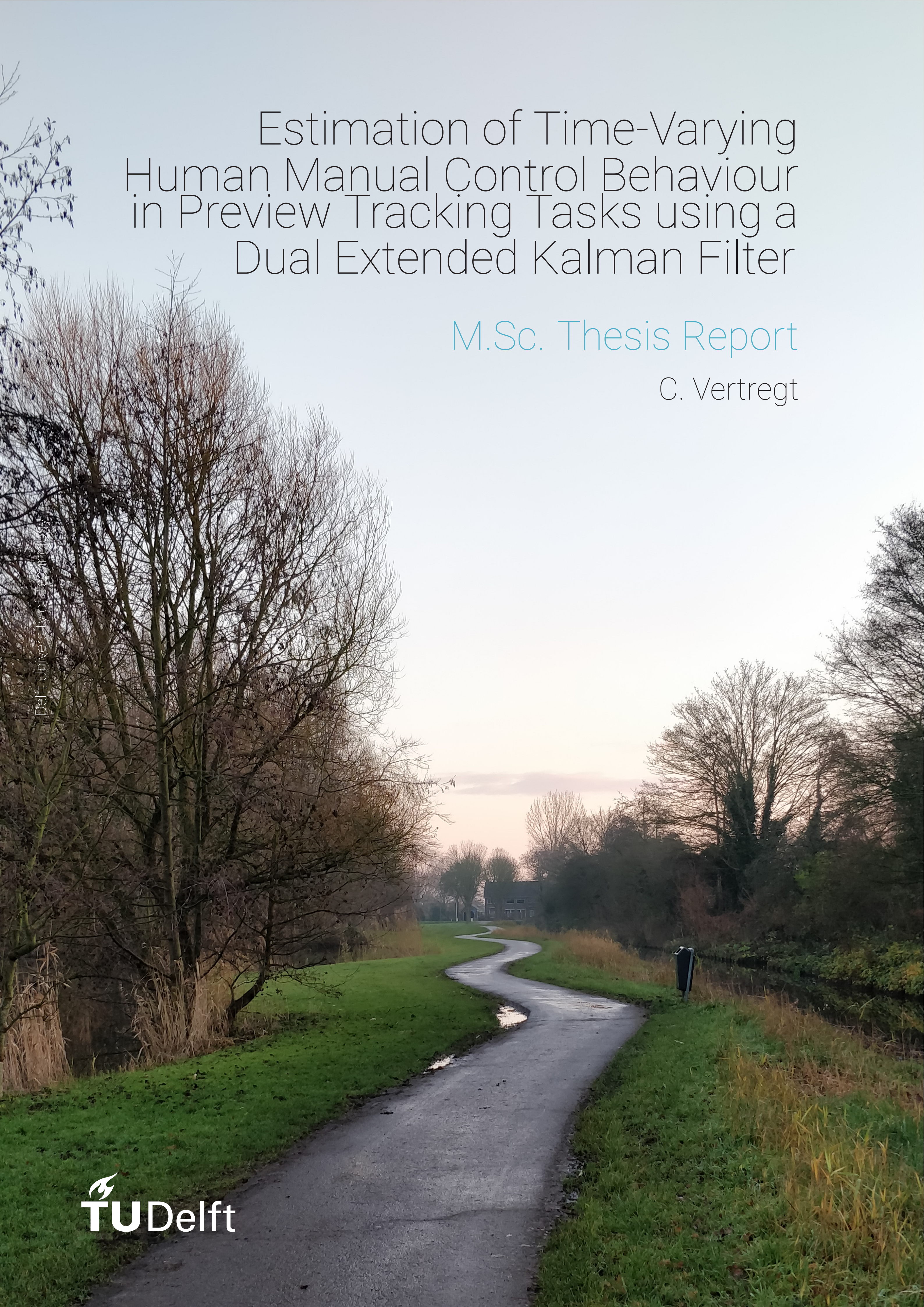


Estimation of Time-Varying Human Manual Control Behaviour in Preview Tracking Tasks using a Dual Extended Kalman Filter

M.Sc. Thesis Report

C. Vertregt

Estimation of Time-Varying Human Manual Control Behaviour in Preview Tracking Tasks using a Dual Extended Kalman Filter

Master of Science Thesis Report

by

C. Vertregt

to obtain the degree of Master of Science at

Faculty of Aerospace Engineering · Delft University of Technology

Supervised by:

dr. ir. D.M. Pool
prof. dr. ir. M. Mulder
dr. ir. K. van der El

2 June 2020

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

The undersigned hereby certify that they have read, and recommend to the Faculty of Aerospace Engineering at Delft University of Technology for acceptance, a thesis entitled **"Time-Varying Identification of Manual Control Behaviour in Preview Tracking Tasks using Kalman Filters"**, submitted by **C. Vertregt**, in partial fulfillment of the requirements for awarding the degree of **Master of Science**.

Dated: ____________________

Delft, The Netherlands

Assessment Committee:

Professor and Chair:	________________________________
	prof. dr. ir. M. Mulder
Supervisors:	________________________________
	dr. ir. D.M. Pool
	________________________________
	dr. ir. K. van der El
External Examiner:	________________________________
	dr. M.A. Mitici

Preface

This report covers the complete thesis research that was performed in order to obtain the degree of Master of Science in Aerospace Engineering at Delft University of Technology. The research tests the application of a dual extended Kalman filter for the purpose of time-varying estimation of human manual control behaviour in preview tracking tasks. The report is divided in three parts. Part I contains a scientific article that summarizes the research methodology as well as the most notable results. Part II follows up with the preliminary thesis report that was graded for the AE4020 Literature Study. This part includes more background information about previous work in literature and the research approach of this thesis. Finally, Part III consists of appendices to the final report, with a collection of the complete results that formed the basis of the research conclusions, detailed documentation of the code that was used to generate these results and some explorations for further research.

There is a number of people that I would like to thank for supporting and motivating me during this thesis. Daan, your personal and engaging style of supervision gave me the guidance and motivation to deliver my best performance and overcome the difficult moments along the way. Kasper and Max, thank you for your constructive discussions on the content of this thesis, as well as for your invaluable personal advice.

I would also like to thank my fellow students from the Sim 0.08 graduate student room. Sharing pain and pleasure during our theses has made the whole experience infinitely more pleasant. Moreover I would like to thank my parents, sisters and family for their continued support during this research and throughout the years before. And of course, thank you Judith for being my work-from-home colleague, company and sports buddy during this last period of *intelligent lockdown*.

This thesis also concludes the time that I have been proudly calling myself student at Delft University of Technology. The university and its students have offered me a tremendous amount of learning experiences, friendships and memories that I look back on with great pleasure. During these years (all the more during exam periods), the examples set by scientists, engineers and enthusiasts that go out and test the world with a curious and exploratory mindset boosted my interest for the subject matter. A great example is the *Backwards Brain Bicycle*-video from the YouTube channel *Smarter Every Day*, made by American rocket test engineer Destin Sandlin. The deeper insight that he gained from the experience is something that seamlessly fits a practical, engineer's approach to comprehending the world around us:

"Knowledge does not equal understanding."

— Destin Sandlin (Smarter Every Day)

Combining this lesson with the knowledge that I have gained over the past years, I feel prepared to move on from university in pursuit of better understanding of the curious, beautiful and fascinating world around us.

Casper Vertregt
Delft, June 2020

Cover: Vrederustpad, Delft (© C. Vertregt)

Contents

List of Figures

List of Tables

List of Symbols

Roman Symbols

Symbol	Description	Unit
A	State matrix	[−]
B	Input matrix	[−]
C	Output matrix	[−]
D	Feedthrough matrix	[−]
$e(t)$	Error signal, $f_t - y$	[m]
f	Non-linear state function of the state filter	[-]
$f_d(t)$	Disturbance function	[m]
$f_t(t)$	Forcing function	[m]
$f^{\star}_{t,f}(t)$	Far view filtered forcing function signal	[m]
g	Non-linear output function of the DEKF	[-]
H	Transfer function	[−]
H_{CE}	CE transfer function	[−]
H_{HO}	HO transfer function	[−]
H_{NM}	HO neuromuscular transfer function	[−]
$H_{O_{e^\star}}$	HO preview equalisation transfer function	[−]
H_{O_f}	HO far view transfer function	[−]
j	Imaginary number	[−]
K	Gain	[−]
k	Time step index	[−]
K_f	HO far view gain	[−]
K_n	HO near view gain	[−]
K_p	HO equalisation gain	[−]
K_r	Remnant gain	[−]
K_v	HO equalisation lead gain	[−]
$n(t)$	Remnant noise signal	[−]
P_p	Covariance matrix of parameter estimation error	[−]
P_r	Remnant power ratio	[−]
P_s	Covariance matrix of state estimation error	[−]
Q_p	Parameter process noise covariance matrix	[−]
Q_s	State process noise covariance matrix	[−]
R	Sensor noise covariance matrix	[−]
s	Laplace variable	[−]
t	Time	[s]
T_l	Lag time constant	[s]
$u(t)$	HO input	[−]

u_k	HO input at this time step (k)	[−]
w	White noise	[−]
$\boldsymbol{x_s}$	State vector	[−]
$x^+_{s,k-1}$	State vector correction, previous timestep	[−]
$x^+_{s,k}$	State vector correction, this timestep	[−]
$x^-_{s,k-1}$	State vector prediction, previous timestep	[−]
$x^-_{s,k}$	State vector prediction, this timestep	[−]
$x_{s,i}$	Canonical state i	[−]
$y(t)$	CE output	[m]

Greek Symbols

Symbol	Description	Unit
Δt	Time step	[s]
ζ	Damping ratio	[−]
ζ_n	Neuromuscular damping ratio	[−]
$\boldsymbol{\theta}$	Parameter vector	[−]
θ^+_{k-1}	Parameter vector correction, previous timestep	[−]
θ^+_k	Parameter vector correction, this timestep	[−]
θ^-_{k-1}	Parameter vector prediction, previous timestep	[−]
θ^-_k	Parameter vector prediction, this timestep	[−]
σ	Standard deviation	[−]
σ_r	Remnant signal standard deviation	[−]
σ_u	HO input standard deviation	[−]
$\tau^\star$	Apparent preview time delay	[s]
τ_f	Far view preview time	[s]
τ_n	Near view preview time	[s]
τ_s	Suspension time	[s]
τ_v	HO visual time delay	[s]
ω	Frequency	[rad/s]
ω_b	Break frequency	[rad/s]
ω_n	Neuromuscular frequency	[rad/s]

Subscripts

Symbol	Description	Unit
e	Error signal *or* HO equalisation	[−]
f	Far view	[−]
k	Time step index	[−]
n	Near view *or* neuromuscular (*or* remnant, only in scientific article)	[−]
p	Parameter filter	[−]
r	Remnant	[−]
s	State filter	[−]
u	HO input signal	[−]

List of Abbreviations

ARX Auto-Regressive models with eXogenous input
CE Controlled Element
DEKF Dual Extended Kalman Filter
DI Double Integrator
EKF Extended Kalman Filter
FoFu Forcing Function
HO Human Operator
KF Kalman Filter
LPV Linear Parameter Varying
MIMO Multiple Input, Multiple Output system
MISO Multiple Input, Single Output system
MLE Maximum Likelihood Estimation
NRMSE Normalised Root Mean Square Error
PBSID Predictor-Based Subspace Identification
RDI Recursive Delay Identifier
RMSE Root Mean Square Error
SI Single Integrator
SISO Single Input, Single Output system
TF Transfer Function
UKF Unscented Kalman Filter
VAF Variance Accounted For

I
Scientific Article

To be graded for AE5310 Final Thesis

Estimation of Time-Varying Human Manual Control Behaviour in Preview Tracking Tasks Using a Dual Extended Kalman Filter

Casper Vertregt, *Author* Daan M. Pool, Kasper van der El and Max Mulder, *Supervisors*

***Abstract*—This research applied a Dual Extended Kalman Filter (DEKF) to time-varying human operator (HO) parameter estimation in preview tracking tasks. The preview time parameter was of particular interest, as the amount of preview can be highly variable in practical situations. The filter was centered around a linear cybernetic HO model with six identifiable parameters for single-integrator and eight for double-integrator control tasks. The DEKF was applied to a range of time-varying simulations, both remnant-free as well as with realistic remnant based on an experimentally identified first-order model. In addition, the tests were validated using existing experimental tracking data. By keeping the HO physical limitation parameters constant (i.e. neuromuscular parameters ω_{nms} and ζ_{nms} and time delay τ_v), good estimation results were obtained, particularly for tracking of SI systems. General guidelines for the sensitive tuning process of the filter are proposed. The presence of coloured remnant noise in double-integrator tracking was found to greatly affect the quality of the estimates. Future versions of the filter should explicitly include models of remnant noise, to reduce its sensitivity and ease the tuning process. The obtained algorithm is applicable to single sets of measurement data without *a priori* assumptions on time-variance or the need for averaging. With some additional development, this makes the DEKF a suitable candidate for practical applications, such as driver monitoring and advanced driver assistance systems in the automotive industry.**

***Index Terms*—Manual control, parameter estimation, preview time, Dual Extended Kalman Filter (DEKF).**

I. Introduction

HUMAN controllers are known to adapt continuously to variable circumstances [1]. Factors triggering these conscious and subconscious behavioural changes could for example be changing controlled element (CE) dynamics, environmental conditions or variable mental and physical state of the operator [2]. Understanding this time-varying behaviour is one of the major challenges within the field of cybernetics [3]. Knowledge of how humans use time-variance in their control of systems is, for example, applicable in the development of Advanced Driver Assistance Systems (ADAS) [4]. Haptic feedback systems could support the changing behaviour that the driver exhibits, while the state and performance of the driver is monitored real-time [5].

For such a system to work, robust and accurate time-varying estimation of the driver's response dynamics is required. Driving is a common example of a preview control task [6]. In a preview control task, a human operator (HO) steers a controlled element to follow a predefined target signal as closely as possible. Through different senses, humans can not only observe the current CE state, but they also enable us to anticipate events some time in the future [7]. In the case of driving, the target (the road) and the sources of preview information (primarily vision) are clearly identifiable.

The authors are with the Section Control & Simulation, Department of Control & Operations, Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands (e-mail: c.vertregt@student.tudelft.nl; {d.m.pool, k.vanderel, m.mulder}@tudelft.nl)

To build up to a system that can contribute to ADAS or haptic feedback, it is essential to include the human response to preview. The amount of preview information that is available to or used by human controllers is typically expressed in terms of time, distance or gaze angle relative to a point on the target ahead [8]. In this research, preview time will be consistently used to express preview. Available preview is often restricted, due to for example obstacles, terrain, weather, lighting or other environmental conditions [9]. Research has shown that limited preview influences the behaviour of human controllers [9]. Since the sources of preview limitations are inherently time-varying, the operator's behaviour should be identified varying over time as well.

Classical system identification for human operators is usually performed in the frequency domain [2], [10]. There have been studies that successfully incorporate time-varying behaviour in a Linear Parameter Varying (LPV) model as function of grip force, by applying frequency domain identification to local linear time-invariant (LTI) models [11]. For true time-varying estimation, however, without the need for experiments correlating external measurements to parameter variations, frequency domain methods are not directly suited.

Predictor-Based Subspace Identification was applied for parameter estimation of time-varying LPV-models in single axis compensatory tracking tasks [12]. Also, a method based on Maximum Likelihood Estimation (MLE) has been successfully applied in a dual-axis control experiment with commercial pilots [13]. Both of these methods were successful, but do require prescribed scheduling functions for the type and shape of time-variance.

Several techniques that require no *a priori* assumptions on or measurements of parameter variance have been tested for human control as well. Zaal and Sweet applied a two-step method based on wavelets transforms [14]. Although this method showed quick convergence for idealised tracking behaviour without remnant, convergence declined for simulations with significant remnant power. Estimation using auto-regressive models with exogenous inputs (ARX-models) was shown to be more robust to remnant noise, both on simulated as well as experimental tracking data. The method was reasonably successful in the detection of operator adaptation in a compensatory tracking experiment [15]. However, ARX estimation has so far been unable to incorporate variations in time shifts and delay parameters, which form an essential part of preview tracking [15].

Finally, Kalman filters have been a prevalent choice for HO parameter estimation [16]–[21]. There are generally two options for simultaneous state and parameter estimation using Kalman filters. In *joint* estimation, parameters are augmented to the state vector. The resulting, typically nonlinear system is then filtered with a nonlinear Kalman filter adaptation. The most elemental of these, the extended Kalman filter (EKF), worked for computer generated control signals, but diverged when applied to experimentally measured human control data [16]. The addition of a recursive delay identifier with bicubic interpolation, together with a parameter smoother, did improve the quality of concurrent identification of linear parameters and delay time [17]. Another type of Kalman filter, the Unscented Kalman Filter (UKF) has been applied successfully to compensatory tracking tasks [18] and pilot control inputs of simulated flights [19].

The second option is *dual* estimation [22]. The state and parameter vectors are split across separate filters, that run concurrent to each other. Splitting the filter in two concurrent elements increases its tracking robustness and speed, which in turn improves the convergence of the filter [22]. A Dual Extended Kalman Filter (DEKF) was applied to both simulated and experimental compensatory HO tracking data with notable results [20]. The algorithm was remarkably accurate for different types of time-variance. Moreover, it allowed for the inclusion of a time delay by means of a Padé approximation, making it a suitable candidate for time-varying parameter estimation in preview tracking tasks. A comparative study between dual and joint versions of the extended and unscented Kalman filter in driving field tests showed similar promising results [21].

This research aims to estimate time-varying HO parameters in preview tracking tasks, with particular focus on the preview time. Dual estimation based on extended Kalman filters is used to achieve this. Four questions are central throughout this research:

1) Can time-varying HO parameters in preview control tasks be estimated with a Dual Extended Kalman Filter?
2) How should the DEKF be initialised and tuned to achieve good estimation performance?
3) How do differences in HO behaviour when tracking single or double integrator CE dynamics affect the DEKF's estimation performance?
4) What are the effects of remnant noise on the estimation performance of the DEKF?

The paper is structured as follows. Section II gives a general description of preview tracking tasks and the model that is used to describe human operator behaviour. Section III follows up with the details of the DEKF as used in this research. The main analysis of filter performance is done using computer simulations, that allow for controlled testing with HO realism by means of remnant noise and time-variance. Section IV explains how these simulations were constructed and the methodology behind the evaluation order. Section V then shows the most important results. In Section VI the DEKF is applied to actual experimental data for validation. Finally, a discussion of results and the main conclusions can be found in Sections VII and VIII, respectively.

II. Preview Tracking Task

A realistic preview control task, such as car driving, introduces a large number of variables simultaneously. Previous research has shown that, for example, road layout and perspective [23]–[25], multi-loop feedback signals [26] and vestibular motion cues [27] all have an impact on human control behaviour. This makes it difficult to apply a novel algorithm under controlled conditions. Instead, the DEKF is initially applied to human control behaviour in preview *tracking tasks* [28]. A diagram of a preview tracking task is depicted in Fig. 1. In its simplest form, the target signal f_t and current system state y are presented to the HO visually by means of symbols on a screen, as shown in Fig. 1. Preview information is made available with a line, indicating the target signal for some time in the future. The controlled element in a tracking task commonly has simplified dynamics, such as a single-integrator (SI) or double-integrator (DI) [2].

A model for human control in preview tracking tasks was experimentally identified by Van der El *et al.* [10]. This cybernetic preview model is particularly suited for time-varying parameter estimation because of its parameters, which are physically interpretable. This means that their nominal values and limits in normal human behaviour have been tested extensively. The investigated experimental conditions include the effects of system dynamics [29], forcing function bandwidth [30], available preview information [9] and gaze tracking with spacial occlusions [31]. Moreover, the model also been tested for the transition from an abstract preview tracking task towards realistic more driving conditions, by means of experiments with linear perspective [25], vestibular feedback [27] and multi-loop feedback in realistic driving scenarios [26].

A. Quasi-Linear Operator Model

The quasi-linear, cybernetic model for human operators in preview tracking tasks was formulated as an extension to the established crossover model for compensatory tracking [2]. Instead of tracking the true error between the forcing function and the current system state, compensatory tracking is performed based on an *internally calculated error* $e^{\star}(t)$, see Fig. 1. For the simulations and estimation, the simplest version of the original model is used is this research, without the near viewpoint response [10]. As originally noted, the use of near view is restricted to higher frequencies, not all operators show this type of behaviour and its contributions to the overall model are small [10].

The HO model, shown in Fig. 1 consists of four elements [10]:

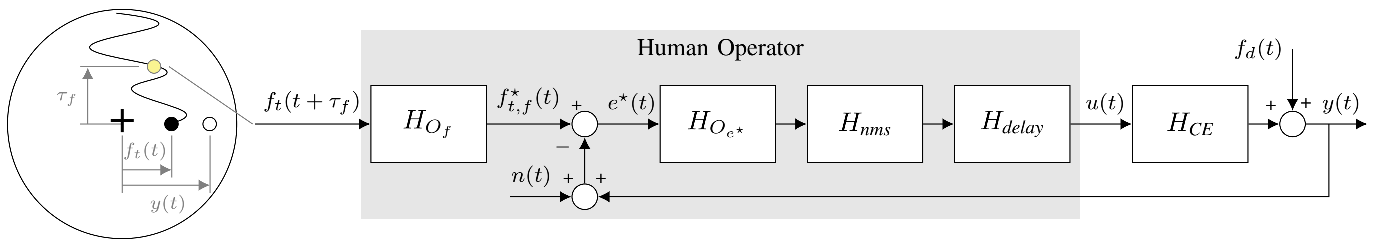


Fig. 1. Control diagram of a preview tracking task [10]. The yellow circle indicates the viewpoint that the HO uses for control (not visible during the task)

1) Preview filter: The previewed target signal is low-pass filtered by the HO, since the highest frequency oscillations cannot be accurately followed. This filter, see (1), is characterised by preview gain K_f and time constant $T_{l,f}$. For the purpose of this research, the time constant is rewritten as a break frequency $\omega_{b,f} = 1/T_{l,f}$. Subtracting the filtered target $f^{\star}_{t,f}(t)$ from the current output $y(t)$ results in the aforementioned $e^{\star}(t)$.

$$H_{O_f}(j\omega) = \frac{K_f}{1+T_{l,f}j\omega} = K_f\frac{\omega_{b,f}}{\omega_{b,f}+j\omega} \quad (1)$$

For SI controlled element dynamics, $\omega_{b,f}$ is kept at a constant value beyond the forcing function bandwidth, since the target is not low-pass filtered in the case of SI control [9]. This effectively reduces the filter to a simple gain K_f.

2) HO equalization: It was experimentally demonstrated that operators adjust their dynamics such that the open loop response from $e^{\star}$ to y approximates a single integrator [10]. This finding is in accordance with the crossover model for compensatory tracking [2]. For a CE with SI dynamics, the HO will thus control with a single gain K_p. In order to control DI dynamics, the HO introduces lead control, by means of a lead time $T_{L,e}$. For the purpose of estimation, it is more convenient to write this lead control with an additional gain ($K_v = K_pT_{L,e}$), in order to reduce parameter couplings.

$$H_{O_{e^\star}}(j\omega) = K_p(1+T_{L,e}j\omega) = K_p + K_v j\omega \quad (2)$$

3) Neuromuscular limitations: Human control actions are limited by the neuromuscular system, typically modeled as a second-order system, characterised by neuromuscular frequency ω_{nms} and neuromuscular damping ζ_{nms}:

$$H_{nms}(j\omega) = \frac{\omega_{nms}^2}{(j\omega)^2 + 2\zeta_{nms}\omega_{nms}j\omega + \omega_{nms}^2} \quad (3)$$

4) Visual time delay: Finally, human controllers exhibit a processing delay in their control behaviour (τ_v). For time-varying simulations and Kalman filter estimation, it is however not possible to use the conventional exponential expression of the delay (see (4)). Instead, delays are modeled with a Padé approximation to obtain a fractional transfer function (m indicates the order of the approximation) [32]:

$$H_{delay}(j\omega) = e^{-\tau_v j\omega} \approx \frac{\sum_{j=0}^{m}\frac{(2r-j)!}{j!(r-j)!}(-\tau_v j\omega)^j}{\sum_{k=0}^{m}\frac{(2r-k)!}{k!(r-k)!}(\tau_v j\omega)^k} \quad (4)$$

B. Operator Remnant

The preview model presented above is quasi-linear, meaning that the nonlinear human control behaviour is linearised around an operating point. The unexplained behaviour that remains is known as the remnant ($n(t)$). Recently, human remnant in preview tracking tasks was experimentally measured and modeled [33]. The research demonstrated that white noise passed through a first-order low pass filter (H_n, see 5) provides adequate accuracy as a remnant model in pursuit and preview tracking. The noise source is located in the feedback loop of system output y. This first-order remnant model was originally proposed for compensatory tracking by [34], and is characterised by break frequency $\omega_{b,n}$ and remnant gain K_n:

$$H_n(j\omega) = \frac{K_n}{\omega_{b,n}+j\omega} \quad (5)$$

The remnant gain K_n is used to scale the power contribution of remnant in HO input (u_n) relative to the total signal (u), according to remnant power ratio P_n (see (6)), calculated by means of the power-spectral density function (PSD) [33].

$$P_n = \frac{\sigma_{u,n}^2}{\sigma_u^2} \quad (6)$$

In their review, Van der El *et al.* concluded that the remnant power ratio is relatively invariant to changes in preview time and display configuration [33]. It does vary as a function of CE dynamics, target bandwidth and tracking task type. Moreover, the break frequency $\omega_{b,n}$ was shown to only vary as a function of CE dynamics.

III. The Dual Extended Kalman Filter

The problem of estimating the parameters of the HO model of Fig. 1 based on measured f_t, u and y is a concurrent state and parameter estimation problem. System states as well as model parameters have to be estimated simultaneously. In this research, a Dual Extended Kalman Filter is applied. The principle of a dual filter is to divide states and parameters over two separate filters [22]. The *state filter* (indicated with subscript s throughout this article) estimates the states of the system, while the *parameter filter* (indicated with subscript p) estimates the unknown parameter values of the human operator. These filters run concurrently and use each other's results in the process. Splitting the filter in two concurrent elements improves its convergence, robustness and speed [22]. The dual estimation technique, applied to human preview control is illustrated in Fig. 2.

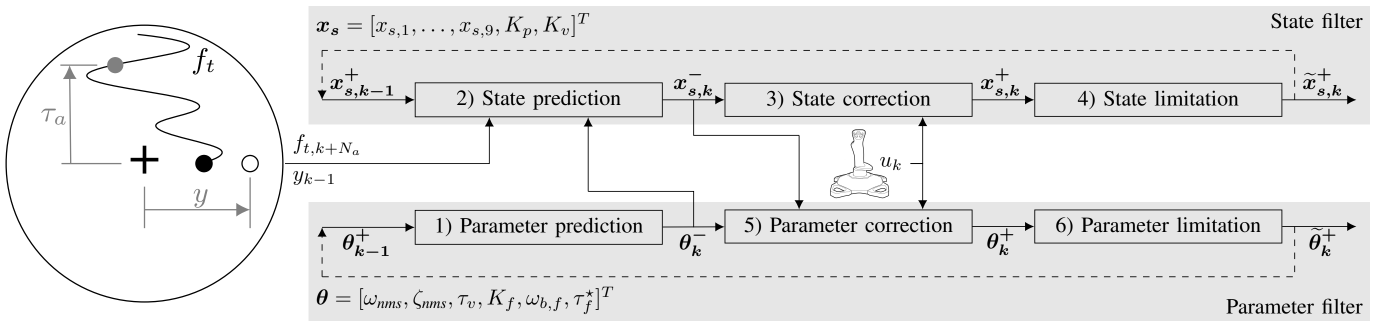

$\boldsymbol{x_s} = [x_{s,1}, \ldots, x_{s,9}, K_p, K_v]^T$
State filter
$\boldsymbol{x^+_{s,k-1}}$
2) State prediction
$\boldsymbol{x^-_{s,k}}$
3) State correction
$\boldsymbol{x^+_{s,k}}$
4) State limitation
$\boldsymbol{\widetilde{x}^+_{s,k}}$
f_t
τ_a
y
$f_{t,k+N_a}$
y_{k-1}
u_k
$\boldsymbol{\theta^+_{k-1}}$
1) Parameter prediction
$\boldsymbol{\theta^-_k}$
5) Parameter correction
$\boldsymbol{\theta^+_k}$
6) Parameter limitation
$\boldsymbol{\widetilde{\theta}^+_k}$
$\boldsymbol{\theta} = [\omega_{nms}, \zeta_{nms}, \tau_v, K_f, \omega_{b,f}, \tau^\star_f]^T$
Parameter filter

Fig. 2. The Dual Extended Kalman Filter in preview tracking tasks: state and parameter filter with limitations (after [20])

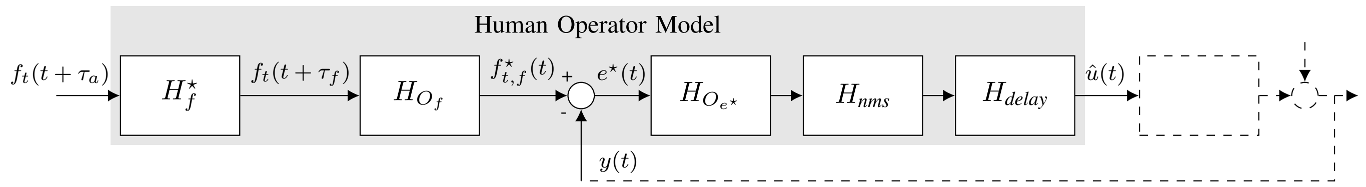


Fig. 3. Isolated HO model

A. Filter Implementation

The isolated human operator model that forms the basis of the Kalman filter estimation, is depicted in Fig. 3. Two input signals feed into the model, the target signal f_t and the feedback of CE output, y. The model's only output is the (modeled) HO stick input $\hat{u}$. Here, these signals are all considered as discrete measurements, sampled at 100 Hz. Note from Fig. 3 that the model starts with a delay ($H^\star_f$) that transforms the anticipated target, shifted τ_a seconds ahead, into the previewed target, τ_f seconds ahead, through an identifiable (positive) delay. This step is explained in Section III-B.

For use within the DEKF, the HO model of Fig. 3 is transformed into a state-space form. This requires both delay blocks $H^\star_f$ and H_{delay} to be expressed as a fractional transfer function by means of a Padé approximation, (4). The result is a non-linear state-space expression for the state filter:

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}}(t) = f(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t), f_t(t+\tau_a), y(t)) + \boldsymbol{w_s}(t) \tag{7}$$
$$\hat{u}(t) = g(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t)) + v(t) \tag{8}$$

with state vector $\boldsymbol{x_s} = [x_{s,1}, \ldots, x_{s,9}, K_p, K_v]^T$ consisting of canonical states, augmented with the HO equalisation gains. The number of canonical states depends on the Padé order for $\tau^\star_f$ and τ_v. Third-order approximations are used for both delays, leading to nine canonical states. For the delays considered (0.1 up to 1 s), this order is the minimum that results in close approximation for the frequency range of the signals involved. The full nonlinear equations f and g are listed in App. A. The equalisation gains are placed in the state filter, since they only appear in the output equation g (see App. A) [20]. $\boldsymbol{\theta}(t)$ is the parameter vector, estimated by the parameter filter. The zero mean Gaussian process and measurement noises are indicated by $w_s(t)$ (with 11×11 covariance matrix Q_s) and $v(t)$ (with variance R), respectively.

The parameter filter state-space equations are given by:

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{w_p}(t) \tag{9}$$
$$\hat{u}(t) = g(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t)) + v(t) \tag{10}$$

with $\boldsymbol{\theta} = [\omega_{nms}, \zeta_{nms}, \tau_v, K_f, \omega_{b,f}, \tau^\star_f]^T$. Similar to the compensatory case, the dynamics of the parameters are modeled as a "random walk", driven by zero mean white process noise $\boldsymbol{w_p}$, with a 6×6 covariance matrix Q_p [20].

The discrete-time filter equations of the state and parameter filters are divided in six steps (see Fig. 2). The prediction and correction steps are exactly as used in the DEKF for the compensatory case [20]. The limitation steps are added in this research, and will be explained in Section III-C. The mathematics of this six-step algorithm can be expressed as follows:

1) Parameter prediction:

$$\boldsymbol{\theta^-_k} = \boldsymbol{\theta^+_{k-1}} \tag{11}$$
$$P^-_{p,k} = \Phi_{p,k-1} P^+_{p,k-1} \Phi^T_{p,k-1} + \Gamma_{p,k-1} Q_p \Gamma^T_{p,k-1} \tag{12}$$

2) State prediction:

$$\boldsymbol{x^-_{s,k}} = \boldsymbol{x^+_{s,k-1}} + f(\boldsymbol{x^+_{s,k-1}}, \boldsymbol{\theta^-_k}, f_{t,k+N_s}, y_{k-1})\Delta t \tag{13}$$
$$P^-_{s,k} = \Phi_{s,k-1} P^+_{s,k-1} \Phi^T_{s,k-1} + \Gamma_{s,k-1} Q_{s,k} \Gamma^T_{s,k-1} \tag{14}$$

3) State correction:

$$S_{s,k} = G_{s,k} P^-_{s,k} G^T_{s,k} + R \tag{15}$$
$$r_k = u_k - g(\boldsymbol{x^-_{s,k}}, \boldsymbol{\theta^-_k}) \tag{16}$$
$$K_{s,k} = P^-_{s,k} G^T_{s,k} S^{-1}_{s,k} \tag{17}$$
$$\boldsymbol{x^+_{s,k}} = \boldsymbol{x^-_{s,k}} + K_{s,k} r_k \tag{18}$$
$$P^+_{s,k} = (I - K_{s,k} G_{s,k}) P^-_{s,k} (I - K_{s,k} G_{s,k})^T + K_{s,k} R K^T_{s,k} \tag{19}$$

4) State limitation:

$$\boldsymbol{\widetilde{x}^+_{s,k}} = \boldsymbol{x^+_{s,k}} - D^T_{s,k}(D_{s,k} D^T_{s,k})^{-1}(D_{s,k}\boldsymbol{x^+_{s,k}} - \boldsymbol{d_{s,k}}) \tag{20}$$

5) Parameter correction:

$$S_{p,k} = G^{tot}_{p,k} P^-_{p,k} (G^{tot}_{p,k})^T + R \tag{21}$$

$$K_{p,k} = P^-_{p,k} (G^{tot}_{p,k})^T S^{-1}_{p,k} \tag{22}$$

$$\boldsymbol{\theta}^+_{\boldsymbol{k}} = \boldsymbol{\theta}^-_{\boldsymbol{k}} + K_{p,k} r_k \tag{23}$$

$$P^+_{p,k} = (I - K_{p,k} G^{tot}_{p,k}) P^-_{p,k} (I - K_{p,k} G^{tot}_{p,k})^T + K_{p,k} R K^T_{p,k} \tag{24}$$

6) Parameter limitation:

$$\widetilde{\boldsymbol{\theta}}^+_{\boldsymbol{k}} = \boldsymbol{\theta}^+_{\boldsymbol{k}} - D^T_{p,k}(D_{p,k} D^T_{p,k})^{-1}(D_{p,k}\boldsymbol{\theta}^+_{\boldsymbol{k}} - \boldsymbol{d_{p,k}}) \tag{25}$$

The following symbols appear in these equations:

k, the discrete time step

Δt, the time step

$\Phi_{s,k-1}, \Phi_{p,k-1}$, discrete transition matrices, with

$$\Phi_{s,k-1} = \left.\frac{\partial f(\boldsymbol{x_s}, \boldsymbol{\theta}^-_{\boldsymbol{k}}, f_{t,k+N_s}, y_{k-1})}{\partial \boldsymbol{x_s}}\right|_{\boldsymbol{x_s}=\boldsymbol{x}^+_{\boldsymbol{s,k-1}}}$$

$\Gamma_{\{s,p\},k-1}$, discrete input identity matrices

$$G_{s,k} = \left.\frac{\partial g(\boldsymbol{x_s}, \boldsymbol{\theta}^-_{\boldsymbol{k}})}{\partial \boldsymbol{x_s}}\right|_{\boldsymbol{x_s}=\boldsymbol{x}^-_{\boldsymbol{s,k}}}$$

$$G^{tot}_{p,k} = \left.\frac{dg(\boldsymbol{x}^-_{\boldsymbol{s,k}}, \boldsymbol{\theta})}{d\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^-_{\boldsymbol{k}}}$$

$P_{\{s,p\},k}$, the state and parameter covariance matrices

$Q_{\{s,p\}}$, the process noise covariance matrices

$K_{\{s,p\},k}$, the Kalman gains

r_k, the filter innovation

$D_{\{s,p\},k}, \boldsymbol{d}_{\{s,p\},k}$, constraint matrices (see Section III-C)

Not all parameters of parameter vector $\boldsymbol{\theta}$ appear explicitly in the output equation, g. Therefore, the calculation of total derivative of g with respect to the parameter vector, indicated by $G^{tot}_{p,k}$, is necessary [20]. The steps required to calculate this derivative are given in App. A.

To reduce the problem dimension and improve estimation performance, parameters can be kept constant during estimation. In such case, the parameter is removed from either $\boldsymbol{x_s}$ or $\boldsymbol{\theta}$. Furthermore, symbolic occurrences in state and output functions f and g are replaced by the *a priori* assumed numerical value. As a result, the parameter is also removed from any Jacobians calculated during estimation.

B. Anticipation Time

One of the challenges in identifying preview tracking, is the preview time parameter τ_f itself. By responding to the target ahead, HO preview control models essentially include a negative time delay, which makes the system non-causal. To solve this issue, a new reference for the timing of signals is defined by means of an *anticipation time* τ_a, graphically illustrated in Fig. 4. This redefined reference must lie ahead of the previewed viewpoint that people use in their tracking.

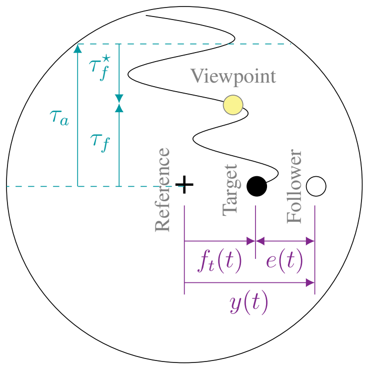


Fig. 4. Preview time (τ_f), anticipation time (τ_a) & apparent time delay ($\tau_f^\star$). Viewpoint and annotations are indicative and not visible to the operator.

The deterministic target of the preview tracking task is anticipated by τ_a seconds (or $N_a = \tau_a/\Delta t$ samples). As a result, the preview time τ_f, defined positive for a negative delay, can be estimated as a normal *apparent time delay* ($\tau_f^\star$) relative to the new reference:

$$H^\star_f(j\omega) = e^{-(\tau_a - \tau_f)j\omega} = e^{-\tau_f^\star j\omega} \tag{26}$$

The anticipation time τ_a thus defines the upper bound for the preview time. If τ_f would surpass τ_a, the apparent delay $\tau_f^\star$ would become negative, leading to an unstable model and divergence of the filter.

C. Parameter Limits

With a lot of free states and parameters to be estimated, the estimation process is sensitive to divergence when the solution space of parameter values is completely unbounded [16], [20]. Especially a sign inversion of most parameters can cause the estimation to become unstable or produce unrealistic results. As noted before, however, the parameters in the cybernetic preview model have been studied extensively. This knowledge can be augmented to the filtering process by means of applying linear inequality constraints on the parameters.

The aim of these constraints is merely to keep the estimated parameters within reasonable limits, avoiding divergence in the estimation process. Their values are chosen well beyond values observed for operators in earlier tracking experiments, however, to include possible other solutions [9]. The limit values are listed in Table I.

TABLE I
HO PARAMETER LIMIT VALUES

Parameter	**Single Integrator**		**Double Integrator**	
	Lower	Upper	Lower	Upper
K_f [-]	0.2	2.5	0.2	2.5
$\omega_{b,f}$ [rad/s]	*n/a*	*n/a*	0.1	10
$\tau_f^\star$ [s]	0.1	$\tau_a - 0.1$	0.1	$\tau_a - 0.1$
K_p [-]	0.2	2.5	0.05	1
K_v [-]	*n/a*	*n/a*	0.05	1.5
ω_{nms} [rad/s]	6	15	6	15
ζ_{nms} [-]	0.1	0.8	0.1	0.8
τ_v [s]	0.1	0.8	0.1	0.8

The constraints are implemented using an estimate projection, as proposed in [35]. This additional step in the state and parameter correction checks whether the corrected parameters in $\boldsymbol{x}^+_{\boldsymbol{s,k}}$ and $\boldsymbol{\theta}^+_{\boldsymbol{k}}$ exceed their imposed limits. If so, sets of constraint matrices D_k and constraint vectors $\boldsymbol{d_k}$ are constructed that express the applicable parameter boundaries:

$$D_{s,k}\boldsymbol{x}^+_{\boldsymbol{s,k}} = \boldsymbol{d_{s,k}} \tag{27}$$

$$D_{p,k}\boldsymbol{\theta}^+_{\boldsymbol{k}} = \boldsymbol{d_{p,k}} \tag{28}$$

With these matrices, either $\boldsymbol{x}^+_{\boldsymbol{s,k}}$ (20) or $\boldsymbol{\theta}^+_{\boldsymbol{k}}$ (25) are mapped to their respective limits.

D. Filter Initialization

The Kalman filter has a number of settings that may affect estimation accuracy. These are:

- the initial state and parameter values $\boldsymbol{x_{s,0}}$ and $\boldsymbol{\theta_0}$,
- the initial state covariance matrices $P_{s,0}$ and $P_{p,0}$ (the pair P_s, P_p is collectively indicated by P),
- the process noise covariance matrices Q_s and Q_p (similarly, the pair Q_s, Q_p is indicated by Q), and
- the measurement noise variance R.

In the first phases of testing, the states and parameters are initialised by their actual values. Researching the sensitivity to initialisation offset for the parameters in $\boldsymbol{x_{s,0}}$ and $\boldsymbol{\theta_0}$ is left for future work. The diagonal elements of P_p and Q_p (and P_s and $Q_{s,k}$ for augmented parameters) are initialised relative to initial parameter values. These factors are optimized using experiment simulations, which will be explained in Subsection IV-F. At first, the matrices P and Q are initialised as per the example of [20]. For P with values on the diagonals of 0.1 for all canonical states, 10 for the HO gains and neuromuscular frequency, and 1 for the other HO parameters. For Q, percentages of the initial parameter values are used, 100% for gains and 10% for other parameters.

For systems that are expected to have approximately constant Gaussian process and measurement noise, the matrices Q and R are typically kept constant over time. For the application of human identification, however, the source of the noise (remnant), is inherently coloured and much more difficult to quantify. This coloured characteristic is not explicitly taken into account in the HO model that forms the basis of the filter. Instead, the example of Popovici *et al.* is followed, namely to adjust the filter to coloured remnant by making matrices $Q_{s,k}$ and R_k time-varying [20].

To achieve this, the process noise Q_s of the canonical state derivative associated with the error signal ($\dot{x}_{s,e}$), is made proportional to the retrospective variance of the error signal (29). In the model presented in App. A, $\dot{x}_{s,e}$ is the fifth canonical state derivative $\dot{x}_{s,5}$. A retrospective time of 5 seconds (or, equivalently, a retrospective amount of samples $N_{\text{retro}} = t_{\text{retro}}/\Delta t = 500$) and a reference value of 0.4 for q^2 are selected, as reported by [20].

$$Q_{s,k}(\dot{x}_{s,e}) = q^2\sigma^2_{e(k-N_{\text{retro}}:k)} \tag{29}$$

Note that unlike the compensatory system, the error signal is not the last state in the canonical order, since it is not the input to the HO model shown in Fig. 3. The noise term is therefore inserted at the correct location in the Q_s-matrix. For the example in Appendix A, this is at the equation for $\dot{x}_{s,5}$. Also note that for preview systems, the signal should, strictly speaking, be the internally calculated error $e^\star$ (Fig. 1). This signal is, however, not explicitly available, so the variance of the actual error $e(t) = f_t(t) - y(t)$ is used instead.

The process noise covariance (Q-values) of all other canonical states is assumed to be 0. The Q-values of the parameters are invariant, and proportional to the initialisation value. Popovici *et al.* used a factor of 1 for gains and 0.1 for all other parameters [20]. Initially, these factors were copied, however, these values are optimized for the preview case in the second stage of the simulation analysis in Section IV.

The measurement noise is made proportional to the retrospective variance of the input signal u, comparable to the process noise earlier.

$$R_k = r^2\sigma^2_{u(k-N_{\text{retro}}:k)} \tag{30}$$

In their review for compensatory tracking, Popovici *et al.* reported a value of $r^2 = 0.7$ used for simulated data and $r^2 = 4$ used for experimental data [20]. In general, the same values were found to work for the preview case as well. For simulated preview tracking data with remnant noise of double-integrator CE dynamics, the value of $r^2 = 4$ was found to achieve better results, since more resilience to the presence of strong remnant noise was incorporated.

IV. Simulated Batch Estimations: Method

A. Preview simulations

With computer simulations of tracking experiments, the DEKF was systematically tested, as it could be subjected to a wide range of conditions. Filter performance can be accurately determined, since the true parameter values are known. Simulated data were generated using the block diagram of Fig. 1 as a *linear parameter varying* (LPV) model. A fifth order Padé approximation is used to incorporate the time delay in the LPV model. A run-in time of 60 s is added to the 120 s measurement, adding up to a total of 180 s. The closed-loop block diagram is transformed into a set of state-space equations in controllable canonical form (31). The discrete state transition matrices are used to go to the next time step (32). The output is calculated with equation (33).

$$\dot{\boldsymbol{x}}_{\text{sim}} = A_{\text{sim}}(\boldsymbol{\theta}(t))\boldsymbol{x}_{\text{sim}} + B_{\text{sim}}\boldsymbol{u}_{\text{sim}} \tag{31}$$

$$\boldsymbol{x}_{\text{sim},\boldsymbol{k}} = \Phi_{\text{sim},k}\boldsymbol{x}_{\text{sim},\boldsymbol{k}} + \Psi_{\text{sim}}\boldsymbol{u}_{\text{sim},\boldsymbol{k}} \tag{32}$$

$$\boldsymbol{y}_{\text{sim},\boldsymbol{k}} = C_{\text{sim},k}\boldsymbol{x}_{\text{sim},\boldsymbol{k}} + D_{\text{sim}}\boldsymbol{u}_{\text{sim},\boldsymbol{k}} \tag{33}$$

In these equations, the state vector $\boldsymbol{x}_{\text{sim}}$ is composed of N_c canonical states: $\boldsymbol{x}_{\text{sim}} = [x_1, \ldots, x_{N_c}]^T$. The number of canonical states for the simulations is larger than for the DEKF's HO model because it incorporates the full closed loop preview model of Fig 1, including the CE. Its exact number depends on the CE dynamics. The matrices $\Phi_{\text{sim},k}$ and Ψ_{sim} are the discrete versions of $A_{\text{sim}}(\boldsymbol{\theta_k})$ and B_{sim}. Similarly, $C_{\text{sim},k} = C_{\text{sim}}(\boldsymbol{\theta_k})$. The external input $\boldsymbol{u}_{\text{sim}}$ consists of the previewed target signal, the remnant noise and the disturbance signal: $\boldsymbol{u}_{\text{sim},\boldsymbol{k}} = [f_t[k_f], n[k], f_d[k]]^T$. The output $\boldsymbol{y}_{\text{sim}}$ consists of the HO input signal $u(t)$ and the CE output $y(t)$.

Finally, the parameter vector $\boldsymbol{\theta}$ contains all the parameters in the HO model blocks of Fig. 1: $\boldsymbol{\theta} = [K_p, K_v, \omega_{nms}, \zeta_{nms}, \tau_v, K_f, \omega_{b,f}]^T$. Note that for the simulations the preview time τ_f is not part of $\boldsymbol{\theta}$. It is instead used to select the correct sample k_f from the forcing function f_t to use in the input vector $\boldsymbol{u}_{\text{sim},\boldsymbol{k}}$:

$$k_f = k + N_f(t) = k + \frac{\tau_f(t)}{\Delta t} \tag{34}$$

B. Forcing Functions and Controlled Element

As shown in Fig. 1, in preview tracking tasks both a target and a disturbance signal are typically used. Here, a multi-sine target signal f_t is used, as defined in (35), that appears quasi-random to a human operator [2]. The amplitude and frequency compositions (A_i and ω_i, respectively) of the high-bandwidth signals from [30] (4 rad/s) are used. By tuning the phase offsets ϕ_i, ten equivalent realisations were created according to the method presented in [30]. The high-bandwidth signal is selected for two reasons. Firstly, HOs are observed to express lower remnant power for a higher target bandwidth [33]. A second reason is that high-bandwidth signals increase the observability of the preview shift.

In conjunction with the target, a disturbance signal f_d is injected at the system output (see Fig. 1). This signal has a comparable bandwidth to the target and is used to split the preview and compensatory dynamics in the identification process [10]. The disturbance is a multi-sine signal built up of frequencies adjacent to those in the target. Again, the parameters of [30] are used to create ten equivalent realisations.

$$f(t) = \sum_{i=1}^{10} A_i \sin(\omega_i t + \phi_i) \tag{35}$$

Two types of CE dynamics are used, also according to [30], that remain constant throughout a tracking run. The first simulates SI dynamics, according to $H_{\text{CE}}(j\omega) = 1.5/(j\omega)$. The second is a double integrator, $H_{\text{CE}}(j\omega) = 5/(j\omega)^2$.

C. Time-variance

Time-varying behaviour is simulated by means of the LPV model introduced in Section IV-A. The scheduled time-variance that is applied to the HO parameter values ($\boldsymbol{\theta}(t)$) within the model structure results in time-varying behaviour. The linear structure of the HO model remains unaltered throughout each simulation, as well as the CE dynamics and remnant parameters. Constant parameter values $\boldsymbol{\theta}_{\boldsymbol{k}} = \boldsymbol{\theta}$ result in a completely linear simulation.

Similar to previous research, an exponential sigmoid function is used to model a smooth step change in HO parameter values, symmetrical around time T [12], [13], [15], [36]. Earlier work uses maximum rate of change G to express the time window in which the transition takes place. This parameter is rewritten to explicitly show the transition time for 99% of the transition (ΔT):

$$\theta(t) = \theta_0 + \frac{\theta_{\text{end}} - \theta_0}{1 + \mathrm{e}^{-G(t-T)}} = \theta_0 + \frac{\theta_{\text{end}} - \theta_0}{1 + \mathrm{e}^{\frac{2\ln 199}{\Delta T}(T-t)}} \tag{36}$$

D. Remnant settings

Although a remnant model is not explicitly included in the filter description, the simulated signals do include remnant noise to make them as close to real human tracking behaviour as possible. The remnant signal $n(t)$ is formed by a first-order model, fed by Gaussian white noise as outlined in Subsection II-B. The remnant parameters (K_n and $\omega_{b,n}$) are kept constant throughout each simulation, since they are not influenced by varying preview time [33]. The break frequency $\omega_{b,n}$ is set depending on the controlled element, namely $\omega_{b,n} = 10$ rad/s for SI and $\omega_{b,n} = 0.01$ rad/s for DI system dynamics [33].

The remnant power ratio P_n depends on several simulation conditions and is also selected in accordance with [33]. For the selected 4 rad/s-bandwidth target signal, the experimentally measured remnant power ratio is 35% and 55% for SI and DI dynamics, respectively. The corresponding gain in our computer simulations, K_n, is found by means of an iterative process, that calculates the remnant power ratio through spectral analysis [33] for a time-invariant HO simulation. Therefore, the gain selection is done with linear simulations, using the mean parameter values. It would be more correct to adjust the remnant characteristics along with HO time-variance. However, since relatively small amounts of remnant time-variance occur for the scenarios considered in this paper, this is left for future work.

