Online Time-Varying Identification and Detection of Operator Adaptation with Recursive ARX

Graduation Report

W. Plaetinck

TU Delft

Online Time-Varying Identification and Detection of Operator Adaptation with Recursive ARX

Master of Science Thesis

by

W. Plaetinck

to obtain the degree of
Master of Science in Aerospace Engineering at Delft University of Technology,
to be defended publicly on Wednesday January 17, 2018 at 2:00 PM.

Student number: 4219724
Supervisors: Dr.ir. D.M. Pool
Prof.dr.ir. M. Mulder
Dr.ir. M.M. van Paassen

Faculty of Aerospace Engineering
Section of Control and Simulation
Delft, The Netherlands

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: **Online Time-Varying Identification and Detection of Operator Adaptation with Recursive ARX**, submitted by **W. Plaetinck**, in partial fulfilment of the requirements for the award of the degree of **Master of Science**.

Dated: _____

Assessment committee:

Professor:

_____

Prof. dr. ir. M. Mulder

Supervisors:

_____

Dr. ir. D. M. Pool

_____

Dr. ir. M. M. van Paassen

External examiner:

_____

Dr. ir. D. A. Abbink

Preface

This graduation report covers my thesis work for obtaining my Master of Science (MSc) degree in Aerospace Engineering at Delft University of Technology. In Part I a scientific paper is included covering the main results. It focuses on online time-varying identification and detection of operator adaptation in manual control tasks. Part II contains the preliminary graduation report covering a literature study in time-varying identification methods in manual control and an investigation on identifying operator delay online. Part III contains appendices with additional results and supporting material. Both Part I and III are meant for the final thesis work AE5310, part II was already graded for the literature study AE4020.

Throughout the thesis, the research objective changed from proving a method's usability in simulation to an experimental evaluation in the HMI lab. This gave me insight in both types of research, for which I am glad. I am proud I succeeded in mastering the real-time simulation environment Dueca to add my own module in C++, a rather new programming language for me.

This would not have been possible without the excellent supervision by Daan, who offered me flexibility in my work and often reassured me about the track we were following. Thank you to Max and Rene for the enthusiasm and interest in my work, this was motivating at crucial points.

Now it is time to challenge the future. Thank you TU Delft.

W. Plaetinck
Delft, January 2018

Cover image: Betto Rodrigues

Contents

List of Figures

List of Tables

List of Symbols

$\boldsymbol{I}$	Unit matrix	-
$\boldsymbol{u}$	Vector of simulated control output values	-
Δt	Timestep	s
ϵ	Noise signal	-
$\hat{\theta}_0$	Initial coefficient vector for recursive least squares	-
$\hat{\theta}_i$	Estimated coefficient vector at timestep i	-
$\hat{\theta}_n^{OLS}$	Estimated coefficient vector using n samples	-
$\hat{u}$	One-step-ahead prediction of control signal	-
λ	Forgetting factor	-
ω	Frequency	rad/s
ω_b	Controlled element break frequency	rad/s
ω_c	Human operator crossover frequency	rad/s
ω_m	Measurement base frequency	rad/s
ω_{nm}	Neuromuscular natural frequency	rad/s
ω_n	Remnant filter natural frequency	rad/s
Φ	Regression matrix	-
ϕ_f	Sinusoid phase	rad
σ_n^2	Remnant signal variance	deg^2
σ_u^2	Control signal variance	deg^2
τ	Human operator time delay	s
τ_L	Lead time constant	s
τ_l	Lag time constant	s
τ_{est_i}	Estimated time delay at timestep i	s
θ	Coefficient vector	-
θ_c	Continuous time coefficient vector	-
θ_d	Discrete time coefficient vector	-
φ	Regression vector	-
φ_k	Regression vector of kth ARX model	-
ζ_n	Remnant filter damping constant	-
ζ_{nm}	Neuromuscular damping constant	-

A_c	Human operator state space A matrix in continuous time	-
A_d	Human operator state space A matrix in discrete time	-
A_f	Sinusoid amplitude	deg
B_c	Human operator state space B matrix in continuous time	-
B_d	Human operator state space B matrix in discrete time	-
C_c	Human operator state space C matrix in continuous time	-
d	Decimation factor	-
e	Tracking error signal	deg
E_c	Controlled element state space E matrix in continuous time	-
E_d	Controlled element state space E matrix in discrete time	-
f	Forcing function	deg
F_c	Controlled element state space F matrix in continuous time	-
F_d	Controlled element state space F matrix in discrete time	-
G	Sigmoid maximum rate of change	s^{-1}
$H(q)$	Difference equation from error to control signal	-
H_c	Controlled element state space H matrix in continuous time	-
$H_n(j\omega)$	Remnant filter	-
$H_P(j\omega)$	Human operator transfer function	-
$H_P(s)$	Continuous time human operator transfer function	-
$H_p(s)$	Continuous time human operator transfer function	-
$H_p(z)$	Discrete time human operator transfer function	-
$H_{CE}(j\omega)$	Controlled element transfer function	-
$H_{OL}(j\omega)$	Human operator open loop transfer function	-
i	Timestep index i	-
i_{start}	Start index for least squares	-
i_w	Sliding window size	-
K	Number of parallel ARX models	-
k	kth ARX model	-
K_c	Controlled element gain	-
K_e	Error gain	-
k_f	kth sinusoid in forcing function	-
K_i	Recursive least squares gain at timestep i	-
K_n	Remnant gain	-
K_p	Visual gain	-

$K_{\dot{e}}$	Error rate gain	-
M	Sigmoid time of maximum rate of change	s
m	Remnant filter order	-
m_X	Number of columns of matrix X	-
N	Number of samples	-
n	Number of samples used in ordinary least squares	-
n	Remnant signal	deg
n_a	Number of coefficients in A polynomial	-
n_b	Number of coefficients in B polynomial	-
N_f	Number of sinusoids in forcing function	-
n_f	Sinusoid base frequency multiplier	-
n_k	ARX model delay parameter	-
n_{k_a}	Array with delay parameters	-
$n_{k_{est_i}}$	Estimated ARX delay parameter at timestep i	-
$n_{k_{est}}$	Estimated ARX delay parameter	-
$n_{k_{true}}$	True delay parameter	-
n_X	Number of rows of matrix X	-
P_0	Initial covariance matrix for recursive least squares	-
P_1	Initial parameter value	-
P_2	Final parameter value	-
P_i	Recursive least squares covariance matrix at timestep i	-
P_n	Remnant intensity	-
q	Time shift operator	-
t	Simulation time	s
T_m	Measurement time	s
T_n	Remnant time constant	s
T_0	Forgetting factor memory time constant	s
T_s	Sampling frequency	Hz
u	Control signal	-
u_d	Delayed control signal	-
u_{meas}	Measured control signal	-
u_{pred_k}	Predicted control signal of kth ARX model	-
u_{pred}	Predicted control signal	-
u_{sim}	Simulated control signal	-
y	Controlled element position	deg
$\boldsymbol{O}$	Zero matrix	-

I
Scientific article

to be graded for AE5310 Final Thesis

Online Time-Varying Identification and Detection of Operator Adaptation with Recursive ARX Models

Author: W. Plaetinck; Supervisors: D. M. Pool, M. M. van Paassen, and M. Mulder, Control & Simulation Section, Faculty of Aerospace Engineering
Delft University of Technology, Delft, The Netherlands
Email: w.plaetinck@student.tudelft.nl, {d.m.pool, m.m.vanpaassen, m.mulder}@tudelft.nl

***Abstract*—Identifying the time-varying, adapting human operator online could enable adaptive human-machine support systems and attention monitoring for human-in-the-loop tasks. A validated, online identification method, including adaptation detection is missing however. In this paper, online identification was done using low-order ARX models with recursive least squares estimation during an experiment. Eight subjects performed a compensatory tracking task with time-varying controlled element, while explicitly indicating when they detected this change with a button push. The experiment validated the online identification method, but time-varying delay estimation is still lacking. Additionally, two adaptation detection methods were evaluated. The first method was based on detecting the deviation of the human error rate response gain $K_{\dot{e}}$ from *a priori* measured non-adapted behaviour, and was the most successful with a detection accuracy of 57%. The second method used a moving average of the $K_{\dot{e}}$ trace itself as reference for detecting deviations. This is more flexible in practical applications, but led to a lower accuracy of 43%. For both methods, the lag in detection is no issue for applications since it is similar to the human operator detection lag or even smaller. The developed methods are a large step towards adaptive operator support systems in control tasks and operator attention monitoring. Furthermore, the developed methods are useful as a tool in manual control tracking task experiments to learn more about the adaptive and learning human operator.**

I. Introduction

Past efforts in manual control cybernetics resulted mainly in human operator (HO) models for time-invariant behaviour [1] [2]. To further advance the field of cybernetics, time-varying HO behaviour should be investigated [3]. Humans are adaptive, learning controllers which explains why they are often essential in real-life control tasks [4]. On top of that, performing time-varying HO identification in an online fashion during a control task could allow for new applications. For instance, the detection of reduced operator attention or distraction could be possible [5]. Also adaptive human-machine support systems and interfaces, tracking the current operator behaviour, could be enabled [6]. Finally, an online HO adaptation detection tool for use during experiments would be useful for the cybernetics research community.

Two perspectives on dealing with time-varying HO behaviour were followed in previous studies. The time-varying behaviour was either captured in empirically derived logic rules [7] [8] or identified using system identification methods. Studies using the second perspective tried to identify the adaptation to time-varying controlled elements (e.g., [9]) and time-varying feedback elements (visual [10], motion [11], and control manipulator [12]). These were aimed at offline, *a posteriori*, identification. Those studies can be classified to fit either a batch of data at once or fit recursively. Furthermore, they have different ways of introducing the adaptation in the model.

In [9] genetic maximum likelihood estimation is used while the adaptation is assumed to follow a sigmoid shape, the parameters of which are included in the estimation. In [13] a linear parameter varying model is fit with experimentally determined scheduling functions to represent the adaptation. In [14] a two-step method is used: the wavelet transform identifies a time-varying frequency response and then an operator model is fit to that response. All are batch estimation methods and thus not directly applicable online.

On the other hand, recursive methods require no assumption on the shape of adaptation and could be applicable online. A non-parametric method is Finite Impulse Response estimation with RLS [12] [15]. Parametric methods include the extended Kalman filter [16] [10], the unscented Kalman filter [17] and dual extended Kalman filter [18] to estimate HO parameters directly. Another approach is the use of ARX models with recursive least squares (RLS) estimation for the ARX coefficients, which then need a conversion to the HO parameters [19]. Note that in [19], an important extension to the time-variant case is made of HO modeling with an ARX model structure [20] [21] [22].

In the state-of-the-art, a validated online HO identification method, together with a tested algorithm for detecting adaptation using identified HO parameter traces is missing. This paper addresses the issue, with a focus on adaptation to a time-varying controlled element.

In this paper, recursive ARX models are used to develop an online time-varying identification method, combined with detection of HO adaptation. A human-in-the-loop experiment of a compensatory tracking task with time-varying controlled element (CE), with conditions similar to [9], was set up. A fast and slow CE change were given as test conditions to eight test subjects. Subjects were also asked to indicate, by means of a button push, when they detected the change in CE dynamics.

Low-order ARX models with constant assumed time delay were fitted online using RLS. Two adaptation detection methods were applied to the identified time traces of the HO error response gain and error rate response gain. The methods detected a perturbation of the parameter compared to an average measure, either a time-invariant condition average or moving average of the actual time trace. This setup allowed us to directly compare the adaptation detection performance of the proposed methods and also compare the identified adaptation instances to the subjective HO detection times.

The paper is structured as follows. In Section II the control task, identification and adaptation detection methods are discussed. Section III explains the used experimental setup. Results are shown in Section IV and discussed in Section V. Conclusions are drawn in Section VI.

II. Method

A. Control task and operator model

A single-axis, single-channel compensatory tracking task was set up to evaluate the HO identification and adaptation detection methods, similar to Zaal's recent experiment [9]. A block diagram of this task is shown in Fig. 1. The HO steered the CE dynamics H_{CE} such that its output y tracked the forcing function f as closely as possible. The only feedback to the HO was the tracking error e through a compensatory display and the HO provided a single control output u.

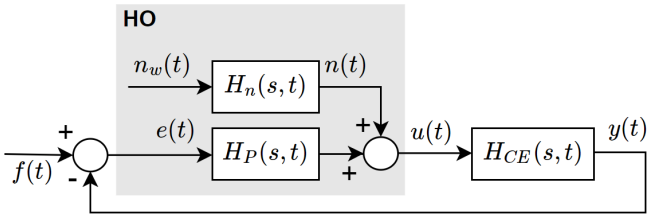


Fig. 1: A compensatory tracking task with time-varying CE dynamics and adapting quasi-linear HO model.

To induce time-varying HO behaviour, the CE dynamics in (1) were changed over time according to a sigmoid scheduling function [9]. Both the gain K_c and the break frequency ω_c were scheduled according to (2), substituting the initial value P_1 and final value P_2 for the respective CE parameter values.

$$H_{CE}(s,t) = \frac{K_c(t)}{s^2 + \omega_c(t)s} \tag{1}$$

$$P(t) = P_1 + \frac{P_2 - P_1}{1 + e^{-G(t-M)}} \tag{2}$$

The sigmoid transition is determined by the maximum rate of change G, and centred on time M. The transition in the CE parameters is shown in Fig. 2, for $M = 40.96s$ and two different settings of G. The initial CE dynamics resemble a single integrator and the final CE dynamics approximate a double integrator, forcing the HO to generate lead [1].

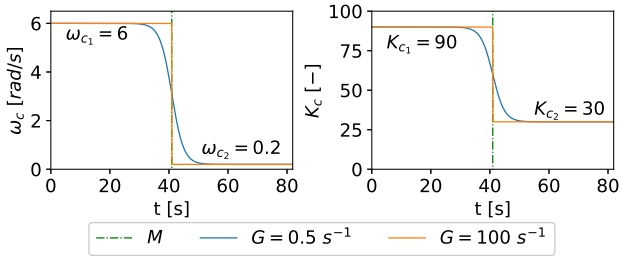


Fig. 2: CE parameters scheduled by sigmoid function for a fast ($G = 100$ s^{-1}) and slow change ($G = 0.5$ s^{-1}), located at $M = T_m/2$. A task run lasts $T_m = 81.92$s.

A validated HO model in time-invariant compensatory control tasks is the following quasi-linear model [1]. It consists of a describing function $H_P(s)$ and an additive noise, called remnant, with filter $H_n(s)$ as in Fig. 1. The remnant was not explicitly modelled. A valid describing function $H_P(s)$ for the considered CE dynamics is (3), with $H_{nm}(s)$ the neuromuscular dynamics [1].

$$H_P(s) = K_e(1 + T_L s)e^{-s\tau}H_{nm}(s) \tag{3}$$

To model a time-varying HO, it was assumed the describing function still holds if all its parameters are free to vary over time. Additionally, to avoid identification ambiguity between HO error response gain K_e and lead time constant T_L, an error rate response gain $K_{\dot{e}}(t) = K_e(t)T_L(t)$ was introduced as in (4). Due to identification method constraints the HO delay τ was assumed constant, which was found valid for the considered task [9]. The neuromuscular dynamics $H_{nm}(s,t)$ are given in (5) and are modelled as a second-order system with natural frequency ω_{nm} and damping ratio ζ_{nm}.

$$H_P(s,t) = (K_e(t) + K_{\dot{e}}(t)s)e^{-s\tau}H_{nm}(s,t) \tag{4}$$

$$H_{nm}(s,t) = \frac{\omega_{nm}^2(t)}{\omega_{nm}^2(t) + 2\zeta_{nm}(t)\omega_{nm}(t)s + s^2} \tag{5}$$

In total, the HO model thus has four time-varying parameters $(K_e, K_{\dot{e}}, \omega_{nm}, \zeta_{nm})$ and one time-invariant parameter (τ) to be estimated.

B. Online time-varying operator identification

Using measured time traces of the tracking error e and control signal u, a model for the HO can be identified. In this paper, an 'autoregressive with external input' (ARX) model structure is used for this purpose [19]. First the ARX coefficients are identified using Recursive Least Squares (RLS). Secondly, these identified coefficients are converted to the actual HO model parameters to complete the identification process.

1) ARX model structure: The general ARX model structure is shown in (6). This is a discrete-time model with time shift operator q, such that $q^{-n_k}e(t) = e(t - n_k)$ to model the HO delay. The HO describing function is approximated by the ratio $\frac{B(q)}{A(q)}$ and delay shift of n_k samples. In ARX models, the HO remnant noise is modelled by the ratio $\frac{1}{A(q)}$, which filters a white noise input signal $\epsilon(t)$. Note that the process and noise dynamics are thus coupled through $A(q)$.

$$u(t) = \frac{B(q)}{A(q)}q^{-n_k}e(t) + \frac{1}{A(q)}\epsilon(t) \tag{6}$$

In this paper, second-order $A(q)$ and $B(q)$ polynomials were chosen to match (4) and thus allow straightforward HO parameter retrieval, as described later. This results in (7) for the HO model approximation $H_P(q)$. The model structure allows to use RLS for the polynomial coefficients, but not for the time shift parameter n_k [19]. An attempt to include n_k in the estimation is described in [23]. However, in further steps the time delay was assumed to be set *a priori* and constant at $\tau = 0.28$ s [9], corresponding to $n_k = 28$ for the timestep $\Delta t = 0.01$ s chosen in the experiment.

$$H_P(q) = \frac{B(q)}{A(q)}q^{-n_k} = \frac{b_0^d + b_1^d q^{-1}}{1 + a_1^d q^{-1} + a_2^d q^{-2}}q^{-n_k} \tag{7}$$

2) Recursive Least Squares: The ARX model can be rewritten in linear regression form as $\hat{u}[i|i-1, \theta_i] = \varphi[i]\theta_i$ for each step i, with the regression vector φ and coefficient vector θ_i in (8) and (9) [24].

$$\varphi[i] = (-u[i-1] \quad -u[i-2] \quad e[i-n_k] \quad e[i-n_k-1]) \tag{8}$$

$$\theta_i = \begin{pmatrix} a_1^d & a_2^d & b_0^d & b_1^d \end{pmatrix}^T \tag{9}$$

An RLS algorithm with exponential forgetting factor was used to estimate and track the ARX coefficients. The RLS gain K_i is calculated from the covariance matrix P_{i-1} of the previous timestep and the current regression vector $\varphi[i]$ in (10). The RLS gain determines to what extent the prediction error between current control output $u[i]$ and estimated output $\varphi[i]\hat{\theta}_{i-1}$ contributes to the coefficient update $\hat{\theta}_i$ in (11). Finally the covariance matrix P_i is updated in (12). The exponential forgetting is taken care of by the λ value in both the RLS gain update (10) and covariance matrix update (12).

$$K_i = P_{i-1}\varphi^T[i](\varphi[i]P_{i-1}\varphi^T[i] + \lambda)^{-1} \tag{10}$$

$$\hat{\theta}_i = \hat{\theta}_{i-1} + K_i(u[i] - \varphi[i]\hat{\theta}_{i-1}) \tag{11}$$

$$P_i = \frac{1}{\lambda}(P_{i-1} - K_i\varphi[i]P_{i-1}) \tag{12}$$

$$\forall\ i = i_{start}, .., N-1, N$$

The RLS is done for every step i from i_{start} till N. The start time must be large enough such that there is enough data available to perform the n_k shift, and initial transient effects due to the HO finding his steady operating point at the start of the task are ignored. Therefore, $i_{start} = 3$ s was chosen. The initial conditions θ_0 and P_0 are given in (13). The forgetting factor, for the 100 Hz data rate, was set to $\lambda = 0.99609$ based on previous work [19]. This corresponds to a memory horizon of 2.56 s.

$$\begin{array}{rcl} \hat{\theta}_0 & = & (-1.85 \quad 0.85 \quad 0.08 \quad -0.08)^T \\ P_0 & = & 10,000\boldsymbol{I}_{4x4} \end{array} \tag{13}$$

3) HO parameter retrieval: The estimated discrete-time ARX coefficients need to be converted to the continuous-time HO describing function parameters. The ARX model can be mathematically written as a discrete-time transfer function $H_P(z)$ using the following Z-transform property: $Z[q^{-n}f(i\Delta t)] = z^{-n}F(z)$. Note that $n_k = \tau/\Delta t$ in (14).

$$H_P(z) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}}z^{-\frac{\tau}{\Delta t}} \tag{14}$$

To retrieve the HO model parameters, this discrete-time transfer function needs to be converted to the continuous-time domain. The zero-order-hold conversion method, as specified in Appendix A, is used for this, which results in the coefficients of (15).

$$H_P(s) = \frac{b_0^c s + b_1^c}{s^2 + a_1^c s + a_2^c}e^{-s\tau} \tag{15}$$

By comparing (15) with the HO model in (4) and (5), the following system of equations to retrieve the HO parameters from the converted ARX coefficients can be derived:

$$\begin{array}{rclcrcl} K_e & = & \frac{b_1^c}{a_2^c} & & K_{\dot{e}} & = & \frac{b_0^c}{a_2^c} \\ \zeta_{nm} & = & \frac{a_1^c}{2\sqrt{a_2^c}} & & \omega_{nm} & = & \sqrt{a_2^c} \end{array} \tag{16}$$

When using higher order ARX models, these relations between the ARX coefficients and the HO model parameters become overdetermined. Thus a model order reduction technique would be required.

C. Adaptation detection

For the considered task, the expected HO adaptation is a reduced error gain K_e and increased error rate gain $K_{\dot{e}}$. Because for a change from approximately single to double integrator to control, HOs will need to generate lead [1]. For the considered change in CE dynamics, the delay and neuromuscular parameters were found not to vary significantly [9]. Note that this is in contrast to **??** which showed 150 ms additional delay for double integrator over single integrator dynamics. Thus the adaptation detection method should be based on either K_e or $K_{\dot{e}}$ time traces.

The proposed detection methods use an average measure of the considered parameter, plus or minus a certain value as margin. In Figs. 3 and 4 the blue area indicates this reference band. The used average measure is method-specific as described below, and the width of the reference band as margin is set by the $\delta_{K_{...}}$ value.

Two different average measures were tested for the reference band, thus leading to two adaptation detection methods:

- *Time-Invariant Condition Average* (TICA): the measure is calculated by averaging the identified parameter traces of several *a priori* measured time-invariant task runs, with only the single-integrator CE dynamics.
- *Moving Average* (MA): the measure is calculated from a moving average over a window of n_s samples up to the current step i, of the identified parameter trace itself. This is demonstrated in (17) for $K_{\dot{e}}$, but K_e is equivalent.

$$\text{MA} = \frac{1}{n_s} \sum_{k=0}^{n_s-1} K_{\dot{e}_{i-k}} \tag{17}$$

Both methods were applied to identified traces of both K_e and $K_{\dot{e}}$. The methods are graphically illustrated in Figs. 3 and 4 for $K_{\dot{e}}$ only, but K_e is equivalent. The reference band represents the non-adapted HO behaviour, with a margin for typical oscillations seen in the parameter value. When the parameter value is outside of this band at a certain timestep, for a period longer than a window with length ΔT, an adaptation is detected and the initial time of the current window saved as t_{detect} further referred to as t_d.

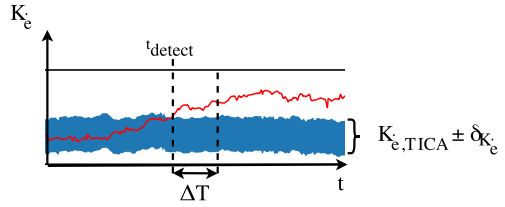


Fig. 3: Graphical illustration of adaptation detection based on the time-invariant condition average (TICA) measure $K_{\dot{e},TICA} \pm \delta_{K_{\dot{e}}}$.

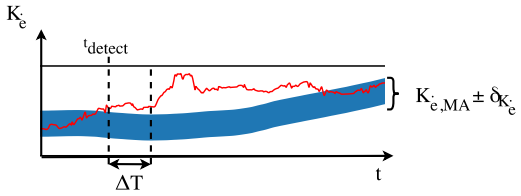


Fig. 4: Graphical illustration of adaptation detection based on the moving average (MA) measure $K_{\dot{e},MA} \pm \delta_{K_{\dot{e}}}$. Note that the MA converges again with the actual time trace after the adaptation took place.

The methods have several settings that can be tuned to minimise false positives and negatives. A false positive occurs when a detection was made outside of the interval $M < t_d < 60$ s, with M the moment of transition. A limit of 60 s was chosen since by then the HO parameter traces should definitely be converged already to the adapted behaviour and thus the detection should have been triggered. A false negative occurs when no detection was made. This could happen when the parameter trace did not go outside of the reference band for longer than ΔT seconds during a task run.

The tunable settings are here referred to as hyperparameters, to distinguish them from the identified HO parameters. They are listed as follows:

- Margin size δ_{K_e} or $\delta_{K_{\dot{e}}}$: the deviation allowed from the average measure without leaving the reference band for each considered HO parameter respectively. The width of the reference band is determined by this value.
- Window size ΔT: the minimum amount of time the current parameter trace should be outside of the reference band.
- Number of samples n_s: for the MA method, the amount of samples taken in the moving average calculation. Naturally, n_s influences the tracking speed of the moving average.
- λ and n_k: since both detection methods depend on the identification results of the recursive ARX, λ and n_k could be seen as hyperparameters of the overall approach as well.

A sensitivity study of the detection performance with respect to the described hyperparameters was done for each method. Each hyperparameter was varied over a relevant range, while keeping the other hyperparameters constant at their initial chosen value. Only λ and n_k, were not included in the study. For each combination, the mean squared error (MSE) of the detection time with respect to the moment of transition M was calculated. This was done over all measured runs in a certain condition, as described later. In case of a too early detection ($t_d < M$) or no detection, the maximum possible MSE was assigned for that run by setting $t_d = 0$ s. The MSE calculation is shown in (18) for $K_{\dot{e}}$, but K_e is equivalent.

$$\text{MSE}(\delta_{K_{\dot{e}}}, \Delta T, n_s) = \frac{1}{n_{runs}} \sum_{j=1}^{n_{runs}} (M - t_d[j; \delta_{K_{\dot{e}}}, \Delta T, n_s])^2 \tag{18}$$

III. Evaluation Setup

To evaluate the online identification performance of recursive ARX and to check whether the adaptation detection methods were accurate, a human-in-the-loop experiment was performed using the described control task. Also adaptation detection times as experienced by the HOs were explicitly measured in the experiment.

A. Apparatus

The experiment was performed in the fixed-base Human-Machine Interaction Laboratory (HMILab) simulator in the Faculty of Aerospace Engineering at Delft University of Technology. The HMILab runs the Delft University Simulation Environment (DUSIME) for real-time distributed simulation [25]. The identification method described in Section II-B was implemented in the environment as a separate C++ module. Task simulation and identification modules ran at 100 Hz.

The control task was implemented as a pitch tracking task. Participants sat in front of a simplified artificial horizon display as in Fig. 5, showing the pitch tracking error as distance between the aircraft symbol and the horizon line. The distance to the display was 75 cm. The display had an update rate of 60 Hz and the display lag for this setup was determined to be 20-25 ms [26] [27].

Control inputs were given with an electro-hydraulic side-stick at the participant's right hand side as in Fig. 6a. The stick's torsional stiffness was 2.5 Nm/rad, its damping 0.22 Nm · s/rad, and its inertia 0.01 kg · m^2 with a moment arm of 9 cm. The stick could only rotate around the pitch axis. A push button on the stick, see Fig. 6b, was used by the participants to indicate when they detected the change in the CE.

On the researcher's monitoring station the tracking performance, time-varying identification results, and the adaptation detection (button push) by the participant could be followed in real-time throughout the experiment.

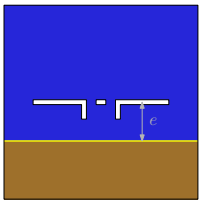


Fig. 5: Pitch compensatory display.

(a) (b)

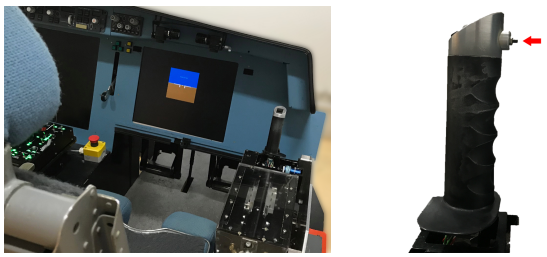


Fig. 6: HMILab setup (a) with push button on control stick (b).

B. Forcing function

The forcing function f was defined as a sum of ten sinusoids with different frequencies, amplitudes and phases as in (19). The frequencies are integer multiples of the base measurement frequency $\omega_m = 2\pi/T_m$.

$$f(t) = \sum_{k=1}^{10} A_f[k]\sin(n_f[k]\omega_m t + \phi_f[k]) \qquad (19)$$

The forcing function was identical to the signal used in previous work [9]. However, it was shifted to the left in time by 10.5 s as in Fig. 7, through adjusting the phases ϕ_f. This allowed a larger magnitude change in pitch in the transition region, possibly giving the HO a more fair change of detecting the CE transition on time. All forcing function parameters are listed in Table I.

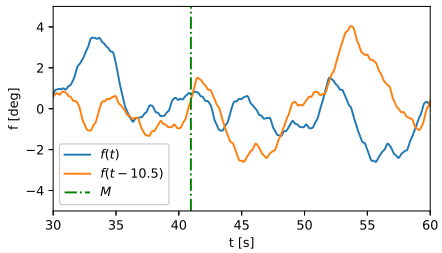


Fig. 7: View of forcing function in the CE transition region, before and after the shift in time.

TABLE I: Forcing function parameters, modified from [9].

k [-]	n_f [-]	$n_f\omega_m$ [rad/s]	A_f [deg]	ϕ_f [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

C. Independent variables

Two independent variables were used: rate of change of the scheduling sigmoid with two levels and run-in time with three levels. In addition, two time-invariant reference conditions were given to the participants. Thus in total eight different conditions were used, as listed in Table II and described below.

TABLE II: Experiment Conditions

Condition	Initial dynamics	Final dynamics	Run-in time [s]	G [1/s]
TI1	H_{CE_1}	-	5	-
TI2	H_{CE_2}	-	5	-
TV12S-T5	H_{CE_1}	H_{CE_2}	5	0.5
TV12S-T10			10	
TV12S-T15			15	
TV12F-T5			5	100
TV12F-T10			10	
TV12F-T15			15	

A transition in CE could be either present or not in a run. The time-invariant CE runs had either $H_{CE_1}(K_c = 90, \omega_c = 6$ rad/s) or $H_{CE_2}(K_c = 30, \omega_c = 0.2$ rad/s) dynamics. The measured behaviour in the TI1 condition was used for the

average measure calculation in the first adaptation detection method.

For runs with time-varying CE ($H_{CE_1}$to H_{CE_2}), the transition was scheduled by a sigmoid with $M = T_m/2$, thus halfway the measurement window. The maximum rate of change had two levels: $G = 0.5$ s^{-1} to represent a gradual change, $G = 100$ s^{-1} to represent an almost instantaneous change.

Each experiment run contained a measurement time of $T_m = 81.92$ s. A minimum run-in time of 5 s was added to allow transients, due to the HO finding his steady operating point, to die out. Furthermore, the run-in time was varied to avoid expectation of the CE transition by the participants. Three levels were used: 5 s, 10 s and 15 s. Care was taken to make sure that the forcing function inside the measurement window remained the same for each level.

D. Participants and experimental procedures

Eight participants volunteered to perform the experiment after providing written informed consent. Seven participants had previous experience with equivalent manual control tasks. Participants were instructed to minimise the tracking error shown on the display and push the button whenever they noticed changed CE dynamics, without giving further information about the conditions.

The experiment contained a training and test part. In the training part, a participant performed at least each time-invariant condition and the two time-varying conditions, all with a run-in time of 5 s. The test part was divided in three blocks with each containing the eight conditions, enabling three repeated measurements. For each block the conditions were presented in a randomised order by a balanced Latin square design. After each run, the root mean square of the tracking error signal e was fed back to the participant as motivation to perform well. A short break was held in between each block to reduce fatigue.

E. Data processing

Using the data from conditions TV12S and TV12F, a two-way repeated measures ANOVA was performed with G and run-in time as factors to check whether they have a significant influence on the subjective HO adaptation detection times. The three repeated measurements with same run-in time were averaged per level for each subject. These averaged detection times were then used as input for the ANOVA.

After verifying normality and sphericity assumptions, the analysis revealed no significant effect due to different run-in times, with $\mathrm{F}(2, 14) = 1.92,\ p > 0.05$. However, G does have a significant influence as expected, with $\mathrm{F}(1, 7) = 15.76,\ p < 0.05$. This test outcome allows the trimming of the experimental data to the final 8192 samples, corresponding to the measurement window T_m. Furthermore, the run-in time levels can now be seen as one set of nine repeated measurements for the respective TV12S and TV12F conditions, for each subject.

To present the identification results, the parameter traces were averaged over those nine runs per time-varying condition (TV12S or TV12F) for each subject.

Adaptation detection is on the other hand still analysed and presented on a run per run basis, with condition repetitions seen as separate results.

IV. Results

The average tracking and identified HO parameter traces per subject are shown in Fig. 8 and 9 for conditions TV12F and TV12S, respectively. Note that these include a bias due to the remnant, as explained in Section II-B. A clear increase in $K_{\dot{e}}$ (Fig. 8d & 9d) can be seen for all subjects in both conditions. The expected decrease in K_e (8c & 9c) is less noticeable and depends on the subject. Neuromuscular parameters in general stay approximately constant (8e-f & 9e-f). The median of the button pushes for each subject is also included (8b & 9b) and will be discussed in more detail below.

A. Subjective adaptation detection

The HO button pushes are classified in a confusion matrix in Fig. 10 according to their correctness for the respective condition. False positives (FP) mean the HO incorrectly pushed (for TI1 & TI2) or pushed before an actual transition happened for TV12S & TV12F, thus $t_{push} < M$. False negatives (FN) mean the HO did not push during the run while a CE transition did in fact occur in TV12S & TV12F. True positives (TP) indicate the HO did push when required in those conditions, thus $M < t_{push}$. True negatives (TN) indicate that the HO did not push the button when it was indeed not required, as in TI1 & TI2.

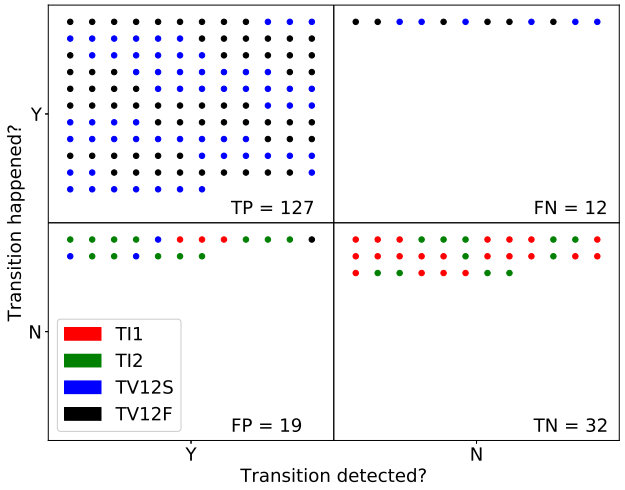


Fig. 10: Confusion matrix of button pushes of all subjects for all conditions (TP = true positives, FP = false positives, TN = true negatives, FN = false negatives.)

The matrix shows that the FPs were mainly incorrect pushes for the time-invariant conditions, with a majority in the TI2 condition. The double integrator dynamics are more difficult to steer and could have confused the subjects in detecting

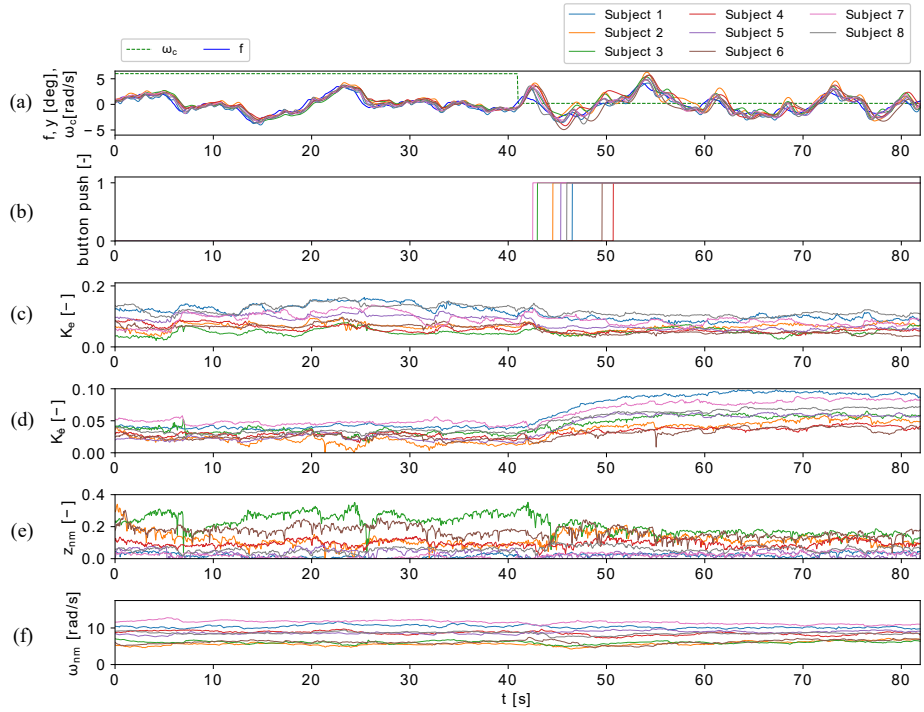


Fig. 8: Averaged time traces of identified HO parameters (c-f) and median of button pushes (b), taken over all nine TV12F runs per subject. For reference, the forcing function and tracking are shown, together with the scheduled ω_c to indicate the fast CE transition (a).

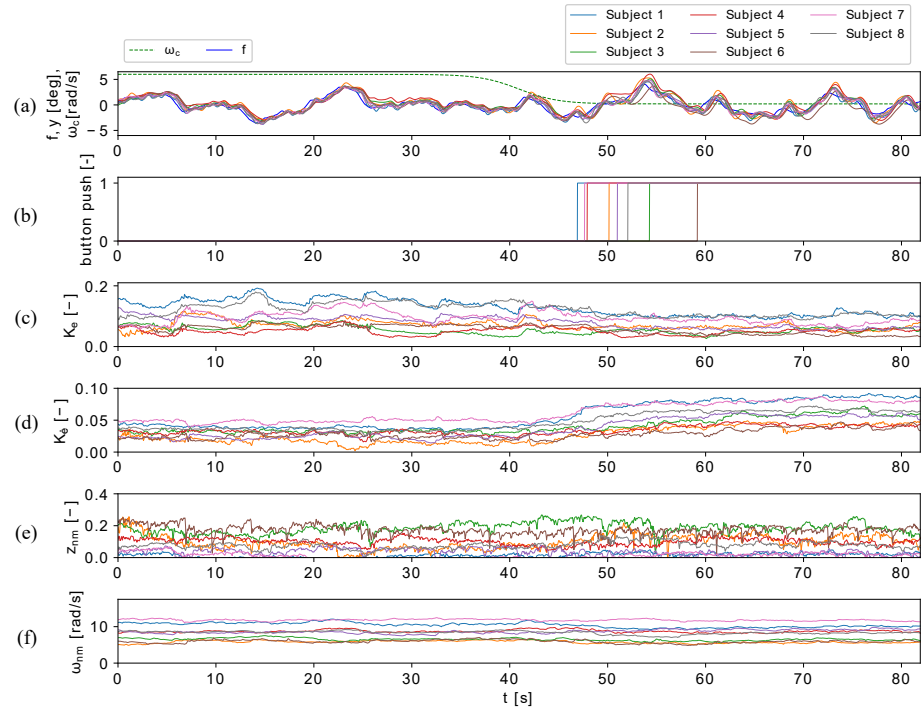


Fig. 9: Averaged time traces of identified HO parameters (c-f) and median of button pushes (b), taken over all nine TV12S runs per subject. For reference, the forcing function and tracking are shown, together with the scheduled ω_c to indicate the slow CE transition (a).

a changed CE. After debriefing, one subject indicated to be unaware of the option to not push the button during a run. This resulted in six of the recorded FPs (three in TI1, three in TI2). Also, when subjects made a detection mistake they often indicated it directly after. FNs were spread among subjects and divided between TV12S and TV12F, thus no direct cause was found. The detection accuracy, as defined in (20), of the HO was 83% for all conditions. It was 88% within the time-varying conditions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{20}$$

By comparing the button push time with the moment of transition M, detection lags can be calculated. Their distribution for all subjects is shown in Fig. 11 for both time-varying conditions. Condition TV12F had a lower median detection lag than TV12S.

