Pontificia Universidad Católica del Perú

Escuela de Posgrado

Effect of the Number of Samples on Parameter Bias in Closed-Loop System Identification with Routine-Operating Data

Tesis para obtener el grado académico de Maestra en Ingeniería de Control y Automatización que presenta:

Jocelyn Alicia Durand Solis

Asesor PUCP (PUCP): *Dr. Carlos Gustavo Pérez Zúñiga*

Co-Asesor de la Universidad no PUCP: *Dr. Yuri Shardt*

Lima, 2025

Informe de Similitud

Yo, Carlos Gustavo Pérez Zúñiga, docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesor PUCP de la tesis titulada "Effect of the Number of Samples on Parameter Bias in Closed-Loop System Identification with Routine-Operating Data." del autor Jocelyn Alicia Durand Solis,

dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 7**%.** Así lo consigna el reporte de similitud emitido por el software Turnitin el 24/11/2025.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha:

Lima, 25 de noviembre del 2025.

<table>
<tr><td colspan="2">Apellidos y nombres del asesor:
Carlos Gustavo Pérez Zúñiga</td></tr>
<tr><td>DNI: 41864666</td><td rowspan="2">Firma:
  </td></tr>
<tr><td>ORCID: 0000-0001-5946-1395</td></tr>
</table>

Honour Statement

I, Jocelyn Alicia Durand Solis, hereby declare that I have written this thesis independently and have not used any other sources than those specified. All thoughts that have been taken either directly or indirectly from other sources have been explicitly identified as such. This thesis has not been submitted for credit in any other degree programme or institution nor has it been previously published.

(Jocelyn Alicia Durand Solis)

Ilmenau, October 29th, 2025

Eidesstattliche Erklärung

Hiermit versichere ich, Jocelyn Alicia Durand Solis, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet. Diese Arbeit wurde in gleicher oder ähnlicher Form bisher noch keiner anderen Prüfungsbehörde vorgelegt oder veröffentlicht.

Abstract

In many industrial applications, system models are identified using routine closed-loop operating data. However, such data are often limited in length and variability due to operational constraints and feedback suppression, which challenge the reliability of parameter estimates. A key question therefore arises: how many data points are needed to obtain trustworthy estimates when working with routine operating data?

This thesis addresses this question by evaluating two complementary methods for constructing confidence regions of model parameters under closed-loop conditions with different data length. The first method is an empirical Monte Carlo approach, which estimates the parameters of the autoregressive with exogenous inputs (ARX) model through repeated simulations and applies kernel density estimation (KDE) to construct the empirical 95% confidence regions. The second method is the sign-perturbed sums (SPS) approach, which guarantees nonasymptotic confidence regions with exact coverage probability under the assumption of symmetric and independent noise.

Using three first-order plus dead-time (FOPDT) process models and different noise levels, the thesis investigates how sample size and noise variance influence the accuracy and robustness of parameter estimates. The results show that classical asymptotic confidence intervals often underestimate uncertainty with small datasets, while SPS maintains valid coverage. Three-dimensional plots of the Monte Carlo results and a summary plot of the median versus data size indicate that, for the three plants studied, at least 2,000 samples are required to obtain accurate, approximately unbiased estimates with a 95% confidence region. Overlays of SPS and KDE confidence regions provide additional insight into bias. They suggest that it is possible to obtain 95% confidence regions with a dataset on the order of 2,000 samples too. The effect of noise variance was also examined. It was found that the controller was largely able to suppress its impact on system identification.

Zusammenfassung

In vielen industriellen Anwendungen werden Systemmodelle auf Basis von Daten identifiziert, die aus dem geschlossenen Regelkreis mit Festwertregelung stammen. Solche Daten sind jedoch aufgrund betrieblicher Einschränkungen und der Rückkopplung häufig sowohl in ihrer Länge als auch in ihrer Variabilität eingeschränkt, was die Verlässlichkeit der Parameterschätzungen beeinträchtigen kann. Daraus ergibt sich die zentrale Frage: Wie viele Datenpunkte sind erforderlich, um bei Daten von Festwertregelung vertrauenswürdige Parameterschätzungen zu erhalten?

