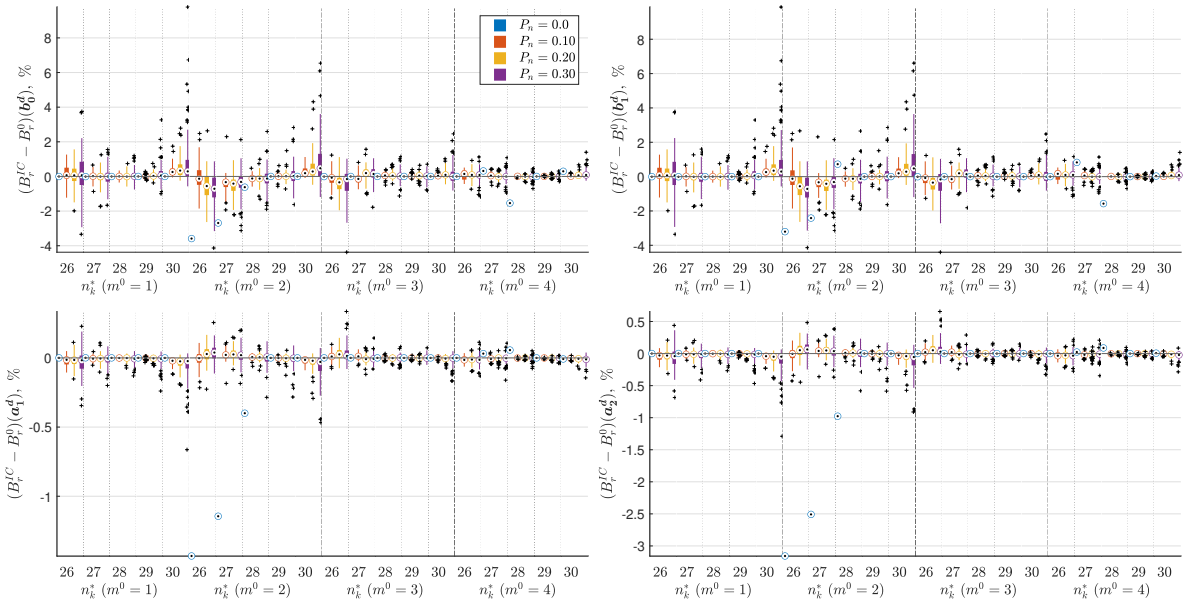


**Figure F.1:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 1: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

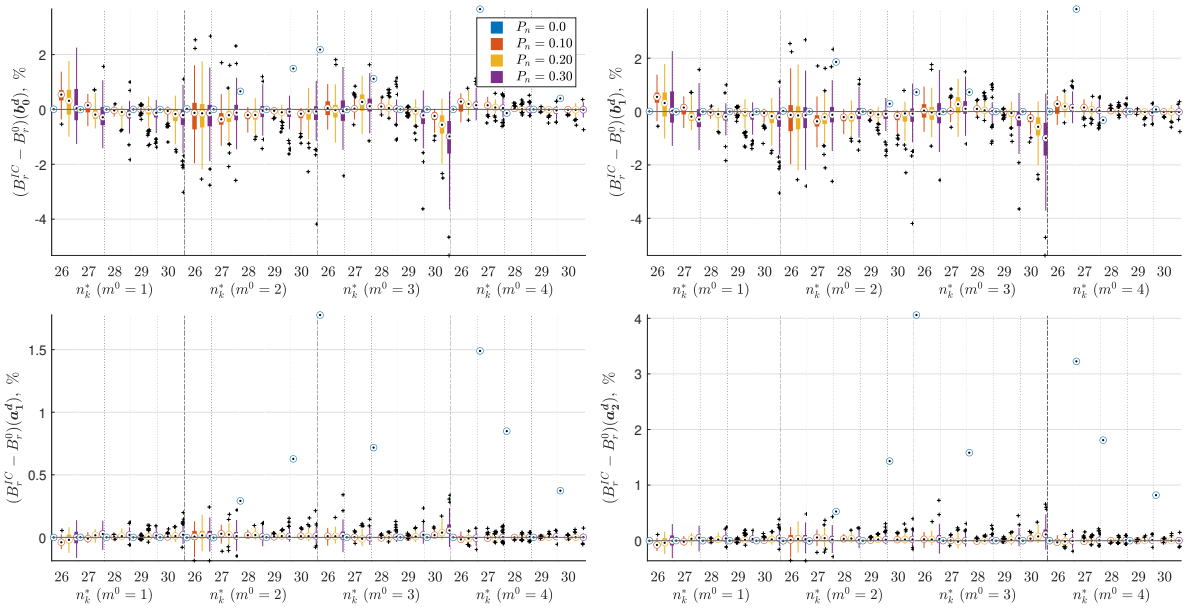


**Figure F.2:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 2: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

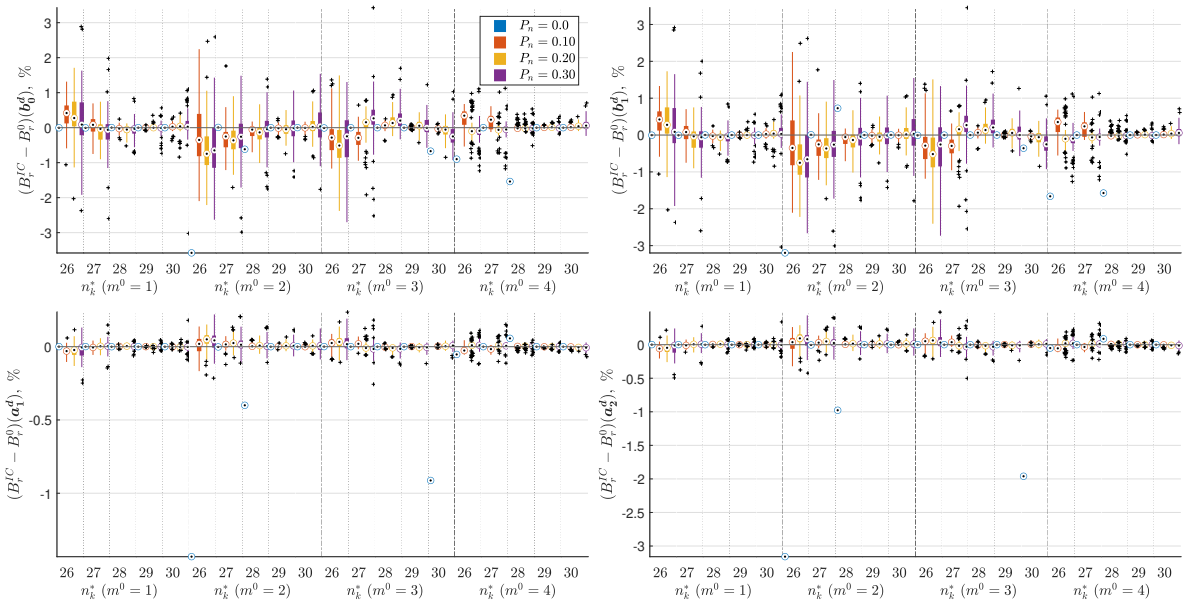


**Figure F.3:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 3: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

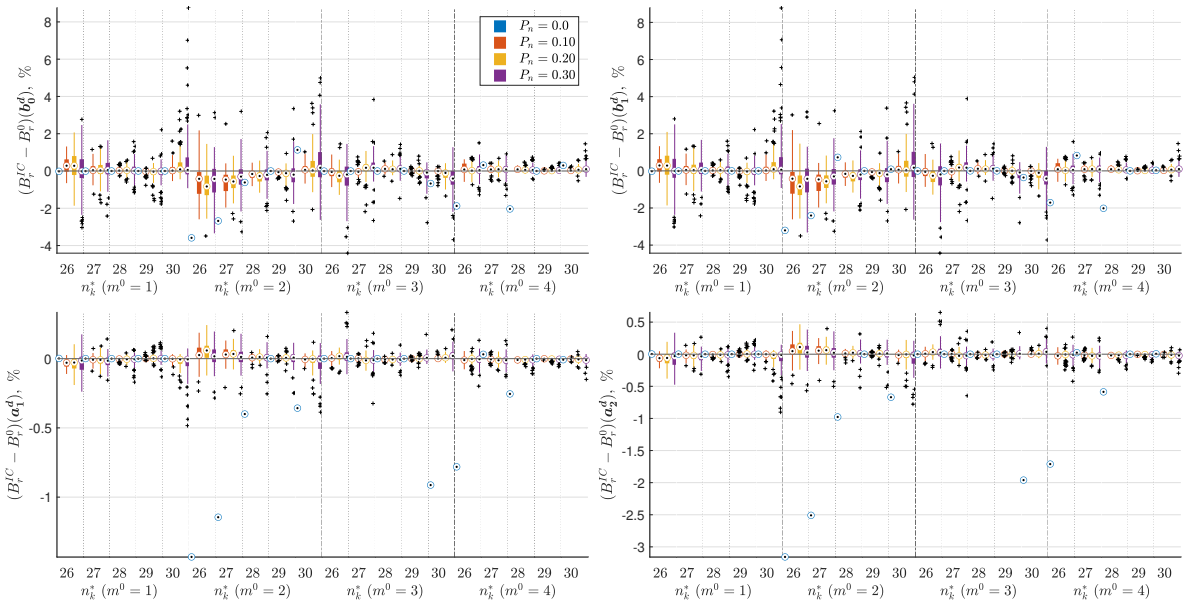


**Figure F.4:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 4: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

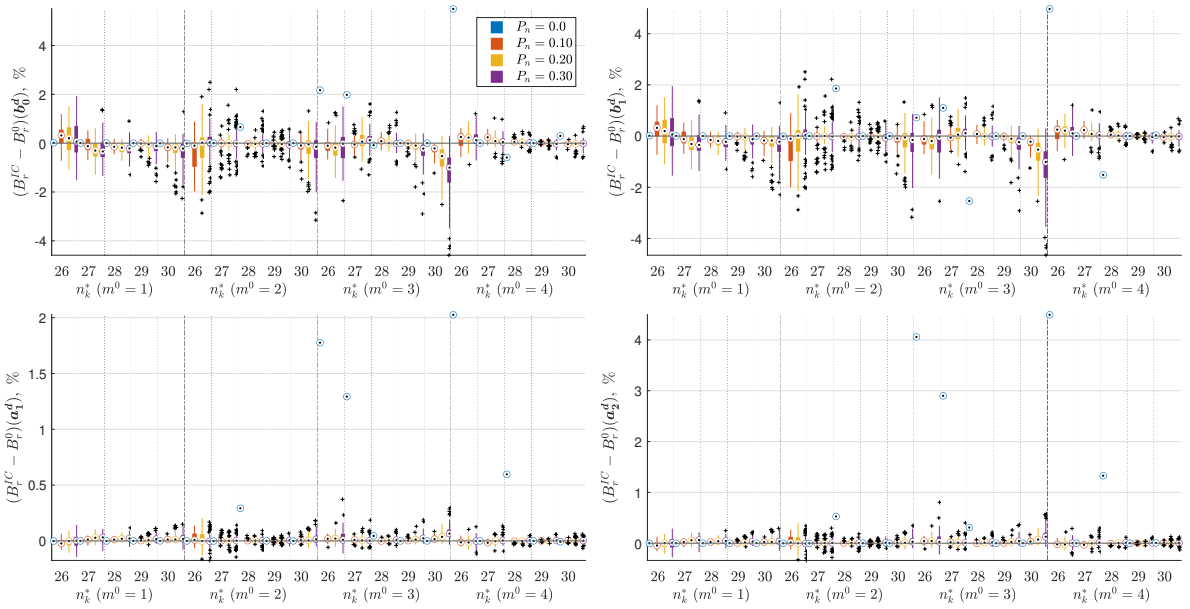


**Figure F.5:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 5: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

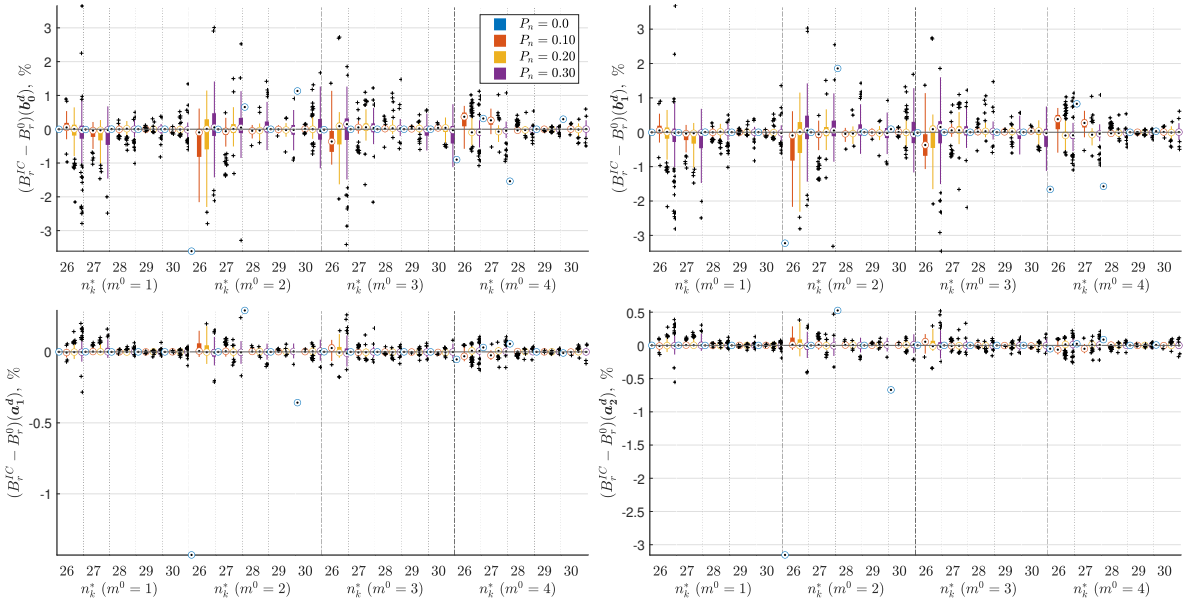


**Figure F.6:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 6: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