A dependent t-test between both conditions was performed as post-hoc analysis to check significance of this result. Since ANOVA indicated no significant influence of run-in time levels on subjective HO adaptation detection, the detection lags for all nine runs over the three levels were averaged per subject as input for the test. The t-test implied that the found difference between TV12F and TV12S is significant with $\text{t}(7) = -3.63,\ p < 0.05$.

Note that the lower detection lag for TV12F is logical since the CE transition completion time of TV12S is later than that of TV12F. However, this also means that the gradual CE change does not give the HO the opportunity to detect the transition before it is fully completed. This is confirmed by plotting the distribution of button pushes from all subjects, for TV12S, against the sigmoid completion percentage as in Fig. 12.

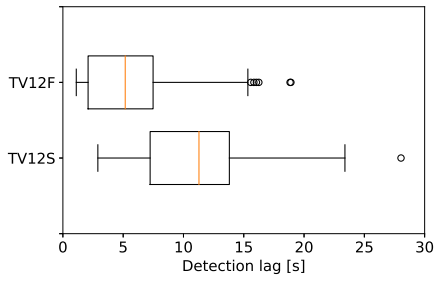


Fig. 11: Distribution of subjective detection lags for all subjects, in TV12F and TV12S conditions.

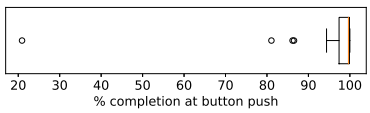


Fig. 12: The percentage of sigmoid transition completion required to trigger the HO adaptation detection, in TV12S.

B. Identified adaptation detection

To demonstrate the working principle of the tested adaptation detection algorithms, example results for TV12F experiment runs are shown in Fig. 13 and 14 for both methods. The initial used hyperparameter settings were based on inspecting the identified HO parameter traces.

Three levels of the detection calculations are shown. First the unfiltered detection is given, which is triggered whenever the actual parameter trace (shown in red) is outside of the blue shaded reference band (13e & 14e). Then detections that have a length longer than the ΔT threshold are filtered (13f & 14f). Finally, only the first occurrence of such a detection is kept, resulting in a step signal (13g & 14g).

Per method, these calculations are done for both the K_e and $K_{\dot{e}}$ parameters. Note that in the example in Fig. 14, no detection was in the end found based on the K_e parameter since the actual parameter trace did not go out of the reference band for longer than ΔT seconds.

1) Detection performance: The detection results for all runs and subjects are classified in a confusion matrix in Table III. FP and FN are defined as in Section II-C, note that the FP definition for the methods is more strict than for the subjective HO adaptation detection. For both TV12F and TV12S, the TICA method combined with the $K_{\dot{e}}$ parameter had the highest number of TPs.

TABLE III: Confusion matrix for both methods and time-varying conditions. Hyperparameters: $\delta_{K_e} = 0.06$, $\delta_{K_{\dot{e}}} = 0.02$, $\Delta T = 3$ s, $n_s = 1500$.

Condition	Method	Parameter	TP	FP	TN	FN
TV12F	TICA	K_e	12	33	0	27
		$K_{\dot{e}}$	41	27	0	4
	MA	K_e	0	15	0	57
		$K_{\dot{e}}$	20	12	0	40
TV12S	TICA	K_e	6	30	0	35
		$K_{\dot{e}}$	41	26	0	4
	MA	K_e	2	10	0	59
		$K_{\dot{e}}$	11	7	0	53

2) Sensitivity study of detection performance: To reduce misclassification, a sensitivity study of the detection performance to the hyperparameters choice was done. Fig. 15 shows the results of the sensitivity study. A better choice of hyperparameters was possible and apparently they should be different for both detection methods.

Setting $\delta K_e = 0.05$ and $\delta K_{\dot{e}} = 0.02$ for the TICA method gave minimum MSE. For the MA method this was achieved for $\delta K_e = 0.03$ and $\delta K_{\dot{e}} = 0.012$. The initial ΔT and n_s values were kept since also in the new situation they indicate a MSE close to the minimum, as visible in Fig 16.

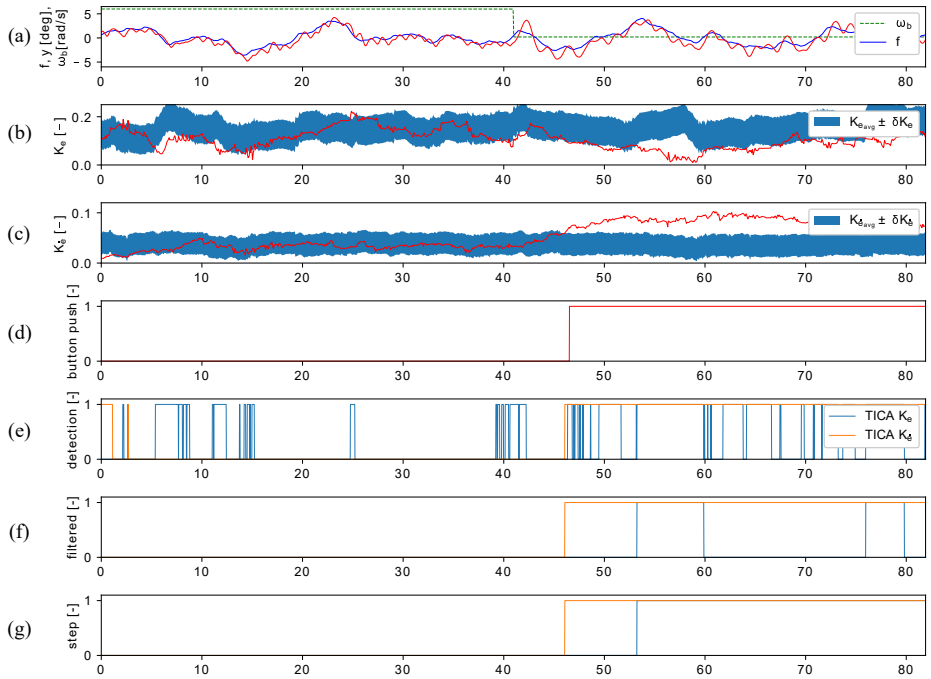


Fig. 13: Adaptation detection using TICA. Hyperparameters: $\delta K_e = 0.06$, $\delta K_{\dot{e}} = 0.02$, $\Delta T = 3$ s.

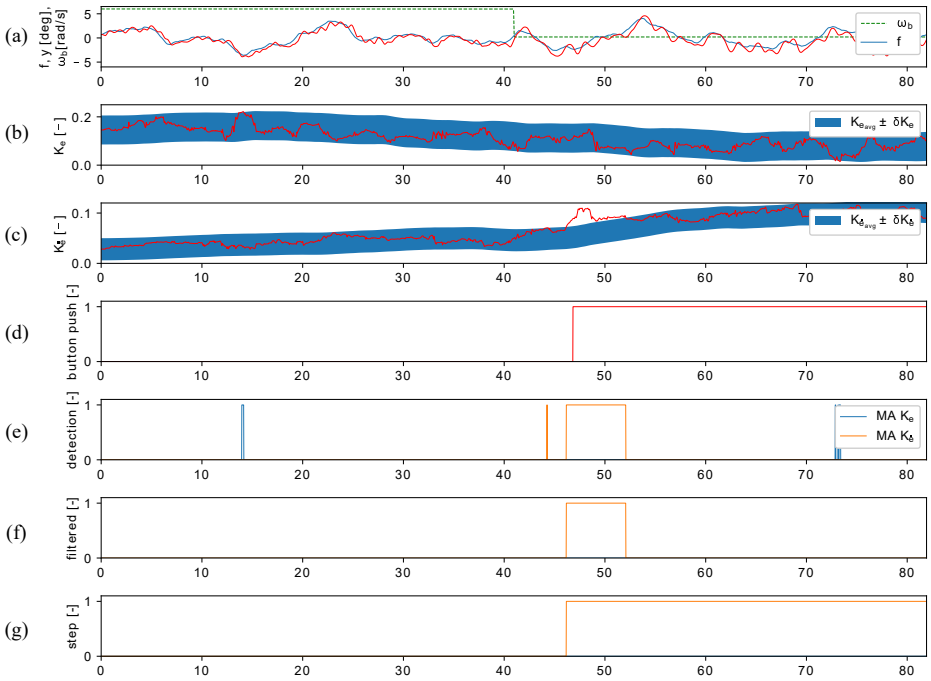


Fig. 14: Adaptation detection using MA. Hyperparameters: $\delta K_e = 0.06$, $\delta K_{\dot{e}} = 0.02$, $\Delta T = 3$ s, $n_s = 1500$.

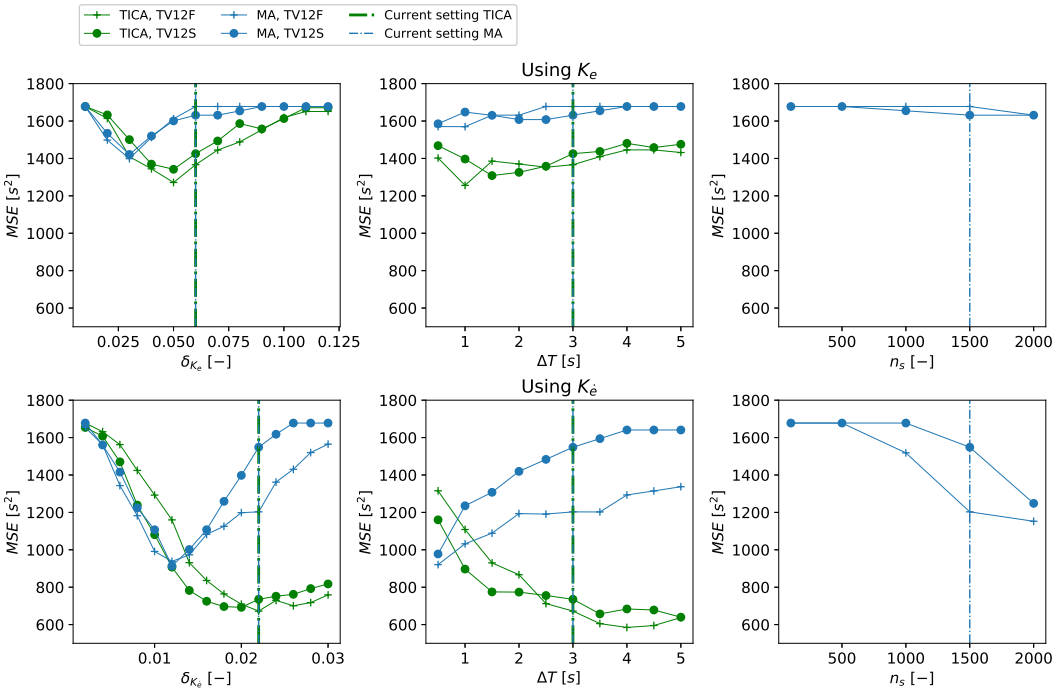


Fig. 15: Sensitivity study varying one hyperparameter at a time, while keeping the others constant at the initial chosen setting. Both TICA and MA methods, combined with both K_e and $K_{\dot{e}}$ as input were analyzed for both time-varying conditions.

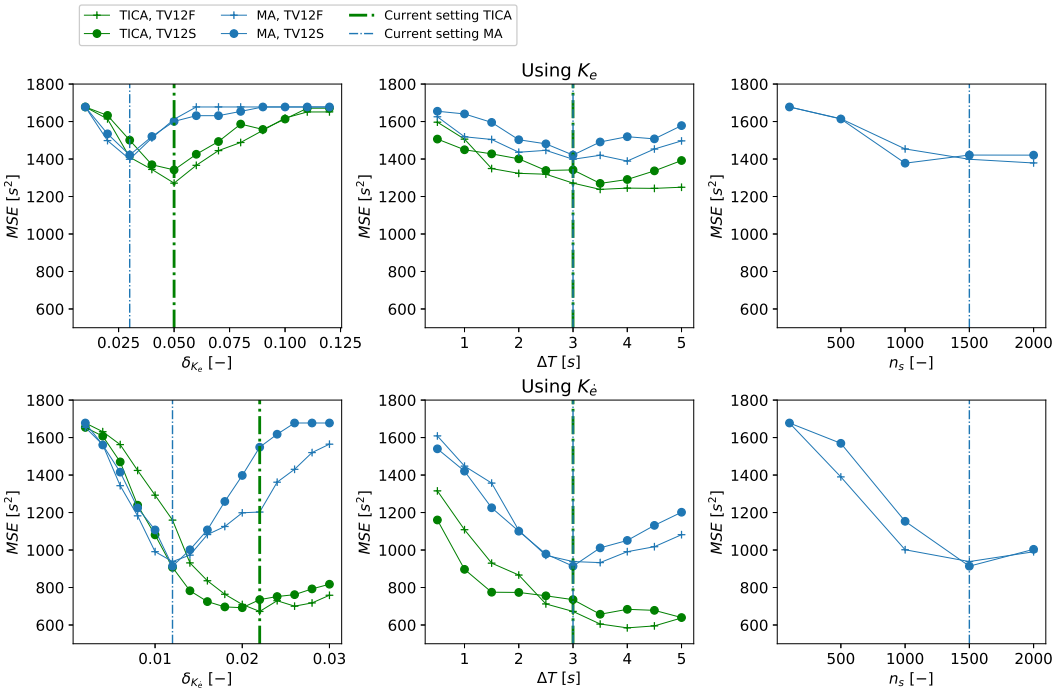


Fig. 16: New sensitivity study, after only modifying the δ_{K_e} and $\delta_{K_{\dot{e}}}$ settings. The new situation shows no further need to modify ΔT and n_s to significantly reduce MSE.

After tuning the hyperparameters, the confusion matrices were updated and changes in the classification indicated. Detection accuracies (Acc) were calculated according to (20). Table IV shows a slight improvement in TPs for TICA K_e combined with less FNs, meaning more detections were attempted. TICA $K_{\dot{e}}$ classification performance remained stable. There is no huge influence of the condition, indicating the type of CE transition, on the detection accuracy of each method.

Table V shows a slight improvement in TPs for MA K_e but a major one for MA $K_{\dot{e}}$. Also the amount of FNs dropped considerably for both MA K_e and $K_{\dot{e}}$, meaning more detections were attempted and those were divided between TPs and FPs. The type of condition does not influence the detection accuracy achieved by each method.

When comparing the detection accuracies (Acc) in both tables, there is a significant difference between the methods. Looking at TV12F, TICA $K_{\dot{e}}$ has the highest accuracy with 57%. MA $K_{\dot{e}}$ is runner-up with 43%. The difference is explained by the 9% more FPs and 6% more FNs for MA $K_{\dot{e}}$. Similar conclusions are drawn for TV12S.

TABLE IV: Confusion matrix for TICA method in the time-varying conditions. Hyperparameters: $\delta K_e = 0.05$, $\delta K_{\dot{e}} = 0.02$, $\Delta T = 3$ s.

Condition	Parameter	TP	Δ	FP	Δ	FN	Δ	Acc
TV12F	K_e	16	+4	41	+8	15	-12	22%
	$K_{\dot{e}}$	41	+0	24	-3	7	+3	57%
TV12S	K_e	11	+5	36	+6	24	-11	15%
	$K_{\dot{e}}$	41	+0	22	-4	8	+4	58%

TABLE V: Confusion matrix for MA method in the time-varying conditions. Hyperparameters: $\delta K_e = 0.03$, $\delta K_{\dot{e}} = 0.012$, $\Delta T = 3$ s, $n_s = 1500$.

Condition	Parameter	TP	Δ	FP	Δ	FN	Δ	Acc
TV12F	K_e	9	+9	50	+35	13	-44	12%
	$K_{\dot{e}}$	31	+11	30	+18	11	-29	43%
TV12S	K_e	9	+7	48	+38	14	-45	13%
	$K_{\dot{e}}$	30	+19	29	+22	12	-41	42%

C. Comparison of subjective/identified adaptation detection

With the adaptation detection methods now fine-tuned, both methods are compared with the HO for adaptation detection performance. Fig. 17 shows the distribution of detection lags for all TP cases of the subjective HO detection and the detection methods. Only TICA and MA with the $K_{\dot{e}}$ parameter as input are shown since they have, in general, the most TPs.

To check significance of the comparison, independent Welch's t-tests were used. This test can deal with different sample sizes in the compared groups. This is required because the amount of runs in each boxplot is not necessarily the same, since only runs with a TP detection for the respective detection method are included. Furthermore, the test should be independent since a TP HO detection and a TP detection by one of the methods are independent events.

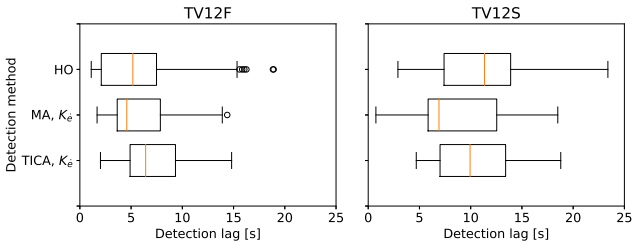


Fig. 17: Detection lags of true positive detection cases from the most successful methods and the HO, for all subjects.

In the TV12F condition, detection lags are similar for both HO and the algorithmic methods. Between the HO and the TICA method no significant difference was found, with $\mathrm{t}(103.1) = -1.62, p > 0.05$. Also no significant difference is present between the HO and the MA method, with $\mathrm{t}(79.4) = 0.0016, p > 0.05$.

In the TV12S condition, both methods outperform the HO when taking the median as the central measure. They detect the HO adaptation earlier than the HO itself, the MA method does this 4.5 s faster and the TICA method 1.3 s faster. The result is significant for the MA method, with $\mathrm{t}(83.2) = 2.68, p < 0.05$. It is insignificant for the TICA method, with $\mathrm{t}(97.0) = 1.42, p > 0.05$.

V. Discussion

Two adaptation detection methods based on identified HO parameter time traces were tested to determine their accuracy. A human-in-the-loop experiment forcing time-varying HO control behaviour was done to collect the required data and evaluate the feasibility of online HO model identification using recursive ARX.

A. Online, time-varying identification performance

The results show that recursive ARX is a feasible online, time-varying identification method for the HO model considered. This adds to the work in [19]. However, bias due to HO remnant will always be present in the results due to the ARX model structure.

With the current method no online time-varying HO delay identification was possible, since delay cannot be included in the RLS update. Including a Padé approximation for delay would allow this, but then problems appear in the conversion of ARX coefficients to the HO model parameters due to the higher ARX model order. Thus now only a constant assumption can be adjusted to match the HO delay.

B. Adaptation detection performance

The TICA method based on detecting parameter deviations from *a priori* measured time-invariant behaviour of the $K_{\dot{e}}$ parameter was found to be the most accurate. However, its detection accuracy is only 57% compared to the subjective HO

adaptation detection with an accuracy of 88%. The lower accuracy is mainly due to the extra 42 FPs the TICA $K_{\dot{e}}$ method had in both time-varying conditions combined, compared to the HO. The moving average method MA with $K_{\dot{e}}$ is runner-up, while TICA and MA combined with K_e gave clearly inferior results.

Looking at detection lags for the TP cases only, the MA method with $K_{\dot{e}}$ did detect adaptation significantly earlier than the HO itself: 4.3 s earlier for gradual transitions in CE. For immediate CE transitions, detection lags were similar for both methods.

The proposed TICA and MA methods thus do not match the classification qualities of the HO yet. However whenever they result in an TP, the tested methods are likely sufficiently quick to detect HO adaptation for driving adaptive HO support systems.

As mentioned, both methods suffered from FP and FN detections which are due to inherent fluctuations in HO parameters. The following general causes of FPs and FNs were found by analysing the data run by run:

- For TICA, K_e and $K_{\dot{e}}$: more runs in the time-invariant condition might be needed to better quantify the reference non-adapted behaviour. Now only a three run average was used, and the resulting reference band does not cover the full variability in parameter traces when no adaptation takes place. This leads to a lot of FPs.
- For TICA and MA, K_e: the K_e parameter estimates are found to oscillate with a too large magnitude compared to the magnitude of the change at adaptation. Therefore, the detection method does not work well for this parameter, leading to many FPs. Advise is to not use K_e.
- For TICA and MA, K_e and $K_{\dot{e}}$: some subjects showed almost no adaptation and thus the changes in K_e and $K_{\dot{e}}$ were too small to detect. Perhaps those subjects needed more motivation or were not excited enough by the used forcing function. This leads to FNs. Furthermore, when the subject is not tracking the forcing function very well momentarily, the HO parameters go outside of the spread band although no change in CE happened. This leads to FPs. Both issues are problems with the HO in general, not only due to the gathered data.

The methods could be improved by performing a proper optimisation routine for the hyperparameters, including a training/validation data split to determine hyperparameter values that generalise. Also, tuning per subject could be warranted for certain applications to improve detection accuracy.

C. Implementation of shown methods

Possible applications for the adaptation detection methods are attention monitoring and adaptive HO support systems (e.g. haptic feedback). For these applications a weighing will have to be made between reducing the amount of FPs or FNs at the cost of TPs. Hyperparameter tuning can be used to emphasise FP or FN suppression as much as possible. The severity of a misclassification in realistic settings will determine which classification error should be reduced the most.

An advantage of both methods is that, due to a focus on detection, the parameter biases in the identification are permissible, as long as they are present in both non-adapted and adapted conditions.

The MA method has two big advantages over the TICA method for implementation. MA is based only on the current identified parameter trace, which makes it more practical since no prior time-invariant behaviour measurements are needed. Furthermore, it can easily detect separate changes in CE, as the reference band always converges to the adapted behaviour. Whenever the parameter goes out of the band again, a new detection is simply triggered.

A disadvantage of both methods is the fact that the actual time the detection is triggered, is equal to $t_{trigger} = t_d + \Delta T$. The ΔT window is necessary to avoid too many false positives, but causes a detection delay. Nevertheless the algorithm still knows when exactly the adaptation started, but for true real-time detection, this issue still needs to be solved.

On top of that, all adaptation detection results presented in this paper are likely dependent on the excitation offered by the forcing function in the considered task.

D. Further research

A comparative study of the current adaptation detection results with those achieved using another promising online, time-varying identification solution based on Kalman filtering [10] [17] [18] could be done. The use of Kalman filtering would allow different options to solve the problem of online time-varying identification as well [18].

Further research should also check the influence of forcing function properties on detection performance, bearing in mind that in a realistic task no similar excitation might happen.

Finally, applying the developed methods to control tasks with a different kind of time-varying element could investigate the general usability of the adaptation detection methods.

VI. Conclusion

This paper proved the feasibility of online, time-varying human operator (HO) identification with recursive ARX model fitting by performing a human-in-the-loop experiment with compensatory tracking task and time-varying controlled element (CE). Online time delay estimation was however still missing in the identification.

Two adaptation detection algorithms were tested on the gathered data. The algorithm based on detecting the deviation of the HO error rate response gain $K_{\dot{e}}$ from *a priori* measured non-adapted behaviour, called TICA method, was most successful with a detection accuracy of 57%. Using a moving average of the $K_{\dot{e}}$ parameter trace itself as reference, called MA method, is more flexible in practical applications. But it was found to result in an accuracy of 43%, with 9% more false positives (FP) and 6% more false negative (FN) detections compared to the TICA method.

The adaptation detection performance of the proposed methods was compared with subjective detection by the HO. HO detection accuracy remained 31% higher than achieved with the TICA $K_{\dot{e}}$ method. Using that method resulted in 30% more FPs and 2% more FNs than the HO. In case of a true positive detection, the MA method has a significantly smaller detection lag than the HO when a gradual CE transition is happening. The lag is reduced by 4.3 s. For fast CE transitions, both methods have similar detection lag to the HO.

Regarding applicability, the proposed methods detect adaptation likely quick enough. However, currently they do not allow real-time detection due to the required detection window. Furthermore, the methods should be further tuned to a FP/FN balance suitable for their applications.

For the first time online time-varying identification with adaptation detection was realised. This is a big step towards adaptive HO support systems in control tasks and HO attention monitoring. Furthermore, the developed methods are useful as a tool in manual control tracking task experiments to learn more about the adaptive and learning HO.

REFERENCES

[1] D. T. McRuer and H. R. Jex, "A Review of Quasi-Linear Pilot Models," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-8, no. 3, pp. 231–249, Sep. 1967.

[2] M. M. Lone and A. K. Cooke, "Review of pilot models used in aircraft flight dynamics," *Aerospace Science and Technology*, 2014.

[3] M. Mulder, D. M. Pool, D. A. Abbink, E. R. Boer, P. M. T. Zaal, F. M. Drop, K. van der El, and M. M. van Paassen, "Manual Control Cybernetics: State-of-the-Art and Current Trends," *IEEE Transactions on Human-Machine Systems*, 2017.

[4] L. R. Young, "On Adaptive Manual Control," *IEEE Transactions on Man-Machine Systems*, vol. 10, no. 4, pp. 292–331, Dec. 1969.

[5] A. Ameyoe, P. Chevrel, E. Le-Carpentier, F. Mars, and H. Illy, "Identification of a Linear Parameter Varying Driver Model for the Detection of Distraction," *IFAC-PapersOnLine*, vol. 48, no. 26, pp. 37–42, 2015.

[6] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: smoothly shifting control authority?" *Cognition, Technology and Work*, vol. 14, no. 1, pp. 19–28, Mar 2012.

[7] R. A. Hess, "Modeling Pilot Control Behavior with Sudden Changes in Vehicle Dynamics," *Journal of Aircraft*, vol. 46, no. 5, pp. 1584–1592, September-October 2009.

[8] ——, "Modeling Human Pilot Adaptation to Flight Control Anomalies and Changing Task Demands," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 655–666, 2016.

[9] P. M. T. Zaal, "Manual Control Adaptation to Changing Vehicle Dynamics in Roll-Pitch Control Tasks," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 5, pp. 1046–1058, 2016.

[10] 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," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 28, no. 1, pp. 89–99, Jan. 1998.

[11] P. M. T. Zaal and D. M. Pool, "Multimodal Pilot Behavior in Multi-Axis Tracking Tasks with Time-Varying Motion Cueing Gains," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference, National Harbor (MD)*, no. AIAA-2014-0810, Jan. 2014.

[12] M. Olivari, F. M. Nieuwenhuizen, H. H. Bülthoff, and L. Pollini, "Identifying Time-Varying Neuromuscular System with a Recursive Least-Squares Algorithm: a Monte-Carlo Simulation Study," in *Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics, San Diego (CA)*, Oct. 2014, pp. 3573–3578.

[13] R. F. M. Duarte, D. M. Pool, M. M. van Paassen, and M. Mulder, "Experimental Scheduling Functions for Global LPV Human Controller Modeling," in *Proceedings of the the 20th IFAC World Congress, Toulouse, France*, ser. IFAC-PapersOnLine, vol. 50, no. 1, Jul. 2017, pp. 15 853–15 858.

[14] P. M. T. Zaal and B. T. Sweet, "Estimation of Time-Varying Pilot Model Parameters," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Portland, Oregon, Aug. 8-11*, no. AIAA-2011-6474, 2011.

[15] M. Olivari, "Measuring pilot control behavior in control tasks with haptic feedback," Ph.D. dissertation, University of Pisa, Department of Information Engineering, Jul. 2016.

[16] J. R. Schiess and V. R. Roland, "Kalman Filter Estimation of Human Pilot-Model Parameters," NASA Langley Research Center, Hampton (VA), Technical Report NASA-TN-D-8024, Nov. 1975.

[17] J. Rojer, "Towards Understanding Human Adaptation: Time-Varying Parameter Estimation Utilizing the Unscented Kalman Filter," 2016.

[18] A. Popovici, P. M. T. Zaal, and D. M. Pool, "Dual Extended Kalman Filter for the Identification of Time-Varying Human Manual Control Behavior," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Denver (CO)*, no. AIAA-2017-3666, 2017.

[19] H. van Grootheest, D. M. Pool, M. M. Van Paassen, and M. Mulder, "Identification of Time-Varying Manual-Control Adaptation with Recursive ARX models," 2017.

[20] F. M. Nieuwenhuizen, P. M. T. Zaal, M. Mulder, M. M. van Paassen, and J. A. Mulder, "Modeling Human Multichannel Perception and Control Using Linear Time-Invariant Models," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 4, pp. 999–1013, July-August 2008.

[21] F. M. Drop, D. M. Pool, M. Mulder, and H. H. Bülthoff, "Constraints in Identification of Multi-Loop Feedforward Human Control Models," in *Proceedings of the 13th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Kyoto, Japan*, 2016.

[22] N. Roggenkämper, D. M. Pool, F. M. Drop, M. M. van Paassen, and M. Mulder, "Objective ARX Model Order Selection for Multi-Channel Human Operator Identification," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Washington, D.C.*, no. AIAA-2016-4299, Jun. 2016.

[23] W. Plaetinck, "Online Time-varying Pilot Model Identification," Delft University of Technology, Preliminary Graduation Report, 2017.

[24] L. Ljung, *System Identification: Theory for the User, 2nd Edition*, 1999.

[25] M. M. van Paassen, O. Stroosma, and J. Delatour, "DUECA - Data-Driven Activation in Distributed Real-Time Computation," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit, Denver (CO)*, no. AIAA-2000-4503, Aug. 2000.

[26] K. van der El, D. M. Pool, M. M. van Paassen, and M. Mulder, "Effects of Linear Perspective on Human Use of Preview in Manual Control," *IEEE Transactions on Human-Machine Systems*, 2017.

[27] O. Stroosma, M. M. van Paassen, M. Mulder, and F. N. Postema, "Measuring Time Delays in Simulator Displays," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit, Hilton Head (SC)*, no. AIAA-2007-6562, 2007.

APPENDIX

A. Discrete to continuous-time conversion of ARX coefficients

To convert the discrete time transfer function coefficients $\theta_d = \begin{pmatrix} a_1^d & a_2^d & b_0^d & b_1^d \end{pmatrix}$ to continuous time equivalent $\theta_c = \begin{pmatrix} a_1^c & a_2^c & b_0^c & b_1^c \end{pmatrix}$, a zero-order-hold method is used. The algorithm, based on MATLAB's `d2c` command source files, is as follows:

1) Rewrite transfer function as state space system in controllable canonical form:

$$\boldsymbol{A_d} = \begin{pmatrix} 0 & 1 \\ -a_2^d & -a_1^d \end{pmatrix} \qquad \boldsymbol{B_d} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$
$$\boldsymbol{C_d} = \begin{pmatrix} b_1^d & b_0^d \end{pmatrix} \qquad \boldsymbol{D_d} = 0$$

2) Create a matrix L containing the $\boldsymbol{A_d}$ and $\boldsymbol{B_d}$ matrices:

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{A_d} & \boldsymbol{B_d} \\ \boldsymbol{O}_{n_{B_d} \text{ x } n_{A_d}} & \boldsymbol{I}_{n_{B_d} \text{ x } n_{B_d}} \end{pmatrix} \tag{21}$$

3) Take the matrix logarithm of $\boldsymbol{L}$ and scale:

$$\boldsymbol{M} = \frac{1}{\Delta t} \log \boldsymbol{L} \tag{22}$$

4) Select the new $\boldsymbol{A_c}$ and $\boldsymbol{B_c}$ matrices from $\boldsymbol{M}$. Note that output equation matrices remain the same:

$$\boldsymbol{A_c} = \begin{pmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{pmatrix} \qquad \boldsymbol{B_c} = \begin{pmatrix} M_{1,3} \\ M_{2,3} \end{pmatrix}$$
$$\boldsymbol{C_c} = \boldsymbol{C_d} \qquad \boldsymbol{D_c} = \boldsymbol{D_d}$$

5) Convert the continuous time state space system to a transfer function:

$$H(s) = \boldsymbol{C_c}(s\boldsymbol{I} - \boldsymbol{A_c})^{-1}\boldsymbol{B_c} + \boldsymbol{D_c} \tag{23}$$

II
Preliminary graduation report

as graded for AE4020 Literature Study

1 Introduction

The research field of manual control cybernetics wants to understand how a human operator controls dynamic systems. This knowledge helps in designing both the interfaces and the controlled systems themselves in order to facilitate manual or semi-automatic control. Past efforts have mostly resulted in models applicable to time-invariant human operator behaviour (Lone and Cooke, 2014). However, humans are adaptive, learning controllers and that is why they are especially useful. At this moment there is only limited knowledge available about time-varying human operator behaviour including adaptation and learning in control tasks (Mulder et al., 2016).

Current research trying to explain this adaptive behaviour advances cybernetics theory by understanding the evolving internal representation humans form about a control task (Mulder et al., 2016). More directly, this research is relevant for several applications with humans in the loop. Since simulators are more and more relied upon for operator training, understanding human learning is useful in judging training effectiveness. First steps in this direction are identifying adaptation to systematic changes in reference signals and controlled dynamics (Mulder et al., 2013), or as practical example, motion feedback settings (Zaal and Pool, 2014). Furthermore knowing how humans adapt helps in designing conformant haptic shared control devices (Mulder et al., 2016).

In this thesis an attempt is made at identifying adapting human control behaviour online, i.e. in real time. Next to contributing to the previous research goals, doing this in real-time opens new opportunities. For example, in real-life control tasks reduced attention or distraction could be detected by continuously monitoring the operator by updating an operator model (Ameyoe et al., 2015). Furthermore this could enable adaptive haptic feedback, tracking the current behaviour of the operator (Olivari et al., 2014). In experiments conducted with humans in the loop, online identification can help in adjusting experimental conditions in real time to directly study adaptation behaviour or achieve desired haptic feedback qualities.

With the relevance of online time-varying system identification in manual control now in mind, the project proposal is as follows. Based on the current state-of-the-art in time-varying system identification, the aim of the research is to select a theoretically proven method from literature and implement it in an online fashion for a manual control task. The method could be tested in an example experiment. Next to an evaluation, the work will result in identification code useful for other experiments.

The preliminary thesis report is structured as follows. A literature survey in time-varying system identification methods in manual control is done in Chapter 2. From this a research objective to contribute to the state-of-the-art in cybernetics is derived in Chapter 3. This is continued by a preliminary analysis of the chosen method in simulations. The simulation setup is described in Chapters 4 and 5, with code verification in Chapter 6. A possible solution for the online identification problem is then discussed in Chapter 7. The further research steps needed are then discussed in Chapter 9.

2
Literature Survey

To get an overview of the state of the art in time-varying system identification in manual control, a literature survey is conducted. It treats the different human operator models and associated methods to identify them. This overview will help in formulating the research questions to answer, to achieve the aim of this research.

2.1. Modeling the human operator

In general, the human operator is a multichannel, adaptive, learning, non-linear controller in a control task. In the pilot-vehicle loop both a feedback path and feedforward path can be present as controller, based on visual and motion perception. The neuromuscular system is then used to move the control input device effecting the controlled element dynamics. These paths can be adapted by an internal mental representation of the task learned through experience (Mulder et al., 2016). This is illustrated in Figure 2.1. Four signals are indicated: the forcing function f, the compensatory error e, the control signal u and the controlled element position y.

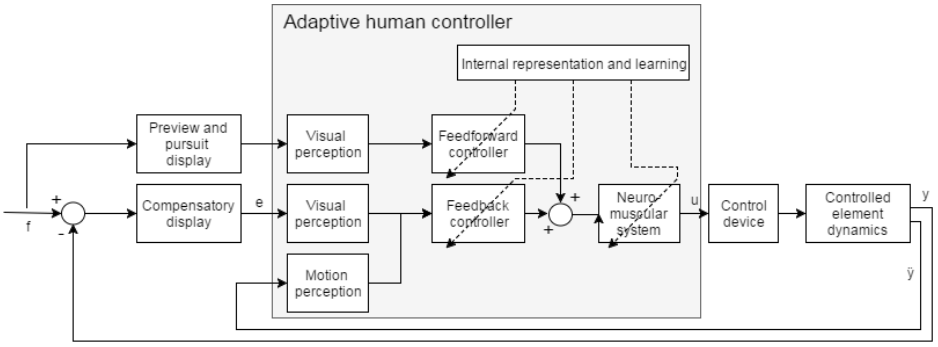


Figure 2.1: The multichannel, adaptive, learning human controller. Adapted from (Mulder et al., 2016).

A general control-theoretic model for the human controller has not been found yet, however, for specific control tasks validated models exist. An overview of human operator models in decades of research is described in (Lone and Cooke, 2014). For this research, the quasi-linear human operator models are most relevant and are focused upon in Section 2.1.1. Other perspectives are shortly discussed in Section 2.1.3.

2.1.1. Quasi-linear human operator models

Up to now the quasi-linear human operator models set up by (McRuer and Jex, 1967) have been most successful in modeling the human controller. (McRuer and Jex, 1967) postulated the crossover model, which is only validated for single channel tracking tasks with visual perception from a compensatory display when all learning is over and no adaptation occurs. This task is shown in Figure 2.2.

The crossover model captures the linear behaviour of the human controller in a describing transfer function $H_P(j\omega)$ and separates the further unexplained behaviour by adding noise called 'remnant' n. The remnant and its corresponding filter $H_n(j\omega)$ are further treated in Section 2.1.2. The model is purely based on input-output behaviour of the human operator. The quasi-linear operator model situated in a compensatory tracking task is shown in Figure 2.3.

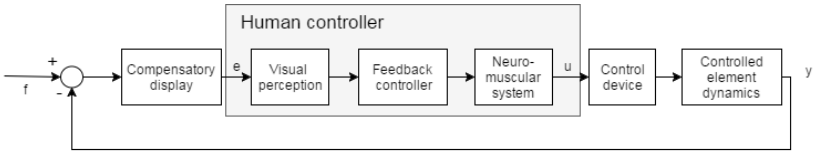


Figure 2.2: The human controller in a tracking task with only visual perception based on a compensatory display, no learning or adaptation included.

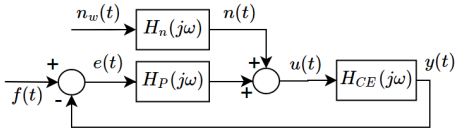


Figure 2.3: The quasi-linear human operator model in a compensatory tracking task.

The crossover model states that humans adjust their control behaviour such that Eq.(2.1) is satisfied in the crossover region when transient behaviour is eliminated. Then adjustment rules state both what the describing function $H_P(j\omega)$ is like according to the controlled element $H_{CE}(j\omega)$ and what its effect is in the frequency domain in terms of crossover frequency ω_c and time delay τ (McRuer and Jex, 1967).

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

The describing function is lumping together the operator's physical limitations and his equalisation. Eq.(2.2) is a general formulation of the describing function, to be adjusted according to the controlled element using the adjustment rules.

Operator limitations are modelled as a lumped time delay for visual perception and information processing. In addition the neuromuscular system (NMS) is modelled as a second order transfer function. Note that this component was added later to McRuer's work. Operator equalisation is modelled as a gain and a lead/lag component, this represents the feedback controller.

$$H_P(j\omega) = K_p\frac{1+\tau_L j\omega}{1+\tau_l j\omega}e^{-j\omega\tau}\frac{\omega_{nm}^2}{\omega_{nm}^2+2\zeta_{nm}\omega_{nm}j\omega+(j\omega)^2} \tag{2.2}$$

For completeness in this survey, it should be mentioned that another quasi-linear operator model is the 'descriptive model'. It focuses more on the physiological human sensors for perception in manual control (Hosman and Stassen, 1999), in order to better understand them. This model is not further considered in this work.