Diese Arbeit geht dieser Frage nach, indem zwei komplementäre Methoden zur Bestimmung von Vertrauensbereichen für Modellparameter im geschlossenen Regelkreis unter Festwertregelung mit unterschiedlichen Datengrößen untersucht werden. Die erste Methode ist ein empirischer Monte-Carlo-Ansatz, bei dem die Parameter eines autoregressiven Modells mit externen Eingängen (ARX) durch wiederholte Simulationen geschätzt und empirische 95%-Vertrauensbereiche mittels Kerndichteschätzung (KDE) konstruiert werden. Die zweite Methode ist der Ansatz der Vorzeichen-gestörten Summen (SPS), der unter der Annahme von unabhängigem und symmetrischem Rauschen nichtasymptotische Vertrauensbereiche mit garantierter Abdeckwahrscheinlichkeit liefert.

Die Auswirkungen von Stichprobengröße und Rauschvarianz auf die Genauigkeit und Robustheit der Parameterschätzungen werden anhand von drei PT_1-mit-Totzeit-Prozessmodellen und unterschiedlichen Rauschpegeln untersucht. Die Ergebnisse zeigen, dass klassische asymptotische Vertrauensintervalle bei kleinen Datensätzen die Unsicherheit häufig unterschätzen, während SPS eine gültige Abdeckung sicherstellt. Dreidimensionale Darstellungen der Monte-Carlo-Ergebnisse sowie eine Übersichtsabbildung des Medianwerts in Abhängigkeit von der Stichprobengröße deuten darauf hin, dass für die drei betrachteten Prozesse mindestens etwa 2 000 Datenpunkte pro ARX-Schätzung erforderlich sind, um akkurate Schätzungen mit einem 95%-Vertrauensbereich zu erhalten. Die Überlagerungen von SPS- und KDE-Vertrauensbereichen liefert zusätzliche Einblicke in den Bias und legt nahe, dass auch mit Datensätzen auch im Bereich von 2 000 Stichproben Parameter mit einem 95%-Vertrauensbereich geschätzt werden können. Zudem wurde der Einfluss der Rauschvarianz untersucht. Der Regler unterdrückt diesen Effekt in der Systemidentifikation weitgehend.

Resumen

En muchas aplicaciones industriales, la identificación de sistemas de los modelos de los procesos se realizan con datos de operación rutinaria en lazo cerrado. Sin embargo, dichos datos suelen estar limitados en tamaño y variabilidad debido a restricciones de operación y a los efectos de la retroalimentación. Esto dificulta la estimación confiable de los parámetros del proceso. Surge entonces una pregunta clave: ¿cuántos datos son necesarios para obtener estimaciones confiables cuando se trabaja con datos de operación rutinaria?

Esta tesis aborda esta cuestión evaluando dos métodos complementarios para la construcción de regiones de confianza de los parámetros del modelo en condiciones de lazo cerrado con conjuntos de datos de diferente tamaño. El primer método consiste en la generación de varios conjuntos de datos con varias simulaciones de Monte Carlo, luego se procede con la estimación de los parámetros del modelo autorregresivo con entradas exógenas (ARX) y se aplica la estimación de densidad del núcleo (KDE) para construir y estimar regiones de confianza empíricas del 95 %. El segundo método es el de sumas perturbadas por signo (SPS), que garantiza regiones de confianza no asintóticas con probabilidad de cobertura exacta, bajo la suposición de ruido simétrico e independiente.

Utilizando tres modelos de proceso de primer orden con tiempo muerto (FOPDT) y diferentes niveles de ruido, la tesis investiga cómo el tamaño de la muestra y la varianza del ruido influyen en la precisión y la robustez de las estimaciones de parámetros. Los resultados muestran que los intervalos de confianza asintóticos clásicos suelen subestimar la incertidumbre con conjuntos de datos pequeños, mientras que SPS mantiene una cobertura válida. Gráficos tridimensionales de los resultados de Monte Carlo y un gráfico resumen de la mediana frente al tamaño de muestra indican que, para los tres procesos estudiados, se requieren al menos 2 000 muestras por ajuste ARX para obtener estimaciones precisas y aproximadamente insesgadas con una región de confianza del 95 %. Las superposiciones de las regiones de confianza de SPS y KDE ofrecen información adicional sobre el sesgo y sugieren que también es posible obtener regiones de confianza del 95 % con conjuntos de datos en el orden de las 2 000 muestras. También se examinó el efecto de la varianza del ruido, y se observó que el controlador suprime en gran medida su impacto en la identificación del sistema.