E. Performance metrics

Different performance metrics are used to assess the quality of the DEKF estimation result. These metrics are calculated for the measurement time of 120 s only, so the run-in time is used for the filter to converge.

1) Relative Parameter Bias: The mean relative parameter bias indicates whether a parameter is systematically over- or underestimated, or not.

2) NRMSE: For non-zero-mean parameter values, the normalised rms error (NRMSE) is used, specifying the relative deviation of the estimated parameter $\hat{\theta}$ relative to the mean true simulated parameter value:

$$\text{NRMSE}(\hat{\theta}) = \frac{1}{\bar{\theta}_{\text{true}}} \sqrt{\frac{1}{N_{\text{meas}}} \sum_{i=0}^{N_{\text{meas}}} \left(\hat{\theta}(i) - \theta_{\text{true}}(i)\right)^2} \tag{37}$$

3) Variance Accounted For: After the estimation is completed, the HO model output $\hat{u}$ can be resimulated using the isolated model and the identified parameter values. The resulting 'estimated' signal $\hat{u}$ can be compared with the true simulated HO input signal (u) by calculating the *Variance Accounted For* (VAF):

$$\text{VAF} = \left(1 - \frac{\sum_{k=1}^{N_{\text{meas}}} |u(k) - \hat{u}(k)|^2}{\sum_{k=1}^{N_{\text{meas}}} u^2(k)}\right) \times 100\% \tag{38}$$

Aside from the VAF of the entire 120 s measurement window, the 10-second moving VAF was used to show the progression of the estimation performance throughout a tracking run.

F. Simulation Batches

To test the DEKF algorithm, computer simulations with different control task types, CE dynamics, HO time-variance and remnant levels are performed. Every simulation with specific settings is repeatedly executed, using different pairs of target and disturbance forcing functions. In case of simulations with remnant, each of these forcing function realisations is simulated with a number of remnant realisations. These collections of equivalent simulations will be called simulation *batches* and are used to increase the statistical power of the performance analysis.

Different experiment batches are divided over three steps:

1) Remnant-free feasibility batches: The first group of batches was used to show the theoretical estimation potential of the algorithm for HO identification in preview tracking tasks on remnant-free HO data. Simultaneously, these batches test the correct implementation of the algorithm. It is of particular interest whether the preview time τ_f can be estimated as an apparent time delay ($\tau_f^\star$), and whether the two delays can be estimated simultaneously with Padé approximations.

This group consists of four distinct simulation batches without remnant, with time-varying K_p for compensatory tracking and time-varying τ_f for preview tracking. They vary from each other by means of

- a compensatory/preview task, and
- for SI/DI dynamics.

Each of these experiment batches was executed for a batch size of 10 target/disturbance pairs, resulting in total in 40 simulations. Since no remnant is simulated for this group, no additional remnant realisations are used. For the preview estimations, anticipation times of $\tau_a = 0.9$ s for SI dynamics and $\tau_a = 1.5$ s for DI dynamics are used, both 0.3 s larger than the highest simulated preview time.

Fig. 5 shows the effect that a variance in τ_f has on the HO input signal u. In this example, τ_f varies sigmoidally from 0.6 s to 0.3 s with a transition time of 20 s, centered around $T = 50$ s. The result of this adaptation is that the HO anticipates the target less, which causes the stick inputs to lag behind. As a second result, the error signal $e^\star$ will become larger, increasing the amplitude of oscillations in u.

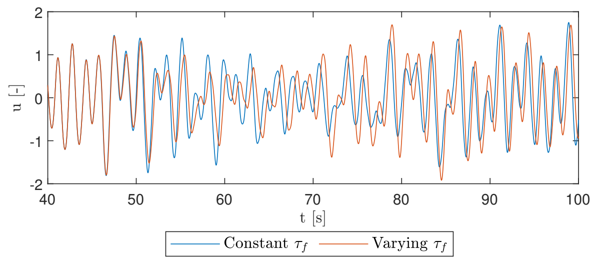


Fig. 5. The effect of time-variance in HO preview (τ_f) on input signal u: a comparison between constant $\tau_f = 0.6$ s and sigmoidal variance from 0.6 s to 0.3 s, centered around $t = 50$ s ($\Delta T = 20$ s, noise-free simulation data).

2) Initialisation optimization: It was noted in [20] that Kalman filter estimation of human control is sensitive to settings and initial conditions and prone to divergence. Hence, a thorough investigation was performed into which settings are suited to obtain good filter performance for preview tracking. The filter has several settings that can be tuned to influence its performance. These are $\boldsymbol{x_{s,0}}$, $\boldsymbol{\theta_0}$, $P_{s,0}$, $P_{p,0}$, Q_s, Q_p and R, as described in Subsection III-D. That subsection also discussed the initialization of $P_{s,0}$ and Q_s for the canonical states, as well as the time-variance introduced in Q_s and R. $\boldsymbol{x_{s,0}}$ and $\boldsymbol{\theta_0}$ are set at their ideal values, i.e. their actual simulated values.

The diagonal process noise variances in Q_s and Q_p of HO parameters were set in [20] as a percentage of their initialisation value: $Q_p[\theta] = q_\theta \theta_0$. For the preview case, these percentages are determined with an optimization procedure. Per parameter, six simulations were used: three realisation sets of forcing functions and realistic remnant power for both SI and DI dynamics (35% or 55%, respectively). In these simulations, the value of the parameter under investigation was varied within their feasible range according to [9], while all others did not vary. An indication for the optimal setting was found by minimizing the NRMSE of the parameter in question:

$$q_{\theta,\text{opt}} = \underset{q_\theta \in (0,1]}{\arg\min} \sum_{\text{sim}=1}^{3} \text{NRMSE}(\hat{\theta}_{\text{sim}} | q_\theta) \tag{39}$$

The optimization was performed for both SI and DI dynamics with MATLAB's `fmincon` function, using the interior-point algorithm. The percentages proposed by [20] were taken as initial conditions. During the estimations in the optimization process, only the gain parameters and the parameter under investigation were estimated. The others were kept constant at their simulated values. Gain parameters were allowed to vary to give the model freedom to adapt to remnant noise.

Because the objective function requires three estimations per function evaluation and multiple evaluations for each iteration step, the optimization is rather time-consuming. Therefore, the number of iterations per optimization was limited to six. At that point, the objective function improved minimally.

3) Performance batches: The goal of the performance batches is to find the DEKF's estimation performance in different experiment scenarios. This group again consists of four batches, with five forcing function sets, each tested with 20 remnant realisations, resulting in 100 simulations per batch in total. Each of these batches is tested with the tuned initialisation settings for P and Q from the previous step. The batches are formed by preview simulations with

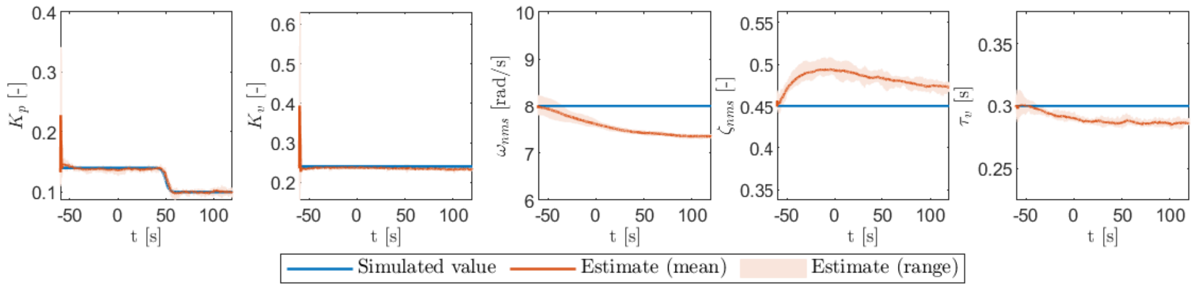
- SI/DI CE dynamics, and
- a remnant power that is half of/the full realistic ratio (17.5%/35% for SI, 27.5%/55% for DI dynamics).

From these batches, the filter performance as a function of CE dynamics and remnant can be analysed. The anticipation times of $\tau_a = 0.9$ s for SI dynamics and $\tau_a = 1.5$ s for DI dynamics are identical to those used in the feasibility batches.

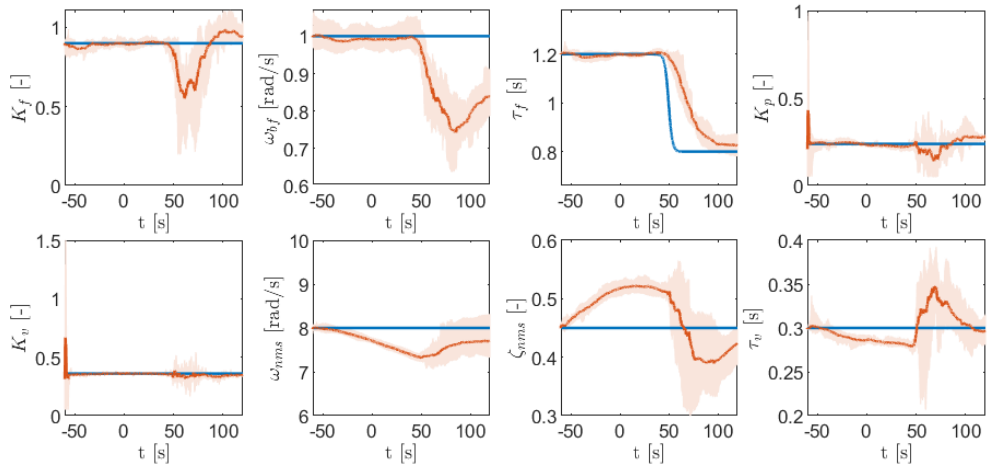
V. Simulated Batch Estimations: Results

A. Remnant-free feasibility batches

The estimation results of the remnant-free feasibility simulations are summarized in Table II. Estimated parameter traces for the compensatory and preview tasks with DI dynamics are shown in Fig. 6.



(a)



(b)

Fig. 6. Remnant-free feasibility batches ($N = 10$): (a) Compensatory tracking, DI dynamics, varying K_p and (b) Preview tracking, DI dynamics, varying τ_f

TABLE II
FEASIBILITY BATCH ESTIMATION RESULTS

Parameter	Comp., SI, K_p var.		Comp., DI, K_p var.	
	Rel. Bias	NRMSE	Rel. Bias	NRMSE
K_p	-0.008	0.016	-0.002	0.026
K_v	–	–	-0.022	0.020
ω_{nms}	-0.018	0.016	-0.071	0.059
ζ_{nms}	+0.176	0.145	+0.071	0.059
τ_v	-0.039	0.033	-0.044	0.036
VAF	*99.6%*		*99.3%*	
Parameter	**Preview, SI, τ_f var.**		**Preview, DI, τ_f var.**	
	Rel. Bias	NRMSE	Rel. Bias	NRMSE
K_f	-0.023	0.063	-0.048	0.130
$\omega_{b,f}$	–	–	-0.111	0.122
τ_f	+0.312	0.252	+0.093	0.109
K_p	-0.073	0.093	-0.028	0.167
K_v	–	–	-0.029	0.063
ω_{nms}	-0.017	0.034	-0.057	0.051
ζ_{nms}	+0.338	0.352	+0.029	0.114
τ_v	+0.142	0.189	+0.007	0.067
VAF	*94.0%*		*96.9%*	

Fig. 6 shows that without remnant noise, the algorithm exhibits promising convergence for all cases. For compensatory tracking, the estimation is nearly perfect. All parameters can be estimated with high accuracy, and the time-variance of K_p has hardly any observable effect on the other identified values. Slight deviations are visible for physical limitation parameters ω_{nms}, ζ_{nms} and τ_v, even before the time-variance takes place. This indicates that the observability of their exact value in the available time-signals is limited and they can exchange their values slightly. These trends are also reflected in the results in Table II. Moreover, control at higher frequencies is typically much stronger for a double integrator CE than for a single integrator [10]. ζ_{nms} was therefore significantly less observable for the SI case, as indicated by the relative bias and NRMSE.

In the preview batches, the DEKF algorithm can also estimate all parameters, including both delays. However, from the linear, remnant-free simulations, it becomes clear that parameters are much more interdependent. The estimation error induced by preview time variations clearly affects the other estimated parameters as well. In and just after the region of time-variance, the spread in the estimates of all parameters can be observed to increase drastically. They can not all simultaneously be uniquely identified from the available signals (f_t, u and y). As a result, the VAF locally dips during the estimation process (see Fig. 7), which in turn causes the overall VAF to be a few percent lower than for the compensatory case (see Table II).

Moreover, parameters ω_{nms}, ζ_{nms} and τ_v show the same general deviation trends that were observed for the compensatory feasibility batch. Finally, parameter estimates of K_p touch the imposed lower limit on several occasions. This illustrates how limits provide an effective way to ensure estimates stay within a feasible range.

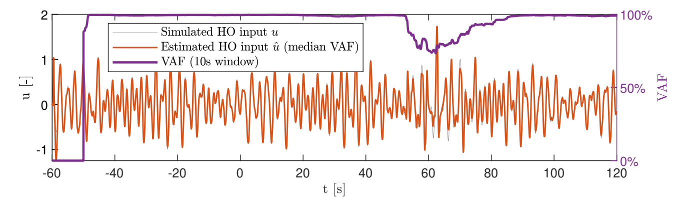


Fig. 7. 10 second retrospective moving VAF for a typical preview DI feasibility estimation. Parameter estimate errors around the τ_f transition region cause the VAF to drop temporarily.

B. Initialisation optimization

1) State and parameter covariance: For the initialisation of a Kalman filter, $P_{s,0}$ and $P_{p,0}$ are typically set rather large. These matrices are updated with each iteration of the filter, and normally converge in mere time steps. The initialisation proposed in [20] was to have diagonal values of 0.1 for canonical states, 10 for gains and neuromuscular frequency and 1 for other parameters. These high values for especially the gains and ω_{nms} caused major parameter volatility in our case in the early stages of filtering, clearly visible as the first jump of K_p and K_v in Fig. 6a. Instead, the parameter variance was initialised based on a percentage of the initial parameter value (comparable to process noise covariance Q): $P_{p,0} = diag([(p_\theta \boldsymbol{\theta_0})^2])$. For ω_{nms} this factor p_θ was set to 0.1, for all parameters, a factor of $p_\theta = 0.5$ was selected. Assigning a more proportionate initial variance to the parameters successfully reduced the erratic initial parameter estimates.

2) Process noise covariance: The diagonal values of Q_p (and Q_s for augmented parameters) are initialised as a percentage of the initial parameter value. For compensatory tracking, [20] used a factor of 1.0 (100%) for gains, and 0.1 (10%) for the other HO parameters. As illustrated by Fig. 8a, the estimated values were highly inconsistent with these settings.

Using the optimization described in Subsection IV-F, Q-initialisation percentages q_θ suitable for the preview problem were found. This resulted in settings of 5% for gains and delays, and 2.5% for the frequency and damping parameters.

The improvement of the new parameters is reflected in both the reduced fluctuations of the typical parameter estimate in a single run, as well as the narrower range of the estimation batch, as illustrated in Fig. 8b. At the same time, these settings did allow for the identification of parameter variance.

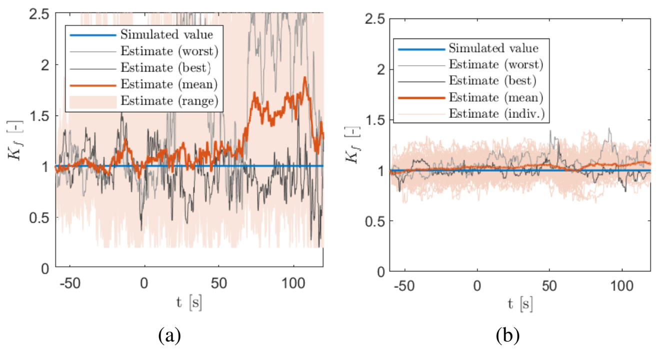


Fig. 8. Illustrative result of the initialisation optimisation: estimation of K_f with (a) the old setting $q_K = 100\%$ and (b) the new setting $q_K = 5\%$ (SI dynamics, $P_n = 35\%$)

TABLE III
ESTIMATION PERFORMANCE: ALL PARAMETERS

Parameter	SI, τ_f var., P_n: 35%		DI, τ_f var., P_n: 55%	
	Rel. Bias	NRMSE	Rel. Bias	NRMSE
K_f	+0.042	0.077	-0.079	0.199
$\omega_{b,f}$	–	–	+0.128	0.193
τ_f	-0.125	0.214	+0.263	0.239
K_p	+0.032	0.106	-0.699	0.581
K_v	–	–	-0.154	0.179
ω_{nms}	+0.000	0.079	+0.169	0.152
ζ_{nms}	+0.992	0.848	+0.035	0.141
τ_v	-0.354	0.328	+0.469	0.414
VAF	*17.8%*		*40.3%*	

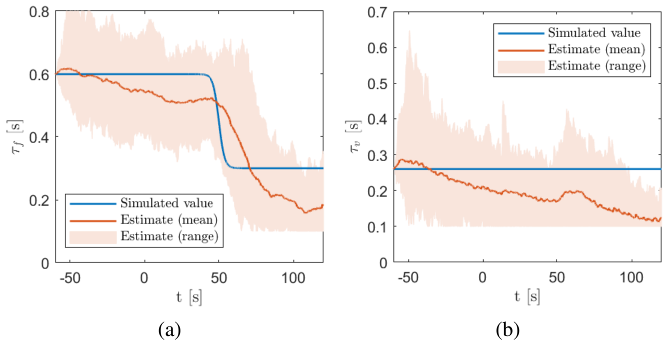


Fig. 9. Exchange of time delay between (a) τ_f and (b) τ_v (SI, $P_n = 35\%$)

C. Performance batches

1) Parameter observability: With the new tuning, the algorithm was tested on preview simulations with remnant to mimic realistic behaviour. The results of the first test, with all parameters free in the estimation, are summarized in Table III.

The parameter interdependence that was already observed in the feasibility batches affects the estimation with remnant noise to a much greater extent. With significant noise levels, the two time delays could not be separated accurately any more. Instead, an exchange of delay time was observed between $\tau_f^\star$ and τ_v, that already starts before the parameter variation (Fig. 9). Note that, since τ_f and $\tau_f^\star$ are related through a sign inversion, a drop in τ_v is compensated with an increasing $\tau_f^\star$ and thus a decreasing τ_f. Due to this exchange, which causes the other parameters to be off as well, the overall performance of the estimation is very low, as expressed by the VAF (17.8%, see Table II). The fluctuating parameter estimates caused by the remnant noise and parameter couplings result in spikes in the resimulated HO input $\hat{u}$ (Fig. 10). These in turn locally lead to extremely low VAF values that significantly influence the overall average.

With all parameters estimated simultaneously, it also becomes clear that the observability of the neuromuscular parameters is limited for the current simulation scenario (i.e. forcing function). Particularly in the SI case, ω_{nms} and ζ_{nms} could drift without greatly affecting the filter output. The performance batches were therefore repeated while keeping τ_v, ω_{nms} and ζ_{nms} constant at their simulated values. This resulted in a significant performance improvement, see Table IV. The improved preview time estimation with constant parameters (compared to Fig. 9) is shown in Fig. 11a.

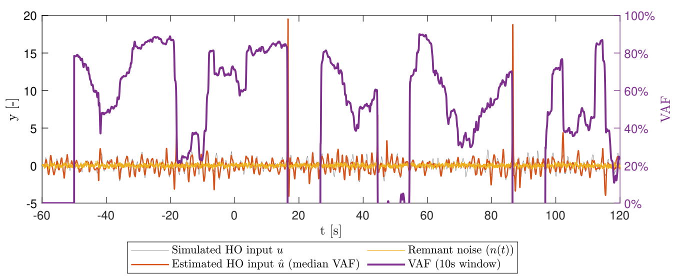
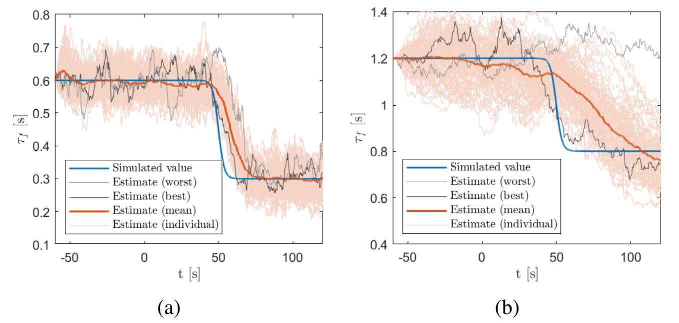


Fig. 10. 10-second retrospective moving VAF for a typical SI estimation with all parameters. Parameter estimate errors create local spikes in identified HO input signal, causing low overall VAF values



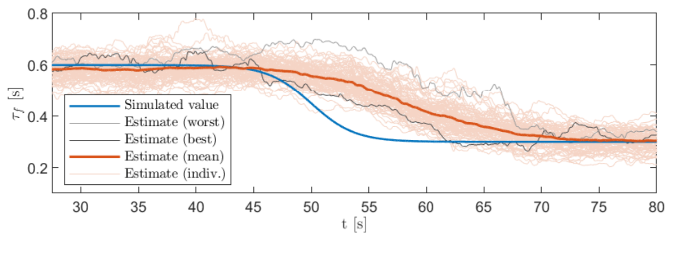


Fig. 11. Estimation of time-varying preview time τ_f with constant τ_v: a comparison between (a) SI ($P_n = 35\%$) and (b) DI dynamics ($P_n = 55\%$). Detail (c) of the SI case highlights the lag in the estimation

2) Effects of System Dynamics: Substantial differences are found in estimation performance depending on the scenario's CE dynamics. Most obviously, the results differ in the observability of certain parameters. The dominant equalization gains, K_p for SI and K_v for DI, have comparable NRMSE results (Table IV). For DI systems, it can be noted that even with a gain K_p at only a fraction of its true value, the VAF is still very high. This leads to the conclusion that for DI dynamics K_p does not contribute enough to the output of the HO model, $\hat{u}$, to make its changes observable under significant remnant.

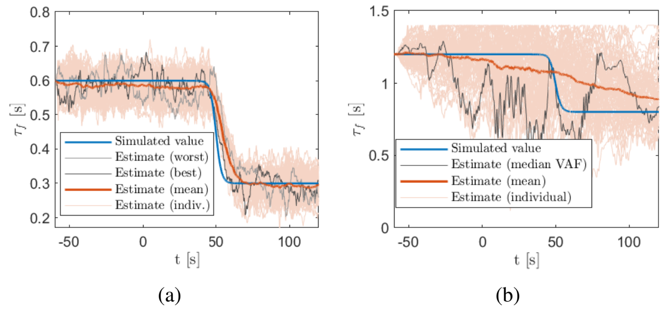
Another interesting difference is present for the preview time τ_f. On average, the identification of time-variance in τ_f is clearly sharper for SI dynamics, with a spread of approximately 0.1 s each way (Fig. 11a). The estimation does lag approximately 15 s on average behind the true parameter time shift, as shown in Fig. 11c. For double integrator dynamics, the estimate shows a much slower detection of the parameter change and a wider spread of roughly 0.2 s (see Fig. 11b). In general, the better parameter observability for SI dynamics leads to a higher overall VAF of 90.3% compared to the 76.4% for the DI case (Table IV).

TABLE IV
PERFORMANCE COMPARISON: CE DYNAMICS (CONSTANT ω_{nms}, ζ_{nms}, τ_v)

Parameter	**SI, τ_f var., P_n 35%**		**DI, τ_f var., P_n 55%**	
	Rel. Bias	NRMSE	Rel. Bias	NRMSE
K_f	-0.016	0.055	-0.181	0.218
$\omega_{b,f}$	–	–	+0.142	0.167
τ_f	+0.056	0.124	+0.072	0.130
K_p	-0.050	0.069	-0.638	0.543
K_v	–	–	-0.059	0.084
VAF	*90.3%*		*76.4%*	

Keeping certain parameters constant is shown to influence, and potentially improve, the estimation results of the other parameters. This raises the question whether keeping all parameters, aside from one, constant at their true value allows for even more accurate tracking of the one remaining parameter. The results of a test where the DEKF only estimates τ_f are presented in Fig. 12. For the SI case, this seems to have the desired effect, see Fig. 12a. The spread of the individual estimates is reduced, especially around the parameter transition region. Also, the lag in the estimation behind the actual parameter step is significantly lower.

For the DI case, however, the opposite is true. A look at the individual estimates that resulted in the median VAF gives



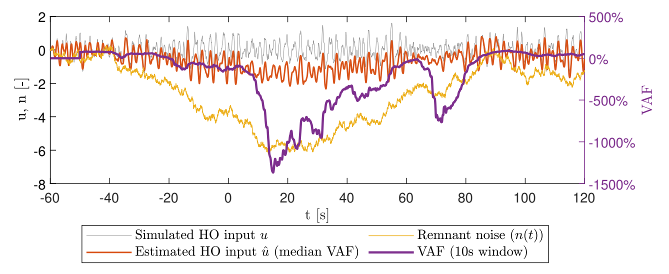
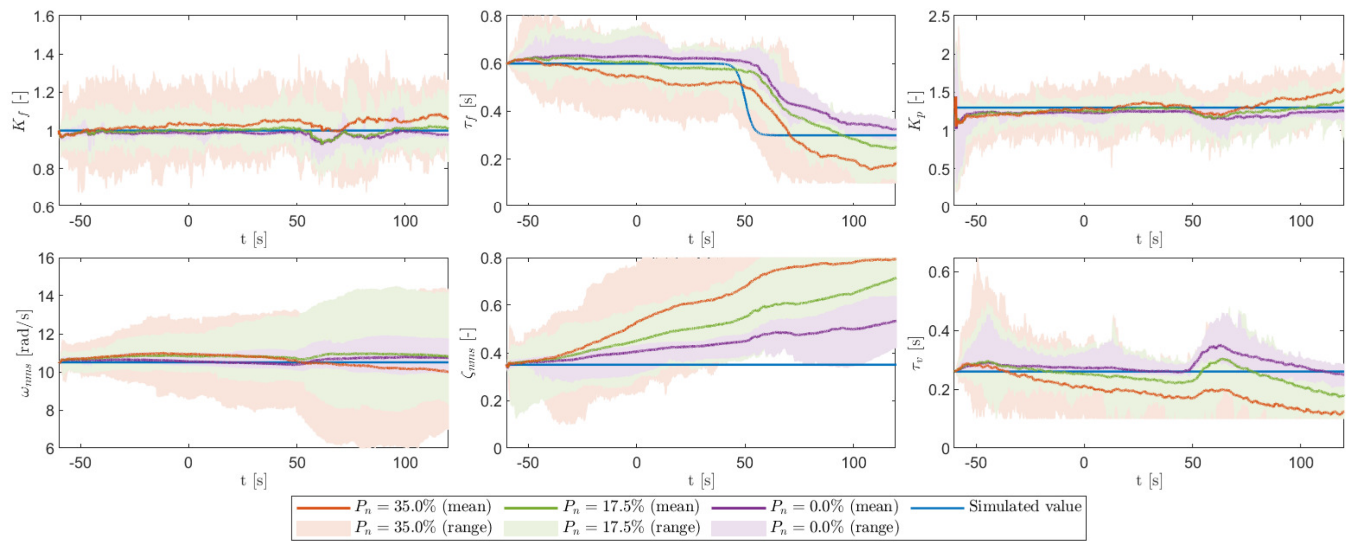


Fig. 12. Estimation of time-varying preview time τ_f with all other parameters constant: a comparison between (a) SI (P_n : 35%, $\omega_{b,n}$: 10 rad/s) and (b) DI dynamics (P_n : 55%, $\omega_{b,n}$: 10 rad/s). Resimulated HO input $\hat{u}$ of the DI case in (c) illustrates the dependence on coloured remnant noise


(a)

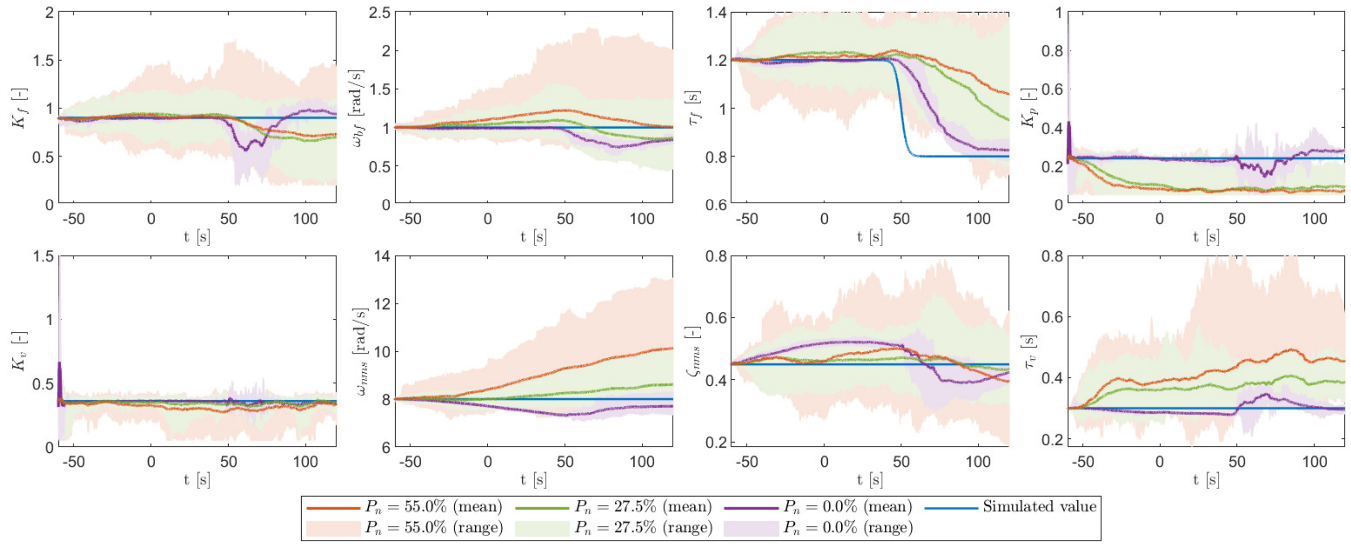

(b)

Fig. 13. Effects of remnant on parameter estimation in a preview tracking task for (a) SI and (b) DI dynamics with time-varying τ_f

insight into why this is the case (Figs. 12b and 12c). Because of the low remnant break frequency ($\omega_{b,n} = 0.01$ rad/s) for DI dynamics, the overall result of the estimate is severely impacted by the coloured properties of the remnant. For SI dynamics, the break frequency is higher than the bandwidth of the target ($\omega_{b,n} = 10$ rad/s). This implies that the remnant noise has a stronger resemblance to white noise, which makes it easier for the DEKF to filter out.

3) Effects of Remnant Power: The effects of remnant on the estimation results for all parameters simultaneously are shown in Fig. 13. Clearly, remnant has a direct effect on the spread in the parameter estimates. The range and the underlying parameter variability increase with remnant power, as expected. The step from half to full remnant power additionally increases the estimation spread. As a result, the estimation accuracy of parameters is also lower. The delay exchange that was noted earlier, most clearly for SI estimation (Fig. 9), becomes more evident as the noise levels increase.

Secondly, with the presence of remnant, the extent to which parameter estimates affect each other is less obvious. The relatively small variance observed for the remnant-free feasibility simulations that was caused by parameter interference, is almost entirely overshadowed by the higher parameter variance resulting from remnant noise.

VI. Experimental Validation

The DEKF algorithm was applied to the experimental tracking data of [9] for validation. In this experiment, the effects of limited preview on HO tracking behaviour were measured for single- and double-integrator dynamics under steady (time-invariant) task conditions. In total, the tracking experiment was performed for eight different conditions for the amount of available preview, ranging from 0 to 2 s. Each experiment run was repeated five times per participant using different target signal realisations. Eight participants took part in the study, resulting in 40 available tracking experiments per condition.

Note that the experiment was designed to invoke constant behaviour, to study the differences between preview time conditions using LTI identification. The estimated parameter time traces are therefore expected to be relatively constant. Unfortunately, there has not yet been a preview tracking task experiment into the effects of time-varying preview.

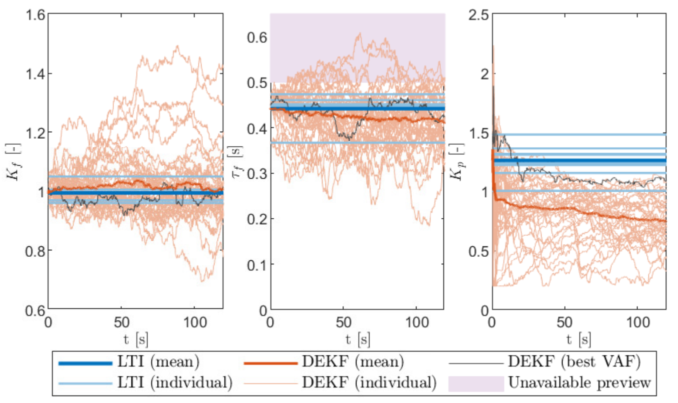
In the original research, it was concluded that there is a limit to the amount of preview operators will use [9]. Available information beyond this *critical preview time* does not contribute anymore to improvement in tracking performance. For SI systems, the critical preview time lies around 0.6 seconds for DI systems this threshold lies around 1.15 seconds [9]. Here, a comparison with the LTI estimation results of two experiment conditions is made. The conditions with 0.5 s and 1.0 s of available preview respectively, were selected for the SI case, while conditions with 0.5 s and 1.66 s of preview were selected for the DI dynamics. The first of each set is picked to see if the DEKF also identifies a preview time close to the available 0.5 seconds, while adhering to this upper limit in available information. The second preview time condition is used to see whether the DEKF finds the same reported upper limit in preview time in the tracking signals, while more information was available to the operators.

For this experimental evaluation, the DEKF was initialised with the LTI identified parameter values for $\boldsymbol{x_{s,0}}$ and $\boldsymbol{\theta_0}$. Also, parameters ω_{nms}, ζ_{nms}, τ_v were kept constant at their LTI-identified values. For the experimental batches, it was found that even tighter initialisation of Q was necessary to restrict excessive parameter variance. The factors q_τ and q_K for the apparent time delay $\tau_f^\star$ and the HO gains were adjusted from 0.05 to 0.01. As reported by [20], $r^2 = 4$ was used for the measurement noise variance R. The anticipation times are set to $\tau_a = 0.9$ s and $\tau_a = 1.1$ s for the low available preview and high available preview SI cases, respectively. These settings ensured ample estimation space for the DEKF, but prevented $\tau_f^\star$ from becoming excessively large, which would reduce the Padé approximation's accuracy. Similarly, settings of $\tau_a = 1.1$ s and $\tau_a = 1.8$ s are used for the DI cases.

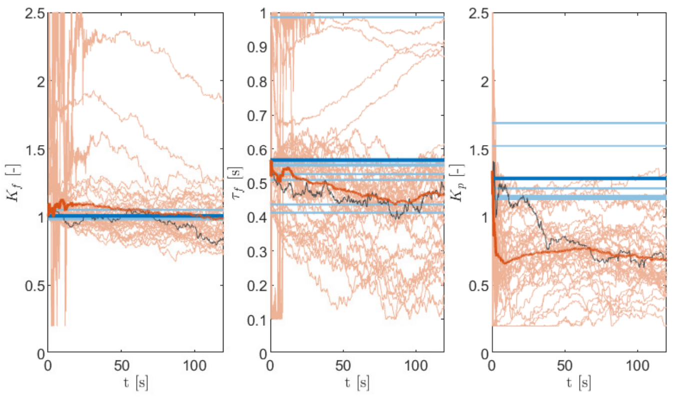
A. *Single Integrator Dynamics*

Results of the estimations for the SI dynamics conditions are summarized in Fig. 14. Several observations stand out from these results. First, the estimate of preview time τ_f generally adhered to the limitations that were expected. The estimated preview time surpassed the available preview by more than 0.05 s on just two occasions. When this happened, the estimated preview time would find its way back to feasible values and it never exceeded 0.6 s. Also, the critical preview time of 0.6 s that was identified in the original research [9], is observable in the time-varying DEKF estimates of the second condition, with the exception of four outliers that match the LTI estimate.

The overall filter performance is compared by means of the average and estimation spread in the 10-second retrospective moving VAF (Fig. 15). It is obvious that for a tracking task that is designed to invoke constant behaviour, LTI estimation outperforms the time-varying approach. Especially on average,



(a)



(b)

Fig. 14. Time-varying estimation of experimental preview tracking data for (a) 0.5s and (b) 1.0s of available preview (SI dynamics). The goal of the experiment was to invoke invariant behaviour, so relatively little time-variance is expected

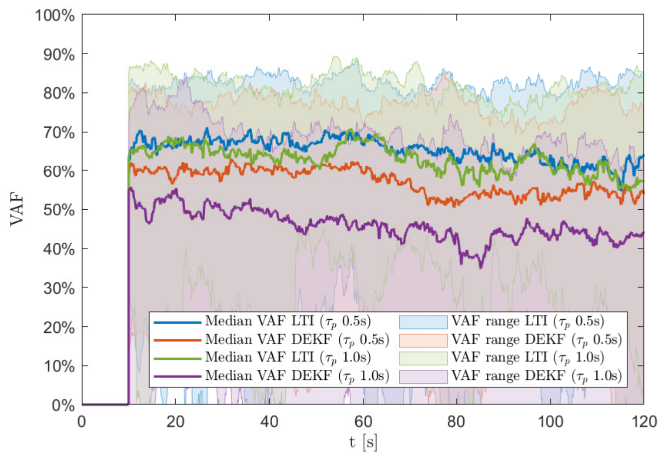
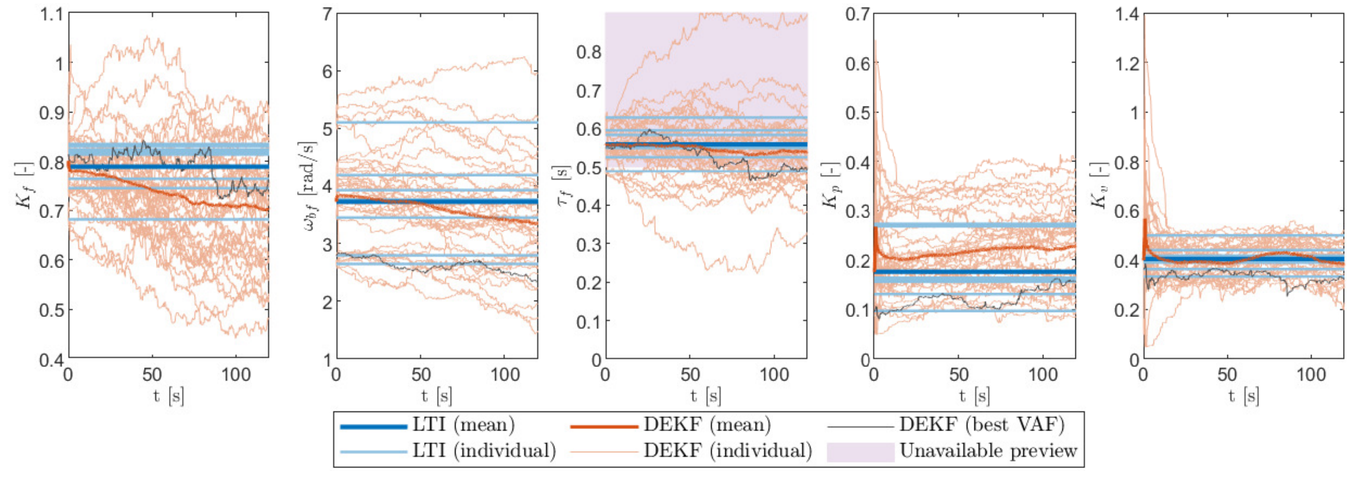


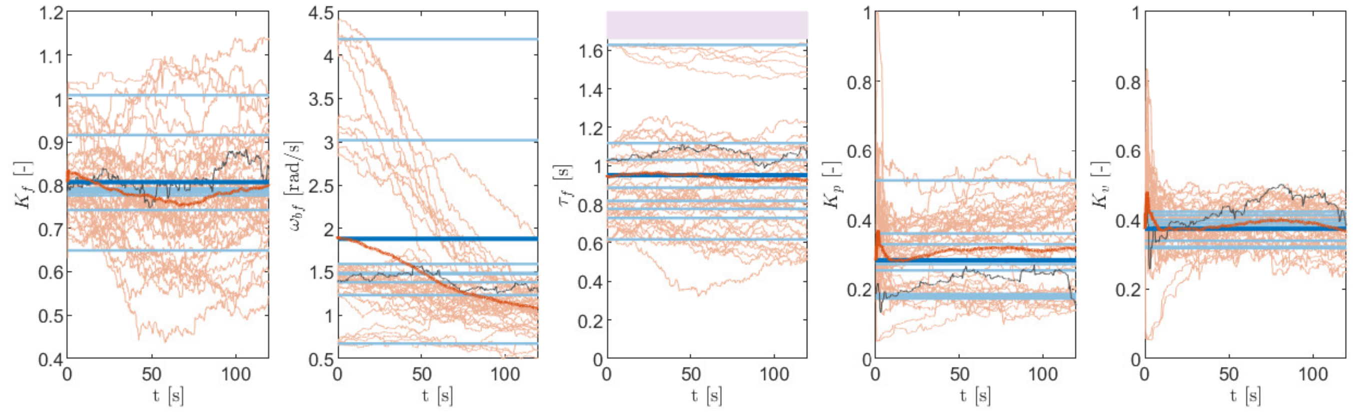
Fig. 15. 10-second retrospective moving VAF comparison for LTI and DEKF identification (SI dynamics)

the LTI method shows an estimation performance that is consistently better than the DEKF.

This validation with experimental results shows that the algorithm is promising for application in real-time systems. Generally, the estimates agree with those found by means of LTI identification. What it shows as well, is that the algorithm still needs to be improved before it is robust enough to be



(a)



(b)

Fig. 16. Time-varying estimation of experimental preview tracking data for (a) 0.5s and (b) 1.66s of available preview (DI dynamics). The goal of the experiment was to invoke invariant behaviour, so relatively little time-variance is expected

applied online. First, for the condition with 1.0s preview, the estimated preview time was very low on a large number of runs, which is not realistic when taking the LTI estimates and the original conclusions about critical preview time into consideration [9]. Also, the filter showed poor convergence in the initial stages on four out of the available 40 runs with ample preview. Finally, the estimate of equalization gain K_p is structurally lower than found with LTI estimation. The lower VAF values for the DEKF estimates compared to their LTI counterparts suggest that there is potential to improve these results with the available signals.

B. Double Integrator Dynamics

Results for DI controlled element dynamics are summarized in Fig. 16. The first thing that stands out, is the preview time estimated for the case with $\tau_p = 0.5$ s (Fig. 16a). In this condition, the available preview is limiting the information used by the operator. Much more so than for the SI case does the DEKF estimate a preview time that is not available to the HO. As Fig. 16a shows, this was also the case for the LTI estimates. Since the preview time is to be estimated from a sample shift between time signals, and because the HO also generates lead by observing the system output, it seems logical that the preview time limit is not perfectly observable. The estimate exceeded 0.65 s for a prolonged amount of time on four individual runs, with one clear outlier. On the contrary, the estimated preview time dips below 0.4 s in only two estimation runs. This supports the finding of [9] that all available preview is used by the HO.

For the condition with ample available preview information ($\tau_p = 1.66$ s), the estimation generally adheres to the critical preview time of 1.15 s. The preview time estimates for one subject that, according to the LTI estimation, did use more preview than the critical limit, trend downwards, away from the unavailable preview boundary. The estimates of the other parameters are also generally in line with the results from the LTI estimation, with the exception of the far view break frequency $\omega_{b,f}$. The DEKF estimates this parameter significantly lower.

The overall estimation performance is expressed by the 10-second retrospective VAF in Fig. 17. It can be seen that the DEKF's results are pretty much equivalent to those obtained with LTI estimation. This was to be expected from the parameter values that are close to equal. To control a system with double-integrator dynamics, lead is introduced, which causes the control input to have more power in the higher frequency

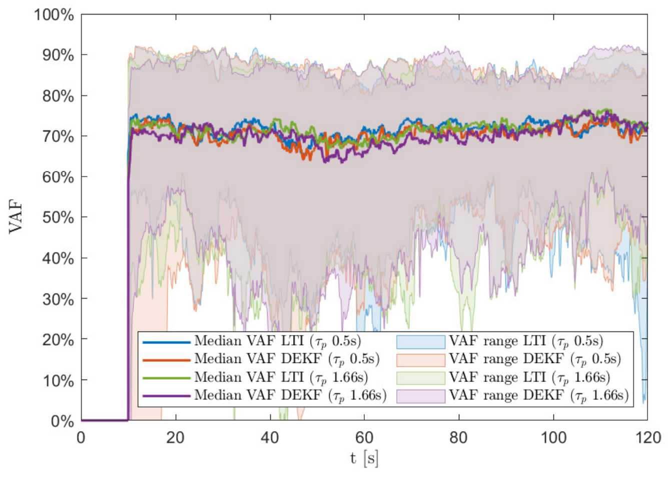


Fig. 17. 10-second retrospective moving VAF comparison for LTI and DEKF identification (DI dynamics)

regions [2], [10]. This results in better observability of the parameter values, and a notable improvement in the overall performance of the DEKF for tracking tasks with DI dynamics.

VII. Discussion

A. Results Discussion

This research explored the potential for time-varying parameter estimation in preview tracking tasks using a Dual Extended Kalman Filter (DEKF). This filter was centered around a linear cybernetic HO model obtained using LTI identification on experimental tracking data [10]. The complexity of this model was kept as low as possible to maximize observability, resulting in six identifiable parameters for tasks with SI dynamics and eight for DI dynamics. The parameter values are estimated, based on two model input signals (the tracking target and the system output feedback), together with one model output for state and parameter corrections (the HO input signal u). Compared to prior work [20], the DEKF was augmented with parameter boundaries, that kept the parameters within their documented feasible range [9]. The preview time parameter was of particular interest, as the amount of preview can be highly variable in practical situations. Preview time was estimated by means of anticipating the target signal ahead of both current state as well as any potential preview.

The filter was applied to a range of time-varying simulations, first completely remnant-free. The remnant-free simulations show that the DEKF with parameter limits has the potential to estimate preview tracking parameters. Using a time-shifted "anticipated" target signal, a time-varying preview time could be successfully identified as a normal (positive) delay parameter. However, even for ideal linear behaviour, interference was observed between the different parameters. This suggests that the observability of individual parameter values based on the available input and output signals is at times minimal. As a result, the estimated time-variance in the preview time was observed to lag behind the true parameter transition. Besides, it took tens of seconds for the other parameters to converge back to their true values.

This interference problem was, however, partly resolved. From the obtained results, an exchange in delay value was observed between τ_v and τ_f. Therefore, keeping τ_v constant greatly enhances the estimate of τ_f. In practical applications, the effects of this simplification are expected to be minimal, as fluctuations in τ_v are likely only within about a tenth of a second [9]. Any offset that is present in the estimate of τ_v adds to the estimate of τ_f due to the delay exchange. These expected additional fluctuations of τ_f fall within the fluctuations that are currently observed already.

For more realistic simulations, remnant noise was included as low-pass filtered Gaussian white noise, with its source located at the feedback of the system output y [33]. Despite the remnant's coloured characteristics, it is accounted for by means of zero-mean Gaussian white noise terms at the appropriate location in both the state and output equations [20]. The assumed variance, represented by covariance matrices Q_s and R, was varied proportional to the tracking error e and the HO input signal u. This gave the filter some resilience to the remnant of a human in a closed-loop task that appears in both the input and output signals of the filter.

Explicitly including the coloured characteristics of remnant noise was found to be a major field of improvement for the current filter implementation. The results of the estimation for only τ_f (Fig. 12) highlight the importance of accurate inclusion of the remnant in the DEKF model. Especially for the DI case, where the coloured characteristics of remnant are more pronounced as a result of the lower remnant break frequency, the current DEKF is unable to accurately estimate parameter values and filtering the coloured noise from the available signals.

By setting the physical limitation parameters (ω_{nms}, ζ_{nms}, τ_v) constant, acceptable results were obtained on average, particularly for simulated SI tracking. However, these results were not always replicable for individual estimation runs. It should be noted that for quality of the estimation result, there is a clear distinction between the NRMSE of individual parameters and the VAF of the entire filter's output. Poor tracking quality of individual parameters (low NRMSE) while simultaneously maintaining good quality of the output of the filter (high VAF) suggests that the model can provide a good fit from the observed signals with multiple configurations of parameters. In such case, the precise values of individual parameters are simply not observable in those signals.

Nevertheless, the significant parameter deviations and fluctuations in estimations as a result of remnant noise did in fact result in low values for the VAF, both locally and globally. This means that a critical attitude should be adopted to any observed local time-variance, when using the filter as a tool in *a posteriori* analysis of observed tracking behaviour. Only larger trends were in some cases shown to truly reflect behavioural adaptation. What this also implies is that the current implementation is not yet sufficient for real-time human-in-the-loop applications. For the filter to form the basis of online driver monitoring or especially haptic feedback systems, a more reliable, consistent and robust estimation performance is required.

Another element that greatly influences the quality of the estimation outcome, is the initialisation of the filter. This research provides a number of guidelines to tuning the filter for various experimental conditions. The revised initialisation for matrices P_s and P_p lowered the observed spikes during the converge phase at start-up of the filter. The initialisation percentages for parameters q_θ in Q_s and Q_p control the level of fluctuations in the estimated parameter traces, together with measurement variance factor r^2. In general, lowering q_θ or increasing r^2 has the effect of increasing sensitivity to parameter adaptation, with the side-effect of higher steady state variance observed in the estimates. The reverse is also true, if the level of parameter fluctuations is to be reduced.

Despite many efforts to develop more specific guidelines, the tuning of all covariance matrices for both filters remains a time-consuming process. Repeated attention was required when changing experiment variables, such as parameter time-variance, CE dynamics and especially simulated remnant inclusion or experimental data. Again, explicit incorporation of the remnant model might provide a solution for this hurdle. The statistical and coloured properties of remnant have been studied for various conditions [33]. These findings could give applicable directions for the settings of the filter.

B. Further Research

This research was a promising first step in the application of a Dual Extended Kalman Filter for the purpose of time-varying HO parameter estimation. The research provides evidence for its viability and as such serves as a proof of concept. A validation of the algorithm on experimental data showed that the DEKF can be successfully applied to experimental human tracking behaviour. However, further improvements are necessary for implementation in practical applications.

1) Development of the algorithm: Especially the analysis on simulated tracking data shows that there is still room for improvement in estimation performance. There are several areas where the current implementation can be further developed. First of all, the knowledge of remnant characteristics is currently not explicitly implemented in the DEKF's system description. The remnant model, that has been experimentally shown to accurately represent human remnant, could however be incorporated [33]. This might not only improve the estimation performance, but could also enable estimation of time-varying remnant characteristics, i.e., power (K_n) and break frequency ($\omega_{b,n}$). Additionally, making the remnant noise time-varying based on the instantaneous HO parameters and system dynamics would make for more realistic simulations.

Secondly, the observability Gramian can serve as an explicit measure to indicate how observable trends in the individual parameters are also locally, i.e. around parameter time-variations [37]. In conjunction with the incorporation of a remnant model, this metric could quantify the improvement that this or other developments make.

Thirdly, the initialisation of the filter could use more attention. Ideally, further testing can result in a general set of heuristic rules for filter tuning based on different experiment conditions. These include CE dynamics, target signal characteristics and remnant power (or true experimental test data). Also, different measurement covariance matrices R could be used for the state and parameter filter.