2.1.2. Remnant noise filters

There is limited theoretical background on how to model the remnant. To have realistic simulations, an assumption for the remnant is necessary. The remnant assumption influences the bias and variance found while identifying parameters in simulation. Generally remnant is generated by passing Gaussian, zero mean white noise through a filter. Different filter choices are found in literature. Furthermore the location where the noise is inserted in the loop is a choice.

(Zaal, 2016) assumes a first order low pass filter as in Eq.(2.3), with a gain K_n tuned to achieve a certain remnant intensity as defined in Eq.(2.4). The noise is injected in the control signal. On the other hand, (Levison et al., 1969) also derived a first order low pass filter, but injected the noise in the error signal.

$$H_n(j\omega) = \frac{K_n}{(0.2j\omega+1)} \tag{2.3}$$

$$P_n = \frac{\sigma_n^2}{\sigma_u^2} \tag{2.4}$$

A remnant filter can also be derived from experimental data. (Zaal et al., 2009) compared the modelled control signal with the actual measured control signal obtained during a compensatory tracking experiment. The difference between those signals gives a time series for the remnant. By calculating the power spectrum of this time series, a third order low pass filter was fitted resulting in Eq.(2.5), with $\omega_n = 12.7$ rad/s and $\zeta_n = 0.26$. The gain K_n can again be tuned and the noise is injected in the control signal.

$$H_n(j\omega) = \frac{K_n\omega_n^3}{((j\omega)^2 + 2\zeta_n\omega_n j\omega + \omega_n^2)(j\omega + \omega_n)} \tag{2.5}$$

2.1.3. Other perspectives

Next to the quasi-linear model above, two other perspectives are common. The 'optimal control model' sees the operator as a controller minimising a cost functional to derive a feedback law, while perceiving the environment is modelled as a Kalman filter (Kleinman et al., 1970). Finally, Hess set up the 'structural operator model', a non-linear model containing switches to change between different operator behaviour in different control situations (Hess, 2006). They could be useful in time-varying operator identification research, but are not often used up to now.

2.1.4. Conclusion

As seen in the next section, most research in time-varying system identification in manual control is done using formulations of the quasi-linear operator model. Reasons for this are its easy parametrisation in physically understandable parameters and its validated use in certain tasks. Therefore only the quasi-linear operator model will be considered further.

2.2. Identifying time-varying operator behaviour

Research in identifying time-varying operator models can be classified according to the method used. The found methods will be judged on their usability online and how accurate they are. A scheme of the considered papers is shown in Figure 2.4.

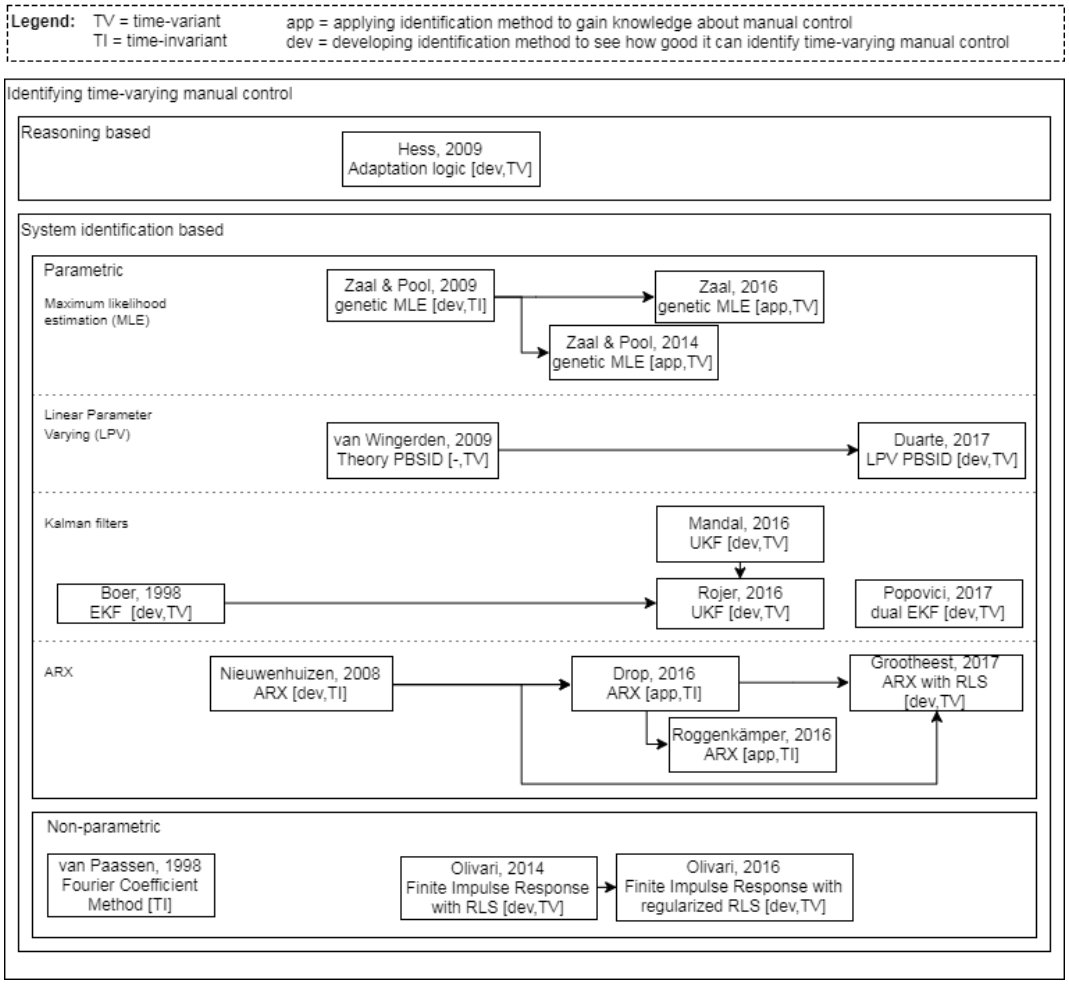


Figure 2.4: Overview of covered studies in the time-varying system identification methods literature survey.

Note that two different perspectives on identifying the adaptation were found. In (Hess, 2009) the adaptation is specified as a logic rule, derived empirically by the author. All other papers considered follow the system identification approach of measuring input-output signals to model the system.

2.2.1. Parametric versus non-parametric approach

In system identification two approaches can be followed. The non-parametric approach only estimates a frequency response directly.

In (Olivari et al., 2014) and (Olivari et al., 2016) first a discrete Finite Impulse Response (FIR) is estimated, which is then converted to a frequency response estimated by the Fourier Transform. This is a two-step method. The FIR estimation is done by recursive least squares (RLS), enabling tracking of time-variations in the system response. Another non-parametric method is the method of Fourier coefficients in (van Paassen and Mulder, 1998). A transfer function is estimated only at the frequencies of the forcing function and is derived analytically. The method is used for time-invariant applications.

All other papers considered are using a parametric approach. The operator is seen as a grey box: a model structure is assumed and its parameters need to be estimated. This approach gives more physical insight into the human controller than non-parametric methods.

2.2.2. Parametric methods

The parametric methods are classified further based on the assumed model structure and the method used for parameter estimation. Four research lines were found in literature. The quasi-linear operator model, with different formulations of the describing function and remnant, is consistently chosen in the studied papers. To force time-varying operator behaviour almost all studied papers change the controlled element dynamics during either simulation or experiments. The following other factors set the methods apart, and these are discussed for each.

- **Allowed adaptation:** To model the time-varying operator behaviour, different assumptions are made regarding how this adaptation is shaped and which operator model parameters can vary in time.
- **Identifying time delay:** There is an additional challenge in identifying time-varying human operator time delays in the assumed models, treated differently by different authors.
- **Type of fitting:** One can fit the assumed models on all data at once, or fit the models recursively.

2.2.2.1 Batch fitting methods

Maximum likelihood estimation: A time-domain technique where the model parameters are estimated using genetic maximum likelihood estimation is used in (Zaal et al., 2009), (Zaal and Pool, 2014) and (Zaal, 2016). Time variation in the operator model is assumed to follow a sigmoid shape which needs to be fitted. Only operator equalisation parameters are allowed to vary while the operator limitation parameters are assumed constant, including the operator time delay. This might not be the case in certain operator adaptation situations. This is a batch fitting method, meaning that the fitting acts on the whole dataset at once.

Fitting LPV state space systems: In (Duarte et al., 2017) a linear parameter varying (LPV) system is identified using the Predictor-Based Subspace Identification (PBSID) algorithm from (van Wingerden and Verhaegen, 2009). For the LPV model a comparison is made between analytically and experimentally determined scheduling functions which need to reflect the human operator adaptation. Again only operator equalisation parameters are assumed to vary.

The time delay is treated separately from the other parameters. Low order LPV models with increasing time delays are fitted and the one with the best quality-of-fit is selected as delay estimate. In the next identification steps, the input signal is shifted with this delay estimate.

2.2.2.2 Recursive fitting methods

Kalman filter estimation: In several studies Kalman filters are used (Boer and Kenyon, 1998) (Rojer et al., 2016) (Mandal and Gu, 2016) (Popovici et al., 2017). They update the parameters every timestep based on predictions and measurements. Fitting is done recursively, enabling tracking of changes in the parameters.

No assumption has to be made regarding the shape of variation the parameters follow and time delay needs no special treatment. The observation equations use a state-space form of the describing transfer function with an nth order Padé approximation for the time delay component.

In (Popovici et al., 2017) a dual Extended Kalman Filter (EKF) is used. A state filter estimates the equalisation dynamics, and a parameter filter estimates the neuromuscular parameters and time delay. Thus all parameters are assumed to be time-varying.

In (Rojer et al., 2016) and (Mandal and Gu, 2016) the Unscented Kalman Filter (UKF) is used. Operator equalisation parameters are assumed to be varying, as well as the time-delay limitation parameter. The other limitation parameters, i.e. of the NMS, are not allowed to vary.

Fitting recursive ARX models: (van Grootheest, 2017) applies ARX models to identify time-varying human operator behaviour. This study extends the work in (Nieuwenhuizen et al., 2008), (Drop et al., 2016) and (Roggenkämper et al., 2016) to the time-varying case. The ARX coefficients are determined using Recursive Least Squares (RLS). Again no underlying assumption regarding the shape of variation has to be made. Operator equalisation parameters are varying, the limitation parameters are assumed constant.

The time delay is assumed constant regardless of the adaptation and can only be an integer multiple of the sample time. Just like in (Duarte et al., 2017), the time delay is estimated a-priori using models with different assumed delays. A-priori means before performing the actual model fitting, resulting in a shifted input signal to take the estimated delay into account.

2.2.3. Conclusion

The aim of this research is to identify time-varying human operator behaviour online. To track these changes online, the recursive fitting methods are suitable. Furthermore methods which do not assume a set shape for the adaptation are more generally applicable. Thus Kalman filtering and fitting of recursive ARX models are promising methods. To further delineate the current literature survey, a choice is made to focus on the recursive ARX method. Its current issues are investigated.

2.3. The recursive ARX method

To build further on the work of (van Grootheest, 2017), a summary is given of the research done and open issues treated. The goal was to find out whether ARX models could be used to model a time-varying human controller by recursive estimation. First the model structure will be described, then the major conclusions regarding the use of ARX to model the human controller are discussed.

2.3.1. ARX model structure and order

The ARX structure is part of a general class of linear models. It is formulated as in Eqs.(2.6)-(2.8). The time shift operator is indicated by q, such that $q^{-1}u(t) = u(t-1)$. The $A(q)$ polynomial acts on the control signal $u(t)$. The $B(q)$ polynomial acts on the error signal $e(t-n_k)$, which could be delayed by n_k timesteps or samples to represent an input-output delay. $\epsilon(t)$ represents a white noise signal. The polynomials are determined by n_a and n_b respectively, indicating the model order.

$$A(q)u(t) = B(q)e(t-n_k) + \epsilon(t) \tag{2.6}$$

$$A(q) = 1 + a_1 q^{-1} + a_2 q^{-2} + ... + a_{n_a} q^{-n_a} \tag{2.7}$$

$$B(q) = b_0 + b_1 q^{-1} + ... + b_{n_b} q^{-n_b+1} \tag{2.8}$$

Rewriting the ARX model in terms of the control output gives Eq.(2.9). The describing function is modelled by the ratio $\frac{B(q)}{A(q)}$ and delay n_k. For straightforward conversion of the ARX model back to the human operator transfer function, $n_a = 2, n_b = 2$ is most suited since in that case no model order reduction is required. The polynomial coefficients and the time delay parameter n_k need to be identified.

$$u(t) = \frac{B(q)}{A(q)}e(t-n_k) + \frac{1}{A(q)}\epsilon(t) \tag{2.9}$$

The model was identified using simulations and experiments similar to the ones set up in (Zaal, 2016). The direct system identification approach was followed, meaning that no correction was done for closed-loop issues in identification. This is allowed according to (Drop et al., 2016).

2.3.2. ARX time delay estimation

The time delay is represented by the n_k parameter as in Eq.(2.10). The time delay was always estimated a-priori because it cannot be included in the least squares used in Section 2.3.3. To estimate the time delay, a set of ARX models is fitted using Ordinary Least Square (OLS) before the actual identification process takes place. Each has a different n_k parameter. The n_k is selected which gives the smallest loss function according to Eq.(2.11). This is the average squared error between the measured control signal $u_{meas}(t)$ and the control signal $u_{pred}(t)$ predicted using the fitted ARX model. Now the error signal is shifted with the estimated $n_{k_{est}}$. The actual identification now takes place with the time-shifted error signal as input and the ARX model is set up with $n_{k_{est}}$.

$$n_k = \frac{\tau}{\Delta t} \tag{2.10}$$

$$V = \frac{1}{N}\sum_{t=1}^{N}(u_{meas}(t) - u_{pred}(t))^2 \tag{2.11}$$

2.3.3. Recursive ARX identification

After delay estimation, decimation was used on the signals to get rid of high-frequency components above the range of interest. With decimation only every dth sample is kept, effectively decreasing the sampling frequency. Different decimation factors d were tried, and best results achieved with $d = 16$. After preprocessing, identification of the ARX model was done for both the time-invariant and time-varying human controller.

In the time-varying case, the fitting of the ARX model needs to be done recursively to enable tracking of changes. This means the ARX coefficients become time-varying. RLS with the concept of forgetting was applied to estimate the coefficients at each timestep. The forgetting factor was tuned to enable good tracking of parameter changes while not being too sensitive to noise. A forgetting factor of $\lambda = 0.99609$ gave best results when sampling at 100Hz. An example estimation run is given in Figure 2.5.

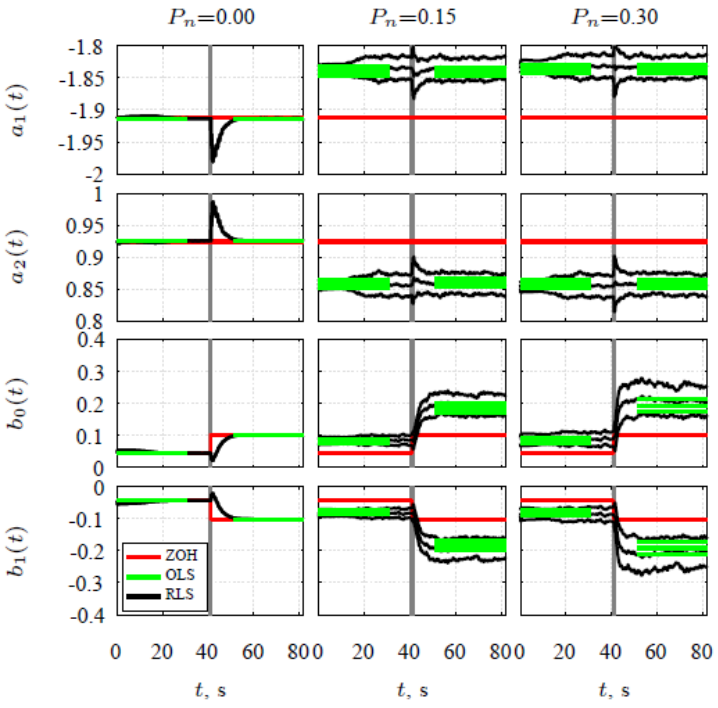


Figure 2.5: Tracking performance of recursive ARX identification for the ARX model coefficients. The simulation according to (Zaal, 2016) contains equalisation adaptation at $t = 50s$, but a constant time delay. A second order remnant filter was assumed, introducing a bias. The ZOH line is the analytical reference value, the OLS and RLS lines are the estimates with their standard deviations. Obtained from (van Grootheest, 2017).

The ARX model is a discrete time transfer function estimate. Thus a conversion of the coefficients to continuous time is necessary. Presumably the Matlab *d2c* command was used with the zero order hold method. Next, the continuous time ARX coefficients are converted to the operator parameters by solving a system of equations, obtained by comparing the continuous time transfer function and the operator model.

2.3.4. Influence of remnant

The $A(q)$ polynomial is modelling both the describing function and the remnant noise in Eq.(2.9). The remnant is modelled as if the white noise $e(t)$ is filtered with $\frac{1}{A(q)}$. Thus any identification method will give biased estimates since the process dynamics and noise dynamics are coupled.

Simulations were done with remnant filters of different order m according to Eq.(2.12), to study their effect on the bias. Furthermore a remnant with the same transfer function as the human neuromuscular response was used. Each filter was tuned to induce the same remnant intensity, the remnant gain was made time-varying to ensure the intensity remained the same throughout the adaptation simulation.

$$H_n^m = \frac{K_n(t)}{(T_n j\omega + 1)^m} \tag{2.12}$$

The highest bias for delay and ARX coefficients occurred when assuming a first order filter as indicated in Figures 2.6-2.7. The parameter estimation was less biased when the remnant filter order was larger than the ARX model order, and especially least biased if the filter was equal to the neuromuscular system response.

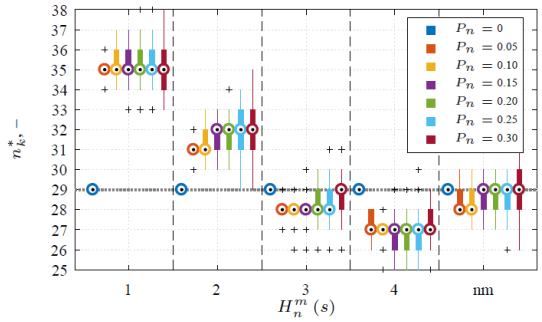


Figure 2.6: Bias in estimated time delay parameter n_k, for different remnant filter orders and intensity levels. Obtained from (van Grootheest, 2017).

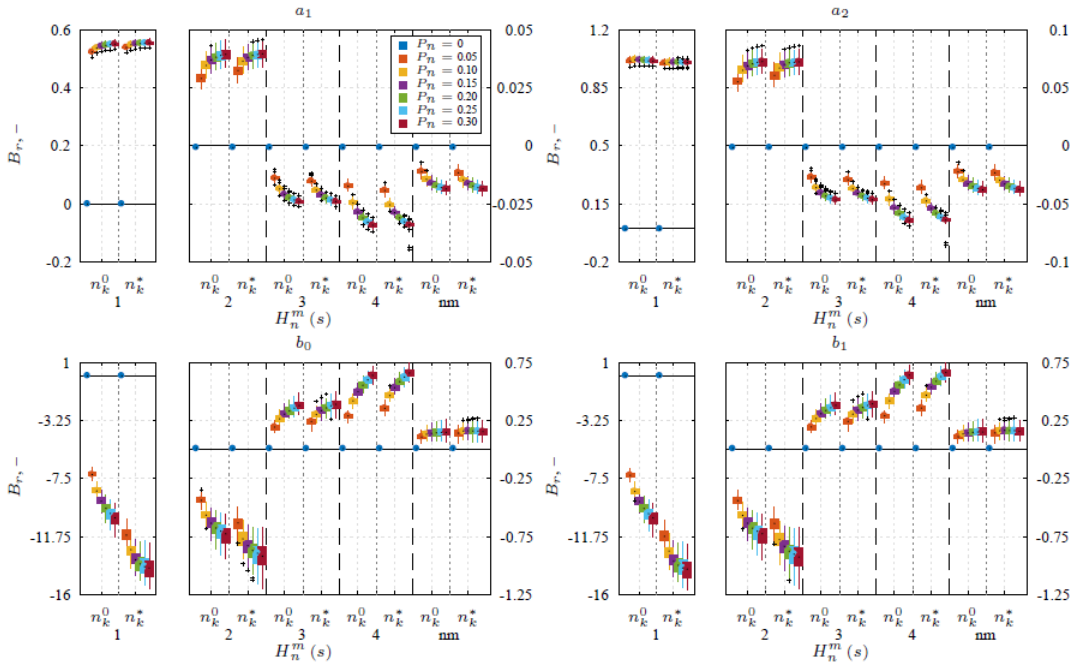


Figure 2.7: Bias in ARX model coefficients, for different remnant filter orders and intensity levels. Obtained from (van Grootheest, 2017).

In experiments with human operators, a definitive answer on how to model the remnant is not known. Thus the bias introduced by the remnant is unknown. Therefore a model structure without coupling between process and noise dynamics could give superior estimation results than with ARX models.

2.3.5. Conclusion

The work of (van Grootheest, 2017) results in three main issues regarding time-varying operator model identification if it should be implemented online. First a way of converting the discrete time ARX coefficients to the operator parameters, without relying on opaque Matlab commands, should be found. Secondly an alternative linear model structure could avoid introducing bias due to remnant, for example Box-Jenkins models, if it is suitable for online implementation. Finally an alternative time delay estimation procedure suitable for online implementation should be developed if this method is selected for further research.

2.4. Identifying time-varying time-delay online

Estimating a time-varying time delay is challenging since it is a non-linearity in the system. This is especially difficult in online estimation in this research. The considered solutions inside cybernetics are the work of (Boer and Kenyon, 1998), (Rojer et al., 2016) and (Popovici et al., 2017). More general solutions inside the system identification field are also considered as in (Björklund, 2003), (Anderson and Moore, 1979), (Zhao et al., 2017) and (Tan, 2004). Computational effort and convergence are important properties to judge the methods, even though they can only be qualitatively judged at this stage.

2.4.1. Solutions in cybernetics field

- In most considered cybernetics papers up to now, the human operator time delay is assumed constant. Even then identification is often not directly possible and the time delay should be estimated before other parameters as discussed earlier for (Duarte et al., 2017) and (van Grootheest, 2017). This is not suitable for online implementation.

- In (Boer and Kenyon, 1998) the EKF is successfully used to directly estimate a time-varying time delay while simultaneously identifying the coefficients of an ARMAX model. The developed Recursive Delay Identifier (RDI) even allows for fractional delay time estimates using interpolation. The approach was tested in two different experiments which specifically aim at inducing time-varying time delay in the human controller: a time-varying gap-postview tracking task and Jex's critical tracking task. The latter consists of a controlled element with growing instability that needs to be stabilised by the human controller (Jex et al., 1966).

- Alternatively the pure time delay could be replaced by an nth order Padé approximation in the operator model, allowing direct Kalman filter estimation for the operator parameters and the time delay as in (Rojer et al., 2016) with the UKF and (Popovici et al., 2017) with the dual EKF.

2.4.2. Solutions in system identification field

In system identification, the following solutions are identified. An overview of time delay estimation methods in time and frequency domain is given in (Björklund, 2003).

- A promising mentioned method is to estimate many different models in parallel with each one assuming a different time delay. The best fitting model is selected at each moment in time according to a quality-of-fit metric. During online estimation this should be done recursively. This 'adaptive estimation through parallel processing' strategy is also described in (Anderson and Moore, 1979). This method could be directly applied to recursive ARX, and is further referred to as parallel recursive ARX. A low and scalable computational load is expected due to the simple RLS parameter estimation.

  An extension to this approach is proposed in (Björklund, 2003). It counters the unavoidable bias introduced in the estimates because of the interaction between noise and the ARX model structure. At first an ARMAX or OE model is fitted to get an idea of the noise model, which will then be used to prefilter the signals used in the ARX model fitting. In an online setting this noise model could then be updated once in a while.

- In (Zhao et al., 2017) the identification of time-varying industrial processes with delay is studied. A hidden Markov chain model is used for the delay, while ARX is used as the process model. The Markov

chain is deemed appropriate because time delays do not jump around randomly but could follow a certain probabilistic switching mechanism, they are correlated in time. This is expressed in the identified transition probabilities between possible time delay values. The identification of both the Markov chain and the model parameters is done using an iterative, recursive Expectation Maximisation (EM) algorithm. Because of the iterative nature of the algorithm, it is unclear how computationally intensive it will be in new applications and whether online implementation is feasible.

- Another solution is based on neural networks in (Tan, 2004). A neural network is specifically trained to estimate a time-varying time delay. Inputs to the neural network are previous delay estimates and errors between actual and predicted output. The training determines the weights given to each of these inputs. Accurate results are achieved with rapid convergence, however a high computational load is seen. This could be reduced once the training procedure is finished, since then no more optimisation must be done online.

2.4.3. Conclusion

Several solutions for time delay estimation, in combination with identifying the other human operator parameters, have been found in the cybernetics and system identification fields. The open issue with recursive ARX regarding time delay estimation can be mitigated by using a parallel processing strategy. Some properties have been listed of each solution, useful in the selection process to obtain the most suitable method for this research.

2.5. Method trade-off

To determine the research direction, a trade-off is performed between the identification methods found in the literature survey. The chosen method will lead to the research objective and questions stated in Chapter 3. The criteria used in the trade-off in Table 2.1 are judged based on the respective papers of the methods, if such results are available. The four criteria are as follows:

- Recursive updating: whether the method can be updated recursively and if other constraints are present.
- Convergence: whether the parameter estimation converges easily and is robust.
- Computational complexity: how much computational power is needed.
- Estimation bias human operator (HO) parameters: bias magnitude as found in previous work.

From the table, the UKF, dual EKF and parallel recursive ARX methods seem the best recursive methods to identify the complete human operator. RDI EKF has no information about the achieved bias and is only focused on estimating delay. Furthermore, by analysing the papers, the dual EKF seems more straightforward to implement than the UKF with limited performance differences, and will be the only retained Kalman filter option. The biggest disadvantage of the dual EKF is the heavy dependence on choosing the initial condition close enough to the actual one to obtain convergence. The parallel recursive ARX method is more robust to this.

Regarding computational complexity the parallel recursive ARX method might be more expensive since many models need to be estimated in parallel. For dual EKF only one model is estimated. The estimation bias for recursive ARX is found to be higher than the bias with dual EKF when comparing their respective paper results. However the time delay bias is still unknown for parallel recursive ARX and could be examined in this research.

To conclude, first the parallel recursive ARX method will be tried to find out its time delay estimation performance as subject of this preliminary thesis. Later a complete comparison can then be done with the dual EKF method and the superior option can be selected for further experimental research.

Table 2.1: Methods suitable to identify the time-varying human operator, including delay, are judged according to four criteria in this table. Colours indicate their score: red = poor, yellow = mediocre, green = good.

Method	Criteria			
	Recursive updating	Convergence	Computational complexity	Estimation bias HO parameters
RDI EKF (Boer and Kenyon, 1998)	Yes	Less dependent on initial conditions	Low	Unknown
UKF (Rojer et al., 2016)	Yes	Heavily dependent on initial conditions	Low	Low
Dual EKF (Popovici et al., 2017)	Yes	Heavily dependent on initial conditions	Low	Low
Parallel recursive ARX (van Grootheest, 2017) (Anderson and Moore, 1979)	Yes	Less dependent on initial conditions	Medium	Unknown for delay, higher for other parameters
Parallel recursive ARX with prefilter (Björklund, 2003)	Yes for ARX, not for prefilter	Less dependent on initial conditions	Medium	Unknown
Markov Chain for delay, ARX for process (Zhao et al., 2017)	Iterations required at each step	No guarantee	Potentially high	Low for delay, higher for other parameters
Neural network for delay, any other method for process (Tan, 2004)	Retraining at each step optional	No guarantee	Potentially high	Low for delay, unknown for other parameters

3

Research Objective and Questions

Online identification has specific requirements for the chosen identification method. The computation time should be low enough to have a sufficiently high update frequency of parameter estimates. Furthermore the method should allow for recursive updating of the model as new data points come in.

From the trade-off concluding the literature survey, a promising method to implement online is to use recursively identified ARX models as treated in (van Grootheest, 2017). Currently the human operator time delay is estimated a-priori to the ARX model fitting and is assumed constant. Time delays can be time-varying in real life and they cannot be estimated a-priori in an online setting. Thus a modification of the recursive ARX method is required. The research objective is set as follows:

The research objective is to contribute to the modeling of the adapting human controller by modifying the recursive ARX method to estimate in an online fashion both equalisation parameters and time delay of a quasi-linear pilot model, during a human-in-the-loop control task experiment forcing adaptive behaviour.

The research objective leads to the following research questions, to be investigated in the subsequent research. The main question is as follows:

Is it possible to use recursive ARX model fitting to identify the human operator equalisation adaptation online during a control task experiment, while also modifying the method to simultaneously estimate the operator's time delay?

The literature survey concluded that the parallel recursive ARX method is a good first try for the modification. This leads to the following specific sub-questions:

1. Regarding the implementation of the method in simulation:
   (a) What simulation setup allows to try out the parallel recursive ARX method?
   (b) Which human operator and controlled element models can be chosen?
   (c) Which human operator adaptation could be introduced in the simulation?
   (d) How should the remnant noise be introduced in the simulation?
   (e) After fitting, which conversion steps are needed to retrieve the operator model parameters from the ARX coefficients?
   (f) What verification tasks can be completed to make sure the simulation is done correctly?
2. Regarding the testing of the method in simulation:
   (a) Which quality-of-fit metric can be used to select the best fitting ARX model at each timestep?
   (b) What is the best sliding window size for such a metric to enable tracking of the model fit?
   (c) What is the influence of the remnant filter choice on the time delay estimation results?
   (d) What is the influence of sampling frequency on the time delay estimation results?
   (e) What are the computational limits of the method? Can an estimate of its execution time be made?

3. Regarding experimental evaluation of the found solution:
    (a) Which kind of experiment will allow to find out the tracking performance and limits of this method?
    (b) Which kind of experiment forces the human operator to significantly adapt his time delay?
    (c) What must be done to implement the chosen method in the research simulator framework?
    (d) Is the tracking of time delay sufficiently accurate and fast?

To answer these questions, the following tangible steps are taken in this preliminary thesis. First the recursive ARX method is implemented in a time-varying human operator simulation. Then the time-varying delay estimation problem is solved and its performance is evaluated. Finally conclusions can be made about the method's usability in order to continue the research later with experimental evaluation.

4

Human Controller Simulation Setup

In order to try out time delay identification strategies, a simulation placing the human operator in a control task is needed. A single-axis, single channel compensatory tracking task is simulated based on (Zaal, 2016). A quasi-linear pilot model is used. The simulation loop is the control task indicated in Figure 2.3. To force time-varying human operator behaviour, the controlled element dynamics change from single to double integrator. The operator will adapt itself accordingly.

The simulation is implemented in Python 3.6.0. This requires basic understanding of each element in the simulation. Matlab would offer straightforward commands for certain elements. However, when implementing the identification strategy (Chapters 5 and 7) in the experiment software platform, the in-depth understanding gained by using a Python simulation will be useful.

In this chapter the simulation loop is described in terms of its input signals, the used models and how time-varying behaviour is introduced. To repeat simulation runs involving noise realisations, a Monte-Carlo implementation is described. Finally a flow chart gives an overview of the steps taken.

4.1. Input signals and simulation time

Two input signals are needed for a simulation run: a forcing function and a remnant realisation. A run has length $T = 100s$, with a measurement time of $T_m = 81.92s$, conveniently chosen for Fourier transforming signals with minimal leakage. A timestep of $\Delta t = 0.01s$ was chosen for the simulation, corresponding to a sampling frequency of 100 Hz.

4.1.1. Forcing function

A forcing function is used to excite the closed loop system consisting of a pilot and controlled element model. It is a summation of N_f sinusoids with different amplitudes $A_f[k]$, frequencies $n_f[k]\omega_m$ and phases $\phi_f[k]$, as in Eq.(4.1). The frequencies are integer multiples of the base measurement frequency $\omega_m = 2\pi/T_m$. The values used are listed in Table 4.1. This is the same forcing function as in (Zaal, 2016).

$$f(t) = \sum_{k=1}^{N_s} A_f[k]\sin(n_f[k]\omega_m t + \phi_f[k]) \tag{4.1}$$

Table 4.1: Forcing function properties used, adapted from (Zaal, 2016).

k_f [-]	n_f [-]	$n_f\omega_m$ [rad/s]	A_f [deg]	ϕ_f [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	-2.189
8	86	6.596	0.061	-2.189
9	139	10.661	0.036	0.875
10	229	17.564	0.025	0.604

4.1.2. Remnant realisation

A remnant realisation is generated for each single simulation run by passing zero mean, unit variance, Gaussian white noise through a filter. Different remnant filters are taken into account since ARX models are partially modelling the noise next to the dynamic system. This way the influence of remnant filter choice on identification performance can be studied. The injection point for the remnant noise was chosen to be the control output signal.

Three filters of different order are mentioned in Table 4.2. The first order filter is from (Zaal, 2016). The second order filter is mentioned in (van Grootheest, 2017), and was found to give the least bias in ARX model identification. The third order filter is also chosen because it was experimentally validated in (Zaal et al., 2009) as resembling the operator remnant noise process.

The remnant gain is tuned to achieve a certain remnant intensity. Tuning was done manually by averaging the intensity of ten simulation runs and then adjusting the gain iteratively until the target intensity was reached. This was done for both the single and double integrator controlled element case for each filter. Note that the actual intensity obtained will differ between runs due to the randomness of the noise realisations. The gains to obtain $P_n = 0.15$ are in Table 4.2.

Table 4.2: Remnant gains for different filters and controlled elements to achieve $P_n = 0.15$.

Remnant filter $H_n(j\omega)$	Controlled element $H_{CE}(j\omega)$	Gain K_n
$\frac{K_n}{(0.2j\omega+1)}$	K/s	0.205
	K/s^2	0.52
$\frac{K_n(t)}{(0.06j\omega+1)^2}$	K/s	0.16
	K/s^2	0.39
$\frac{K_n\omega_n^3}{((j\omega)^2+2\zeta_n\omega_n j\omega+\omega_n^2)(j\omega+\omega_n)}$, $\omega_n = 12.7, \zeta_n = 0.26$	K/s	0.117
	K/s^2	0.275

In time-varying operator behaviour simulations, explained in the next section, the remnant gain immediately switches from one gain to the other at half of the transition period. This approximation becomes more accurate when the scheduling function approaches a step function.

4.2. Time-varying human operator and controlled element models

To simulate a manual control task a human operator model and controlled element model are required. The parameters of those models are made time-varying according to a sigmoid scheduling function as in Eq.(4.2), to enable time-varying human operator behaviour simulations. P_1 and P_2 indicate the initial and final parameter value respectively. The time of maximum rate of change is set by M and the maximum rate of change itself by G.

$$P(t) = P_1 + \frac{P_2 - P_1}{1 + e^{-G(t-M)}} \tag{4.2}$$

4.2.1. Human operator model

The chosen human operator model is the describing function from the McRuer quasi-linear pilot model as in Eq.(2.2). Note that this model was only validated for the time-invariant operator in (McRuer and Jex, 1967). In this thesis it is assumed the model can be extended to time-varying operators, where the model parameters change over time.

A modification is done in the parametrisation of the model. For controlled elements of the studied control task, the lag component can be removed according to the crossover model (McRuer and Jex, 1967). Furthermore $K_p(t)$ and $\tau_L(t)$ are replaced by a proportional and derivative error gain $K_e(t) = K_p(t)$ and $K_{\dot{e}}(t) = K_p(t)\tau_L(t)$ respectively as in Eq.(4.3). This avoids an ambiguity in identification between $K_p(t)$ and $\tau_L(t)$.

$$H_P(j\omega, t) = (K_e(t) + K_{\dot{e}}(t)\, j\omega)e^{-j\omega\tau(t)} \frac{\omega_{nm}^2(t)}{\omega_{nm}^2(t) + 2\zeta_{nm}(t)\omega_{nm}(t)\, j\omega + (j\omega)^2} \tag{4.3}$$

For simulation the transfer function needs to be converted to state-space format. The introduced states have no physical meaning however. The time delay $e^{-j\omega\tau(t)}$ is ignored for the conversion and will be implemented

separately. The resulting continuous time state-space system is given in Eq.(4.4)-(4.5). Note that the system matrices are time-varying.

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \underbrace{\begin{pmatrix}0 & 1\\ -\omega_{nm}^2(t) & -2\zeta_{nm}(t)\omega_{nm}(t)\end{pmatrix}}_{A_c(t)}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + \underbrace{\begin{pmatrix}0\\ 1\end{pmatrix}}_{B_c(t)} e \tag{4.4}$$

$$u = \underbrace{\begin{pmatrix}K_e(t)\omega_{nm}^2(t) & K_{\dot{e}}(t)\omega_{nm}^2(t)\end{pmatrix}}_{C_c(t)}\begin{pmatrix}x_1\\ x_2\end{pmatrix} \tag{4.5}$$

The continuous time state space system could be numerically integrated over time. However, during verification it was found that the simulation needs to happen with discrete time models as this maintains consistency with the discrete identification method, and therefore avoids unwanted biases. Thus the time-varying system needs to be discretised each timestep. In Matlab one would use the *c2d* command, in Python this procedure was implemented by the author as in Figure 4.1. Both A_c and B_c matrices are discretised, resulting in system Eq.(4.6). The C_c matrix remains the same.

$$\dot{\bar{x}} = A_d[i]\bar{x} + B_d[i]e \qquad u = C_c[i]\bar{x} \qquad \forall i = 0,..,N-1,N \tag{4.6}$$

```
import numpy as np
import numpy.matlib as mt
import scipy.linalg as la

def SS_c2d(A,B,dt):
    [m,n] = np.shape(A)
    [m,nb] = np.shape(B)

    top = np.hstack((A,B))*dt
    bot = mt.zeros((nb,n+nb))
    arg = np.vstack((top,bot))

    s = la.expm(arg)

    A_discrete = mt.matrix(s[0:n,0:n])
    B_discrete = mt.matrix(s[0:n,n:n+nb])
    return [A_discrete, B_discrete]
```

Figure 4.1: Python implementation of Matlab's *c2d* for discretising state space matrices.