Acknowledgments

To my beloved mother, Carmen Solis, for her support, patience, and encouragement throughout this master's program.

To the professors from Peru and Germany in the double degree program, for generously sharing their knowledge and for always being open to questions.

I would also like to extend my gratitude to Dr. Oshima for his support and guidance regarding the SPS method.

Table of Contents

List of Figures

List of Tables

List of Symbols and Abreviations

α	Significance level (coverage $1-\alpha$)
θ	Parameter vector
$\chi^2_{\nu,q}$	Chi-squared quantile with ν dof at level q
$\hat{y}(t \mid \theta)$	One-step-ahead prediction
$\mathbb{E}[\cdot]$	Expectation operator
PID	Proportional–Integral–Derivative controller
PI	Proportional–Integral controller
P	Proportional controller
$\mathrm{var}(\cdot)$	Variance
σ, $\hat{\sigma}$	Standard deviation and its estimate
τ_D	Time delay
τ_P	Time constant
$\varepsilon(t)$	Prediction error $y(t)-\hat{y}(t)$
$\varphi(t)$	Regressor vector
$A(q^{-1}), B(q^{-1}), C(q^{-1}), D(q^{-1}), F(q^{-1})$	Model polynomials in q^{-1}
a_1, b_1	ARX(1,1) coefficients
d	Discrete-time input delay (samples)
$e(t)$	Disturbance/noise entering the plant (not the prediction error)

$G(s)$ Continuous-time transfer function

$G(z)$ Discrete-time transfer function

G_c, G_p, G_l Controller, plant, and disturbance-path transfer functions

K Process (steady-state) gain

K_d Derivative gain (ideal form: $K_d = K_p T_d$)

K_i Integral gain (ideal form: $K_i = K_p/T_i$)

K_p Proportional gain

L Dead time / input delay

n_A, n_B, n_C, n_D, n_F Orders of the corresponding polynomials

n_k Input delay (samples) in discrete time

n_X, n_Y Controller-related orders used in identifiability conditions

P Asymptotic covariance matrix of parameter estimates

q^{-1} Backward-shift (unit-delay) operator

$r(t)$ Reference / setpoint

T_d Derivative time constant (ideal form: $T_d = K_d/K_p$)

T_i Integral time constant (ideal form: $T_i = K_p/K_i$)

T_s Sampling period

$t_{\nu,q}$ Student's t quantile with ν dof at level q

$u(t)$ Input / manipulated variable

$y(t)$ Output variable

ACF Autocorrelation Function

ARX AutoRegressive with eXogenous input model

CCF Cross-Correlation Function

CI Confidence Interval

CL Closed Loop

DCS Distributed Control System

FIR Finite Impulse Response (model)

FOPDT First-Order Plus Dead Time

KDE Kernel Density Estimation

LSE Least-Squares Estimate

MPC Model Predictive Control

OL Open Loop

P Proportional controller

PDF Probability Density Function

PEM Prediction-Error Method

PI Proportional–Integral controller

PID Proportional–Integral–Derivative controller

PLC Programmable Logic Controller

PRBS Pseudo-Random Binary Sequence

SNR Signal-to-Noise Ratio

SPS Sign-Perturbed Sums

ZOH Zero-Order Hold

Chapter 1: Introduction

Section 1.1: State of Art

In modern process industries, process modeling based on operational data is an essential tool for monitoring, control design, and system optimization. However, obtaining reliable models from such data can be challenging, particularly when the data are collected under routine closed-loop operation, where external excitation or planned experiments are limited or infeasible (Y. Shardt, 2012). System identification in these conditions is complicated by feedback loops that suppress signal variability and by operational constraints that restrict the length and richness of available data.