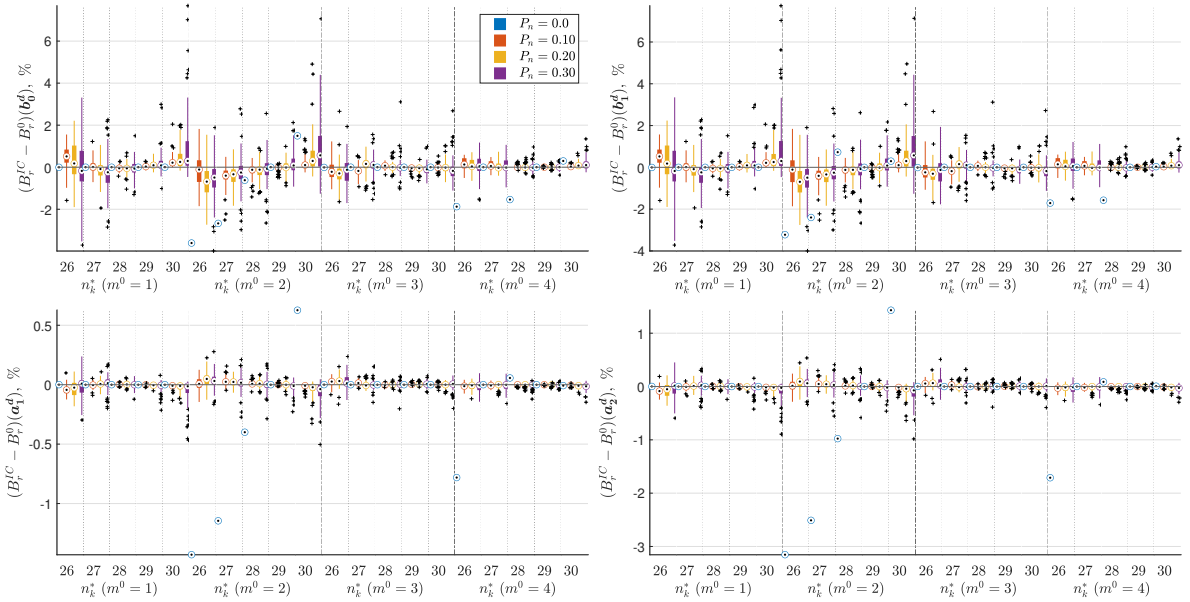


**Figure F.7:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 7: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

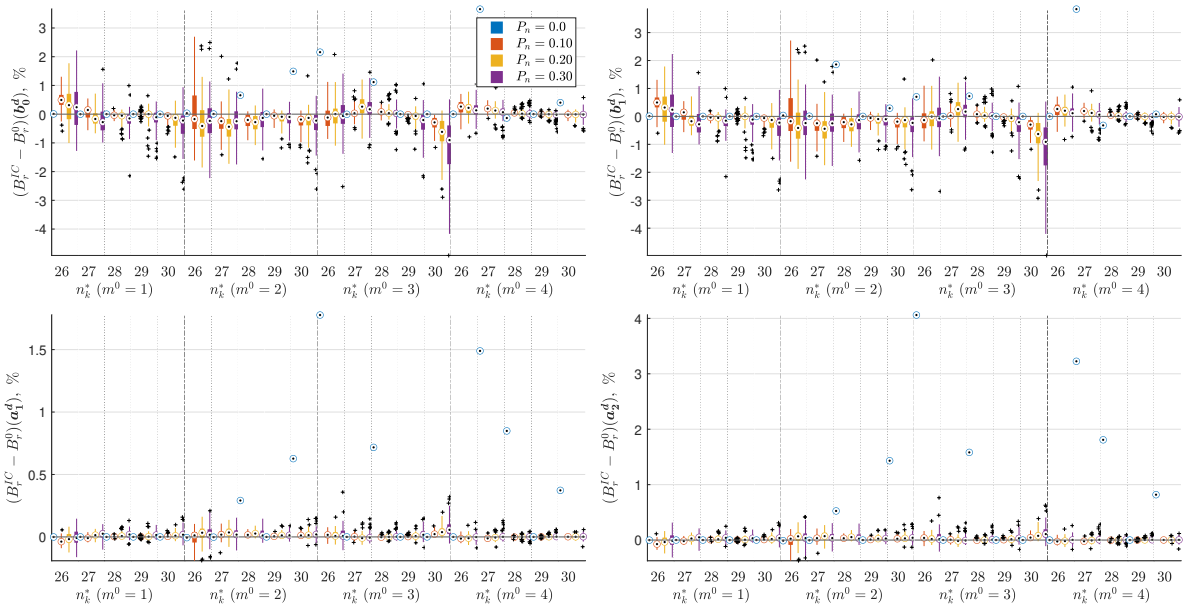


**Figure F.8:** Deviation from reference bias ($B_r^0$) in discrete-time parameters for BJ model with I.C. 8: simulation condition C1, simulation remnant filter orders $m^0 \in \{1, 2, 3, 4\}$, model time-delays $n_k^* \in \{26, 27, 28, 29, 30\}$. Box and Whisker plots made from $M = 100$ realizations.

# G

# Recursive ARX Results

## G.1. Discrete-time parameters

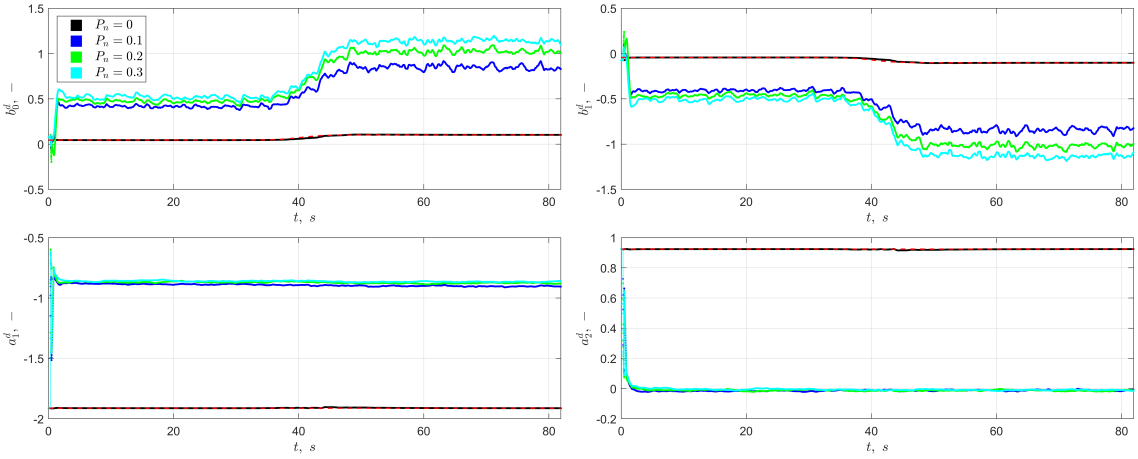


**Figure G.1:** Online estimation results in discrete-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 1$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

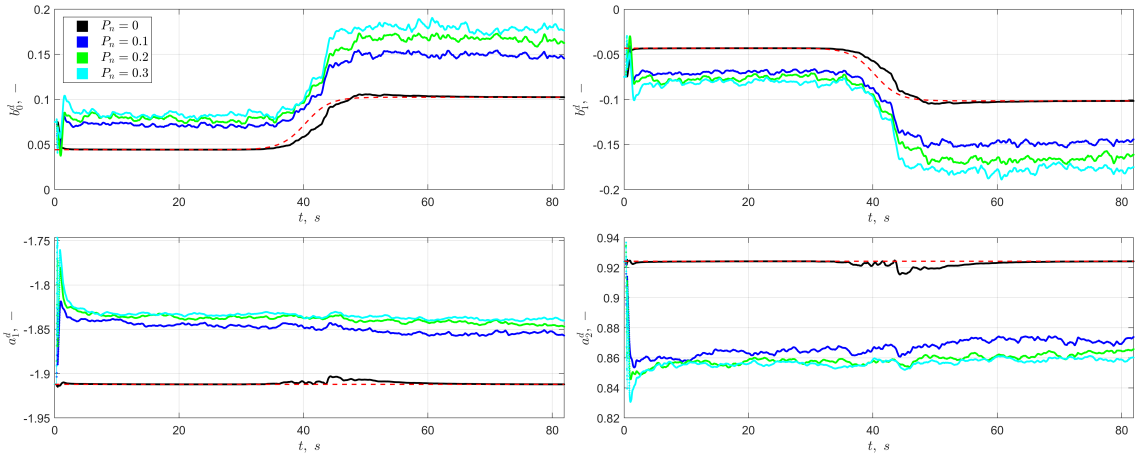


**Figure G.2:** Online estimation results in discrete-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 2$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

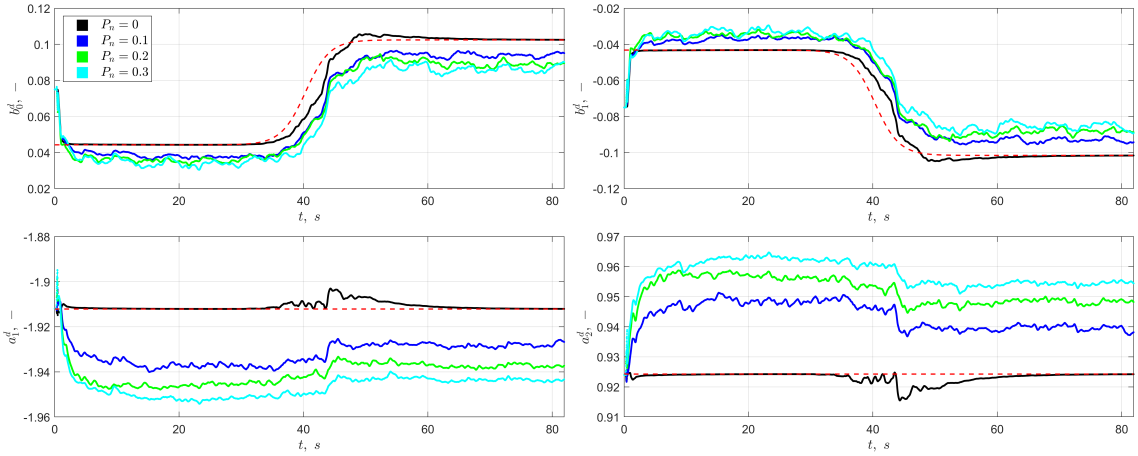


**Figure G.3:** Online estimation results in discrete-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 3$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

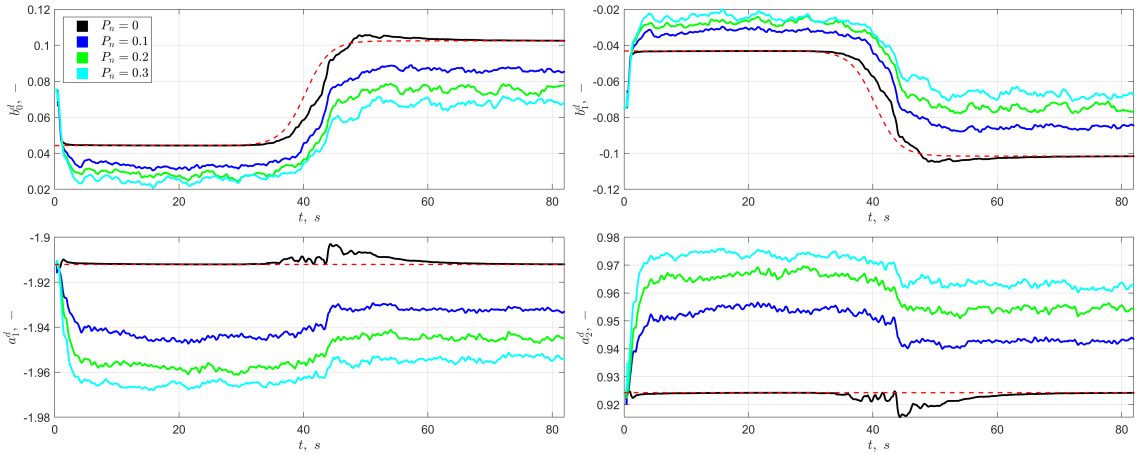


**Figure G.4:** Online estimation results in discrete-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 4$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