Finally, the implementation of parameter limits can perhaps be improved. Enforcing these limits with Kalman gain projection rather than estimate projection was attempted [35]. Since this method also updates the state and parameter covariance based on the imposed limits, this could potentially reduce the effect of a parameter sticking to its limit, instead of returning to more feasible solution space. A forgetting factor, commonly used to diminish ongoing recursive effects over time, could be part of an improved solution [15], [17].

2) Other algorithms: This research showed that it is generally possible to estimate time-varying parameters in preview tracking tasks. This includes preview time, by anticipating the deterministic target signal and estimating preview as a delay parameter. The presented procedure could be used to test other promising algorithms. Primary candidates are different types of Kalman Filters, for example the (D)UKF. This filter potentially improves performance when the modeled system is highly nonlinear because it does not require a linearisation step [38]. Also, it has already been successfully applied to experimental human control data [18], [19], [21].

3) Continued research: The current recursive estimator, or improved future versions, could be subjected to more elaborate testing. Most notably, the simulated time-variance used in this study to test the DEKF's potential was rather limited, and the experiment used for validation did not promote any time-varying behaviour. More extensive time-variance could be simulated by changing CE dynamics over time [13], [15], [20]. This induces time-variance in different parameters simultaneously, forming a very interesting test case.

Since the filter showed promising convergence for experimental data, a dedicated experiment that actively promotes time-varying preview tracking behaviour is an important next step. This could be achieved through the aforementioned variable CE dynamics, varying the available preview, or constructing a forcing function with variable properties. In its current implementation, the filter is suited for real-time parameter identification. This means that the filter can not only be applied *a posteriori*, but also online during a human-in-the-loop experiment. Setups with driver monitoring or haptic feedback, based on the estimated behaviour are thus among the possibilities, even in realistic driving scenarios.

VIII. Conclusion

This research implemented a Dual Extended Kalman Filter for the purpose of estimating time-varying human control in preview tracking tasks. For the first time, adaptations in preview time were successfully observed using a linear HO model, purely based on the previewed target signal and the input and output of the controlled element. With the use of an apparent time delay, relative to a reference in the target signal ahead of the viewpoint of the operator, this preview time was estimated. The explicit use of parameter limits keeps all parameters within their documented, feasible range to avoid filter divergence.

Major improvements in estimation performance were achieved by keeping certain parameters constant. This was found to eliminate interference between parameters due to their limited observability in the available signals. The most notable interference was the exchange of time delay between the apparent preview delay $\tau_f^\star$ and the visual delay τ_v.

For the current filter implementation, remnant noise with strongly coloured characteristics was found to have a great influence on the quality of the estimates. Earlier research suggests that this coloured remnant is particularly present when controlling a system with DI dynamics. Explicitly incorporating a remnant model in the HO model in the DEKF could provide a solution.

A second weakness of the current filter is its sensitivity to tuning and initialisation. Despite attempts to develop specific tuning instructions, only general guidelines for the filter's various settings were found. Some testing was still required between runs of varying experimental conditions to come to acceptable results.

The major strength of the presented DEKF, is that it allows for real-time application, without any a priori scheduling of parameter variations. Validation using experimental tracking data confirmed that the algorithm is a viable method for concurrent state and parameter estimation in preview tracking tasks. With additional development to improve estimation quality, the algorithm has the potential to form the basis in advanced systems that can monitor and assist human operators.

References

[1] L. R. Young, "On Adaptive Manual Control," *Ergonomics*, vol. 12, no. 4, pp. 635–674, 1969.

[2] 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. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1698271

[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*, vol. 48, no. 5, pp. 468–485, Oct. 2018.

[4] L. Li, D. Wen, N.-N. Zheng, and L.-C. Shen, "Cognitive cars: A new frontier for adas research," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 395–407, 2011.

[5] D. A. Abbink and M. Mulder, "Haptic shared control: smoothly shifting control authority?" *Cognition, Technology & Work*, 2011.

[6] T. B. Sheridan, "Three models of preview control," *IEEE Transactions on Human Factors in Electronics*, no. 2, pp. 91–102, 1966.

[7] J. J. Gibson, "Visually controlled locomotion and visual orientation in animals," *British journal of psychology*, vol. 49, no. 3, pp. 182–194, 1958.

[8] M. F. Land and D. N. Lee, "Where we look when we steer," *Nature*, vol. 369, pp. 742 – 744, Jun. 1994.

[9] K. van der El, S. Padmos, D. M. Pool, M. M. van Paassen, and M. Mulder, "Effects of Preview Time in Manual Tracking Tasks," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 5, pp. 486–495, Oct. 2018.

[10] K. van der El, D. M. Pool, H. J. Damveld, M. M. van Paassen, and M. Mulder, "An Empirical Human Controller Model for Preview Tracking Tasks," *IEEE Transactions on Cybernetics*, vol. 46, no. 11, pp. 2609–2621, Nov. 2016.

[11] A. J. Pronker, D. A. Abbink, M. M. Van Paassen, and M. Mulder, "Estimating driver time-varying neuromuscular admittance through lpv model and grip force," in *Proceedings of the the 20th IFAC World Congress, Toulouse, France*, ser. IFAC-PapersOnLine, vol. 50. Elsevier, 2017, pp. 14 916–14 921.

[12] 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, Jul. 2017, pp. 15 853–15 858.

[13] 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.

[14] P. M. T. Zaal and B. T. Sweet, "Estimation of Time-Varying Pilot Model Parameters," in *AIAA Modeling and Simulation Technologies Conference*, 2011, p. 6474.

[15] W. Plaetinck, D. M. Pool, M. M. Van Paassen, and M. Mulder, "Online Identification of Pilot Adaptation to Sudden Degradations in Vehicle Stability," in *Proceedings of the 2nd IFAC Conference on Cyber-Physical & Human-Systems, Miami (FL)*, 2018.

[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. [Online]. Available: http://hdl.handle.net/2060/19760006748

[17] 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.

[18] J. Rojer, D. M. Pool, M. M. van Paassen, and M. Mulder, "UKF-based Identification of Time-Varying Manual Control Behaviour," in *Proceedings of the 14th IFAC Symposium on Analysis Design and Evaluation of Human Machine Systems Tallinn, Estonia*, 2019, pp. 109–114.

[19] T. K. Mandal and Y. Gu, "Online Pilot Model Parameter Estimation Using Sub-Scale Aircraft Flight Data," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego (CA)*, no. AIAA-2016-0636, 2016.

[20] 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.

[21] C. You, J. Lu, and P. Tsiotras, "Nonlinear Driver Parameter Estimation and Driver Steering Behavior Analysis for ADAS Using Field Test Data," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 686–699, 2017.

[22] L. Nelson and E. Stear, "The simultaneous on-line estimation of parameters and states in linear systems," *IEEE Transactions on automatic Control*, vol. 21, no. 1, pp. 94–98, 1976.

[23] H. Godthelp, "Vehicle Control During Curve Driving," *Human Factors*, vol. 28, no. 2, pp. 211–221, 1986.

[24] E. R. Boer, "Satisficing curve negotiation: Explaining drivers' situated lateral position variability," in *Proceedings of the 13th IFAC Symposium on Analysis, Design and Evaluation of Human Machine Systems Kyoto, Japan*, 2016, pp. 183–188.

[25] 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*, vol. 48, no. 5, pp. 496–508, Oct. 2018.

[26] ——, "Identification and Modeling of Driver Multiloop Feedback and Preview Steering Control," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Myazaki, Japan*, Oct. 2018, pp. 1227–1232.

[27] K. van der El, J. Morais Almeida, D. M. Pool, M. M. van Paassen, and M. Mulder, "The Effects of Motion Feedback in Manual Preview Tracking Tasks," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Denver (CO)*, no. AIAA-2017-3472, 2017.

[28] L. D. Reid and N. H. Drewell, "A Pilot Model for Tracking with Preview," in *Proceedings of the 8th Annual Conference on Manual Control*, 1972, pp. 191–204. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0754908#page=196

[29] K. van der El, D. M. Pool, M. M. van Paassen, and M. Mulder, "Effects of Preview on Human Control Behavior in Tracking Tasks With Various Controlled Elements," *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1242–1252, Apr. 2018.

[30] ——, "Effects of Target Trajectory Bandwidth on Manual Control Behavior in Pursuit and Preview Tracking," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 1, pp. 68–78, 2020.

[31] E. Rezunenko, K. van der El, D. M. Pool, M. M. van Paassen, and M. Mulder, "Relating Human Gaze and Manual Control Behavior in Preview Tracking Tasks with Spatial Occlusion," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Myazaki, Japan*, Oct. 2018, pp. 3440–3445.

[32] G. J. Silva, A. Datta, and S. Bhattacharyya, "Controller design via padé approximation can lead to instability," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, vol. 5. IEEE, 2001, pp. 4733–4737.

[33] K. van der El, D. M. Pool, and M. Mulder, "Analysis of Human Remnant in Pursuit and Preview Tracking Tasks," in *Proceedings of the 14th IFAC Symposium on Analysis Design and Evaluation of Human Machine Systems Tallinn, Estonia*, 2019, pp. 145–150.

[34] W. H. Levison, S. Baron, and D. L. Kleinman, "A Model for Human Controller Remnant," *IEEE Transactions on Man-Machine Systems*, vol. 10, no. 4, pp. 101–108, Dec. 1969.

[35] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

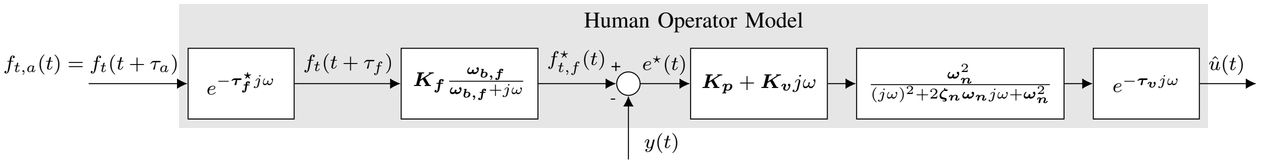
[36] A. van Grootheest, D. M. Pool, M. M. van Paassen, and M. Mulder, "Identification of Time-Varying Manual Control Adaptations with Recursive ARX Models," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Kissimmee (FL)*, no. AIAA-2018-0118, 2018.

[37] G. J. Moszczynski, M. Giamou, J. M. Leung, J. Kelly, and P. R. Grant, "An observability based approach to flight path reconstruction of uninformative coupled aircraft trajectories: A case study considering stall maneuvers for aircraft certification," in *AIAA Scitech 2019 Forum*, 2019, p. 0012.

[38] E. A. Wan and R. Van Der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*. IEEE, 2000, pp. 153–158.

APPENDIX A
STATE-SPACE SYSTEM OF THE HO PREVIEW MODEL

This appendix details the nonlinear state-space equations that were used in the DEKF, analogous to Appendix A of [20] that applied a DEKF to compensatory tracking. The model behind these equations is depicted in Fig. 18. The formation of the state-space equations was explained in Subsection III-A. In order to convert the (exponential) apparent preview delay ($\tau_f^\star$) and visual time delay (τ_v) from Fig. 18 to state-space, they were represented by a third order Padé approximation (see Subsection II-A). Neuromuscular parameters ω_{nms} and ζ_{nms} are referred to as ω_n and ζ_n to save space in the already lengthy equations.



Human Operator Model
$f_{t,a}(t) = f_t(t+\tau_a)$
$e^{-\tau_f^\star j\omega}$
$f_t(t+\tau_f)$
$K_f \frac{\omega_{b,f}}{\omega_{b,f}+j\omega}$
$f^\star_{t,f}(t)$
+
-
$e^\star(t)$
$K_p + K_v j\omega$
$\frac{\omega_n^2}{(j\omega)^2+2\zeta_n\omega_n j\omega+\omega_n^2}$
$e^{-\tau_v j\omega}$
$\hat{u}(t)$
$y(t)$

Fig. 18. Isolated block diagram of the HO in preview tracking tasks with parameters (far view response only) [10]

$$\boldsymbol{x_s} = [x_{s,1} \;\; x_{s,2} \;\; x_{s,3} \;\; x_{s,4} \;\; x_{s,5} \;\; x_{s,6} \;\; x_{s,7} \;\; x_{s,8} \;\; x_{s,9} \;\; K_p \;\; K_v]^T \tag{40}$$

$$\boldsymbol{\theta} = [\omega_n \;\; \zeta_n \;\; \tau_v \;\; K_f \;\; \omega_{b,f} \;\; \tau_f^\star]^T \tag{41}$$

$$\dot{\boldsymbol{x_s}}(t) = f\left(\boldsymbol{x_s}(t), f_{t,a}(t), y(t), \boldsymbol{\theta}(t)\right) + \boldsymbol{w_s}(t) \tag{42}$$

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{w_p}(t) \tag{43}$$

$$\hat{u}(t) = g\left(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t)\right) + v(t) \tag{44}$$

$$f\left(\boldsymbol{x_s}, f_{t,a}, y, \boldsymbol{\theta}\right) = \begin{bmatrix} x_{s,2} \\ x_{s,3} \\ x_{s,4} \\ x_{s,5} \\ -y - 120x_{s,1}\omega_n^2/\tau_v^3 - x_{s,2}(60\tau_v\omega_n^2 + 240\zeta_n\omega_n)/\tau_v^3 - \\ \cdots - x_{s,3}(12\omega_n^2\tau_v^2 + 120\zeta_n\omega_n\tau_v + 120)/\tau_v^3 - x_{s,4}(\omega_n^2\tau_v^3 + 24\zeta_n\omega_n\tau_v^2 + 60\tau_v)/\tau_v^3 - \\ \cdots - x_{s,5}(2\omega_n\zeta_n\tau_v^3 + 12\tau_v^2)/\tau_v^3 + 120x_{s,6}K_f\omega_{b,f}/\tau_f^{\star 3} - \\ \cdots - 60x_{s,7}K_f\omega_{b,f}/\tau_f^{\star 2} + 12x_{s,8}K_f\omega_{b,f}/\tau_f^\star - x_{s,9}K_f\omega_{b,f} \\ x_{s,7} \\ x_{s,8} \\ x_{s,9} \\ f_{t,a} - 120x_{s,6}\omega_{b,f}/\tau_f^{\star 3} - x_{s,7}(60\omega_{b,f}\tau_f^\star + 120)/\tau_f^{\star 3} - \\ \cdots - x_{s,8}(12\omega_{b,f}\tau_f^{\star 2} + 60\tau_f^\star)/\tau_f^{\star 3} - x_{s,9}(\omega_{b,f}\tau_f^{\star 3} + 12\tau_f^{\star 2})/\tau_f^{\star 3} \\ 0 \\ 0 \end{bmatrix} \tag{45}$$

$$g\left(\boldsymbol{x_s}, \boldsymbol{\theta}\right) = \begin{bmatrix} 120x_{s,1}K_p\omega_n^2/\tau_v^3 + x_{s,2}(120K_v\omega_n^2 - 60K_p\omega_n^2\tau_v)/\tau_v^3 + \\ \cdots + x_{s,3}(12K_p\omega_n^2\tau_v^2 - 60K_v\omega_n^2\tau_v)/\tau_v^3 - x_{s,4}(K_p\omega_n^2\tau_v^3 - 12K_v\omega_n^2\tau_v^2)/\tau_v^3 - x_{s,5}K_v\omega_n^2 \end{bmatrix} \tag{46}$$

In accordance with the research for compensatory tracking by Popovici *et al.* , the total derivative $G^{tot}_{p,k}$ is used for the correction step of the parameter filter [20]. This derivative is calculated as follows:

$$G^{tot}_{p,k} = \left.\frac{dg(\boldsymbol{x^-_{s,k}}, \boldsymbol{\theta})}{d\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta^-_k}} \tag{47}$$

$$\left.\frac{dg(\boldsymbol{x^-_{s,k}}, \boldsymbol{\theta})}{d\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta^-_k}} = \left.\frac{\partial g(\boldsymbol{x^-_{s,k}}, \boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta^-_k}} + \left.\frac{\partial g(\boldsymbol{x_s}, \boldsymbol{\theta^-_k})}{\partial\boldsymbol{x_s}}\right|_{\boldsymbol{x_s}=\boldsymbol{x^-_{s,k}}} \frac{d\boldsymbol{x^-_{s,k}}}{d\boldsymbol{\theta}} \tag{48}$$

$$\frac{d\boldsymbol{x^-_{s,k}}}{d\boldsymbol{\theta}} = \left.\frac{\partial f(\boldsymbol{x^+_{s,k-1}}, f_{t,a,k-1}, y_{k-1}, \boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta^-_k}} + \Phi_{s,k-1}\frac{d\boldsymbol{x^+_{s,k-1}}}{d\boldsymbol{\theta}} \tag{49}$$

$$\frac{d\boldsymbol{x^+_{s,k-1}}}{d\boldsymbol{\theta}} = \frac{d\boldsymbol{x^-_{s,k-1}}}{d\boldsymbol{\theta}} - K_{s,k-1}\left.\frac{dg(\boldsymbol{x^-_{s,k-1}}, \boldsymbol{\theta})}{d\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta^-_{k-1}}} \tag{50}$$

Equation 49 is slightly different than originally published in [20]. After the original formulation caused divergence of the filter, it was established after consulting the authors that the discrete transition matrix $\Phi_{s,k-1}$ should be used. The three parameters from the previous time-step ($k-1$) used to compute $\frac{d\boldsymbol{x^+_{s,k-1}}}{d\boldsymbol{\theta}}$ are all initialised as 0 [20].

II
Preliminary Thesis Report

As graded for AE4020 Literature Study

1 Introduction

Vehicles of all different sorts on land, water and through the air have enabled humans to explore the world and even outer space for centuries, making us move quicker and further than otherwise possible. In the act of steering a vehicle, McRuer et al. (1977) distinguished a hierarchy of three levels: navigation, guidance and control. *Navigation* determines how to arrive at our destination from the given start point. It involves choosing the route or course to follow, as well as regularly checking current position and progress. On a more detailed level, the trajectory specific to the current road has to be selected. Based on conditions such as the road ahead, environmental conditions (e.g. rain or mist) and traffic, the human operator (HO) chooses the path to follow. This process is known as *guidance*. Finally, the correct *control* inputs have to be given to the vehicle, such that it tracks the intended path and is stable to external disturbances.

Since vehicles and control tasks have become such a prominent aspect of our daily lives, they are a popular topic of study. There have been ample attempts to describe HO behaviour in technical terms, a field known as *cybernetics*. One of the properties of human behaviour that we currently have limited understanding of, though, is time-variance and adaptability (Mulder et al., 2018). There have been studies into time-variance for compensatory control, the most basic type of control task. But research for the more realistic preview situation is missing.

This thesis aims to fill that gap, by finding a method that can estimate parameters of established preview models in a time-varying fashion. Understanding how human operators combine sensory information, experience and knowledge into their actions does not only add to the state-of-the-art in cybernetics research. It could prove an invaluable source of inspiration for monitoring human performance, as well as assistance and automation (Popovici et al., 2017).

This preliminary thesis report describes the steps taken so far towards the development and testing of this estimation method. The report is structured as follows. First, a literature review of suitable, established preview models, as well as time-varying estimation methods suited for preview tracking was performed in Chapter 2. The subsequent formulation of global and detailed research questions can be found in Chapter 3. Chapter 4 describes how simulation software, capable of simulating realistic, time-varying human control behaviour, was constructed. Thereafter, Chapter 5 explains how the estimation algorithm was implemented such that it could be tested for different simulated runs and experimental data sets. During the preliminary phase of the research, a set of simulations was created to test the potential of promising methods. These *benchmark simulations*, together with the estimation results of the most promising candidate method, can be found in Chapter 6. The most important observations to be taken away from this benchmark are also discussed in that chapter. Finally, Chapter 7 concludes the preliminary phase of this research, and gives an outline for further research.

2 Literature Review

To gain insight in the current state-of-the-art in in the fields of preview tracking and time-varying parameter estimation, a literature survey was conducted. The first subsection is devoted to models that have so far been constructed for preview tracking and more realistic (preview) steering tasks. This is followed by Section 2.2, where time-varying estimation techniques are reviewed that could be used for the case of preview steering.

2.1. Human operator models in preview tracking tasks

This section aims to provide an overview of previous efforts to explain or model human control behaviour, with a focus specifically on preview control. One of the first to recognise preview control as the abstract equivalent of many of our daily activities was Sheridan (1966). In his paper, he develops 3 different model types that *"characterize constrained preview control better than conventional transfer function techniques."*. From there onward, many have made an effort to model human preview control behaviour and current research can often still be categorised as one or a combination of Sheridan's approaches (Van der El et al., 2016). An example is the model proposed by MacAdam (1981) who, continuing on Sheridan's third approach, uses optimal control to predict and continuously update the path that human controllers choose in a lane keeping task.

In general, two approaches to human control research can be recognised. The first line of research initially isolates the sequence from human perception and processing of information towards the resulting action. Subsequently, the *empirical* model structure of a quasi-linear controller is composed based on these findings and clever reasoning. The parameters that are left to be tuned, to distinguish the behavioural differences between participants or test conditions, are then fitted with mathematical optimisation.

A second line of research takes a fundamentally different approach. Instead of starting at the (sub)conscious thought process, traditional black box system identification techniques are applied to derive the transfer function of human control. The input-output relations of the human controller are directly constructed from the mathematical relation between target input signals and the resulting steering action. How humans arrive at this behaviour from their perceptive cues is then attempted to be captured in a meaningful model structure. Again, the human operator is assumed to be *quasi-linear*. This approach is also known as *cybernetics.*

The first subsection looks broadly at how researchers have studied human control. It explains how the human controller was isolated from the complete system. Moreover, it looks at what sets aside preview control from other forms of control and how humans utilise this preview information. Two subsections then continue with the previously mentioned main research lines. Subsection 2.1.2 traces human perception towards constructing empirical control models. This is followed by Subsection 2.1.3, where the models identified with a cybernetic approach are reviewed. Finally, Subsection 2.1.4 examines previous efforts to model the remnant, formed by control behaviour that remains unexplained by the quasi-linear model.

2.1.1. Studying human control

The eventual path traced by a vehicle is the result of a complex process. McRuer and Jex (1967) theorized that is influenced by not only experience, mental and physical state of the driver, but also the vehicle (model) itself and the environment. In order to study control, experiments are therefore typically executed in simulation where the circumstances can be made repeatable for different drivers and test conditions. This holds for example for lighting, weather, traffic and most notably the target path to be tracked. Different pathways have been used depending on the type of experiment or model identification technique. They vary from sequences of real-life segments of road or even existing racing tracks (Sentouh et al., 2009) via artificial tracks with constant curvature corners (Donges, 1978) to much more abstract roads formed by a sum of sines (Van der El et al., 2016). This last variation is particularly convenient for identification in the frequency domain.

Furthermore, to isolate human control more clearly, system dynamics are typically simplified. Especially in relatively abstract scenarios, the system is often reduced to simple gain, single integrator or double integrator dynamics (McRuer and Jex, 1967). More realistic simulations usually employ more elaborate models, for example experiments that involve simulated car driving (Sentouh et al., 2009).

Simulating the environment allows to precisely set the cues available to the human senses, or to limit the degrees of freedom open for control. For a start, experiments are often executed in a fixed-base simulator, removing vestibular cues. Secondly, as Sheridan (1966) already suggested, many historical steering experiments have a *forced* pace, rather than being *self-paced*. This removes throttle as one of the degrees of freedom. The combination of these limitations allows the researcher to look at visual steering, specifically.

A main focal point of research has been to find out how humans use the visual information from preview for the guidance and control parts of steering. This has resulted in the abstract distinction between compensatory, pursuit and preview control tasks, based on the visual information available to the operator (Figure 2.1). In a *compensatory* control task, the only information shown to the operator, is the error between the target signal and the current system state. A *pursuit* tracking task, on the contrary, explicitly shows the target signal for the current time step, as well as the system state. Finally, a *preview* tracking task shows the current system position and the current and future target for some time ahead in the form of a 'spaghetti' track, a winding road or, in a more practical real life case, a cane. Pursuit tracking is in other words simply preview tracking with 0 seconds of preview information.

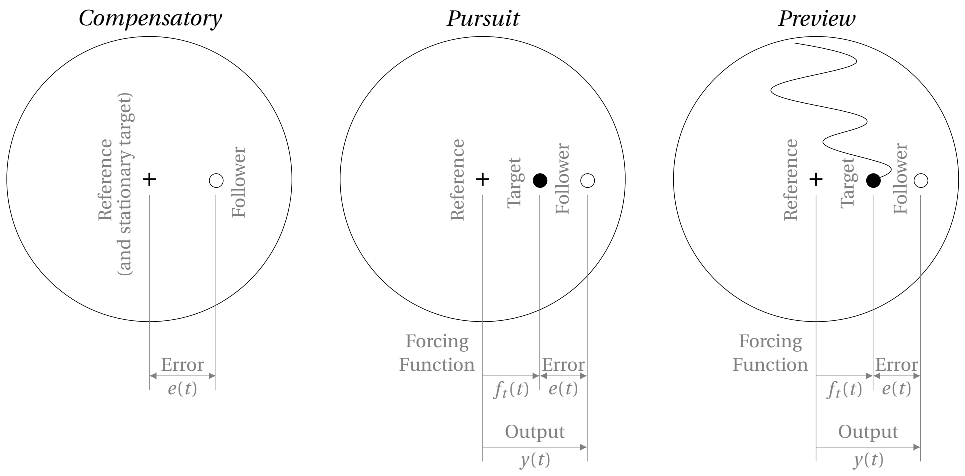


Figure 2.1: Compensatory, pursuit & preview tracking tasks (reproduced & extended from Wasicko et al. (1966))

2.1.2. Two-level driving models

Already early in the development of driver control models, human perception and understanding of the control task at hand have been a logical start point. McRuer et al. (1977) observed that the control task of driving has features of a compensatory, pursuit and precognitive nature. The balance between these features shifts for different maneuvers that make up driving. Compensatory control is used for changes in vehicle dynamics and disturbance rejection. Pursuit and preview steering on the other hand, are essential for path following. And finally, drivers use precognitive behaviour for standard maneuvers such as rapid lane changes. These theories were verified by McRuer et al. with simulated experiments for the different tasks.

As mentioned in the introduction of this chapter, the act of steering a vehicle can alternatively be broken down into the levels of navigation, guidance and control. The first is only involved with global trip preparation and monitoring, and is nowadays mostly taken out of our hands with the arrival of satellite navigation. The latter two, however, are still crucial during manual control of a vehicle. Both these aspects of steering require the immediate and continued attention of a driver, to respond to changing conditions or disturbances.

This shift of attention from (immediate) disturbance rejection to (slightly longer term) path following has lead to the development of so-called two-point models. An early well known model in accordance with this logic, was developed by Donges (1978). By combining the phases of guidance and control (or *"stabilization"* as Donges calls it), he arrives at a quasi-linear controller for his artificial track with a series of 24 constant curvature turns.

A few notable experiments to uncover how humans use visual information for steering have been conducted afterwards. Godthelp (1986) used visual occlusions to show that humans can also successfully steer through corners when their visual feedback is temporarily withdrawn. And in line with Donges' model, Land and Lee (1994) showed that humans do indeed continuously shift their gaze as they are driving. We move our eyes to a tangent point of a curve 1-2 seconds ahead, and keep following this point until the curve has passed. They theorised that this helps us estimate curvature, without any specific reference to absolute distance.
A year later, Land and Horwood (1995) used partial visual occlusion (slits) to demonstrate that drivers can achieve good performance in curve negotiation with two snippets of information. The best performance was obtained with the first strip lying a few meters in front of the vehicle, and the other just under a second further up the road. This supports the idea that we combine anticipatory feed-forward steering for guidance with our lateral position on the road for control.

Following these insights, a few models have been proposed that are nowadays widely established. But how humans precisely extract and use the information from a previewed track is still debated, as outlined in the introduction of Salvucci and Gray (2004). It is argued that that humans are shown to merely use the near and far road information to steer a vehicle, but not what information exactly. Humans are demonstrably bad at inferring road curvature as Donges suggested, and can successfully navigate without enough information for the use of optical flow. The review therefore concludes that, although there is much empirical support for two-point models, *"none of the models explicitly defines perceptually plausible sources of 'near' and 'far' visual information that is used by the driver"*. The proposed model is therefore a PI controller purely based on two viewpoints. Different examples for these cues are given, such as the horizon, a turn's apex or a car ahead. The model is then shown to result in realistic behaviour for some frequent practical scenarios: curve negotiation, corrective steering and lane changing.

Later research has included explicit models of human sensorimotor dynamics and cognition on simulated driving, such as by Sentouh et al. (2009) and Lappi and Mole (2018).

2.1.3. Cybernetic approach

The models presented in the previous section have a structure that relates to the perception and processing of information in human control. However, these empirical models often have physically meaningless parameters that are difficult to interpret. A fundamentally different approach to identifying control behaviour has simultaneously been developed under the name of *cybernetics*. The main approach here is to view the human operator as a black box element in the entire closed-loop system. With precise knowledge of the surrounding dynamics, the operator's transfer function from input to output can be deduced. The goal of the cybernetic approach is to arrive at a comprehensive system description from this transfer function, with a meaningful structure and parameters.

Arguably the most well-known research in the field of human control following these principles is the work from McRuer and Jex (1967). Their research resulted in the crossover model for compensatory tracking, that has become the standard. It forms a basis upon which notable other models are built. The first attempt to model preview tracking using an experimental, cybernetic approach rather than *"empirically"* or *"theoretically"* was by Ito and Ito (1975). They find that preview control can be modeled as an extension to McRuer and Jex's compensatory model, with a *preprocessing* function that optimizes the control action based on the available preview information.

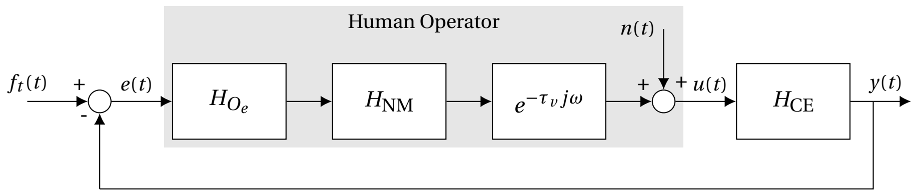


Figure 2.2: Crossover model for compensatory tracking, augmented with neuromuscular limitations (McRuer and Jex, 1967)

Building on this idea, Van der El et al. (2016) have identified a more explicit model for preview tracking (Figure 2.3). Again, the resulting model is shaped as an extension to McRuer and Jex (1967)'s crossover model. However, instead of tracking the actual error between the forcing function and the current system state, compensatory tracking is performed based on some *internally calculated error* ($e^{\star}(t)$). This signal is deduced by the HO by filtering the previewed trajectory using a low-pass preview filter (H_{O_f}) to filtered far-view forcing function $f^{\star}_{t,f}(t)$. It is subsequently compared to the current state $y(t)$ to form $e^{\star}(t)$. A review of the parameters in this model can be found in Section 4.1. The model has been tested in a wide range of experimental conditions, from abstract one-dimensional tracking tasks, via more realistic tracking with different amounts of linear perspective and multiple feedback cues to full simulations of car driving on a realistic road (Van der El, 2018).

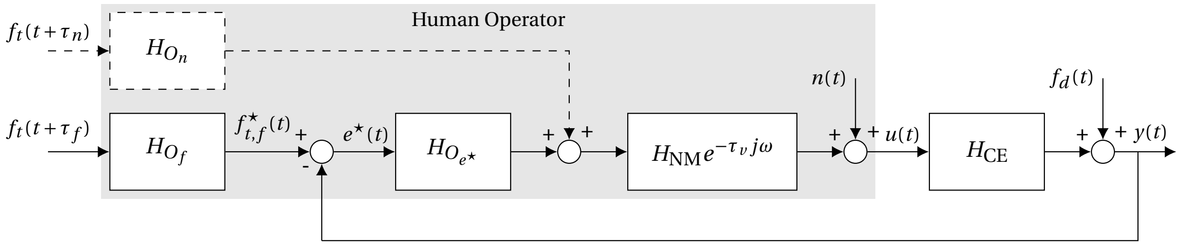


Figure 2.3: Block diagram of a human operator model in preview tracking tasks (Van der El et al., 2016)

Compared to two-level driver models, the cybernetic model identified by Van der El et al. has several major advantages regarding the main goal of this thesis, i.e. online-time-varying parameter estimation:

1. The model structure is simple and intuitive, incorporating preview tracking as an extension to McRuer's crossover model for compensatory tracking.
2. The model is a minimal realisation, meaning that all parameters can be identified uniquely.
3. The parameters in the model are physically interpretable, with well-documented human limits.
4. The model is validated for a wide range of experimental conditions (such as system dynamics, amounts of preview, forcing functions, perspective and task realism) and parameter's sensitivities to changes in experiment conditions have been studied.

2.1.4. Remnant modeling

The quasi-linear models highlighted thus far explain the linear part of the input-output relation of the observed signals. The remainder of the HO input signal that is not covered by the model is known as *remnant*. The remnant is theorised to consist of time-varying behaviour, pure noise and nonlinear behaviour (Van der El et al., 2019a). Despite attempts to explain (part of) this unmodeled behaviour, such as by Popovici et al. (2016), remnant still remains largely unexplained. For the generation of realistic HO signals to test identification methods on, it is however vital to include an accurate remnant.

Two factors are important for quantitative remnant simulations: the remnant model and the remnant power. Levison et al. (1969) conducted a study for compensatory models (such as McRuer's crossover model) with conclusions regarding both these aspects. The research showed that a first order model can give an adequate description of the remnant's colored spectrum (Equation (2.1)). This model only incorporates two additional parameters, namely a remnant gain K_r and either a time constant(T_l) or break frequency ($\omega_{b,r}$). Moreover, the remnant power ratio (Equation (2.2), also referred to as *noise ratio* or *constant of proportionality*) is proportional to (input) signal variance and independent of input parameters or system dynamics.

$$H_r = \frac{K_{r,1}}{1 + T_l j\omega} = \frac{K_{r,2}}{\omega_{b,r} + j\omega} \tag{2.1}$$

$$P_r = \frac{\sigma_r^2}{\sigma_u^2} \tag{2.2}$$

In previous work into time-varying identification of compensatory control, higher order models were also used. Higher order models were tested by Van Grootheest et al. (2018) in combination with ARX model identification. Equation (2.3) shows the similarity with the first order model, where m indicates the order of the model.

$$H_r^m = \frac{K_r}{(1 + T_l j\omega)^m} \tag{2.3}$$

Finally, the third order model of Equation (2.4) was proposed by Zaal et al. (2009) and also used for time-varying ARX identification of compensatory tracking by Plaetinck et al. (2018).

$$H_r = \frac{K_r \omega_r^3}{((j\omega)^2 + 2\zeta_r \omega_r j\omega + \omega_r^2)(\omega_r + j\omega)} \tag{2.4}$$

The remnant power ratio, as referred to above, is the relative amount of remnant, compared to the power of the input signal. The gain K_r, part of all the remnant models presented above, basically provides the knob to tune remnant power. Because the remnant injects into the closed loop system, this gain K_r has to be found iteratively. The remnant signal itself, however, can simply be simulated by passing Gaussian white noise through the remnant filter model.

Recently, remnant for preview tracking specifically was quantitatively reviewed by Van der El et al. (2019a). They concluded that the remnant power does depend on CE dynamics, target bandwidth and tracking task type. However, remnant is shown to be relatively invariant to changes in preview time and display configuration. Moreover, the research demonstrated that Levison et al.'s first order model provides adequate accuracy for pursuit and preview tracking. Also, the break frequency $\omega_{b,r}$ was shown to only depend on CE dynamics. For this research, Levison et al.'s first order model will be used to simulate remnant. The specific values used for the remnant parameters can be found in Table 6.1.

What is thus far unclear, is how the remnant might be affected by the current research. After all, this estimation method is expected to identify time-varying behaviour, which is one of the hypothesised causes of remnant. Part of the remnant could therefore possibly be omitted after time-varying identification, resulting in a difference in remnant model and power. Since time-varying behaviour is only one of multiple contributors to remnant, however, and since the current method is expected to identify only part of this time-varying behaviour, it is deemed fair to assume that existing remnant models will stay valid.

2.2. Online, time-varying parameter estimation in human control

The models describing human control behaviour that were reviewed in the previous chapter all have one important characteristic in common. In order to identify the parameters of the quasi-linear models, they assume constant behaviour of the operator throughout an experiment run. However, humans are known to be self-learning controllers that continuously adapt their behaviour (Mulder et al., 2018; Young, 1969). It is therefore of interest to study this adaptive side of human control. Several methods, applied to models differing in shape and complexity, have been investigated in the past with varying success.

This section presents an overview of the most notable advances that have been made in the field of time-varying HO identification. Subsection 2.2.1 lists the studies using different types of Kalman Filters. Subsection 2.2.3 subsequently reviews other methods that have been studied. Subsection 2.2.4 finally presents a trade-off of the reviewed methods for the case of preview tasks.

2.2.1. Time varying identification with Kalman Filters

The linear Kalman Filter is an optimal Linear Quadratic Estimator (LQE) that can be applied to linear systems. It was derived by Kalman (1960). Many alterations to the linear Kalman Filter exist, to make it suitable for all kinds of nonlinear applications. At the time that Kalman Filters were still a relative novelty, Schiess and Roland (1975) already tried an Extended Kalman Filter (EKF) on identification of McRuer and Jex's crossover model for compensatory tracking. Success was limited, as it proved difficult to estimate parameters for data with realistic amounts of remnant.

Different kinds of Kalman Filters have since been prevalent choices for the identification of time-varying human operators. Boer and Kenyon (1998) revisited the EKF, but in order to obtain satisfactory results, several elements were added. The algorithm is augmented with a recursive delay identifier (RDI) with bicubic interpolation improve success on concurrent identification of linear parameters and delay time. Moreover, to improve parameter estimates, a smoother was applied to the filter output.

Mandal and Gu (2016) and Rojer et al. (2019) continued the exploration of Kalman Filters with the successful application of the Unscented KF (UKF) to flight simulations and compensatory tracking.

The application of Kalman filters in human control identification is far from limited to more abstract tracking task experiments. You et al. (2017) have for example applied several filter types on extensive field test driving experiments. Both the EKF as well as the UKF have been applied, each of them in a *joint* filter as well as in a split *dual* filter configuration. More information on dual estimation can be found in Subsection 2.2.2. The driver model that was used, is the two-point model with sensorimotor dynamics by Sentouh et al. (2009). This model contains 12 identifiable parameters. The two-point model structure was reviewed earlier in this chapter, in Subsection 2.1.2. With their effort, they successfully identified gradual variations in driver dynamics over time. Moreover, they could compare experienced to novice drivers.

2.2.2. The Dual Extended Kalman Filter

Popovici et al. (2017) applied the Dual Extended Kalman Filter (DEKF) to both simulated and experimental data with notable success. The Dual Extended Kalman Filter was conceived by Nelson and Stear (1976). The principle of dual filtering is that the (fast-varying) system states and (relatively slow-varying or constant) HO parameters are each placed in a separate, dedicated filter. These filters run concurrently, allowing for simultaneous estimation of states and parameters. The main advantage of this technique over *joint* estimation, though, is that the size of the covariance matrices is reduced. This enhances the likelihood of convergence.

Applied to compensatory tracking, the system behind the state filter is the crossover model (Figure 2.2). The model takes tracking error $e(t)$ as external input signal. It is transformed to a state-space equation of controllable canonical form. In order to include the exponential time delay in this format, a Padé approximation of third order is used (for more information, see Subsection 4.1.3). The total set of parameters is split up in a state vector and a parameter vector. The first consists of five canonical states and two HO equalisation gains: $\boldsymbol{x_s} = [x_{s,1}, \ldots, x_{s,5}, K_p, K_v]^T$. The parameter vector then contains the neuromuscular parameters and time delay: $\boldsymbol{\theta} = [\omega_n, \zeta_n, \tau_v]^T$. The filter works as depicted in Figure 2.4.

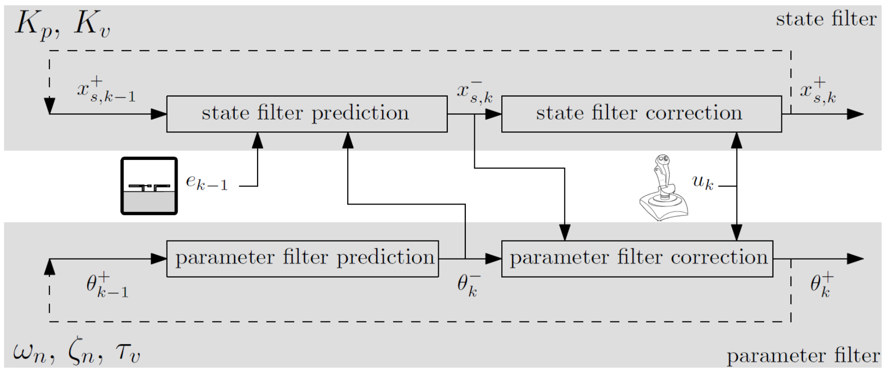


Figure 2.4: DEKF for compensatory tracking (from Popovici et al. (2017))

Together with Gaussian process noise $\boldsymbol{w_s}(t)$, the state equation of the state filter can be expressed as Equation (2.5). The full nonlinear function f can be found in Appendix A of Popovici et al. (2017). Equation (2.6) describes the state equation of the parameter filter. The parameters are generally assumed constant and only driven by a *random walk*. Thus, their variance is completely captured by (Gaussian) process noise $\boldsymbol{w_p}(t)$. Finally, the output equation for both filters is given in Equation (2.7), where $v(t)$ indicates the Gaussian measurement noise of the system. Again, the full function g can be found in Popovici et al. (2017), Appendix A.

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}}(t) = f(\boldsymbol{x_s}(t), e(t), \boldsymbol{\theta}(t)) + \boldsymbol{w_s}(t) \tag{2.5}$$

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{w_p}(t) \tag{2.6}$$

$$u(t) = g(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t)) + v(t) \tag{2.7}$$

Any type of Kalman Filter uses several vectors and matrices in its (discrete time) filtering equations. For the DEKF applied to compensatory tracking, this means that the concurrent state and parameter filters have their own versions of the following vectors and matrices at each discrete time step k:

- 7×1 state vector $\boldsymbol{x_{s,k}} = [x_{s,1},\dots,x_{s,5},K_p,K_v]^T$ and 3×1 parameter vector $\boldsymbol{\theta_k} = [\omega_n,\zeta_n,\tau_v]^T$
- 7×7 state covariance matrix $P_{s,k}$ and 3×3 parameter covariance matrix $P_{p,k}$
- 7×7 state process noise cov. matrix $Q_{s,k}$ of $\boldsymbol{\omega_s}$ and 3×3 parameter process noise cov. matrix Q_p of $\boldsymbol{\omega_p}$
- 1×1 measurement noise covariance R_k of v (shared by both filters)

For systems that are expected to have approximately constant Gaussian process and measurement noise, matrices Q and R are typically kept constant over time. For the application of human identification, however, this is not necessarily the case. In their implementation of the filter, Popovici et al. have therefore built time-variance into matrices $Q_{s,k}$ and R_k. The state process noise of final canonical state x_5 is taken to be proportional to the variance of the tracking error of the last five seconds with scaling factor q^2 (Equation (2.8)). For all other states and parameters, the process noise is kept constant. Similarly, measurement noise covariance R_k is taken proportional to the variance of the HO input signal u of the last 5 seconds (Equation (2.9)).

$$Q_{s,k} = diag\left(\begin{bmatrix}0 & 0 & 0 & q^2\sigma^2_{e(k-5/\Delta t:k)} & K_{p,0} & K_{v,0}\end{bmatrix}\right) \tag{2.8}$$

$$R_k = r^2\sigma^2_{u(k-5/\Delta t:k)} \tag{2.9}$$

With the DEKF as presented, simultaneous estimation of all HO parameters proved to be possible for both simulated as well as experimental tracking data. It was noted, though, that converge proved challenging and highly dependent on the initialisation parameters of the filter.

2.2.3. Other identification methods

Also aside from Kalman Filters, various methods have been tested on time-varying human operator data. One that has been given attention recently is identification based on recursive ARX models. Van Grootheest et al. (2018) finetuned the implementation and Plaetinck et al. (2018) went on to apply these methods experimentally. The method was shown to detect and estimate time-varying behaviour with reasonable success. ARX models do however have a fixed model structure, meaning that an additional step is necessary to transform the acquired parameter values back to the ones from a corresponding cybernetic model. Moreover, they were found to be unfit for estimating human time delay (Plaetinck et al., 2018).

Secondly, Maximum Likelihood Estimation (MLE) was used by Zaal et al. (2009) to identify parameter variations in multi-axis compensatory tracking. This method was later also applied in a dual-axis control experiment with commercial pilot operators by Zaal (2016). In similar but single-axis compensatory tracking simulations, Duarte et al. (2017) tested Predictor-Based Subspace Identification on Linear Parameter Varying models. Both of these methods were successful, but do require *a priori* assumed scheduling functions for the type and shape of time-variance.

Another way to employ LPV models in human control, was explored by Pronker et al. (2017). In this research, a set of LTI models was constructed for different experiment conditions. Rather than researching parameter variations over time, the research resulted in an interpolated LPV model set as a function of driver grip force. The LPV model was composed of a set of gains, first order, and second order subsystems in observable form. The interpolation was done with polynomial interpolation of order 2, yielding fit results of well over 90% at the tested conditions.

Each individual LTI model is first fitted using spectral analysis techniques. Applying this method to the *time*-varying case is therefore only possible for complete passages of the forcing function(s). This means that, compared to the original experimental setup of for example Van der El et al. (2016), the forcing functions should be shortened considerably and run longer than its fundamental period. After the first fundamental period, enough data is available each time step to fit an LTI model. These LTI models can subsequently be interpolated with polynomials to find a time-varying LPV progression.

Alternatively, stepping away from spectral analysis would drop the long experiment run-in. However, the identification of the LTI models for interpolation then requires one of the other (recursive) algorithms evaluated above. The polynomial interpolation would then serve the purpose of a smoother.

Finally, wavelets were tested by Zaal and Sweet (2011). Although this method showed quick convergence for idealised tracking behaviour, convergence declined for simulations with significant remnant power.

2.2.4. Method tradeoff

This subsection compares the identification methods presented above, to select the most promising candidate. In order to choose a method suitable for preview tracking tasks, four trade-off criteria are used:

- It must be possible to estimate delay parameters. Time delays introduce major non-linearities in the model and are therefore difficult to identify. In compensatory tracking, the delay parameter has had particular attention (Boer and Kenyon, 1998; Plaetinck et al., 2018). The preview model of Van der El et al. (2016) does not only contain a delay, but also at least one preview time (i.e. a 'negative' delay).
- The identification method must be able to converge under (significant amounts of) remnant noise.
- Identification should ideally be possible online. This means that the computations necessary for convergence can be executed within a real-time time frame. Moreover, the identification should not require any a priori assumptions on the variations to be detected. Finally, the method must be able to provide a parameter update every time step, limiting the usability of spectral methods.
- Ideally, the method can estimate parameters in the Van der El model directly, without the need of conversion afterwards.

Table 2.1 summarizes the performance of each of the algorithms in the previous subsections, based on these criteria. From this trade-off, it becomes clear that Kalman Filters are a promising candidate for time-varying parameter estimation in preview tracking tasks. Because of its good converge properties, the DEKF is the filter that will be given a first attempt.

Table 2.1: Trade-off of different time-varying identification methods

ID method	**Estimates time delays**	**Converges with remnant**	**Online / no scheduling**	**Direct estimation of parameters**
UKF (Rojer et al., 2019)	🟩 Yes, with Padé approximation	🟨 Convergence depends on initial conditions	🟩	🟩
EKF with RDI (Boer and Kenyon, 1998)	🟩 Delay is estimated as a spline interpolated sample shift	🟨 Convergence depends on initial conditions	🟨 The EKF output is smoothed, causing identification delay	🟩
DEKF (Popovici et al., 2017)	🟩 Yes, with Padé approximation	🟨 Convergence depends on initial conditions	🟩	🟩
Recursive ARX (Plaetinck et al., 2018; Van Grootheest et al., 2018)	🟥 No estimation of delays possible	🟩	🟩	🟥 Parameters calculated from ARX structure
MLE (Zaal, 2016; Zaal et al., 2009)	🟩	🟩	🟥 Only best fit of Assumed Parameter Variation (APV)	🟩
LPV PBSID (Duarte et al., 2017)	🟥 PBSID does not easily estimate a time-varying delay	🟩	🟥 A priori assumed scheduling function required	🟥 Requires parameter retrieval after identification
Polynomial LPV interpolation (Pronker et al., 2017)	🟩	🟩	🟨 After 1 FoFu period, or only as smoother	🟩
Wavelets (Zaal and Sweet, 2011)	🟩	🟥 Very sensitive to remnant noise	🟩	🟥 Parameters obtained from frequency response

3 Research Questions & Approach

Preview steering is a very frequent type of control task in the daily life of many. However, from the literature survey summarised in the previous chapters, it becomes clear that research into time-varying identification of preview tracking behaviour is very limited. This thesis aims to change that, by developing a method that can estimate the parameters governing preview tracking in a time-varying manner.

From the literature survey, a suitable model that describes preview tracking for different scenarios and that fits in with other existing literature, is the model by Van der El et al. (2016). This model is a realisation of minimal order, meaning that the parameters can be identified uniquely. Moreover, the parameters in the model are physically interpretable, with well documented human limits for different conditions. This means that there is enough knowledge available to tune, restrict or refine the algorithm(s) used for estimation.

The first promising candidate for estimation is the Dual Extended Kalman Filter (DEKF) as applied to compensatory tracking by Popovici et al. (2017). This algorithm has shown to have the capabilities of estimating delay parameters, which form a vital part of preview tracking. Moreover, the method has good convergence properties, being successfully applied to real human data. The research objective thus becomes:

The research goal is to contribute to the identification and understanding of time-varying, adaptive human control behaviour in preview tracking tasks by applying the Dual Extended Kalman Filter to simulated and experimentally acquired data for time-varying and online parameter identification of Van der El's quasi-linear human operator model.

3.1. Research questions

This objective leads to the formulation of a number of research questions. The main research question is presented below, followed by a number of more detailed sub-questions that are categorised, leading the way to some intermediate milestones in the research:

Can the parameters of Van der El's model for human control behaviour in a preview tracking task be estimated in a time-varying, online manner using a Dual Extended Kalman Filter, including the preview time and time delay parameters?

1. Regarding the *simulation* of a *time-varying human operator*:
   (a) How can a simulation be set up, such that it can produce realistic, artificial, time-varying human preview control behaviour?
   (b) How should remnant noise be included in the simulation?
   (c) How can the simulation be validated, to guarantee that it produces correct results?

2. Regarding *convergence & sensitivity* of the parameter estimation
   (a) How many and which parameters of preview tracking can be estimated simultaneously?
   (b) How can the preview time parameter be estimated, being a negative time delay?
   (c) Is it possible to estimate the preview time parameter using a DEKF?
   (d) Is it possible to estimate both preview time as well as time delay?
   (e) How sensitive is the algorithm to changes in initial conditions?

3. Regarding *performance* of the parameter estimation
    (a) What metric can be used to measure the accuracy of the parameter estimation?
    (b) Is it possible to estimate constant behaviour accurately?
    (c) How accurately can time-varying behaviour be estimated?
    (d) What can be done to improve the parameter fit of the algorithm?
    (e) Can the most accurate method be applied online? Both in terms of computational structure as well as computational speed.

4. Regarding *experimental validation* of the parameter estimation method
    (a) What type of experiment should provide the data to find out whether the algorithm works on actual human tracking data?
    (b) Does the algorithm work on experimental tracking data?
    (c) Does human tracking show time-varying behaviour for a constant task?
    (d) Can behavioural changes be identified for variable tracking tasks using the algorithm?