An example for this application is the use of model predictive control (MPC), which depends on models identified from routine operating data and the control performance is directly affected by the quality of model identification, especially under input and output constraints (Yusuf, 2019). This raises a fundamental question: when only routine operating data are available, how many data points are sufficient to obtain trustworthy parameter estimates?

This issue is evaluated in this work considering that the process is modeled as a FOPDT system, a common structure in control and chemical engineering that captures the essential dynamics and delays of many processes. When working with small datasets from routine closed-loop operations, the estimated model parameters (such as process gain and time constant) may become biased or imprecise (Y. Shardt, 2012). The term bias refers to the difference between the true parameter value and the expected value of an estimator. These inaccuracies can interfere with control monitoring, make fault detection less reliable, or lead to poor controller adjustments.

Shardt and Huang (2011b) investigated how sampling time, process delay, and controller dynamics influence the identifiability of models obtained from routine closed-loop operating data. Using a general Prediction Error Model framework, they presented theoretical conditions described in Theorem 1, Eq. (2.13), which define when a process can be identified from closed-loop data even without external excitation. The theorem establishes a relationship among the model orders that must hold for the system to be identifiable. In their work, this is examined by simulations and experiments on a heated-tank setup, they observed that the discrete time delay changes with the chosen sampling time. When the sampling is too slow, important dynamic features can be lost, and the identified model may no longer reflect the true system behavior. The results also suggest that the controller's relative order influences also the identifiability of the system. Taken together, the study shows that even data collected during normal plant operation as routine operating data can be used for identification, as long as the sampling time and delay are appropriate for reliable system identification.

Parameter uncertainty is often assessed using asymptotic methods, which assume large sample sizes and Gaussian noise. Under such conditions and depending on the model, the least-squares estimate (LSE) or the general prediction-error method (PEM) can be used to estimate the model parameters

(Ljung, 1999). However, in small-sample settings, these asymptotic results may be unreliable. The resulting confidence regions often underestimate the true uncertainty, which provides a misleading sense of precision (Campi, Ko, & Weyer, 2009).

To overcome these limitations, recent research has investigated exact, nonasymptotic methods for constructing confidence regions that remain valid with finite datasets. Among these, the SPS method stands out. Originally developed for open-loop linear systems (Csáji, Campi, & Weyer, 2012), SPS constructs confidence regions around a nominal estimate (such as LSE) with a guaranteed user-specified coverage probability. Its key assumption is symmetric and independent noise. SPS has been extended to closed-loop systems and different model structures, which makes it especially relevant when data are collected under feedback control (Csáji & Weyer, 2015).

Despite its theoretical strengths, the practical applicability of the SPS method to closed-loop routine operating data remains an open topic. The method has been extensively studied in simulation settings by Csáji and Weyer (2015), and demonstrated its ability to construct exact, nonasymptotic confidence regions in linear regression models, including in feedback systems. More recently, Oshima *et al.* (Oshima, Kim, Shardt, & Sotowa, 2024) have also contributed to the study, construction and validation of nonasymptotic confidence regions for linear regression models in closed-loop scenarios with the SPS method.

Section 1.2: Research objectives

Building upon these foundations, this work includes an evaluation of the SPS method and a Monte-Carlo-based empirical approach: repeated simulations of a closed-loop system are conducted, and synthetic datasets are generated under varying sample sizes and noise conditions. For each configuration, ARX model parameters are estimated, and kernel density estimation (KDE) is used to construct the empirical confidence regions that capture the central 95% of the estimated parameter distribution (Chen, 2017; Silverman, 2018). The median is used as a robust measure of central tendency. This approach allows the quantification of bias and variance as functions of data length and noise, offering a practical benchmark for evaluating small-sample behavior. The empirical Monte Carlo + KDE method shows how the confidence regions behave in practice and this method also give starting parameter estimates according to different data length and noise variance configurations to construct and evaluate the SPS region, which provides exact finite-sample guarantees. Together, these approaches help identify when asymptotic theory fails, when nonasymptotic methods succeed, and how trustworthy parameter estimates can be achieved from small amounts of routine operating data.