## G.2. Continuous-time parameters

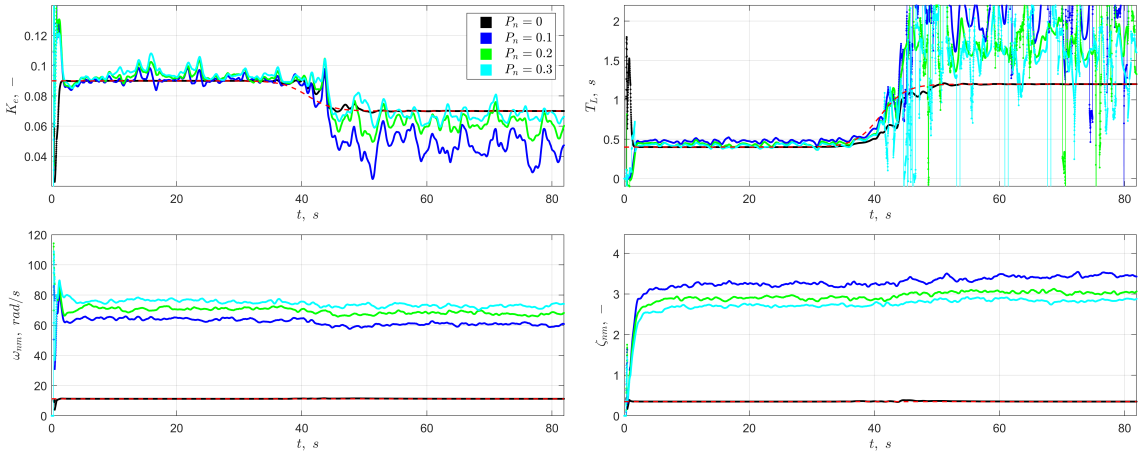


**Figure G.5:** Online estimation results in continuous-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 1$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

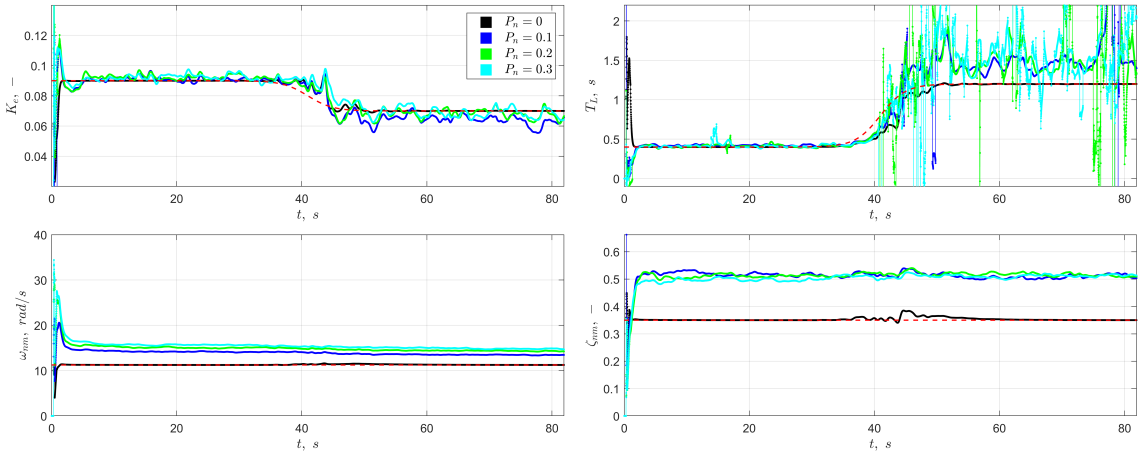


**Figure G.6:** Online estimation results in continuous-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 2$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

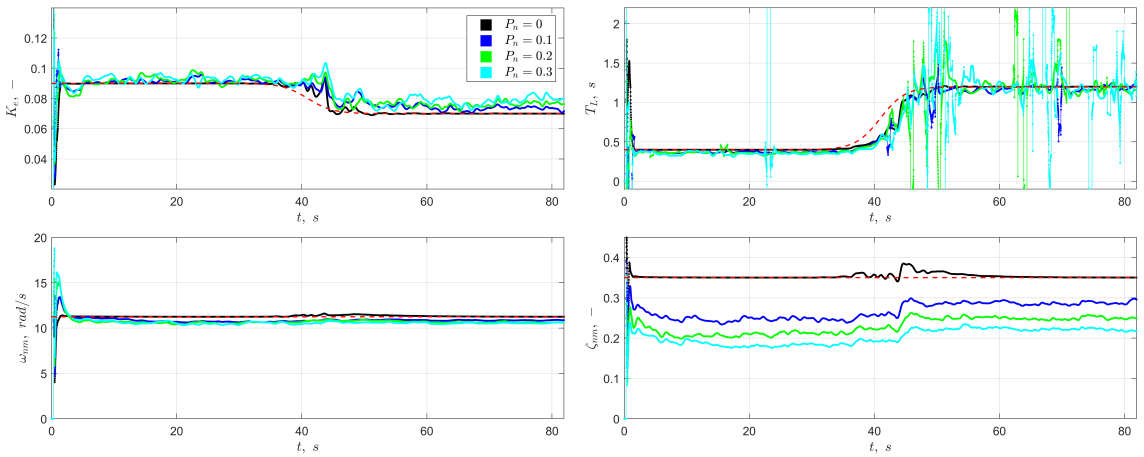


**Figure G.7:** Online estimation results in continuous-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 3$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

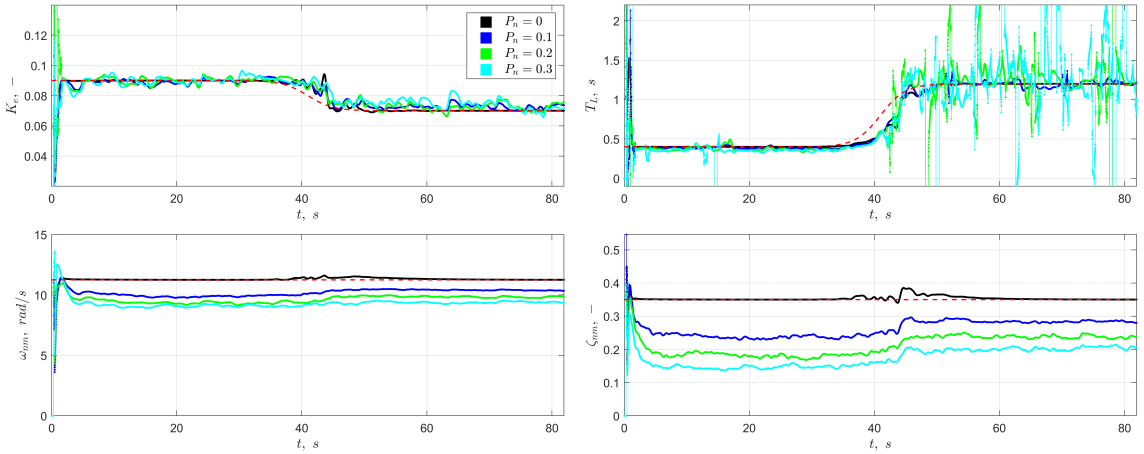


**Figure G.8:** Online estimation results in continuous-time parameters for recursive ARX model: simulation condition C3, simulation remnant filter order $m^0 = 4$, model time-delay $n_k^* = 29$, noise levels $P_n \in \{0.0, 0.10, 0.20, 0.30\}$. Obtained values averaged from $M = 100$ realizations.

# Part III

# Final Report Appendices

# H

# General Simulation Results for Recursive Estimation

## H.1. Simulation Condition C1

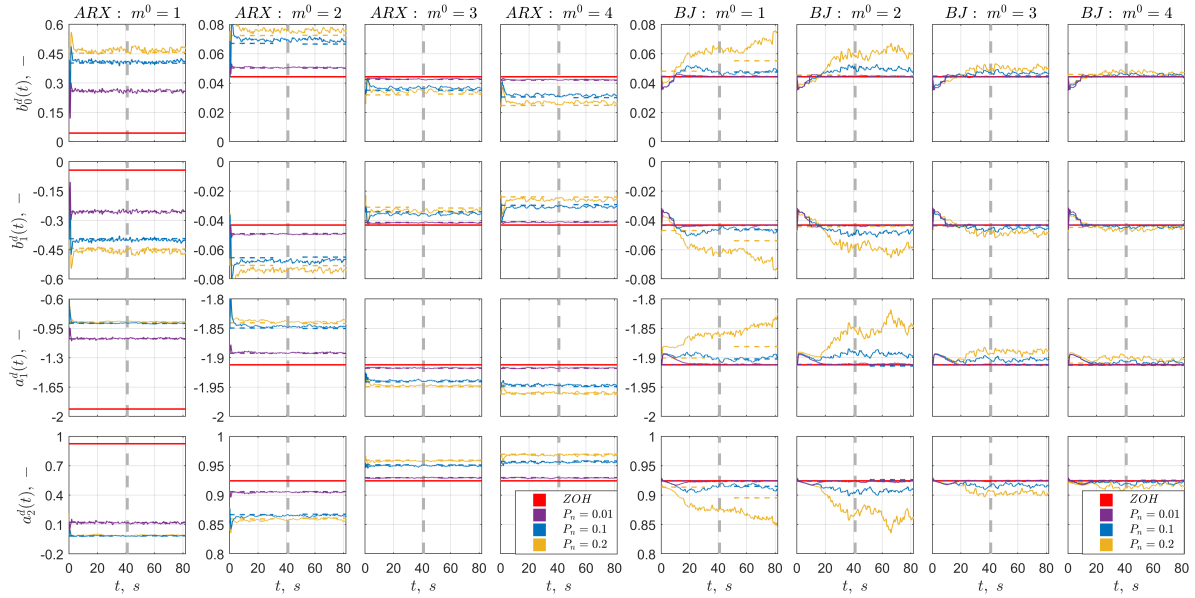


**Figure H.1:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C1, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

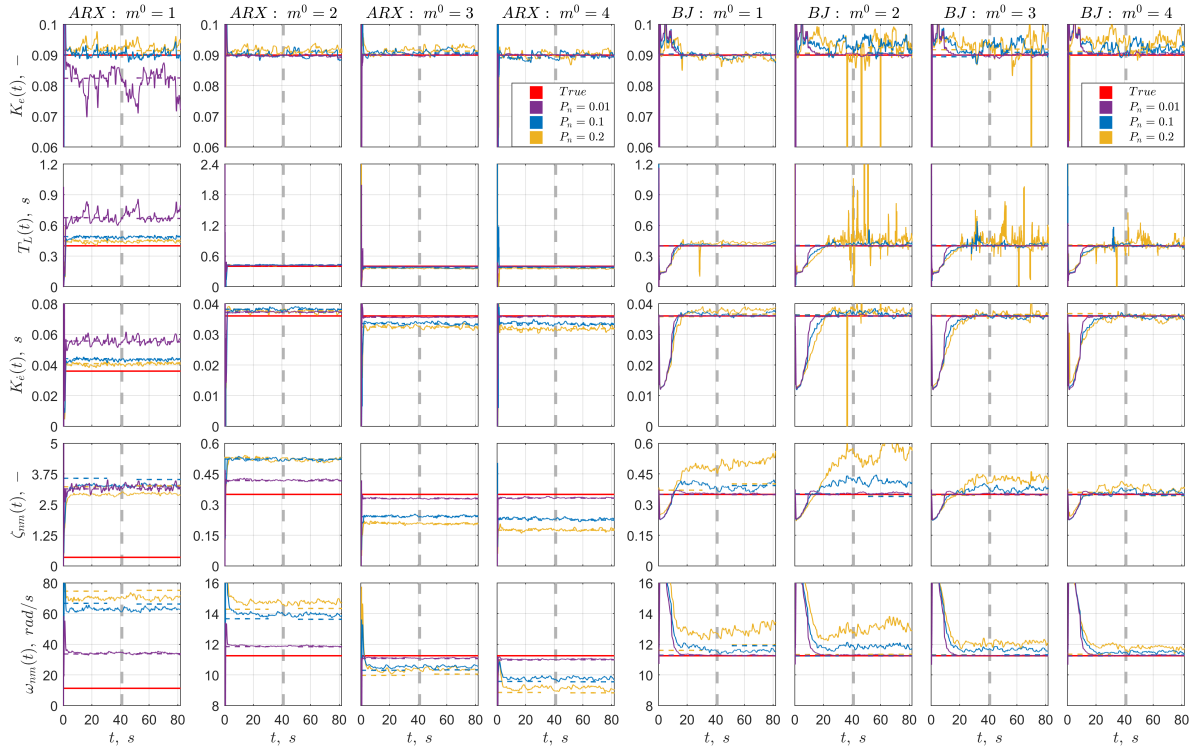


**Figure H.2:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C1, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