3.2. Research outline

In order to answer these questions sequentially, a research outline has been set up, consisting of 4 steps:

1. Simulate time-varying human tracking behaviour
2. Implement, test and refine suitable estimation methods
3. Improve performance and analyse sensitivity of successful method(s)
4. Apply thoroughly tested method(s) to experimental data

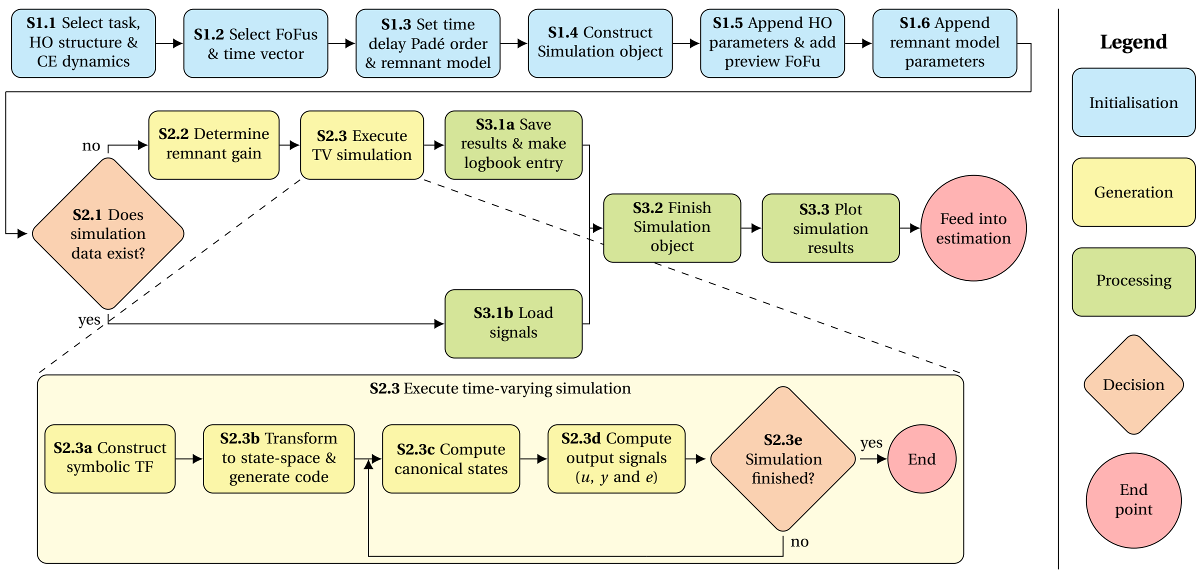
As mentioned in the introduction of this report, Chapter 4 covers the setup of time-varying HO simulations and Chapter 5 subsequently details the implementation of the DEKF. Subsequently, the preliminary research was carried out. A few standard sets of simulation data were generated to get a first impression of the estimation performance and to initially tune its settings. These *benchmark simulations* contain both constant as well as time-varying behaviour. The results of some initial estimation runs on the benchmark simulations can be found in Chapter 6. The most important points to take from these experiments for the continuation of the research follow in that same chapter. Finally, plans are formulated for tuning the filter to improve the performance. This is done using more elaborate simulations that stretch the boundaries of the parameter envelope and introduce additional forcing functions and remnant realisations. Also, the scrutinised method will lastly be tested on actual experimental control data. The execution of these last steps is outside the scope of the preliminary thesis and will be investigated later during this research. A more detailed discussion on how to approach future steps is given in Chapter 7.

4

Simulation of Human Tracking Behaviour

The execution of a human-in-the-loop experiment is a time-consuming process, taken up by the design of a proper experiment, finding the participants, taking the measurements and analysing the resulting data. As reviewed in Section 2.1, previous research has resulted in many mathematical models that approximate human behaviour in a preview tracking task. Moreover the models allow for testing of estimation algorithms with known input parameters and outside of the boundaries of normal human behaviour. Therefore, the DEKF as described in Subsection 2.2.2 was first applied to sets of simulated test data.

The procedure to simulate human behaviour was implemented in MATLAB and is divided in 3 stages. A schematic overview of the entire flow for the HO simulation is depicted in Figure 4.1. During the first, *initialization* stage (in blue), the settings for the simulation, the tracking task and HO model are selected. This is followed by the *generation* stage (in yellow), in which the simulated HO data is obtained. Finally, the simulation is ended in the *processing* stage (in green), where the data is stored and visualised if requested. The following 3 subsections will describe the operations behind these phases in more detail. The final section of this chapter explains how the simulations were verified. The entire simulation and estimation code is set up in an Object Oriented coding structure.



S1.1 Select task, HO structure & CE dynamics
S1.2 Select FoFus & time vector
S1.3 Set time delay Padé order & remnant model
S1.4 Construct Simulation object
S1.5 Append HO parameters & add preview FoFu
S1.6 Append remnant model parameters
no
S2.2 Determine remnant gain
S2.3 Execute TV simulation
S3.1a Save results & make logbook entry
S2.1 Does simulation data exist?
S3.2 Finish Simulation object
S3.3 Plot simulation results
Feed into estimation
yes
S3.1b Load signals
S2.3 Execute time-varying simulation
S2.3a Construct symbolic TF
S2.3b Transform to state-space & generate code
S2.3c Compute canonical states
S2.3d Compute output signals (u, y and e)
S2.3e Simulation finished?
yes
End
no
Legend
Initialisation
Generation
Processing
Decision
End point

Figure 4.1: Flowchart of Human Operator simulations

4.1. Simulation initialization

In the first stage of simulation, the settings for different components of the process are selected and the necessary preparations are done.

4.1.1. Initialisation steps

To start with, the type of task and HO block diagram structure (compensatory or preview, with or without near view response) and the controlled element (single or double integrator) are selected in step S1.1. The specific CE dynamics are taken in accordance with the original research source of the forcing function, and can be found in Table 6.1.

Subsequently, the forcing function is selected in step S1.2, together with the time step and the experiment measurement time window. The forcing functions (both target and disturbance) all consist of a sum of sines:

$$f(t) = \sum_{i=1}^{N_f} A_i \sin \omega_i t + \phi_i \tag{4.1}$$

These target signals are *pseudorandom*, meaning that they appear random to the operator. However, classical (spectral) system identification techniques can be used on the signals in other to retrieve the operator's dynamics and transfer function (Van der El, 2018, p.31). During the preliminary phase, the broadband forcing functions from the bandwidth experiment of Van der El et al. (2019b) were used. Five equivalent realisations were available for the target, all paired with the same realisation for the disturbance signal. The signal also defines the run time of the experiment, being the inverse of the fundamental frequency of the multi-sine signals. An additional setting, though is the run-in time. This is additional time added before the actual simulation measurement time, used to resolve the initial transient response of the HO simulation.

Step S1.3 sets the order for the effective time delay (τ_v). The time delay is modeled with a Padé approximation, described in more detail in Subsection 4.1.3. Subsequently, a specific class object for the simulation is created in step S1.4. By building the code in an Object Oriented structure, all settings of the simulations (and later also estimations) are available at any moment, making analysis easier and more structured.

Step S1.5 in the initialization stage is the selection of HO parameters. The full set of parameters can be viewed in Figure 4.2, a more detailed version of van der El et al.'s model that was explained in Subsection 2.1.3. There are at most four major blocks with parameters, depending on the model selection in step S1.1. For compensatory tracking tasks, only the two blocks in the closed loop are used. The first is the *HO equalisation* block, with visual gain K_p and lead gain K_v (=$K_p T_{L,e}$, with $T_{L,e}$ the lead time constant). The second describes the *HO limitations*, with neuromuscular frequency ω_n, neuromuscular damping ζ_n and visual time delay τ_v.

For preview tracking with only the far view response, the lower left *far view* block is added. This introduces three additional parameters: far view gain K_f, far view break frequency $\omega_{b,f}$ and far view preview time τ_f. If *near view* is included in the model, the near view gain K_n and near view preview time τ_n form the final two of in total ten model parameters. These parameters are defined all as time series, to accommodate simulation of time-varying human operators.

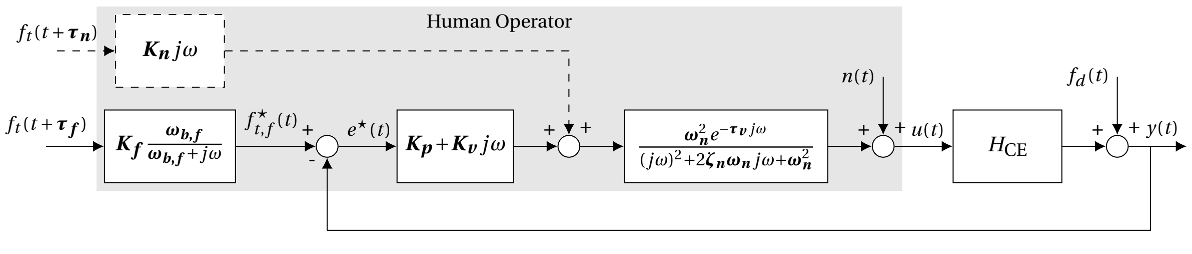


Figure 4.2: Block diagram with HO parameters in preview, pursuit and compensatory tracking tasks (Van der El et al., 2016)

Finally, in step S1.6, the remnant parameters are set. As outlined in Subsection 2.1.4, a first order model, fed by Gaussian white noise is used to model the remnant. Break frequency $\omega_{b,r}$ will be set depending on CE dynamics, in accordance with (Van der El et al., 2019a). The remnant power differs per simulation and the gain is found through an iterative process, based on this remnant power. A more elaborate explanation of this iterative procedure is given in Section 4.2. An overview of the settings used in each simulation can be found in Table 6.1.

4.1.2. HO parameter time-variance

Different ways to make the parameters time varying have are implemented. To simulate gradual adaptation of a parameter in time, an exponential sigmoid function was used. This is in line with earlier research (Duarte et al., 2017; Plaetinck et al., 2018; Zaal, 2016) This sigmoid function depends on 4 parameters: the start value, the final value after transition, the point of inflection of the transition (midpoint, T_{inflect}) and the duration spanning 99% of the transition ($\Delta T_{\text{inflect}}$). This sigmoid function has the shape of Equation (4.2). Note that this formula is slightly different to the one used in previous research (Duarte et al., 2017; Plaetinck et al., 2018; Zaal, 2016). The exponent is rewritten to explicitly reveal the 99% transition time, rather than only the more abstract maximum slope G. The full derivation of this formula, as well as the relation between $\Delta T_{inflect}$ and G, is given in Appendix A.

$$y_{\text{par}} = y_{\text{start}} + \frac{y_{\text{end}} - y_{\text{start}}}{1 + \mathrm{e}^{-\frac{2\ln 199}{\Delta T_{\text{inflect}}}(t - T_{\text{inflect}})}} \tag{4.2}$$

Secondly, parameters can be made time-varying with a sinusoidal shape, as was done by Popovici et al. (2017). They depend on the mean value (μ), amplitude (A), period (T, in seconds) and phase (ϕ, in seconds) according to:

$$y_{\text{par}} = \mu + A \cdot \sin\left(2\pi \frac{t - \phi}{T}\right) \tag{4.3}$$

4.1.3. Padé approximation

The simulations and estimations in this research are executed in the time domain to allow for scheduled time-variance of parameters. This requires a (non-linear) state-space formulation of the system, to which an ODE solver can be applied.

Unfortunately, the exponential formulation of a time delay only exists in the frequency domain. In order to express this delay in the time domain, a Padé approximation is used. With the approximation, the time delay is written as a (fractional) transfer function, that can be transformed to a state space representation.

The Padé approximation, or Padé approximant, is a rational power series function approximator. In general, the approximant P of order m/n (or if $m = n$, simply of order m) has the shape:

$$P_{m/n} = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{k=1}^{n} b_k x^k} \tag{4.4}$$

With equal orders of the numerator and denominator, the Padé approximation is specifically suitable to model time delays as rational transfer functions. Firstly because the exponential function's power series representation is well suited for the Padé form. An secondly, because the amount of poles is equal to the amount of zeros. This prevents the transfer function from becoming improper ($n_{\text{zeros}} > n_{\text{poles}}$), which is an issue you do have with the normal power series form of the exponential delay. An expression for exponential time delay transfer functions, modeled as Padé approximation of order m, was given by Silva et al. (2001):

$$H_\tau = e^{-\tau s} \quad \rightarrow \quad P_{\tau,m} = \frac{\sum_{j=0}^{m} \frac{(2r-j)!}{j!(r-j)!}(-\tau s)^j}{\sum_{k=0}^{m} \frac{(2r-k)!}{k!(r-k)!}(\tau s)^k} \tag{4.5}$$

As is clear from its name, the Padé approximation can never replicate the full effect of a time delay. How good of an approximation it is, is visible by comparing Bode diagrams of both transfer functions. Where the exponential delay drops off in phase with increasing frequency, the approximant levels off at some point. With decreasing delay or with increasing order, the approximation will follow this drop for longer.

4.2. HO data generation

When all the settings for the simulation are applied, it is time to obtain the data. The first step S2.1 in the *generation* phase, is checking whether this specific simulation has already been performed. If this is the case, the data is loaded (step S3.1b). Otherwise the remnant gain is found (step S2.2) and the simulation is performed to generate the HO steering input (u) and resulting system output (y) and error (e) signals (step S2.3). These steps are clarified in more detail in Subsection 4.2.1. Subsection 4.2.2 explains how the transformation from block diagram to state-space system (S2.3b) was performed.

4.2.1. Executing the simulation

As mentioned before, the remnant gain is found through iteratively simulating the system in an optimization loop. By adjusting the remnant gain each iteration, and comparing the resulting remnant power to the required power, the required gain is found. Executing a time-varying simulation takes a considerable amount of time, so finding the gain with the full simulation iteratively is not feasible.

Instead, the gain selection is done with linear (`lsim`) simulations. If the system has any time-varying HO parameters, `lsim` is not directly possible. The time-varying parameters are in that case approximated. For sigmoidal or sinusoidal parameters, the mean value is used to determine the required remnant gain. After the actual time-varying simulation, the remnant power is checked once again. If it deviates too much (more than 5%) from the desired power ratio, the gain is adjusted and the simulation is repeated.

At the heart of the time-varying simulations is the crossover model for compensatory tracking (McRuer and Jex, 1967). For preview tracking, the model identified by Van der El et al. (2016), depicted in Figure 4.2, is used (see also Subsection 2.1.3). This block diagram is transformed into a set of symbolic transfer functions (S2.3a) (multiple since we have a MIMO system).

These transfer functions are then multiplied out and subsequently converted into a state-space system in controllable canonical form. Converting the MIMO transfer functions into a single state-space system is not directly trivial. Subsection 4.2.2 elaborates on how this step was performed. The state and output equations are each written to automatically generated MATLAB function files (S2.3b). Note that the output equation produces two different signals, namely the HO stick input u and the CE output y.

With these functions prepared, the simulation loop is started. For each time step, the HO parameters are extracted from their time series. Subsequently, the MATLAB function of the state equation is evaluated for one time step using `ode54` in step S2.3c. The inputs to this function are the forcing function, the old state and the HO parameters. The output equation is then used to convert the updated state to the u and y signal values (step S2.3d). Steps S2.3c and S2.3d are repeated for each time step, until the total simulation is completed. Finally, the error signal $e = y - f_t$ is computed, and the results are appended to the Simulation object.

The result is a discrete time simulation. Time-varying behaviour is simulated by means of a *Linear Parameter Varying* model (LPV). The block diagram of the model remains unchanged compared to the linear structure originally identified by Van der El et al. (2016). Scheduled time-variance in the (originally constant) parameters of the model makes the HO time-varying.

4.2.2. Constructing a MIMO state-space system

An important step in the simulation is the conversion from a set of transfer functions to a single MIMO state-space system. For a single (SISO) transfer function, a simple conversion can be applied to *controllable canonical* form. The transformation from Equation (4.6) to the system of Equation (4.7) illustrates this conversion for a third order, strictly proper system. As shown, the denominator of the original transfer function determines the dynamics (A-matrix) of the state-space, while the numerator appears in the C matrix. To illustrate the newly formed canonical state-space, the block diagram is also included in Figure 4.3.

$$\frac{Y(s)}{U(s)} = \frac{b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0} \tag{4.6}$$

$$\begin{aligned}\begin{bmatrix}\dot{x_1}\\ \dot{x_2}\\ \dot{x_3}\end{bmatrix} &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -a_0 & -a_1 & -a_2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} u\\ y &= \begin{bmatrix}b0 & b1 & b2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} + \begin{bmatrix}0\end{bmatrix} u\end{aligned} \tag{4.7}$$

Now consider the slightly more complicated example of Figure 4.4 (compensatory tracking). As we can see, all the inputs feed into the same closed loop system. Moreover, the outputs are also directly taken from this closed loop. With 2 inputs and 2 outputs, this system has 4 transfer functions each with the structure $\frac{OL}{1+CL}$. So for example, the transfer function from input 2 ($n(t)$) to output 2 ($y(t)$) looks like:

$$H_{2,2} = \frac{H_{\mathrm{CE}}}{1 + H_{\mathrm{HO}} H_{\mathrm{CE}}} \tag{4.8}$$

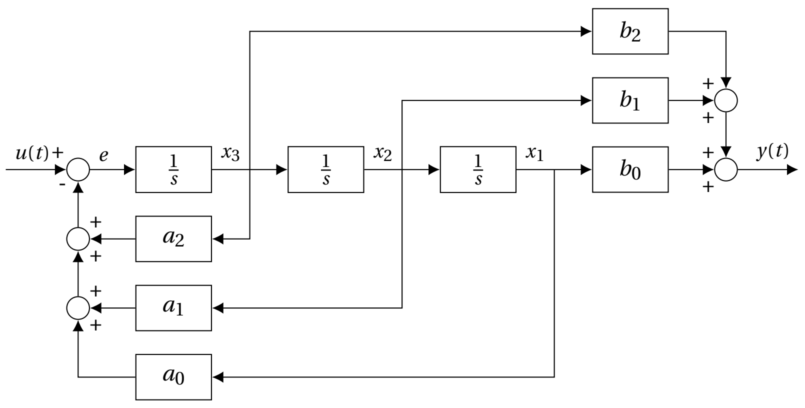


Figure 4.3: Example block diagram of a strictly proper, third order controllable canonical form (Equation (4.7))

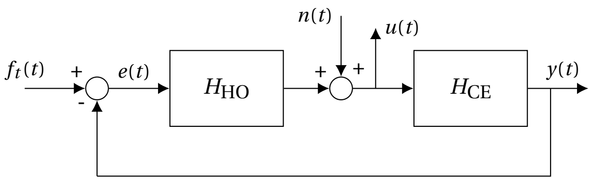


Figure 4.4: Block diagram of a compensatory tracking MIMO system

Looking more closely, one can note that the denominator will be the same for each of the 4 transfer functions, since the closed loop remains identical. Therefore, the A matrix that can be used for the resulting state-space system can be kept identical to the A matrix one would get for a single transfer function. One can also realise that the different inputs can simply be added as different columns in the B matrix. This results in a superposition of responses to each of the two individual inputs (Equation (4.9)).

$$\dot{\boldsymbol{x}} = \left[\boxed{\begin{matrix}\cdot & \cdot & \cdot \\ \cdot & A_{\mathrm{CL}} & \cdot \\ \cdot & \cdot & \cdot\end{matrix}}\right]\boldsymbol{x} + \left[\begin{array}{cc} B_1 & B_2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1 & 1 \end{array}\right]\begin{bmatrix} f_t \\ n \end{bmatrix} \tag{4.9}$$

Next we have to look at the output equation. Since all the internal canonical states ($\boldsymbol{x}$) are equal (identical A-matrices), and since we are dealing with linear models, the output equations originating from the same inputs can simply be added together according to the principle of superposition. Also, for each of the two outputs, the superimposed equations can simply be stacked underneath each other, resulting in Equation (4.10). Since the transfer from n to u is a direct feedthrough, this transfer function adds a nonzero element in the D-matrix.

$$\begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} \boxed{\cdot \quad C_{1,1}+C_{2,1} \quad \cdot} \\ \boxed{\cdot \quad C_{1,2}+C_{2,2} \quad \cdot} \end{bmatrix}\boldsymbol{x} + \left[\begin{array}{cc} D_{1,1} & D_{2,1} \\ 0 & 1 \\ 0 & 0 \\ D_{1,2} & D_{2,2} \end{array}\right]\begin{bmatrix} f_t \\ n \end{bmatrix} \tag{4.10}$$

Finally, we have a look at the block diagram for preview tracking (Figure 4.5). Note that the near view response, shown in Figure 4.2, is taken out for now, but can be added according to the same logic if required. The main difference with the previous example, is that not all inputs feed into the same closed loop anymore. The first input (f_t) has dynamics before entering the loop. This means that the closed loop part of the transfer function is not equal to the other transfer functions, and thus the internal canonical states will be different.

There are two ways to overcome this issue. The first is simpler, resulting in a larger, non-minimal realisation of the system. The other is more clever and does result in a minimal realisation. Since a minimal realisation is not required for the simulation, the easier method is opted here. For the estimation, however, it is necessary to have a minimal realisation of the HO system, in order to lower the amount of states to be estimated. Making a minimal realisation HO model is therefore described in Subsection 5.2.1.

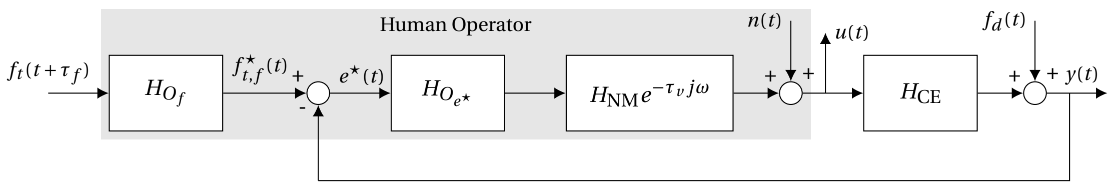


Figure 4.5: Block diagram of a preview tracking MIMO system (from Van der El et al. (2016))

The easiest way to end up with one state-space system, in this case, is by creating the state-space for each input individually and appending them together. In the process of appending, a distinction is made between transfer function sets for the different inputs and outputs. The equations from one input to the different outputs do end up with the same canonical states ($\boldsymbol{x}$), because all the outputs are directly connected to the same closed loop. Put differently, the denominator $(1+CL)$ is always identical from one input to different outputs. Thus the A-matrix for each of the transfer functions of a given input are equal. The B-matrices are equal anyway, because of the nature of the controllable canonical form. The resulting state-space system can be found in Equations (4.11) and (4.12).

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boxed{\begin{matrix}\cdot&\cdot&\cdot\\ \cdot&A_1&\cdot\\ \cdot&\cdot&\cdot\end{matrix}} & & \\ & \boxed{\begin{matrix}\cdot&\cdot&\cdot\\ \cdot&A_2&\cdot\\ \cdot&\cdot&\cdot\end{matrix}} & \\ & & \boxed{\begin{matrix}\cdot&\cdot&\cdot\\ \cdot&A_3&\cdot\\ \cdot&\cdot&\cdot\end{matrix}} \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} \overset{B_1}{\boxed{\begin{matrix}0\\ \cdot\\ 1\end{matrix}}} & & \\ & \overset{B_2}{\boxed{\begin{matrix}0\\ \cdot\\ 1\end{matrix}}} & \\ & & \overset{B_3}{\boxed{\begin{matrix}0\\ \cdot\\ 1\end{matrix}}} \end{bmatrix}\begin{bmatrix}f_t\\ n\\ f_d\end{bmatrix} \tag{4.11}$$

$$\begin{bmatrix}u\\ y\end{bmatrix} = \begin{bmatrix} \boxed{\cdot\ C_{1,1}\ \cdot} & \boxed{\cdot\ C_{2,1}\ \cdot} & \boxed{\cdot\ C_{3,1}\ \cdot} \\ \boxed{\cdot\ C_{1,2}\ \cdot} & \boxed{\cdot\ C_{2,2}\ \cdot} & \boxed{\cdot\ C_{3,2}\ \cdot} \end{bmatrix}\boldsymbol{x} + \begin{array}{ccc} D_{1,1} & D_{2,1} & D_{3,1} \\ \boxed{0} & \boxed{1} & \boxed{0} \\ \boxed{0} & \boxed{0} & \boxed{1} \\ D_{1,2} & D_{2,2} & D_{3,2} \end{array}\begin{bmatrix}f_t\\ n\\ f_d\end{bmatrix} \tag{4.12}$$

4.3. Processing

After the time traces are generated, the results can be put to use in the *processing* stage. Newly generated time traces are saved or old data is loaded in step S3.1. When saving the data, the properties of the simulation are added to the SimLogbook. The logbook is a MATLAB `struct` object, which can be printed as a CSV file. It summarizes the setup information of all simulations that have already been generated for easy reference. The Simulation object is now completed. Optionally, the simulation results can be visualised in multiple formats (i.e. time domain and frequency domain plots). The signals can now serve as the input to an Estimation.

4.4. Code verification

The final section of this simulation chapter explains how the simulation code is verified. These verification methods are mostly a byproduct of early, unsuccessful simulation attempts. A detailed breakdown about where these attempts went wrong, can be found in Appendix B. In general, the verification is done in two distinct, independent ways:

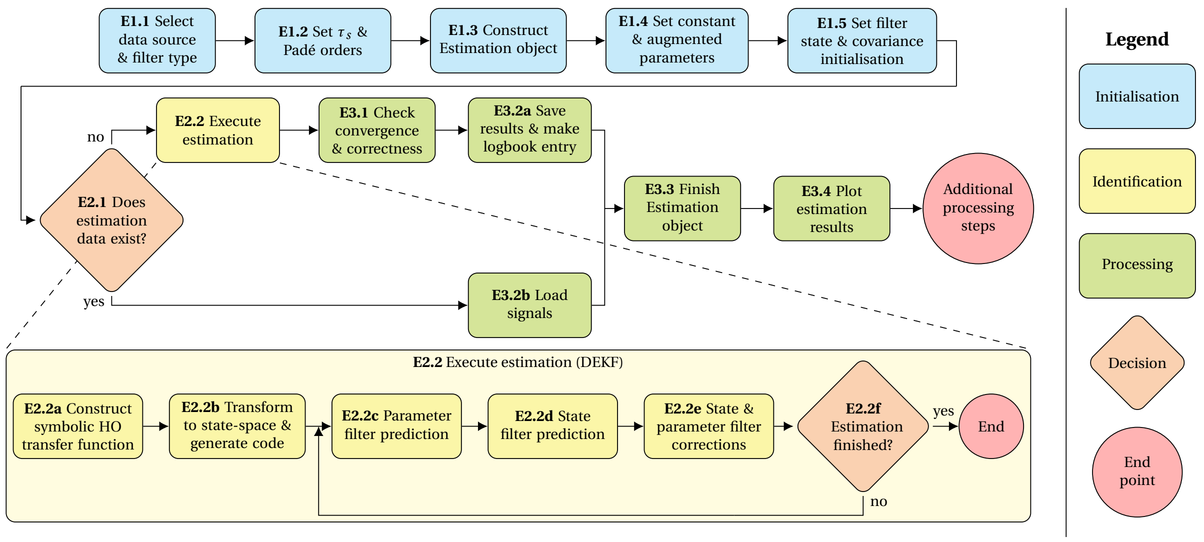
1. Linear (time-invariant) simulations can simply be verified by running an `lsim` command on the same system as the one constructed in simulation step S2.3a. After all, for constant parameter values, the time-varying LPV simulation simply reduces to a linear system.
2. Time-varying simulations with SI system dynamics are verified by a `Simulink` model. Unfortunately, the `Simulink` model does not always converge for simulations with DI dynamics.

Together, these verification methods cover all linear and some time-varying simulations. Only time-varying, DI simulations cannot be verified. However, the DI dynamics and the time-varying parameters are verified independently with methods 1 and 2, respectively. Moreover, the current implementation was also verified for a time-varying DI case using the method of Appendix B.2. Therefore, the verification is deemed complete.

5
Parameter Estimation in Preview Tracking

The main goal of this research is to identify the parameters in Van der El et al.'s human operator model for preview tracking from simulated and experimental tracking data. The simulations developed in the previous chapter are merely to generate artificial human control data. The data can be used to develop and test methods for estimation in a more controlled manner. This chapter will explain how the estimation methods are implemented for this research.

The general outline of the estimation procedure is sketched in Figure 5.1. Initially, this follows a similar outline to the Simulations, with the three phases of *initialization*, *identification* and *processing*. Each of these phases is elaborately described in the following three sections. Important elements in the estimation procedure are the concept of *suspension time*, the deriviation of a minimal order HO model and an alteration to the original DEKF algorithm. These topics therefore have dedicated subsections (Subsections 5.1.2, 5.2.1 and 5.2.2, respectively).



E1.1 Select data source & filter type
E1.2 Set τ_s & Padé orders
E1.3 Construct Estimation object
E1.4 Set constant & augmented parameters
E1.5 Set filter state & covariance initialisation
no
E2.2 Execute estimation
E3.1 Check convergence & correctness
E3.2a Save results & make logbook entry
E2.1 Does estimation data exist?
E3.3 Finish Estimation object
E3.4 Plot estimation results
Additional processing steps
yes
E3.2b Load signals
E2.2 Execute estimation (DEKF)
E2.2a Construct symbolic HO transfer function
E2.2b Transform to state-space & generate code
E2.2c Parameter filter prediction
E2.2d State filter prediction
E2.2e State & parameter filter corrections
E2.2f Estimation finished?
yes
End
no
Legend
Initialisation
Identification
Processing
Decision
End point

Figure 5.1: Flowchart of Human Operator parameter estimation

5.1. Estimation initialisation

In the first, *initialization* phase of estimation (in blue), all the settings of the estimation are applied.

5.1.1. Initialisation steps

The initialisation of the estimation consists of 5 steps. First, the data source and the estimation method are selected (step E1.1). Secondly, the orders of the Padé approximations to be used for effective time delay and apparent time delay are specified in step E1.2. *Apparent time delay* is the result of a mathematical trick that makes it possible to estimate preview time (a negative time delay) as a normal positive delay. This is more elaborately explained in Subsection 5.1.2. The Padé approximation was already explained in Subsection 4.1.3.

With this information, the Estimation class object is constructed in step E1.3. In the next step E1.4, the states that are to be kept constant during estimation are specified, together with their value. Finally, the initialization parameters of the estimation procedures are set in E1.5. These are for example initial HO parameter guesses or initial covariance matrices (P, Q & R, see Subsection 2.2.2).

5.1.2. Suspension time

One of the challenges in identifying preview tracking, is the preview element itself. By definition, preview looks out into the future of the forcing function signal. This makes the system non-causal and introduces a negative time delay. Inserting this negative delay in a Padé approximation causes the system to become divergent.

To solve this issue, a mathematical trick is used, that has been named *suspension time*. A new reference for the timing of signals is defined, that lies somewhat in the future. This moment must be ahead of the preview time that people use in their tracking. This is graphically illustrated in Figure 5.2, where the forcing function runs into the display from top to bottom. The new reference line as well as the viewpoint of the operator are indicated.

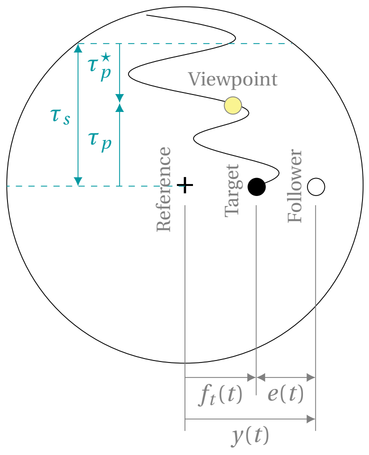


Figure 5.2: Preview time (τ_p), suspension time (τ_s) & apparent time delay ($\tau_p^\star$)

Another way of thinking about this, is that the start point of measurement time is simply redefined. The clock has already started running, but the visual forcing function signal (and optionally motion cues) are *suspended* for τ_s seconds. For the experiment, this does not make a difference. The experiment shifts τ_s seconds on our experiment timeline.

Mathematically, however, this has the effect of an artificial time delay being applied to all forcing function and disturbance signals. The result of this mathematical trick is that the preview time τ_p, which is normally a negative delay, can be estimated as a positive *apparent time delay* ($\tau_p^\star$) relative to the new reference. The suspension time, apparent time delay and preview time (defined positive for a negative delay) are related according to:

$$\tau_p^\star = \tau_s - \tau_p \tag{5.1}$$

From Equation (5.1), it becomes obvious that the suspension time defines the upper bound for the preview time. If τ_p would surpass τ_s, the apparent delay would become negative. This in turn results in instabilities and divergence of the system.

As mentioned in Subsection 4.1.3, a Padé approximation becomes more accurate with higher order or with a lower time delay. For a given order, it is thus favourable to minimize $\tau_p^\star$ by choosing the suspension time as close as possible to the preview time. On the contrary, a small apparent time delay limits the freedom during the estimation of $\tau_p^\star$ (and thus τ_p). The resulting balance of high approximation accuracy versus ample estimation space is one to watch very closely in the process of identifying HO behaviour.

5.2. Parameter identification

With the data loaded and the settings for estimation applied, the filter algorithm can do its work. The algorithm subject in this thesis, the Dual Extended Kalman Filter, was explained in Subsection 2.2.2. Kalman filters require a model of the system at hand, in this case the isolated human operator (the grey box of Figure 4.2).

First, a symbolic transfer function of this model is formulated in step E2.2a. Subsequently, the transfer function is transformed into nonlinear state-space form (step E2.2b). To maximize the likelihood of convergence during the estimation process, it is vital that the system description is a minimal realisation. Subsection 5.2.1 explains how a minimal realisation state-space of the HO preview model was obtained. The final nonlinear state space equations can be found in Appendix C.

As explained in Subsection 2.2.2, the DEKF splits the problem, separating the states from the parameters. The parameters that appear in the denominator of the HO transfer function end up in the state equation (see Subsection 4.2.2). The parameters in Figure 4.2, that only appear in the numerator (and thus only in the output equation) get augmented in the state filter ($K_f, K_p, K_v(,K_n)$). The other parameters form a separate parameter vector, that forms the basis of the parameter filter. This results in the following state and parameter vector for the far-view only preview model, respectively: $\boldsymbol{x_s} = [x_{s,1},\dots,x_{s,9},K_p,K_v,K_f]^T$, $\boldsymbol{\theta} = [\omega_n,\zeta_n,\tau_v,\omega_{b,f},\tau_f^\star]^T$. Note that the preview system has nine canonical states that capture the internal dynamics.

The DEKF algorithm was very clearly described by Popovici et al. (2017) in Equations (6)-(15) and (42)-(45). This procedure was strictly followed. The main steps are depicted in the detail box of Figure 5.1. Note that the recursive steps E2.2c-E2.2e are in accordance with Figure 2.4 of the original implementation. During the first tests of the filter however, Equation (44) of the original algorithm was found to cause divergence. This equation was therefore rederived. A detailed breakdown of the problem, and the rederived solution can be found in Subsection 5.2.2. With this updated equation in place, the filter was found to be functional as intended.

5.2.1. Constructing a minimal realisation of the HO model

The Kalman Filters used for estimation rely on a system model to work. The model is used to compute control action predictions from the current state and a forcing function update. To reduce the amount of states that need to be estimated alongside the parameters, it is of utmost importance to isolate a *minimal order* representation of the human operator model. The basic procedure is the same as described in Subsection 4.2.2:

1. Separate transfer functions for each IO pair are formulated that together describe the entire system
2. Each TF is transformed to a state-space system
3. The different systems are either merged or appended to form one state-space model

This construction was assisted by combining MATLAB and `Simulink` to verify the result. For the construction of a minimal order state-space for other, more complicated block diagrams, this procedure might be useful. It is described in detail in Appendix D

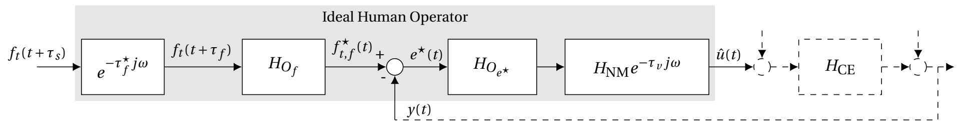


Figure 5.3: Isolated block diagram of the HO in preview tracking tasks, far view only (Van der El et al., 2016)

Figure 5.3 shows the block diagram of the HO model, with as additional block the effects of *suspension time* (τ_s) and *apparent time delay* (τ_a) as explained in Subsection 5.1.2. Firstly, it can be noted that the idealised model is a MISO system. The remnant input and the disturbance signal lie outside of this isolated HO model. However, the target signal as well as the system state output of the previous time step do both appear as inputs to the model. The model has only one output: the theoretical HO stick input $\hat{u}(t)$.

Secondly, the model is open loop, so no integration on top of the orders of dynamics of the separate blocks is added. Finally, both inputs insert at different points into the model. This means that part of the dynamics applies to all signals (equalization and neuromuscular dynamics) and part of the dynamics is applied only to the forcing function (preview dynamics).

A minimal order model can be constructed by splitting up the system at the summation point (Figure 5.4). Two simple open-loop systems then emerge, that can each be transformed into a state-space representation.

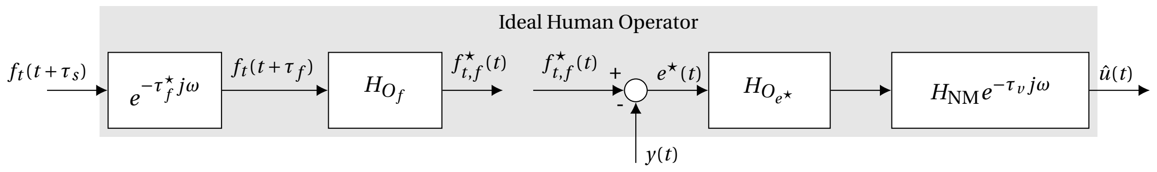


Figure 5.4: Split version of the isolated block diagram of the HO in preview tracking tasks, far view only (Van der El et al., 2016)

The first part is a simple SISO-system, the second system has two inputs (do keep note of the minus sign for the feedback signal input y). Since the canonical form ensures that the realisation for each of the two parts is minimal, and because the two parts on either side of the split are independent, the resulting description is also of minimal order. Merging these two systems can be done by making use of a property of the controllable canonical form. In this system type, the canonical states start in order at the *output* of the system and move their way up the integrations towards the input of the system. This is visualised by the block diagram in Figure 4.3 for the example used earlier.

Equations (5.2) and (5.3) illustrate what the combined state-space system looks like. The states and state matrix of the first system are thus appended *below* the second system, propagating the newly added canonical states of the preview system ($\boldsymbol{x}_1$) below the ones from neuromuscular dynamics and equalisation ($\boldsymbol{x}_2$. Besides, the output equation of system 1 is appended to the last row of the first system in the state equation. The error of this system (the derivative of the first canonical state), is thus equal to the sum of the newly introduced input $-y$ and the output of the previous part. Finally, the output equation of the combined system is simply equal to the output of the second part, with some added columns in the C-matrix to make up for the additional states. For this research, the addition of the near view response, or further extensions such as for car driving, were not yet implemented. However, using similar steps such as described here and in Subsection 4.2.2, this should be relatively straightforward. The full state-space equations as they were derived according to this procedure, can be found in C.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boxed{\begin{matrix}\cdot&\cdot&\cdot\\ \cdot & A_2 & \cdot\\ \cdot&\cdot&\cdot\end{matrix}} & \\ & \boxed{\cdot\ C_1\ \cdot} \\ & \boxed{\begin{matrix}\cdot&\cdot&\cdot\\ \cdot & A_1 & \cdot\\ \cdot&\cdot&\cdot\end{matrix}}\end{bmatrix} \begin{bmatrix}\boxed{\begin{matrix}\cdot\\ \boldsymbol{x}_2\\ \cdot\end{matrix}}\\ \boxed{\begin{matrix}\cdot\\ \boldsymbol{x}_1\\ \cdot\end{matrix}}\end{bmatrix} + \begin{bmatrix} & \overset{B_2}{\boxed{\begin{matrix}0\\ \cdot\\ -1\end{matrix}}} \\ \overset{D_1}{\boxed{0}} & \\ \underset{B_1}{\boxed{\begin{matrix}0\\ \cdot\\ 1\end{matrix}}} & \end{bmatrix}\begin{bmatrix}f_t\\ y\end{bmatrix} \tag{5.2}$$

$$\hat{u} = \begin{bmatrix} \boxed{\cdot\ C_2\ \cdot} & \cdot & 0 & \cdot \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 & \overset{D_2}{\boxed{0}} \end{bmatrix}\begin{bmatrix}f_t\\ y\end{bmatrix} \tag{5.3}$$

5.2.2. DEKF state derivative correction

In the original application of the filter, the Jacobian of output function g with respect to the parameter vector (G_p) is replaced by a total derivative formulation (G_p^{tot}) (Popovici et al., 2017, Eqs. (42)-(45)). This is done to include states that only appear in the denominator of the transfer function, and thus only in the state equation, explicitly in the evaluation of this Jacobian. During initial testing, however, it was found that Equation (44) diverges in just a few time steps:

$$\frac{d\boldsymbol{x}^-_{s,k}}{d\boldsymbol{\theta}} = \frac{\partial f(\boldsymbol{x}^+_{s,k-1}, e_{k-1}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial f(\boldsymbol{x}^+_{s,k-1}, e_{k-1}, \boldsymbol{\theta})}{\partial \boldsymbol{x}^+_{s,k-1}} \frac{d\boldsymbol{x}^+_{s,k-1}}{d\boldsymbol{\theta}} \tag{44, Popovici et al. (2017)}$$

After examining of the units of the vectors and matrices in the equation, it was found that the representation as presented in the paper could not be correct. Note that during differentiation, the units of a quantity get divided by the units of the quantity that is being differentiated with. For example differentiating distance with respect to time results in a unit transformation from $[m]$ to $[m/s]$. As can be seen in Equation (5.4), the right hand side of the equation has an additional division by seconds. This is caused by the evaluation of function f resulting in time derivative $\dot{\boldsymbol{x}}$ with units $[\boldsymbol{x}]/s$.

$$\begin{aligned}\frac{[\boldsymbol{x}]}{[\boldsymbol{\theta}]} &= \frac{[f(\boldsymbol{x}, e, \boldsymbol{\theta})]}{[\boldsymbol{\theta}]} + \frac{[f(\boldsymbol{x}, e, \boldsymbol{\theta})]}{[\boldsymbol{x}]}\frac{[\boldsymbol{x}]}{[\boldsymbol{\theta}]} \\ &= \frac{[\dot{\boldsymbol{x}}]}{[\boldsymbol{\theta}]} + \frac{[\dot{\boldsymbol{x}}]}{[\boldsymbol{x}]}\frac{[\boldsymbol{x}]}{[\boldsymbol{\theta}]} \\ &= \frac{[\boldsymbol{x}]}{s[\boldsymbol{\theta}]} + \frac{[\boldsymbol{x}]}{s[\boldsymbol{\theta}]}\end{aligned} \tag{5.4}$$

Therefore, Equation (44) was rederived in Equation (5.5) by calculating the total derivative of the Taylor expansion of Equation (8) for $x^-_{s,k}$, using a chain rule for differentiating the function evaluation term. Note that the equation looks very similar to the original Equation (44), but that the multiplication with time step Δt prevents the matrix from diverging. Also, the units of right hand side and left hand side do match in Equation (5.5).

$$\boldsymbol{x}^-_{s,k} = \boldsymbol{x}^+_{s,k-1} + f(\boldsymbol{x}^+_{s,k-1}, e_{k-1}, \boldsymbol{\theta}^-_k)\Delta t \qquad \text{(8, Popovici et al. (2017))}$$

$$\frac{d\boldsymbol{x}^-_{s,k}}{d\boldsymbol{\theta}} = \frac{d\boldsymbol{x}^+_{s,k-1}}{d\boldsymbol{\theta}} + \left(\left.\frac{\partial f(\boldsymbol{x}^+_{s,k-1}, e_{k-1}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^-_k} + \left.\frac{\partial f(\boldsymbol{x}_s, e_{k-1}, \boldsymbol{\theta})}{\partial \boldsymbol{x}_s}\right|_{\boldsymbol{x}_s=\boldsymbol{x}^+_{s,k-1}} \frac{d\boldsymbol{x}^+_{s,k-1}}{d\boldsymbol{\theta}} \right)\Delta t \tag{5.5}$$

With this updated equation in place of the original Equation (44), the filter was found to converge.

5.3. Processing

Finally, the results of the estimation process are compiled, analysed and refined during *processing*. First, a check is performed to see if the parameters have converged and whether they are correct (step E3.1). Checking for convergence is done by checking that the resulting parameter traces contain no `Inf` or `NaN` values.

Convergence thus does not say anything about the quality of the solution yet. This is done by a second check. For a nonzero original parameter value, the normalised root mean squared error (NRMSE) must be smaller than a certain threshold. By default, this threshold is set to 25% to account for some oscillation of the estimated parameter around the real value.

$$\text{NRMSE} = \frac{1}{\bar{\theta}_{\text{true}}}\sqrt{\frac{1}{N_{\text{meas}}}\sum_{i=0}^{N_{\text{meas}}}(\theta_{\text{DEKF}}(i) - \theta_{\text{true}}(i))^2} \quad < \quad 0.25 \tag{5.6}$$

For values that are zero, it is not possible to make a relative comparison, since that would involve dividing by 0. Therefore, for these parameters, or the ones very close to zero ($|\bar{p}| < 0.05$) the criterion becomes that the root mean squared error (RMSE) should not exceed this threshold of 0.05 (see Equation (5.7)). Checking for correct values is of course only possible if the data source is a simulation and not an experiment, such that the correct value θ_{true} is available.

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{meas}}}\sum_{i=0}^{N_{\text{meas}}}(\theta_{\text{DEKF}}(i) - \theta_{\text{true}}(i))^2} \quad < \quad 0.05 \tag{5.7}$$

If the estimation is carried out for the first time, the results can be saved and logged for future reference in step E3.2. Step E3.3 then gives the option to plot the time traces, followed by a few steps with some more advanced analysis. The further analysis is still in development for the final research. The ideas for additional analysis steps can be found in Section 7.1.

6 Preliminary Results

The previous chapters have described the selection and implementation of preview models in time-varying simulation and estimation. With those tools, the core research of this thesis was executed. As indicated in the research outline in Section 3.2, the first tests are performed using benchmark simulations. How these are precisely set up is discussed in Section 6.1, together with some samples of the simulation results. The following Section 6.2 the provides the details of the estimation settings that are applied to the different simulations. Section 6.3 shows the results of the benchmark estimations with the DEKF.

The preliminary results are subsequently used to assess how promising the DEKF is for identifying time-varying parameters in preview tracking. These observations are identified in Section 6.4. The final section of this chapter, Section 6.5, will discuss the areas that still need attention going forward.

6.1. Benchmark simulations

The general goal of the benchmark simulations is to test whether the selected algorithm(s) are suitable for time-varying identification of preview parameters. Eight simulations (Table 6.1), building up in complexity and realism, systematically assess the algorithm's potential for identification. Of the simulations indicated in blue in this table, a representative sample is shown below.

- **Sim 1** is a simulation of **compensatory tracking** used to verify the results of Popovici et al. (2017). A double integrator is used to allow for the use of all 5 HO parameters. The forcing function, system dynamics and parameter values are taken from the same sources as the other simulations. The equalization gains are lowered slightly, though, to make the simulation more stable. Moreover, remnant with the power ratio and break frequency reported by Popovici (30%, 3 rad/s) are included.
- **Sim 2** is the simulation that forms the **baseline** of this benchmark. It is a far-view-only preview simulation according to the model from Van der El et al. (2016) with SI dynamics.
  - As found by Van der El et al. (2016) and later research, the contribution of the near view response is very limited. Excluding it reduces the estimation problem to its minimal size.
  - Similarly, SI dynamics make estimation convergence more likely, since it omits one parameter (K_v). Besides, the loop is more stable to parameter variances than for the DI case.
  - The parameter values are taken from the theoretical optimized values in Van der El et al. (2018), as well as the CE transfer functions.
  - The forcing function is taken from Van der El et al. (2019b), to have multiple equivalent realisations and a matching disturbance signal readily available. The widest bandwidth (4 rad/s) is chosen for the forcing functions, to include as much information in the signals as possible and thus invoke the clearest parameters for estimation. The runs have a measurement time of $120s$. To this time, $60s$ of run-in time is added, to minimize transient simulation responses.
  - The time-delay (τ_v) is included as a fourth order Padé approximation, one order higher than used in the estimation process by Popovici et al. (2017). As explained in Subsection 5.1.2, a Padé approximation becomes more accurate with lower delay or higher order. Therefore, with a higher order approximant, the simulation will definitely encompass the estimation's precision. Choosing one order higher still limits the computational complexity and thus time. Also, this will hopefully prevent the algorithm from estimating delays too easily from simulated data, since the approximation

in the estimation matches the simulated delay too closely. Note that especially for a fourth order approximation, the phase of the time delay's frequency response only starts deviating beyond the bandwidth of the forcing function.
  - This simulation is free of any remnant injection.

- **Sim 3** introduces **time-varying behaviour** in the main parameter of interest, preview time τ_v. In all other aspects,the simulation is equal to Sim 2.
- **Sim 4** adds **remnant** to the baseline simulation, according to the first-order model of Levison et al. (1969). The remnant power ratio is taken relatively low at 15%. This percentage is taken as a compromise between adding realistic noise content and not slimming down the changes of convergence too much. Later on during this research, in the sensitivity simulations, more realistic amounts of remnant will be injected to determine the limits of the estimation algorithm(s).
- **Sim 5** combines the **time-variance** of Sim 3 (sigmoidal τ_f), with the **remnant** power ratio of Sim 4.
- **Sim 6** applies the same settings of Sim 5 to a **different target realisation**.
- **Sim 7** adds more time-variance by means of a **second sigmoidal parameter**: equalization gain K_p.
- The final **Sim 8** is a simulation with **DI dynamics**, introducing also the last parameter K_v and more unstable system dynamics. The parameter values are adjusted to the double integrator system. Again,this simulation has 15% remnant and sigmoidal τ_f.