In contrast to the work of Csáji and Weyer (2015), which studies the SPS method and demonstrated its exact finite-sample properties for open-loop, closed-loop and generic linear regression settings, this thesis applies the SPS framework to closed-loop identification using routine operating data. This work overlays the exact non-asymptotic SPS confidence regions with empirical Monte Carlo + KDE–based regions, enabling a direct comparison between theoretical and practical confidence regions. Additionally, the analysis is performed on a first-order plus dead-time (FOPDT) process, a representative industrial system where delay and feedback significantly affect identifiability. The results show that both the asymptotic and non-asymptotic confidence regions exhibit a high degree of overlap and successfully contain the true plant parameters, providing new empirical validation of the SPS method under realistic closed-loop operating conditions.

The goal of this thesis is to determine how many data points are needed to obtain a reliable parameter estimate under closed-loop routine operating conditions. This work also investigates how data length, noise level, and plant dynamics influence the accuracy and robustness of parameter estimation under the mentioned conditions.

Section 1.3: Methodology

The following diagram in Fig. 1.1 summarizes the methodology applied to obtain asymptotic and nonasymptotic regions and overlays.

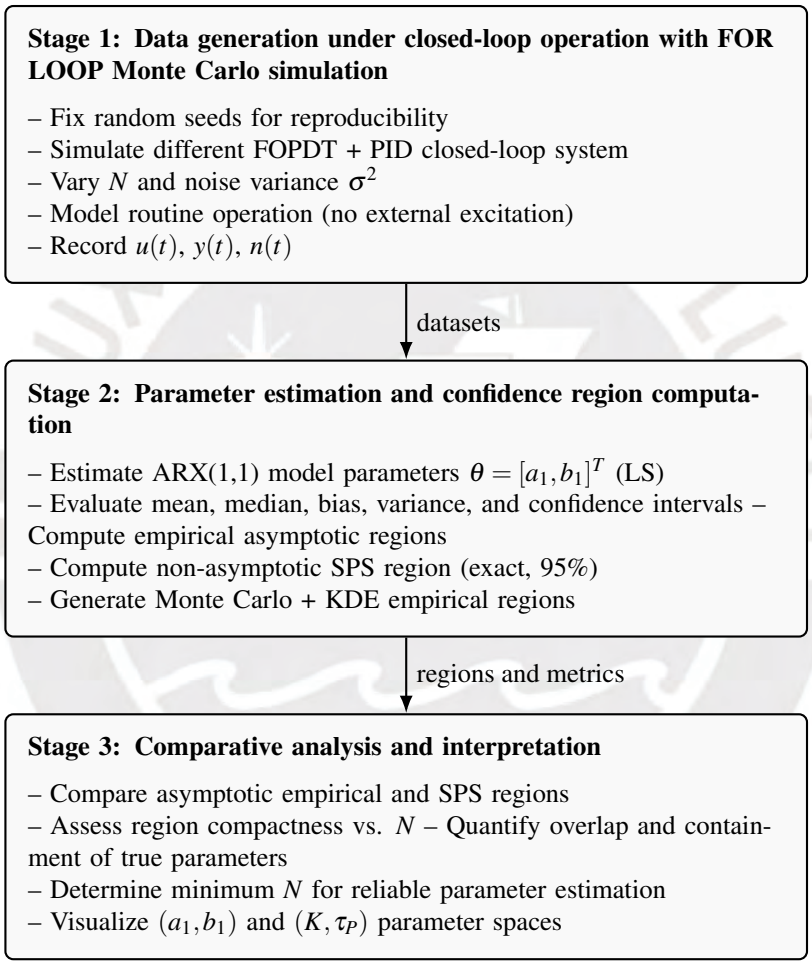


Figure 1.1: Summary of the methodology for closed-loop system identification and confidence region analysis.

Chapter 2: Background Information

Section 2.1: Introduction to System Identification

System identification is the process of developing mathematical models that describe the dynamics of a system, based on observed input-output data. Accurate models permit control performance monitoring, fault diagnosis, and controller tuning. For these reasons they are very important in the process industry. Models can be classified as white-box, grey-box, or black-box depending on how much physical knowledge is available.