# H.2. Simulation Condition C2

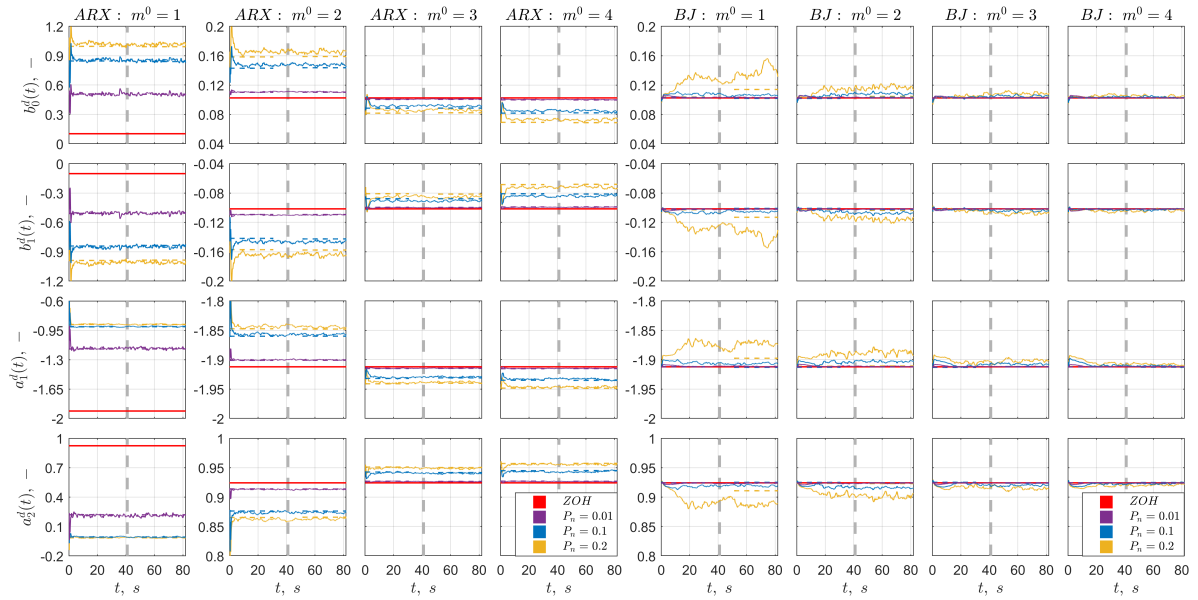


**Figure H.3:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C2, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

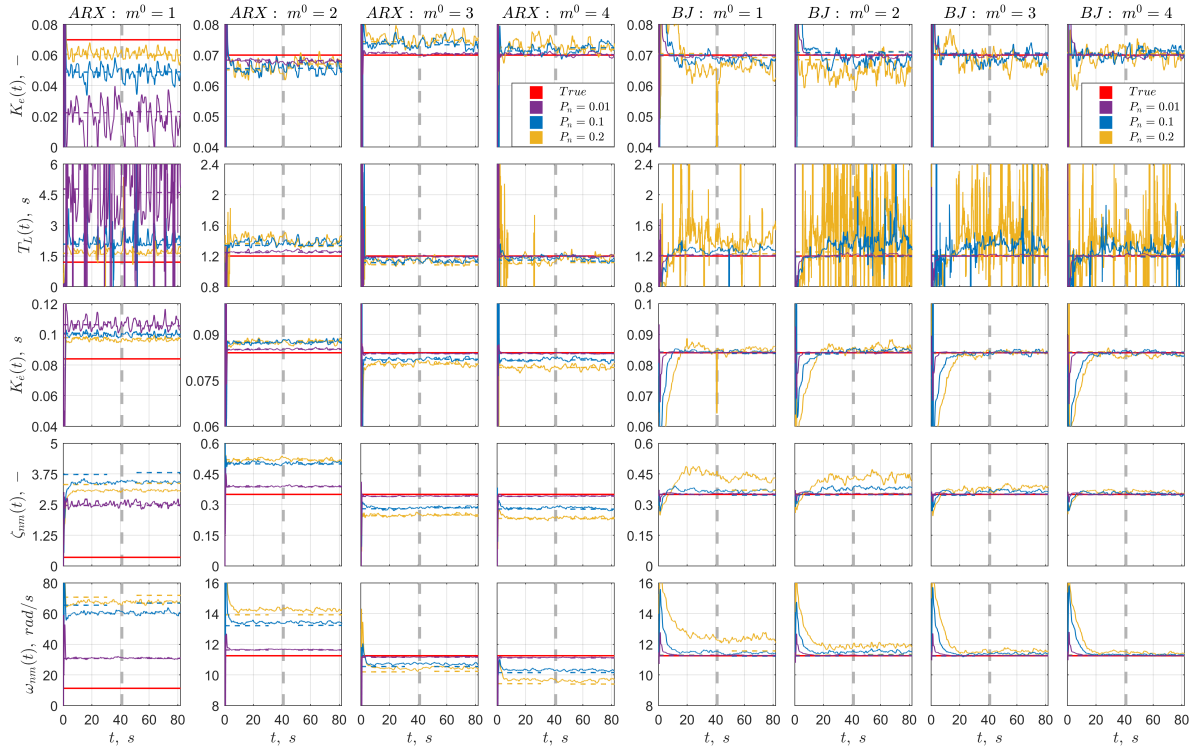


**Figure H.4:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C2, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

# H.3. Simulation Condition C3

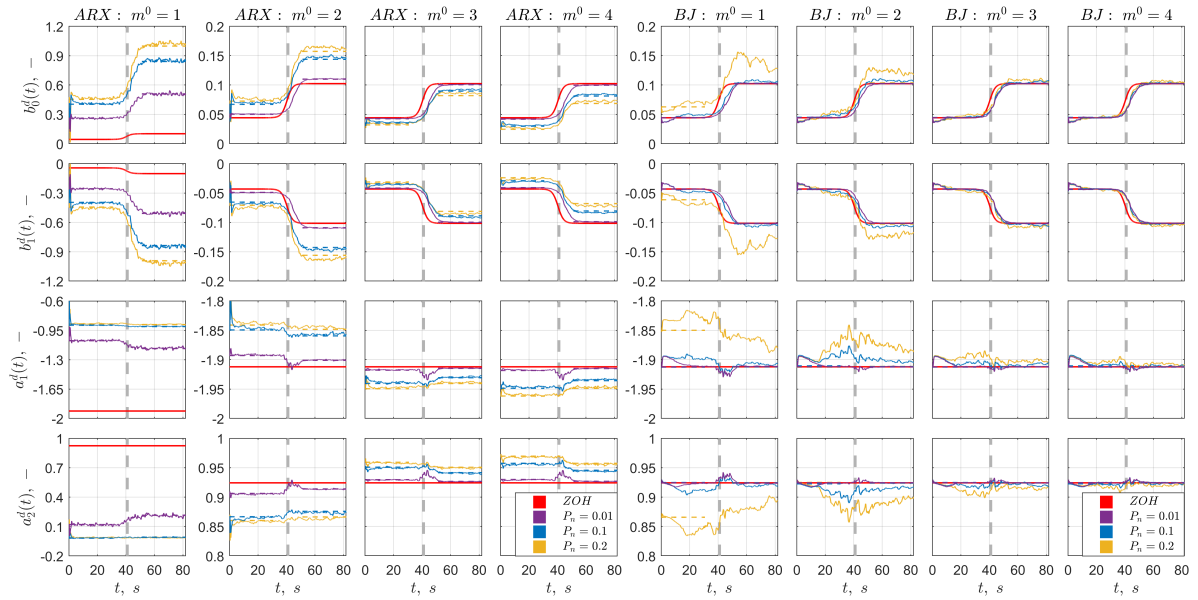


**Figure H.5:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C3, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

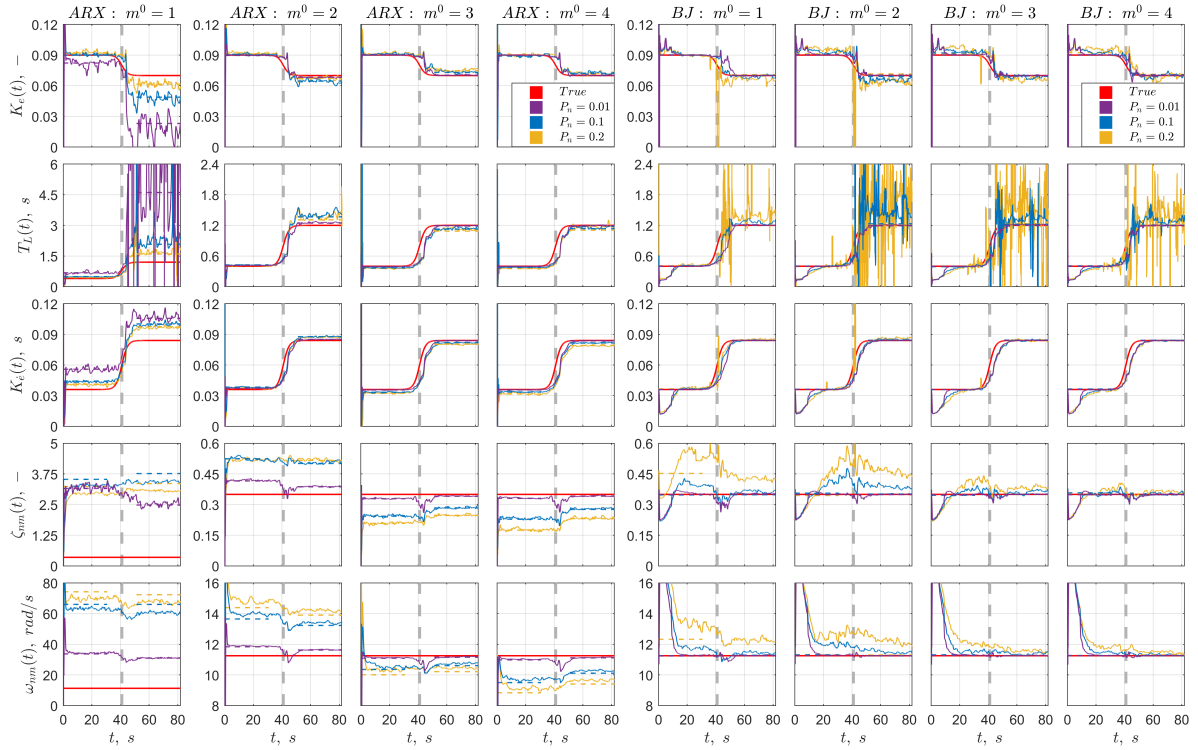


**Figure H.6:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C3, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

# H.4. Simulation Condition C4

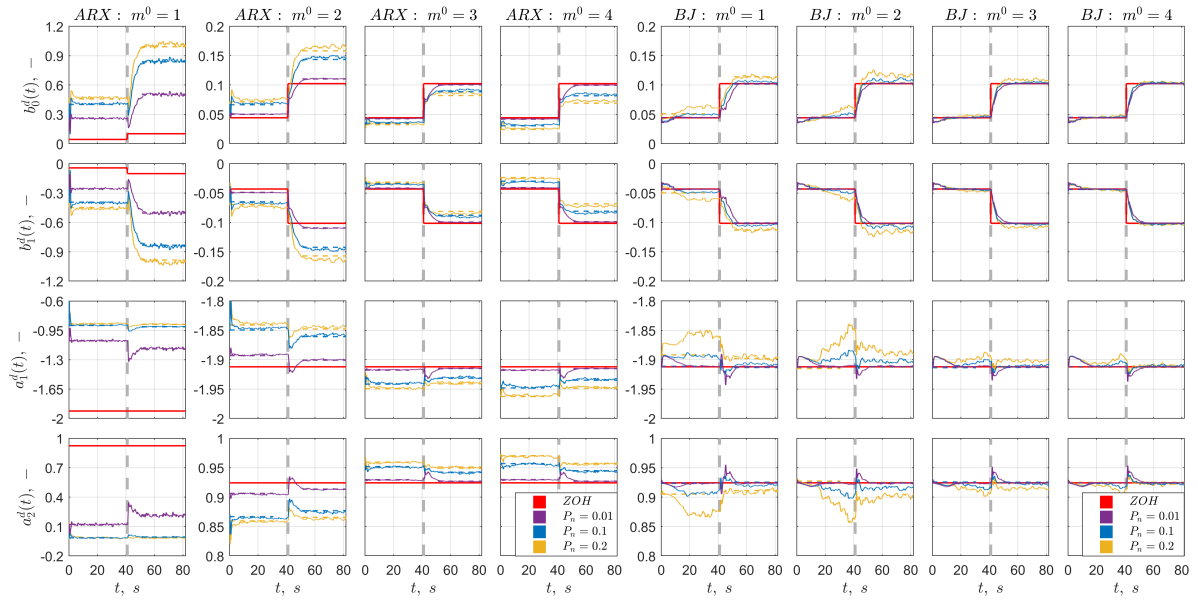


**Figure H.7:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C4, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