Table 6.1: Benchmark simulation settings (*sigmoidal set as ($v1$,$v2$,T_{infl},Δ_{infl}), **sinusoidal set as (Mean,Amp,Period,Phase))

Sim	**Task/Time-variance**	**CE type (tf)**	**Target signal (T_{sim}+run-in)**	**Remnant model (parameters)**	**HO parameters**
1	Compensatory, Invariant	DI ($H_C = \frac{5}{s^2}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	Levison ($P_r(0.30)$, $K(1.44), \omega_b(3)$)	$K_p(0.1), K_v(1.5), \omega_n(8), \zeta_n(0.45), \tau_v(0.3)$
2	Prev. (far only), Invariant	SI ($H_C = \frac{1.5}{s}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	None	$\tau_f(0.6), K_f(1), \omega_f(3.3), K_p(1.3), K_v(0), \omega_n(10.5), \zeta_n(0.35), \tau_v(0.26)$
3	Prev. (far only), τ_f varying	SI ($H_C = \frac{1.5}{s}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	None	$\tau_f(\text{sigm}, 0.3, 0.6, 50, 10)^*$ $K_f(1), \omega_f(3.3), K_p(1.3), K_v(0), \omega_n(10.5), \zeta_n(0.35), \tau_v(0.26)$
4	Prev. (far only), Invariant	SI ($H_C = \frac{1.5}{s}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	Levison ($P_r(0.15)$, $K(2.12), \omega_b(3)$)	$\tau_f(0.6), K_f(1), \omega_f(3.3), K_p(1.3), K_v(0), \omega_n(10.5), \zeta_n(0.35), \tau_v(0.26)$
5	Prev. (far only), τ_f varying	SI ($H_C = \frac{1.5}{s}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	Levison ($P_r(0.15)$, $K(2.14), \omega_b(3)$)	$\tau_f(\text{sigm}, 0.3, 0.6, 50, 10)^*$ $K_f(1), \omega_f(3.3), K_p(1.3), K_v(0), \omega_n(10.5), \zeta_n(0.35), \tau_v(0.26)$
6	Prev. (far only), τ_f varying	SI ($H_C = \frac{1.5}{s}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 2 (120+60s)	Levison ($P_r(0.15)$, $K(2.13), \omega_b(3)$)	$\tau_f(\text{sigm}, 0.3, 0.6, 50, 10)^*$ $K_f(1), \omega_f(3.3), K_p(1.3), K_v(0), \omega_n(10.5), \zeta_n(0.35), \tau_v(0.26)$
7	Prev. (far only), τ_f, K_p varying	SI ($H_C = \frac{1.5}{s}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	Levison ($P_r(0.15)$, $K(2.02), \omega_b(3)$)	$\tau_f(\text{sine}, 0.4, 0.2, 40, 0)^{**}$ $K_f(1), \omega_f(3.3)$, $K_p(\text{sig}, 1.35, 0.85, 70, 5)^*$ $K_v(0), \omega_n(10.5), \zeta_n(0.35), \tau_v(0.26)$
8	Prev. (far only), τ_f varying	DI ($H_C = \frac{5}{s^2}$)	Bandwidth 4 $\frac{\text{rad}}{\text{s}}$, real. 1 (120+60s)	Levison ($P_r(0.15)$, $K(1.58), \omega_b(3)$)	$\tau_f(\text{sigm}, 0.6, 1.0, 50, 10)^*$ $K_f(0.9), \omega_f(1), K_p(0.24), K_v(0.36), \omega_n(8), \zeta_n(0.45), \tau_v(0.3)$

Below, representative samples of four of the benchmark simulations can be found. The simulations plotted are numbers 1, 2, 5 and 8 from Table 6.1. This shows the differences between

- Compensatory and preview tracking (sim 1 vs sim 2)
- Time-varying and invariant behaviour (sim 2 vs sim 5)
- Simulations with and without remnant (sim 2 vs sim 5)
- Single and double integrator dynamics (sim 5 vs sim 8)

As was explained in Section 4.4, different ways of verification are available depending on the type of simulation. For invariant simulations, an `lsim` simulation was executed as well. For SI system dynamics, a `Simulink` implementation was used to verify the results.

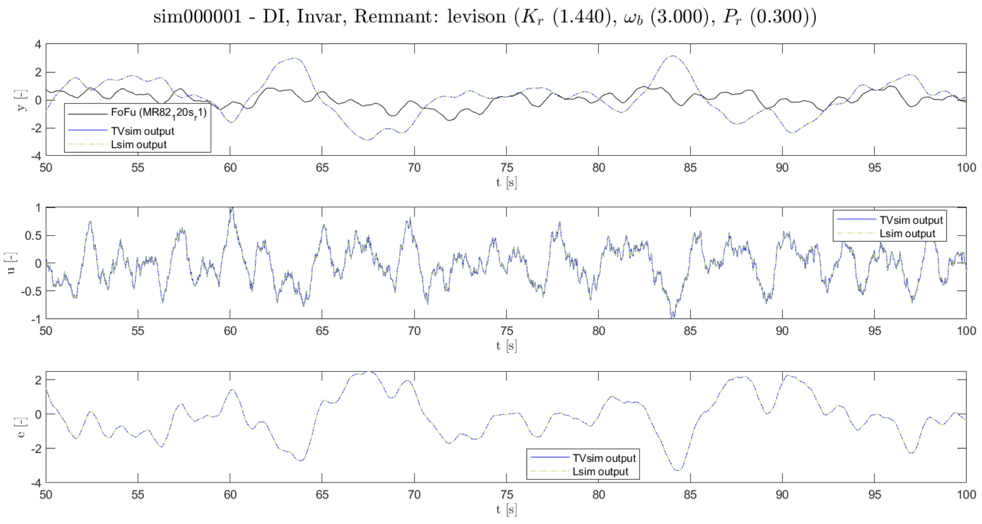


Figure 6.1: Benchmark simulation 1 (representative sample)

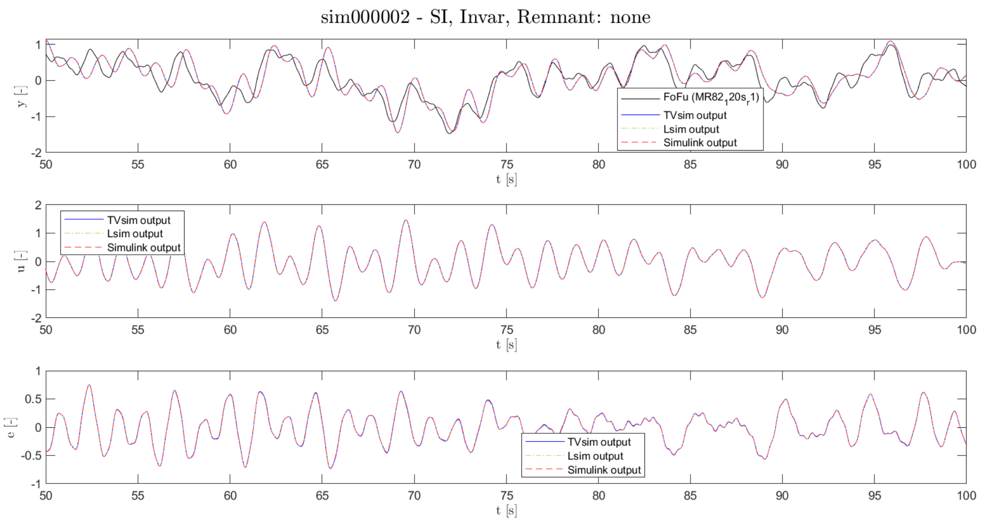


Figure 6.2: Benchmark simulation 2 (representative sample)

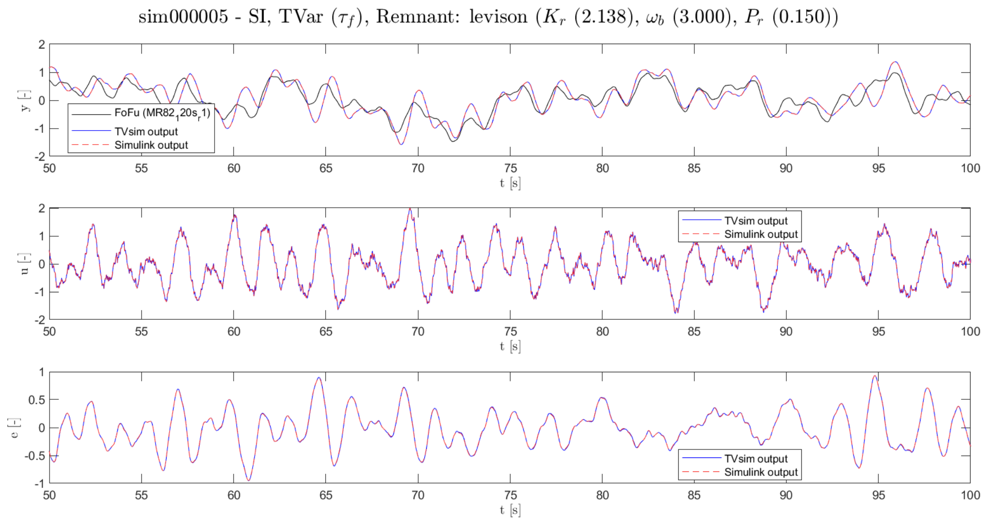


Figure 6.3: Benchmark simulation 5 (representative sample)

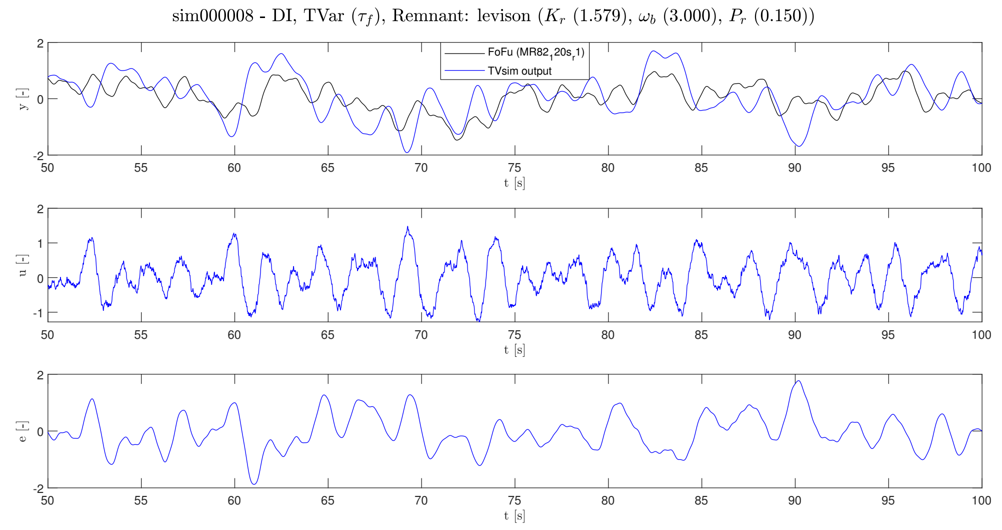


Figure 6.4: Benchmark simulation 8 (representative sample)

6.2. Estimation settings

The primary candidate to test for time-varying estimation is the Dual Extended Kalman Filter (DEKF). As explained in Figure 5.1/Section 5.1, several settings have to be applied. There are four major settings to tune:

- Which states and parameters to include in the estimation process and which to keep constant.
- Initial values of states ($\boldsymbol{x_{s,0}}$) & parameters ($\boldsymbol{\theta_0}$). Both the ones being estimated and those kept constant.
- The covariance matrices of the states and parameters ($P_{s,0}$, $P_{p,0}$), the process noise ($Q_{s,0}$, $Q_{p,0}$) and the measurement noise (R_0).
- Suspension time τ_s.

As a starting point, the amount of variables kept constant during estimation is the main setting that was varied. It reduces the complexity of the estimation. The other settings were taken as similar as possible to how (Popovici et al., 2017) applied them successfully to compensatory tracking data. Table 6.2 summarizes the estimations performed to scrutinize each benchmark simulation. As explained in Subsection 2.2.2, the covariance matrices Q_s and R are made variable, proportional to the variance of the error σ_e^2 and stick input σ_u^2. The parameters q^2 and r^2 are taken to be 0.4 and 0.7, respectively (Popovici et al., 2017). σ_u^2 and σ_e^2 are updated with the last 5 seconds of measurement data as a rolling average value, and are initialised with the first 5 seconds of data.

Note that state vector $\boldsymbol{x_s}$ first starts with a number of canonical states initialised with 0, resulting from the system conversion described in Subsection 5.2.1 (5 for compensatory tracking and 9 for preview tracking). Since the preview time τ_f never exceeds 0.6s in the SI runs, the suspension time τ_s is initially set to 1s for sims 2-7 (with some exceptions). For the simulations with remnant, more margin was sometimes necessary (see Est 5.1-5.5). For sim 7, τ_s is therefore set to 1.25s. For sim 8, τ_s is set to 1.5s.

The estimations marked in blue can be found in the next section. The rest is appended in Appendix E.

Table 6.2: Benchmark estimation settings ($^* D[\cdots]$ indicates diagonal square matrix)

Est	**Summary**	**Const. pars. (value)**	**State/Parameter initialization**	P **initialization**	Q **initialization**	R_0
1.1	Compensatory, Invariant, DI, FoFu 1, P_r 30%	–	$\boldsymbol{x_{s,0}} = [0\ (\times 5), K_p(0.1), K_v(0.15)]$ $\boldsymbol{\theta_0} = [\omega_n(8), \zeta_n(0.45), \tau_v(0.3)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 5), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 4), q^2\sigma_e^2, K_{p,0}, K_{v,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{n,0}, 0.1\zeta_{n,0}, 0.1\tau_{v,0}]$	$r^2\sigma_u^2$
2.1	Preview, Invariant, SI, FoFu 1, P_r 0%, $\tau_s = 1$s	–	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_v(0), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_n(10.5), \zeta_n(0.35), \tau_v(0.26), \omega_{b,f}(3.33), \tau_f^\star(0.4)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1, 1, 10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{v,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{n,0}, 0.1\zeta_{n,0}, 0.1\tau_{v,0}, 0.1\omega_{bf,0}, 0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
2.2	Preview, Invariant, SI, FoFu 1, P_r 0%, $\tau_s = 1$s	$\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_v(0), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^\star(0.4)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{v,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
2.3	Preview, Invariant, SI, FoFu 1, P_r 0%, $\tau_s = 1$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^\star(0.4)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
3.1	Preview, τ_f varying, SI, FoFu 1, P_r 0%, $\tau_s = 1$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^\star(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
3.2	Preview, τ_f varying, SI, FoFu 1, P_r 0%, $\tau_s = 1$s	$K_v(0), K_f(1)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$, $\omega_{b,f}(3.33)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3)]$ $\boldsymbol{\theta_0} = [\tau_f^\star(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
4.1	Preview, Invariant, SI, FoFu 1, P_r 15%, $\tau_s = 1$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^\star(0.4)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
4.2	Preview, Invariant, SI, FoFu 1, P_r 15%, $\tau_s = 1$s	$K_v(0), K_f(1)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$, $\omega_{b,f}(3.33)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3)]$ $\boldsymbol{\theta_0} = [\tau_f^\star(0.4)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$
5.1	Preview, τ_f varying, SI, FoFu 1, P_r 15%, $\tau_s = 1$s	–	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_v(0), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_n(10.5), \zeta_n(0.35), \tau_v(0.26), \omega_{b,f}(3.33), \tau_f^\star(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1, 1, 10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{v,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{n,0}, 0.1\zeta_{n,0}, 0.1\tau_{v,0}, 0.1\omega_{bf,0}, 0.1\tau^\star_{f,0}]$	$r^2\sigma_u^2$

Table 6.3: Benchmark estimation settings (continued) ($^{*}D[\cdots]$ indicates diagonal square matrix)

Est	Summary	Const. pars. (value)	State/Parameter initialization	P initialization	Q initialization	R_0
5.2	Preview, τ_f varying, SI, FoFu 1, P_r 15%, $\tau_s = 1$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^{\star}(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
5.3	Preview, τ_f varying, SI, FoFu 1, P_r 15%, $\tau_s = 1.25$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^{\star}(0.95)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
5.4	Preview, τ_f varying, SI, FoFu 1, P_r 15%, $\tau_s = 1$s	$K_v(0), K_f(1)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$, $\omega_{b,f}(3.33)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3)]$ $\boldsymbol{\theta_0} = [\tau_f^{\star}(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
5.5	Preview, τ_f varying, SI, FoFu 1, P_r 15%, $\tau_s = 1.25$s	$K_v(0), K_f(1)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$, $\omega_{b,f}(3.33)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3)]$ $\boldsymbol{\theta_0} = [\tau_f^{\star}(0.95)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
5.6	Preview, τ_f varying, SI, FoFu 1, P_r 15%, $\tau_s = 1.25$s	$K_v(0), K_f(1.1)$, $\omega_n(9.5)$, $\zeta_n(0.4)$, $\tau_v(0.3)$, $\omega_{b,f}(5)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3)]$ $\boldsymbol{\theta_0} = [\tau_f^{\star}(0.95)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
6.1	Preview, τ_f varying, SI, FoFu 2, P_r 15%, $\tau_s = 1$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^{\star}(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
6.2	Preview, τ_f varying, SI, FoFu 2, P_r 15%, $\tau_s = 1$s	$K_v(0), K_f(1)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$, $\omega_{b,f}(3.33)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.3)]$ $\boldsymbol{\theta_0} = [\tau_f^{\star}(0.7)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
7.1	Preview, τ_f, K_p var., SI, FoFu 1, P_r 15%, $\tau_s = 1.25$s	$K_v(0)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.35), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(3.33), \tau_f^{\star}(0.85)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
7.2	Preview, τ_f, K_p var., SI, FoFu 1, P_r 15%, $\tau_s = 1.25$s	$K_v(0), K_f(1)$, $\omega_n(10.5)$, $\zeta_n(0.35)$, $\tau_v(0.26)$, $\omega_{b,f}(3.33)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(1.35)]$ $\boldsymbol{\theta_0} = [\tau_f^{\star}(0.85)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
8.1	Preview, τ_f varying, DI, FoFu 1, P_r 15%, $\tau_s = 1.5$s	$\omega_n(8)$, $\zeta_n(0.45)$, $\tau_v(0.3)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(0.24), K_v(0.36), K_f(1)]$ $\boldsymbol{\theta_0} = [\omega_{b,f}(1), \tau_f^{\star}(0.9)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10, 10]$ $\boldsymbol{P_{p,0}} = D[10, 1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{v,0}, K_{f,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\omega_{bf,0}, 0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$
8.2	Preview, τ_f varying, DI, FoFu 1, P_r 15%, $\tau_s = 1.5$s	$K_f(0.9)$, $\omega_n(8)$, $\zeta_n(0.45)$, $\tau_v(0.3)$, $\omega_{b,f}(1)$	$\boldsymbol{x_{s,0}} = [0\ (\times 9), K_p(0.24), K_v(0.36)]$ $\boldsymbol{\theta_0} = [\tau_f^{\star}(0.9)]$	$\boldsymbol{P_{s,0}} = D[0.1(\times 9), 10, 10]$ $\boldsymbol{P_{p,0}} = D[1]$	$\boldsymbol{Q_{s,0}} = D[0\ (\times 8), q^2\sigma_e^2, K_{p,0}, K_{v,0}]$ $\boldsymbol{Q_{p,0}} = D[0.1\tau_{f,0}^{\star}]$	$r^2\sigma_u^2$

6.3. Results: parameter time traces & filter convergence

The figures below show a selection of the estimation results for the runs described in the previous section. The full set can be found in Appendix E.

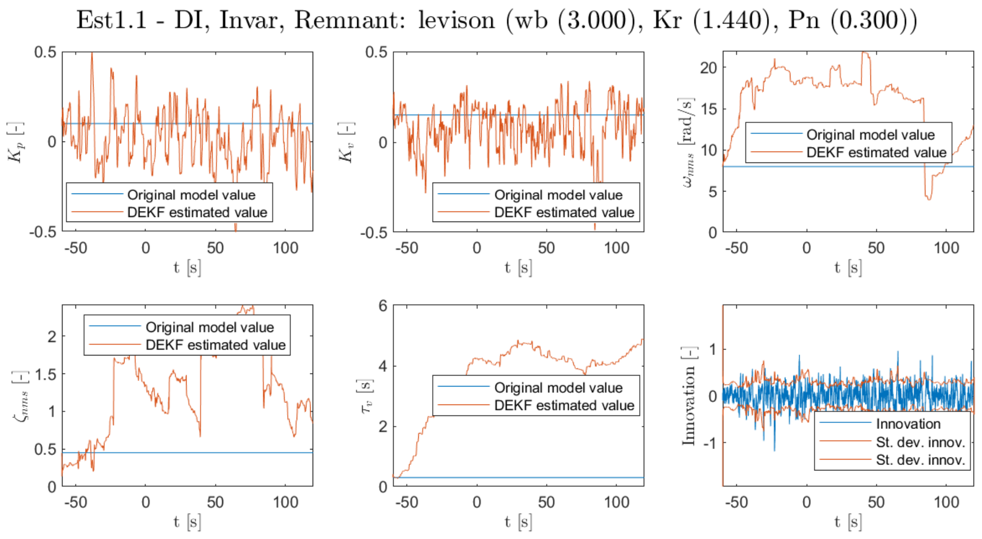


Figure 6.5: Benchmark estimation 1.1

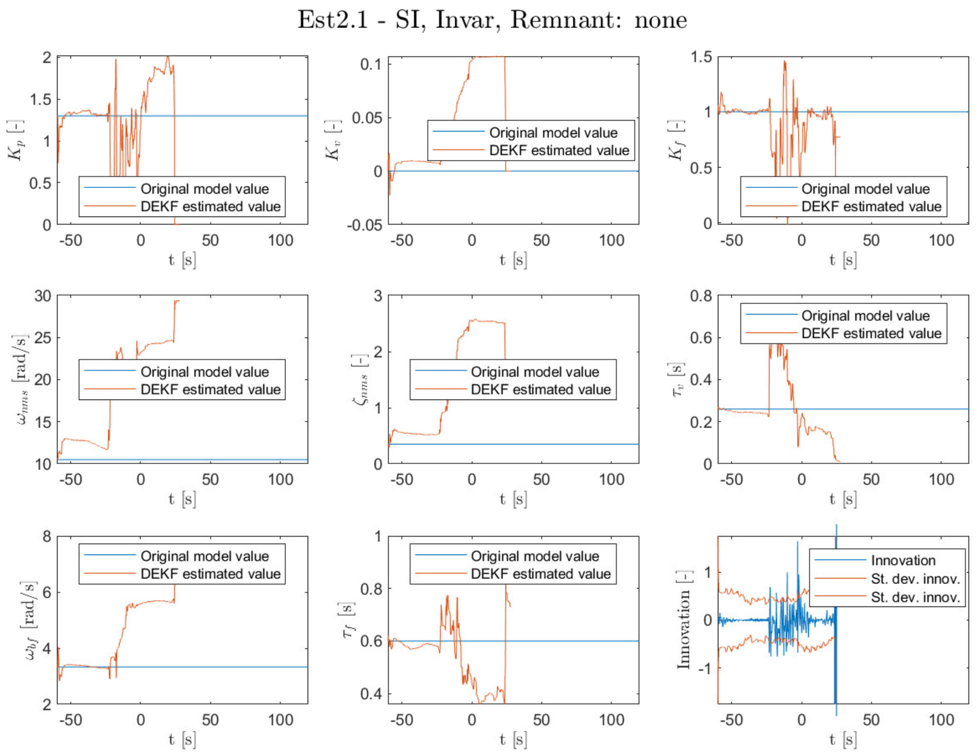


Figure 6.6: Benchmark estimation 2.1

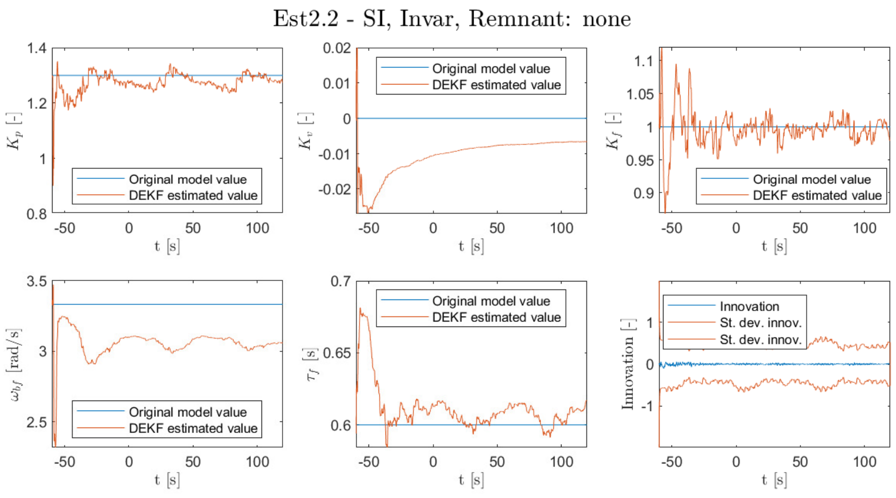


Figure 6.7: Benchmark estimation 2.2

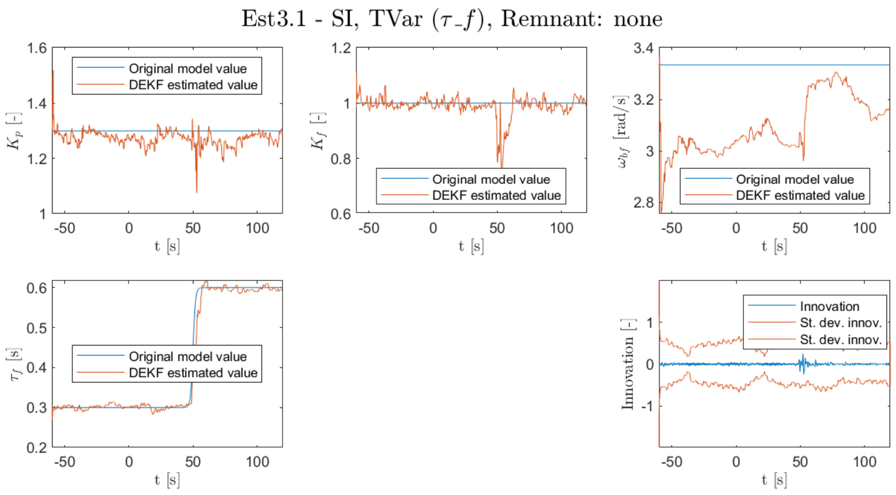


Figure 6.8: Benchmark estimation 3.1

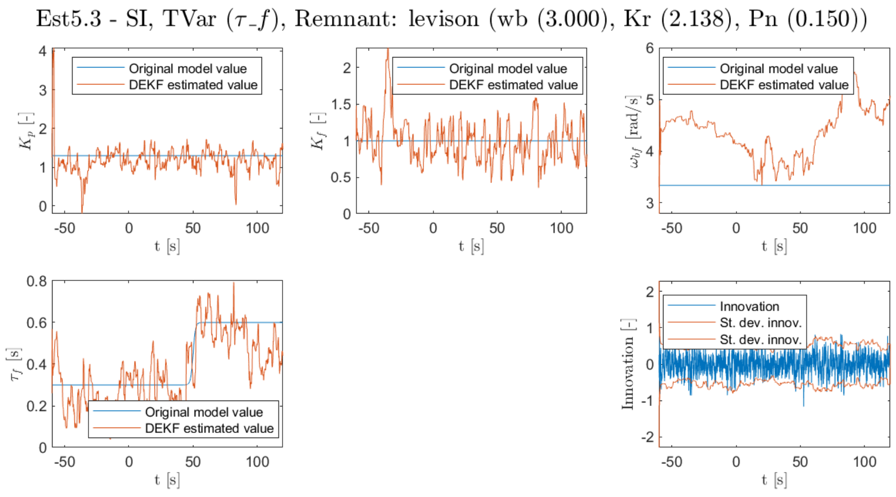


Figure 6.9: Benchmark estimation 5.3

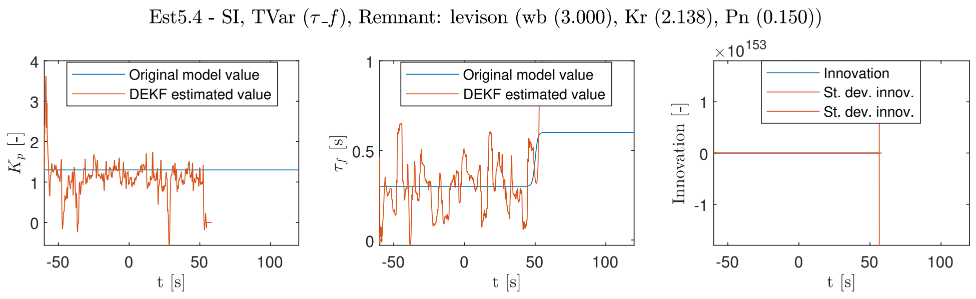


Figure 6.10: Benchmark estimation 5.4

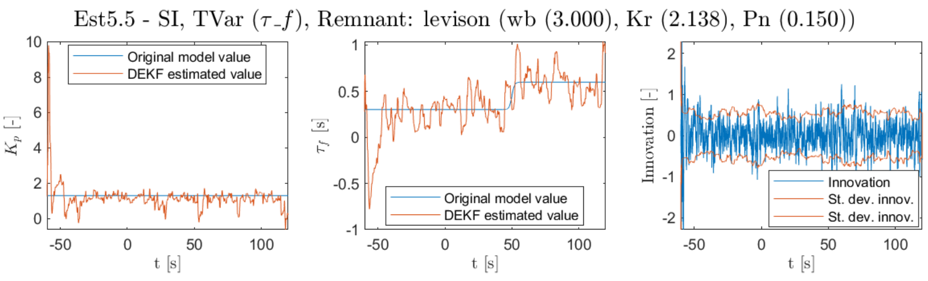


Figure 6.11: Benchmark estimation 5.5

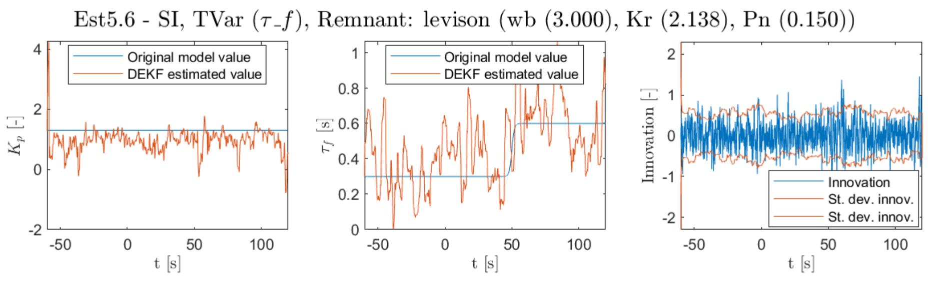


Figure 6.12: Benchmark estimation 5.6

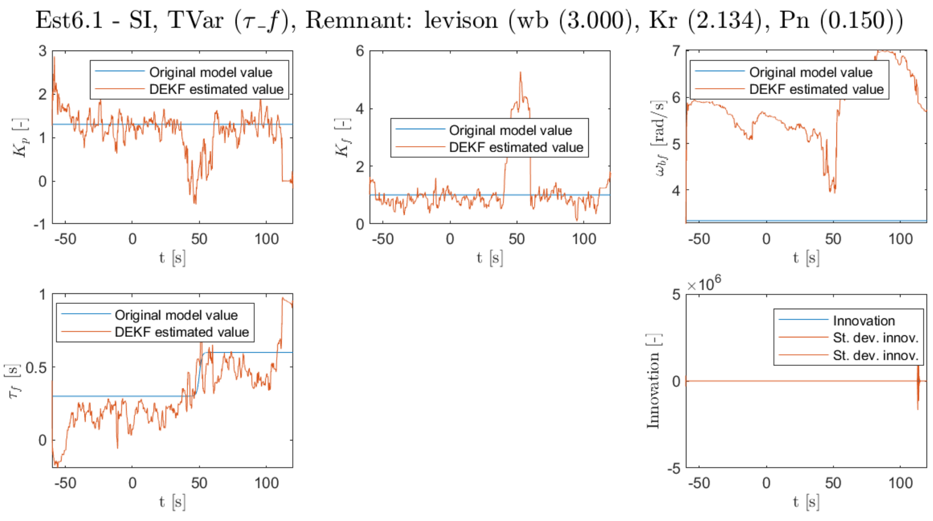


Figure 6.13: Benchmark estimation 6.1

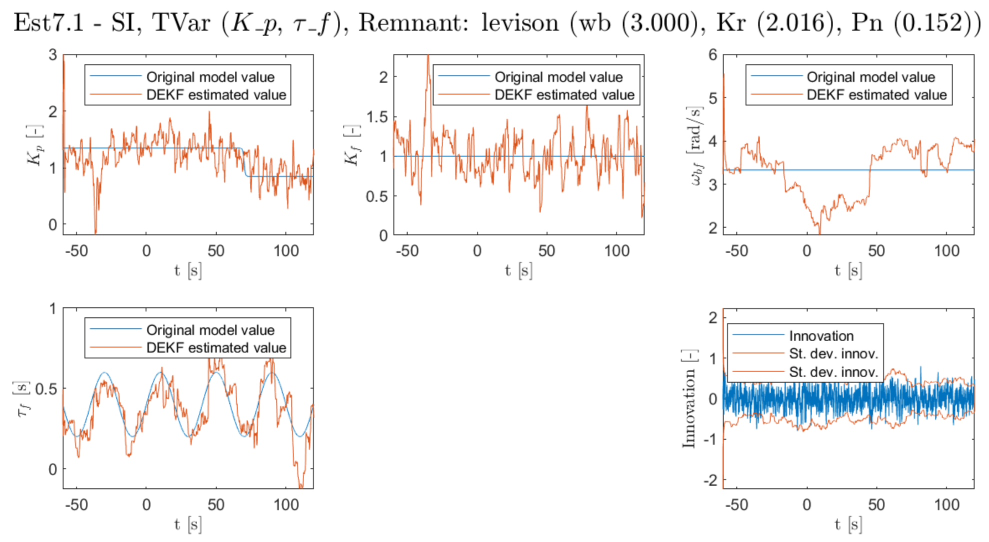


Figure 6.14: Benchmark estimation 7.1

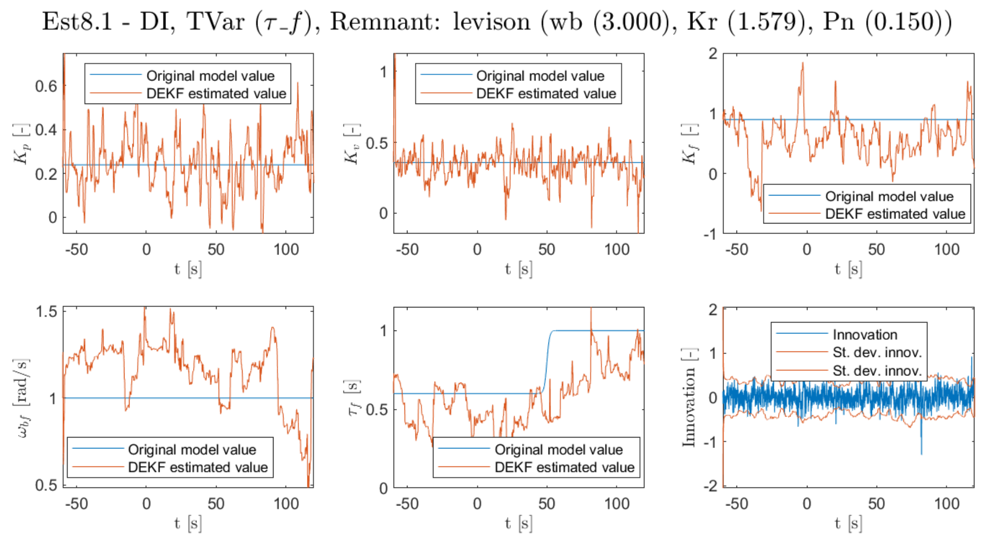


Figure 6.15: Benchmark estimation 8.1

6.4. Observations of the benchmark estimations

Using the preliminary results of the benchmark estimations as presented in Section 6.3, one can now assess the DEKF applied to identifying time-varying parameters in preview tracking. This section will discuss the observations that can be made from these results. The next section will then identify some areas that still need attention going forward. A plan for these improvements can be found in the next chapter.

Most importantly, it can be concluded that the DEKF is a promising candidate for at least partial parameter estimation of preview tracking tasks. The benchmark estimations from 2.2 onwards (Figures 6.7 to 6.15) have shown that the Dual Extended Kalman Filter has the capability to converge for preview tracking tasks, when keeping the neuromuscular parameters and time delay constant. These parameters have well-documented values and are believed to stay relatively constant anyway (Zaal, 2016). Although keeping them constant in a time-varying estimation is suboptimal, it is not insurmountable.

Parameter estimation for both SI and DI system dynamics was shown to be possible. However, since the response is much more sensitive to parameter changes, the identification results were typically better for SI experiments.

During the early tests of benchmarking, it was found that in the preview filter H_{O_f}, one should estimate a break frequency instead of a time constant. The time constant ($T_{l,f}$) is represented by a very small numerical value, that is very sensitive to minor changes. Moreover, it easily crosses to negative values, causing problems in the estimation process. Using its inverse break frequency ($\omega_{b,f}$) solves these issues, making the estimation process much more robust.

It was also demonstrated that using a suspension time results in successful identification of preview time (τ_f). This was visible for example in estimation 2.2 (Figure 6.7).

The DEKF has shown to be able to converge for simulations with remnant, for example in estimation 5.3 (Figure 6.9). Admittedly, the remnant power during benchmark was still relatively low. From the innovation, it becomes obvious that the variance band of innovation for simulations without remnant was wide. This means that the covariances of states (process noise) and of the HO input ('sensor' noise) were assumed to be fairly strong. In the case of simulations with remnant, the variance and actual innovation were already much closer together.

Aside from remnant, the DEKF was also shown to be capable of tracking time-varying behaviour (see est. 3.1, Figure 6.8). This tracking becomes very accurate under ideal conditions (no remnant noise). Even tracking of multiple time-varying parameters simultaneously is possible, as proven by estimation 7.1 (Figure 6.14). It was remarkable to see how accurate the DEKF could trace the sinusoidal preview time, even in the case of a noisy simulation.

Finally, the DEKF has shown to be able to cope with assumed parameters that are not at their correct value. In estimation 5.6 (Figure 6.12), all assumed parameters were deviated approximately 10% from their actual value in simulation. This did not seem to significantly affect the quality of estimation, but analysis with more extreme deviations should be applied to test this further.

6.5. Discussion & areas of improvement

Although the benchmark estimations were already reasonably successful in showing potential in parameter estimation with a DEKF, they also identified numerous issues and weaknesses of the current implementation.

At least for the current algorithm and initialisation settings, estimating all parameters simultaneously proves to be impossible. This even holds for the ideal scenario of estimation 2.1, with an SI system and no remnant (Figure 6.6). Especially when estimating two time-delay parameters at once, at least one is shown to diverge and break the entire estimation effort.

Secondly, the time traces are still relatively erratic. Refinements to the algorithm or its initialisation are necessary. Especially in the case of simulations with remnant, large variations can be observed (e.g. estimations 5.1, 7.1 and 8.1, Figures 6.9, 6.14 and 6.15). It seems that the algorithm tries to compensate remnant noise by adjusting parameters, especially gains K_p and K_f.

This volatility poses a problem in the identification of time-varying behaviour. This becomes very obvious in estimation 7.1. In the case of τ_f,the volatility is bounded enough to represent the actual parameter variation relatively clearly. For K_p, however, the contrary is true. The variations are in this case so large, that the sigmoidal step can hardly be recognised. Similarly, parameters K_f and $\omega_{b,f}$ can hardly still be recognised as being constant from their identified time trace.

A third phenomenon was observed very clearly in estimation 6.1 (Figure 6.13). When the DEKF estimates a sigmoidal parameter jump, its effects ripple through to other parameters as well. It could be the case that this issue can be resolved similarly to the previous point. After all, both relate to the number of parameters that can adapt in a single or a few time steps.

A final observation was made regarding the implemented suspension time. In a number of cases, the estimated preview time overshot the suspension time (e.g., estimations 5.4 and 6.1). This in turn caused instability and divergence of the estimation process. Increasing suspension time and thus adding more margin resolved this issue, as shown by estimations 5.4 and 5.5 (Figures 6.10 and 6.11). As already described in Subsection 5.1.2, making the suspension time too large results in a loss of precision. Hence, the suspension time remains a setting to tune carefully.

7 Conclusions & Recommendations for Further Research

This research aims to identify time-varying, adaptive human control behaviour in preview tracking tasks. To achieve this, Kalman filters are applied to the preview model of Van der El et al. (2016). The main advantage of this model over others, is that its parameters have a physical meaning with well known values and limits for human operators. It also consists of a realisation of minimal order, which ensures that the parameters can be identified uniquely.

On the side of estimation with Kalman Filters, the Dual Extended Kalman Filter (DEKF) was a promising candidate in particular, since it has been applied successfully to compensatory tracking under similar conditions before by Popovici et al. (2017). The DEKF is suited to estimate delay parameters, which form a vital part of preview tracking. Moreover, the method has been successfully applied to real human data.

So far, a simulation framework has been set up, capable of simulating realistic, time-varying control behaviour in preview tracking tasks. The simulation incorporates remnant noise according to a first-order model by Levison et al. (1969). Besides, the Dual Extended Kalman Filter was implemented to identify the parameters of this tracking behaviour. To ensure maximum likelihood of convergence, the HO model was specifically formulated as a minimal order state-space system.

With this software, a set of eight benchmark simulations were constructed. These build up systematically to test the potential of the DEKF for increasing complexity and realism. It was shown that estimation using the DEKF is possible for the preview model, excluding the neuromuscular parameters and the feedback time delay. This is the case for simulations with both SI and DI dynamics, relatively low amounts of remnant, and time-varying parameters.

By *suspending* the simulation for τ_s seconds, effectively redefining the start time of measurement, the preview time can successfully be estimated as a regular, non-negative delay parameter.

The complexity of the problem did have to be reduced by excluding a subset of parameters from the estimation process. Parameters that were deemed suitable, are the neuromuscular parameters and the feedback time delay. Earlier research suggests that these parameters are likely to stay constant for a given experiment, as they describe the physical limitations of the human operator (Zaal, 2016). Especially taking out the feedback delay greatly improved the chances of convergence. Estimating two delay parameters simultaneously proved to be too ambitious, at least with the settings as applied during benchmarking.

Looking back at the research questions of Chapter 3, questions 1a,b,c and 2b,c have so far been answered, all with positive outcome. These questions cover the simulation of time-varying preview behaviour as well as the potential of convergence using a DEKF for parameter estimation.

A partial or preliminary answer was found for questions 2a,d,e and 3b,c regarding the sensitivity of filter initialisation and accuracy of the outcome. This does mean that, to find a satisfactory answer to these and other remaining questions, additional research is still needed. The next section will outline what steps are still to be taken as part of this thesis work. Thereafter, the final section of this report will be devoted to sketching the outlines of the work that could follow after successful completion of this thesis.

7.1. Proposed plans for research continuation

Moving forward from the benchmark phase, the main goal is to stretch the performance of parameter estimation with the DEKF in preview tracking as far as possible. Several areas were identified that can still be improved upon:

1. The time traces of the parameters should be as smooth as possible. This means that ideally, a clear distinction can be made between time-constant and time-varying behaviour.
2. The estimation process should be robust, with minimal chance of diverging.
3. The effects of the identification of a time-varying parameter on the other parameters should be as minimal as possible.
4. The amount of remnant that is added to the HO input, should be increased to more realistic levels.
5. The number of parameters that can be identified simultaneously, should ideally be as high as possible.

Firstly, the steps to improve estimation results will be discussed in the next subsection. Thereafter, more extensive research will be done using the sensitivity simulations, discussed in Subsection 7.1.2. Finally, the tested method will be applied to experimental data. What data will be used is explained in Subsection 7.1.3.

7.1.1. Minimizing the parameter estimation error

The next step to take in this research, is to improve the match between the estimated parameters and their actual values. In order to assess the quality of a parameter estimation, different metrics are proposed. A good candidate is the *normalised root mean square error* (NRMSE). It is computed as in Equation (7.1), by taking the RMS of the difference between actual parameter values θ_{true} and their estimated quantities θ_{DEKF}, and dividing it by the mean true parameter. The lower this value, the closer the estimated parameter is to the actual parameter value.

$$\text{NRMSE} = \frac{1}{\bar{\theta}_{\text{true}}}\sqrt{\frac{1}{N_{\text{meas}}}\sum_{i=0}^{N_{\text{meas}}}\left(\theta_{\text{DEKF}}(i)-\theta_{\text{true}}(i)\right)^2} \tag{7.1}$$

A second metric that can be used for the evaluation of an estimation, is the *variance accounted for* (VAF), also used by Van der El et al. (2016). This value indicates how closely the outcome of the identified HO ($\hat{u}$) matches the actual stick input u from a simulation or experiment. In order to compute the VAF, the simulation will be repeated using the original target signals and the parameter traces as estimated by the DEKF.

$$\text{VAF} = \left(1 - \frac{\sum_{k=1}^{N_{\text{meas}}}\left|u(k)-\hat{u}(k)\right|^2}{\sum_{k=1}^{N_{\text{meas}}} u^2(k)}\right)\times 100\% \tag{7.2}$$

Similarly, the VAF can also be calculated of the previous, resimulated HO stick input $\hat{u}$, and a new simulation with the original smooth parameter traces, but without any remnant noise. If this VAF ends up lower than the one previously calculated, that would support the hypothesis that the erratic parameters estimated by the DEKF explain noisy HO stick inputs due to remnant.

To reduce the variance of the estimated parameters and increase the chances of convergence, there is a number of options to look into:

1. First and foremost the initialisation of the DEKF can be finetuned. The main parameter that regulates the variance of the estimated parameters are process noise covariance matrices Q_s and Q_p. Furthermore, the sensor noise matrix R, has an influence on the balance between previously identified parameter values and new incoming data.
2. To avoid divergence of the filter, limits to estimated parameters can be introduced in the DEKF. After all for most parameters, human limits and capabilities have been extensively researched. Simon (2010) has made an overview of several augmentations to Kalman Filters that can incorporate (soft or hard) state constraints to parameters or update rates in various ways. This enables the user to exploit the additional knowledge in favour of better filtering performance.
3. Finally, parameter traces could be smoothed afterwards. This option of a post-processing filter comes with several considerations. First of all, it introduces the explicit assumption into the estimation process that parameter changes are gradual. Secondly, post-processing comes with a trade-off. Either smoothing introduces delay in estimation process, or the algorithm loses its real time applicability if the delay is compensated for afterwards.

7.1.2. Sensitivity simulations

Before moving to experimental human operator data, the DEKF will be scrutinised with an additional set of simulations. These *sensitivity* simulations test a wider range of conditions and will therefore include

- more diversity in time-variance: different sinusoidal and sigmoidal variations,
- the same forcing functions shifted by phase, to rule out the effect of timing of time-varying behaviour at a certain point in the target,
- more forcing function realisations (the realisations from the 4 rad/s bandwith case of Van der El et al. (2019b) that have not yet been used),
- more remnant realisations and higher, more realistic remnant powers.

In the process of estimation, the limits and sensitivity of the DEKF initialisation will be researched.

7.1.3. Experimental data

As a final step, the fully tested method will be applied to actual experimental control data. There are no plans for an extensive, dedicated experiment in the current thesis research. The main reason for this is the time constraint. Properly designing, executing and analysing an experiment is a time-consuming process. The thorough preparation that this requires, is an excellent topic for future research. There are however some ideas on what this experiment might look like. These thoughts can be found in the next section.

Luckily, there is enough suitable data available from previous experiments to apply parameter estimation using a DEKF on human operator data. A suitable candidate for supplying actual measurements is the research done by Van der El et al. (2018). In this research, human operators were tested for multiple preview times. Preview time is the parameter with the highest priority in the current thesis. Thus, using these experiments allows for extensive testing of the accuracy of the DEKF for different preview scenarios. An added benefit of using this research, is the fact that the constant parameters that were identified, are particularly well backed up by a theoretical expectation. This means that the identified values can serve as a good reference.

Additionally, an explicit quantitative comparison with the results of Plaetinck et al. (2018) and Rojer et al. (2019) could be made, using their experimental data.

7.2. Beyond this thesis

As expressed at the start of this thesis, the main goal of this research is to contribute to the identification and understanding of time-varying, adaptive human control behaviour. With a working identification method, there are numerous interesting leads for future research to follow up on.

Firstly, one can think of extending the research into the *theoretical background of time-varying behaviour in preview tracking tasks.* The currently researched method could be employed to investigate how constant human operators actually are under experiment conditions that are currently held for 'constant'. Testing this explicitly would validate one of the major assumptions that has so far been common for research in the field of cybernetics (Mulder et al., 2018).

Moreover, one could optimize the progression of one or more parameters over time in an experiment run, exposing how much there is to gain when human operators vary things such as their preview time (τ_f). This is similar to what Van der El et al. (2018) did for compensatory tracking with constant parameters.

Secondly, this thesis could form the basis for a (range of) new experiment(s). A main topic of interest would be to invoke time-varying behaviour and analyse how HO's notice and adapt to these changing conditions. Conditions to vary could for example be

- Forced preview. One could think of time-varying variations to an idealised tracking task such as Van der El et al. (2018), or a realistic scenario (Land and Lee, 1994).
- Gradual CE transformation, such as used by Plaetinck et al. (2018) or Zaal (2016)

An experiment could also give insight in the process of acquiring a new skill. The progression from a novice to an experienced operator, as well as the immediate differences in adaptability between skilled and unskilled operators could be studied in more detail.

Thirdly, after ample testing, this method could form the basis for adaptive steering assistance, such as an autopilot or haptic feedback. Finally, the approach of simulation and testing that was presented in this thesis could also be applied to other algorithms for time-varying identification. As indicated in Section 2.2, there are other promising methods, such as the UKF (Rojer et al., 2019), or LPV interpolation (Pronker et al., 2017).

A
Derivation of the Sigmoidal Parameter Adaptation Formula

In previous research, a sigmoidal adaptation is often used to simulate or invoke time-varying systems or behaviour (Plaetinck et al., 2018; Van Grootheest et al., 2018; Zaal, 2016). With a sigmoidal function, the point in time at which the change occurs can be easily defined (T_c), well as how gradual this change is. This last factor is often expressed as the maximum rate of change (G in Equation (A.1)). However, for HO parameter adaptations, it is deemed much more intuitive to express this rate of change as the time it takes to go through this transition, ΔT.

$$\theta_i(t) = \theta_{i,1} + \frac{\theta_{i,2} - \theta_{i,1}}{1 + e^{-G(t-T_c)}} \tag{A.1}$$

The original Equation (A.1) is therefore slightly rewritten in order to have ΔT appear in the formula explicitly. This derivation is started from the basic formula of a sigmoid function (Equation (A.2), top row). As Figure A.1 shows, this standard sigmoid ranges from 0 to 1, and is point-symmetric about $(0, 0.5)$.

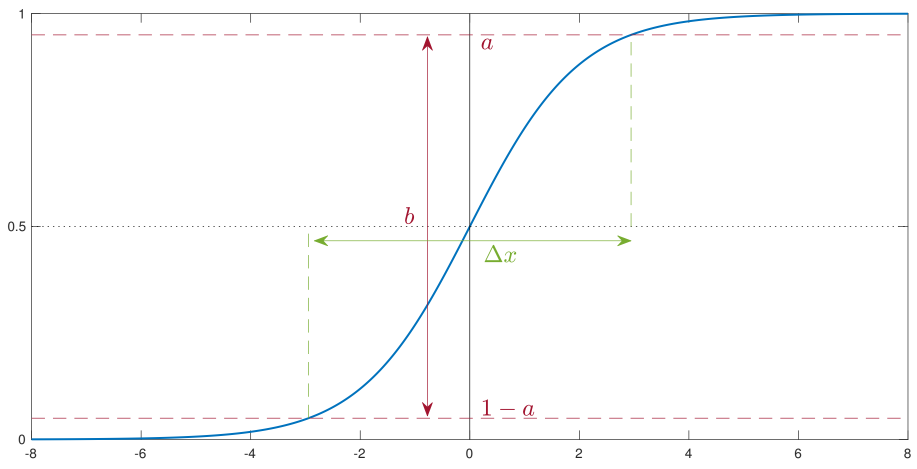


Figure A.1: Sigmoid function $y = \frac{1}{1+e^{-x}}$ with transition percentage b and transition domain Δx

To define the total transition domain Δx, one needs to know the x-value of a certain threshold a. The corresponding domain to go from $1-a$ to a, Δx, can then be found by simply doubling this x-value because of symmetry around the y-axis. At the same time, symmetry can also be used to rewrite the upper threshold a to the percentage of the transition that falls within Δx (b, see Equation (A.3)). This starts with the realisation that $1-a$ is *half* the percentage that is *not* considered for the transition: $(1-b)/2$.

$$\begin{aligned} y = \frac{1}{1+\mathrm{e}^{-x}} &= a \\ \mathrm{e}^{-x} &= \frac{1}{a} - 1 = \frac{1-a}{a} \\ x &= -\ln\frac{1-a}{a} = \ln\frac{a}{1-a} \\ \Delta x &= 2\ln\frac{a}{1-a} = 2\ln\frac{b+1}{1-b} \end{aligned} \tag{A.2}$$

$$\begin{aligned} 1-a &= \frac{1-b}{2} \\ a &= \frac{b+1}{2} \\ \rightarrow \frac{a}{1-a} &= \frac{2a}{2-2a} = \frac{b+1}{2-(b+1)} = \frac{b+1}{1-b} \end{aligned} \tag{A.3}$$

To make the change b within ΔT seconds, the original function needs to be scaled relative to the y-axis. Subsequently, the new function is shifted along the t-axis to the correct transition point T_c. Finally, the function is scaled and shifted along the y-axis to arrive at the final expression for the required sigmoidal parameter. This is done in Equation (A.4). By comparing the final result with the original equation, one can observe that $G = \frac{\Delta x}{\Delta T} = \frac{2}{\Delta T}\ln\frac{b+1}{1-b}$ for any given transition percentage b and transition time ΔT. So for 99% of the total transition to occur within ΔT, we simply have to substitute $b = 0.99$.

$$\begin{aligned} y = \frac{1}{1+\mathrm{e}^{-x}} &\rightarrow y = \frac{1}{1+\mathrm{e}^{-\frac{\Delta x}{\Delta T}t}} = \frac{1}{1+\mathrm{e}^{-\left(\frac{2}{\Delta T}\ln\frac{b+1}{1-b}\right)t}} \\ &\rightarrow y = \frac{1}{1+\mathrm{e}^{-\frac{2}{\Delta T}\ln\frac{b+1}{1-b}(t-T_c)}} \\ &\rightarrow \theta_i(t) = \theta_{i,1} + \frac{\theta_{i,2}-\theta_{i,1}}{1+\mathrm{e}^{-\frac{2}{\Delta T}\ln\frac{b+1}{1-b}(t-T_c)}} = \theta_{i,1} + \frac{\theta_{i,2}-\theta_{i,1}}{1+\mathrm{e}^{-\frac{2\ln 199}{\Delta T}(t-T_c)}} \end{aligned} \tag{A.4}$$

B
Unsuccessful Simulation Efforts

This appendix lists some failed attempts that were done to simulate time-varying tracking behaviour. The methodology behind the code and reasons for failure or discontinuation are included for future reference of students and researchers working on the current or similar code. Some elements of these efforts are still present in the code, and are still used, for example for verification purposes.