White-box models are obtained completely from first principles and physical laws. They are easy to understand and usually give accurate descriptions of system behaviour. However, they can be difficult to use when the system is complex or has unknown dynamics (Y. A. Shardt, 2022). An example of this kind of model is the mass–spring–damper system, which can be described by the second-order differential equation shown in Eq. (2.1).

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t) \tag{2.1}$$

On the other hand, black-box models are obtained completely from data and do not rely on any explicit or assumed physical estructures or laws. They are particularly flexible and useful when only input-output measurements are available. Unfortunately, these models are often harder to interpret. A black-box example is the finite impulse response (FIR) model, which relates the current output only to past input values, without including any autoregressive terms. It contains no feedback and is simply a finite weighted sum of past inputs as shown in Eq. (2.2), and where the dynamics can be also represented by polynomial operators in the backward shift q^{-1} as shown in Eq. (2.3).

$$y(t) = \sum_{k=1}^{n_b} b_k\, u(t-k) + e(t), \tag{2.2}$$

$$y(t) = B(q^{-1})\, u(t) + e(t), \tag{2.3}$$

Between these extremes lie grey-box models, which combine structure with flexibility as shown in Eq. (2.4). This type of model include partial knowledge of the system while estimating unknown parameters from data.

$$\dot{x}(t) = f(x(t), u(t), \theta) \tag{2.4}$$

Traditionally, the system identification is often carried out under open-loop conditions. Here the process is excited with test signals such as pseudo-random binary sequences or step inputs. While this approach is effective for commissioning or non-critical applications, it is often infeasible for unstable open loop or safety-critical systems. In such cases, identification must instead be performed in closed loop. However, feedback introduces correlation between inputs and disturbances, complicating the estimation of the parameters (Y. A. Shardt, 2022).

Closed-loop system identification can be divided into two categories: The first one is with external excitation, where reference are varied or test signals are applied. The second one is without external excitation, also referred to as routine operating data, where the setpoint remains fixed and variability arises only from disturbances (Y. A. Shardt & Huang, 2011b).

In this thesis, the model structure of interest was the ARX model. It was chosen due to its simplicity for closed-loop identification using routine operating data. It extends the basic FIR model representation by including both past inputs and past outputs as regressors as shown in Eq. (2.5). Its standard form is shown in Eq. (2.6), where $A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}$ captures the autoregressive part, $B(q^{-1}) = b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}$ represents the input dynamics, and $e(t)$ is a white noise disturbance.

$$y(t) = -\sum_{i=1}^{n_a} a_i y(t-i) + \sum_{k=1}^{n_b} b_k u(t-k) + e(t), \tag{2.5}$$

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t), \tag{2.6}$$

In general, a model can be written in regression form as in Eq. (2.7), where $\varphi(t)$ is the regressor vector containing past inputs and outputs, and θ is the parameter vector. Because the ARX model is linear in its parameters, it can be efficiently estimated using the least-squares method. At the same time, it remains flexible enough to capture essential process dynamics and input delays, which are common in practical control systems.

$$y(t) = \varphi(t)^T \theta + e(t), \tag{2.7}$$

Model parameters for structures such as FIR and ARX, which are linear in their parameters, are usually estimated using the LSE method (Ljung, 1999; Y. A. Shardt, 2022), which minimizes the sum of squared error between observed data and a linear regressor, and yields the parameter estimates as shown in Eq. (2.8).

$$\hat{\theta}_{LS} = \arg\min_{\theta} \sum_{t=1}^{N} \left(y(t) - \varphi(t)^T \theta \right)^2 \tag{2.8}$$

Another common used estimation method in system identification is PEM, which estimates the parameters by minimizing the prediction error $\varepsilon(t, \theta)$, defined in Eq. (2.9) as the difference between the measured output $y(t)$ and the model's one-step-ahead prediction $\hat{y}(t|\theta)$ based on the chosen model. The

prediction function $h(t,\theta)$ may depend on past inputs, past outputs, and noise terms. The PEM estimator is then defined in Eq. (2.10).

$$\hat{y}(t|\theta) = h(t,\theta), \quad \varepsilon(t,\theta) = y(t) - \hat{y}(t|\theta), \tag{2.9}$$