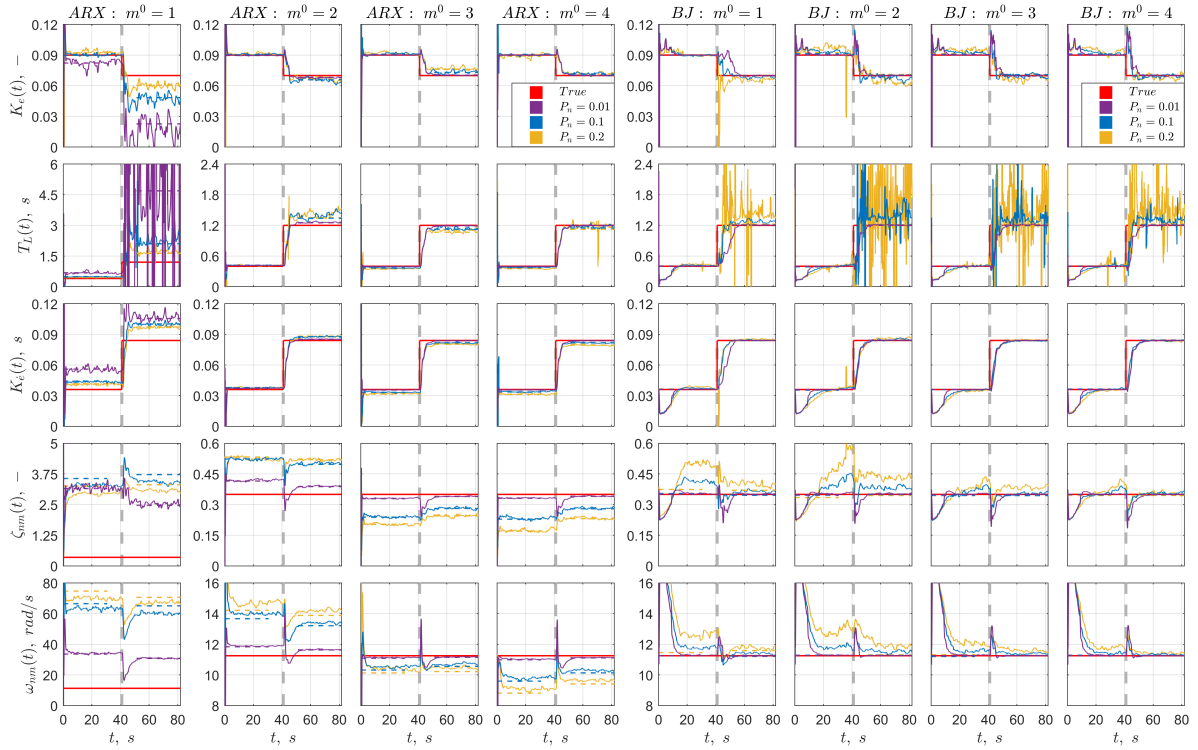


**Figure H.8:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C4, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

# H.5. Simulation Condition C5

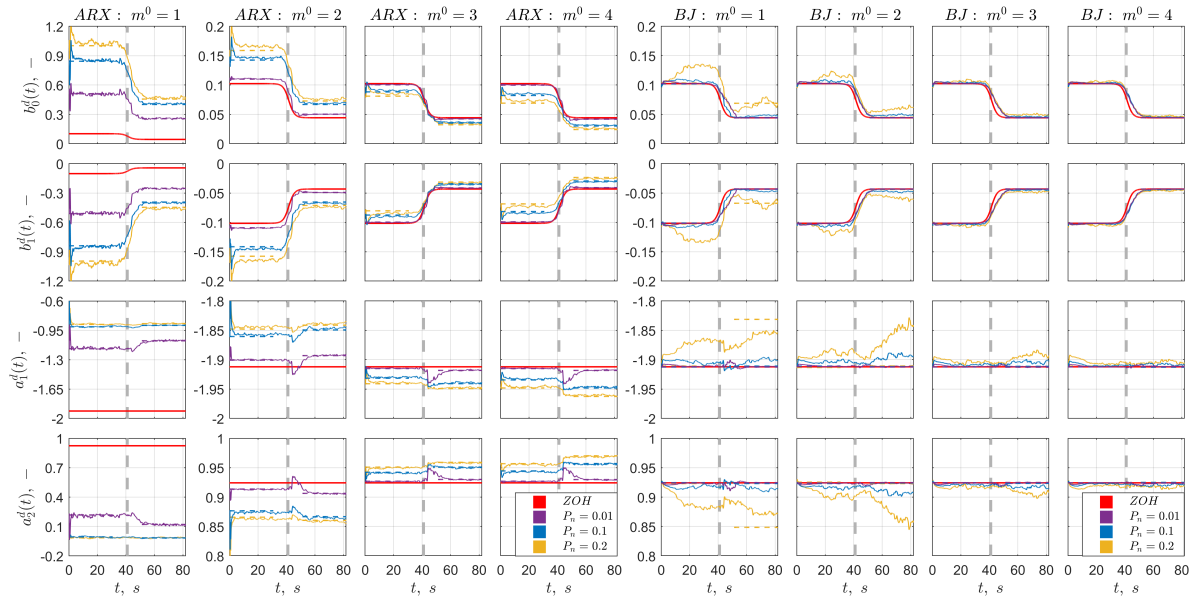


**Figure H.9:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C5, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

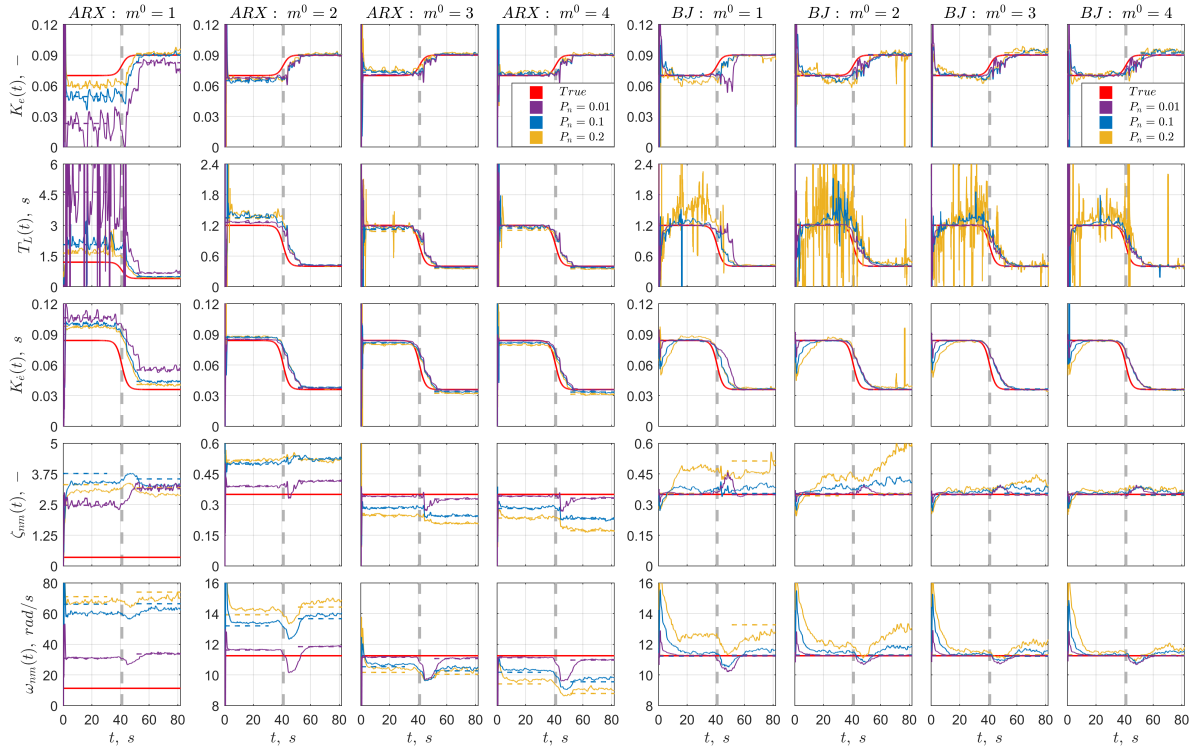


**Figure H.10:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C5, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

# H.6. Simulation Condition C6

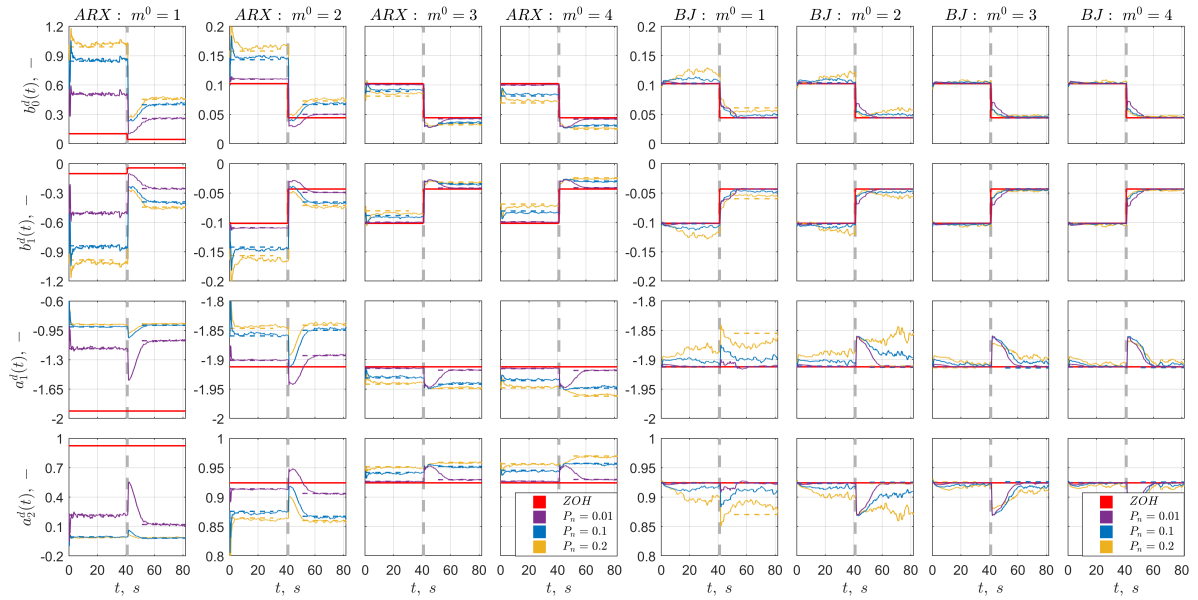


**Figure H.11:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in discrete-time parameters. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C6, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

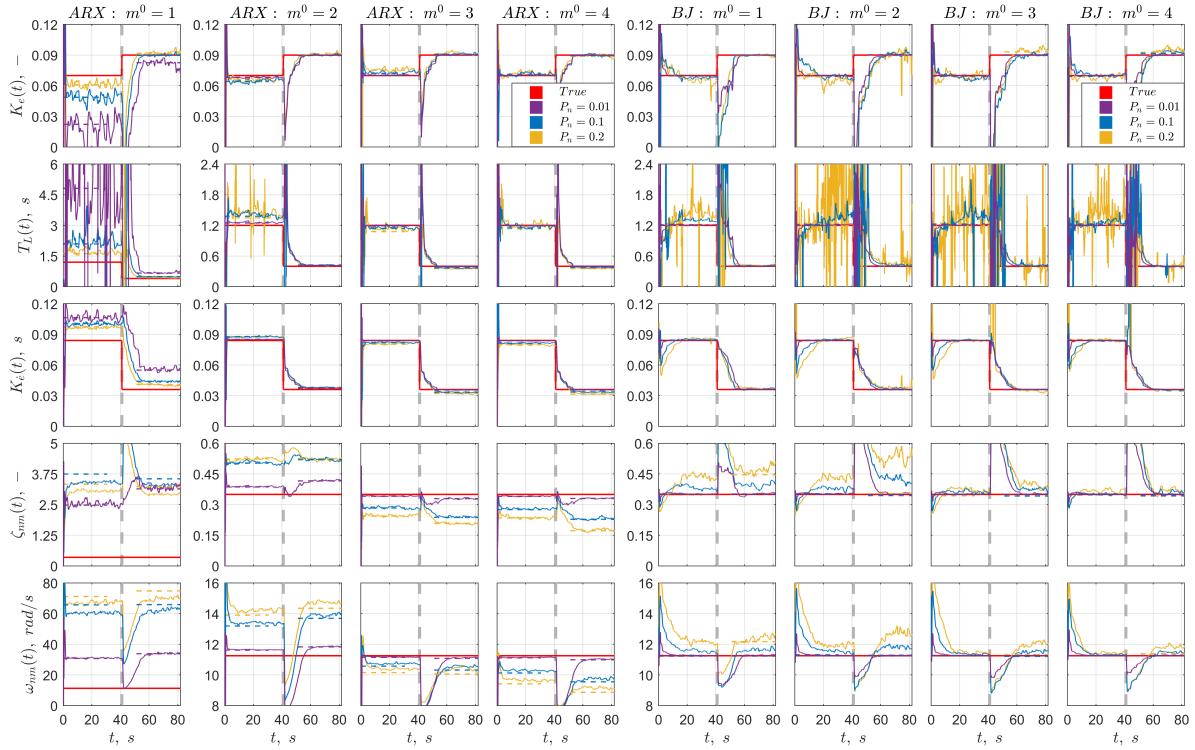


**Figure H.12:** Simulation results of recursive ARX($n_k^* = 29$) and BJ($n_k^* = 29, m^* = 1$) algorithms in HO coefficients. ARX estimations: OLS (dashed line), RLS (continuous line). BJ estimations: PEM (dashed line), RPEM (continuous line). simulation condition C6, simulation remnant filter order $m^0 \in \{1, 2, 3, 4\}$, noise levels $P_n \in \{0.01, 0.10, 0.20\}$. Obtained values averaged from $M = 100$ realizations.