B.1. Time-varying `Simulink` model

The first simulation effort was constructed by means of a time-varying `Simulink` model. `Simulink` has many options for linear models with time-varying parameters, making it relatively easy to get a quick implementation.

This effort was unsuccessful because:

- Solving at required sample time instances: It proves to be difficult to precisely instruct `Simulink` to output results at fixed time instances, which is a requirement for the subsequent parameter estimation. When trying to solve for a fixed time step, the simulation would raise the following warning upon user interruption: *The specified buffer for 'Variable Transport Delay' was too small. During simulation, the buffer size was temporarily increased.* Efforts to resolve this warning were not successful.
- The simulation does not converge for double integrator dynamics.

It might well be possible to solve both of these problems. However, since there was a desire to perhaps also make the simulation available in Python in the future, working with `Simulink` was discontinued. The code as it stands now can be relatively easily translated to a full Python version. Moreover, implementing the actual differential equations in MATLAB also increased the understanding of the problem and solution method. This as opposed to making a 'black box' `Simulink` model, that would be difficult to understand and debug once it would produce inexplicable results.

B.2. Matlab `lsim` simulation per time step

A second effort to simulate the nonlinear system, had a similar layout to the eventual (current) simulation. During each time step, all parameters are kept constant, so the resulting system is in fact linear. Hence, its dynamic response can be computed with a simple `lsim` command for the duration of that time step. By executing each subsequent time step, with the final state of the previous step as input conditions, the time signals with varying parameters would emerge.

This effort was unsuccessful because:

- `lsim` startup time. `lsim` is really quick at generating the response for long simulations of a constant linear model. However, the function does have a considerable startup time. Having to start `lsim` every time step again, you do run into problems of extremely slow code progression.

A verification of the results for this method did show, however, that the resulting time traces are in fact correct. This setup was therefore reiterated to the current method, where the model is solved in each time step with `ode45`, using an automatically generated nonlinear state-space representation of the block diagram. By doing it this way, the startup time for `lsim` in each step is avoided, resulting in much better performance. For more information, see Section 4.2.

B.3. Code generation with readable parameters

The last iteration included a minor difference compared to the final layout. In this pipeline, the code also auto-generated files with the differential equations, solved by `ode45`. The parameters inputs to the function would be presented as two vectors though, one containing the parameter names and one containing their values. This format follows the example of Daan Pool's original MLE code. In the execution of the function, each parameter name/value pair would be transformed into a variable using a `for`-loop. It was found that evaluating this loop for each time step, resulted in code performance that was comparable to the `lsim` code described in the previous subsection.

C
State-Space System of HO Preview Model

This appendix details the nonlinear state-space equations that were used in the DEKF, analogous to Appendix A of Popovici et al. (2017) that applied a DEKF to compensatory tracking. The model behind these equations is depicted in Figure C.1. The formation of this model and the state-space equations was explained in Section 5.2, particularly Subsection 5.2.1. Subsection 2.2.2 contains a more elaborate discussion of the DEKF. In order to convert the (exponential) apparent preview delay ($\tau_f^\star$) and time delay (τ_v) from Figure C.1 to state-space, a third order Padé approximation was used. For more information, see Subsection 4.1.3.

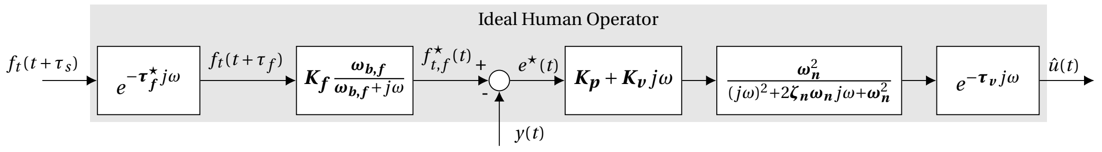

Ideal Human Operator
$f_t(t+\tau_s)$
$e^{-\tau_f^\star j\omega}$
$f_t(t+\tau_f)$
$K_f \frac{\omega_{b,f}}{\omega_{b,f}+j\omega}$
$f_{t,f}^\star(t)$
+
-
$e^\star(t)$
$y(t)$
$K_p + K_v j\omega$
$\frac{\omega_n^2}{(j\omega)^2+2\zeta_n\omega_n j\omega+\omega_n^2}$
$e^{-\tau_v j\omega}$
$\hat{u}(t)$

Figure C.1: Isolated block diagram of the HO in preview tracking tasks with parameters, far view only (Van der El et al., 2016)

$$\boldsymbol{x_s} = [x_{s,1} \;\; x_{s,2} \;\; x_{s,3} \;\; x_{s,4} \;\; x_{s,5} \;\; x_{s,6} \;\; x_{s,7} \;\; x_{s,8} \;\; x_{s,9} \;\; K_p \;\; K_v \;\; K_f]^T \tag{C.1}$$

$$\boldsymbol{\theta} = [\omega_n \;\; \zeta_n \;\; \tau_v \;\; \omega_{b,f} \;\; \tau_f^\star]^T \tag{C.2}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}}(t) = f\left(\boldsymbol{x_s}(t), f_{t,s}(t), y(t), \boldsymbol{\theta}(t)\right) + \boldsymbol{w_s}(t) \tag{C.3}$$

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{w_p}(t) \tag{C.4}$$

$$\hat{u}(t) = g\left(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t)\right) + v(t) \tag{C.5}$$

$$f\left(\boldsymbol{x_s}, f_{t,s}, y, \boldsymbol{\theta}\right) = \begin{bmatrix} x_{s,2} \\ x_{s,3} \\ x_{s,4} \\ x_{s,5} \\ -y - 120x_{s,1}\omega_n^2/\tau_v^3 - x_{s,2}(60\tau_v\omega_n^2 + 240\zeta_n\omega_n)/\tau_v^3 - \\ \cdots - x_{s,3}(12\omega_n^2\tau_v^2 + 120\zeta_n\omega_n\tau_v + 120)/\tau_v^3 - x_{s,4}(\omega_n^2\tau_v^3 + 24\zeta_n\omega_n\tau_v^2 + 60\tau_v)/\tau_v^3 - \\ \cdots - x_{s,5}(2\omega_n\zeta_n\tau_v^3 + 12\tau_v^2)/\tau_v^3 + 120x_{s,6}K_f\omega_{b,f}/\tau_f^{\star 3} - \\ \cdots - 60x_{s,7}K_f\omega_{b,f}/\tau_f^{\star 2} + 12x_{s,8}K_f\omega_{b,f}/\tau_f^\star - x_{s,9}K_f\omega_{b,f} \\ x_{s,7} \\ x_{s,8} \\ x_{s,9} \\ f_{t,s} - 120x_{s,6}\omega_{b,f}/\tau_f^{\star 3} - x_{s,7}(60\omega_{b,f}\tau_f^\star + 120)/\tau_f^{\star 3} - \\ \cdots - x_{s,8}(12\omega_{b,f}\tau_f^{\star 2} + 60\tau_f^\star)/\tau_f^{\star 3} - x_{s,9}(\omega_{b,f}\tau_f^{\star 3} + 12\tau_f^{\star 2})/\tau_f^{\star 3} \end{bmatrix} \tag{C.6}$$

$$g\left(\boldsymbol{x_s}, \boldsymbol{\theta}\right) = \begin{bmatrix} 120x_{s,1}K_p\omega_n^2/\tau_v^3 + x_{s,2}(120K_v\omega_n^2 - 60K_p\omega_n^2\tau_v)/\tau_v^3 + \\ \cdots + x_{s,3}(12K_p\omega_n^2\tau_v^2 - 60K_v\omega_n^2\tau_v)/\tau_v^3 - x_{s,4}(K_p\omega_n^2\tau_v^3 - 12K_v\omega_n^2\tau_v^2)/\tau_v^3 - x_{s,5}K_v\omega_n^2 \end{bmatrix} \tag{C.7}$$

In this research, parameters were kept constant to reduce problem dimension and increase chances of convergence. The parameter was then removed from $\boldsymbol{x_s}$ or $\boldsymbol{\theta}$. Symbolic occurrences in the equations above were replaced by the numerical value. Thus it was also removed from any Jacobians calculated during estimation.

D

Using `Simulink` to Construct Minimal Realisation State-Space Systems

In Subsection 5.2.1, some rules were established to convert simple MIMO block diagrams to a minimal realisation state-space system. However, this might not always be obvious, especially for complicated block diagrams. If finding a minimal realisation proves to be challenging, the procedure in MATLAB/`Simulink` explained below might provide a helpful tool. In global terms, this procedure works as follows:

1. Make an abstract, simplified version of your block diagram in `Simulink`, keeping the structure of nodes similar to the original diagram.
2. Populate each block with a second order transfer function, with unique prime coefficients. The highest order of each denominator can get the coefficient 1.
3. Use MATLAB's `linmod` command to transform the block diagram into a `sys`-object.
4. Print the state space in the Command window and retrace the individual $A-, B-, C-\&D$-matrices of each block in the original `Simulink` diagram.
    - If necessary, interchange states until they are in canonical order
5. Analyse the final result and think about if and how this is sensible (for example according to the logic of Subsections 4.2.2 and 5.2.1)

Figure D.1 shows the result of steps 1 and 2 for a simplification of Figure 5.3. Notice how this block diagram exactly copies the structure of Figure 5.3, except for the fact that directly adjacent blocks are merged. Moreover, all the coefficient terms in the transfer functions are unique primes. This makes it possible to trace back each coefficient in any state-space individually, by computing the prime factors of each value in the state-space matrices (in this case, all coefficients appear individually).

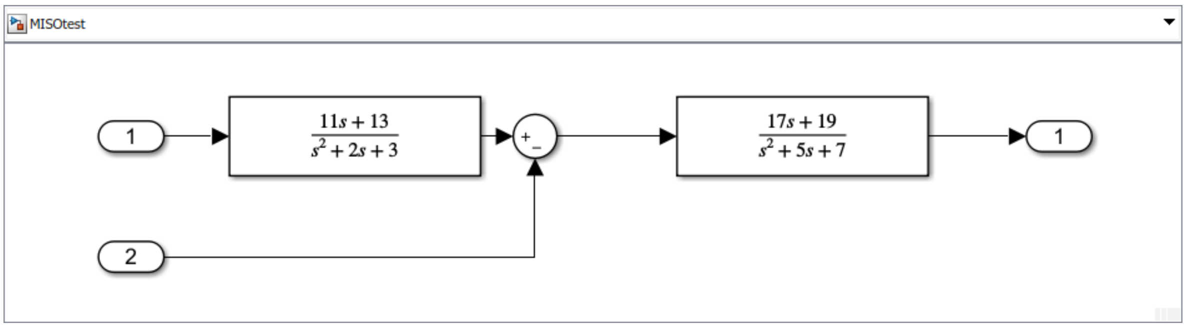


Figure D.1: Simplified block diagram of Figure 5.3 for the retrieval of transfer function coefficients in a state-space system

Equation (D.1) shows the result of the MATLAB command `sys = linmod('MISOtest')`, made with MATLAB release R2019b.

$$\texttt{sys.a} = \begin{bmatrix} -5 & -7 & 11 & 13 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -2 & -3 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \texttt{sys.b} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$
$$\texttt{sys.c} = \begin{bmatrix} 17 & 19 & 0 & 0 \end{bmatrix}, \quad \texttt{sys.d} = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{D.1}$$

It is easy to note that its contents match two stacked controllable canonical forms, if states x_1 & x_2 and states x_3 & x_4 are interchanged. To do this, interchange rows 1 & 2 and rows 3 & 4 in matrices `sys.a` and `sys.b`. Moreover, interchange columns 1 & 2 and columns 3 & 4 in matrices `sys.a` and `sys.c`. The final result should look like Equation (D.2). This state-space structure is exactly identical to the one represented by Equations (5.2) and (5.3), for which the logic behind it can be found in Subsection 5.2.1.

$$\texttt{sys.a} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -7 & -5 & 13 & 11 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -3 & -2 \end{bmatrix}, \quad \texttt{sys.b} = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$
$$\texttt{sys.c} = \begin{bmatrix} 19 & 17 & 0 & 0 \end{bmatrix}, \quad \texttt{sys.d} = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{D.2}$$

E Complete Results of Benchmark Estimations

This appendix displays the full time trace results of all benchmark estimations. A detailed description of these estimations and its settings can be found in Section 6.2.