With the discrete time state space system, the undelayed control output is calculated. Delay is taken into account by index shifting the control output by a corresponding number of timesteps as in Eq.(4.7). Index shifts can only be integers. This is an exact representation of delay, as long as the delay is an integer multiple of the sample time. The index shift is varying in size as the time delay is varied.

$$u_d[i] = u[i - \text{int}\left(\frac{\tau[i]}{\Delta t}\right)] \qquad \forall i = 0,..,N-1,N \tag{4.7}$$

4.2.2. Controlled element model

The controlled element transfer function is given in Eq.(4.8). It is capable of representing both single and double integrator dynamics by varying the break frequency $\omega_b(t)$. For simulation it is converted to the state space system in Eq.(4.9)-(4.10). Again a discretisation is done at each timestep using the algorithm in Figure 4.1 for the E_c and F_c matrices. The discretised state space system for the controlled element is then given by Eq.(4.11).

$$H_{CE}(j\omega) = \frac{K_c(t)}{(j\omega)^2 + \omega_b(t)\, j\omega} \tag{4.8}$$

$$\begin{pmatrix}\dot{z}_1\\ \dot{z}_2\end{pmatrix} = \underbrace{\begin{pmatrix}0 & 1\\ 0 & -\omega_b(t)\end{pmatrix}}_{E_c(t)}\begin{pmatrix}z_1\\ z_2\end{pmatrix} + \underbrace{\begin{pmatrix}0\\ 1\end{pmatrix}}_{F_c(t)} u \tag{4.9}$$

$$y = \underbrace{\begin{pmatrix}K_c(t) & 0\end{pmatrix}}_{H_c(t)}\begin{pmatrix}z_1\\ z_2\end{pmatrix} \tag{4.10}$$

$$\dot{\bar{z}} = E_d[i]\bar{z} + F_d[i]u \qquad y = H_c[i]\bar{z} \qquad \forall i = 0,..,N-1,N \tag{4.11}$$

4.3. Simulation conditions

Several simulation scenarios are considered in the following work, corresponding to different time-variant and time-invariant human operator behaviours. Furthermore a distinction is made between remnant noise active or not. If noise is active, different remnant filters are possible. These choices are encoded in the condition ID CX-NX-X as in Eq.(4.12).

$$\underbrace{C1}_{\text{Condition}} - \underbrace{N1}_{\text{Noise active (N1) or not (N0)}} - \underbrace{1}_{\text{Filter order}} \tag{4.12}$$

The studied conditions are described in Table 4.3. They are simulated for all remnant filters. The corresponding parameters for the human operator model and controlled element model are shown in Table 4.4, both before (P_1) and after (P_2) adaptation following the sigmoid scheduling function. Conditions C1, C2 and C3 are obtained from (Zaal, 2016), they were determined experimentally. In C4 the time delay adaptation is chosen by the author, with the constraint to keep the simulation stable. Condition C5 is then a combination of the adapting operator in both equalisation and time delay.

Table 4.3: Descriptions of studied simulation conditions and their scheduling function.

Condition	Description	Scheduling function	
		$G\ [s^{-1}]$	$M\ [s]$
C1	Time-invariant operator, single integrator controlled element	-	-
C2	Time-invariant operator, double integrator controlled element	-	-
C3	Time-variant operator in equalisation, single to double integrator controlled element	0.5	50
C4	Time-variant operator in time delay, single integrator controlled element	10	50
C5	Time-variant operator in equalisation and time delay, single to double integrator controlled element	10	50

Table 4.4: Parameters for operator and controlled element models as function of condition number.

	Human operator										Controlled element			
	$K_e\ [-]$		$K_{\dot{e}}\ [-]$		$\tau\ [s]$		$\omega_{nm}\ [rad/s]$		$\zeta_{nm}\ [-]$		$K_c\ [-]$		$\omega_b\ [rad/s]$	
	1	2	1	2	1	2	1	2	1	2	1	2	1	2
C1	0.09	-	0.036	-	0.28	-	11.25	-	0.35	-	90	-	6	-
C2	-	0.07	-	0.084	-	0.28	-	11.25	-	0.35	-	30	-	0.2
C3	0.09	0.07	0.036	0.084	0.28	0.28	11.25	11.25	0.35	0.35	90	30	6	0.2
C4	0.09	0.09	0.036	0.036	0.28	0.32	11.25	11.25	0.35	0.35	90	90	6	6
C5	0.09	0.07	0.036	0.084	0.28	0.32	11.25	11.25	0.35	0.35	90	30	6	0.2

4.4. Simulation flow-chart

Because of the introduced noise the signals and results are stochastic processes. To say something useful about the process, different realisations need to be generated. Therefore a Monte-Carlo simulation structure is created, as shown in Figure 4.2. An overview of all major steps in the simulation is given.

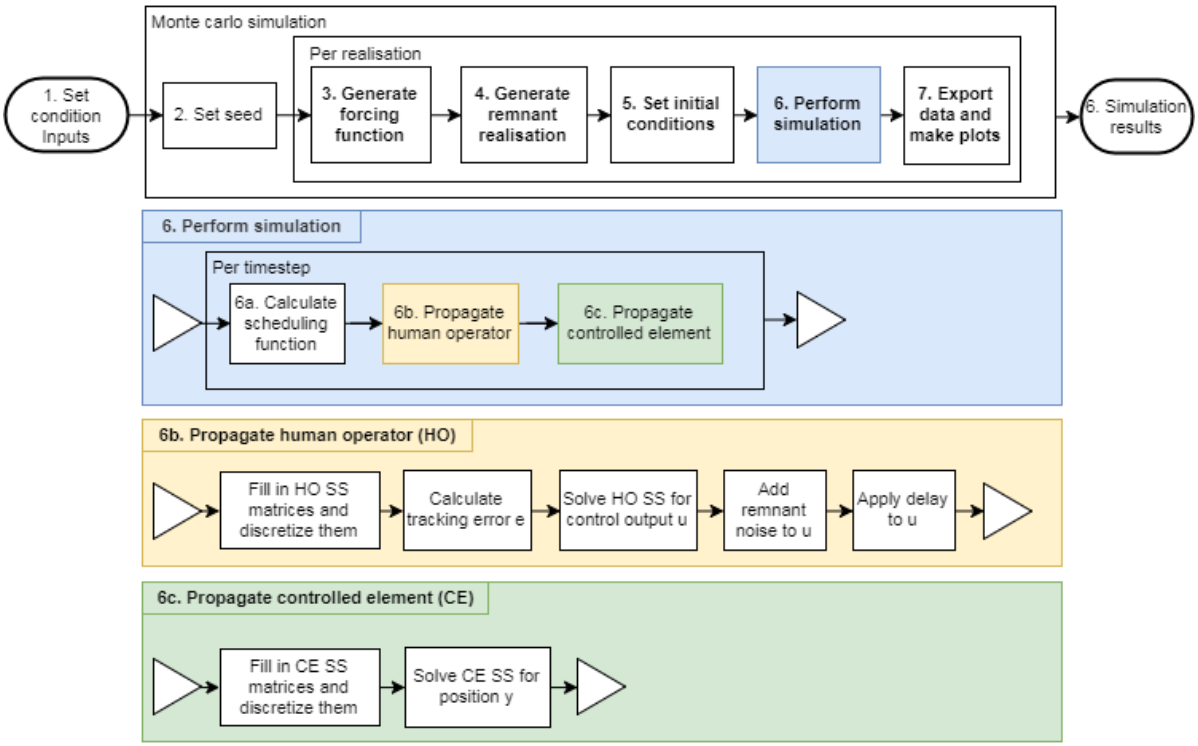


Figure 4.2: Flow chart of simulation loop.

5

Human Controller Identification Setup

Using the simulated time traces of the error and the control signal, a model for the human operator can be identified. A linear difference equation with 'AutoRegressive with eXternal input'(ARX) structure of certain order is used. The coefficients of this ARX model need to be identified. Then these coefficients need to be converted to the actual pilot model parameters to complete the identification process. The approaches used for these two steps are described in this chapter.

5.1. ARX model fitting

In the literature review in Section 2.3 the general ARX model structure is described. In the work of (van Grootheest, 2017), an investigation is done which model order gives the best identification results. However, the order is constrained by the parameter retrieval method used. Therefore a second order ARX model is assumed to allow for straightforward parameter retrieval as described later.

The ARX model structure is repeated in Eq.(5.1). Note that $e(t-n_k) = q^{-n_k}e(t)$. Substituting the $A(q)$ and $B(q)$ polynomials corresponding to a second order model ($n_a = 2, n_b = 2$), then Eq.(5.2) is the resulting model from error signal to control signal that needs to be identified.

$$u(t) = \frac{B(q)}{A(q)}q^{-n_k}e(t) + \frac{1}{A(q)}\epsilon(t) \tag{5.1}$$

$$H(q) = \frac{u(t)}{e(t)} = \frac{b_0^d + b_1^d q^{-1}}{1 + a_1^d q^{-1} + a_2^d q^{-2}}q^{-n_k} \tag{5.2}$$

The ARX model can be rewritten in linear regression format using the one-step-ahead predictor $\hat{u}(t|t-1)$ in Eq.(5.3) (Ljung, 1999). This results in the regression vector and coefficient vector as in Eq.(5.4)-(5.6). The coefficients can be estimated using the least squares approach. The delay parameter n_k cannot be included in the regression vector and needs to be specified before applying least squares estimation. This will be treated in Chapter 7.

Ordinary least squares (OLS) is performed for the time-invariant operator parts of the simulation, thus before and after the change. Recursive least squares (RLS) is used for both time-invariant and time-variant operator, i.e. during the change as well. They are described in the next sections.

$$\hat{u}(t|t-1) = [1 - A(q)]u(t) + B(q)e(t) \tag{5.3}$$

$$\hat{u}[i|i-1,\theta] = \varphi[i]\theta \tag{5.4}$$

$$\varphi[i] = \begin{pmatrix} -u[i-1] & -u[i-2] & e[i-n_k] & e[i-n_k-1] \end{pmatrix} \tag{5.5}$$

$$\theta = \begin{pmatrix} a_1^d & a_2^d & b_0^d & b_1^d \end{pmatrix}^T \tag{5.6}$$

5.1.1. Ordinary least squares estimation

OLS estimation is done when the simulation run is completed, since then all data points are known. OLS estimation is performed only for the time-invariant parts of the run (before and after the adaptation), since it acts in batch mode and cannot track changes in the parameters.

The OLS criterion to be minimised is given in Eq.(5.7). The corresponding OLS estimator is given in Eq.(5.8) using n data points. Regression matrix Φ and the vector of simulated control output values $\boldsymbol{u}$ are then defined in Eq.(5.9)-(5.10). Care should be taken to choose i_{start} such that all entries in the regression

matrix are known. Furthermore a run-in time equal to two seconds is chosen to let transients die out in the simulation. This means $i_{start} = 200$.

$$\hat{\theta}_n^{OLS} = \arg\min_{\theta} \frac{1}{n} \sum_{i=i_{start}}^{i_{start+n}} (u[i] - \varphi[i]\theta)^2 \tag{5.7}$$

$$\hat{\theta}_n^{OLS} = (\Phi^T\Phi)^{-1}\Phi^T \boldsymbol{u} \tag{5.8}$$

$$\Phi = \begin{pmatrix} \varphi[i_{start}] \\ \varphi[i_{start}+1] \\ \cdots \\ \varphi[i_{start}+n] \end{pmatrix} \tag{5.9}$$

$$\boldsymbol{u} = \begin{pmatrix} u[i_{start}] & u[i_{start}+1] & \cdots & u[i_{start}+n] \end{pmatrix}^T \tag{5.10}$$

5.1.2. Recursive least squares estimation

To track the adaptation of the human operator, the RLS algorithm with forgetting factor is used during the simulation run. Initial conditions for the coefficient vector $\hat{\theta}_0$ and for the covariance matrix P_0 are specified in Eq.(5.11). The forgetting factor was tuned in (van Grootheest, 2017) and is taken as $\lambda = 0.99609$. The RLS algorithm update steps are given in Eq.(5.12). Again, estimation can only start when all entries in the regression vector are known, however, the run-in time of $i_{start} = 200$ already takes care of that.

$$\hat{\theta}_0 = \begin{pmatrix} -1.85 & 0.85 & 0.08 & -0.08 \end{pmatrix}^T \qquad P_0 = 10,000\boldsymbol{I}_{4x4} \tag{5.11}$$

$$\begin{array}{rcll} K_i & = & P_{i-1}\varphi^T[i](\varphi[i]P_{i-1}\varphi^T[i] + \lambda)^{-1} & \\ \hat{\theta}_i & = & \hat{\theta}_{i-1} + K_i(u[i] - \varphi[i]\hat{\theta}_{i-1}) & \forall i = i_{start},..,N-1,N \\ P_i & = & \frac{1}{\lambda}(P_{i-1} - K_i\varphi[i]P_{i-1}) & \end{array} \tag{5.12}$$

5.2. Parameter retrieval

To retrieve the human operator model parameters from the identified ARX coefficients, a parameter retrieval process is followed. The ARX model as in Eq.(5.2) can be converted to a discrete time transfer function as in Eq.(5.13) using the following Z-transform property: $Z[q^{-n}f(i\Delta t)] = z^{-n}F(z)$. Note that $n_k = \tau/\Delta t$.

$$H_p(z) = \frac{b_0^d + b_1^d z^{-1}}{1 + a_1^d z^{-1} + a_2^d z^{-2}} z^{-\frac{\tau}{\Delta t}} \tag{5.13}$$

This discrete time transfer function should be converted to continuous time domain, since the human operator model is defined in that domain. In Matlab this is done easily using the *d2c* command by specifying a conversion method and the timestep the discrete model uses. In Python an equivalent function to *d2c* will need to be programmed. The conversion result is a continuous time transfer function as in Eq.(5.14).

$$H_p(s) = \frac{b_0^c s + b_1^c}{s^2 + a_1^c s + a_2^c} e^{-s\tau} \tag{5.14}$$

The final step is to compare the continuous time identified transfer function coefficients with the analytical human operator transfer function shown in Eq.(5.15). A system of equations can be set up to retrieve the human operator parameters as in Eq.(5.16). The delay is treated in Chapter 7.

$$H_P(s) = \frac{K_e\omega_{nm}^2 + K_{\dot{e}}\omega_{nm}^2 s}{\omega_{nm}^2 + 2\zeta_{nm}\omega_{nm}s + s^2} e^{-s\tau} \tag{5.15}$$

$$\begin{array}{rclrcl} K_e & = & \frac{b_1^c}{a_2^c} & K_{\dot{e}} & = & \frac{b_0^c}{a_2^c} \\ \zeta_{nm} & = & \frac{a_1^c}{2\sqrt{a_2^c}} & \omega_{nm} & = & \sqrt{a_2^c} \end{array} \tag{5.16}$$

Now it becomes clear why a second order model is advantageous for parameter retrieval: the obtained system is neither under- or overdetermined. Higher order ARX models would lead to an overdetermined system, requiring model order reduction in order to solve for the operator parameters.

5.2.1. Implementing Matlab's *d2c* in Python

The goal of the *d2c* function is to convert the discrete time transfer function coefficients estimates, $\theta_d = \begin{pmatrix} a_1^d & a_2^d & b_0^d & b_1^d \end{pmatrix}$, to continuous time transfer function coefficients $\theta_c = \begin{pmatrix} a_1^c & a_2^c & b_0^c & b_1^c \end{pmatrix}$. Four different conversion methods can be chosen for the command. It was found that the 'zero order hold' (ZOH) method is the most straightforward one to implement in Python and later in the experiment software.

The steps in Table 5.1 are identified in the conversion process by analysing the Matlab command source code files. Next to the Matlab implementation, the corresponding step in the Python implementation by the author is stated.

Table 5.1: Steps in converting the discrete time transfer functions coefficients to continuous time in Matlab and Python. [path] = *C:\Program Files\MATLAB\R2017a\toolbox\shared\controllib \engine*

Step	Matlab implementation	Python implementation
1	Input to d2c() is a transfer function, convert it first to state space format and call corresponding d2c() function for a state space system. Source: *[path]\+ltipack\@ tfdata\d2c.m*	Write the transfer function in controllable canonical form state space system as in Eq.(5.17). Fill in matrices with the coefficients.
2	Check if state space system is proper. Then call utInvDiscretize.m Source: *[path]\+ltipack\@ssdata\d2c.m*	/
3	The specified method is ZOH. Create expanded matrix containing $\boldsymbol{A_d}$ and $\boldsymbol{B_d}$ as in Eq.(5.18). Then call utScaledLogm.m with that expanded matrix as input. Source: *[path]\+ltipack\@ssdata\utInvDiscretize.m*	Create expanded matrix containing $\boldsymbol{A_d}$ and $\boldsymbol{B_d}$ as in Eq.(5.18).
4	Prescale the matrix using mscale.m. Then take the matrix logarithm using logm.m. Finally apply postscaling to the result using lrscale.m Source: *[path]\numerics\utScaledLogm.m*	Ignore pre- and postscaling. Just apply Numpy's logm() function to the expanded matrix.
5	Divide result by sample time. Select the new $\boldsymbol{A_c}$ and $\boldsymbol{B_c}$ matrices from the expanded matrix. $\boldsymbol{C_c}$ and $\boldsymbol{D_c}$ are equal to $\boldsymbol{C_d}$ and $\boldsymbol{D_d}$ respectively. Source: *[path]\+ltipack\@ssdata\utInvDiscretize.m*	Divide result by sample time. Select the new $\boldsymbol{A_c}$ and $\boldsymbol{B_c}$ matrices from the expanded matrix. $\boldsymbol{C_c}$ and $\boldsymbol{D_c}$ are equal to $\boldsymbol{C_d}$ and $\boldsymbol{D_d}$ respectively.
6	Convert the continuous time state space system to a continuous time transfer function. Source: *[path]\+ltipack\@ssdata\d2c.m* & Source: *[path]\+ltipack\@tfdata\d2c.m*	Convert continuous time state space system to a continuous time transfer function by calculating the coefficients using Eq.(5.19).

$$\boldsymbol{A_d} = \begin{pmatrix} 0 & 1 \\ -a_2^d & -a_1^d \end{pmatrix} \qquad \boldsymbol{B_d} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \boldsymbol{C_d} = \begin{pmatrix} b_1^d & b_0^d \end{pmatrix} \qquad \boldsymbol{D_d} = 0 \tag{5.17}$$

$$L = \begin{pmatrix} \boldsymbol{A_d} & \boldsymbol{B_d} \\ \boldsymbol{O}_{n_{B_d}\ x\ n_{A_d}} & \boldsymbol{I}_{n_{B_d}\ x\ n_{B_d}} \end{pmatrix} \tag{5.18}$$

$$H(s) = \boldsymbol{C_c}(s\boldsymbol{I} - \boldsymbol{A_c})^{-1}\boldsymbol{B_c} + \boldsymbol{D_c} \tag{5.19}$$

5.3. Identification flow-chart

The functional steps in the identification are shown in Figure 5.1. Note that the ARX model fitting and parameter retrieval is done for a whole range of ARX models with different delay parameter. This will be explained in detail in Chapter 7.

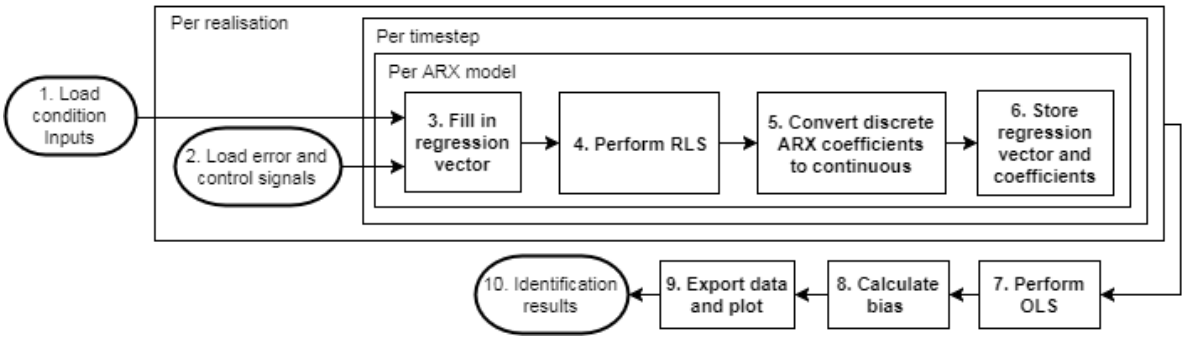


Figure 5.1: Flow chart of identification loop.

6
Code verification

Before using the developed simulation to perform research, verification of the code ensures correct conclusions can be drawn from its results. This is done by checking simplified cases up to the complicated case. The simulation and identification loop are verified separately.

6.1. Verification of simulation loop

A comparison is done between two operator transfer functions. The first is the known analytical transfer function from the operator model. The second is an identified transfer function obtained with the Fourier Coefficient (FC) method, using only the error input and control output signals of the operator. This is done in a remnant free, time-invariant behaviour simulation run (C1-N0) as shown in Figure 6.2a.

The FC method is an open-loop identification method giving an estimate of the true transfer function at the frequencies of the forcing function (van Paassen and Mulder, 1998). Time series data from the simulation is directly used, giving an independent estimate. In Figure 6.1 the transfer functions are shown to be a close match. Similar results were obtained for condition C2-N0. This implies that the state-space implementation of the operator and controlled element models and their propagation is done correctly.

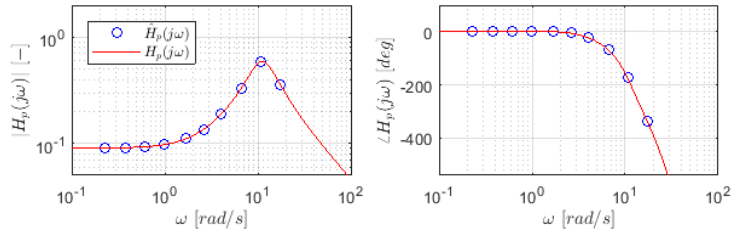


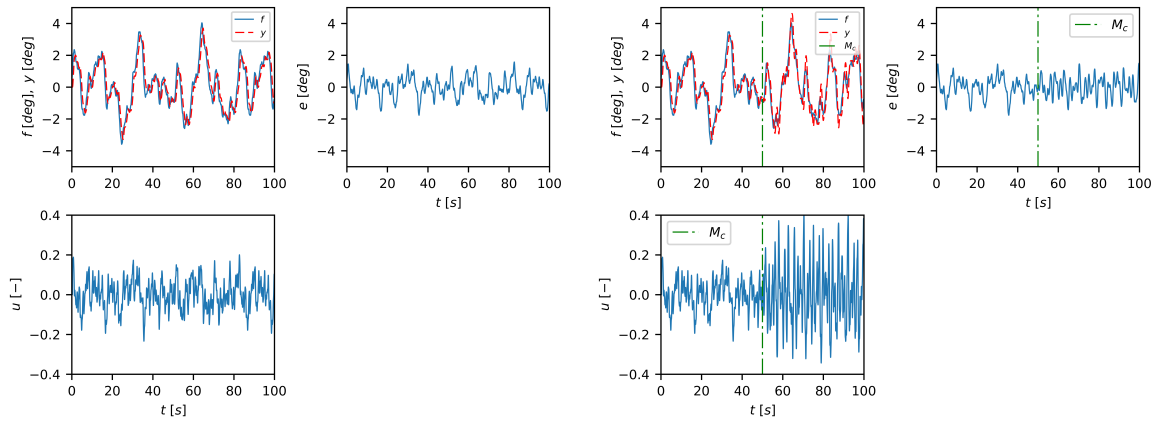
Figure 6.1: Comparison between analytical and identified transfer function using the FC method, for condition C1-N0.

Finally, a time-varying human operator simulation run is shown in Figure 6.2b according to condition C3-N0. As expected larger amplitude control action can be seen for the double integrator. Further verification for the time-varying simulation can be done by verifying the identification loop.

6.2. Verification of identification loop

To verify the identification loop both the OLS and RLS estimation of the ARX model need to be checked. This is done for the time-invariant and time-varying behaviour simulations respectively. Furthermore the conversion of the discrete ARX model coefficients to the actual operator model parameters is verified in the time-varying case. No remnant noise is added, since this would add a bias disturbing the verification. Without remnant, theoretically a zero bias in parameter estimation is expected.

Reference values for the ARX coefficients are found by discretising the known continuous time human operator transfer function for the different simulated conditions. This is done using Matlab's *c2d* command and the results are listed in Table 6.1. Note that the results are dependent on the method and timestep assumed in the discretisation. The reference values will be used in plots to verify whether the estimation bias is zero.



(a) C1-N0: Time-invariant, single integrator.

(b) C3-N0: Time-variant, single to double integrator.

Figure 6.2: Time traces of remnant-free simulation.

Table 6.1: Reference values for discrete ARX model coefficients valid for conditions C1-N0, C2-N0 and C3-N0, assuming ZOH discretisation with timestep $\Delta t = 0.01\,s$

Coefficient	Before HO adaptation (C1, C3)	After HO adaptation (C2, C3)
b_0^d	0.04428	0.1024
b_1^d	-0.04318	-0.1016
a_1^d	-1.912	-1.912
a_2^d	0.9243	0.9243

In Figure 6.3a the coefficient time traces estimated by OLS and RLS are shown for condition C1-N0. The OLS estimation perfectly overlaps with the reference values. The estimation relative bias can be calculated from Equation 6.1. They are listed in Table 6.2. Their small magnitudes show that the OLS estimation is correct, with only insignificant numerical precision errors remaining. Furthermore the RLS estimation converges to the OLS estimated coefficients at about 10s, after a 2s run-in time. This time to convergence is dependent on the forgetting factor selected, the fact that it converges is a verification that the RLS method is implemented correctly.

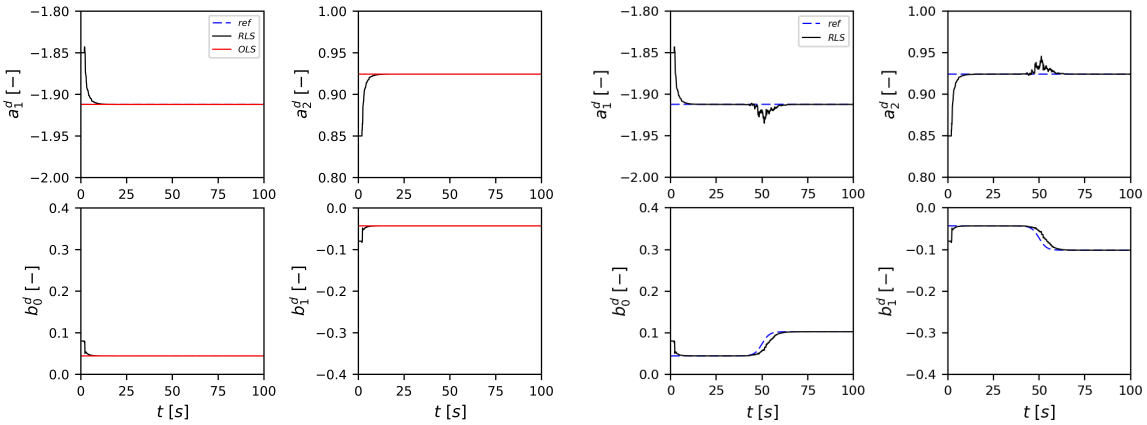
$$B_r = \frac{\hat{\theta} - \theta}{\theta} \tag{6.1}$$

Table 6.2: Relative bias in ARX coefficients with OLS estimation for condition C1-N0.

Coefficient	B_r	Coefficient	B_r
a_1^d	5.91e-5	b_0^d	-7.69e-5
a_2^d	-3.14e-5	b_1^d	5.51e-5

With the time-invariant case verified, the more complicated time-varying case can be checked. Only the equalisation adaptation scenario C3-N0 is taken into account since then a straightforward comparison of the results can be made with (van Grootheest, 2017). In Figure 6.3b the RLS estimation results for the ARX coefficients are shown. After converging, the RLS follows the change in the reference coefficients due to operator adaptation as expected.

The final step to be verified is the conversion of the identified ARX coefficients to the operator model parameters. This conversion is done at every timestep and can be compared with the reference parameter values set in the simulation inputs. The reference values are again scaled by the sigmoid function used to introduced the adaptation. Figure 6.4 shows the evolution in time, indicating that the conversion is done correctly.



(a) C1-N0: Time-invariant, single integrator.

(b) C3-N0: Time-variant, single to double integrator.

Figure 6.3: Time traces of estimated ARX model coefficients in remnant-free simulation, compared with the actual reference values.

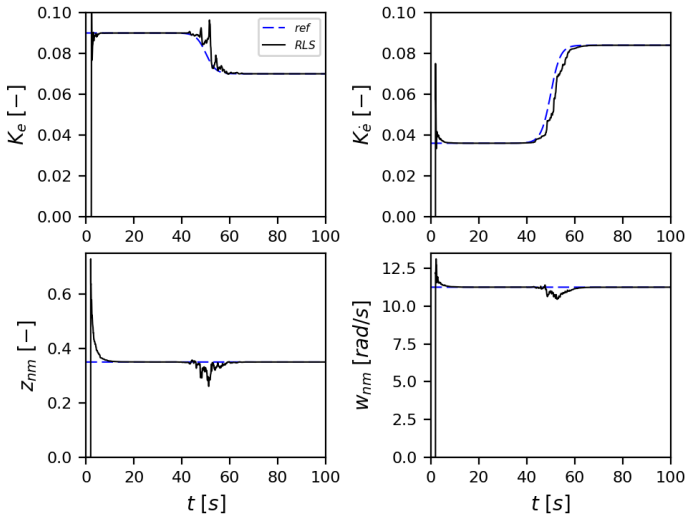


Figure 6.4: C3-N0: Time traces of estimated operator model parameters, with the actual reference values.

7

Time delay estimation with parallel recursive ARX

From the literature study a promising strategy to perform time-varying delay estimation is based on 'adaptive estimation through parallel processing' in (Anderson and Moore, 1979). Applied to recursive ARX, this means a range of ARX models is estimated in parallel, each one assuming a different delay parameter n_k. At each timestep, a quality-of-fit metric is calculated for each ARX model. The delay parameter corresponding to the best fitting ARX model is then selected as delay estimate. Thus the estimate varies over time according to how the actual time delay varies.

In Section 7.1 the implementation of parallel recursive ARX in identification and prediction is described. Two questions arise regarding the performance of this method. In Section 7.2, the sensitivity of the chosen quality-of-fit metrics to time delay error is investigated. Then the tracking ability of the method is judged based on time-varying delay simulations in Section 7.3. For both questions the effect of remnant noise is also treated.

7.1. Implementation in identification and prediction

Parallel recursive ARX estimation is easily implemented in the identification loop. Instead of updating a single ARX model at each timestep, a for-loop iterates over a range of n_k values, stored in an array n_{k_a} with length k. This leads to regression vectors with different delay shifts. The RLS update is performed for each regression vector and the resulting coefficient estimates and covariances are saved for the range of ARX models.

For prediction, a separate loop was made to easily experiment with different quality-of-fit metrics. The identified ARX models are used to predict the human operator output. The one-step-ahead prediction takes as input the error signal obtained in the simulation loop and the RLS coefficient estimates obtained in the identification loop. By iterating over the timesteps, Eq.(7.1) results in the predicted control signal $u_{pred_k}[i]$ for each ARX model k.

$$u_{pred_k}[i] = \varphi_k[i, n_{k_a}[k]] \cdot \hat{\theta}_k[i] \qquad \forall\ i = i_{start}, ..., N-1, N\ \&\ k = 0, ..., K-1, K \tag{7.1}$$

Next, the quality-of-fit between each u_{pred_k} and the simulated control signal u_{sim} is calculated. To enable tracking of changes in time delay, the quality-of-fit is recalculated every timestep over a window with a length of i_w samples. This allows to select the best fitting ARX model at each moment. The Sliding Variance Accounted For (sVAF) in Eq.(7.2) measures how much of the variance in u_{sim} is explained by u_{pred_k}. A higher sVAF value is a better fit.

$$sVAF_{k,i} = \max\left\{0, \left(1 - \frac{\sum_{j=i-i_w}^{i} \left(u_{sim}[j] - u_{pred_k}[j]\right)^2}{\sum_{j=i-i_w}^{i} u_{sim}^2[j]}\right) \cdot 100\right\}$$

$$\forall\ i = i_{start} + i_w, ..., N-1, N\ \&\ k = 0, ..., K-1, K \tag{7.2}$$

The best fitting ARX model is selected at each timestep according to Eq.(7.3). The corresponding delay parameter $n_{k_{est_i}}$ is then converted to a time delay by Eq.(7.4) and stored as the time delay estimate at that timestep. When all predictions are done, plots are generated comparing the actual and estimated time delay from both metrics. A flow chart of these steps in the prediction loop is given in Figure 7.1.

$$n_{k_{est_i}} = n_{k_a}[\arg\max_k (sVAF_{k,i} \; \forall \; k = 0,...,K-1,K)] \qquad \forall \; i = i_{start},...,N-1,N \tag{7.3}$$

$$\tau_{est_i} = \Delta t \cdot n_{k_{est_i}} \qquad \forall \; i = i_{start},...,N-1,N \tag{7.4}$$

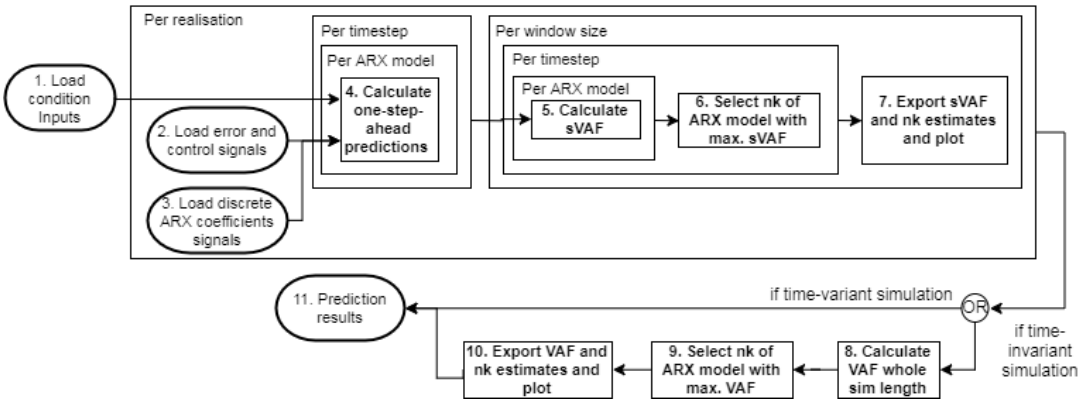


Figure 7.1: Flowchart of prediction loop.

7.2. Sensitivity of quality-of-fit to time delay error

The goal is to find out how sensitive the value of the VAF metric is to errors in time delay estimation. A larger error sensitivity is preferred since then the ARX model with the correct time delay can be found easier. The sensitivity is studied in the noise free and noise present case.

7.2.1. Without remnant noise

The noise free case is studied using the data from a single time-invariant operator simulation run with condition C1-N0. The metric is calculated with u_{sim} and u_{pred_k} for each ARX model k over the whole simulation length at once, after the run-in time of i_{start} samples. This is equivalent to setting $i_w = N - i_{start}$ in Eq.(7.2). In Figure 7.2 the resulting VAF distribution over the ARX models with different delay parameter n_k is shown.

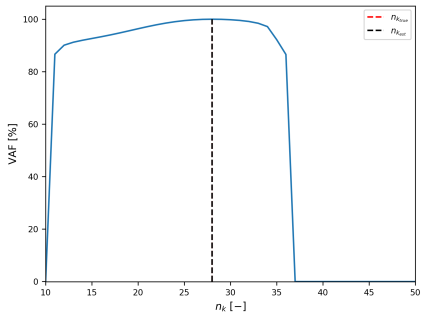
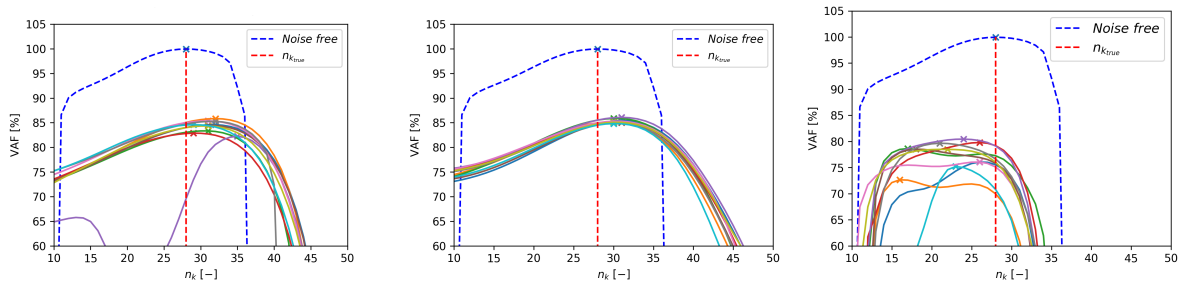


Figure 7.2: Quality-of-fit VAF metric for condition C1-N0.

The extremum is reached for $n_{k_{true}}$. However the curve is very flat around the true delay value, meaning that it is not very sensitive to time delay estimation error. The differences in VAF are due to a bias introduced in the ARX coefficients, when the time delay assumed is wrong. Note that the huge jumps in VAF value to zero indicate that predictions became unstable because the time delay assumed in the ARX model is too far from the true one.

7.2.2. With remnant noise

To find out whether the metric can identify the correct time delay also in the presence of noise, the previous simulation is done with remnant noise active. The remnant intensity was set at $P_n = 15\%$ using each of the filters discussed in Section 4.1. Several simulation runs are then done with different realisations. The effect on the VAF distribution is shown for several realisations in Figure 7.3. The noise-free case is plotted as reference. For all filters, the VAF values achieved are lower than in the noise-free case. This happens because the noise introduces an additional bias in the ARX coefficients, adding to the bias present when having a wrong time delay. For the first and second order filters, there is a broader stability range, while the third order filter gives a more narrow stability range. Most importantly, the point of maximum VAF, indicated by a cross, does not correspond anymore to $n_{k_{true}}$ in all cases.

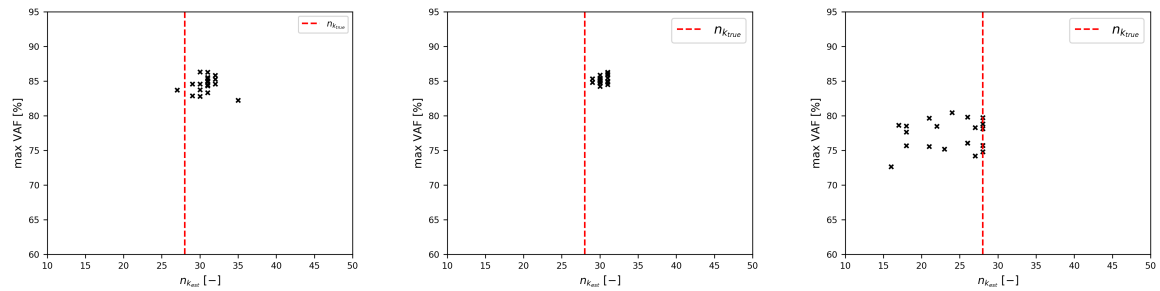


(a) C1-N1-1: first order filter. (b) C1-N1-2: second order filter. (c) C1-N1-3: third order filter.

Figure 7.3: Quality-of-fit VAF metric distribution versus delay parameter n_k for different remnant filters.

In Figure 7.4 the distribution of maximum VAF for twenty realisations is shown in a scatter plot for each filter. Each cross represents the extremum of the VAF metric in a certain realisation, plotted against the estimated delay parameter $n_{k_{est}}$. Again it is shown that the metric seems to have difficulties to achieve its extremum for $n_{k_{true}}$ in most of the realisations.

The first and second order filter overestimate the time delay, the third order filter underestimates the delay. This matches results obtained with same order filters in (van Grootheest, 2017). Additionally, the spread in estimates is a lot smaller for the first and second order filters compared to the third one. Only the third order filter has realisations in which the true delay parameter was estimated.



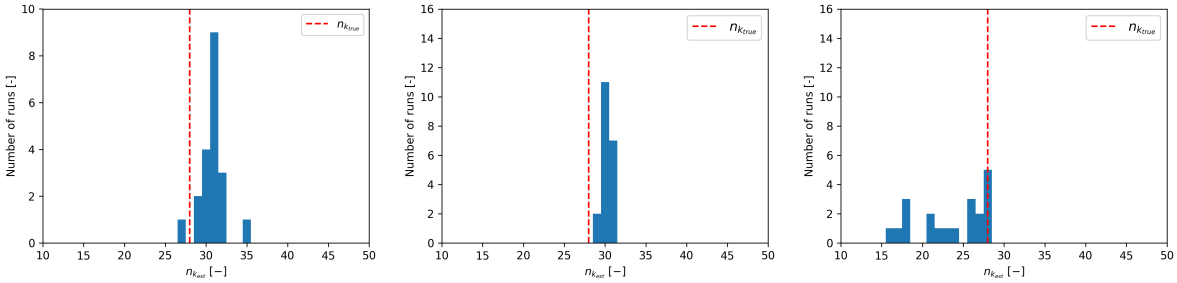
(a) C1-N1-1: first order filter. (b) C1-N1-2: second order filter. (c) C1-N1-3: third order filter.