# I

# Experimental Results for Recursive Estimation

## I.1. Simulation Condition C1

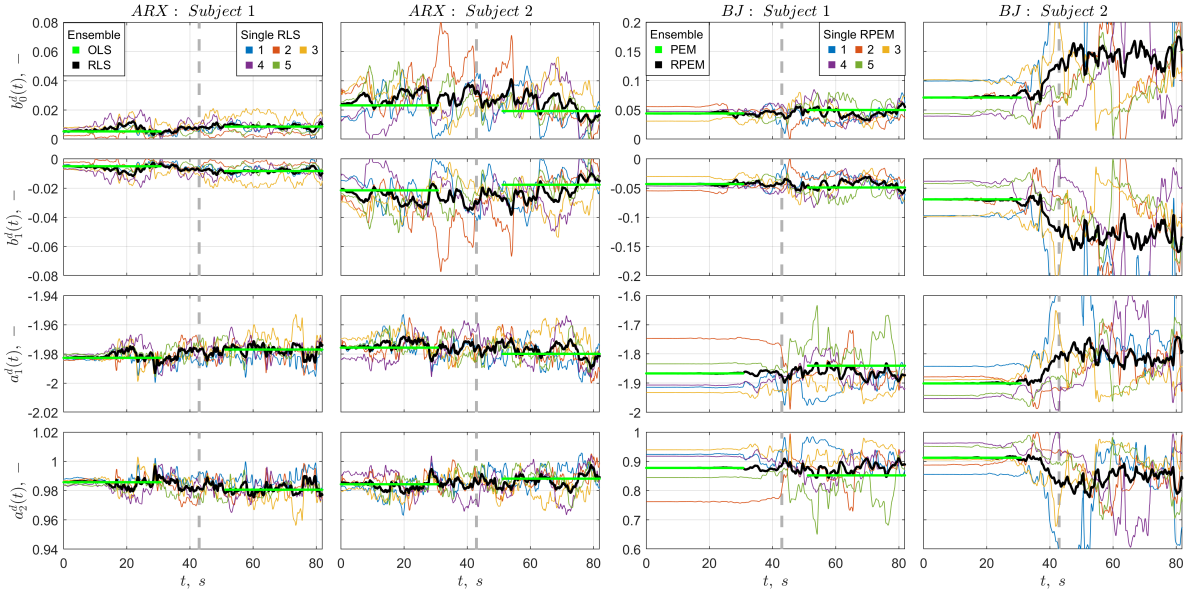


**Figure I.1:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in discrete-time parameters for subjects 1 and 2. Simulation condition C1. Obtained values averaged from 5 runs.


**Figure I.2:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in HO coefficients for subjects 1 and 2. Simulation condition C1. Obtained values averaged from 5 runs.

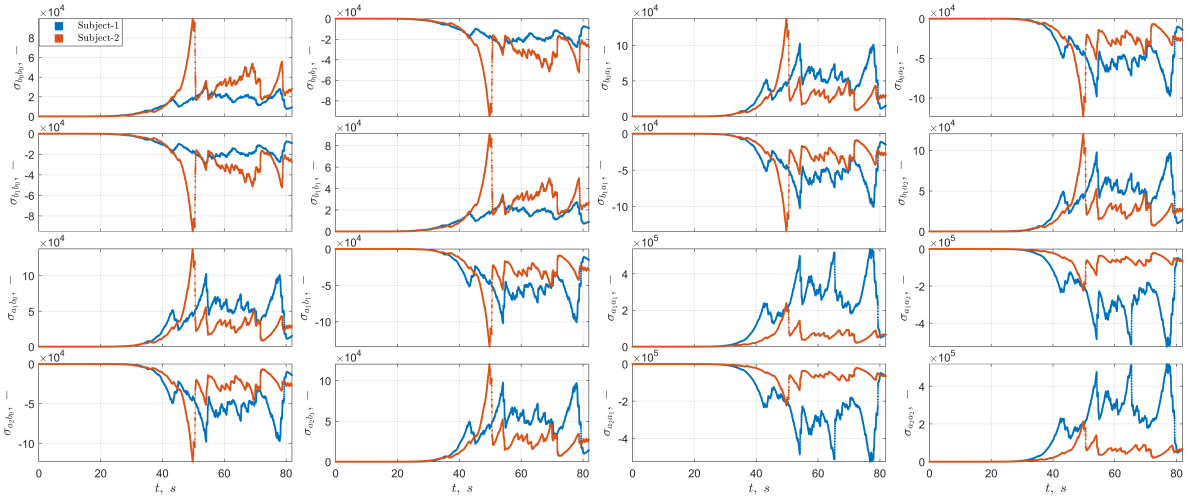


**Figure I.3:** Experimental covariance matrix of RPEM algorithm with BJ($m^* = 1$) for discrete-time parameters for subjects 1 and 2. Simulation condition C1. Obtained values averaged from 5 runs.

# I.2. Simulation Condition C2

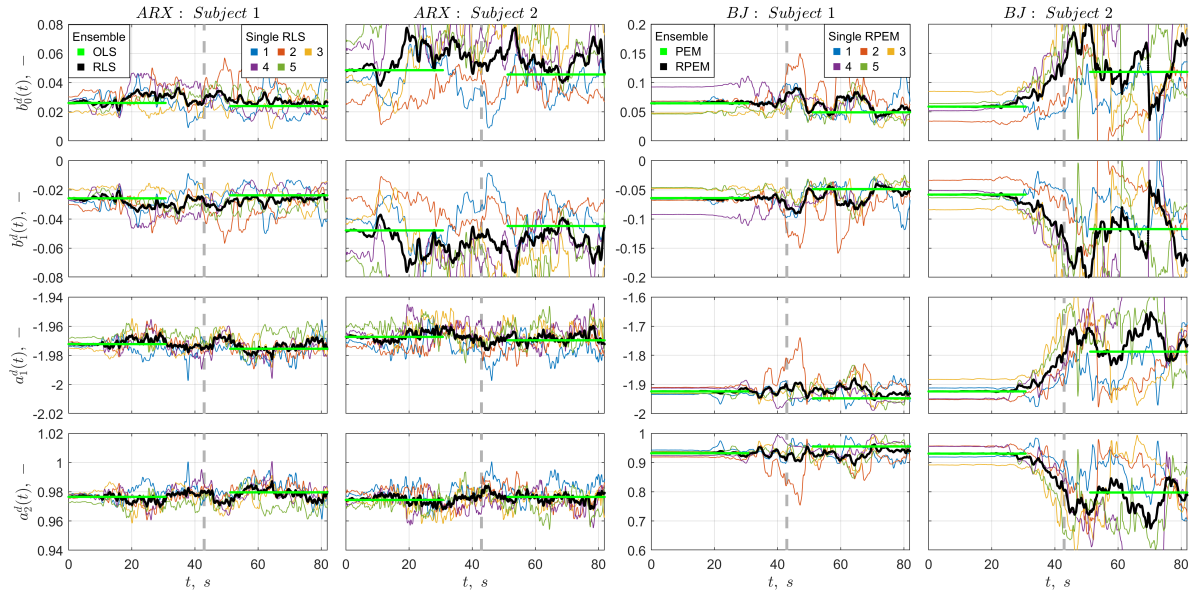


**Figure I.4:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in discrete-time parameters for subjects 1 and 2. Simulation condition C2. Obtained values averaged from 5 runs.

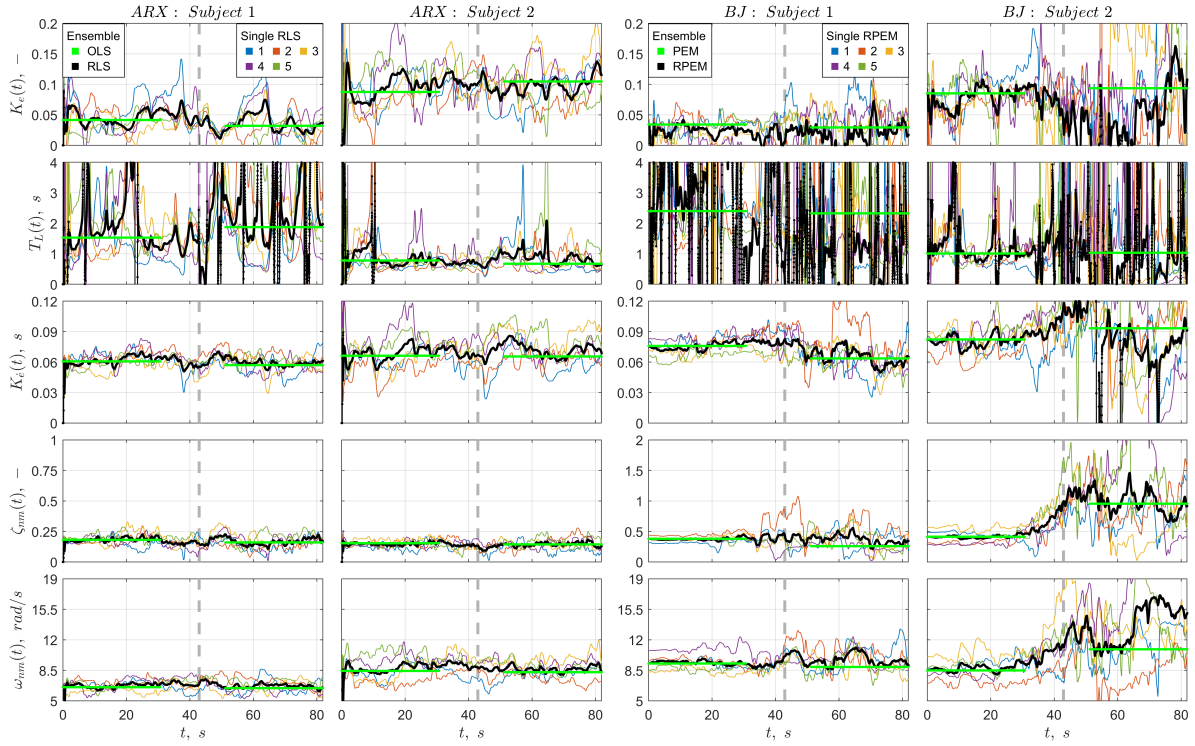


**Figure I.5:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in HO coefficients for subjects 1 and 2. Simulation condition C2. Obtained values averaged from 5 runs.

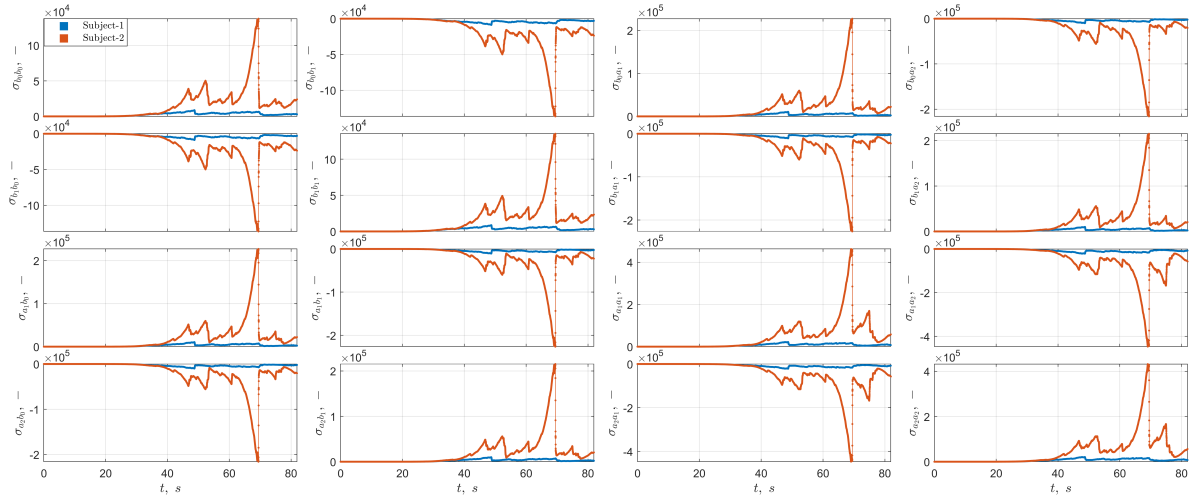


**Figure I.6:** Experimental covariance matrix of RPEM algorithm with BJ($m^* = 1$) for discrete-time parameters for subjects 1 and 2. Simulation condition C2. Obtained values averaged from 5 runs.