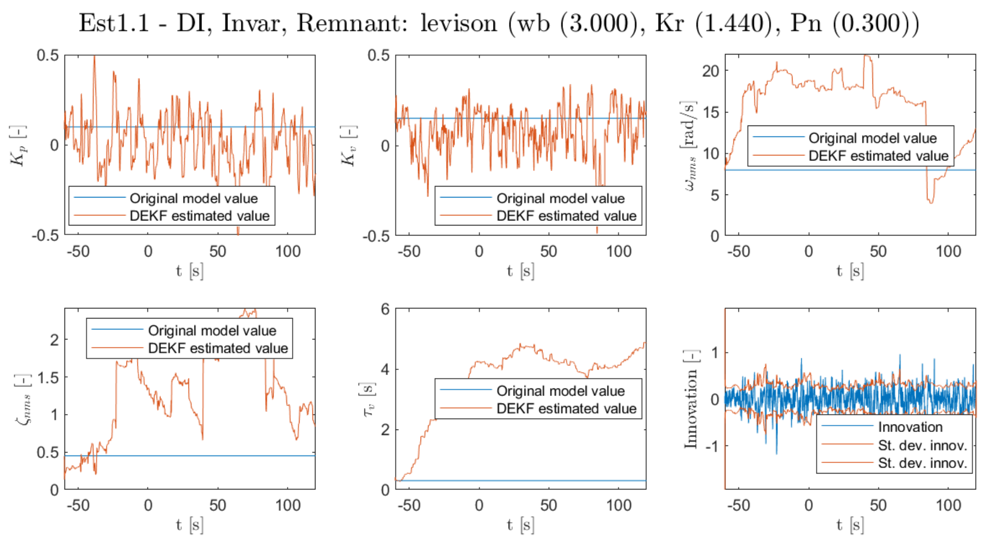


Figure E.1: Benchmark estimation 1.1

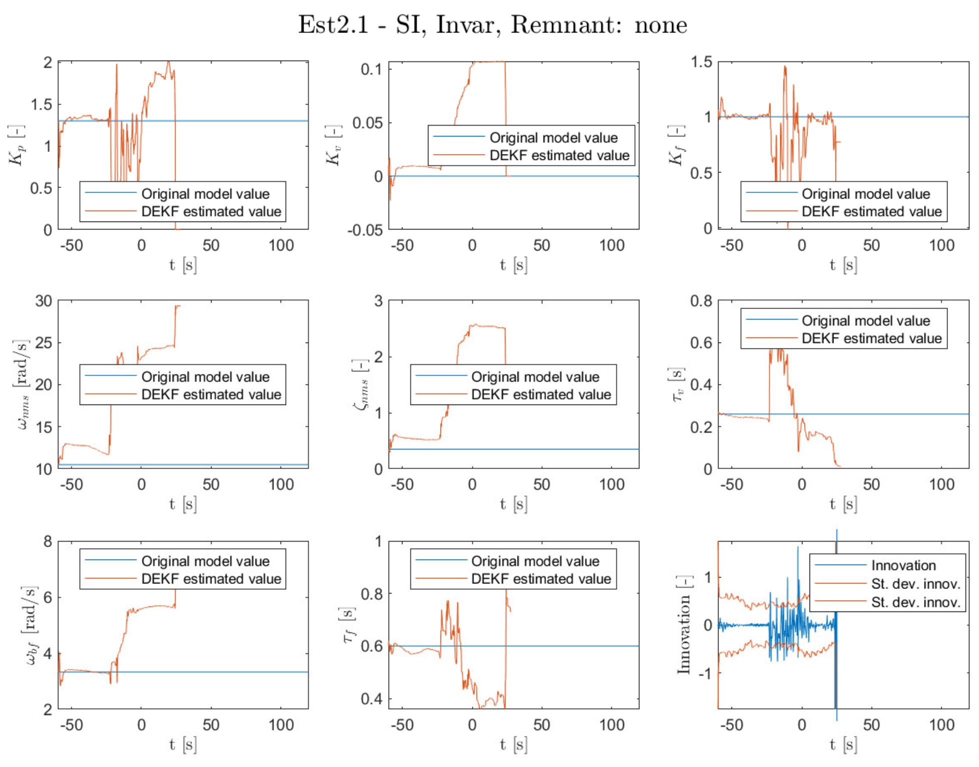


Figure E.2: Benchmark estimation 2.1

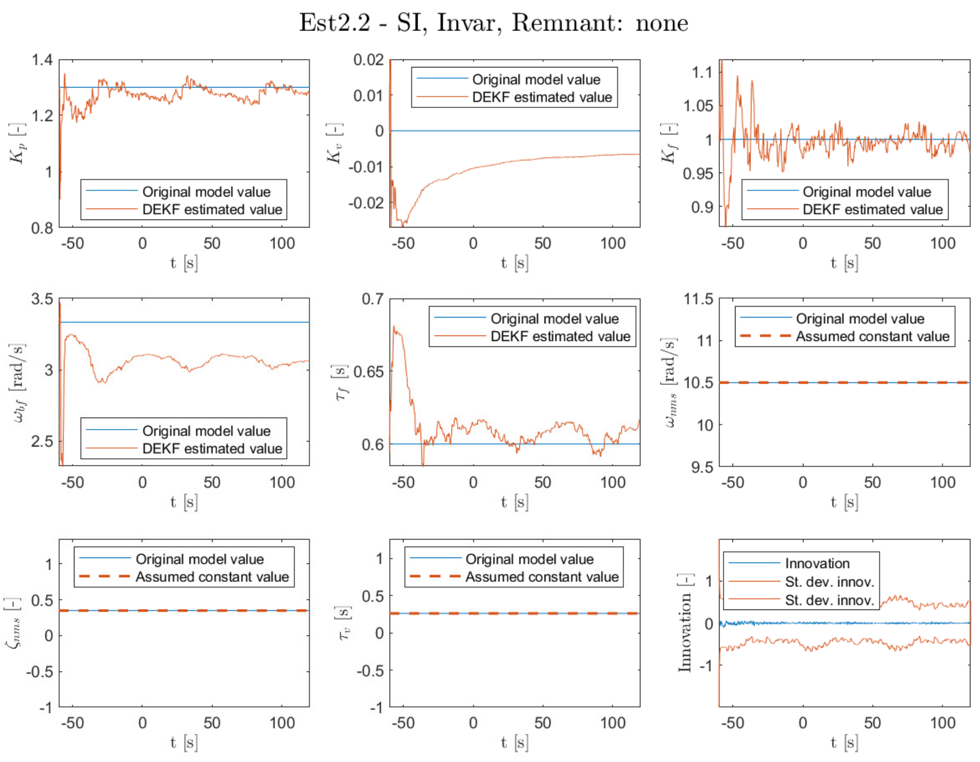


Figure E.3: Benchmark estimation 2.2

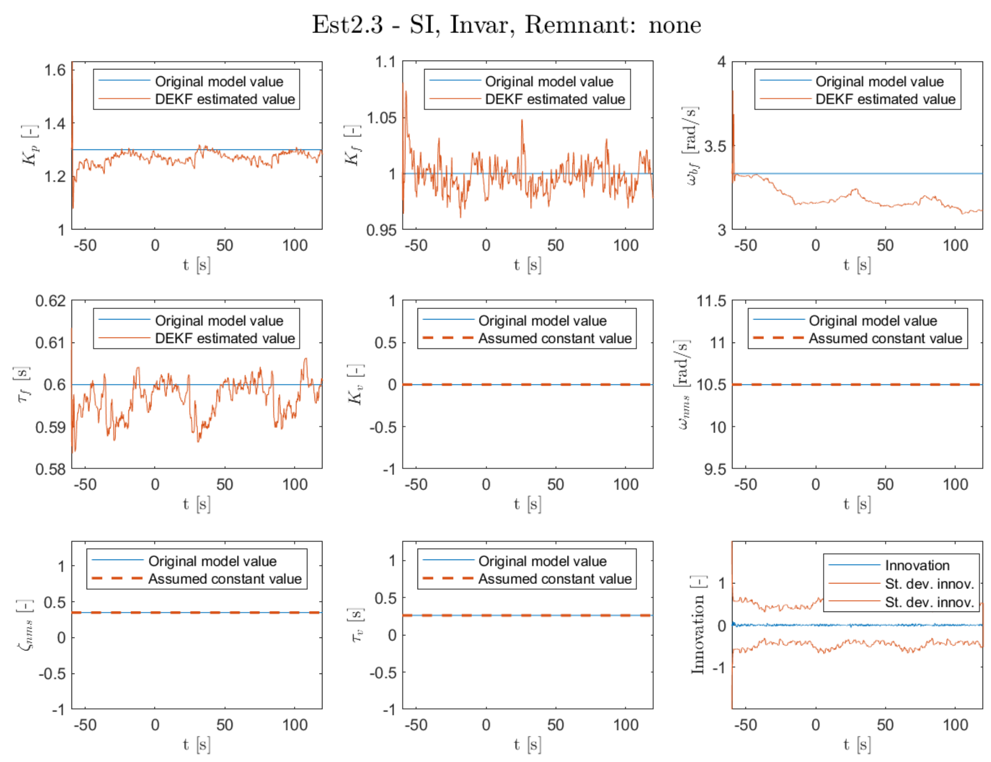


Figure E.4: Benchmark estimation 2.3

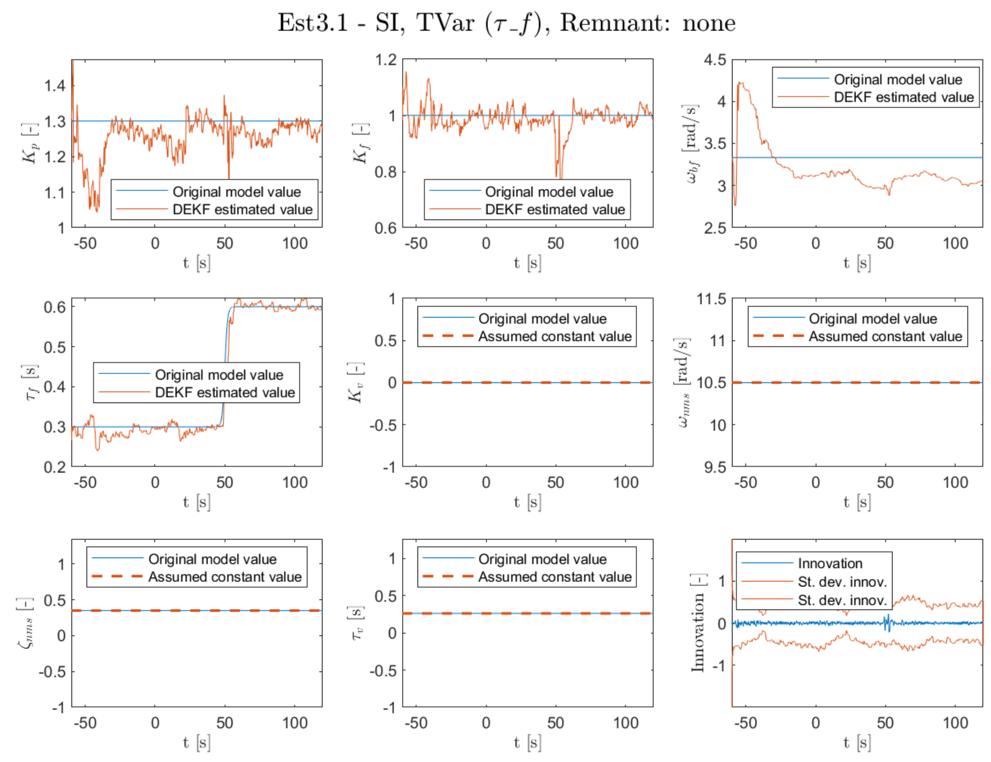


Figure E.5: Benchmark estimation 3.1

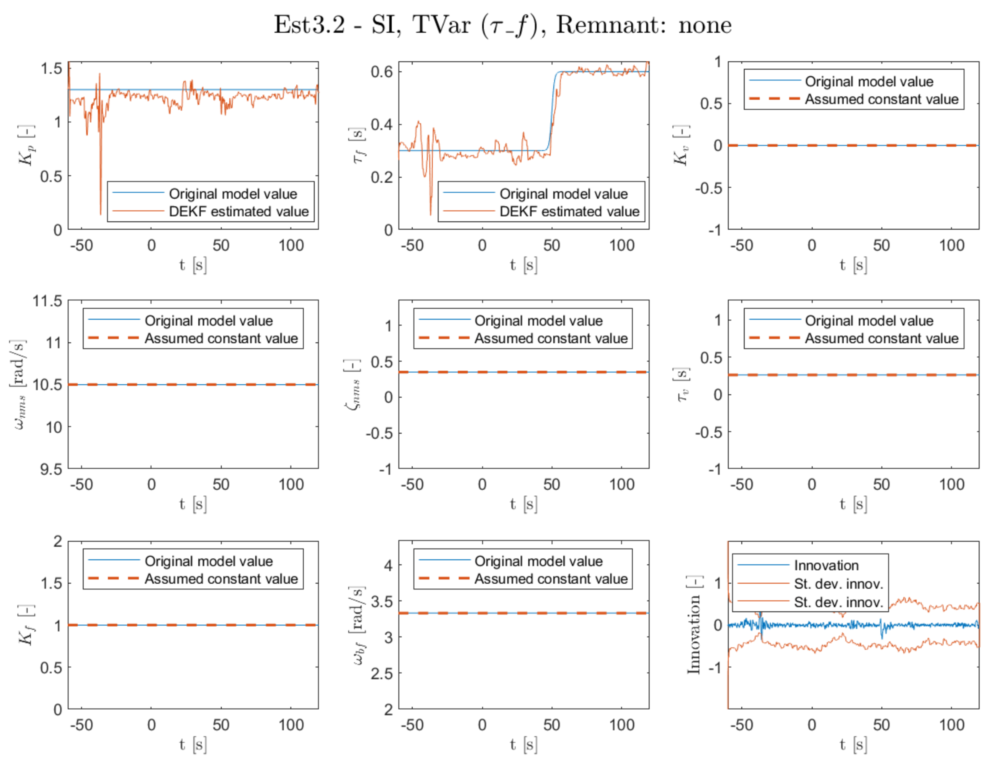


Figure E.6: Benchmark estimation 3.2

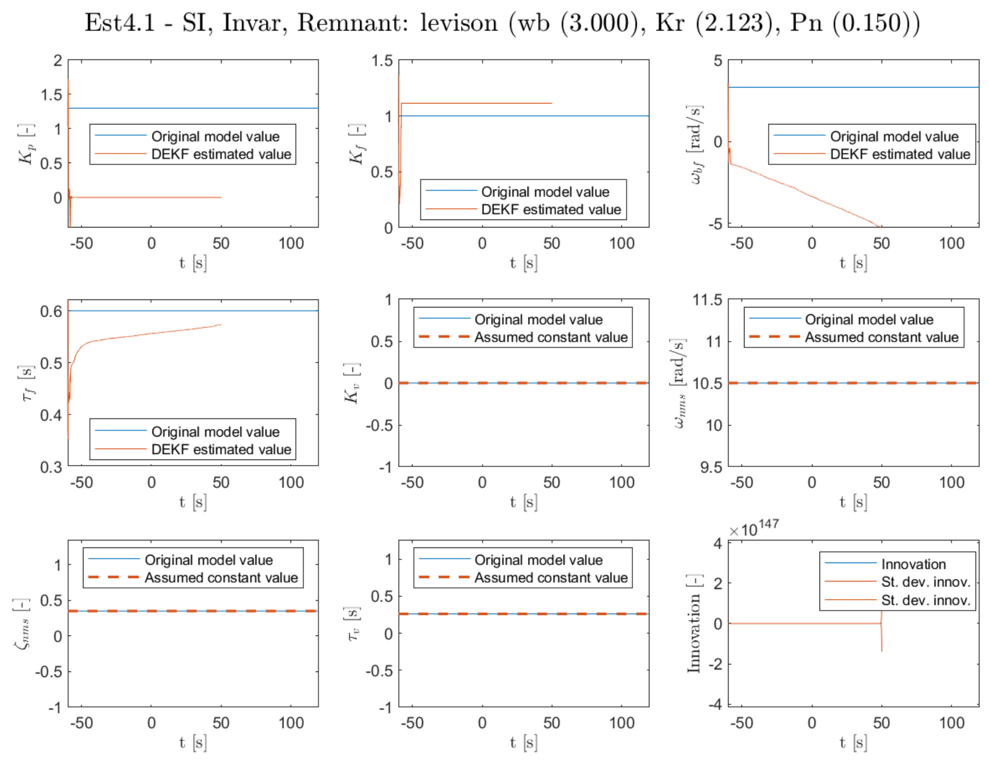


Figure E.7: Benchmark estimation 4.1

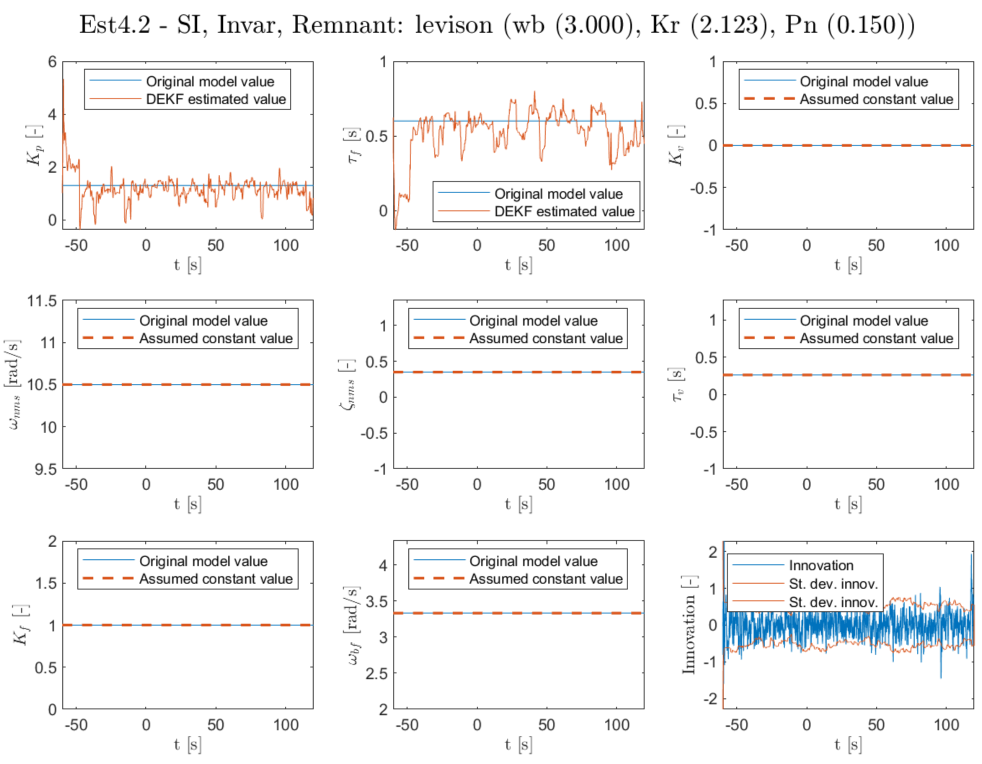


Figure E.8: Benchmark estimation 4.2

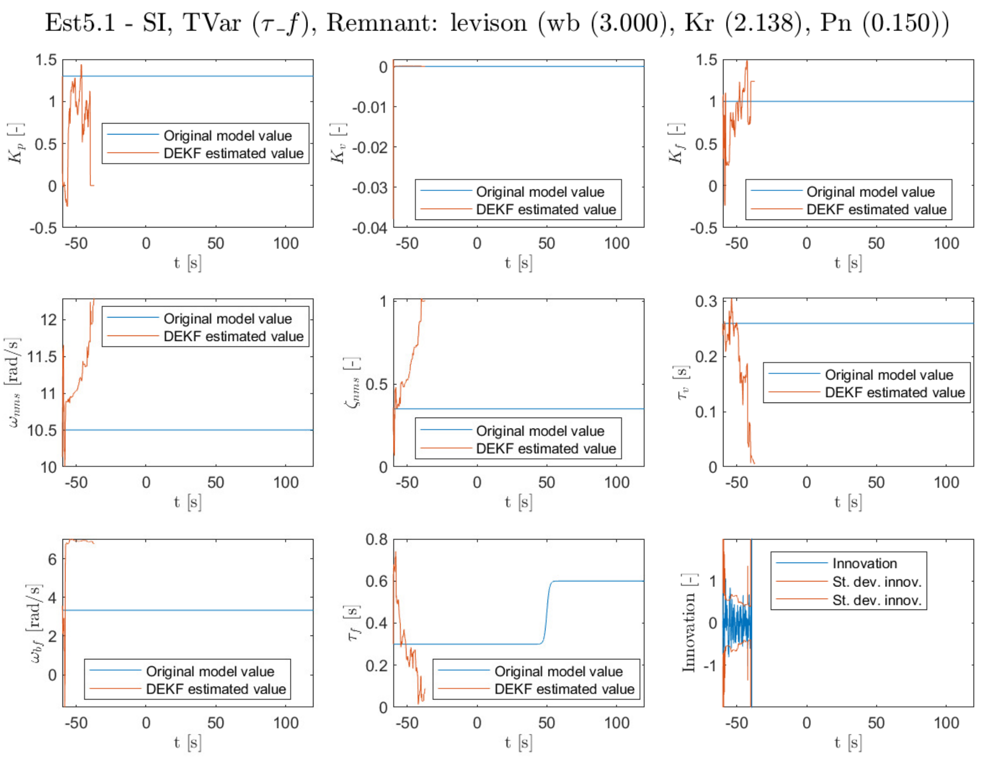


Figure E.9: Benchmark estimation 5.1

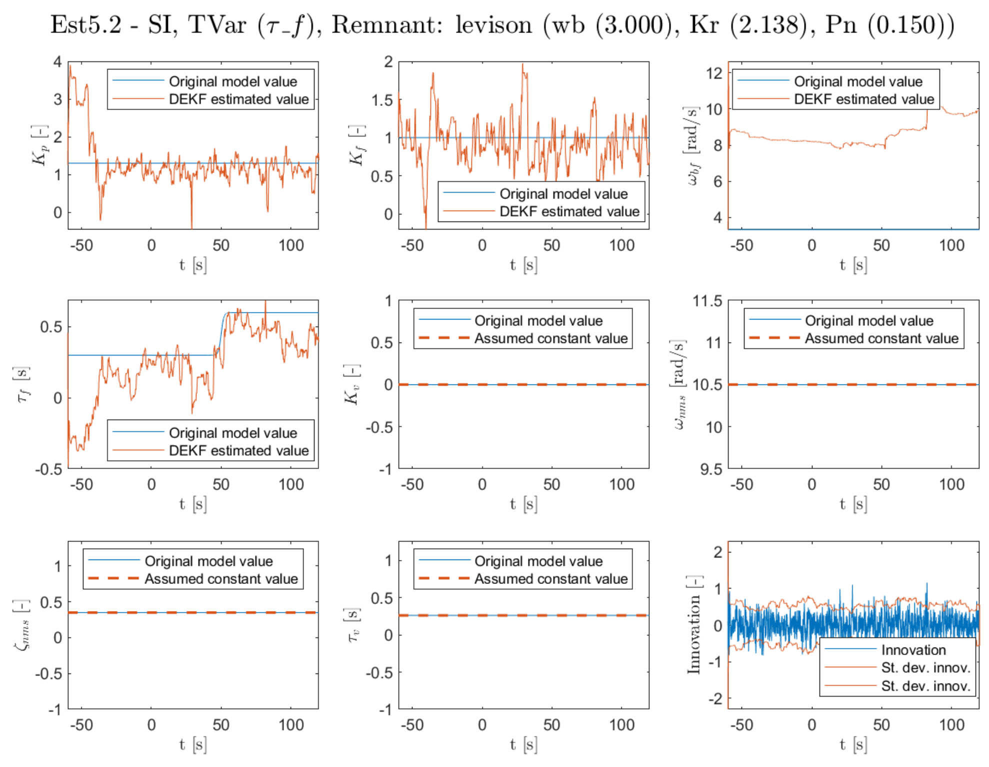


Figure E.10: Benchmark estimation 5.2

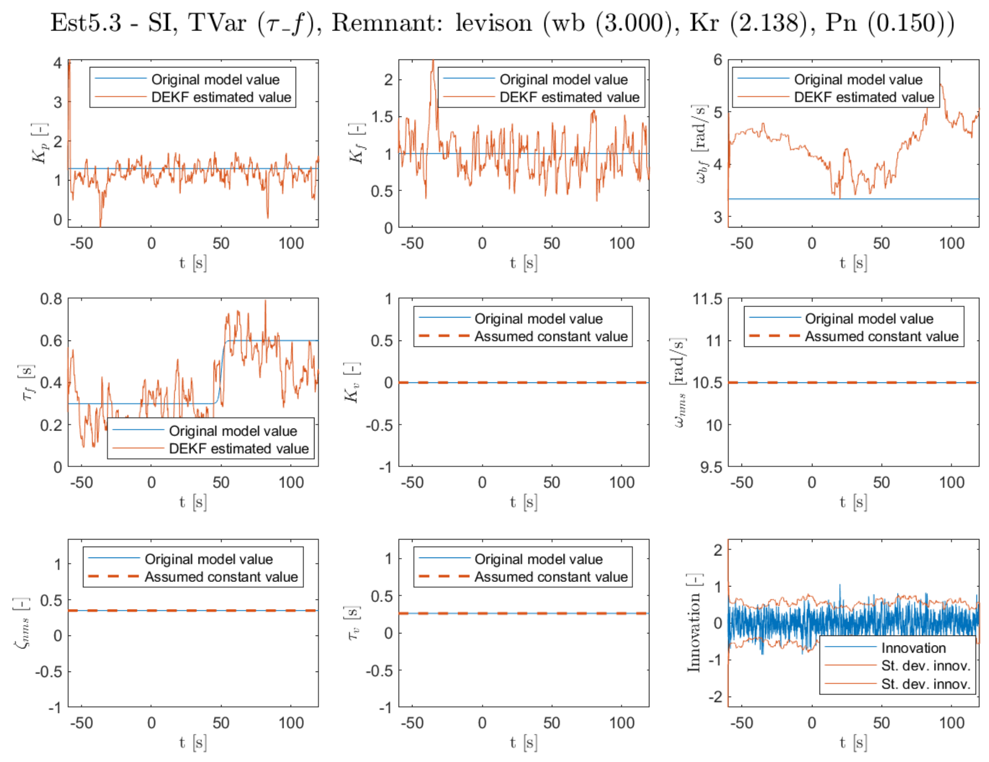


Figure E.11: Benchmark estimation 5.3

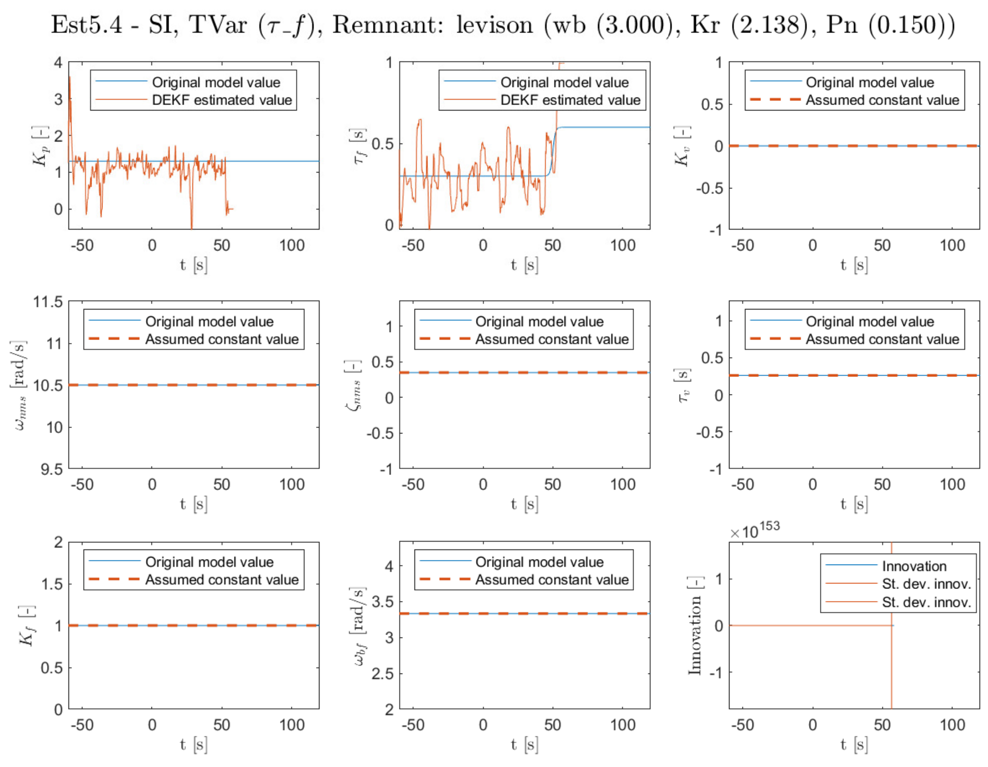


Figure E.12: Benchmark estimation 5.4

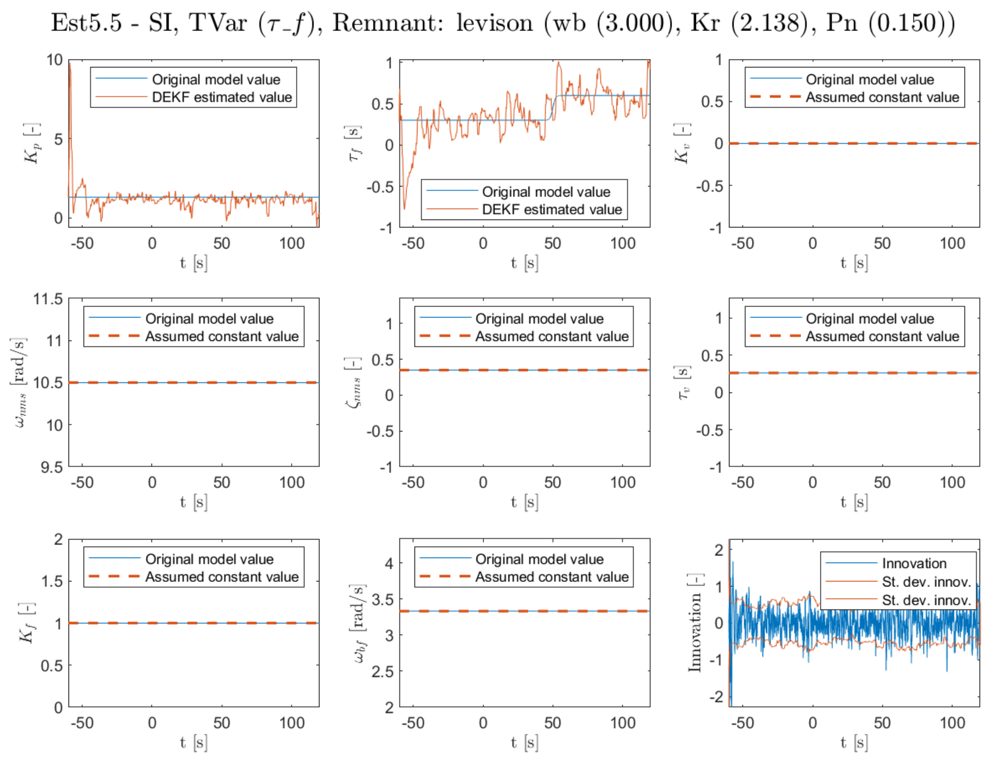


Figure E.13: Benchmark estimation 5.5

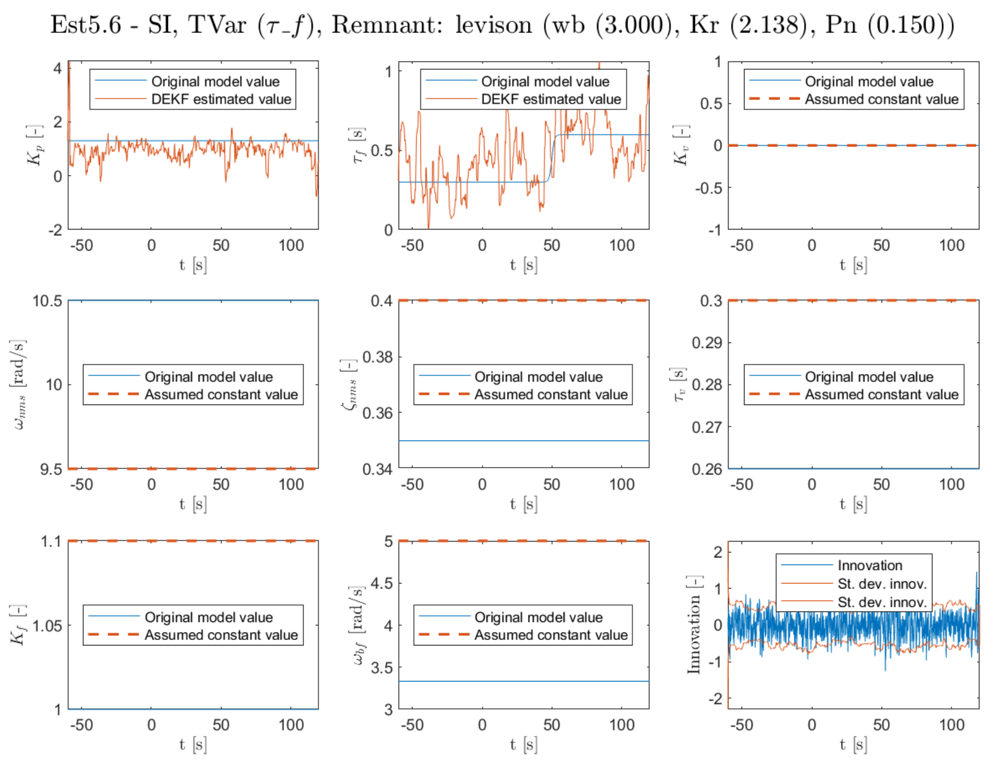


Figure E.14: Benchmark estimation 5.6

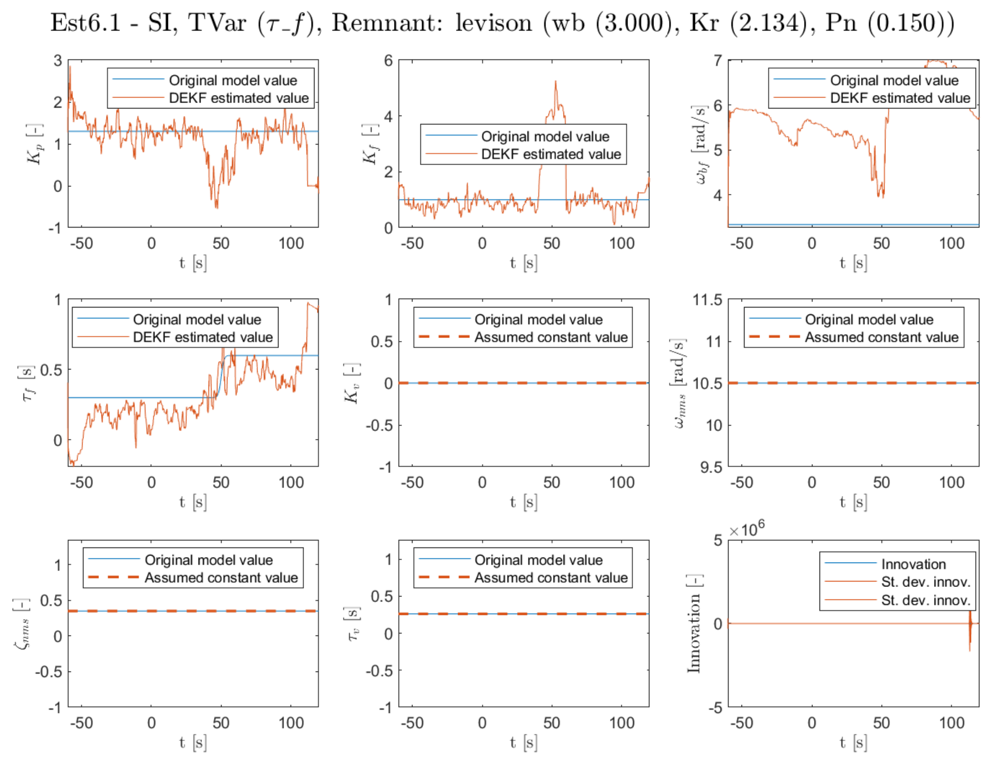


Figure E.15: Benchmark estimation 6.1

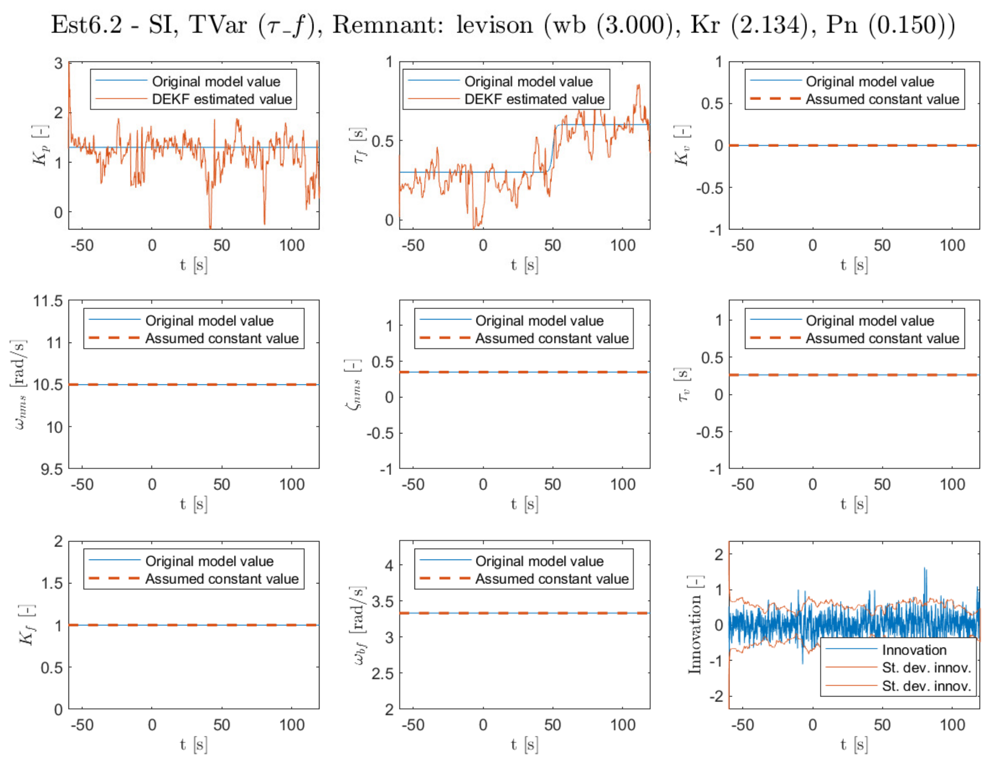


Figure E.16: Benchmark estimation 6.2

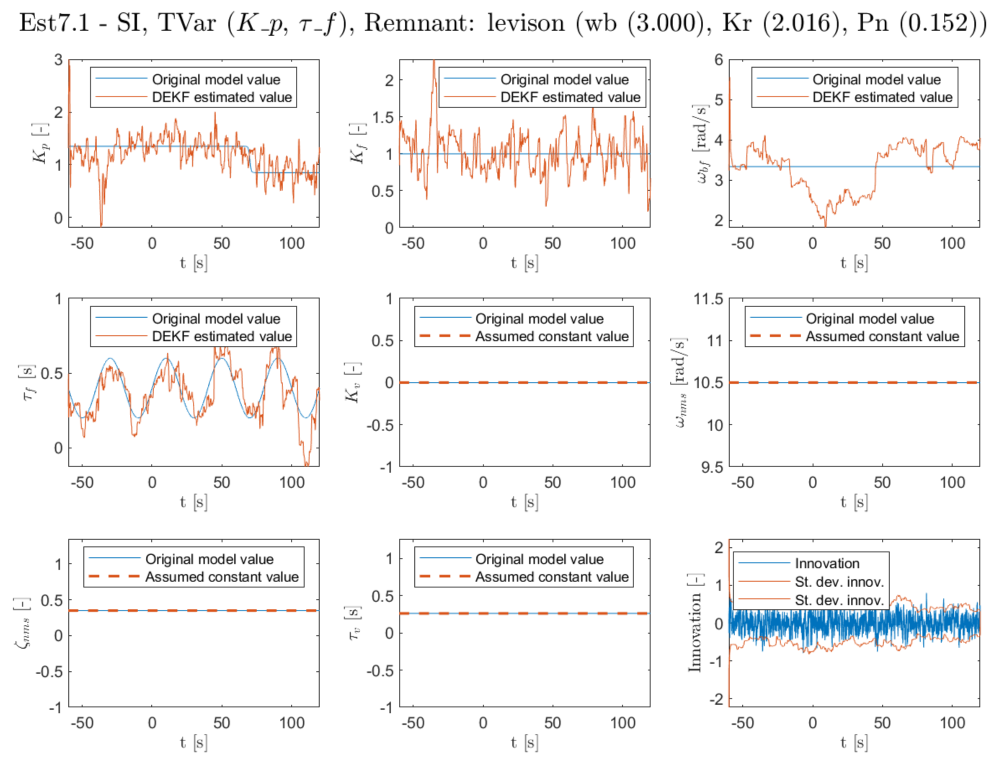


Figure E.17: Benchmark estimation 7.1

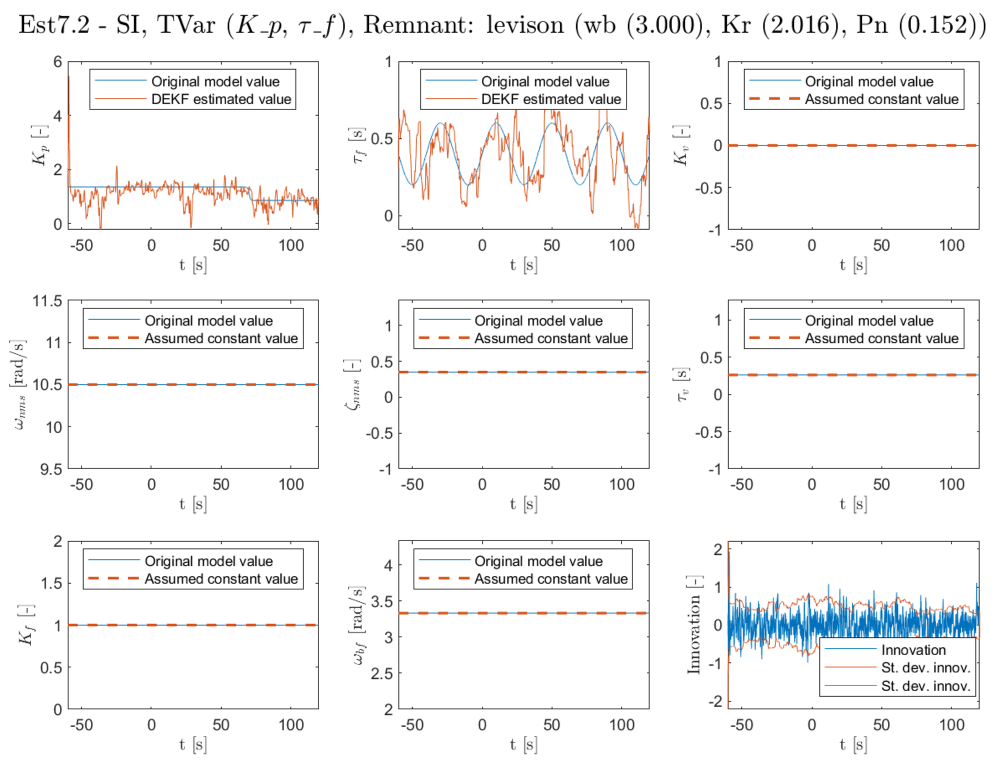


Figure E.18: Benchmark estimation 7.2

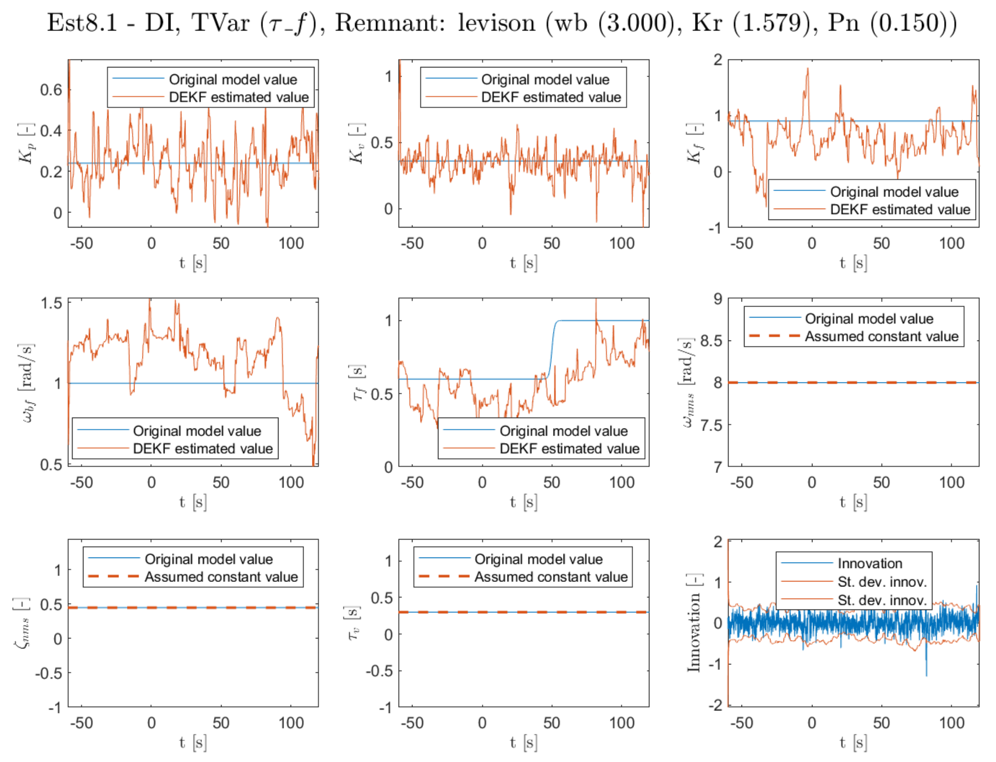
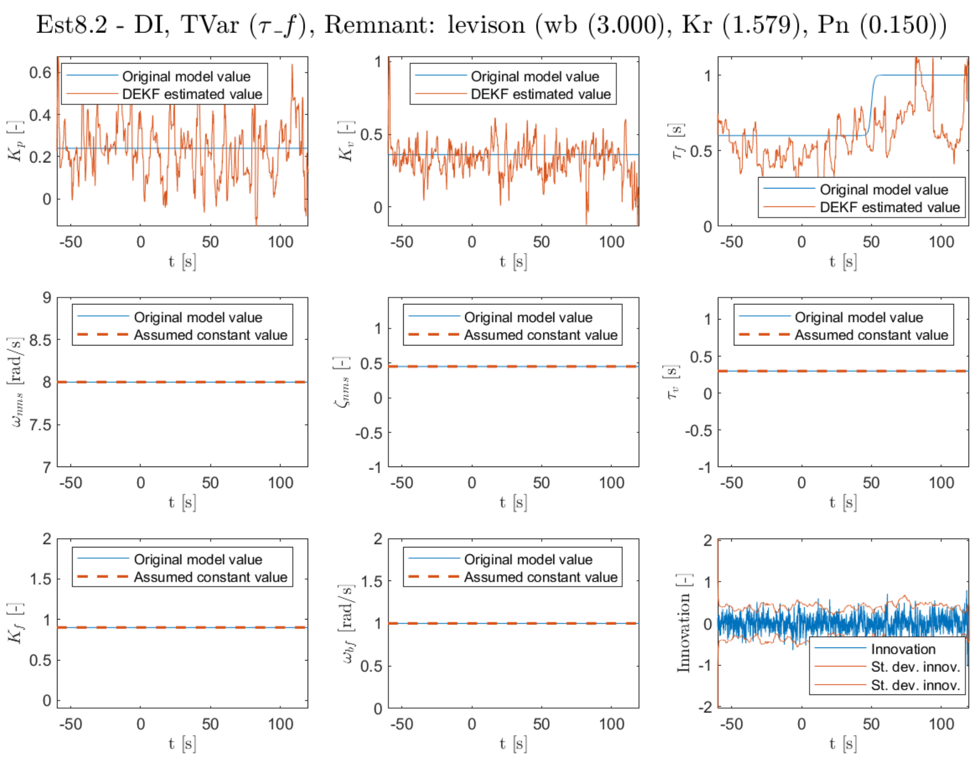


Figure E.19: Benchmark estimation 8.1



s

Figure E.20: Benchmark estimation 8.2

III

Final Thesis Report Appendices

To be graded for AE5310 Final Thesis

F

Errata of the Preliminary Thesis Report

In the continuation of the research after the preliminary stage, new insights in the subject matter led to some fairly substantial changes throughout the research approach. This appendix aims to list the most important changes, that are not yet reflected in the Preliminary phase (Part II of this final report). In general, in case of inconsistencies between the scientific article (Part I) and the body of the report (Part II), the article should always be leading.

The following errata were located:

- Section 4.1: the location of the remnant was changed from the u signal to the feedback of y (compare Figure 4.2 to Fig. 1 in the article). This is in line with the remnant analysis for preview tracking tasks by van der El et al. (2019a).
- Subsection 4.2.1: The remnant gain calculation based on the power ratio is done using the PSD, as explained in van der El et al. (2019a). This is also explained in the scientific article, Section IV-D.
- Subsection 4.2.1: The way of computing time step updates in simulations was changed from using MATLAB's `ode45` function to a discrete LPV model. How this works is described in Section IV-A of the article.
- Subsection 5.1.2: *Suspension* time in the preliminary phase was renamed to *anticipation* time, as this was deemed more intuitive. A consistent explanation of the concept can be found in the article, Section III-B.
- Subsection 5.2.2: The implementation of the total derivative $G_{p,k}^{\mathrm{tot}}$ was changed back to the original equation (Eq. (44) of Popovici et al. (2017)), after it was found that the second Jacobian, $\frac{\partial f}{\partial \boldsymbol{x_s}}$ was to be replaced by its discrete counterpart $\Phi_{s,k-1}$. The correct version of the algorithm can be found in the scientific article, Section III-A and Appendix A.
- As explained in Chapter 7, parameter limits were not yet implemented in the preliminary phase, but only added to the algorithm in later stages. They are therefore not yet mentioned in Chapter 5.
- There are some minor differences in notation in the report compared to the article. For example, subscript r is used for *remnant* in the preliminary report. In the article, the subscript n means *remnant*, whereas the it typically means *near view* (not used in the article) or *neuromuscular* (*nms* in the article) in the report.

G
Complete Results of Estimations in the Scientific Article

This appendix contains all the estimations that formed the basis of the scientific paper of this thesis (part I). The estimation results show all parameters, including the constant ones. For each parameter, the mean result, best result and worst result by NRMSE and the individual estimations are shown.

Moreover, a typical estimation, selected by means of the median VAF is identified from the batch results. For this estimation, the resimulated HO input signal $\hat{u}$ is shown, together with HO input u of the original simulation, the remnant signal n and a 10-second retrospective VAF value that quantifies the quality of the fit locally.

G.1. Feasibility phase

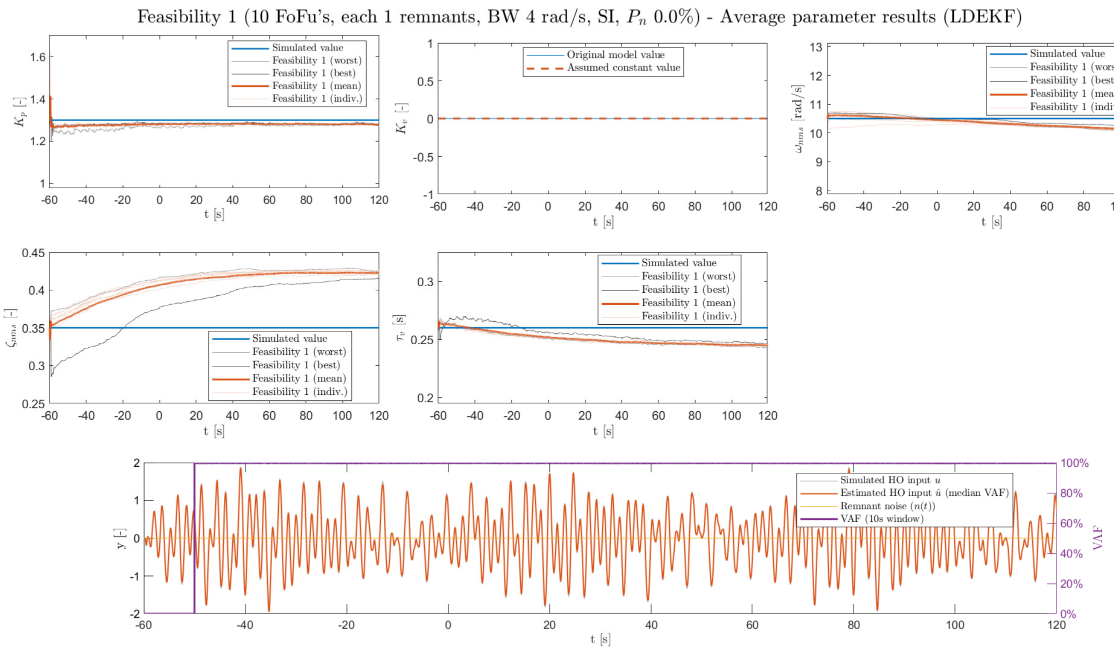


Figure G.1: Feasibility 1: Compensatory, SI, invariant, remnant-free

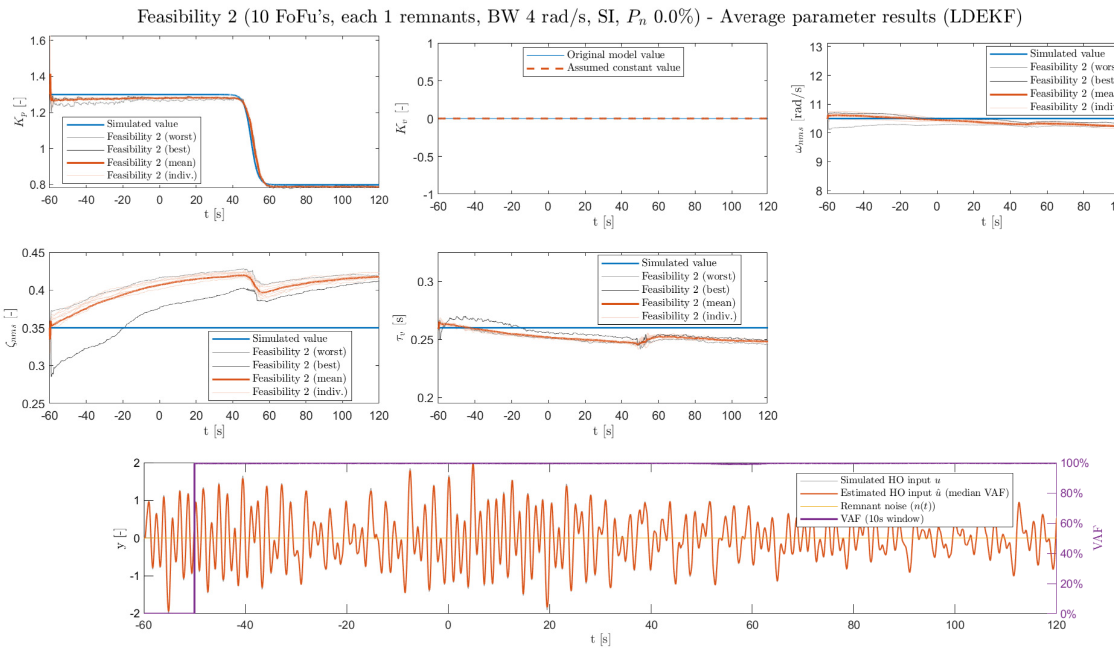


Figure G.2: Feasibility 2: Compensatory, SI, K_p varying, remnant-free

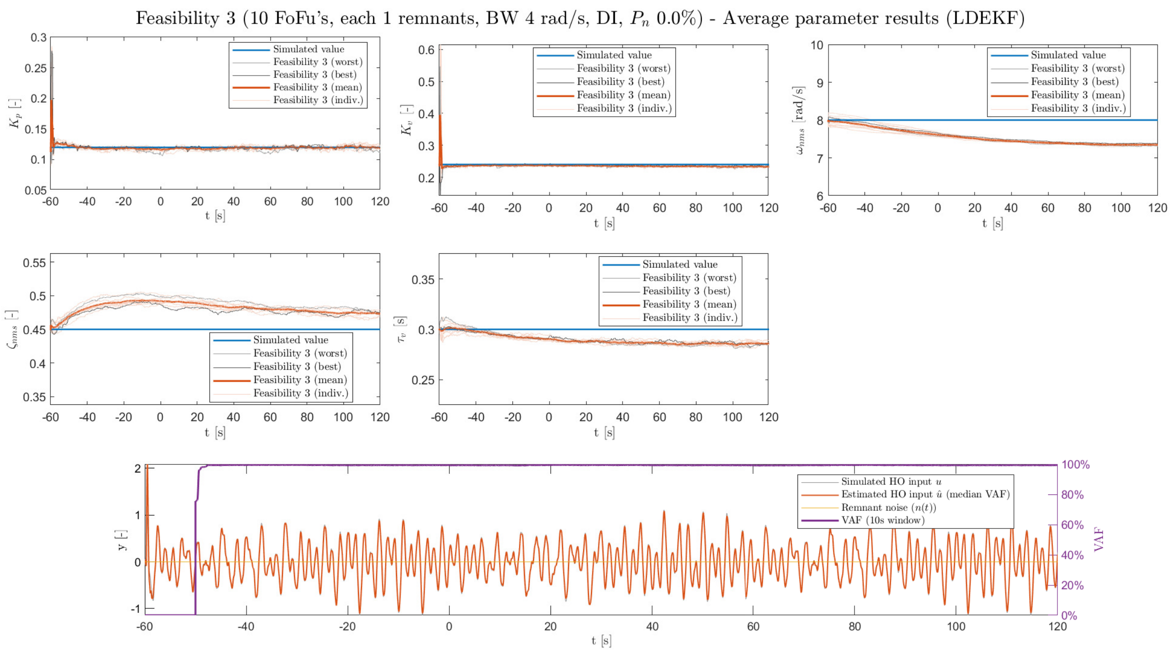


Figure G.3: Feasibility 3: Compensatory, DI, invariant, remnant-free

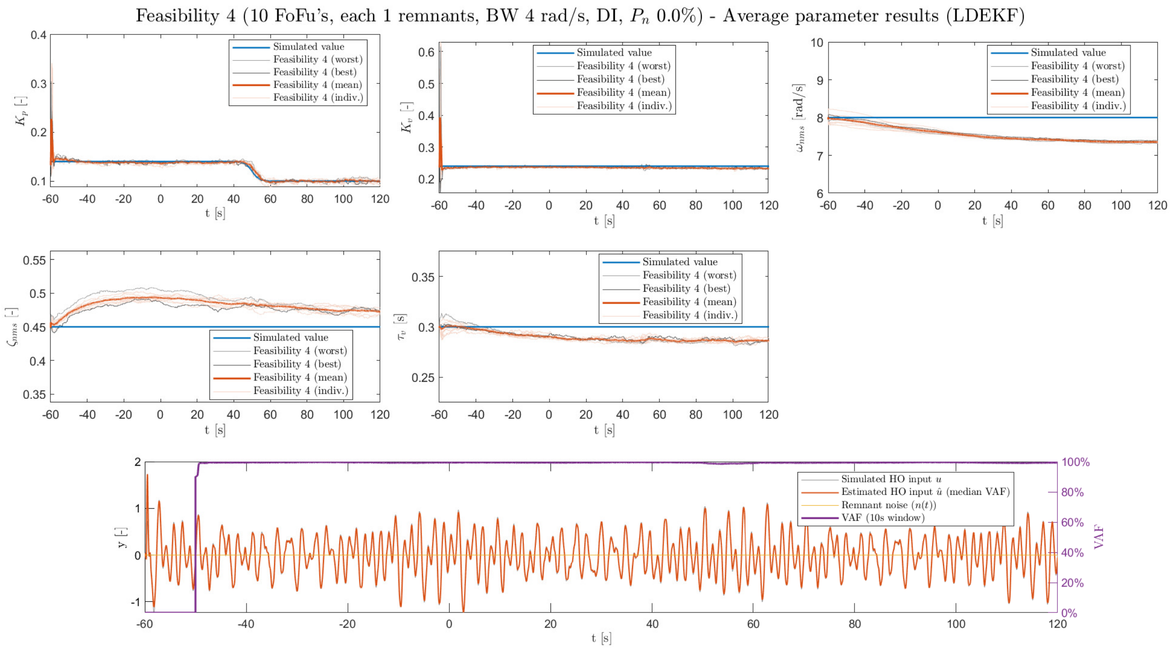


Figure G.4: Feasibility 4: Compensatory, DI, K_p varying, remnant-free

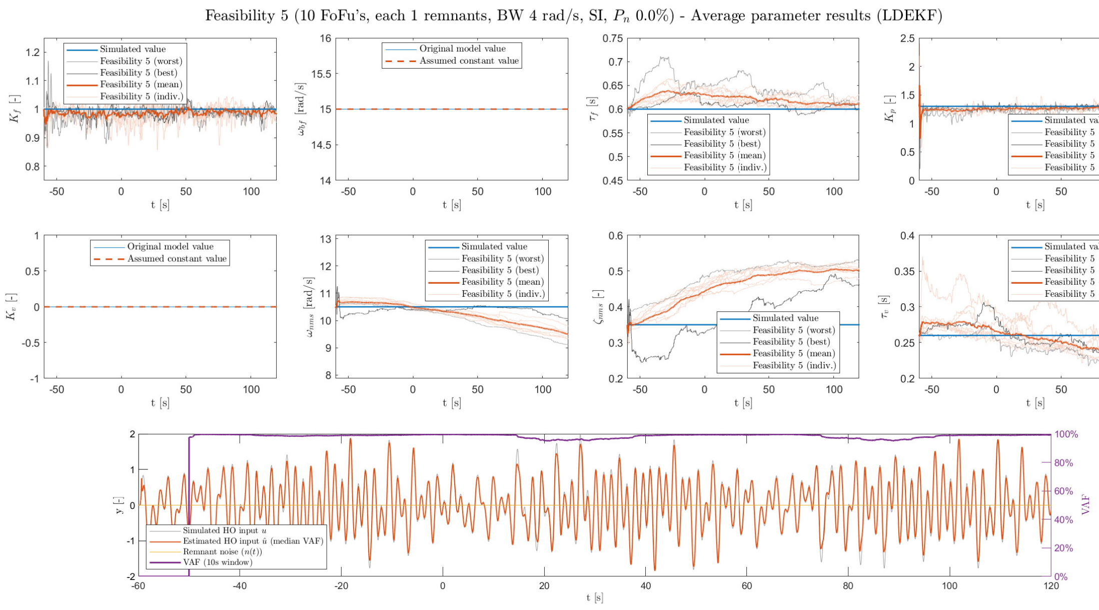


Figure G.5: Feasibility 5: Preview, SI, invariant, remnant-free

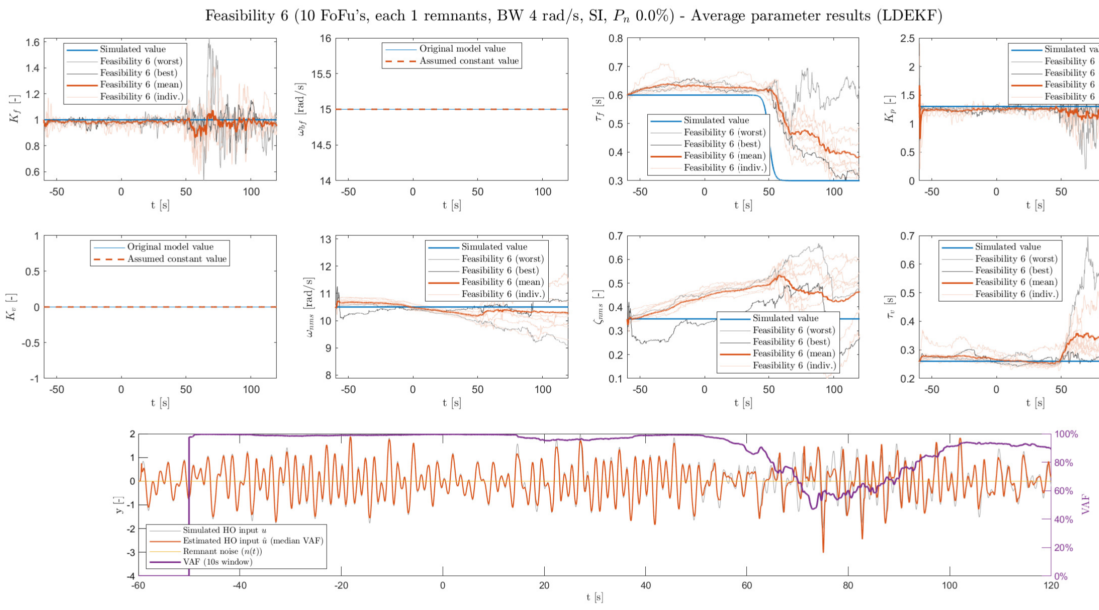


Figure G.6: Feasibility 6: Preview, SI, τ_f varying, remnant-free

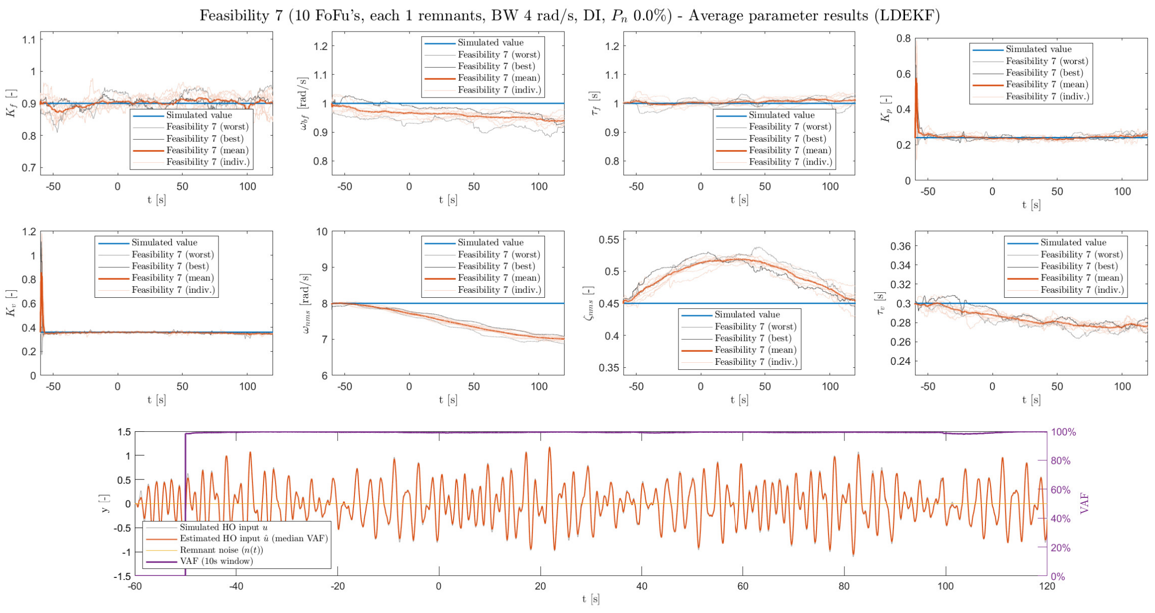


Figure G.7: Feasibility 7: Preview, DI, invariant, remnant-free

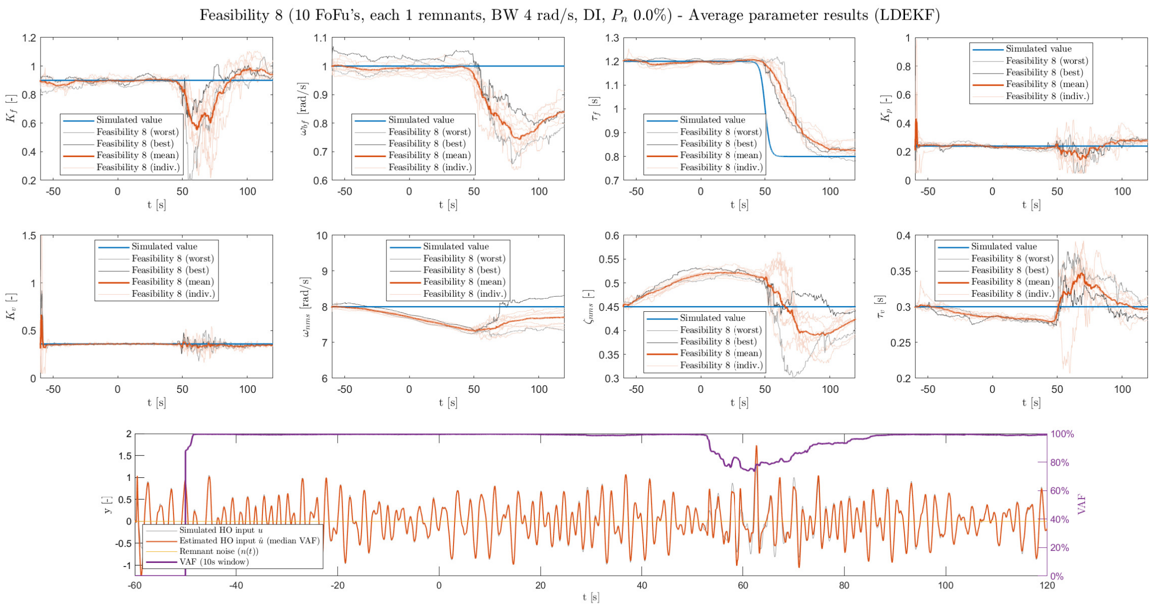


Figure G.8: Feasibility 8: Preview, DI, τ_f varying, remnant-free

G.2. Performance phase

G.2.1. All parameters

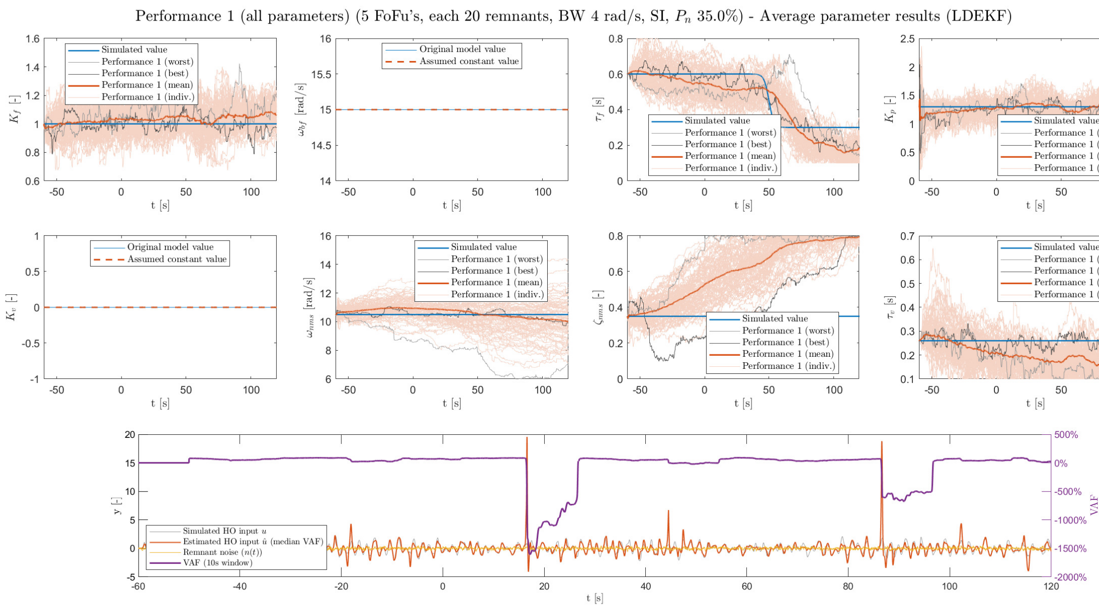


Figure G.9: Performance 1: Preview, SI, τ_f var., full remnant 35%

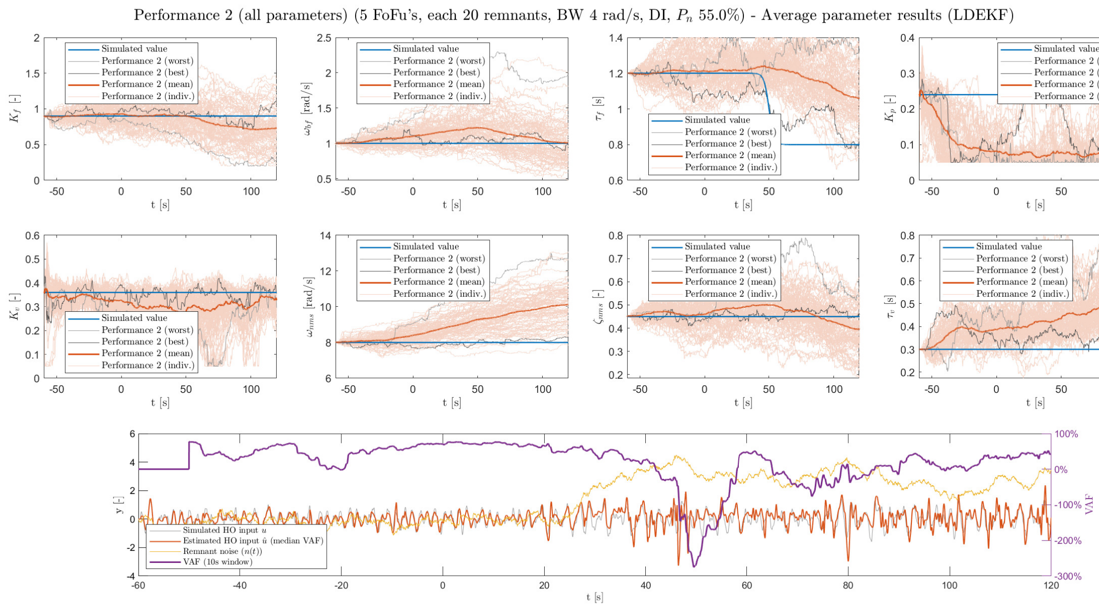


Figure G.10: Performance 2: Preview, DI, τ_f var., full remnant 55%

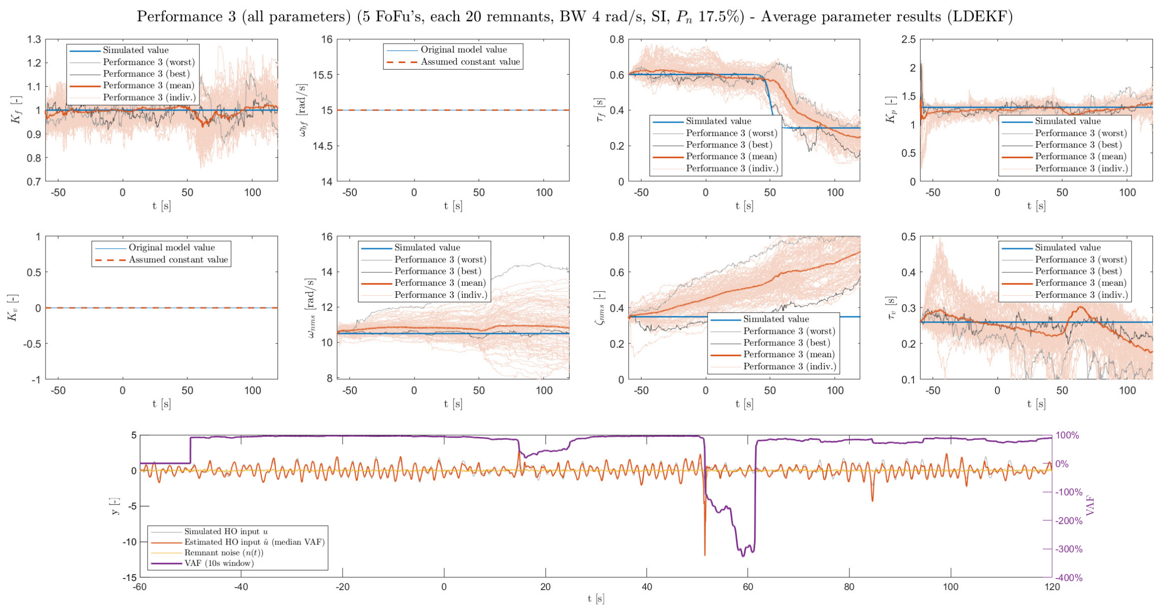


Figure G.11: Performance 3: Preview, SI, τ_f var., half remnant 17.5%

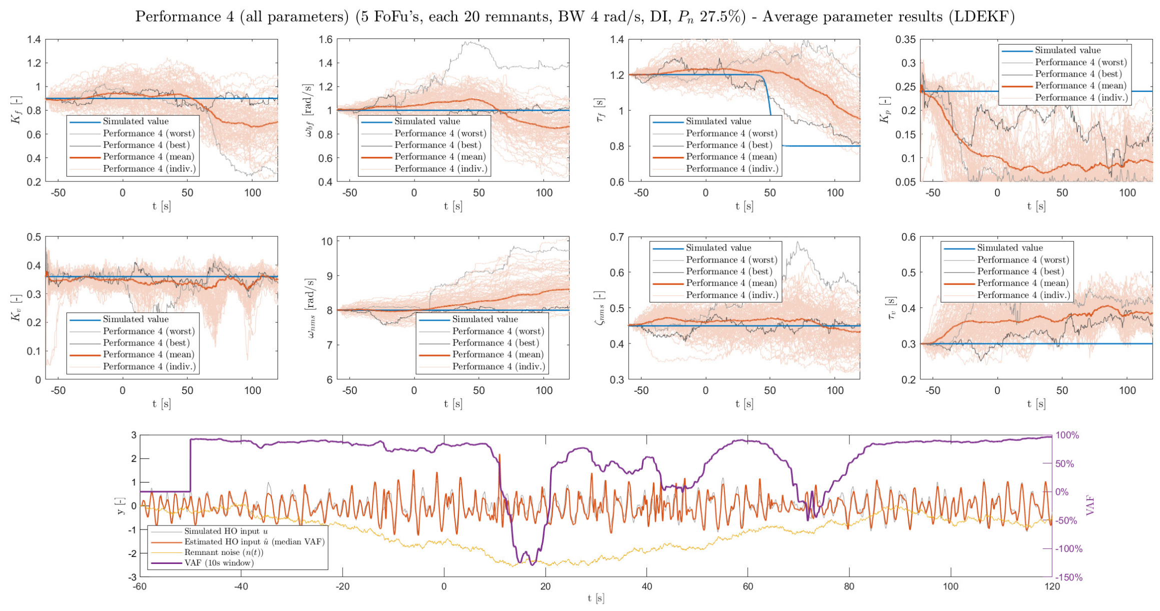


Figure G.12: Performance 4: Preview, DI, τ_f var., half remnant 27.5%

G.2.2. Constant Physical limitations

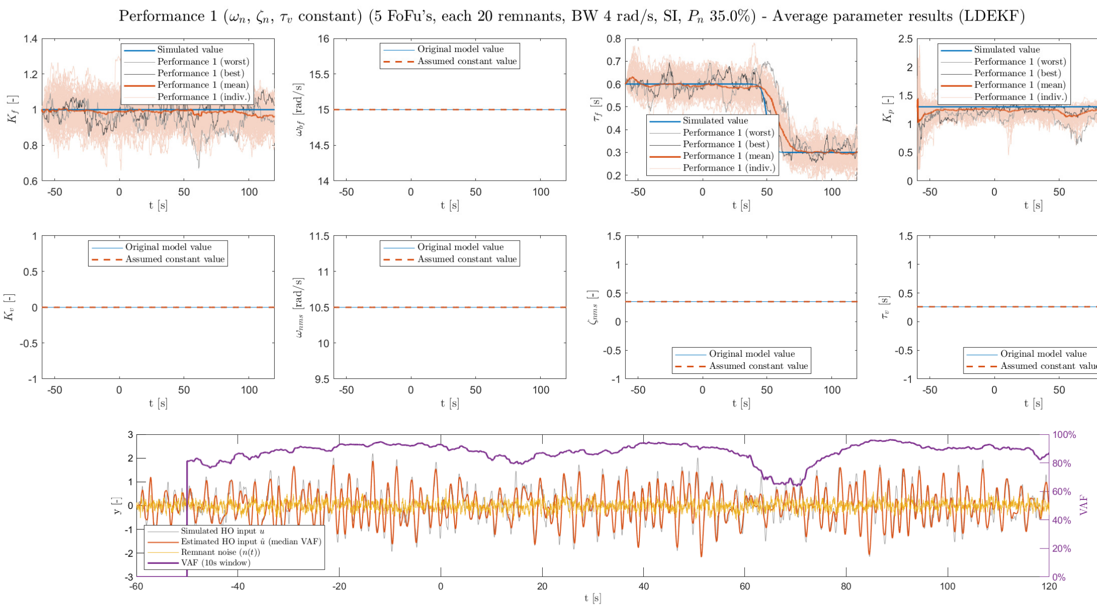


Figure G.13: Performance 1: Preview, SI, τ_f var., full remnant 35%

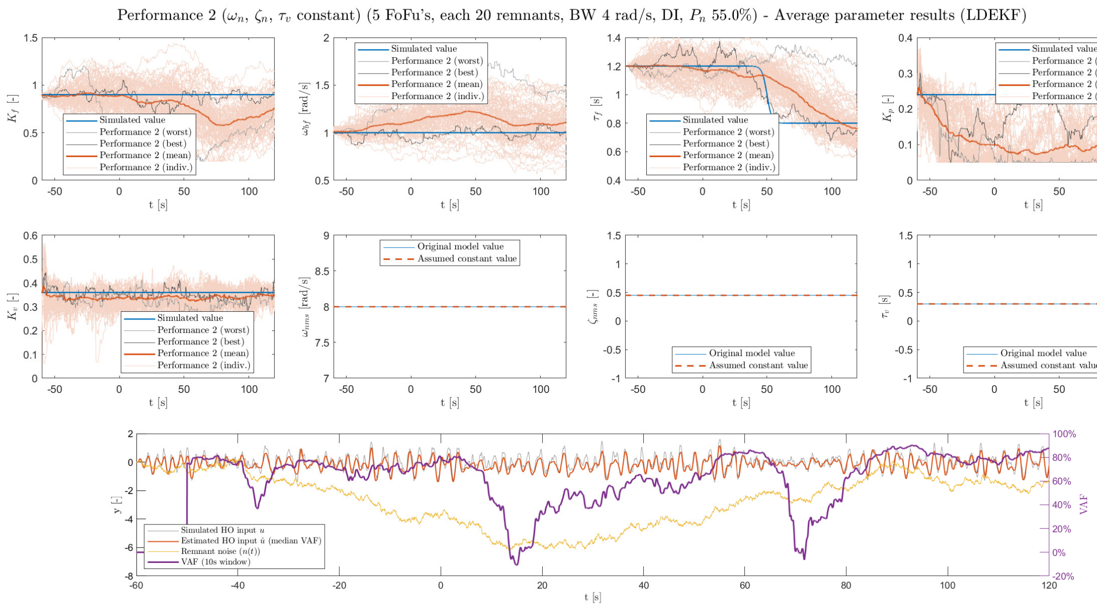


Figure G.14: Performance 2: Preview, DI, τ_f var., full remnant 55%

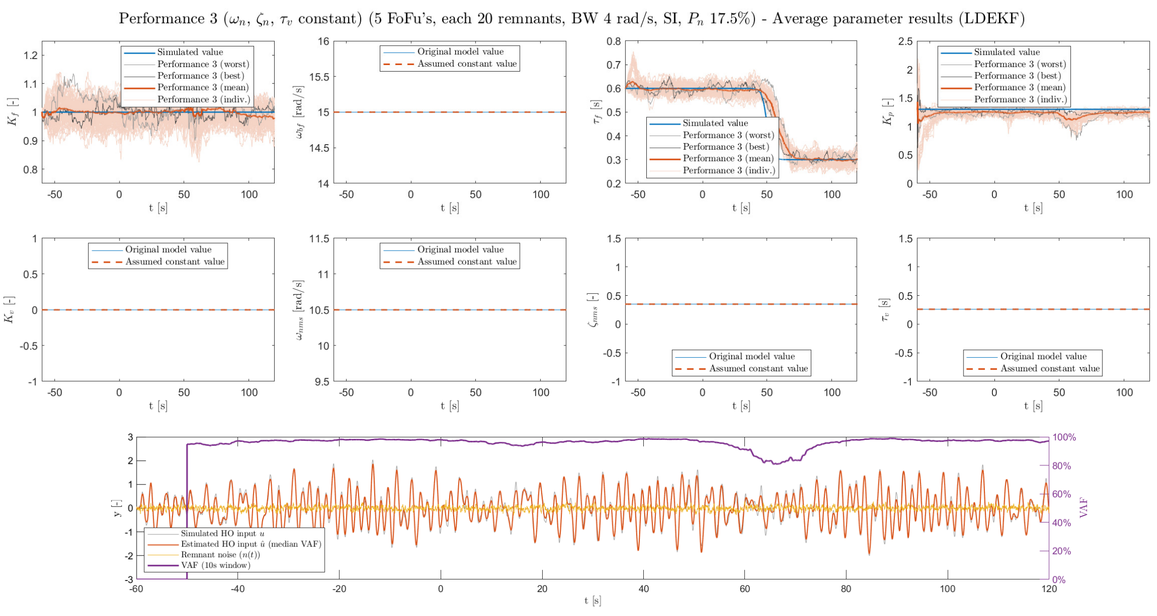


Figure G.15: Performance 3: Preview, SI, τ_f var., half remnant 17.5%

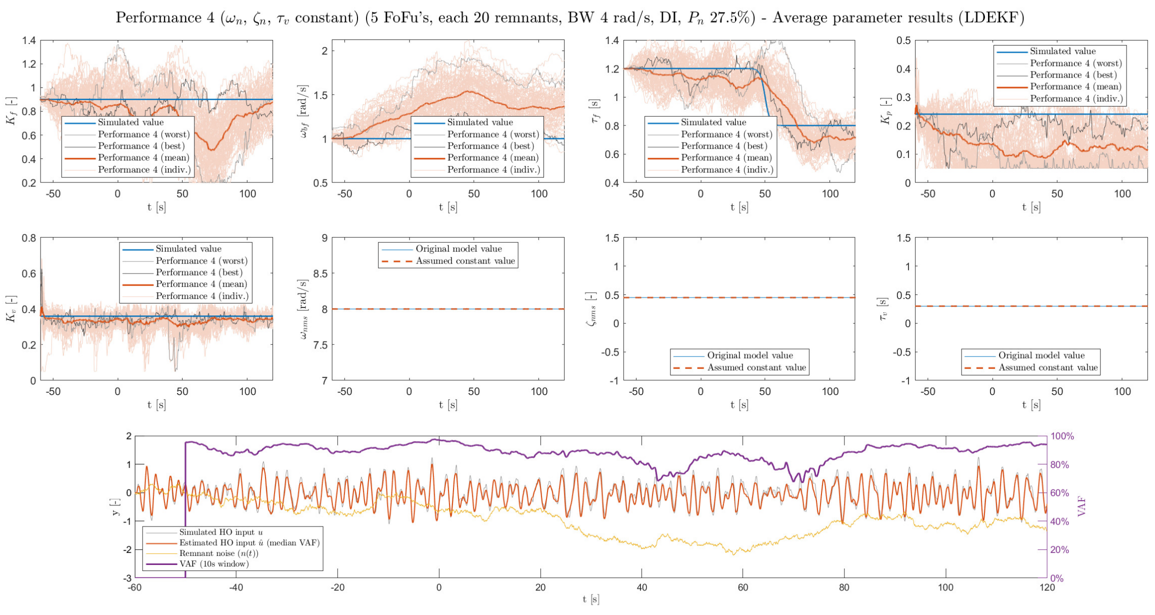


Figure G.16: Performance 4: Preview, DI, τ_f var., half remnant 27.5%

G.2.3. Only preview time

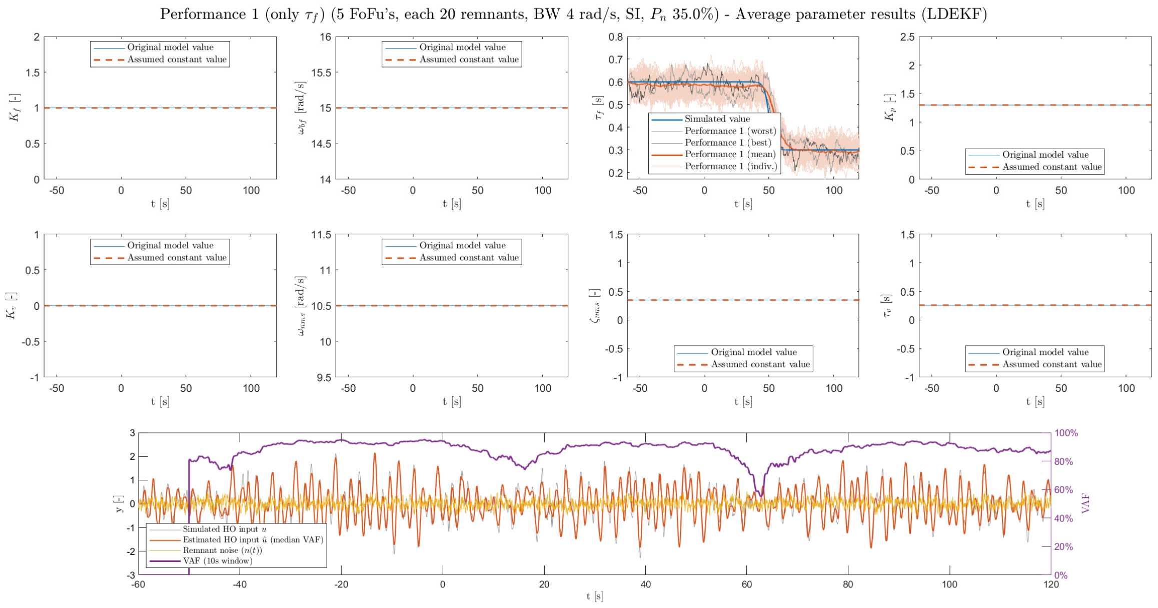


Figure G.17: Performance 1: Preview, SI, τ_f var., full remnant 35%

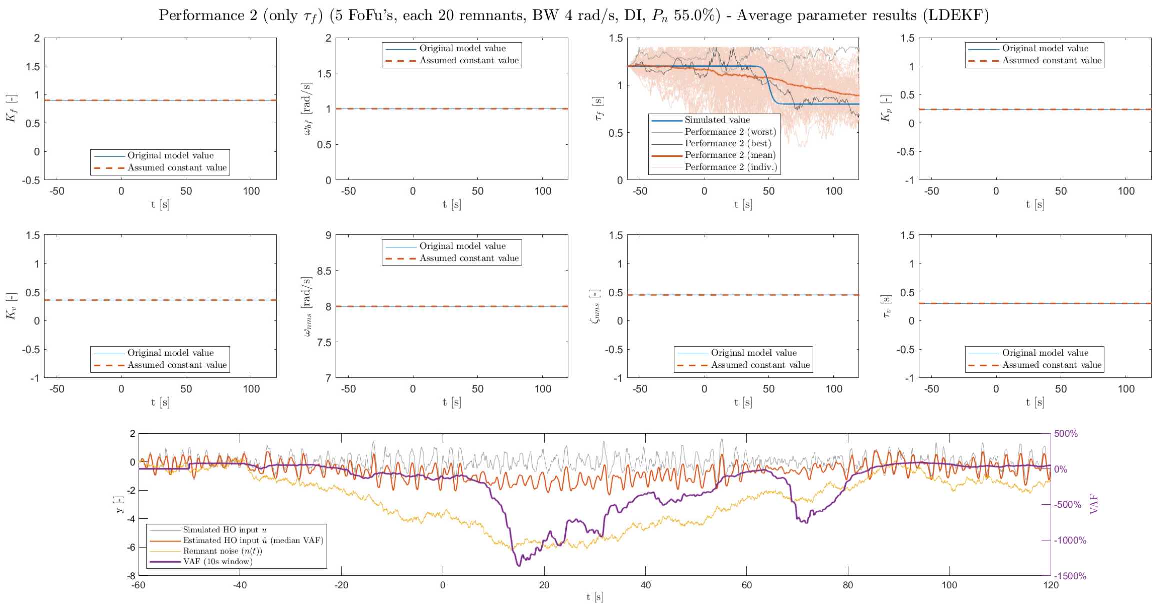


Figure G.18: Performance 2: Preview, DI, τ_f var., full remnant 55%

H Code Documentation

In order to perform this research, MATLAB was used to simulate tracking tasks, estimate behavioural parameters and analyse the results. This appendix will give an overview of this MATLAB code, as an example or a starting point for future research. The text is therefore meant for future users. *N.B. Since the aim is to progress the code in future research, any information found here may not completely reflect the current state of the code. Moreover, this appendix is by no means complete, but merely meant to express the ideas behind the code setup and structure. The code itself should always be annotated with ample comments to clarify its purpose.*

The simulation and estimation software has an Object Oriented code structure. Most of the code is divided in a number of *Classes*, with a hierarchical structure. In general, a class is an entity that has both *Properties* (MATLAB) (called Attributes in Python), as well as *Methods* attached to it. Properties are types of data that hold information on the specific details of the class instance. Methods on the other hand, are *functions* used to do something with these properties, for example plotting results.