Figure 7.4: Scatter plot of metric extrema and estimated delay parameters for twenty realisations, using the different remnant filters.

To check whether any convergence is happening towards a certain time delay estimate when combining all realisations, a histogram is included in Figure 7.5 showing the distribution of $n_{k_{est}}$.

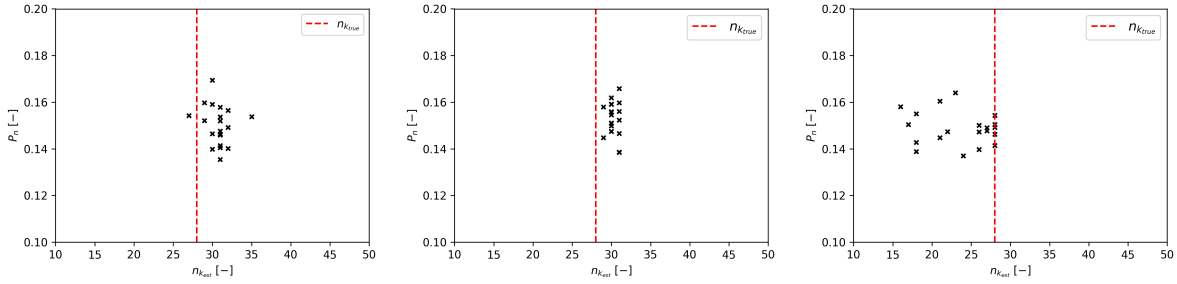
The first order filter converges to $n_{k_{est}} = 31$, while the second order filter goes more to $n_{k_{est}} = 30$. Note again the larger spread in estimates for the third order filter and that it is the only one for which $n_{k_{true}}$ was estimated.

As verification, a scatter plot showing the actual remnant intensity P_n obtained in each run versus $n_{k_{est}}$ estimated in that run is shown in Figure 7.6. The plots indicate that the $n_{k_{est}}$ result is mainly dependent on the remnant realisation, and not on small changes in remnant intensity.



(a) C1-N1-1: first order filter. (b) C1-N1-2: second order filter. (c) C1-N1-3: third order filter.

Figure 7.5: Histograms showing the distribution of estimated delay parameters for twenty realisations, using the different remnant filters.



(a) C1-N1-1: first order filter. (b) C1-N1-2: second order filter. (c) C1-N1-3: third order filter.

Figure 7.6: Remnant intensity versus estimated delay parameter for twenty realisations using the different remnant filters.

7.2.3. Conclusion

To conclude the VAF metric is rather insensitive to errors in time delay estimation due to the small differences in magnitude between different n_k values. In the noise free case the extremum is reached for $n_{k_{true}}$. When noise is included, the extrema reached do not always correspond to $n_{k_{true}}$, leading to biased time delay estimates. Two causes are hypothesised for this. Firstly, the sensitivity to time delay estimation error could be masked by the introduced noise. Secondly, with noise a bias appears in the ARX model coefficients. Because of that, different combinations of coefficients and time delay parameter could result in the best quality-of-fit. That combination does not necessarily correspond anymore to the actual time delay.

7.3. Tracking time-varying time delay

To evaluate the tracking performance of parallel recursive ARX, the sliding window size is varied between three lengths. The window size is inherently a trade-off between sensitivity to noise and how fast changes in time delay will be tracked. Two different time-varying operator scenarios are considered. In the first one, adaptation occurs only in time delay. In the second one, adaptation occurs both in time delay and equalisation as well, since the controlled element is varied. Both the noise free and noise present case are considered.

7.3.1. Without remnant noise

First the noise free case is considered as verification of time delay estimation with parallel recursive ARX. In condition C4-N0 only the time delay is varying. This enables to see whether the delay estimates using the sliding window VAF metric converge to the actual time delay.

As first view, a colour map is made of the sVAF values achieved over time for all ARX models. This view is shown in Figure 7.7. Three different window sizes are considered. As overlay, the line following the maximum VAF over time is plotted, indicating $n_{k_{est}}$. Furthermore $n_{k_{true}}$ is included for reference. A second view directly shows the time delay estimation over time for all windows as in Figure 7.8.

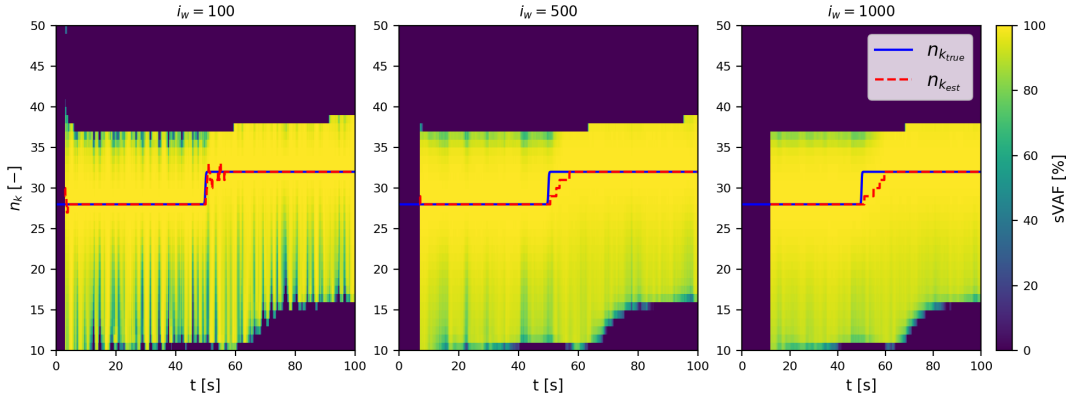


Figure 7.7: Colour map of sVAF metric over time for condition C4-N0.

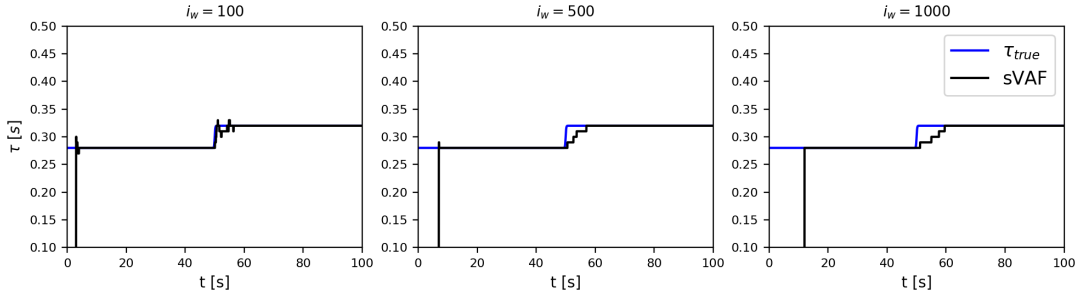


Figure 7.8: Delay estimation over time for condition C4-N0.

The window size influences the starting time of the delay estimation, as enough samples need to be present before the sliding window can be calculated. The estimated delay then converges quickly to the actual delay for all window sizes.

In Figure 7.7, note the large band of high VAFs, indicating very small differences between the best ARX model and its neighbours. Furthermore one can see more oscillations in the VAF values for the smaller window size, because less samples are used to calculate the quality-of-fit and thus it is more sensitive to local conditions. This effect is smoothed out as the window size increases. After the operator adaptation, the range of stable ARX models reduces at the lower end of delay estimates.

Figure 7.8 shows that the time delay is well tracked. Naturally the smallest window size tracks the change in delay the best. For larger window sizes it takes longer before the estimate converges to the actual delay.

Finally a time-varying behaviour simulation where both operator equalisation and time delay vary, is considered. In Figure 7.10 it can be seen that the tracking is done well but a bit slower as in the case where delay is the only time-varying element. This is probably due to the fact that ARX coefficients now also need to converge to their new value corresponding to the double integrator in the time-variant behaviour simulation, before a reliable time delay estimate can be made.

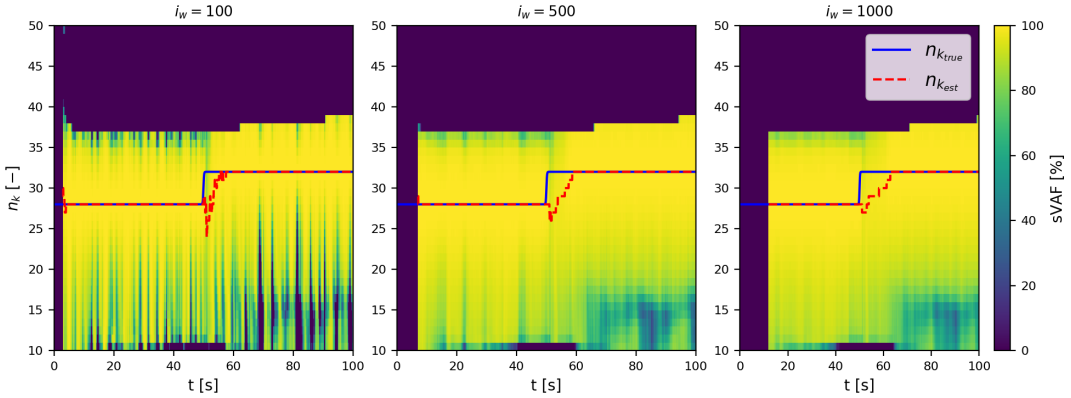


Figure 7.9: Colour map of sVAF metric over time for condition C5-N0.

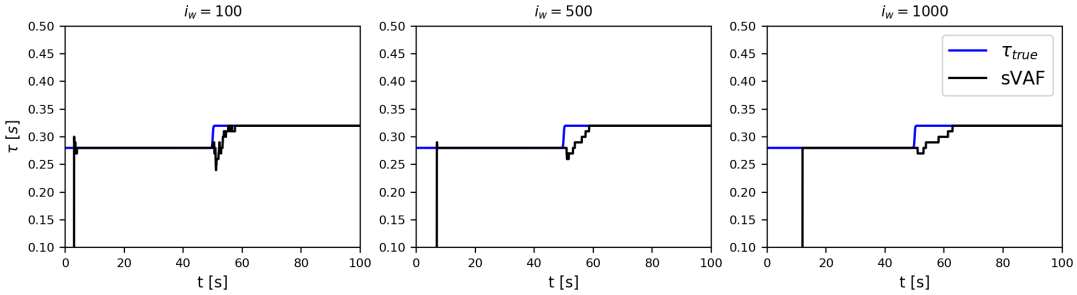


Figure 7.10: Delay estimation over time for condition C5-N0.

7.3.2. With remnant noise

For the case with remnant noise active, a distinction is made again between a time-invariant and time-variant behaviour simulation. This is done to check the tracking performance in presence of noise but without adaptation, which could be deteriorated when the sliding window VAF metric is too sensitive to the noise. Next time-varying behaviour simulations are used to draw the final conclusions about the usability of the parallel recursive ARX method.

Time-invariant behaviour simulation

Condition C1-N1 was used again in combination with each studied remnant filters. One realisation was used to plot the evolution of the VAF metric for each window size, as shown in Figures 7.11-7.13.

As was shown before, there were already difficulties to identify the true delay parameter in the time-invariant scenario where the whole simulation run was used as window for the VAF quality-of-fit metric. This is naturally also the case for the sliding window VAF metric. In these runs the time delay is actually constant, but $n_{k_{est}}$ is sensitive to noise. The results are the best for the second order filter. Furthermore window size $i_w = 100$ is useless, since it is too sensitive to noise.

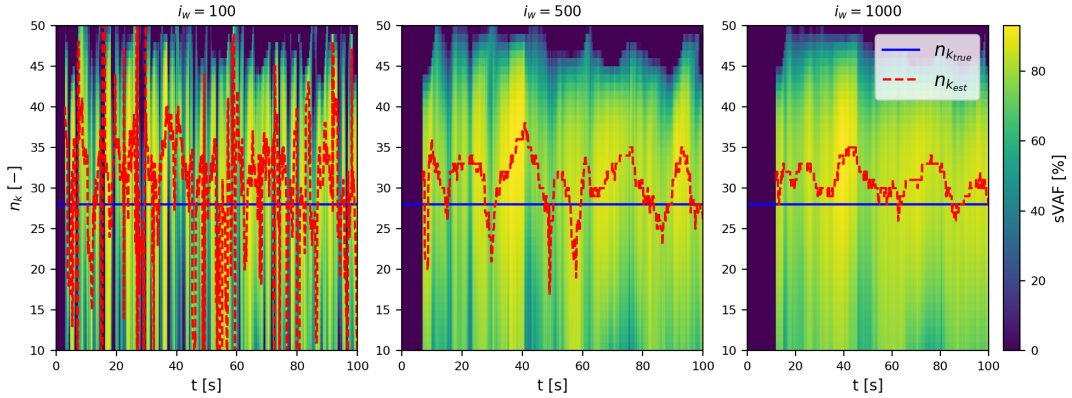


Figure 7.11: Colour map of sVAF metric over time for condition C1-N1-1 with first order filter for one realisation.

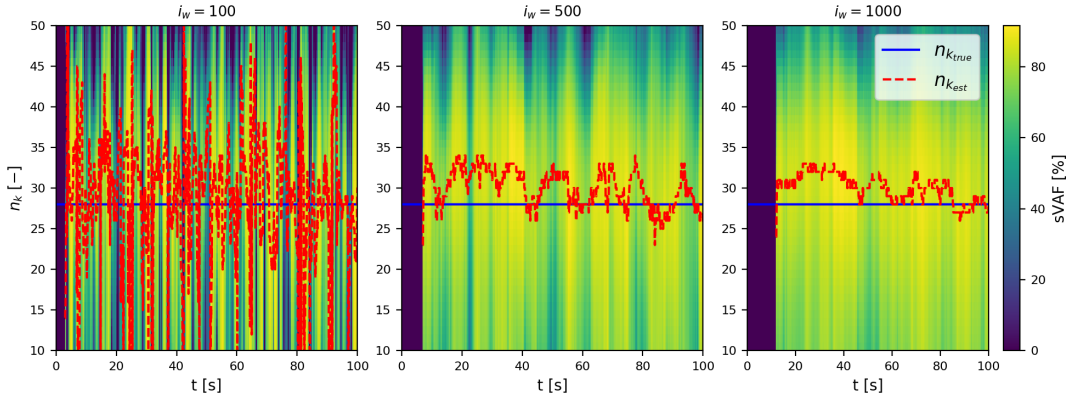


Figure 7.12: Colour map of sVAF metric over time for condition C1-N1-2 with second order filter for one realisation.

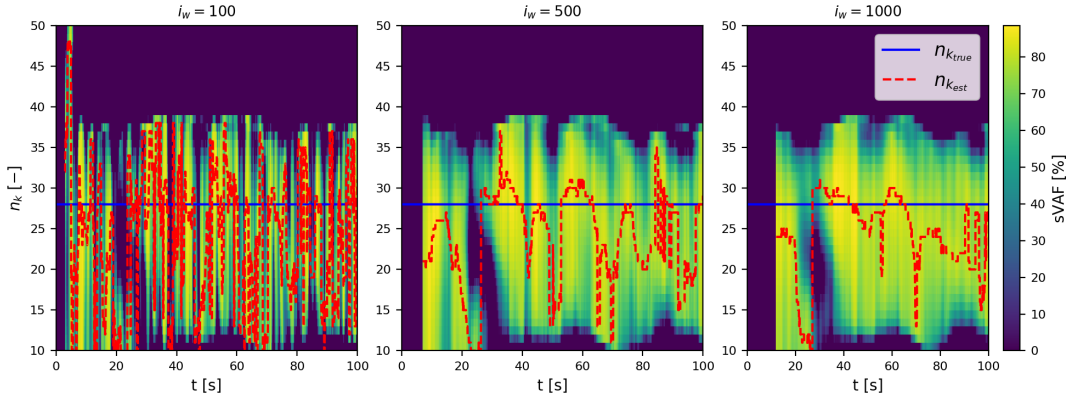


Figure 7.13: Colour map of sVAF metric over time for condition C1-N1-3 with third order filter for one realisation.

Time-variant behaviour simulation

This simulation is done only with the second order filter because it gave the best results in the time-invariant case. The delay estimation in conditions C4-N1-2 and C5-N1-2 for one realisation is shown in Figures 7.14 and 7.15. Only window sizes $i_w = 500$ and $i_w = 1000$ are shown. In both conditions a change in the average level of time delay can be seen, however the result is oscillating heavily and biased. Whether or not these results will be achieved as well in real-life experiments depends on how the human remnant noise is in reality.

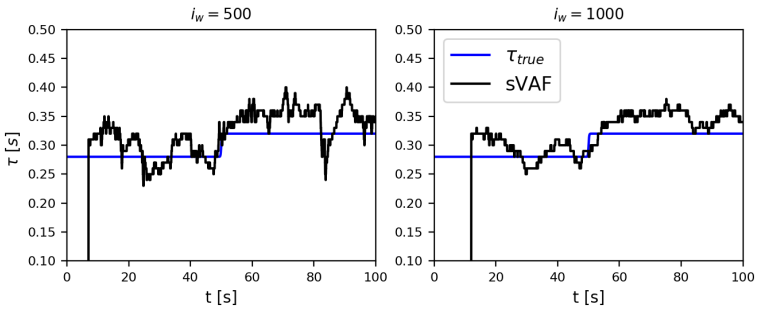


Figure 7.14: Delay estimation over time in condition C4-N1-2 for one realisation.

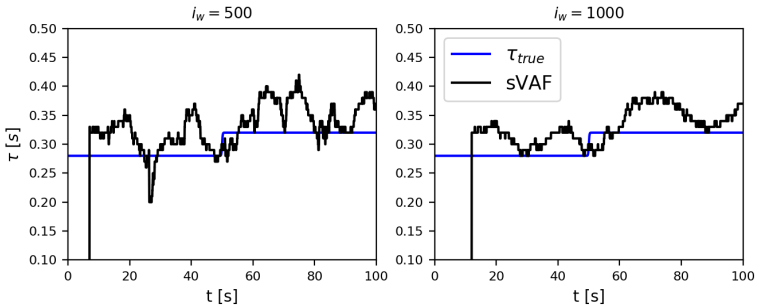


Figure 7.15: Delay estimation over time in condition C5-N1-2 for one realisation.

7.4. Online implementation

To implement the parallel recursive ARX procedure the steps in Figure 7.16 need to be implemented in the experiment software platform. Note that the sVAF is only calculated for one window size.

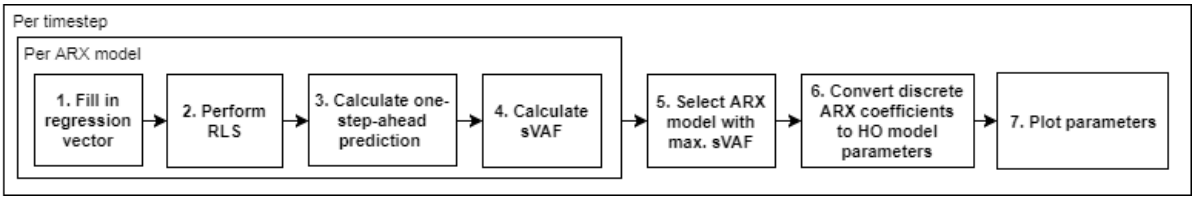


Figure 7.16: Steps for online identification and prediction.

Since the estimated parameters should be updated frequently, it is necessary to investigate how fast these steps can be executed. In the Python implementation the elapsed time for some critical steps was determined. In Table 7.1 the execution times are listed for the case where 40 ARX models are estimated in parallel.

Thus for one timestep, the procedure takes approximately 0.126s. This is quite long, and would mean that an update frequency of 8Hz is only possible for the human operator parameters. However the method should be implemented in the experiment software platform. This is written in C++, thus the method could probably execute faster since C++ is a compiled language. Conclusions about the parameter estimate update frequency can therefore only be drawn once the method is implemented in C++.

System on Intel i5-2450M CPU @ 2.50 GHz with 8GB RAM. Background processes may have influenced these estimates.

Table 7.1: Approximate execution time of selected steps in identification-prediction algorithm in Python. For steps 2, 3 and 4 there are 40 different ARX models estimated.

Step	Execution time [s]
2. Perform RLS	0.040
3. Calculate one-step-ahead prediction	0.040
4. Calculate sVAF	0.040
6. Convert discrete ARX coefficients to HO model parameters	0.006
Sum	0.126

7.5. Results summary

As expected, when looking at results from the investigation in Section 7.2 in the case with noise present, the tracking of the time delay is also very sensitive to the remnant noise in Section 7.3. The second order filter gives good results, but the question is now how different real remnant noise is from this assumption. This will either improve estimation performance or make it worse. It can only be investigated when using human operator data obtained from an experiment.

The sliding window sizes $i_w = 500$ and $i_w = 1000$ give acceptable results, however with a sampling rate of 100 Hz this means a change in time delay is only detected after 5 or 10 seconds respectively. This could be too slow in certain situations where time delay is continuously varying. An increased sampling frequency could reduce this estimation delay for time delay, since more samples are available for the sliding window.

A quick check for execution time of the parallel recursive ARX method was performed for the Python implementation. It shows that currently the update frequency of the parameters is rather low, and it is hoped that when implementing the method in the experiment software platform, which is a compiled language, the update frequency could increase to a feasible level.

8

Conclusion

The aim of this research is to identify time-varying human operator behaviour online during experiments. The goal of the preliminary study was to find a suitable method for this and check its feasibility in a simulation. The literature survey gave an overview of human operator models and system identification methods used in cybernetics. The quasi-linear human operator model by McRuer is the most used one. The most suitable time-varying identification methods with a potential online implementation are the Extended Kalman Filter and ARX models with recursive least squares.

The latter was investigated in (van Grootheest, 2017) for offline applications. Two main issues were discovered. Firstly, the ARX model is unavoidably also fitted to the introduced noise and this causes a bias in the estimated ARX coefficients. Secondly, the time-varying human operator delay cannot be included in the least squares estimation, thus requiring a different approach.

Looking at different time delay estimation methods from the field of system identification, the following potential solution was identified. A range of ARX models is estimated in parallel, each one assuming a different delay parameter. At each timestep, a quality-of-fit metric is calculated for each ARX model. The delay parameter corresponding to the best fitting ARX model is then selected as delay estimate. The delay can however only be an integer multiple of the sampling time. The method is called parallel recursive ARX.

To check feasibility a time-varying human operator simulation was made in Python according to the work of (Zaal, 2016). The operator is adapting to a sigmoid-scheduled change of single integrator to double integrator controlled element. The whole range of ARX models was fitted using recursive least squares with forgetting factor and then the resulting coefficients were converted to the human operator model parameters. For that conversion a special focus was needed on implementing Matlab's d2c command in Python.

Using the identified ARX models, the predicted control signal was generated. The VAF metric with sliding window determined the quality-of-fit between the simulated and predicted control signals at each timestep. Then the best fitting ARX model was selected and its corresponding assumed time delay was the estimate.

The influence of the sliding window size and the assumed remnant filter on the time delay tracking was investigated. A window size of 500 samples and the second order remnant filter gave best results. However the time delay estimate was still oscillating with a high spread. Adaptation could be seen, however it is not clear whether the method will still work when in real life adaptations are smaller. Finally the execution time was analysed to check whether online implementation is feasible. However no conclusions could be drawn yet because for the experiment the algorithm will be implemented in C++, which might be a lot faster than the Python implementation.

9
Further Research

The following further research steps proceed towards online identification of the human operator. They are split in two phases. Phase I consists of further developing parallel recursive ARX and implementing it in the experiment software platform. Phase II considers the experimental evaluation itself.

Phase I

- The time delay estimation method will be validated by using experimental data as input for the identification and prediction loop. Data from a previous experiment at the faculty following the work of (Zaal, 2016) is available.
- The effect of sampling frequency on the delay estimation needs to be investigated using simulation.
- Then the parallel recursive ARX algorithm can be implemented in the experiment software platform, called DUECA. At first the code from an existing experiment, according to (Zaal, 2016), will be analysed and the time delay estimation algorithm inserted.

Phase II

In Phase II the time delay tracking performance of the parallel recursive ARX algorithm could be experimentally evaluated. The experiment could be set up to specifically focus on forcing the human operator to adapt the time delay. Possible experiments include:

- Follow the existing experiment done in (Zaal, 2016) where the controlled element changes from approximately single to approximately double integrator dynamics. Alternatively, a pure single and double integrator could be used. According to (McRuer and Jex, 1967), there should be an adaptation in operator time delay then.
- In (Boer and Kenyon, 1998) a time-varying gap postview tracking task is used to specifically force adaptation of the operator time delay.
- The critical tracking task of (Jex et al., 1966) could be used, which consists of a controlled element with growing instability that needs to be stabilised by the human controller. There is an associated adaptation in operator time delay.
- A more exotic experiment is the one described in (Zaal et al., 2013). The display type could be changed from an explicit representation of self-motion to an optical flow based representation. This increases the human operator time delay significantly.

Suppose the bias present in the time delay estimates with the current method is unacceptable, the following alternatives obtained from the literature survey could be considered:

- Extend the parallel recursive ARX method with a prefilter (Björklund, 2003). The prefilter is a noise model obtained from a fitted ARMAX model to the human operator signals. It is used to filter the noise out of the input and output signals, before the ARX model is fitted. The ARMAX model requires a nonlinear fitting procedure. The associated computational load could be a problem online, however, the prefilter could be identified at a lower rate than the ARX model or a-priori to mitigate this.

- Another option is to use the Box-Jenkins model structure, which allows independent identification of the noise and system dynamics. The online feasibility of this method requires an investigation however, since it requires a nonlinear fitting procedure and its potential for recursive fitting is not clear.
- Finally, a possible last resort is to switch to the work of (Popovici et al., 2017) where a dual extended Kalman filter is successfully used to estimate both human operator equalisation and time delay. The method is not yet tried out in an online setting during an experiment.

III
Appendices

to be graded for AE5310 Final Thesis

A

Appendices to preliminary graduation report

A.1. Monte Carlo simulation for parallel recursive ARX Identification of delay and other HO Parameters

In the preliminary graduation report only a single simulation run was shown of the parallel recursive ARX identification (parARX) method used to identify a time-varying human operator (HO) time delay. Furthermore, the focus was only on the HO delay parameter, while also the other HO parameters (K_e, $K_{\dot{e}}$, ω_{nm} and ζ_{nm}) were estimated by parARX. In this appendix, HO parameter estimation results for a Monte Carlo simulation with 100 realisations are discussed.

A.1.1. Parallel recursive ARX for HO delay identification

Since remnant noise is a stochastic process, the identification results of a large amount of realisations of the simulation should be combined. This way a more general conclusion can be drawn about the usability of parARX for HO delay estimation. The delay traces were averaged over 100 simulation runs, and 2σ bounds were calculated. This is done for conditions C1-N1-2 and C5-N1-2 with two different sVAF window lengths i_w, as shown in Figure A.1.

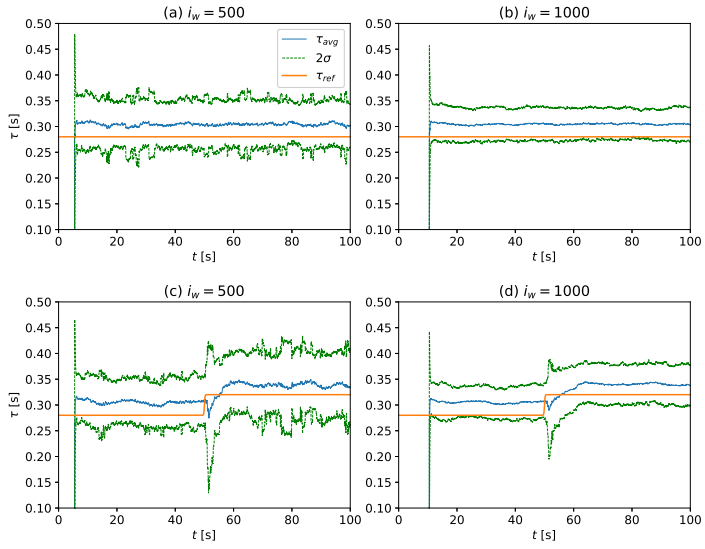


Figure A.1: Result of delay identification with parARX, averaged over 100 realisations of conditions C1-N1-2 (a-b) and C5-N1-2 (c-d) for two window lengths i_w. The 2σ bounds and the actual delay τ_{ref} are shown as well for reference.

Delay τ is overestimated on average, in both the time-invariant (Figure A.1a-b) and time-variant case (Figure A.1c-d). The time-variant case also shows the significant estimation lag before the delay is converged to its new level. Although the bounds are smaller for a larger window length, note that the 2σ bounds cover already a large

region in which possible time delay adaptations might happen. This reduces the ability of parARX to detect small delay adaptations.

On top of that, the single run results in Section 7.3 already indicated the large oscillations in the estimation compared to the magnitude of adaptation. This also reduces the practical usability of parARX for online identification of delay.

A.1.2. Identification of other HO parameters

Next to parARX, which relies on many ARX models with different delay assumption, one can use a single recursively identified ARX model with a constant delay assumption (referred to as singleARX, with $n_k = 28$ and $\tau = 0.28$ s). A comparison is made of identification performance between parARX and singleARX for estimation of HO parameters other than delay.

Results for condition C3-N1-2 are shown in Figure A.2. Condition C3-N1-2 contains HO adaptation from single to double integrator in all HO parameters, except for the HO delay which is taken constant. The parARX method shows more bias for K_e and similar bias for the other parameters. This is likely due to the overestimated delay with parARX, while singleARX assumes the correct delay.

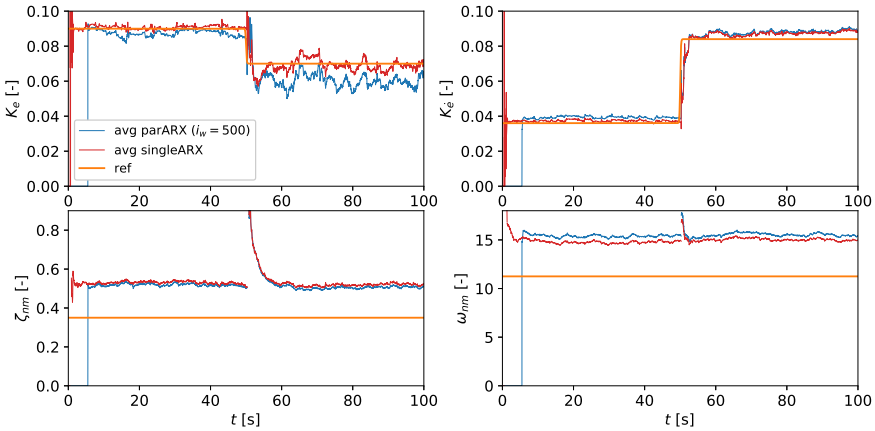


Figure A.2: Result of HO parameter identification with parARX ($i_w = 500$ only) and singleARX, averaged over 100 realisations of condition C3-N1-2. The actual HO parameters are shown as well for reference.

Figure A.3 shows identification results for C5-N1-2. This condition also contains an adaptation in HO delay. The biases achieved with parARX remain very similar to those in condition C3-N1-2. However, the biases for singleARX do change. After adaptation, for K_e & ζ_{nm} the bias increases and for $K_{\dot{e}}$ & ω_{nm} the bias decreases. This is due to the wrong delay being assumed in singleARX after adaptation. Depending on whether there is an overestimation or underestimation (as in this case), the bias magnitudes will either increase or decrease for the different HO parameters.

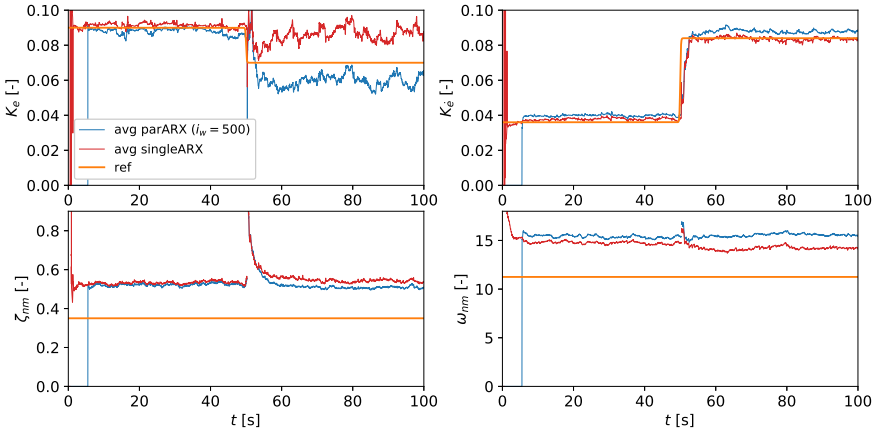


Figure A.3: Result of HO parameter identification with parARX ($i_w = 500$ only) and singleARX, averaged over 100 realisations of condition C5-N1-2. The actual HO parameters are shown as well for reference.

A.2. Parallel recursive ARX identification at 1000 Hz

To extend the preliminary graduation report, the effect of sampling frequency on human operator identification with recursive ARX is investigated. Up to now identification was done at 100 Hz. Perhaps a higher sampling frequency improves the tracking of the human operator model parameters. In this appendix, the modifications required in the simulation and identification to run at 1000 Hz are described. Furthermore the identification results at this higher sampling frequency are discussed and a conclusion is made whether they improved.

A.2.1. Modifications to simulation and identification

Simulation runs now need to be done with a timestep of $\Delta t = 0.001s$. This is as simple as changing the timestep value in the code, except for the remnant noise generation. The noise needs to be scaled according to the time step, to have equal remnant intensity in the 100Hz and 1000Hz case. The generated white noise array n_w is scaled according to Eq. A.1, before passing it through the remnant filter. This approach makes the remnant gains independent of sampling frequency.

$$n_{w_{scaled}} = \frac{n_w}{\Delta t} \tag{A.1}$$

The following modifications in the identification process are required. First, the analytical human operator transfer function needs to be discretised again using Matlab's *c2d* command with the new timestep as input. This results in updated discrete ARX coefficients shown in Table A.1, for use as reference in plots.

Table A.1: Reference values for discrete ARX model coefficients valid for condition C3 and C5, assuming ZOH discretisation with timestep $\Delta t = 0.01s$ and $\Delta t = 0.001s$

Coefficient	Before HO adaptation		After HO adaptation	
	$\Delta t = 0.01s$	$\Delta t = 0.001s$	$\Delta t = 0.01s$	$\Delta t = 0.001s$
b_0^d	0.04428	0.004544	0.1024	0.01059
b_1^d	-0.04318	-0.004533	-0.1016	-0.01058
a_1^d	-1.912	-1.992	-1.912	-1.992
a_2^d	0.9243	0.9922	0.9243	0.9922

The recursive least squares (RLS) used in the identification is dependent on a forgetting factor, tuned in (van Grootheest, 2017) for the 100 Hz case. To keep the same weighing when the sampling frequency is increased, the factor needs to be adjusted. It is selected such that the memory horizon size in time is kept constant, according to Eqs. A.2-A.3. This means an increased number of samples is used while the same time-span of data is covered in the RLS update. Not adjusting the forgetting factor resulted in unstable identification results at 1000 Hz for simulations with remnant noise.

Furthermore the initial condition for the RLS is chosen in the neighbourhood of the new reference values to enable fair comparison. The time required for convergence depends on how close initial conditions are chosen.

$$T_{0_{100Hz}} = \frac{T_{s_{100Hz}}}{1-\lambda_{100Hz}} = \frac{0.01}{1-0.99609} = 2.5575\ s \tag{A.2}$$

$$\lambda_{1000Hz} = 1 - \frac{T_{s_{1000Hz}}}{T_{0_{100Hz}}} = 1 - \frac{0.001}{2.5575} = 0.999609 \tag{A.3}$$

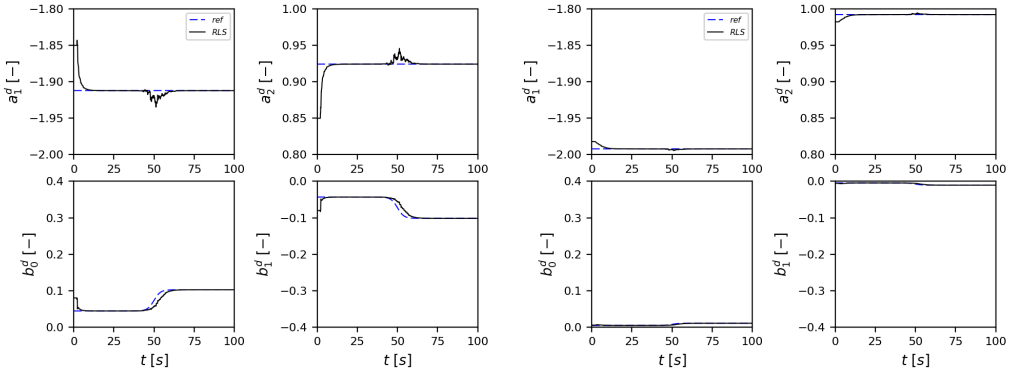
Finally, in parallel recursive ARX identification the best fitting model is selected at each time instant with the sVAF metric. Its window size is now adjusted to keep the same time span in the 1000 Hz case as in the 100 Hz case. This means more samples are used for the sVAF calculation in the 1000 Hz case.

A.2.2. Identification results at 1000 Hz sampling

Identification is first done at 1000 Hz for conditions C3-N0 and C3-N1-2. They can show whether recursive ARX identification improves with increased sampling frequency, adding to the work of (van Grootheest, 2017). Then time-varying time delay condition C5-N1-2 can be evaluated, requiring the use of parallel recursive ARX.

Constant delay, without remnant noise: For condition C3-N0, recursive ARX identification is applied while assuming a constant and known time delay ($n_k = 28$ for 100Hz and $n_k = 280$ for 1000Hz, thus $\tau = 0.28s$). In

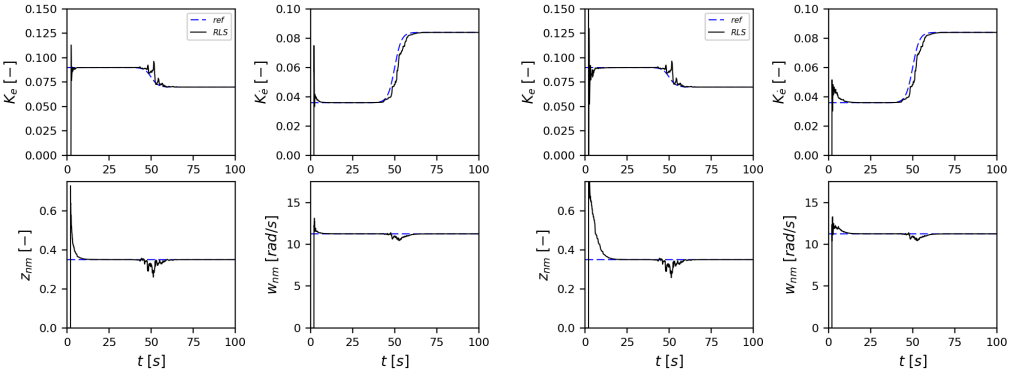
Figure A.4, the RLS with adjusted forgetting factor converges correctly in the 1000 Hz case. Figure A.5 shows the human operator model parameters, with no major differences in tracking performance between 100 Hz and 1000 Hz.



(a) 100 Hz.

(b) 1000 Hz.

Figure A.4: Time traces of estimated ARX coefficients for condition C3-N0, assuming $\tau = 0.28s$. Estimation with recursive ARX.