The main point behind PEM is that the predictor $\hat{y}(t|\theta)$ in Eq. (2.9) depends on the chosen model structure. For ARX models, the one-step-ahead predictor is linear in the parameters, and in this case PEM reduces to LSE as in Eq. (2.7). In general, however, PEM provides a broader framework for system identification: one always minimizes the prediction error, but depending on the model structure, the optimization may be simple (closed-form LSE) or more complex, requiring iterative numerical methods.

$$\hat{\theta}_{PEM} = \arg\min_{\theta} \sum_{t=1}^{N} \big(y(t) - \hat{y}(t|\theta)\big)^2. \tag{2.10}$$

Regardless of the estimation method that has been selected, residual analysis is essential for validating model quality. Residuals $\varepsilon(t) = y(t) - \hat{y}(t)$ should ideally be white noise, or in other words, uncorrelated, zero-mean, and independent of past inputs. Deviations from these properties suggest model mismatch, underfitting, or insufficient excitation. Statistical tools such as the autocorrelation function (ACF) and cross-correlation function (CCF) are commonly used to assess residuals (Y. A. Shardt, 2022).

For reliable system identification, it is also important that the estimated parameters are both unbiased and consistent. Consistency means that, with infinite or certain amount of data points, the estimates converge to the true parameter values. Low variance is equally important, since it ensures that confidence intervals around the estimates are tight enough to enable meaningful statistical tests of model quality (Ljung, 1999).

Beyond these asymptotic properties, a fundamental prerequisite in the closed-loop routine operating data setting is identifiability. To formalize this, the general prediction error model is considered as shown in Eq. (2.12), where $A(q^{-1}), B(q^{-1}), C(q^{-1}), D(q^{-1}), F(q^{-1})$ are polynomials in the backward shift operator q^{-1}, and n_k is the input delay.

$$n_k \approx \lfloor L/T_s \rfloor \tag{2.11}$$

According to the theorem 1 of identifiability (Y. A. Shardt & Huang, 2011b), such a system can be identified without external excitation if the inequality among the polynomial orders holds as in Eq. (2.13), where n_A, n_B, n_C, n_D, n_F denote the polynomial orders of the corresponding model components and n_X, n_Y corresponds to $n_X = n_Y = 0$ for proportional (P) controllers, $n_X = n_Y = 1$ for proportional and integral (PI) controllers and $n_X = 2$ and $n_Y = 1$ for proportional, integral, and derivative (PID) controllers. This condition ensures that different parameter vectors yield distinct probability distributions of the observed data, thereby guaranteeing identifiability.

$$A(q^{-1})y(t) = \frac{B(q^{-1})}{F(q^{-1})}u(t-n_k) + \frac{C(q^{-1})}{D(q^{-1})}e(t), \tag{2.12}$$

$$\max(n_Y + n_k - n_F - n_A,\, n_Y - n_B) \;\geq\; n_D + \min(n_C + n_F + n_Y,\, n_A + n_F + n_Y,\, n_B + n_X), \tag{2.13}$$

For the ARX model structure some polynomials orders can be simplified as $n_C, n_D, n_F = 0$ and $C(q^{-1}) = D(q^{-1}) = F(q^{-1}) = 1$. Then, the identifiability condition simplifies to Eq. (2.14), which requires that the combined orders of the autoregressive part, the input polynomial, and the delay provide sufficient independent information for parameter recovery. In practical terms, this means that the closed-loop data must contain enough dynamic richness from past inputs and outputs to uniquely identify the ARX parameters. Only when this identifiability condition is satisfied and enough data is available, can consistent estimates be expected from routine closed-loop operating data.

$$\max(n_Y + n_k - n_A,\, n_Y - n_B) \;\geq\; \min(n_Y,\, n_A + n_Y,\, n_B + n_X), \tag{2.14}$$

In conclusion, system identification relies on choosing appropriate model structures and estimation methods suited to the available data and system configuration. Residual analysis completes the process by verifying the model's adequacy, which is crucial before deploying it for control, monitoring, or diagnostics.