This appendix gives an overview of the code structure, as well as the different classes, functions and scripts that were used for this thesis. The governing structure of the classes is depicted in Figure H.1. The classes in purple, constructed to contain sampled and timed data are described in Appendix H.1. The classes in blue, that contain the actual Simulations, Experiments and Estimations, are described in Appendix H.2. The classes in green, that are used to execute and evaluate batches of tracking runs, are further explained in Appendix H.3. Finally, Appendix H.4 will detail some miscellaneous other utilities that are coded.

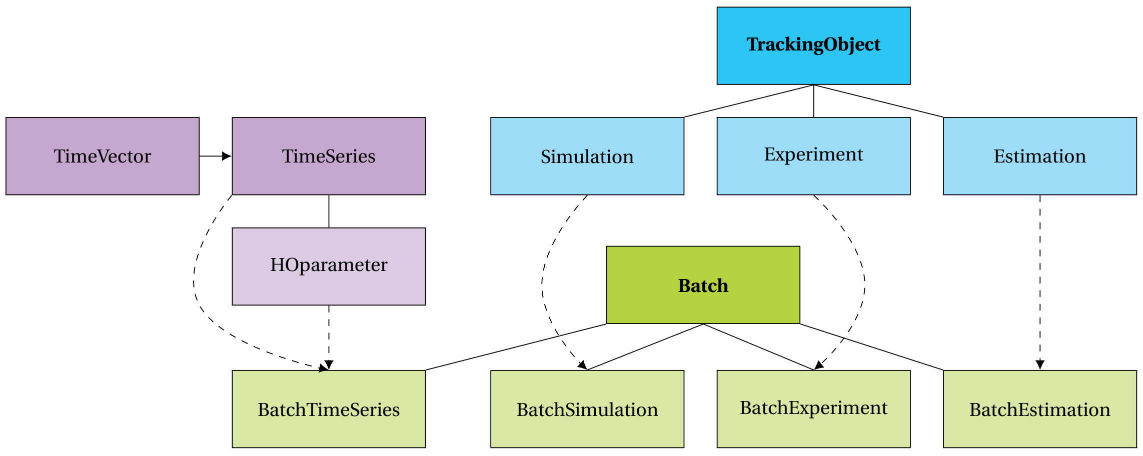


Figure H.1: Class hierarchy of MATLAB code

N.B. In Python, functions have a very flexible way of accepting arguments by means of required (normal) arguments, keyword arguments (with a default value), optional arguments (`*args`) and optional keyword arguments (`**kwargs`). Although MATLAB is more restrictive, the functionality of keyword arguments can be mimicked as it is possible in MATLAB to provide only some of the listed input arguments. By providing a default value for arguments that are further down the line (`if nargin < ...`), supplying the argument is optional, making it similar to a keyword argument in Python. This structure is used often throughout the code.

H.1. Timed data

All signals in the simulations and experiments of tracking tasks are discrete and sampled at a regular time interval. Therefore, two main classes are constructed that bundle the available information. The first is the *TimeVector*, that contains all relevant information about the timing of the data. The second is the *TimeSeries* class, that couples these time labels to the correct sample values. Finally, the *HOparameter* class is a subclass to TimeSeries that is used for the time-varying model parameters of LPV models.

H.1.1. TimeVector

The *TimeVector* class creates the applicable time labels for data samples. These are constructed based on four separate inputs:

- **Measurement time**: indicates the time that is actually used for analysis. In preview tracking tasks, this is typically 120 s (van der El et al., 2016).
- **Time step**: the time difference between consecutive samples.
- **Run-in time**: indicates the time that is used to obtain consistent HO or estimation performance.
- **Anticipation time** (renamed from suspension time): the amount of time margin used by the estimation algorithm to estimate the preview time delay. The concept was explained in Subsection 5.1.2.

Based on these inputs, a unique time vector can be constructed. Apart from the aforementioned four properties, the TimeVector class also lists some useful things such as the total summed time, the total number of samples and fundamental frequency.

H.1.2. TimeSeries

The *TimeSeries* class provides the foundation of all timed signals in the code. This class couples the appropriate values to the time labels given by the associated TimeVector. The easiest way to initialise an instance of a TimeSeries is therefore with a TimeVector and a vector of values of the same total length.

Plotting Aside from time and values, the TimeSeries class also holds plot labels and properties. This makes plotting different signals together very rapid. A quick plot of multiple TimeSeries can be made by using the function `plotTS` on a primary TimeSeries, and providing the other TimeSeries to plot as optional arguments. In order to plot TimeSeries data together in subplots, the function `plotTSinAxis` can be used by providing the required target set of MATLAB *Axes*.

Analysis There are a number of built-in analysis tools in the TimeSeries class. These include for example the minimum, maximum, mean, variance, VAF, RMSE and NRMSE. Moreover, the auto-PSD is automatically calculated from the measurement-time data and can be plotted easily with the function `plotPSD`. Again, multiple TimeSeries can easily be plotted together by providing them as arguments.

Mathematical operations TimeSeries can be added, subtracted, multiplied or divided. These operations are performed on the values of the class. A new TimeSeries instance is subsequently constructed, using the TimeVector of the original TimeSeries and the newly calculated values.

HOparameter The *HOparameter* class is a subclass of TimeSeries that is used to specify either the parameter inputs to the LPV simulation, or the time-varying parameter traces that form the output of an Estimation. The only additional information that the HOparameter contains, on top of normal TimeSeries, are the inputs that form their simulated shape (especially for sines and sigmoidal parameters), as well as an easy way to construct these standardised time-varying parameters.

H.2. Tracking Objects

Tracking tasks form the backbone of this research. They provide a controlled way of testing and analysing human control behaviour. The input and output signals that describe the result of a tracking task can be obtained by both computer simulation as well as an actual experiment. In essence, these instances should be interchangeable. Moreover, this thesis research tries to extract time-varying HO parameters from the same set of input and output signals by means of a DEKF.

Therefore, these three classes, *Simulation*, *Experiment* and *Estimation*, have a common superclass called *TrackingObject*. This superclass contains the properties and methods that are shared. These include:

- Properties with **tracking task information**: forcing functions, CE dynamics, time properties, etc.
- Properties and methods that list names of the **applicable HO parameters and canonical states** in the state and parameter vectors $\boldsymbol{x_s}$ (Xs) and $\boldsymbol{\theta}$ (Th)
- Methods for **saving and loading** the different instances (particularly Simulations and Estimations). This will be explained in H.2.4.

H.2.1. Simulation

The *Simulation* class object handles the logic for tracking task simulations. How the simulations are set up was already discussed in Section IV of the scientific paper and in Chapter 4, based on the flow diagram that is copied in Figure H.2. This section couples the flow steps to their associated MATLAB functions, together with pointers to specific locations where more information can be found if required. It is easiest to understand this section while following along in the code.

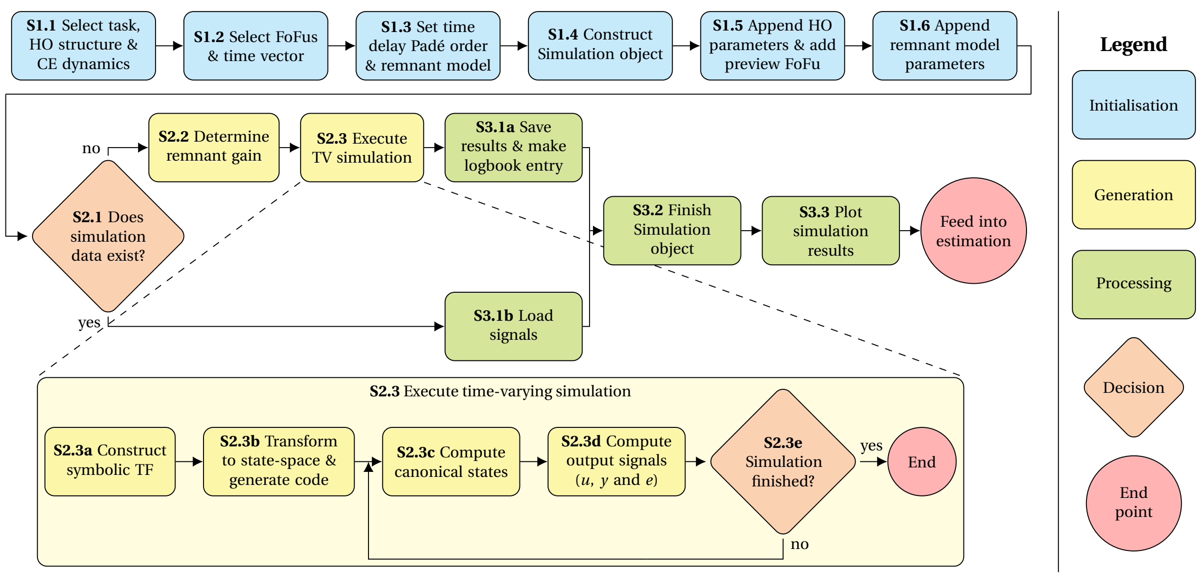

S1.1 Select task, HO structure & CE dynamics
S1.2 Select FoFus & time vector
S1.3 Set time delay Padé order & remnant model
S1.4 Construct Simulation object
S1.5 Append HO parameters & add preview FoFu
S1.6 Append remnant model parameters
no
S2.1 Does simulation data exist?
yes
S2.2 Determine remnant gain
S2.3 Execute TV simulation
S3.1a Save results & make logbook entry
S3.1b Load signals
S3.2 Finish Simulation object
S3.3 Plot simulation results
Feed into estimation
S2.3 Execute time-varying simulation
S2.3a Construct symbolic TF
S2.3b Transform to state-space & generate code
S2.3c Compute canonical states
S2.3d Compute output signals (u, y and e)
S2.3e Simulation finished?
yes
no
End
Legend
Initialisation
Generation
Processing
Decision
End point

Figure H.2: Flowchart of Human Operator simulations

Initialisaton The code is structured such that the initialising the different settings is as apparent as possible. A '*main*'-script with an example of how this is done, is `PrevSim.m`. The steps S1.1-S1.3 are listed stright away, followed by the construction of the desired Simulation object using these settings (S1.4).

Subsequently, the HO parameters are defined by appending `HOparameter` class objects with the desired shape to a *struct*. Note that the *fieldnames* of this *struct* must be identical to the parameters listed in the parameter vector property (`Sim.Th_names`), in order for them to be recognised. The entire *struct* is appended to the Simulation with the method `Sim.addHOpars` (step S1.5).

Similarly, the remnant model parameters are appended supplied in the form of a *struct* using the method `Sim.addRemnantPars`. Note that the desired remnant power ratio P_n should be provided (see Section II-B of the scientific article, or Subsection 2.1.4). The `calculateRemnantGain` method that is automatically called when appending the parameters will then automatically find the correct gain matching the remnant power ratio (S2.2). This ratio is calculated using the PSD function as described by van der El et al. (2019a).

Data Generation The Simulation is now ready to generate data. This can simply be done by calling the `Sim.executeTVsim` method, which contains the check of step S2.1 and subsequently executes all the substeps S2.3a-e. Note that for verification purposes, a linear simulation can also be executed using `lsim` and a simulation with SI dynamics can also be executed using `Simulink` (see Section 4.4 and Appendix B for more information). To execute these simulations as well, use the `Sim.executeAll` method.

The substeps S2.3a-e are spread out over different methods and functions. After a small sequence of nested function calls (`genCanonicalFiles` and `genTransferFunction`), step S2.3a is performed by the separate function `makeModelTF`. This step is placed in a separate file, because it is shared by the Simulation and Estimation classes. Since Simulations and Estimations run exactly the same code, errors based on version differences are avoided. Moreover, this way, the model can be easily swapped out for a different one, depending on the needs of the user. For example, the model behind a Simulation/Estimation can be easily replaced by a more advanced multi-loop car driving model without much effort. The only things that need to be adapted are this `makeModelTF` file and the HO parameters that are appended to a TrackingObject with of the `listHOparameters` method.

The transformation from transfer function to state-space system (S2.3b) is subsequently executed in the continuation of the `genCanonicalFiles` function. This is done according to the logic explained in Subsection 4.2.2 for full system simulations and according to Subsection 5.2.1 for resimulations of the isolated HO model. The state space system is subsequently transformed using a series of functions into a set of automatically generated function files that can be evaluated for each time step of the simulation. All of these functions form a seperate module, the *SymbolicCanonical* subfolder. This module is an adaptation of MLE identification code by Daan Pool. To retain some of its original structure, is kept in a separate folder. And, again, because it is shared between the Simulation and Estimation classes. The original module from Daan is changed in some aspects to improve its useability. The `eval` statements are replaced as much as possible, to make debugging easier. Moreover, some changes were implemented regarding the automatically generated function files, drastically improving their speed of execution in the simulation process.

After generating the function files, the code returns to the `executeTVsim` method of the Simulation class, where the simulation steps are executed. As outlined most clearly in Section IV-A of the scientific paper, this is done by means of a discrete LPV model:

$$\dot{\boldsymbol{x}}_{\text{sim}} = A_{\text{sim}}(\boldsymbol{\theta}(t))\boldsymbol{x}_{\text{sim}} + B_{\text{sim}}\boldsymbol{u}_{\text{sim}} \tag{H.1}$$

$$\boldsymbol{x}_{\text{sim},\boldsymbol{k}} = \Phi_{\text{sim},k}\boldsymbol{x}_{\text{sim},\boldsymbol{k}} + \Psi_{\text{sim}}\boldsymbol{u}_{\text{sim},\boldsymbol{k}} \tag{H.2}$$

$$\boldsymbol{y}_{\text{sim},\boldsymbol{k}} = C_{\text{sim},k}\boldsymbol{x}_{\text{sim},\boldsymbol{k}} + D_{\text{sim}}\boldsymbol{u}_{\text{sim},\boldsymbol{k}} \tag{H.3}$$

Processing After the simulation is completed, the data can be saved for future reference (S3.1a). The simulated signals are saved in a `.mat`-file and the elemental settings of the simulation that were specified in initialisation steps S1.1-S1.6 are saved to the `simLogbook` in the Logbook folder. Alternatively, if a simulation is already executed, the data can be loaded (S3.1b). In that case, the data is simply appended from the logfile. Especially for large batches, this can save a considerable amount of time, as each time-varying simulation takes several seconds to complete but a fraction of that time to load. A more detailed description of the saving and loading Simulation objects can be found in Appendix H.2.4.

With the data in place, the Simulation is finished. Its results can be plotted using the `Sim.plotResults` method (S3.3). Moreover, the Simulation can serve as an input to an Estimation run.

H.2.2. Experiment

With the *Experiment* class, data from a previously executed experiment can be loaded in a similar format to a Simulation. As such, the estimation algorithms that were tested using simulated data can be applied directly.

In this research, the preview time experiment by van der El et al. (2018) was used to evaluate the performance of the DEKF on experimental tracking data. In this experiment, the effects of limited preview on HO tracking behaviour were measured for single- and double-integrator dynamics under steady (time-invariant) task conditions. In total, the tracking experiment was performed for eight different conditions for the amount of available preview, ranging from 0 to 2 s. With the function `createFromPreviewTimeExp` the experiment results are loaded with the input of the desired CE type (DI or SI), participant (1-8), condition number (1-8) and run number (1-5).

H.2.3. Estimation

The *Estimation* class object handles the logic for HO parameter estimation from tracking task control behaviour. Similar to the Simulation class, the general estimation procedure was already discussed in Section III of the scientific paper and in Chapter 5, based on the flow diagram that is copied in Figure H.3. This section couples the flow steps to their associated MATLAB functions, together with pointers to specific locations where more information can be found if required. It is easiest to understand this section while following along in the code.

Initialisaton The code is structured such that the initialising the different settings is as easy as possible. A '*main*'-script with an example of how this is done, is `PrevEst.m`. The steps E1.1-E1.3 are listed stright away, followed by the construction of the desired Estimation object using these settings (S1.4).

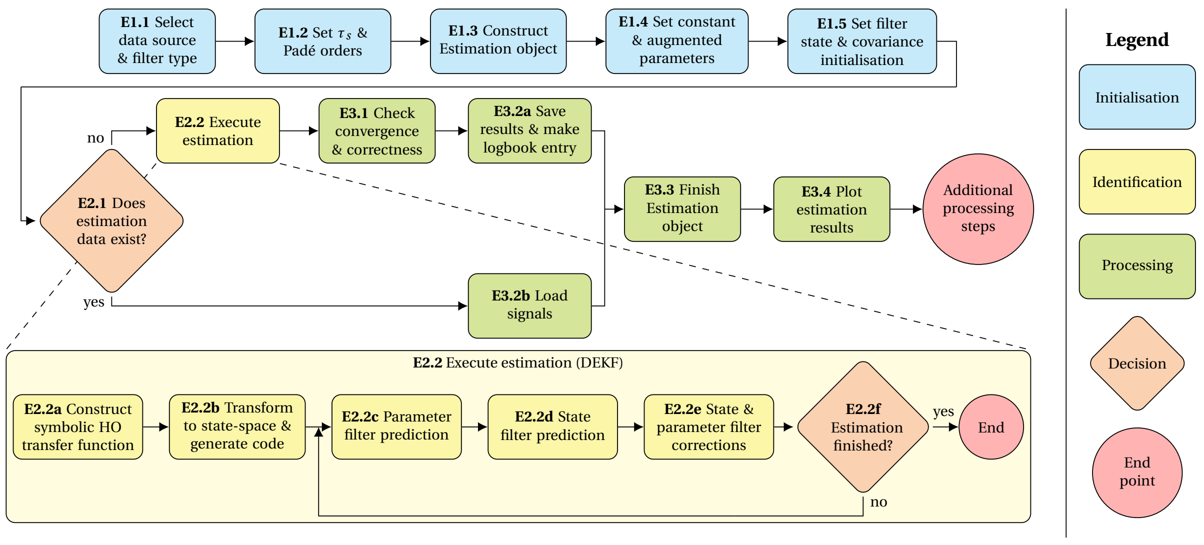

E1.1 Select data source & filter type
E1.2 Set τ_s & Padé orders
E1.3 Construct Estimation object
E1.4 Set constant & augmented parameters
E1.5 Set filter state & covariance initialisation
E2.2 Execute estimation
E3.1 Check convergence & correctness
E3.2a Save results & make logbook entry
no
E2.1 Does estimation data exist?
yes
E3.3 Finish Estimation object
E3.4 Plot estimation results
Additional processing steps
E3.2b Load signals
E2.2 Execute estimation (DEKF)
E2.2a Construct symbolic HO transfer function
E2.2b Transform to state-space & generate code
E2.2c Parameter filter prediction
E2.2d State filter prediction
E2.2e State & parameter filter corrections
E2.2f Estimation finished?
yes
End
no
Legend
Initialisation
Identification
Processing
Decision
End point

Figure H.3: Flowchart of Human Operator parameter estimation

Afterwards, the HO parameters that need to be augmented to the state vector are listed by the user. The method `Est.augmentParameters` subsequently moves them to the state vector. Note that the names should match parameters listed in the parameter vector (`Th_names`), otherwise the parameter is not augmented (a warning is thrown for each parameter that is not recognised). If an EKF or LEKF (limited EKF with parameter limits) is selected in E1.1, all parameters are automatically augmented.

The constant parameters are provided as a *struct* with parameter names and their constant value, similar to HO parameters and remnant parameters for Simulations. Again, the *fieldnames* of the *struct* must match parameter names in the parameter vector (`Th_names`). The constant parameter *struct* is appended with the *Est.addConstantPars* method.

In a similar fashion the parameter limits are applied (which were not yet in place in the preliminary phase, but would have a separate step E1.5, pushing the current E1.5 to E1.6). The fields of the *struct* are now given by 1×2-arrays with the lower and upper limits. If only one limit is to be provided, use `NaN` for the other.

Finally, the filter initialisation values are selected (E1.5). The specific variables to tune were described in detail in Section III-D. The estimation is correctly initialised by means of the `Est.initFilter` method. Covariance percentages for P and Q can be provided both for specific parameters, as well as for parameter groups. So `Qperc.K = 0.05` will apply an initialisation percentage of 5% to all HO gains (i.e. K_f, K_n, K_p, K_v).

Parameter identification With the filter initialised, it can now be executed. Again, this can be done with a single command, `Est.execute`, that runs through all the steps E2.1 and E2.2a-E2.2f.

The first steps of the process are very similar to a Simulation. Through the same set of nested function calls, the separate function `makeModelTF` is called to generate a symbolic transfer function (E2.2a). The *SymbolicCanonical* module, by means of the `genCanonicalFiles` function, is then used to generate a symbolic state-space description of the model, but this time the isolated HO model is used. Subsequently, auto-generated function files are written for the state-space functions f and g, as well as the jacobians (E2.2b). See the explanation at the Simulation class for more info. *N.B. Strictly speaking, the* `genCanonicalFiles` *function was already called as part of the initalisation process, because the exact amount of canonical states is required for the initalisation of the state vector and state filter covariance matrices. These canonical states are dependent on the exact model implementation. However, process-wise, the steps E2.2a and E2.2b fit better with the identification phase of the estimation*

The Estimation object then calls the `dekf_new_limits` function, that executes the estimation (steps E2.2c-E2.2f). Since it is quite a long algorithm, it is placed in a separate function to keep overview. This function can be used for all types of extended Kalman filters, both dual and joint, and both with or without applied limits. The function strictly follows the algorithm described in detail in Section III-A of the paper. Note that the Limitation steps are not in Figure H.3, as they were not yet implemented in the Preliminary phase. They can be through of as having a separate field between steps E2.2e and E2.2f.

Processing The Estimation class has several Dependent properties that automatically assess the convergence and performance of the different parameter estimates (E3.1). As described in Section 5.3, the convergence is only determined based on the occurrance of `NaN` or `Inf` values, whereas the correctness is based on an NRMSE-criterion. The Estimation data can be saved and loaded (E3.2), similar to a Simulation. More details are given in Appendix H.2.4.

The Estimation is now complete, and can be analysed in various ways:

- A single estimated parameter trace, or all estimated parameters simultaneously, can be plotted using the `plotParameter` and `plotAllParameters` methods (E3.4). The second can be tuned to include the constant parameters, as well as the innovation of the Kalman filter throughout the estimation.
- The output of the HO model, the stick input signal u, can be resimulated using the estimated parameter values ($\hat{u}$). This signal can be compared to the u-signal of the original simulation by plotting both together (the `plotReSim` method), by means of the overall VAF (`VAFestimated`) and a 10-second retrospective VAF (`movingVAFestimated`, presented as a TimeSeries object). Similarly, this can be done for the idealised u-signal, without added remnant noise, and the u-signal of the actual simulation (`movingVAFideal`).
- The `plotVAF` method plots the resimulated ($\hat{u}$) and original (u) stick input signals together with the 10-second retrospective VAF and the remnant noise signal n.
- More analysis tools are available for Estimation batches. See Appendix H.3 for more information.

H.2.4. Logbook

The execution of simulations and estimations is a time-consuming process. Especially if large batches are executed, the runtimes add up. To save time, it is possible to save and load previously executed Simulation and Estimation runs with the *LogBook* module in the equally named folder.

Storing the entire class object would add up to large amounts of storage space. Moreover, version issues arise in case the code is developed, potentially rendering a previously saved TrackingObject useless or corrupted. Therefore, the minimal result signals of a TrackingObject are saved in a separate logfile, and the settings of the Object are stored in a *logbook*. The logbook is a *struct* object with the settings of all logged Simulations or Estimations. Each of these classes has a separate logbook, `simLogbook` and `estLogbook`, respectively. They are auto-generated in the *LogBook*-folder and updated once an object is logged.

The logbook *structs* can be printed to a human readable CSV-file, such that the logged Simulations and Estimations, together with their properties and most important results, can be inspected. This is done with the `printLogbook` function in the module folder. The folder also contains some functions that prepare the logbook for printing, or can assist in editing an existing logbook.

H.3. Batches

In order to assess the performance of the DEKF for a larger range of remnant and forcing function realisations, so-called *batches* are used. The Batch superclass provides the framework to bundle different TrackingObjects, as well as TimeSeries. A Batch of Simulations or Estimations has identical settings for each element (referred to as '*children*'). The only thing differentiating the children is the realisation of FoFu pair or remnant noise.

The *Data* property of a Batch that contains the children therefore structures them in an $m \times n$ array, where m stands for the number of noise realisations and n stands for the number of forcing function realisations. In case of actual experiment data, dimension m is used for the different participants, and n are their individual runs (which can contain different forcing function realisations, depending on the experiment design).

For examples on how to employ Batches, the files `PrevSimBatch` (for simulations), `PrevEstBatch` (for estimations on simulated batches) and `PrevEstBatchExp` (for estimations on experiment conditions) can be reviewed.

The Batch superclass contains logic that can execute an operation on all Data elements, the `foreach` function. The input to this function is a MATLAB function handle. An example is the `getDataProperty` function, which simply returns a grid with the requested Property for each Data element.

Subclasses The Batch class is implemented for Simulations, Experiments and Estimations, as well as for TimeSeries to ease the grouped analysis. Most of the Batch subclasses are not particularly complex. They most importantly serve to group their children together, and make it easier to apply batch-wide analysis. Examples of this can be found in the *BatchTimeSeries* class, that has functions to return interesting quantities of the batch per time step, such as the mean, median, upper bound or standard deviation.

The *BatchEstimation* class has some tools that can be used to analyse and compare different batches:

- several plotting functions can visualise the results of the entire batch: `plotAllAvgPars`, `plotSelectAvgPars`, `plotAvgParameter`. The first two take several optional arguments and additional batches can even be supplied for comparison in the same plot. Examples of these plots can be found in the scientific paper of Part I and in Appendix G.
- The progression of parameter biases can be plotted with `plotAllParBias`
- the `writeResultTable` method can be used to generate a LaTeX-table that summarizes the batch results. Again, multiple batches can be supplied as variable arguments for side-by-side comparison.

H.4. Other utilities

H.4.1. Guided User Interface

The '*main*'-scripts that contain the code instructions with settings for the specific simulation(s) or estimation(s) to perform (i.e. `PrevSim` and `PrevEst`), are designed to provide as much overview as possible, while at the same time being complete. They almost work as a guided user interface, where the programmer only has to change some specific values to obtain and analyse the required results. This led to the idea that an actual GUI might be useful, for starting users to get accustomed to the software, and for rapid testing of new ideas. The GUI, renamed to Cassis (short for Casper's Simulation Suite) by Tjeerd during a Preview-group meeting, is a App wrapped around the code. It can be found in the *GUI* folder in the repository. It provides an intuitive way to play around with the basic functionalities presented earlier this appendix. Not all functionalities of the code are implemented in the GUI (yet). It merely serves as a starting point, more advanced and extensive analyses should be performed by coding.

H.4.2. LPVmodel class

The *LPVmodel* class is a utility to rapidly simulate and analyse symbolic transfer function expressions of LPV models. The class can be initialised symply from a string input, that will automatically be transformed into a symbolic expression. Alternatively, an existing symbolic equation can be used to initialise a class instance. This class was a recent development, and its functionalities are therefore still limited to quickly simulating one-dimensional transfer functions and plotting results. Future developments can for example include MIMO systems or a 3D-bode plot, which visualises the progression of the system's magnitude and phase response as the parameters change value.

H.4.3. DEKF initialisation optimization

To find suitable initialisation settings for the DEKF applied to preview tracking, an optimisation step was used, described in Section IV-F-2 of the article in Part I. This optimization finds optimal parameter process noise covariance initialisation settings, based on the NRMSE of the estimated parameter. The `optimizeDEKFinit` script is used for this optimisation step. These optimisation runs can be logged in a separate CSV-logbook in the LogBook module, by means of the `addOptim2Logbook` function.

H.4.4. Forcing function generator

This research required multiple equivalent forcing function realisations. In order to generate these realisations, the `MakeFoFu` script is used (located in the *Utilities* folder). After choosing the required amount (N) of forcing functions and the setting desired amplitude and frequency characteristics, a large number of equivalent realisations are created using a random set of phase offsets. From these realisations, N are selected based on the desired *crest factor*, which is the ratio of the fofu's maximum and root mean square.

I

Future Algorithm & Analysis Improvements

One of the main conclusions of this research, as presented in the scientific paper of Part I, was that the current implementation of the DEKF can be and needs to be improved for the intended applications. This appendix contains some preliminary ideas on how these improvements can be implemented.
N.B.: The information below is the result of short preliminary studies. Unfortunately, there was not enough time to implement and test the ideas. As such, be mindful that it could still be incorrect or inconsistent.

I.1. Remnant model implementation

The most fundamental weakness of the current implementation, is the incorporation of remnant noise. Remnant noise is not explicitly modeled, but accounted for by means of Gaussian white noise at the HO model's feedback input y and output u. The variance of this noise is adjusted to the variance of signals y a e, respectively, by means of q^2 and r^2, as was explained in Subsection 2.2.2. For SI dynamics, this implementation was shown to work relatively well, since the remnant noise is predominantly white in the target frequency region ($\omega_{b,r}$ = 10 rad/s). For DI dynamics, however, the research showed that the more strongly coloured remnant noise ($\omega_{b,r}$ = 0.01 rad/s) causes estimation problems as explained in Section V-C-2 of the article.

A solution to this problem could be to explicitly incorporate remnant noise in the HO model. This has not been worked out yet, but some initial thoughts can already be laid out. The HO model with a first-order remnant model, would look like Figure I.1. At this point, it is not clear whether the remnant model can be successfully implemented while maintaining the open loop structure. Perhaps, the full closed loop model, including the CE output feedback, should be used in order to integrate the feedback effect of remnant noise.

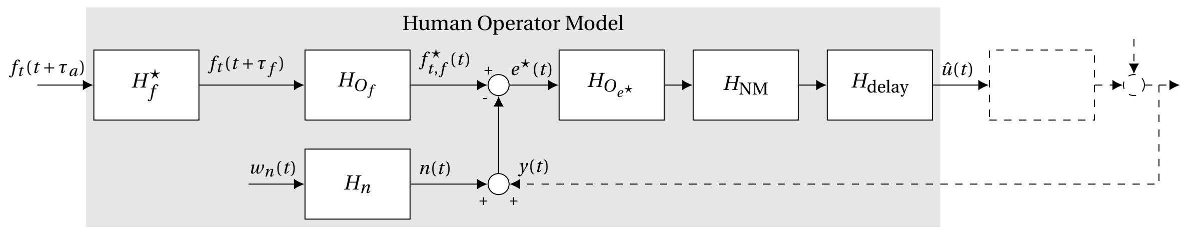


Figure I.1: Isolated HO model with incorporated remnant noise model

The nonlinear model state-space equations would look similar to the ones presented in the article. However, the noise matrix G (see Equation (I.1)) will likely have much more significant role. In the original model, this matrix was equal to the identity matrix ($G = I$), and $\boldsymbol{w_s} = \mathcal{N}(0, diag(Q_s))$. In the revised model, w_n becomes an element in $\boldsymbol{w_s}$ and G should make the mathematical link between the internal states of the remnant model H_n and the canonical states in the closed loop of the HO model. Note that the nonlinear output function was renamed h in order to avoid confusion (Equation (I.2)).

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}}(t) = f(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t), f_t(t+\tau_a), y(t)) + G\boldsymbol{w_s}(t) \tag{I.1}$$

$$\hat{u}(t) = h(\boldsymbol{x_s}(t), \boldsymbol{\theta}(t)) + v(t) \tag{I.2}$$

I.2. Parameter limits: Kalman gain projection

In the scientific paper, parameter limits are enforced using estimation projection (Simon, 2010). A second implementation of the parameter limitation was tried, using the principle of Kalman gain projection (Simon, 2010). The equations for this implementation, that replaces the the current state and parameter limitation steps in the article, are given below. Using the constraint matrices, a modified Kalman gain $\widetilde{K}$ is computed that maps either $\boldsymbol{x}^{+}_{\boldsymbol{s,k}}$ (I.3) or $\boldsymbol{\theta}^{+}_{\boldsymbol{k}}$ (I.6) to its feasible solution space. The main difference of mapping the estimate through the Kalman gain, rather than directly, is that the respective covariance matrix correction P^{+}_{k} is also mapped.

The idea behind using this rather than estimate projection is that P is also readjusted by applied limits, since P is recursively updated with the Kalman gain. This in turn should is thought to make parameters 'stick' less to their limits. As soon as a parameter meets its limit, it tends to stay there and not recover into feasible solution space easily.

This however did not work entire as intended. The correction to the Kalman gain can sometimes be exhorbitantly high ($\mathcal{O} \gg 10^{45}$). This happens when the measurement error r_k is extremely low, which is typically the case for the initialisation stages of the filter. It has not been investigated whether this is merely an artifact of simulated behaviour, or that this also shows up for experimental data.

Looking at (I.4) and (I.7), a significant parameter correction is only possible if the product of $\widetilde{K}_k r_k$ is significantly high. The Kalman gain thus has to compensate the small error when meeting a limit. Since the P matrix update is calculated with $\widetilde{K}_k$, this causes the elements in P to be enormous as well. And since P is calculated recursively, it never really gets rid of the high values.

The covariance matrix P then becomes ill-conditioned for the remainder of the estimation as it recursively updates. The estimated value can easily shoot from one limit to the other. It might be interesting to test this again, but then with a forgetting factor, which is commonly used to diminish ongoing recursive effects over time Boer and Kenyon (1998); Plaetinck et al. (2018).

State limitation with Kalman gain projection

$$\widetilde{K}_{s,k} = K_{s,k} - D^{T}_{s,k}(D_{s,k}D^{T}_{s,k})^{-1}(D_{s,k}\boldsymbol{x}^{+}_{\boldsymbol{s,k}} - \boldsymbol{d_{s,k}})(r^{T}_{k}S^{-1}_{s,k}r_k)^{-1}r^{T}_{k}S^{-1}_{s,k} \tag{I.3}$$

$$\widetilde{\boldsymbol{x}}^{+}_{\boldsymbol{s,k}} = \boldsymbol{x}^{-}_{\boldsymbol{s,k}} + \widetilde{K}_{s,k}r_k \tag{I.4}$$

$$P^{+}_{s,k} = (I - \widetilde{K}_{s,k}G_{s,k})P^{-}_{s,k}(I - \widetilde{K}_{s,k}G_{s,k})^{T} + \widetilde{K}_{s,k}R\widetilde{K}^{T}_{s,k} \tag{I.5}$$

Parameter limitation with Kalman gain projection

$$\widetilde{K}_{p,k} = K_{p,k} - D^{T}_{p,k}(D_{p,k}D^{T}_{p,k})^{-1}(D_{p,k}\boldsymbol{\theta}^{+}_{\boldsymbol{k}} - \boldsymbol{d_{p,k}})(r^{T}_{k}S^{-1}_{p,k}r_k)^{-1}r^{T}_{k}S^{-1}_{p,k} \tag{I.6}$$

$$\widetilde{\boldsymbol{\theta}}^{+}_{\boldsymbol{k}} = \boldsymbol{\theta}^{-}_{\boldsymbol{k}} + \widetilde{K}_{p,k}r_k \tag{I.7}$$

$$P^{+}_{p,k} = (I - \widetilde{K}_{p,k}G^{tot}_{p,k})P^{-}_{p,k}(I - \widetilde{K}_{p,k}G^{tot}_{p,k})^{T} + \widetilde{K}_{p,k}R\widetilde{K}^{T}_{p,k} \tag{I.8}$$

I.3. Observability Gramian

The research shows that the observability of certain HO parameters is limited. For example, the feasibility batches in the paper showed that the parameters ω_n, ζ_n and τ_v already start to drift from their true value in the early, invariant stages of the simulation. The observability Gramian (sometimes Grammian) could serve as an explicit measure to indicate how observable trends in the individual parameters are also locally, i.e. around parameter time-variations Moszczynski et al. (2019). In conjunction with the incorporation of a remnant model, this metric could quantify the improvement that this or other developments make.

The observability Gramian is basically the square of the observability matrix, as explained in the graduation thesis of Miletović (2014, Eq. 4.15). In general, the rank of the observability Gramian determines whether a system is fully observable or not (Chen, 1998). Moreover, the observability Gramian can also provide information on the observability of individual states. The condition number of the Gramian, which is the ratio of smallest and largest eigenvalue, can used to determine whether the Gramian is ill-conditioned and thus whether the system contains poorly observable states Miletović (2014). Another research that uses the eigenvalues of the Gramian to find more information on the observability of states, is Moszczynski et al. (2019).

The research by Qi et al. (2014) provides more information and references on how "*the degree of observability can be quantified by making use of a variety of different measures of the empirical observability Gramian*". These methods make use of the Gramian's eigenvalues, trace, determinant and the aforementioned condition number. How this works exactly was not yet looked into in further detail due to time constraints and left for future work.

Note that there is a difference between the theoretical, infinite Gramian and an 'intermediate' n-step Gramian.

The first is the theoretical concept, indicating the observability of initial states after an infinite observation time (Chen, 1998). This Gramian is often computed using the Lyapunov equation, as this is a much more computationally feasible compared to an infinite sum or integration.

The n-step Gramian, on the other hand, gives an indication of the initial states after n discrete observation steps. As pointed out by Miletović (2014), for time-invariant discrete systems of the form:

$$\boldsymbol{x}_{k+1} = \Phi_k \boldsymbol{x}_k + \Psi_k u_k + \Gamma_k \boldsymbol{w}_k \tag{I.9}$$

$$\boldsymbol{y}_k = H_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{I.10}$$

the n-step Gramian W_{O_n} can be calculated by:

$$W_{O_n} = O_n^T O_n = H_0^T H_0 + \sum_{k=1}^{n} (\Phi_k^T)^k H_k^T H_k (\Phi_k)^k \tag{I.11}$$

This means that, for time-varying systems, where the transition matrix varies per time step, the n-step Gramian is found using:

$$W_{O_n} = O_n^T O_n = H_0^T H_0 + \sum_{k=1}^{n} \Phi_1^T \Phi_2^T \cdots \Phi_k^T H_k^T H_k \Phi_k \cdots \Phi_2 \Phi_1 \tag{I.12}$$

Since the DEKF is a dual filter, observability Gramians can be calculated for both of the concurrent filters. For the state filter $\Phi_k = \Phi_{s,k-1}$ and $H_k = G_{s,k}$ and for the parameter filter $\Phi_k = \Phi_{p,k-1}$ and $H_k = G_{p,k}^{\text{tot}}$. Note that there is a difference in notation between the work of Miletović and this thesis. Miletović indicates the discrete transition matrix Φ from time step $k-1$ to k with the final index k, whereas in the DEKF algorithm in the scientific article, this same transition matrix is indicated by the initial index $k-1$.

For the DEKF as presented in the scientific paper (see Section III-A of the paper in Part I for the detailed notation), two types of n-step Gramians for each filter are expected to provide useful information:

1. One-step Gramian The first is the 1-step Gramian W_O at each time step k. This Gramian provides infomation on the observability from one time step to the next, and is therefore the most likely candidate for interesting results. The progression of Gramians over time could thus provide information on the progression of the observability of individual parameters.

$$W_{O,k} = \Phi_k^T G_k^T G_k \Phi_k \tag{I.13}$$

2. n-step Gramian over the full measurement range A second option would be the n-step Gramian that is calculated with the discrete transition matrices of all time steps within the measurement window. The run-in time is left out, since this is used to make the filter converge. This Gramian could provide information on the observability of the initial after the entire measurement window has past

$$W_{O,\text{meas}} = \sum_{k=k_{t=0}}^{N_{\text{meas}}} \Phi_1^T \Phi_2^T \cdots \Phi_k^T H_k^T H_k \Phi_k \cdots \Phi_2 \Phi_1 \tag{I.14}$$

Bibliography

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.

C.-T. Chen. *Linear system theory and design.* Oxford University Press, Inc., 1998.

E. Donges. A Two-Level Model of Driver Steering Behavior. *Human Factors*, 20(6):691–707, December 1978.

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.

H. Godthelp. Vehicle Control During Curve Driving. *Human Factors*, 28(2):211–221, 1986.

K. Ito and M. Ito. Tracking Behavior of Human Operators in Preview Control Systems. *Electrical Engineering in Japan (English translation of Denki Gakkai Ronbunshi)*, 95(1):120–127, 1975.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

M. F. Land and J. Horwood. Which parts of the road guide steering? *Nature*, 377:339 – 340, September 1995. doi: 10.1038/377339a0.

M. F. Land and D. N. Lee. Where we look when we steer. *Nature*, 369:742 – 744, June 1994. doi: 10.1038/369742a0.

O. Lappi and C. Mole. Visuomotor Control, Eye Movements, and Steering: A Unified Approach for Incorporating Feedback, Feedforward, and Internal Models. *Psychological Bulletin*, 144(10):981–1001, October 2018. doi: 10.1037/bul0000150.

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, December 1969. doi: 10.1109/TMMS.1969.299906.

C. C. MacAdam. Application of an optimal preview control for simulation of closed-loop automobile driving. *IEEE Transactions on systems, man, and cybernetics*, 11(6):393–399, 1981.

T. K. Mandal and Y. Gu. Online Pilot Model Parameter Estimation Using Sub-Scale Aircraft Flight Data. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego (CA)*, number AIAA-2016-0636, 2016. 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. doi: 10.1109/THFE.1967.234304. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1698271`.

D. T. McRuer, R. W. Allen, D. H. Weir, and R. H. Klein. New Results in Driver Steering Control Models. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 19(4):381–397, August 1977. ISSN 0018-7208.

I. Miletović. Optimal Reconstruction of Flight Simulator Motion Cues. Master's thesis, Delft University of Technology, March 2014.

G. J. Moszczynski, M. Giamou, J. M. Leung, J. Kelly, and P. R. Grant. An observability based approach to flight path reconstruction of uninformative coupled aircraft trajectories: A case study considering stall maneuvers for aircraft certification. In *AIAA Scitech 2019 Forum*, page 0012, 2019. doi: 10.2514/6.2019-0012.

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*, 48(5):468–485, October 2018. doi: 10.1109/THMS.2017.2761342.

L. Nelson and E. Stear. The simultaneous on-line estimation of parameters and states in linear systems. *IEEE Transactions on automatic Control*, 21(1):94–98, 1976.

W. Plaetinck, D. M. Pool, M. M. van Paassen, and M. Mulder. Online Identification of Pilot Adaptation to Sudden Degradations in Vehicle Stability. In *Proceedings of the 2nd IFAC Conference on Cyber-Physical & Human-Systems, Miami (FL)*, 2018.

A. Popovici, P. M. T. Zaal, D. M. Pool, and M. Mulder. Relating Eye Activity Measures to Human Controller Remnant Characteristics. In *Proceedings of the 13th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Kyoto, Japan*, 2016.

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.

A. J. Pronker, D. A. Abbink, M. M. Van Paassen, and M. Mulder. Estimating driver time-varying neuromuscular admittance through LPV model and grip force. In *Proceedings of the the 20th IFAC World Congress, Toulouse, France*, volume 50 of *IFAC-PapersOnLine*, pages 14916–14921. Elsevier, 2017. doi: 10.1016/j.ifacol.2017.08.2539.

J. Qi, K. Sun, and W. Kang. Optimal pmu placement for power system dynamic state estimation by using empirical observability gramian. *IEEE Transactions on power Systems*, 30(4):2041–2054, 2014.

J. Rojer, D. M. Pool, M. M. van Paassen, and M. Mulder. UKF-based Identification of Time-Varying Manual Control Behaviour. In *Proceedings of the 14th IFAC Symposium on Analysis Design and Evaluation of Human Machine Systems Tallinn, Estonia*, pages 109–114, 2019.

D. D. Salvucci and R. Gray. A two-point visual control model of steering. *Perception*, 33(10):1233–1248, December 2004. ISSN 1468-4233. doi: 10.1068/p5343.

J. R. Schiess and V. R. Roland. Kalman Filter Estimation of Human Pilot-Model Parameters. Technical Report NASA-TN-D-8024, NASA Langley Research Center, Hampton (VA), November 1975. URL `http://hdl.handle.net/2060/19760006748`.

C. Sentouh, P. Chevrel, F. Mars, and F. Claveau. A Sensorimotor Driver Model for Steering Control. In *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2462 – 2467, October 2009. ISBN 978-1-4244-2793-2. doi: 10.1109/ICSMC.2009.5346350.

T. B. Sheridan. Three models of preview control. *IEEE Transactions on Human Factors in Electronics*, (2): 91–102, 1966.

G. J. Silva, A. Datta, and S. Bhattacharyya. Controller design via padé approximation can lead to instability. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, volume 5, pages 4733–4737. IEEE, 2001.

D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, 2010.

K. van der El. *How Humans Use Preview Information in Manual Control.* PhD thesis, Delft University of Technology, Faculty of Aerospace Engineering, December 2018.

K. van der El, D. M. Pool, H. J. Damveld, M. M. van Paassen, and M. Mulder. An Empirical Human Controller Model for Preview Tracking Tasks. *IEEE Transactions on Cybernetics*, 46(11):2609–2621, November 2016. doi: 10.1109/TCYB.2015.2482984.

K. van der El, S. Padmos, D. M. Pool, M. M. van Paassen, and M. Mulder. Effects of Preview Time in Manual Tracking Tasks. *IEEE Transactions on Human-Machine Systems*, 48(5):486–495, October 2018. doi: 10.1109/THMS.2018.2834871.

K. van der El, D. M. Pool, and M. Mulder. Analysis of Human Remnant in Pursuit and Preview Tracking Tasks. In *Proceedings of the 14th IFAC Symposium on Analysis Design and Evaluation of Human Machine Systems Tallinn, Estonia*, pages 145–150, 2019a.

K. van der El, D. M. Pool, M. M. van Paassen, and M. Mulder. Effects of Target Trajectory Bandwidth on Manual Control Behavior in Pursuit and Preview Tracking. *IEEE Transactions on Human-Machine Systems*, 2019b. doi: 10.1109/THMS.2019.2947577.

A. van Grootheest, D. M. Pool, M. M. van Paassen, and M. Mulder. Identification of Time-Varying Manual Control Adaptations with Recursive ARX Models. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference, Kissimmee (FL)*, number AIAA-2018-0118, 2018. doi: 10.2514/6.2018-0118.

R. J. Wasicko, D. T. McRuer, and R. E. Magdaleno. Human pilot dynamic response in single-loop systems with compensatory and pursuit displays. Technical report, SYSTEMS TECHNOLOGY INC HAWTHORNE CA, December 1966.

C. You, J. Lu, and P. Tsiotras. Nonlinear Driver Parameter Estimation and Driver Steering Behavior Analysis for ADAS Using Field Test Data. *IEEE Transactions on Human-Machine Systems*, 47(5):686–699, 2017.

L. R. Young. On Adaptive Manual Control. *Ergonomics*, 12(4):635–674, 1969.

P. Zaal and B. Sweet. Estimation of Time-Varying Pilot Model Parameters. In *AIAA Modeling and Simulation Technologies Conference*, page 6474, 2011.

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, D. M. Pool, Q. P. Chu, M. M. van Paassen, M. Mulder, 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. doi: 10.2514/1.42843.