# I.3. Simulation Condition C3

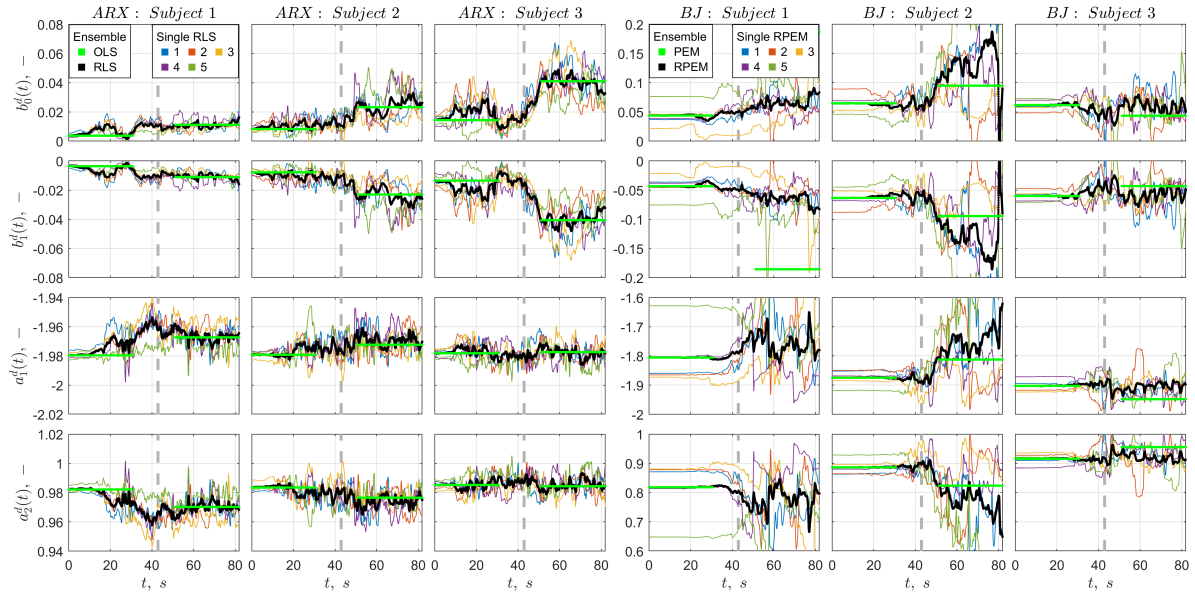


**Figure I.7:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in discrete-time parameters for subjects 1, 2 and 3. Simulation condition C3. Obtained values averaged from 5 runs.

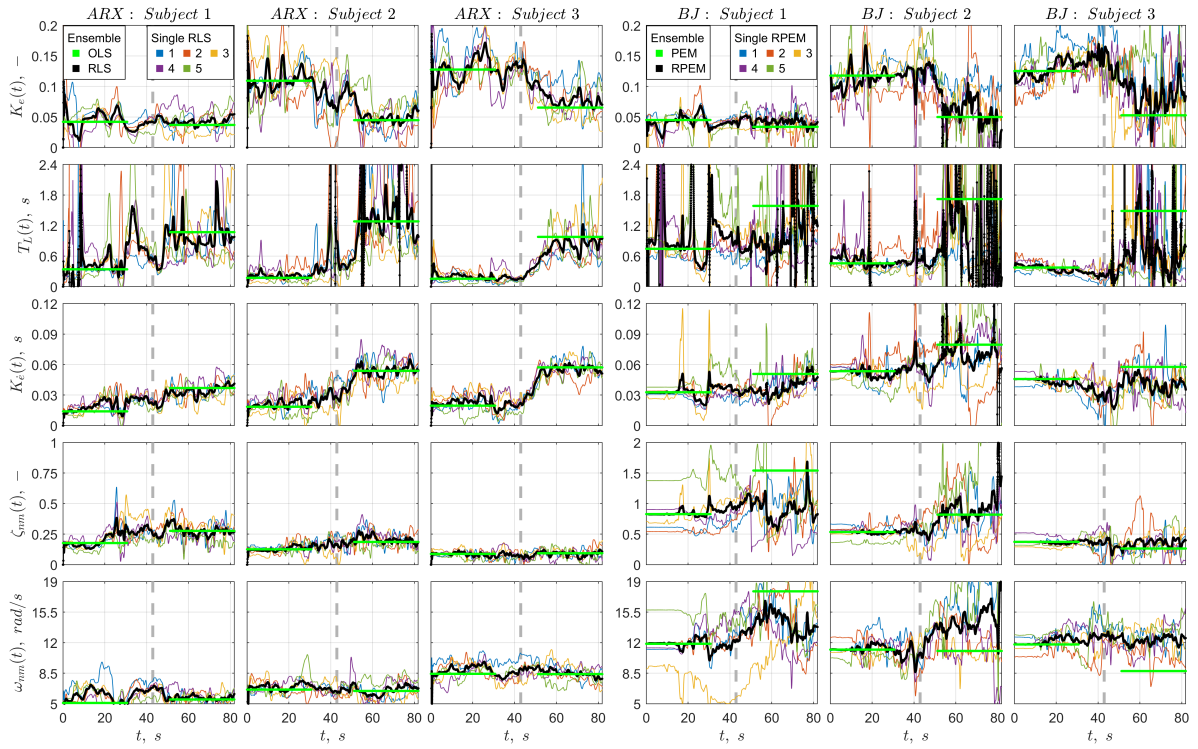


**Figure I.8:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in HO coefficients for subjects 1, 2 and 3. Simulation condition C3. Obtained values averaged from 5 runs.

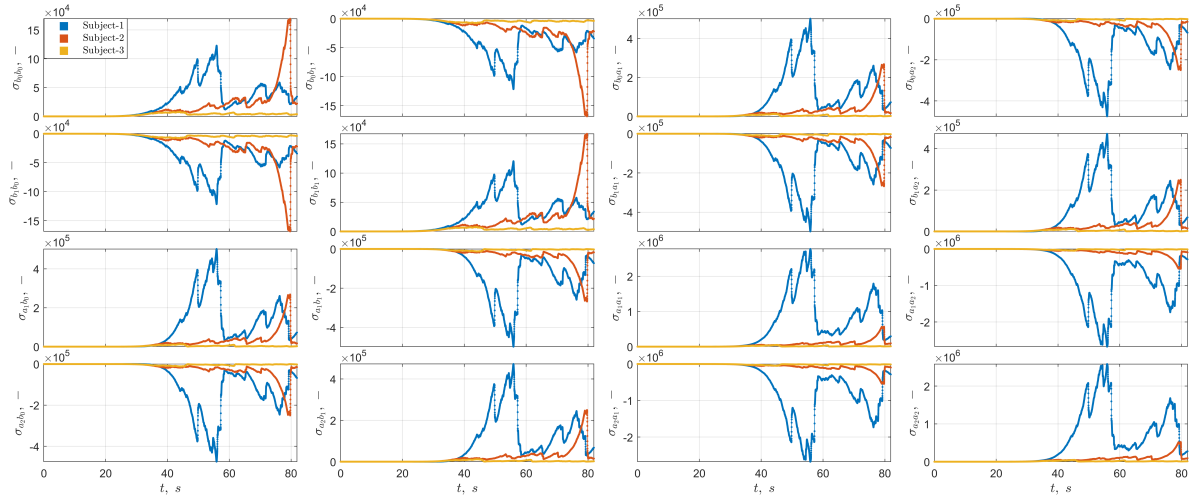


**Figure I.9:** Experimental covariance matrix of RPEM algorithm with BJ($m^* = 1$) for discrete-time parameters for subjects 1, 2 and 3. Simulation condition C3. Obtained values averaged from 5 runs.

# I.4. Simulation Condition C4

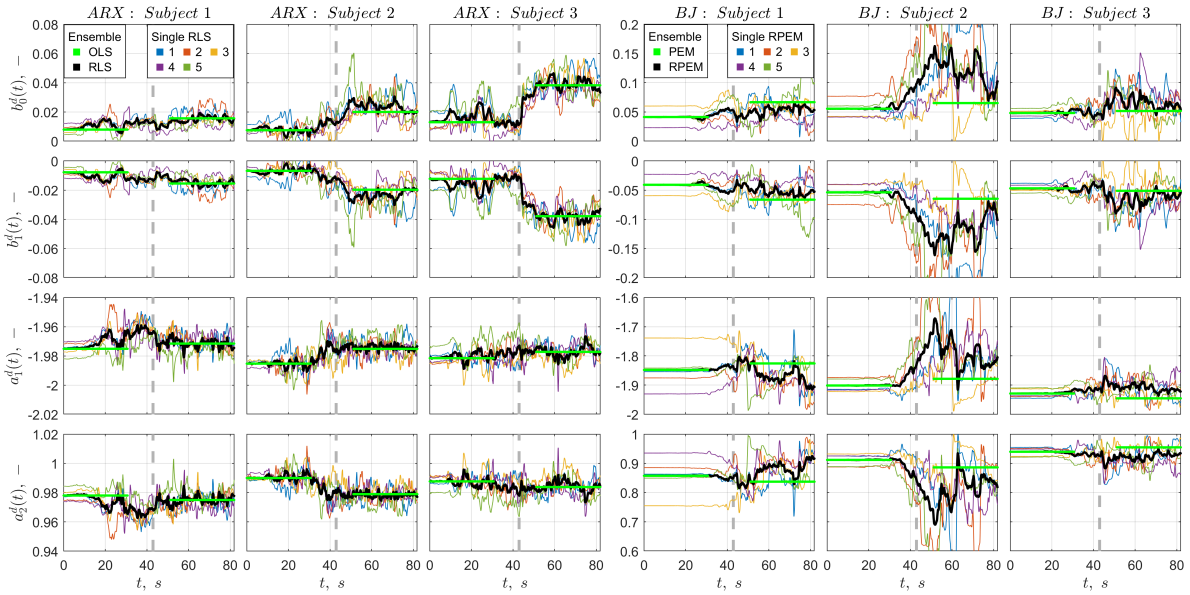


**Figure I.10:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in discrete-time parameters for subjects 1, 2 and 3. Simulation condition C4. Obtained values averaged from 5 runs.

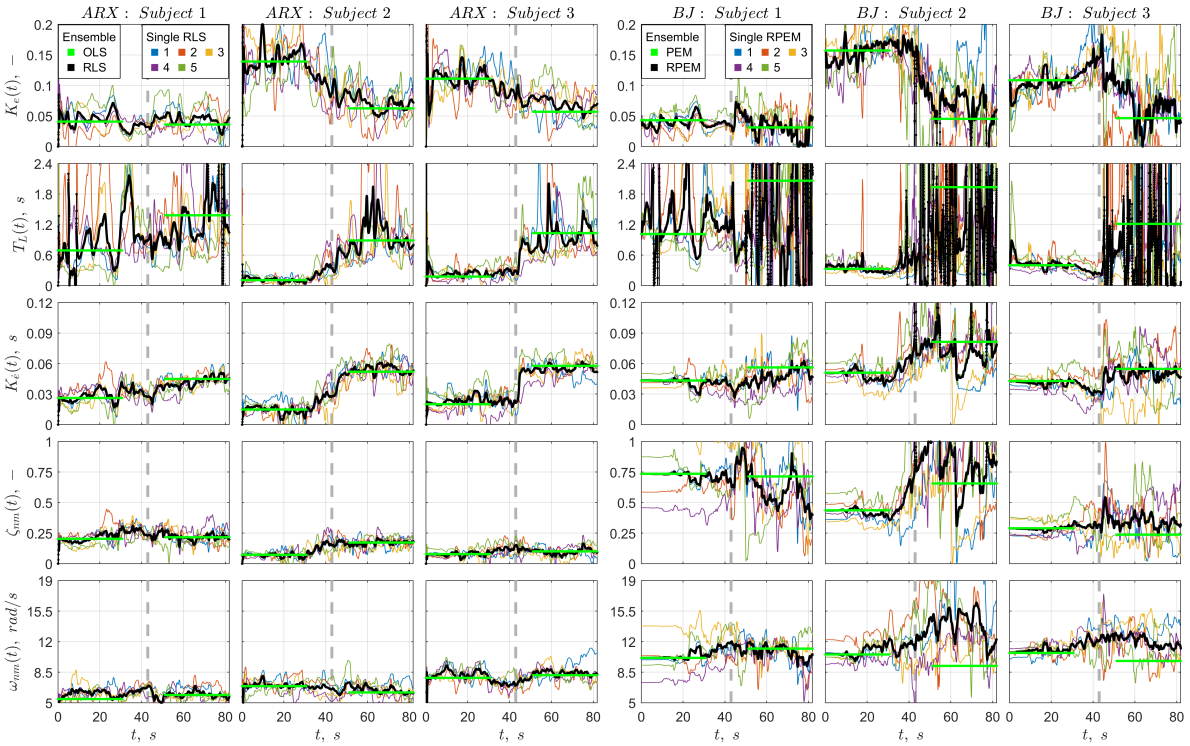


**Figure I.11:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in HO coefficients for subjects 1, 2 and 3. Simulation condition C4. Obtained values averaged from 5 runs.

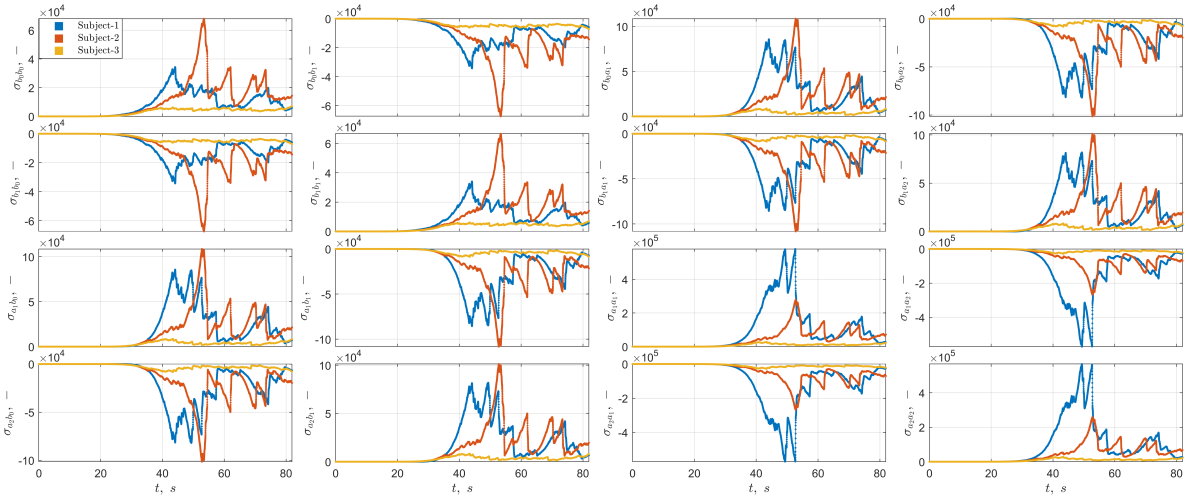


**Figure I.12:** Experimental covariance matrix of RPEM algorithm with BJ($m^* = 1$) for discrete-time parameters for subjects 1, 2 and 3. Simulation condition C4. Obtained values averaged from 5 runs.