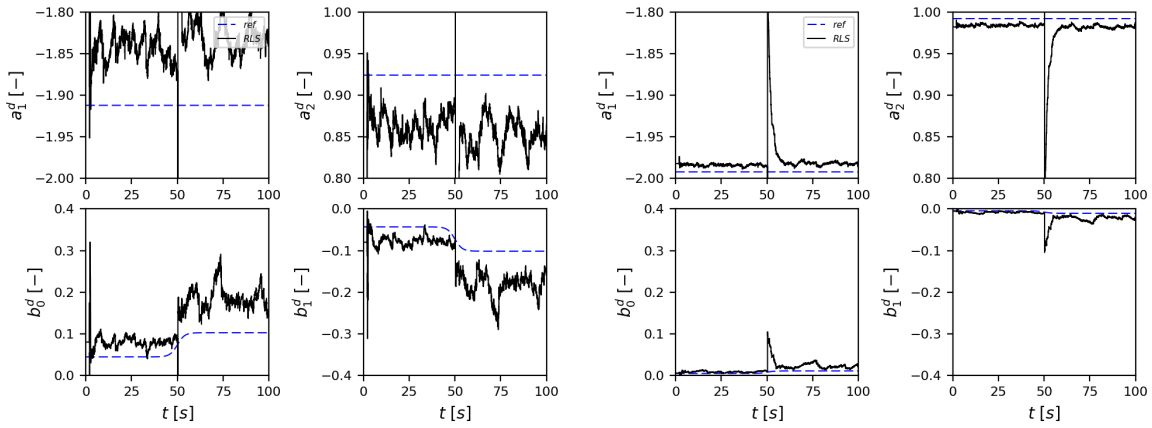


(a) 100 Hz.

(b) 1000 Hz.

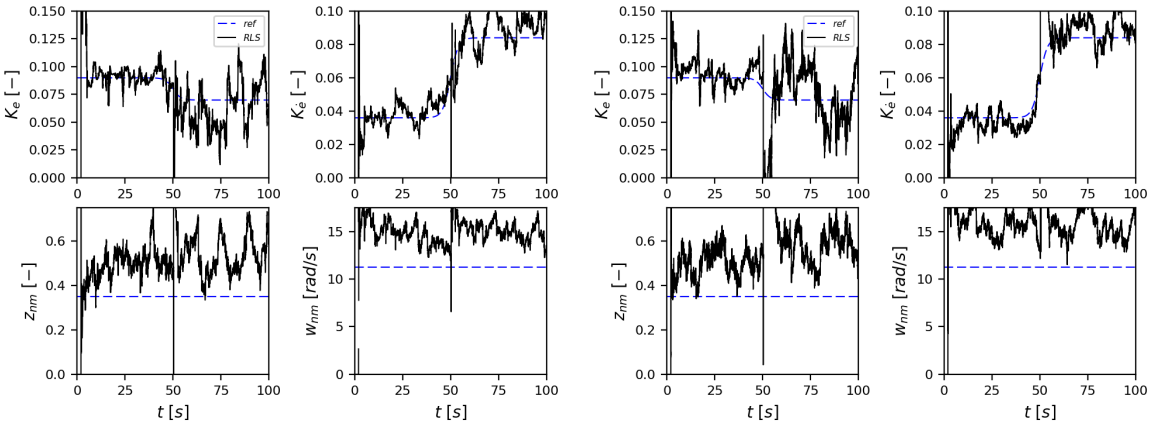
Figure A.5: Time traces of estimated operator model parameters for condition C3-N0, assuming $\tau = 0.28s$. Estimation with recursive ARX.

Constant delay, with remnant noise: Remnant noise is now added in the simulation. The remnant intensity in both the 100 Hz and 1000 Hz was checked to be the same. The identification results for the ARX coefficients in Figure A.6 show that both cases have a bias due to remnant while also showing the operator adaptation. Figure A.7 with operator model parameters gives a better view on the tracking performance in both cases. It can be seen that 1000 Hz sampling is not resulting in a major improvement.



(a) 100 Hz. (b) 1000 Hz.

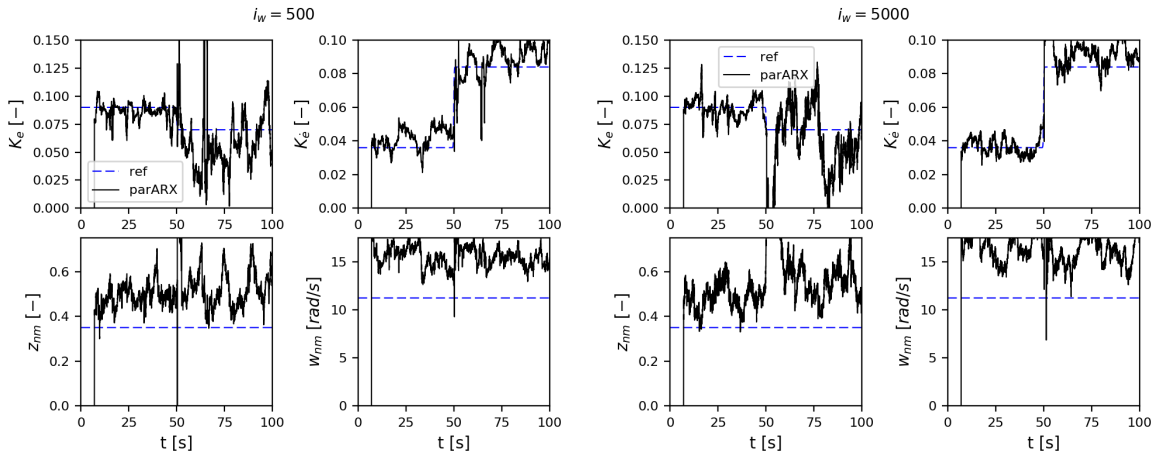
Figure A.6: Time traces of estimated ARX coefficients for condition C3-N1-2, assuming $\tau = 0.28s$. Estimation with recursive ARX.



(a) 100 Hz. (b) 1000 Hz.

Figure A.7: Time traces of estimated operator model parameters for condition C3-N1-2, assuming $\tau = 0.28s$. Estimation with recursive ARX.

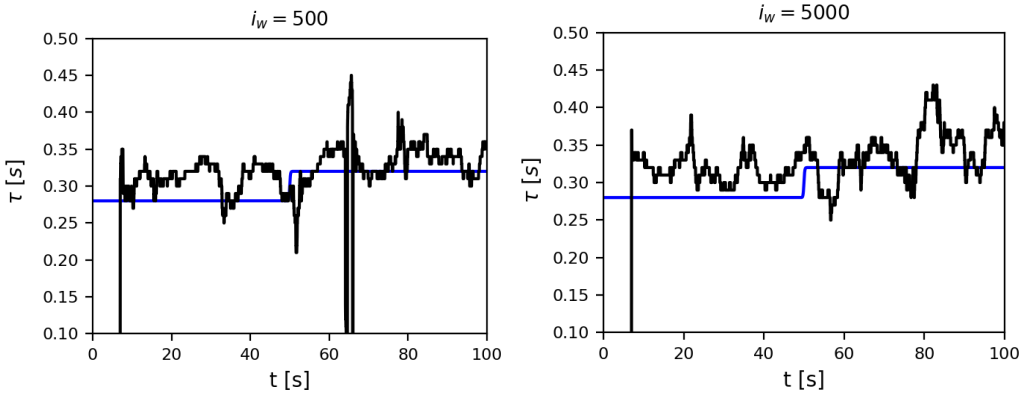
Time-varying delay, with remnant noise: For the time-varying delay case, condition C5-N1-2, the parallel recursive ARX method is used. Thus at every time step the best fitting ARX model and corresponding parameter estimates are chosen. Figure A.8 shows that the tracking of the model parameters does not improve with increasing sampling frequency. Additionally estimates of the time delay are shown in Figure A.9, with no significant improvement in tracking in the 1000 Hz case.



(a) 100 Hz, with window size $i_w = 500$.

(b) 1000 Hz, with adjusted window size $i_w = 5000$.

Figure A.8: Time traces of estimated operator model parameters for condition C5-N1-2. Estimation with parallel recursive ARX.



(a) 100 Hz, with window size $i_w = 500$.

(b) 1000 Hz, with adjusted window size $i_w = 5000$.

Figure A.9: Time traces of estimated time delay for condition C5-N1-2. Estimation with parallel recursive ARX.

A.3. Applying parallel recursive ARX on experimental data

As validation of the parallel recursive ARX method (parARX), it is applied to experimental data obtained in the experiment described in Part I. Additionally, it is compared with using a single recursively identified ARX model with a constant delay assumption (referred to as the singleARX method, with $n_k = 28$ and $\tau = 0.28$ s). Runs in condition TV12F for one subject are considered.

Figure A.10 shows identification results for a single experimental run. The singleARX method can start at $t = 0$ s, while parARX can only start at $t = 5$ s because of the sliding window used in the VAF calculation. The parameter traces are similar in both methods for K_e, ζ_{nm} and ω_{nm}. For the $K_{\dot{e}}$ parameter the traces diverge for some periods in time. When looking simultaneously at τ, in those periods its estimation is lacking. This is due to the fact that not one ARX model had a positive VAF value in those periods, as shown in Figure A.11. However, all the ARX models did deliver parameter estimation during those periods. This issue is not present in the singleARX method, a clear advantage in online identification.

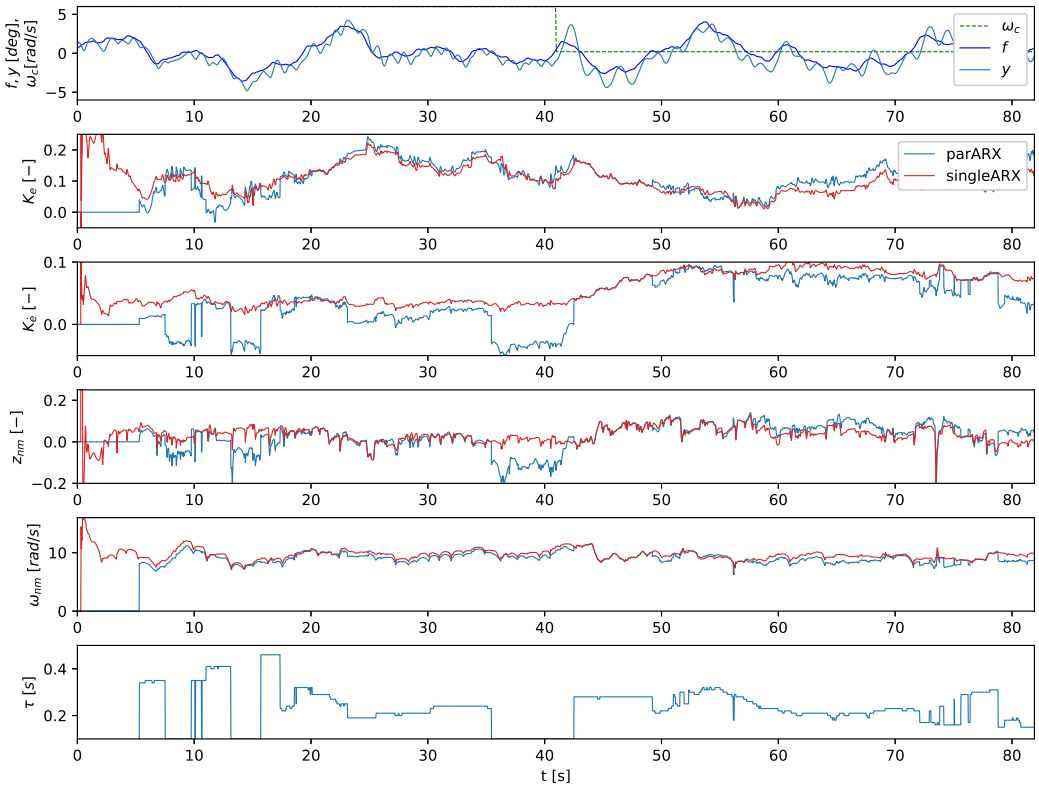


Figure A.10: Comparison of HO identification with parARX method ($i_w = 500$) and singleARX method, for a single run of a single subject in condition TV12F.

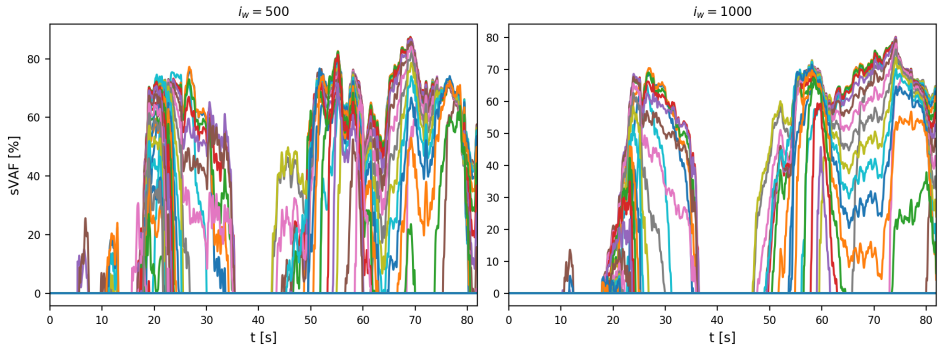


Figure A.11: Plot of sVAF for both sliding window sizes used in parARX method. Each trace presents the sVAF of an ARX model with a certain delay assumption. There are periods where all sVAF traces are zero.

The same comparison was done when averaging identification results over all runs done by the subject in condition TV12F, resulting in Figure A.12. The drop-outs in the estimation with parARX are now smoothed by the averaging. The τ estimation shows on average a relatively constant delay before and after the adaptation, as expected for this condition according to (Zaal, 2016).

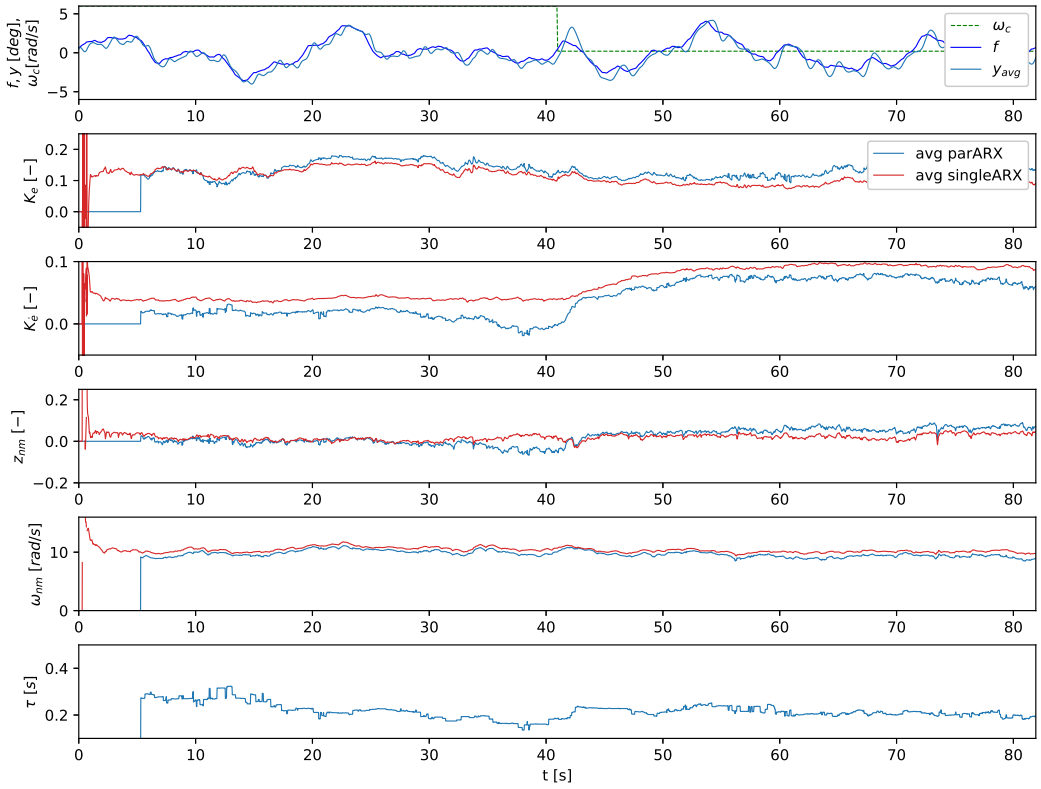


Figure A.12: Comparison of HO identification with parARX method ($i_w = 500$) and singleARX method, averaged for all runs of a subject in condition TV12F.

B

Appendices to scientific article

B.1. Implementing recursive ARX in DUECA

For the experiment described in the scientific article, the recursive ARX identification method was implemented inside an existing DUECA project "TVDynamicsExp" used in (van Grootheest, 2017). The modified project was called "TVIdentExp". The modifications are listed below and indicated in Figure B.1.

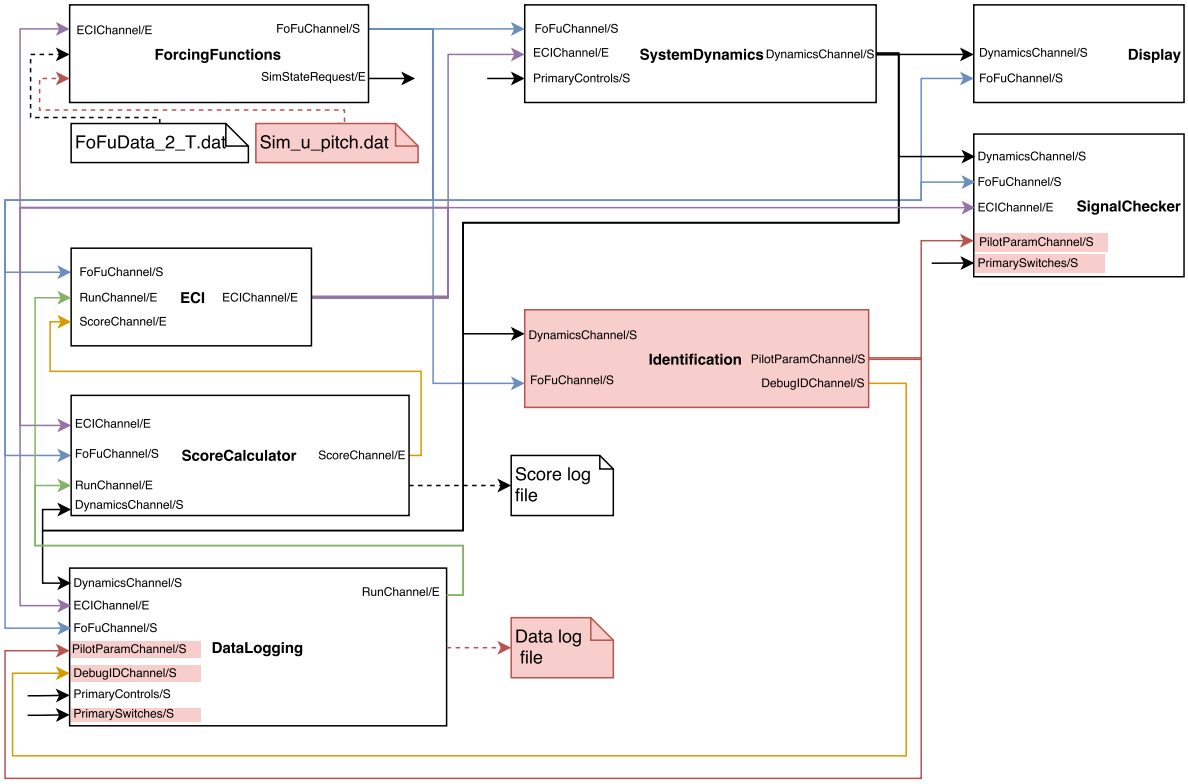


Figure B.1: Flow chart of DUECA project "TVIdentExp", showing modules and channels. Modifications with respect to the "TVDynamicsExp" project are indicated in red.

Modifications, classified by module:

- **Identification**: a new module was added in which the recursive ARX method and the conversion of the ARX coefficients to operator model parameters were implemented. Two output streaming channels were created, namely: PilotParam channel containing the identified operator model parameters and Debug channel containing recursive least squares debug information.
- **SignalChecker**: in the window shown at the HMIlab monitoring station, plots were added showing time traces of the ARX coefficients and the operator parameters. Furthermore a plot was added to show the status of the frontal push button on the side stick. This information came from the PrimarySwitches channel.

- **Datalogging**: added logging of PilotParam, Debug and PrimarySwitches channels to the log file.
- **ECI**: the experiment conditions described in the scientific article were added as selectable options to the experiment control interface.
- **ForcingFunctions**:
  - in the input FoFuData_2_T.dat file, the forcing function was shifted to the left by adjusting the phases as specified in the scientific article.
  - Furthermore a replay mode was added which forces DUECA to load the control signal and pitch signal from Sim_u_pitch.dat and replay it through the FoFuChannel.
  - To allow for an adjustable run-in time of the simulation, the forcing function length is adjusted. Additional run-in time is seen as negative time t.

B.1.1. Verification

To verify the new identification module, a replay mode was incorporated in the DUECA simulation. If enabled, both the human operator control signal and pitch signal are loaded from a file instead of being generated by the operator. Identification is then done using the loaded control signal and the resulting error signal. The operator parameter traces are then compared with results from identification in the developed Python simulation using the same input traces. The Python simulation was already verified in Chapter 6.

In Figure B.2 the traces for the operator parameters are compared, showing no difference after convergence from the initial condition to the actual parameters. The initial convergence might be different between both cases due to numerical precision errors.

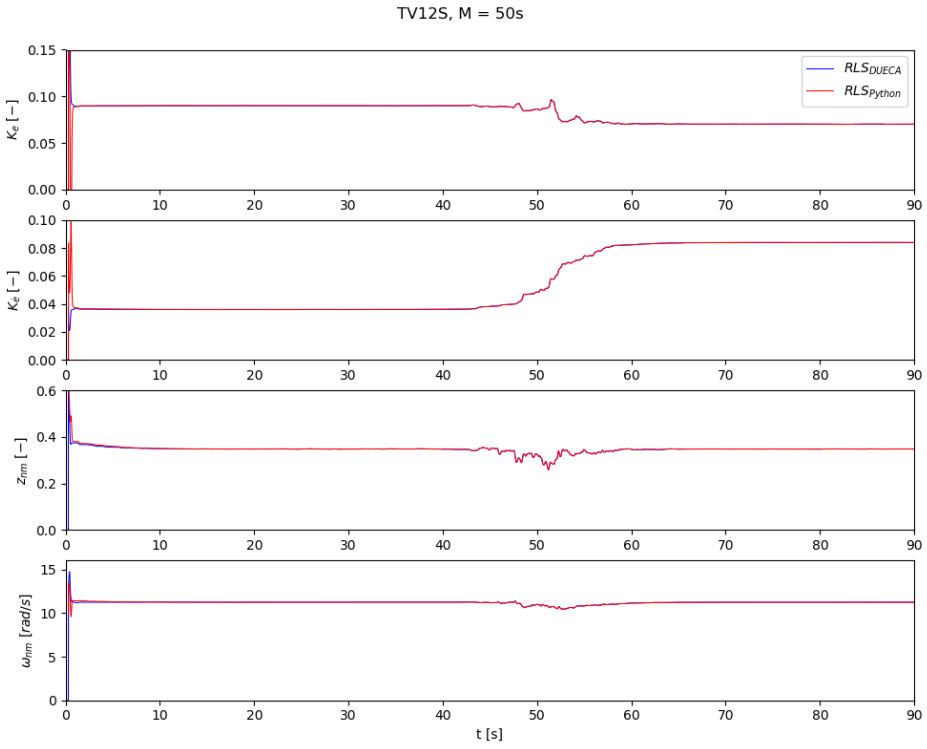


Figure B.2: Comparison between identification in DUECA and Python based on the same control and error signals.

B.2. Experiment briefing for participants

EXPERIMENT BRIEFING

ONLINE TIME-VARYING PILOT MODEL IDENTIFICATION

Thank you for your contribution to this scientific endeavour! You will be participating in a tracking experiment in the research simulator in the HMI laboratory. The adaptation of the human controller to changing controlled element dynamics is investigated. This briefing will introduce you to the experiment and what is expected of you as a participant.

GOAL OF THE EXPERIMENT

The goal of this experiment is to investigate the performance of an online identification algorithm to identify a time-varying model of the adapting human controller. The results can be used to evaluate the algorithm's performance and its use in future operator adaptation detection tools.

PITCH TRACKING TASK WITH CHANGING DYNAMICS

The task you will be trained in is a pitch tracking task. In this task, it is your goal to continuously minimize a deviation of the current pitch angle from a desired pitch angle, as shown in Figure 1. The tracking task can be compared to a glide-slope following task in an aircraft, in which the only variable controlled is the pitch angle.

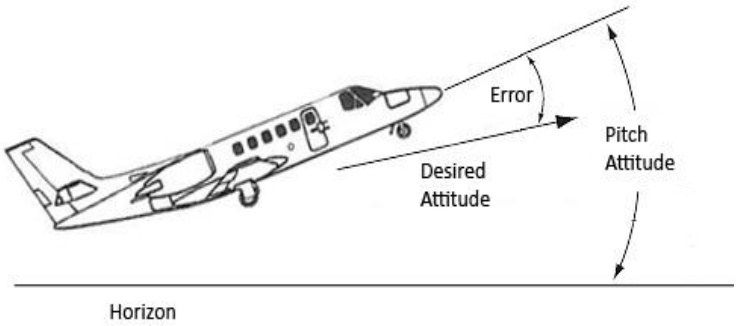


FIGURE 1: SCHEMATIC REPRESENTATION OF A PITCH TRACKING TASK

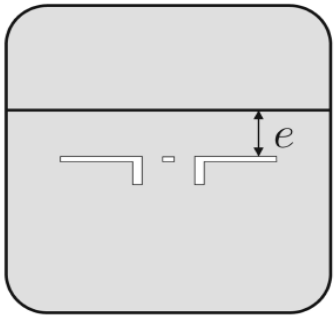


FIGURE 2: COMPENSATORY DISPLAY

The error between the desired and current pitch angle will be displayed on the primary flight display, shown in Figure 2. The aircraft's attitude is displayed by fixed wings and the error is displayed by a moving horizontal line, on a contrasting background. It is important that you try to keep the error as small as possible. For giving control inputs, you will use a side-stick on the right-hand side of the seat.

In the experiment, the dynamics of the system you are controlling might change during the tracking run. In addition to minimizing the tracking error, it is your task to shortly press the button on the front of the stick (Figure 3) as soon as possible after you detect a change in the controlled system dynamics.

FIGURE 3: control stick with front push button indicated.

PROCEDURE

First a training session will familiarize you with the tracking task at hand for all considered system dynamics to be controlled. Furthermore some practice runs with changing dynamics will be done where you will also practice indicating a change in dynamics happened through the button push.

This initial training is followed by the real experiment session, in which all different experiment conditions (changes in system dynamics) will be presented in a randomized order. The full experiment will last around 60mins, with a break halfway. In case you feel you are getting tired or losing focus, please indicate this: a small break can be taken in between any two runs.

Each tracking run, the subsequent procedure is followed:

1. The researcher applies the settings for the next run
2. The researcher checks whether the participant is ready to proceed and initiates the run
3. The participant performs the tracking task and gives a sign when he notices a different system behaviour by pushing the stick button, the button push is confirmed by the researcher
4. The researcher informs the participant of the performance in the completed run
5. The researcher checks whether the participant feels any signs of reduced focus

B.3. Experiment consent form template

EXPERIMENT CONSENT FORM

ONLINE TIME-VARYING PILOT MODEL IDENTIFICATION

Researcher: Wouter Plaetinck Supervisor: Daan Pool

Name: _____ Age: ___

Have you participated in a tracking experiment before: Yes/No

Do you play computer games regularly: Yes, I play(ed) _____hour per week /No

Do you have a pilot licence or flying experience: Yes/No

Are you prone to motion sickness: Yes/No

If you are sensitive to motion sickness, it is firmly recommended that you do not participate in this experiment

Please provide your signature below to indicate that you agree to participate in this experiment. Signing this form does not annul the responsibilities of the researcher and Delft University of Technology towards you as a participant.

I hereby confirm that I have read the experiment briefing. Also, I affirm that I understand the experiment instructions, and I declare that I voluntarily participate in this experiment. Finally, I have been informed of the fact that I can opt-out of participating in the experiment at any time.

DATE: SIGNATURE:

B.4. Experiment results

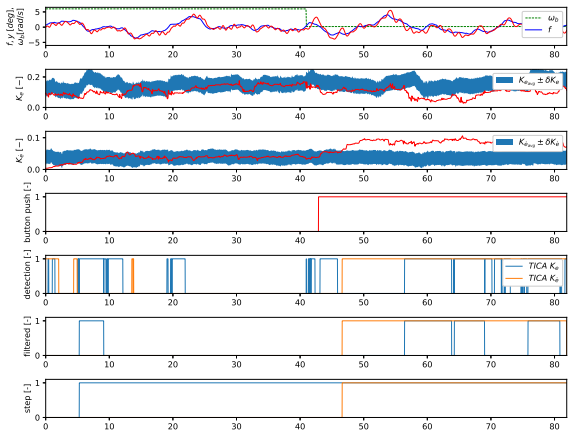
This section contains figures with time traces showing adaptation detection results per run. Only the final repeated measurement of each run-in time level is shown, classified per time-varying condition (TV12F and TV12S) and per subject. The hyperparameters used for both adaptation detection methods are in Table B.1. The figures will help the future researcher to investigate cases where the adaptation detection did not work or triggered a detection too late.

Table B.1: Hyperparameters used in both methods for the shown figures.

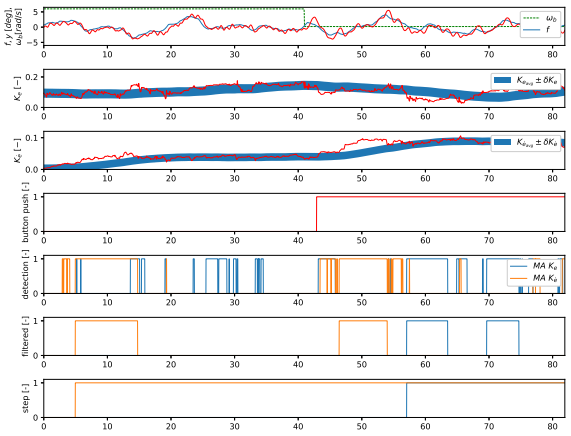
	δK_e [-]	$\delta K_{\dot{e}}$ [-]	ΔT [s]	n_s [-]
TICA	0.05	0.02	3	-
MA	0.03	0.012	3	1500

B.4.1. Condition TV12F

Subject 1

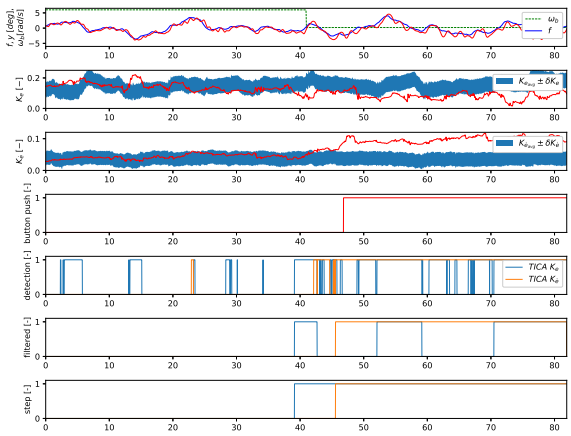


(a) TICA method

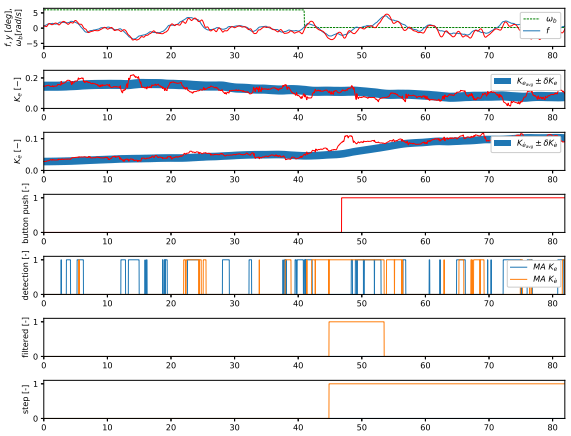


(b) MA method

Figure B.3: Run-in time 5 s, run 26.

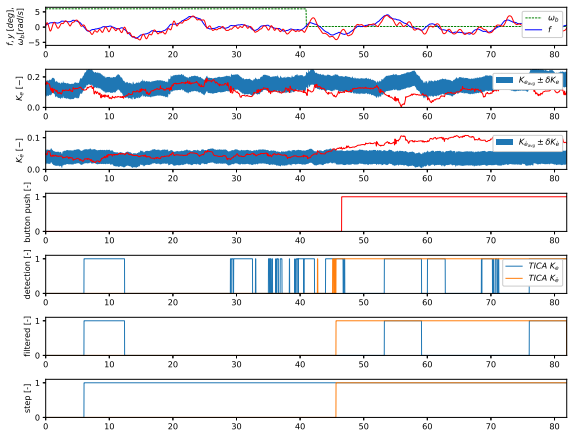


(a) TICA method

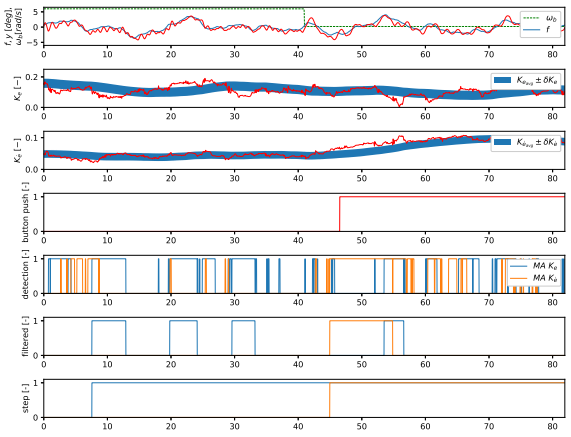


(b) MA method

Figure B.4: Run-in time 10 s, run 22.



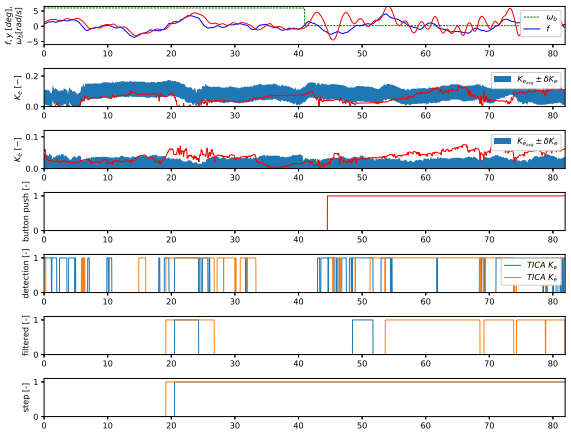
(a) TICA method



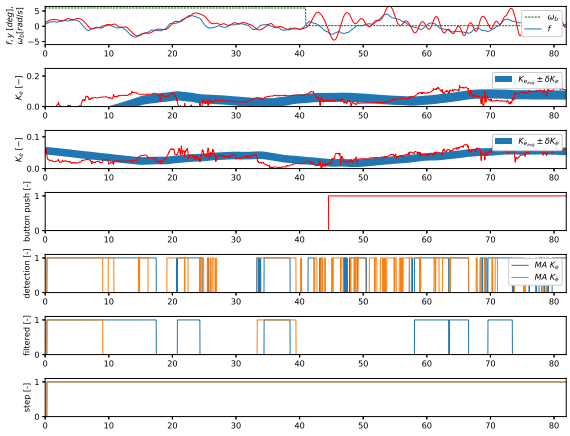
(b) MA method

Figure B.5: Run-in time 15 s, run 23.

Subject 2

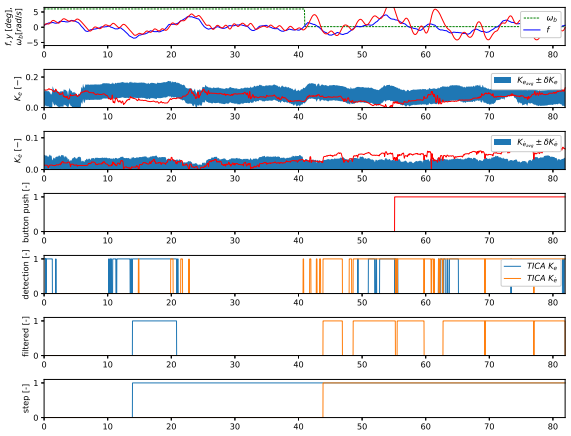


(a) TICA method

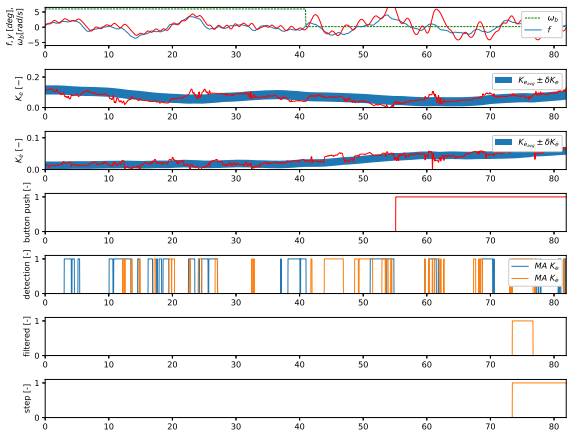


(b) MA method

Figure B.6: Run-in time 5 s, run 25.

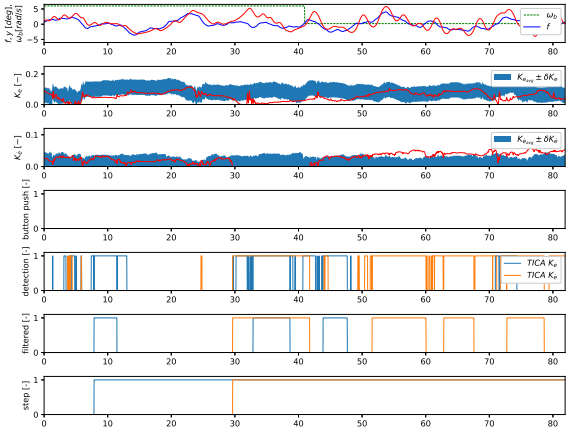


(a) TICA method

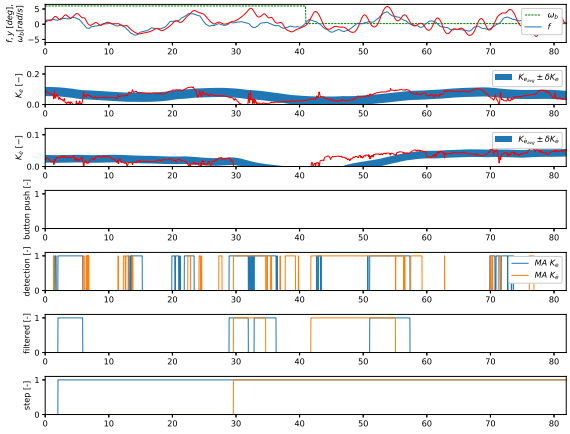


(b) MA method

Figure B.7: Run-in time 10 s, run 26.



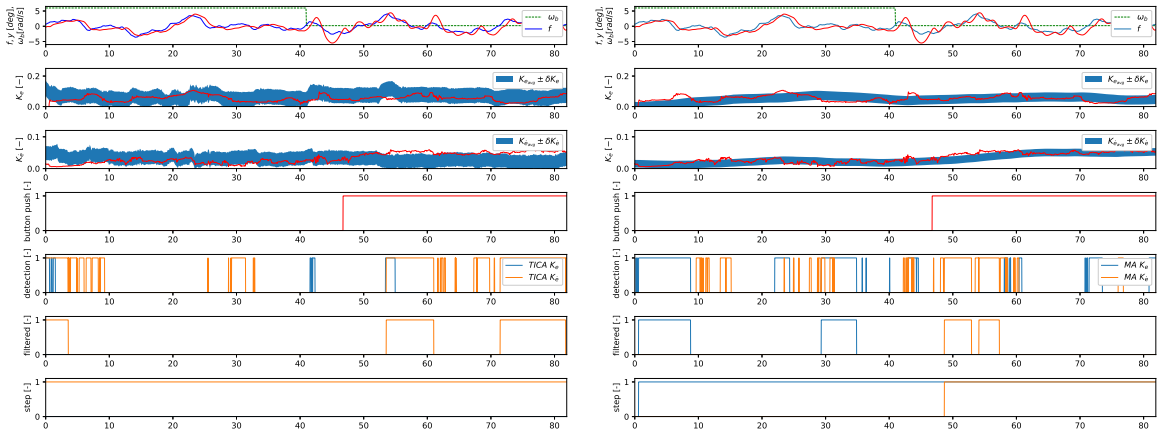
(a) TICA method



(b) MA method

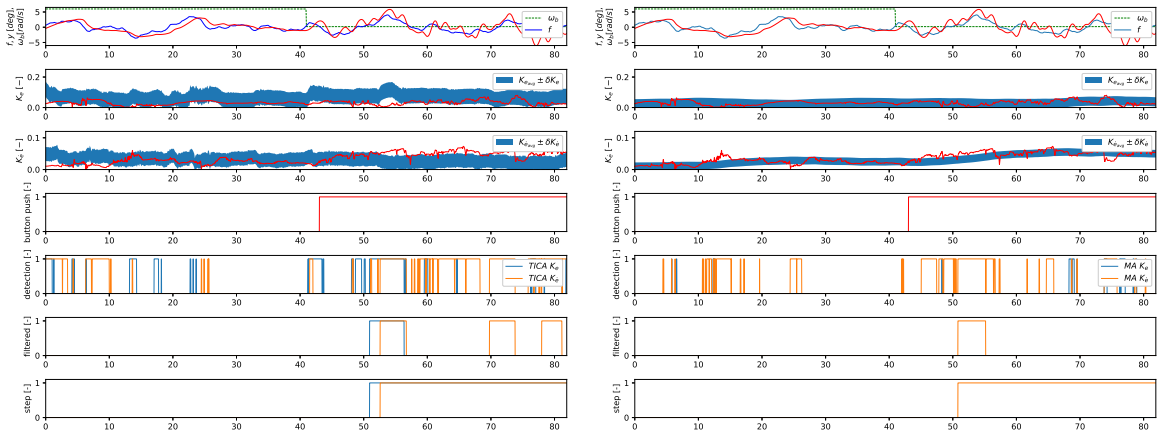
Figure B.8: Run-in time 15 s, run 29.

Subject 3



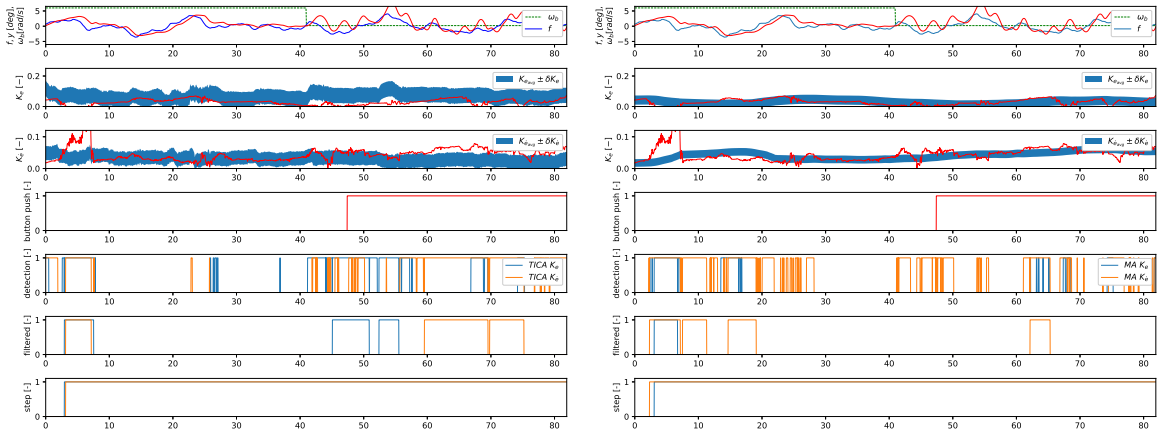
(a) TICA method (b) MA method

Figure B.9: Run-in time 5 s, run 26.



(a) TICA method (b) MA method

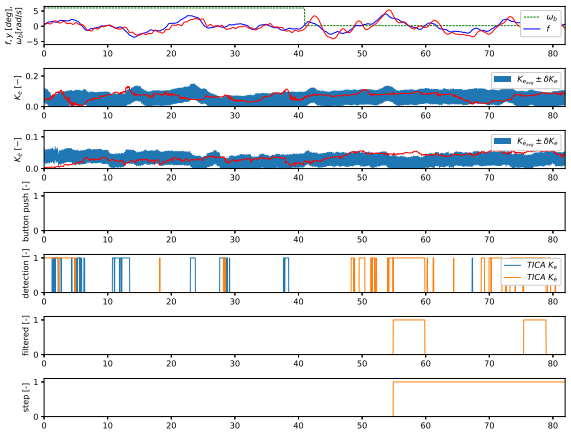
Figure B.10: Run-in time 10 s, run 25.



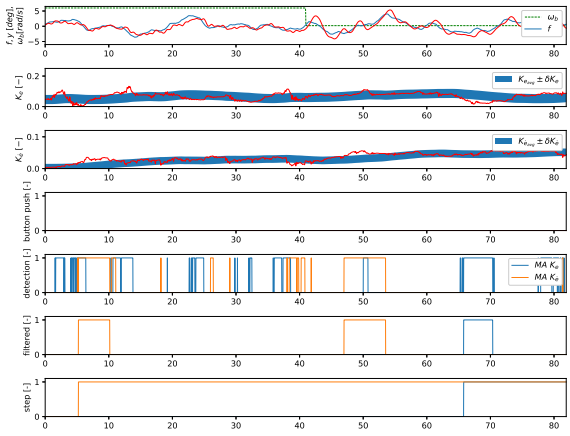
(a) TICA method (b) MA method

Figure B.11: Run-in time 15 s, run 29.

Subject 4

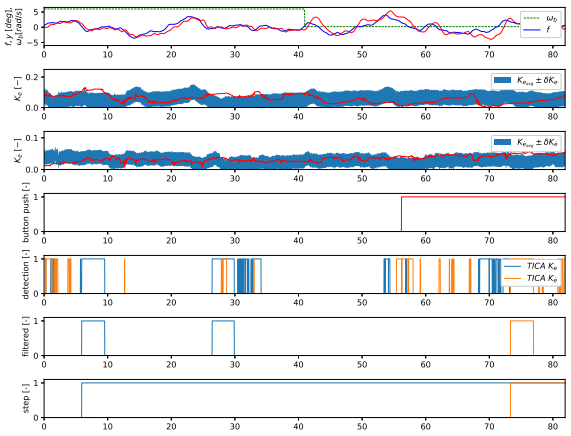


(a) TICA method

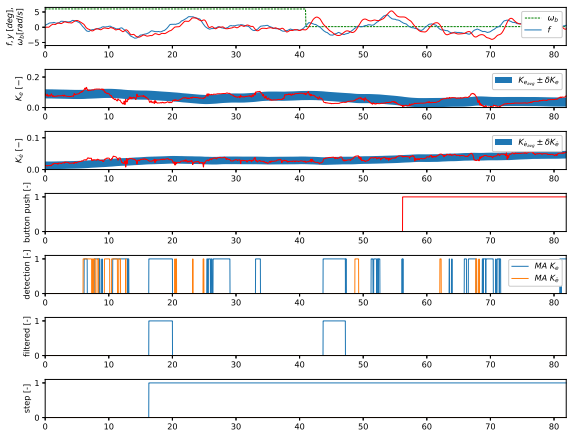


(b) MA method

Figure B.12: Run-in time 5 s, run 23.

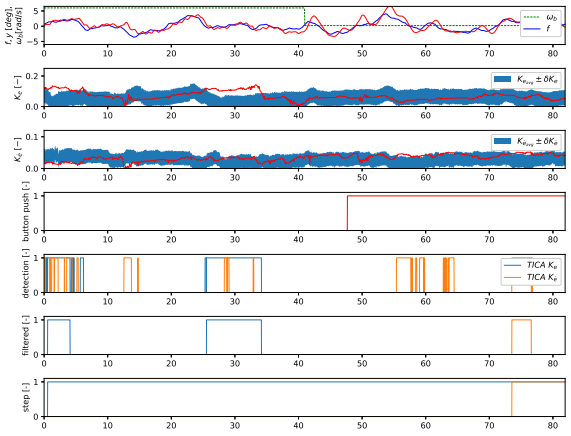


(a) TICA method

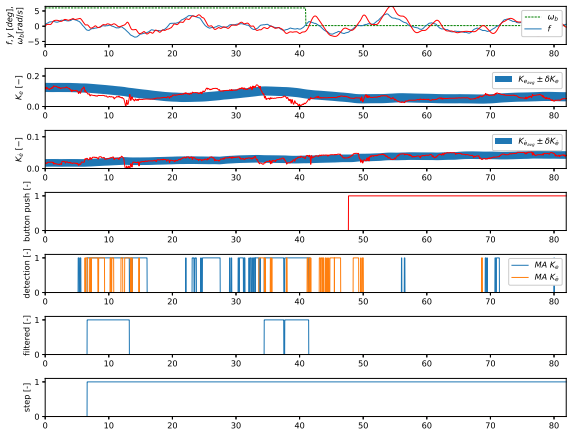


(b) MA method

Figure B.13: Run-in time 10 s, run 26.



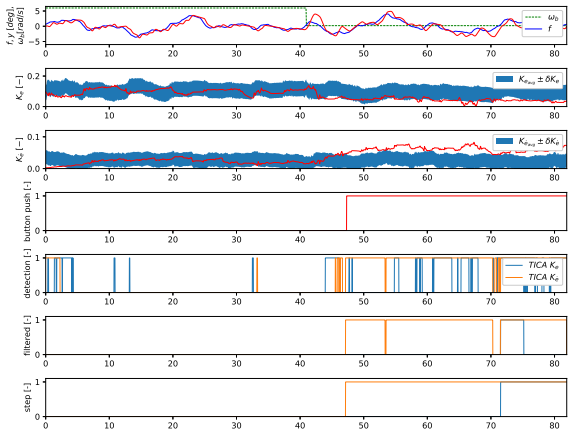
(a) TICA method



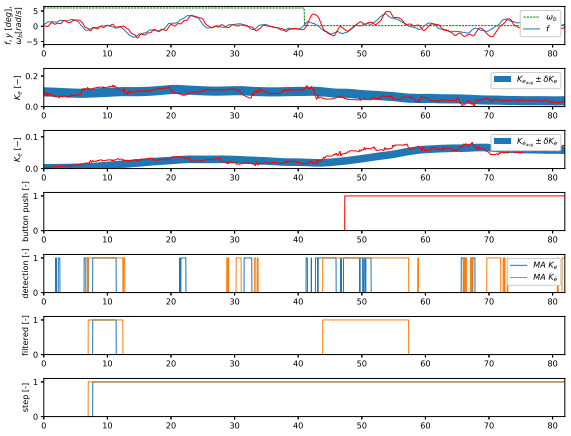
(b) MA method

Figure B.14: Run-in time 15 s, run 25.

Subject 5

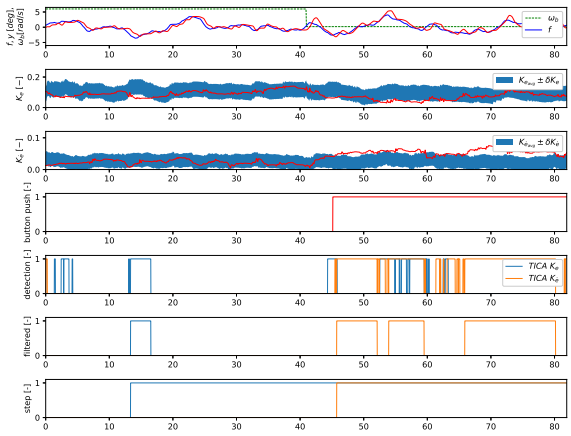


(a) TICA method

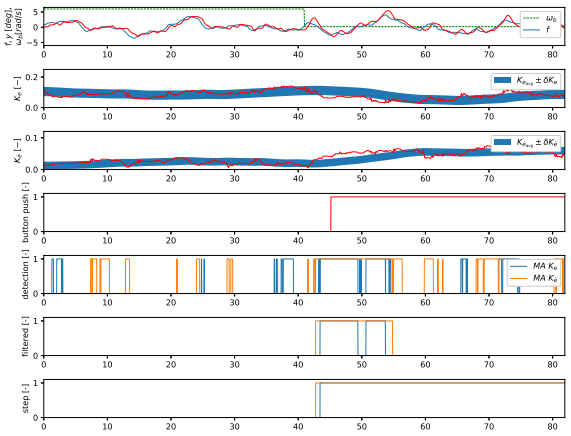


(b) MA method

Figure B.15: Run-in time 5 s, run 24.

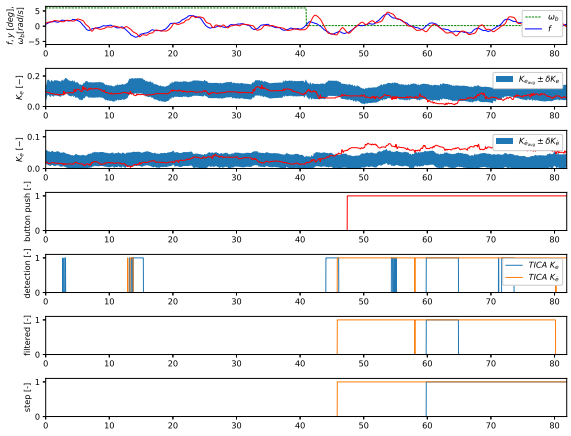


(a) TICA method

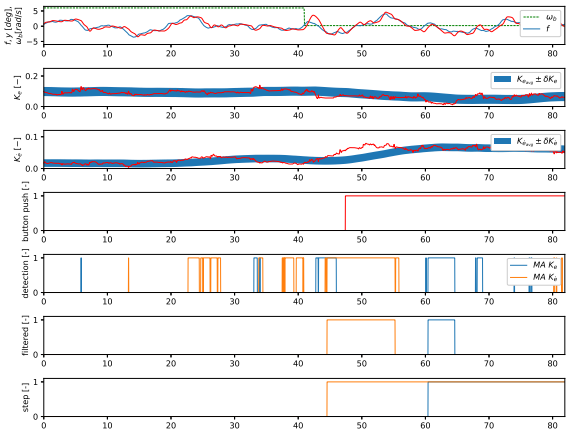


(b) MA method

Figure B.16: Run-in time 10 s, run 27.



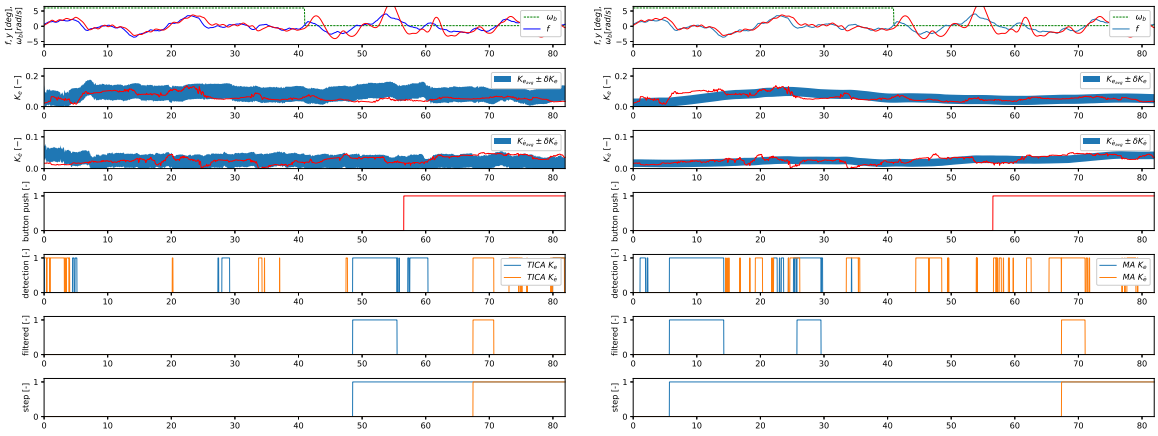
(a) TICA method



(b) MA method

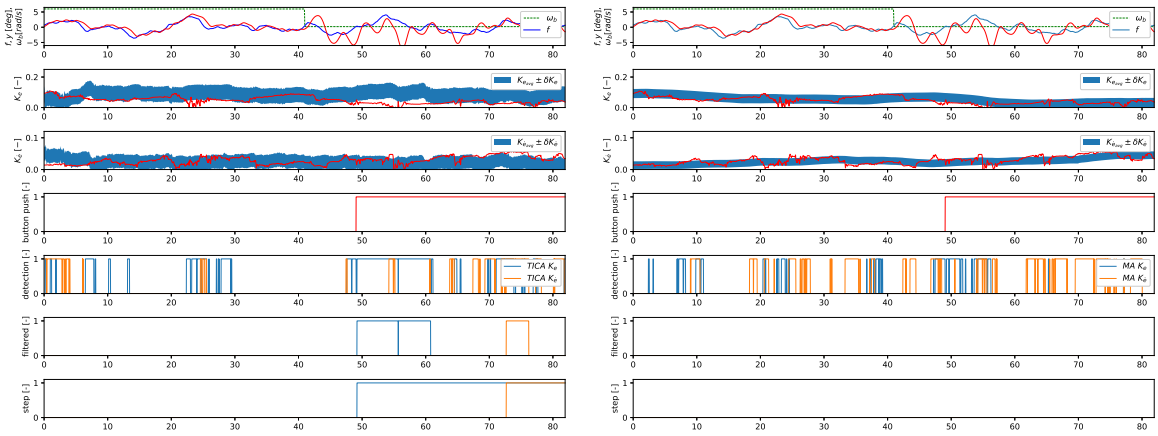
Figure B.17: Run-in time 15 s, run 25.

Subject 6



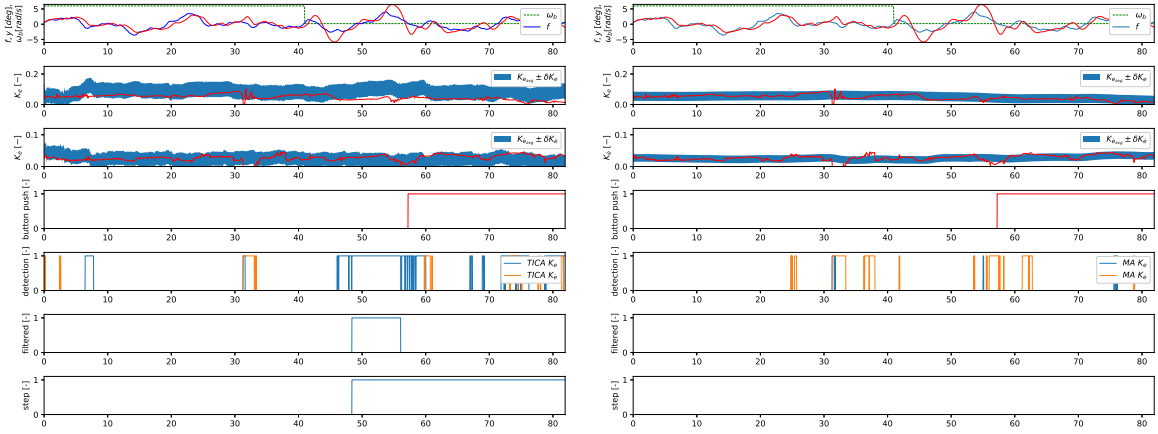
(a) TICA method (b) MA method

Figure B.18: Run-in time 5 s, run 24.



(a) TICA method (b) MA method

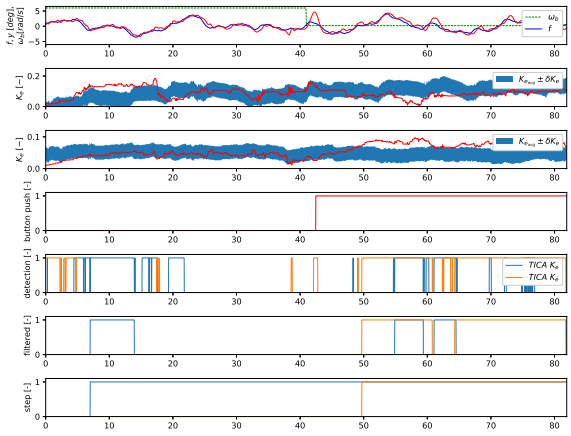
Figure B.19: Run-in time 10 s, run 26.



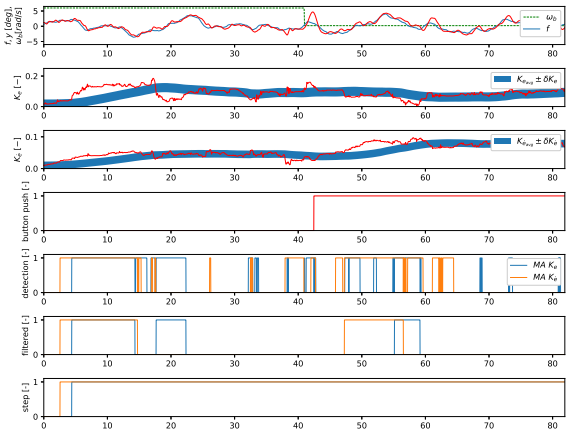
(a) TICA method (b) MA method

Figure B.20: Run-in time 15 s, run 30.

Subject 7

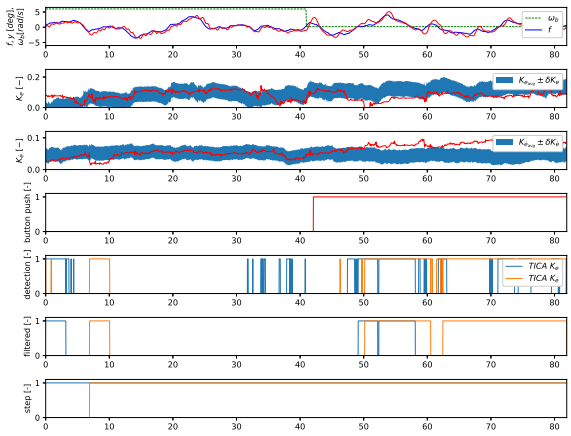


(a) TICA method

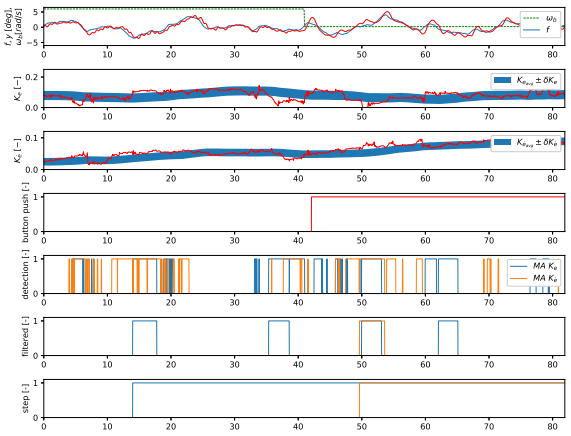


(b) MA method

Figure B.21: Run-in time 5 s, run 22.

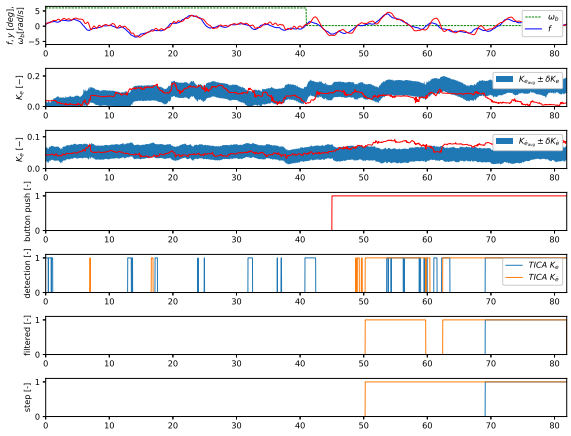


(a) TICA method

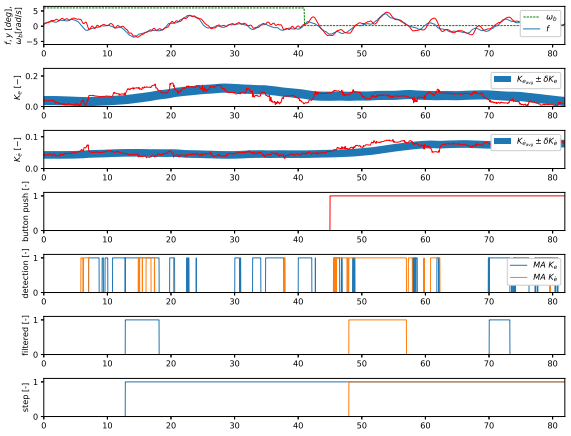


(b) MA method

Figure B.22: Run-in time 10 s, run 24.



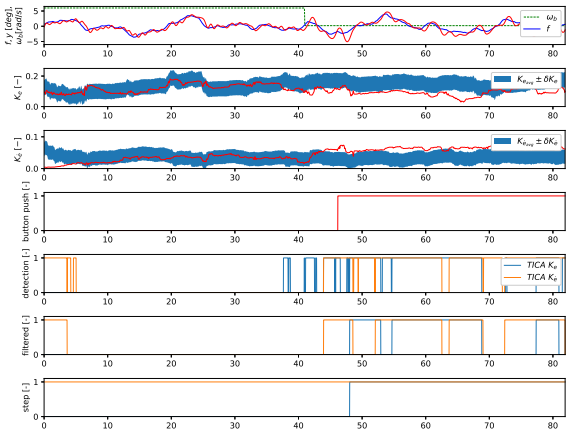
(a) TICA method



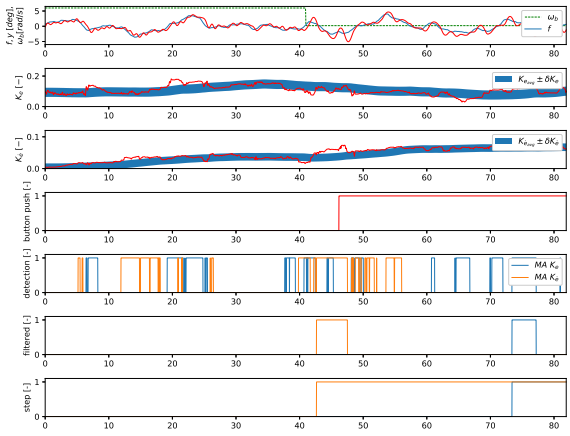
(b) MA method

Figure B.23: Run-in time 15 s, run 25.

Subject 8

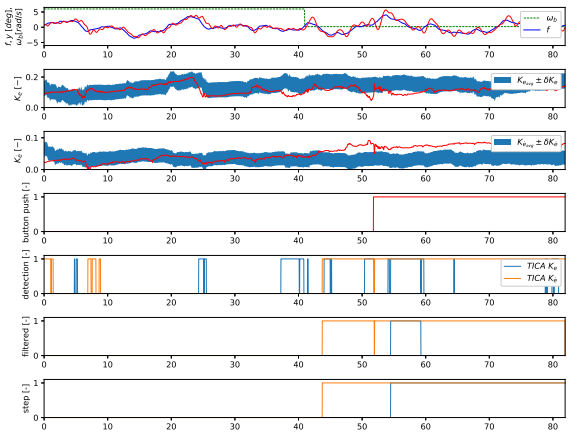


(a) TICA method

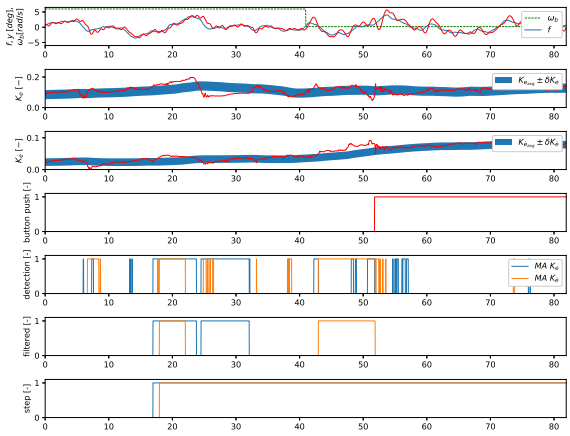


(b) MA method

Figure B.24: Run-in time 5 s, run 24.

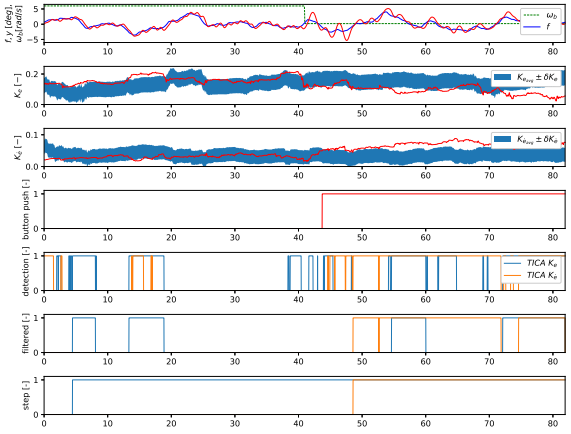


(a) TICA method

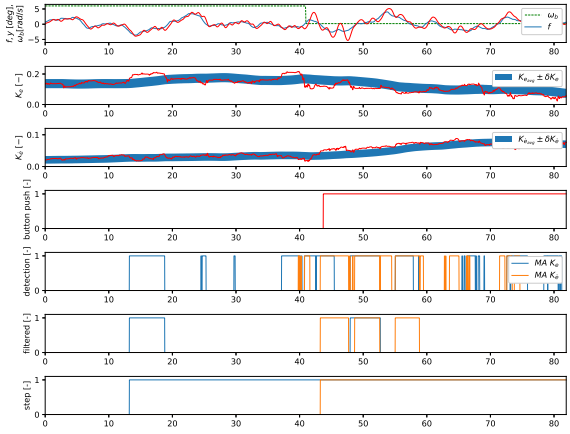


(b) MA method

Figure B.25: Run-in time 10 s, run 23.



(a) TICA method

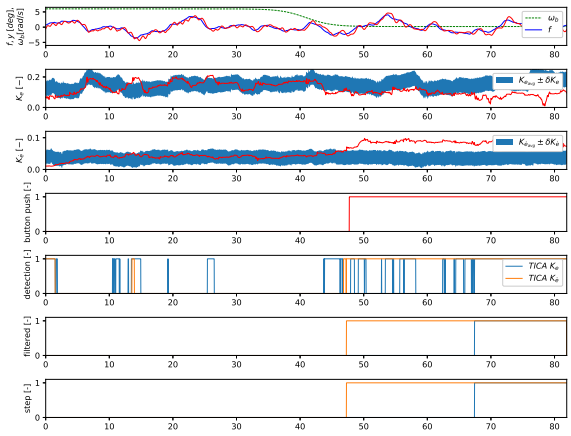


(b) MA method

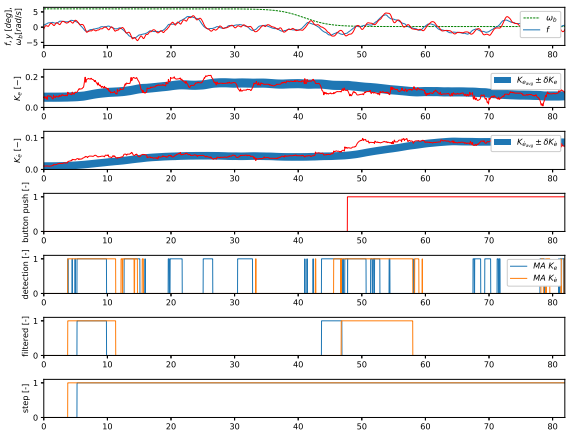
Figure B.26: Run-in time 15 s, run 27.

B.4.2. Condition TV12S

Subject 1

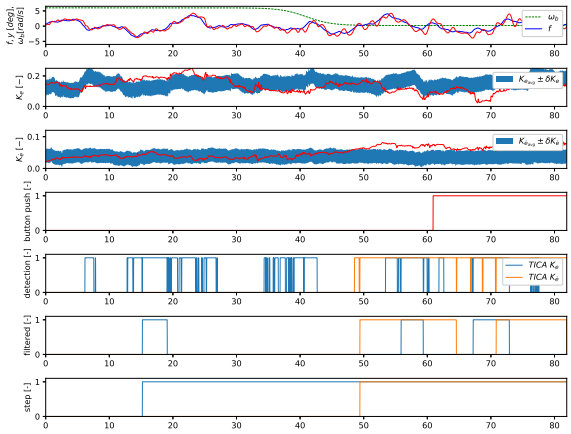


(a) TICA method

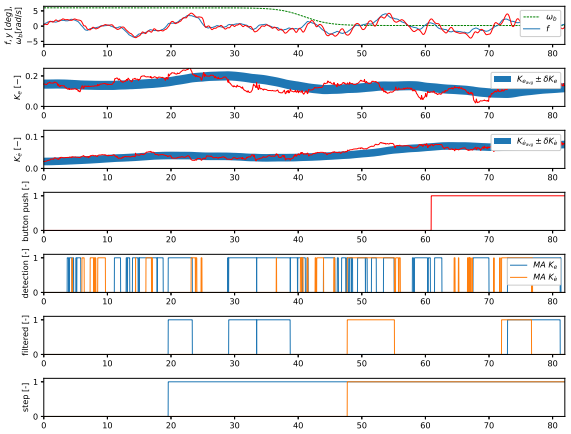


(b) MA method

Figure B.27: Run-in time 5 s, run 24.

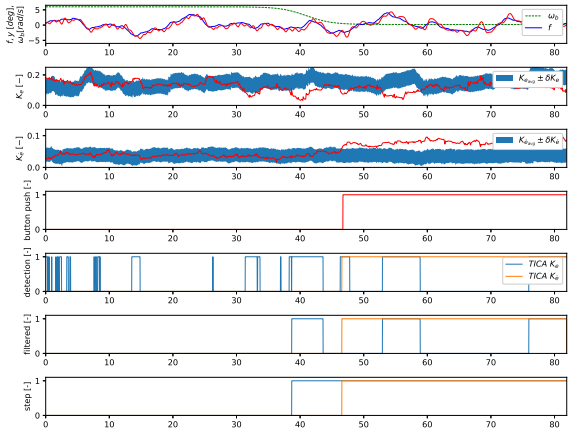


(a) TICA method

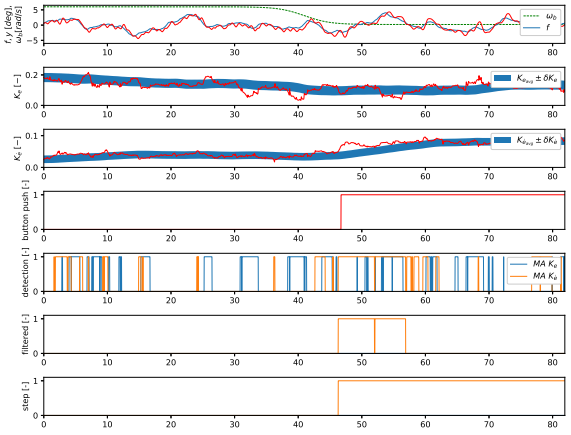


(b) MA method

Figure B.28: Run-in time 10 s, run 28.



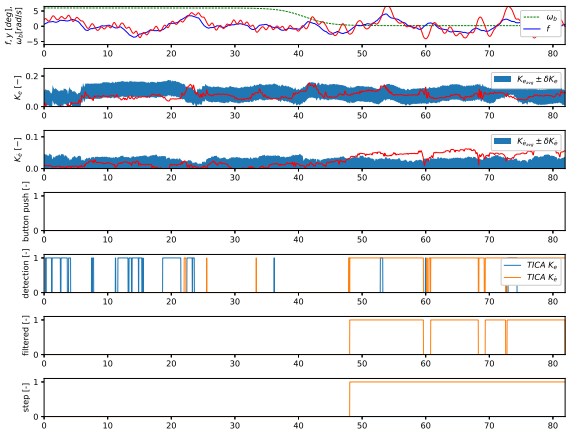
(a) TICA method



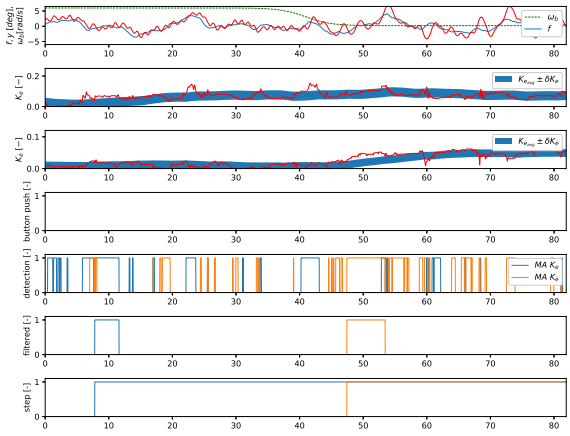
(b) MA method

Figure B.29: Run-in time 15 s, run 25.

Subject 2

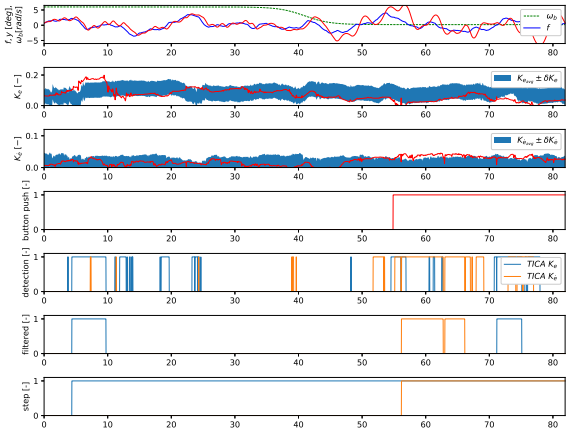


(a) TICA method

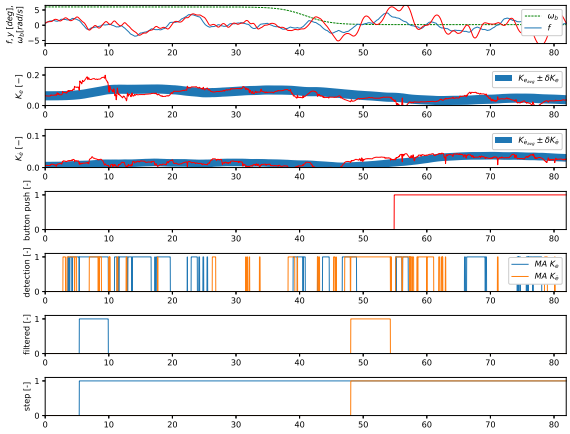


(b) MA method

Figure B.30: Run-in time 5 s, run 27.

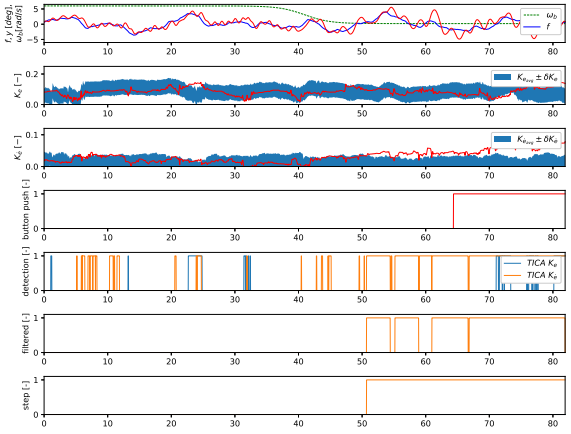


(a) TICA method

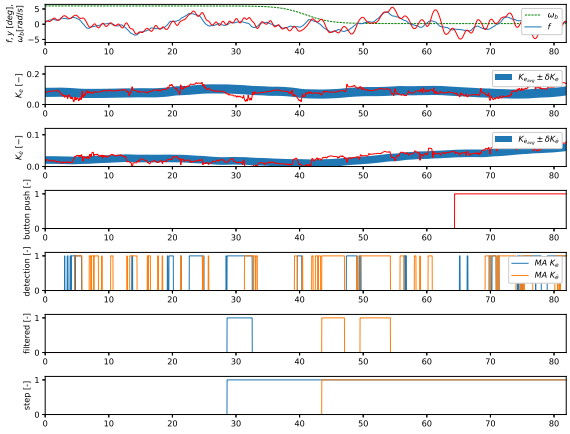


(b) MA method

Figure B.31: Run-in time 10 s, run 31.