# I.5. Simulation Condition C5

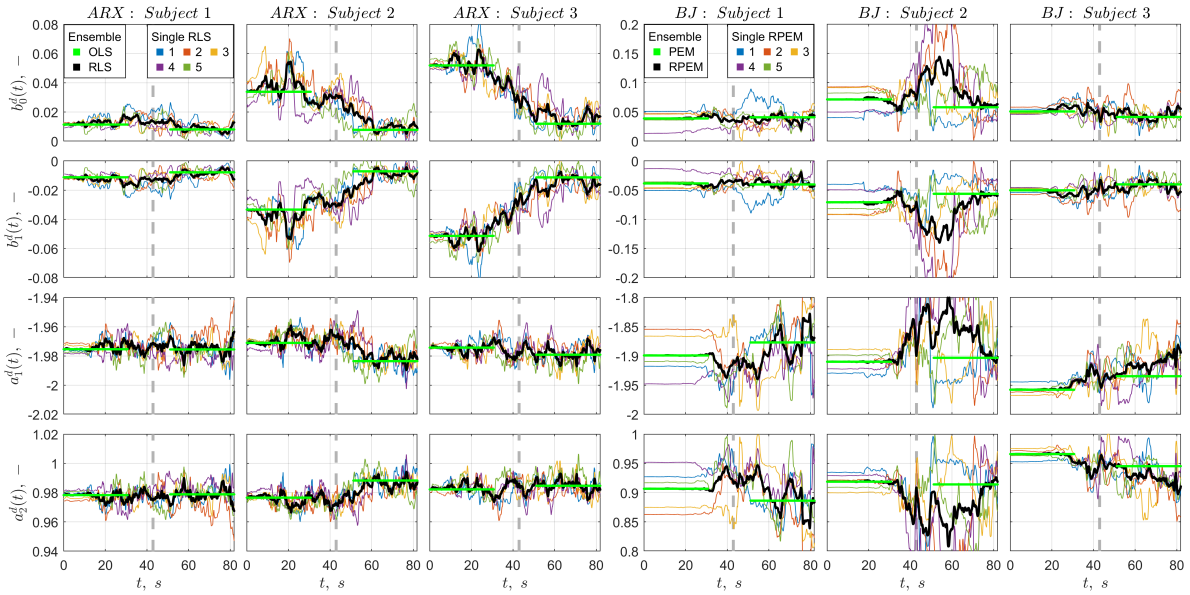


**Figure I.13:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in discrete-time parameters for subjects 1, 2 and 3. Simulation condition C5. Obtained values averaged from 5 runs.

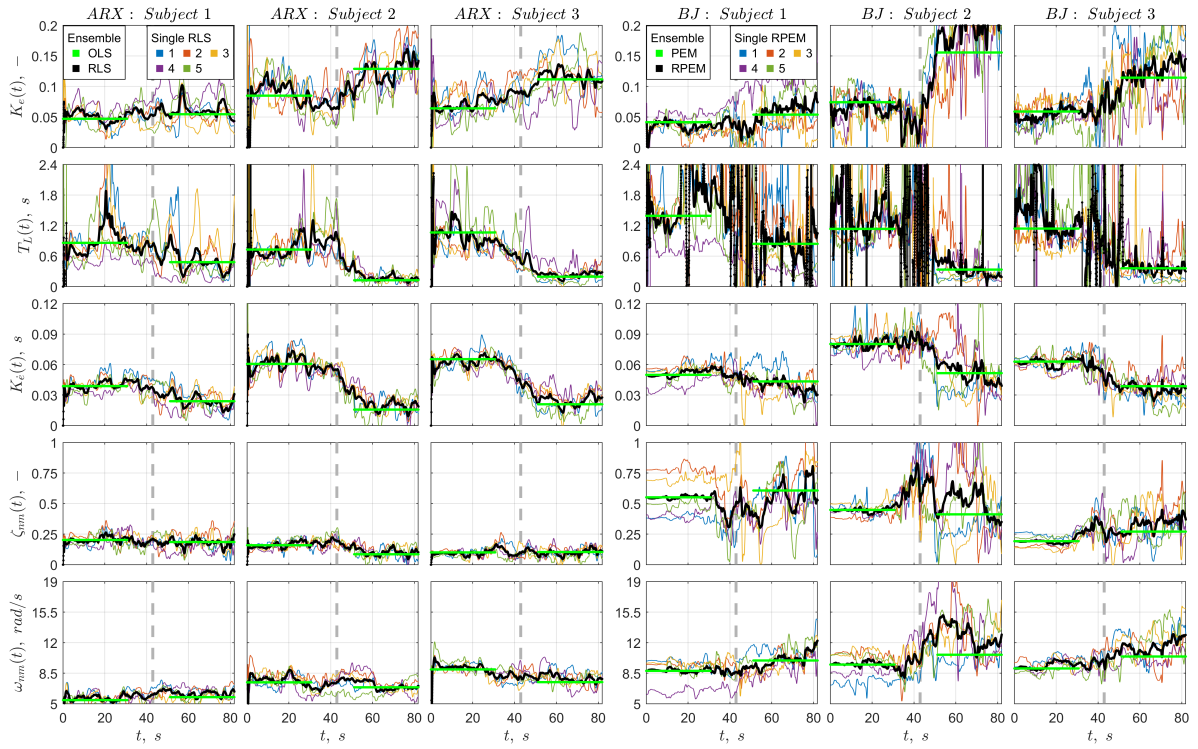


**Figure I.14:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in HO coefficients for subjects 1, 2 and 3. Simulation condition C5. Obtained values averaged from 5 runs.

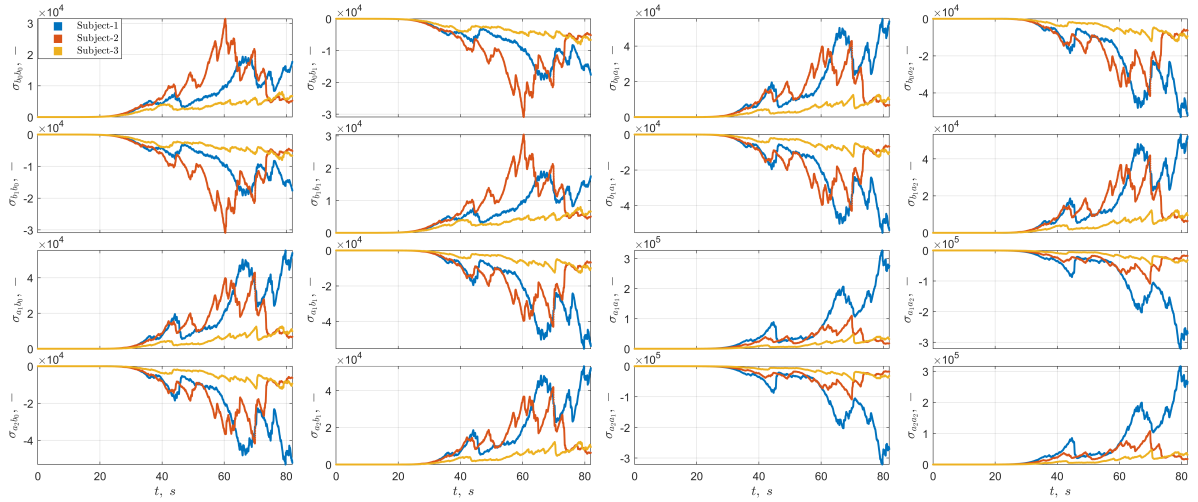


**Figure I.15:** Experimental covariance matrix of RPEM algorithm with BJ($m^* = 1$) for discrete-time parameters for subjects 1, 2 and 3. Simulation condition C5. Obtained values averaged from 5 runs.

# I.6. Simulation Condition C6

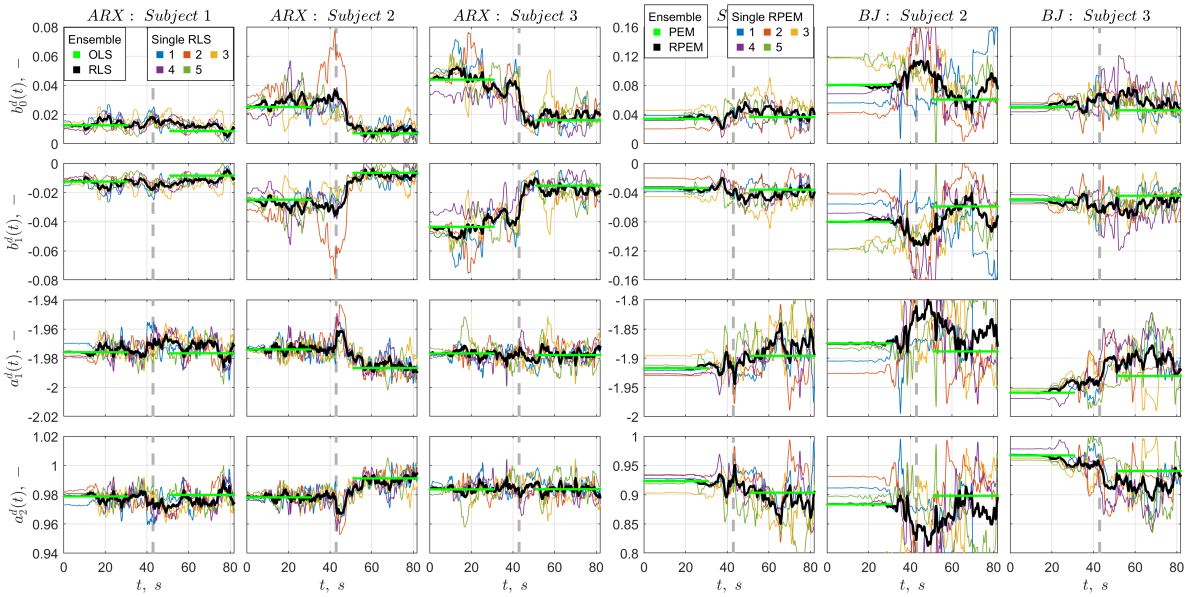


**Figure I.16:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in discrete-time parameters for subjects 1, 2 and 3. Simulation condition C6. Obtained values averaged from 5 runs.


**Figure I.17:** Experimental results of recursive ARX and BJ($m^* = 1$) algorithms in HO coefficients for subjects 1, 2 and 3. Simulation condition C6. Obtained values averaged from 5 runs.

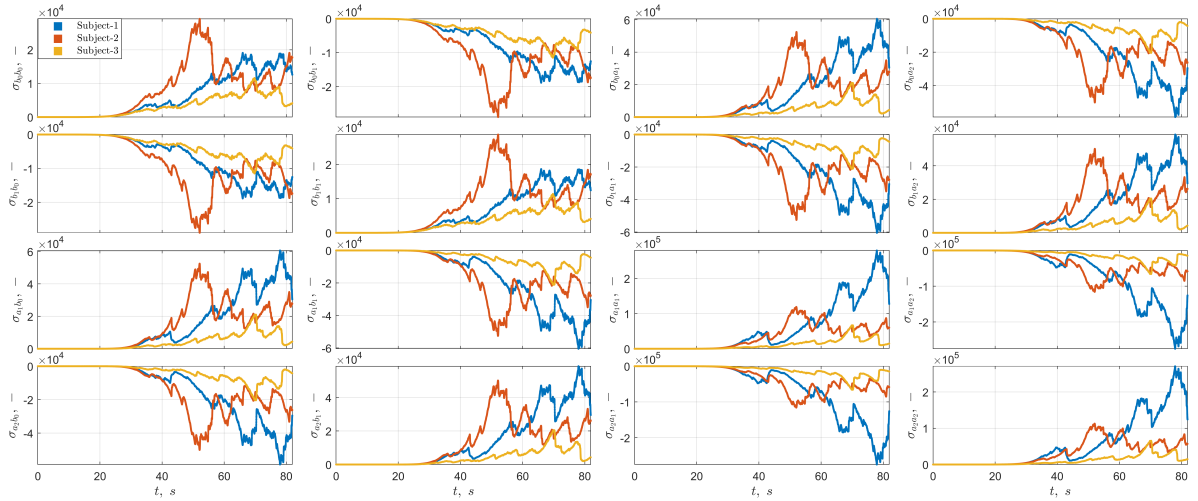


**Figure I.18:** Experimental covariance matrix of RPEM algorithm with BJ($m^* = 1$) for discrete-time parameters for subjects 1, 2 and 3. Simulation condition C6. Obtained values averaged from 5 runs.