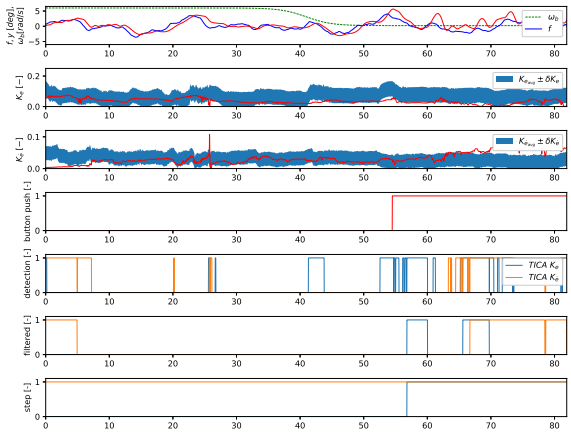
(a) TICA method



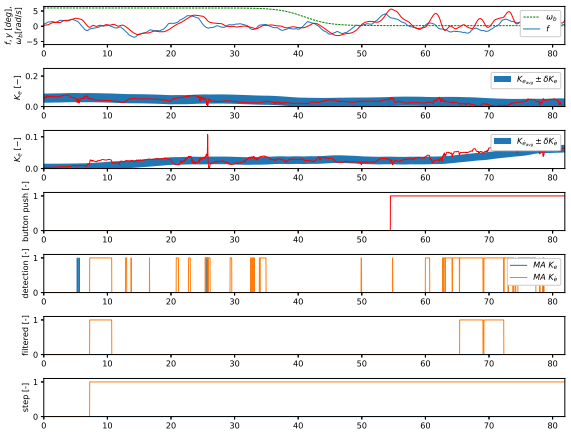
(b) MA method

Figure B.32: Run-in time 15 s, run 28.

Subject 3

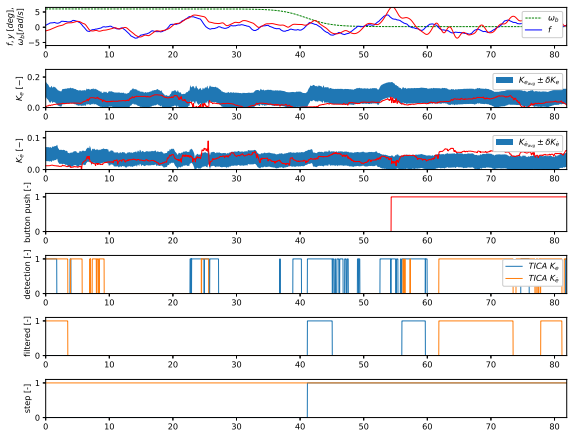


(a) TICA method

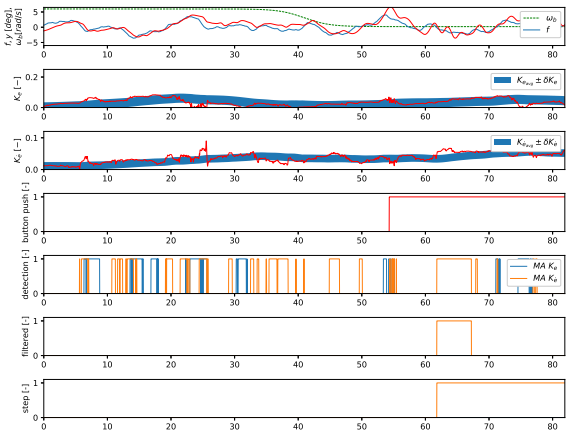


(b) MA method

Figure B.33: Run-in time 5 s, run 23.

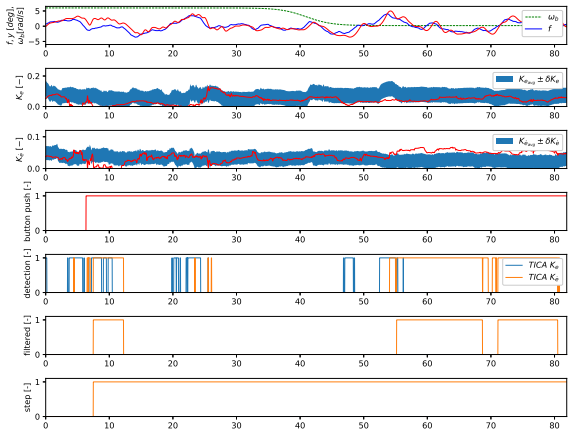


(a) TICA method

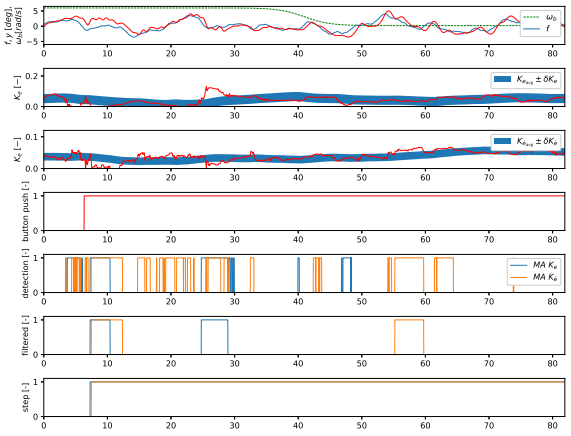


(b) MA method

Figure B.34: Run-in time 10 s, run 28.



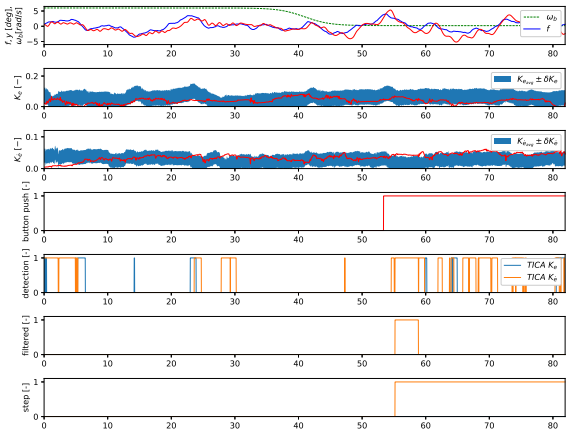
(a) TICA method



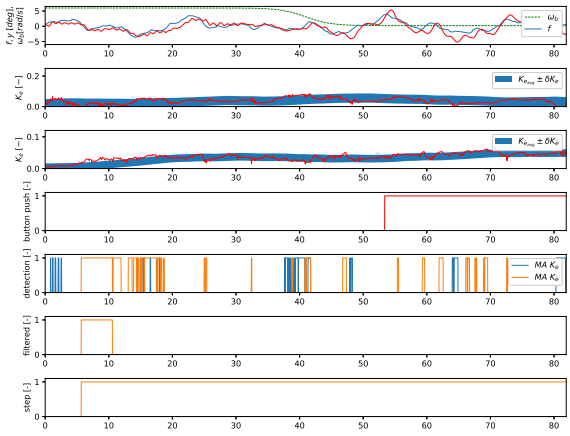
(b) MA method

Figure B.35: Run-in time 15 s, run 22.

Subject 4

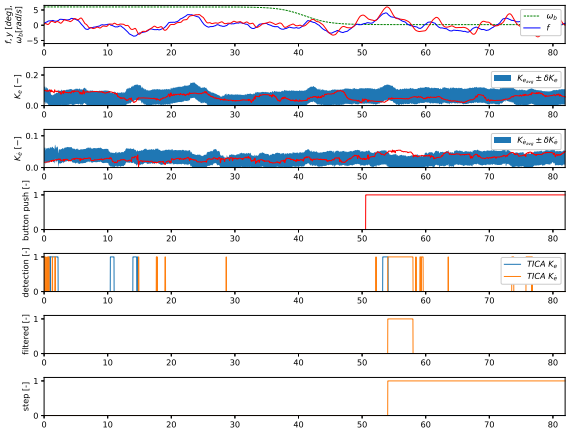


(a) TICA method

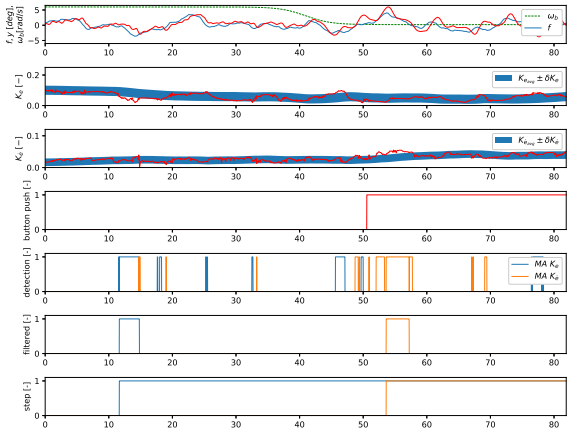


(b) MA method

Figure B.36: Run-in time 5 s, run 28.

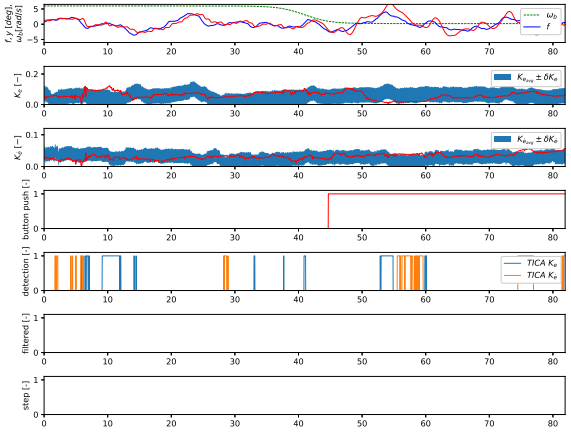


(a) TICA method

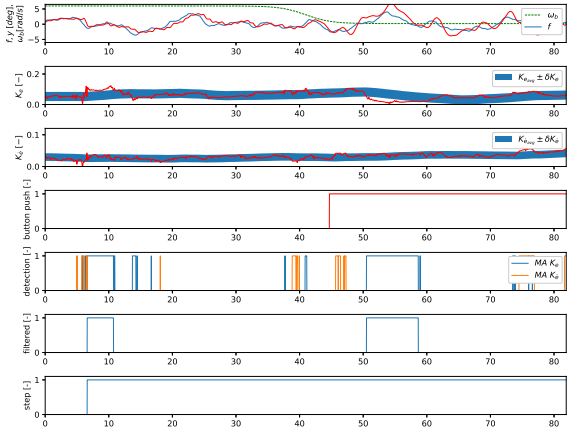


(b) MA method

Figure B.37: Run-in time 10 s, run 21.



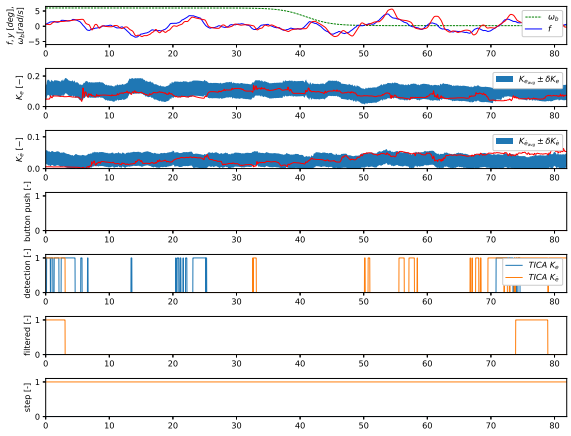
(a) TICA method



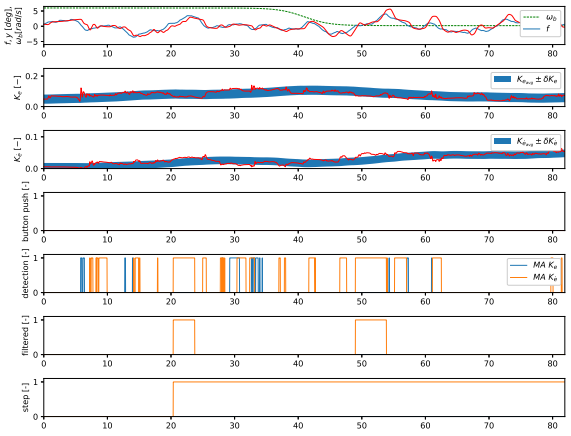
(b) MA method

Figure B.38: Run-in time 15 s, run 22.

Subject 5

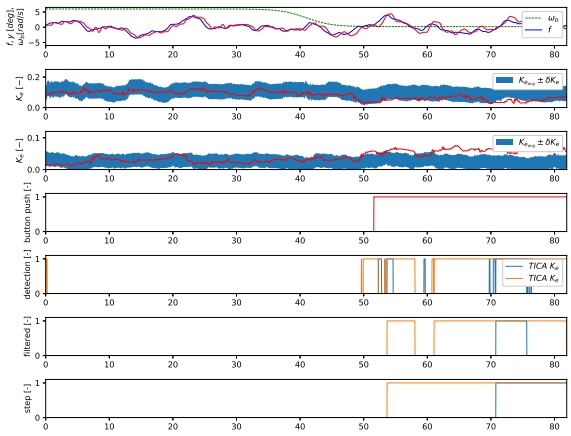


(a) TICA method

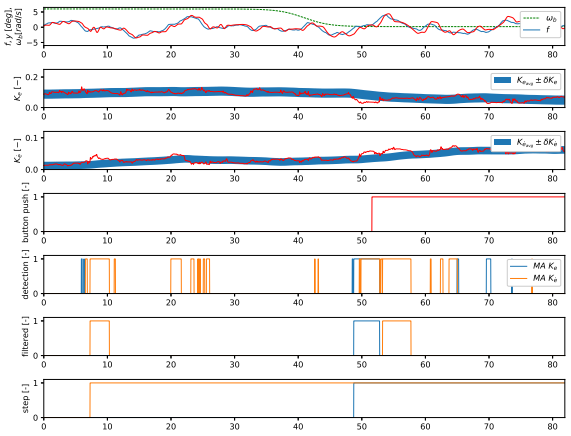


(b) MA method

Figure B.39: Run-in time 5 s, run 21.

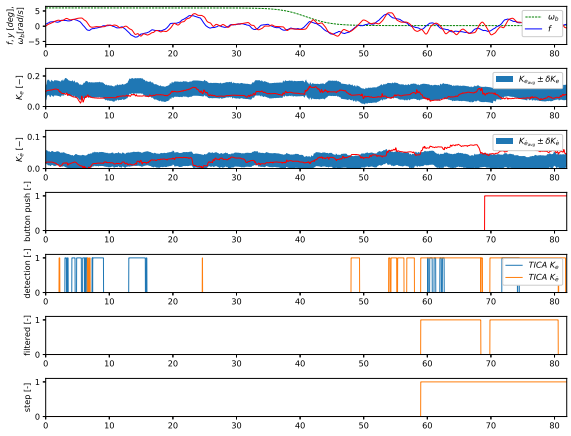


(a) TICA method

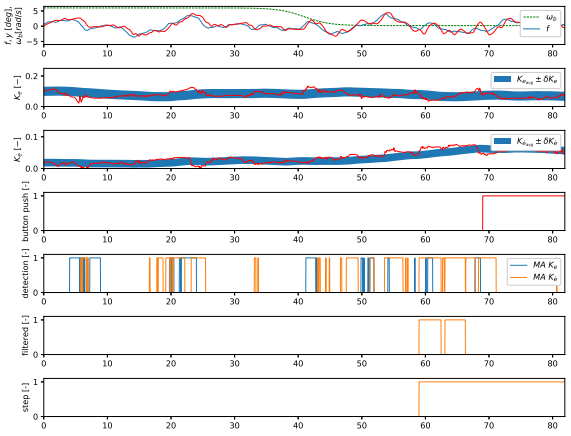


(b) MA method

Figure B.40: Run-in time 10 s, run 28.



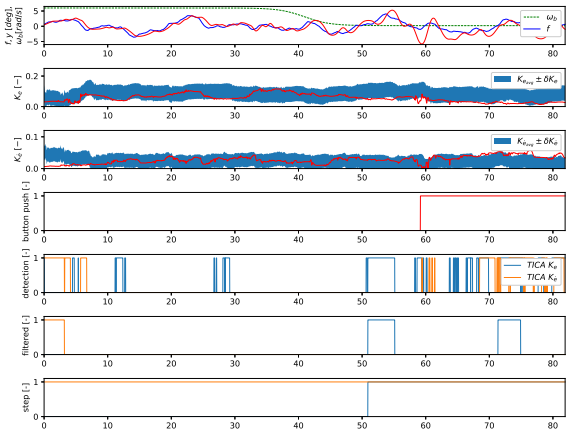
(a) TICA method



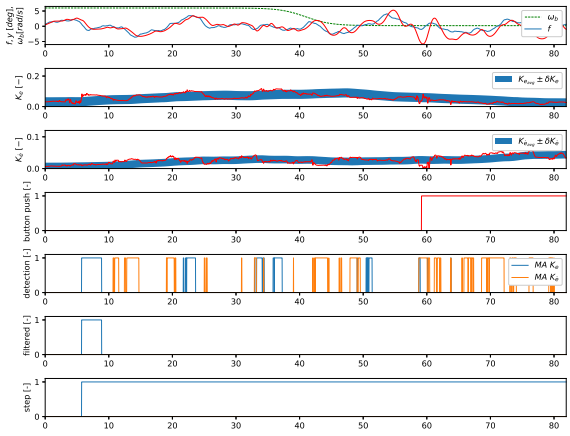
(b) MA method

Figure B.41: Run-in time 15 s, run 26.

Subject 6

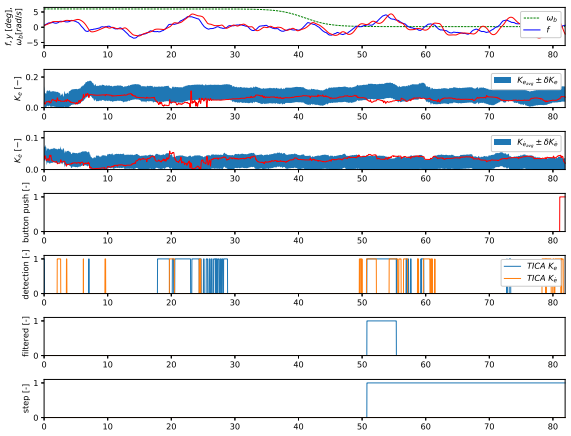


(a) TICA method

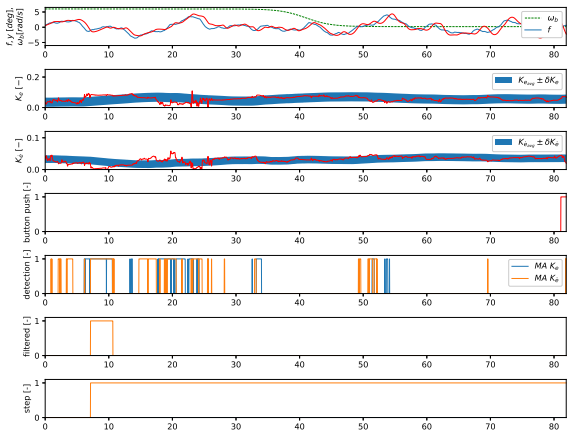


(b) MA method

Figure B.42: Run-in time 5 s, run 32.

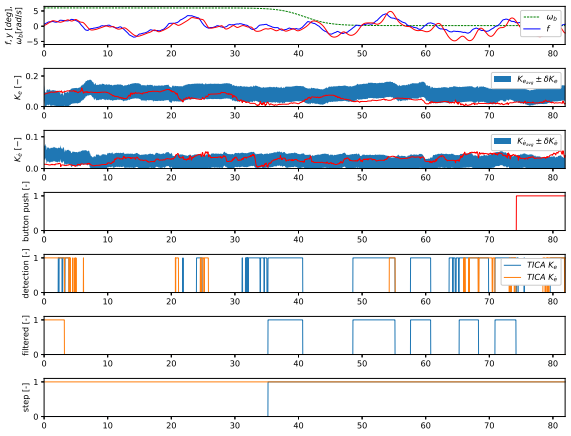


(a) TICA method

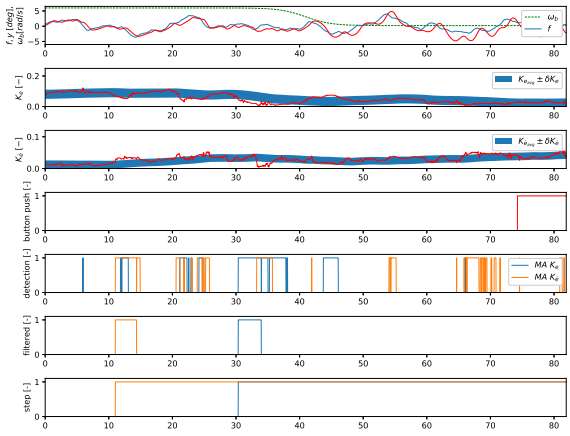


(b) MA method

Figure B.43: Run-in time 10 s, run 31.



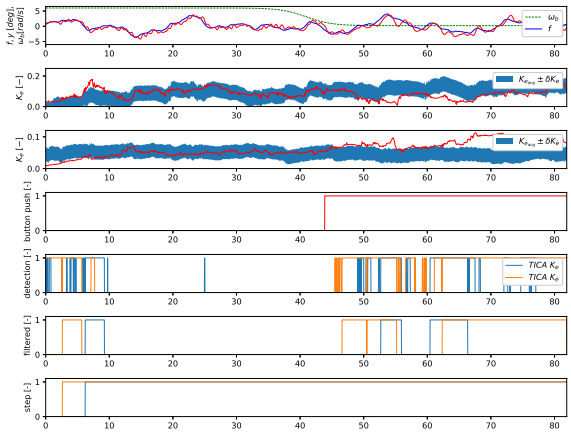
(a) TICA method



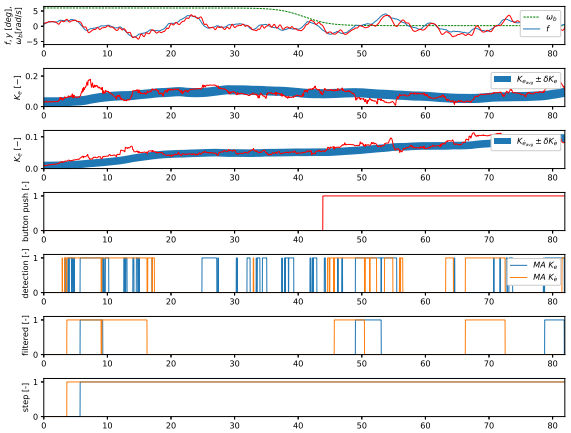
(b) MA method

Figure B.44: Run-in time 15 s, run 29.

Subject 7

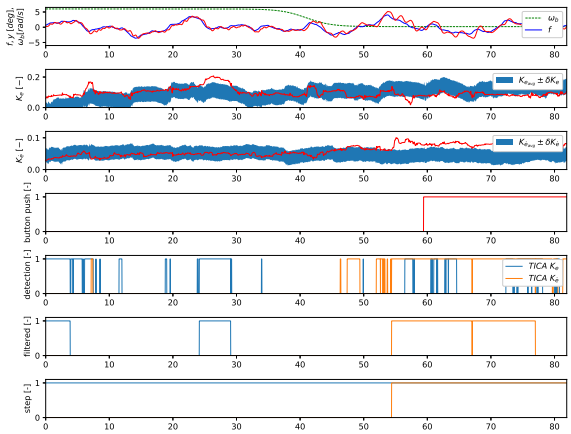


(a) TICA method

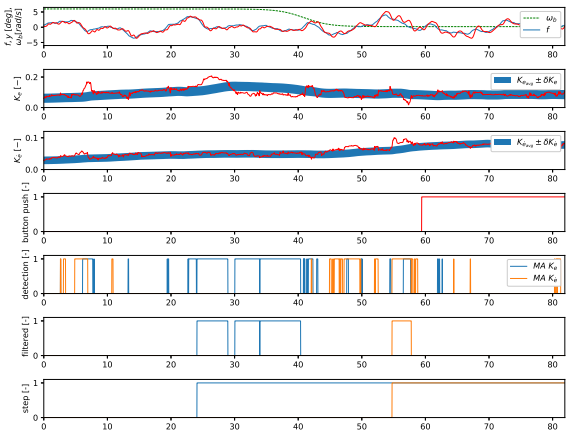


(b) MA method

Figure B.45: Run-in time 5 s, run 23.

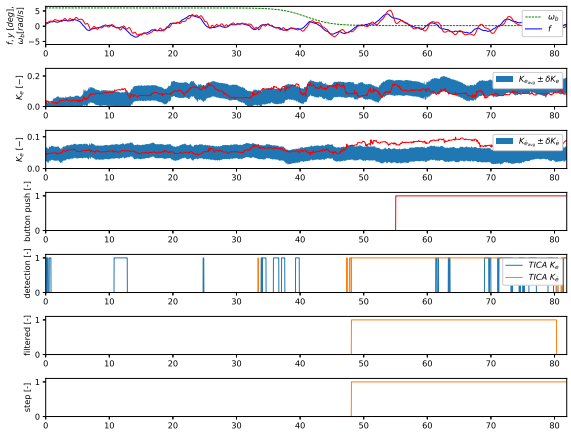


(a) TICA method

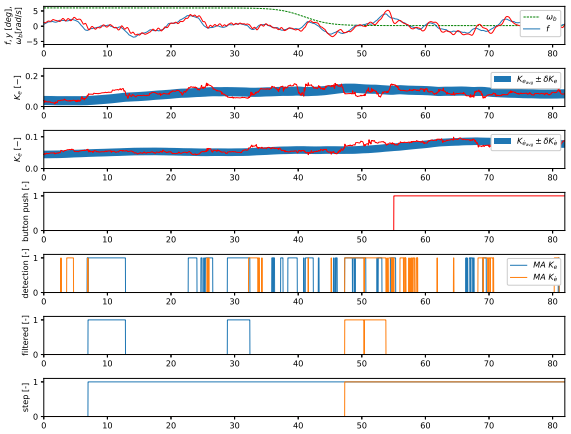


(b) MA method

Figure B.46: Run-in time 10 s, run 29.



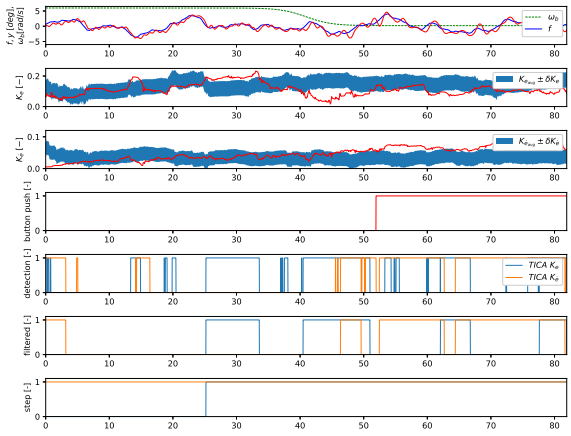
(a) TICA method



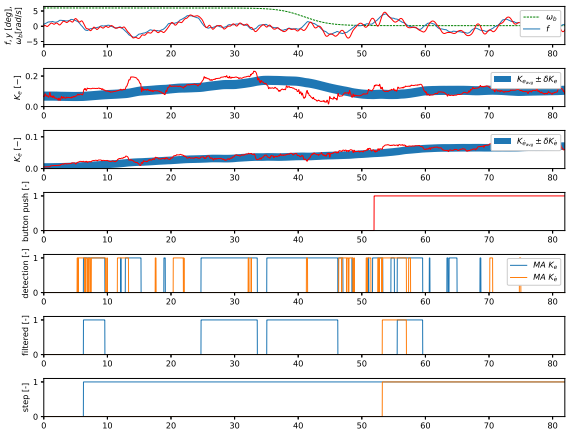
(b) MA method

Figure B.47: Run-in time 15 s, run 27.

Subject 8

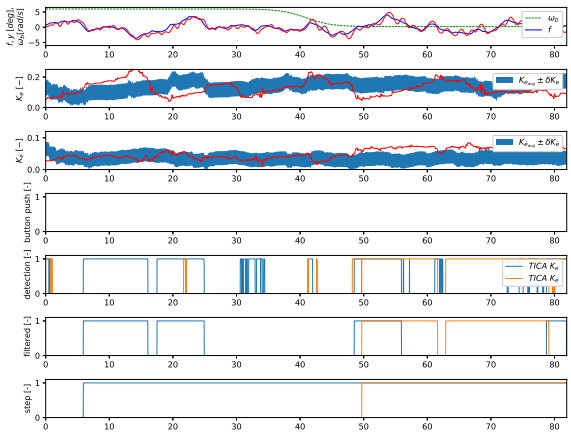


(a) TICA method

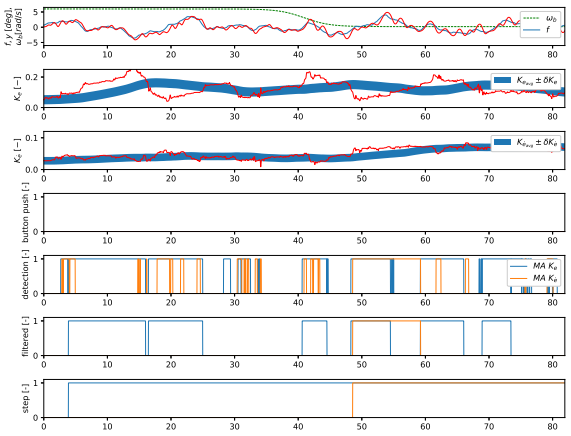


(b) MA method

Figure B.48: Run-in time 5 s, run 25.

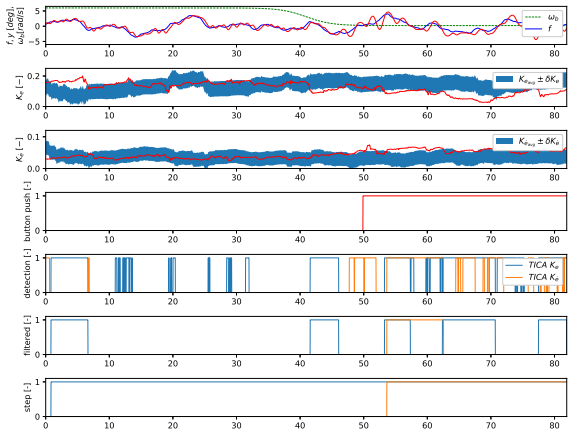


(a) TICA method

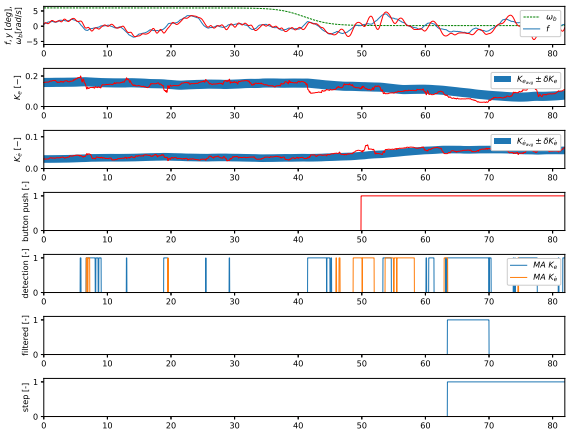


(b) MA method

Figure B.49: Run-in time 10 s, run 26.



(a) TICA method



(b) MA method

Figure B.50: Run-in time 15 s, run 29.

Bibliography

A. Ameyoe, P. Chevrel, E. Le-Carpentier, F. Mars, and H. Illy. Identification of a Linear Parameter Varying Driver Model for the Detection of Distraction. *IFAC-PapersOnLine*, 48(26):37–42, 2015. ISSN 2405-8963.

B. D.O. Anderson and J. B. Moore. Optimal filtering. *Englewood Cliffs*, 21:22–95, 1979.

S. Björklund. *A Survey and Comparison of Time-delay Estimation Methods in Linear Systems*. Division of Automatic Control, Department of Electrical Engineering, Linköpings Universitet, 2003. ISBN 9789173738705.

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. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 28(1):89–99, January 1998. doi: 10.1109/3468.650325.

F. M. Drop, D. M. Pool, M. Mulder, and H. H. Bülthoff. Constraints in Identification of Multi-Loop Feedforward Human Control Models. In *Proceedings of the 13th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Kyoto, Japan*, 2016.

R. F. M. Duarte, D. M. Pool, M. M. van Paassen, and M. Mulder. Experimental Scheduling Functions for Global LPV Human Controller Modeling. In *Proceedings of the the 20th IFAC World Congress, Toulouse, France*, volume 50 of *IFAC-PapersOnLine*, pages 15853–15858, July 2017. doi: 10.1016/j.ifacol.2017.08.2329.

R. A. Hess. Simplified Approach for Modelling Pilot Pursuit Control Behaviour in Multi-loop Flight Control Tasks. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 220 (2):85–102, 2006. ISSN 0954-4100. doi: 10.1243/09544100JAERO33.

R. A. Hess. Modeling Pilot Control Behavior with Sudden Changes in Vehicle Dynamics. *Journal of Aircraft*, 46(5):1584–1592, September-October 2009. doi: 10.2514/1.41215.

R. Hosman and H. Stassen. Pilot's Perception In the Control of Aircraft Motions. *Control Engineering Practice*, 7(11):1421–1428, 1999. ISSN 0967-0661. doi: https://doi.org/10.1016/S0967-0661(99)00111-2.

H. R. Jex, J. D. McDonnell, and A. V. Phatak. A Critical Tracking Task for Manual Control Research. *IEEE Transactions on Human Factors in Electronics*, HFE-7(4):138–145, 1966. ISSN 0096-249X. doi: 10.1109/THFE.1966.232660.

D. L. Kleinman, S. Baron, and W. H. Levison. An Optimal Control Model of Human Response Part i: Theory and Validation. *Automatica*, 6(3):357–369, 1970. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/0005-1098(70)90051-8.

W. H. Levison, S. Baron, and D. L. Kleinman. A Model for Human Controller Remnant. *IEEE Transactions on Man-Machine Systems*, 10(4):101–108, 1969. ISSN 0536-1540. doi: 10.1109/TMMS.1969.299906.

L. Ljung. *System Identification: Theory for the User, 2nd Edition*. 1999. ISBN 1070-9932. doi: 10.1109/MRA.2012.2192817.

M. M. Lone and A. K. Cooke. Review of Pilot Models Used in Aircraft Flight Dynamics. *Aerospace Science and Technology*, 2014. doi: 10.1016/j.ast.2014.02.003.

T. Mandal and Y. Gu. *Online Pilot Model Parameter Estimation Using Sub-Scale Aircraft Flight Data*. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2016. doi: doi:10.2514/6.2016-0636.

D. T. McRuer and H. R. Jex. A Review of Quasi-Linear Pilot Models. *IEEE Transactions on Human Factors in Electronics*, HFE-8(3):231–249, September 1967. ISSN 0096-249X.

M. Mulder, P. Zaal, D. M. Pool, H. J. Damveld, and M. van Paassen. *A Cybernetic Approach to Assess Simulator Fidelity: Looking back and looking forward*. Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2013. doi: doi:10.2514/6.2013-5225.

M. Mulder, D. M. Pool, D. A. Abbink, E. R. Boer, and M. M. van Paassen. Fundamental Issues in Manual Control Cybernetics. *IFAC-PapersOnLine*, 49(19):1–6, 2016. ISSN 2405-8963. doi: http://dx.doi.org/10.1016/j.ifacol.2016.10.429.

F. M. Nieuwenhuizen, P. M. T. Zaal, M. Mulder, M. M. van Paassen, and J. A. Mulder. Modeling Human Multichannel Perception and Control Using Linear Time-Invariant Models. *Journal of Guidance, Control, and Dynamics*, 31(4):999–1013, July-August 2008. doi: 10.2514/1.32307.

M. Olivari, F. M. Nieuwenhuizen, H. H. Bülthoff, and L. Pollini. Identifying Time-Varying Neuromuscular System with a Recursive Least-Squares Algorithm: a Monte-Carlo Simulation Study. In *Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics, San Diego (CA)*, pages 3573–3578, October 2014. doi: 10.1109/SMC.2014.6974484.

M. Olivari, J. Venrooij, F. M. Nieuwenhuizen, L. Pollini, and H. Bülthoff. *Identifying Time-Varying Pilot Responses: a Regularized Recursive Least-Squares Algorithm.* AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2016. doi: doi:10.2514/6.2016-1182.

A. Popovici, P. M. T. Zaal, and D. M. Pool. Dual Extended Kalman Filter for the Identification of Time-Varying Human Manual Control Behavior. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Denver (CO)*, number AIAA-2017-3666, 2017. doi: 10.2514/6.2017-3666.

N. Roggenkämper, D. M. Pool, F. M. Drop, M. M. van Paassen, and M. Mulder. Objective ARX Model Order Selection for Multi-Channel Human Operator Identification. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Washington, D.C.*, number AIAA-2016-4299, June 2016. doi: 10.2514/6.2016-4299.

J. Rojer, D.M. Pool, M.M. Van Paassen, and M. Mulder. Towards Understanding Human Adaptation: Time-Varying Parameter Estimation Utilizing the Unscented Kalman Filter. 2016.

Y. Tan. Time-Varying Time-Delay Estimation For Nonlinear Systems Using Neural Networks. *Int. J. Appl. Math. Comput. Sci.*, 14(1):63–68, 2004.

H.A. van Grootheest. Identification of Time-Varying Manual-Control Adaptation with Recursive ARX Models. 2017.

M. M. van Paassen and M. Mulder. Identification of Human Operator Control Behaviour in Multiple-Loop Tracking Tasks. *Seventh IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems*, 1998.

J. W. van Wingerden and M. Verhaegen. Subspace Identification of Bilinear and LPV Systems for Open- and Closed-loop Data. *Automatica*, 45(2):372–381, 2009. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2008.08.015.

P. M. Zaal, D. M. Pool, Q. P. Chu, M. Mulder, M. M. Van Paassen, and J. A. Mulder. Modeling Human Multimodal Perception and Control Using Genetic Maximum Likelihood Estimation. *Journal of Guidance, Control, and Dynamics*, 32(4):1089–1099, 2009. ISSN 0731-5090. doi: 10.2514/1.42843.

P. M. T. Zaal. Manual Control Adaptation to Changing Vehicle Dynamics in Roll-Pitch Control Tasks. *Journal of Guidance, Control, and Dynamics*, 39(5):1046–1058, 2016. doi: 10.2514/1.G001592.

P. M. T. Zaal and D. M. Pool. Multimodal Pilot Behavior in Multi-Axis Tracking Tasks with Time-Varying Motion Cueing Gains. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference, National Harbor (MD)*, number AIAA-2014-0810, January 2014. doi: 10.2514/6.2014-0810.

P. M. T. Zaal, F. M. Nieuwenhuizen, M. M. van Paassen, and M. Mulder. Modeling Human Control of Self-Motion Direction with Optic Flow and Vestibular motion. *IEEE Transactions on Cybernetics*, 43(2):544–556, 2013. ISSN 2168-2267. doi: 10.1109/TSMCB.2012.2212188.

Y. Zhao, A. Fatehi, and B. Huang. A Data-Driven Hybrid ARX and Markov Chain Modeling Approach to Process Identification with Time-Varying Time Delays. *IEEE Transactions on Industrial Electronics*, 64(5):4226–4236, 2017. ISSN 0278-0046. doi: 10.1109/TIE.2016.2597764.