Section 2.2: FOPDT Models, PID Control, ZOH Discretization and sampling time

FOPDT models are widely used to describe industrial processes because they can capture the dynamic behaviour a first order system. They are characterized by three parameters: the steady-state gain K, the time constant τ_P, and the input delay L. The continuous-time transfer function is given by Eq. (2.15), where K represents the steady-state input–output relation, τ_P defines the process response speed, and L models transport or measurement delays. In many real thermal or chemical processes, these parameters can have a clear physical interpretation. For example, τ_P can be linked to heat transfer or mixing dynamics, while L reflects propagation or sensor lag.

$$G(s) = \frac{K}{\tau_P s + 1} e^{-Ls}, \tag{2.15}$$

To control such processes in closed-loop, Proportional–Integral–Derivative (PID) controllers are widely applied. This controller is expressed in parallel form with a first-order derivative filter as shown in Eq. (2.16), where K_p is the proportional gain, T_i the integral time, T_d the derivative time, and N_f the filter coefficient (Bernhardsson & Åström, 2016; Y. A. Shardt, 2024). The derivative filter is crucial in practice, since it reduces the amplification of high-frequency noise while maintaining the anticipatory effect of derivative action.

$$G_c(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N_f} s + 1} \right), \tag{2.16}$$

Another common controller used in industry is the Proportional–Integral (PI) controller, which removes the derivative term but keeps the integral action to eliminate steady-state error. Its transfer function is described in Eq. (2.17). These controllers are particularly popular in industry because they offer a good balance between simplicity, robustness, and performance, especially for processes that are not strongly affected by fast dynamics or high-frequency noise.

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s}\right). \tag{2.17}$$

Because digital controllers work with sampled data, continuous-time models must be converted into discrete-time form. Several discretization methods are available, each with its own advantages and limitations (Franklin, Powell, & Emami-Naeini, 2002; Ogata, 2010). Among these, the Zero-Order Hold (ZOH) method is particularly important for industrial applications, since it assumes that the input signal remains constant during each sampling interval T_s. This assumption matches the behaviour of real sample-and-hold devices and preserves system stability. In contrast, Tustin's bilinear transform typically offers better frequency approximation but introduces frequency warping. Figure 2.1 illustrates the difference between ZOH and Tustin discretization for a FOPDT step response. Due to its simplicity and widespread use in process control, this thesis uses the ZOH method.

When choosing an appropriate sampling time for discretization, it is important to consider the Nyquist sampling theorem. The theorem says that the sampling frequency f_{sampling} must be at least twice as large as the highest frequency $f_{\max}$ present in the signal in order to capture the main information. Its mathematical formula is shown as follows:

$$f_{\text{sampling}} \geq 2\, f_{\max}$$

The Nyquist frequency is half of the sampling frequency and corresponds to the highest frequency that can be correctly reconstructed from sampled data. It means that for frequencies above this value, the signal cannot be recovered. Its formula is equal to

$$f_{\text{Nyquist}} = 0.5\, f_{\text{sampling}}$$

In summary, if the sampling rate is too low, important information of the signal can be lost, producing distortions known as aliasing. In essence, the Nyquist theorem ensures that discretization does not discard important information from the original continuous signal (Y. A. Shardt, 2022).

Because of this, the choice of sampling time τ_s has a high impact in the quality of the resulting model. Sampling too quickly causes the estimated discrete-time parameters to approach -1, which makes the system harder to identify reliably. On the other hand, sampling too slowly can lead to aliasing. For practical system identification, it is recommended to select $T_s \in [0.1\,\tau_{\min},\, 0.2\,\tau_{\min}]$ (Y. A. Shardt, 2022) or $T_s = \tau_{\min}/3$ (Y. A. Shardt & Huang, 2011b). These recommendations gives a balance between preserving relevant process dynamics and avoiding numerical issues such as "poles near -1".

With ZOH discretization, the continuous-time FOPDT model becomes as shown in Eq. (2.18), where the discrete parameters are indicated in Eq. (2.19).

$$G_p(z) = \frac{b_1\, z^{-\hat{n}_k}}{1 + a_1 z^{-1}}, \tag{2.18}$$

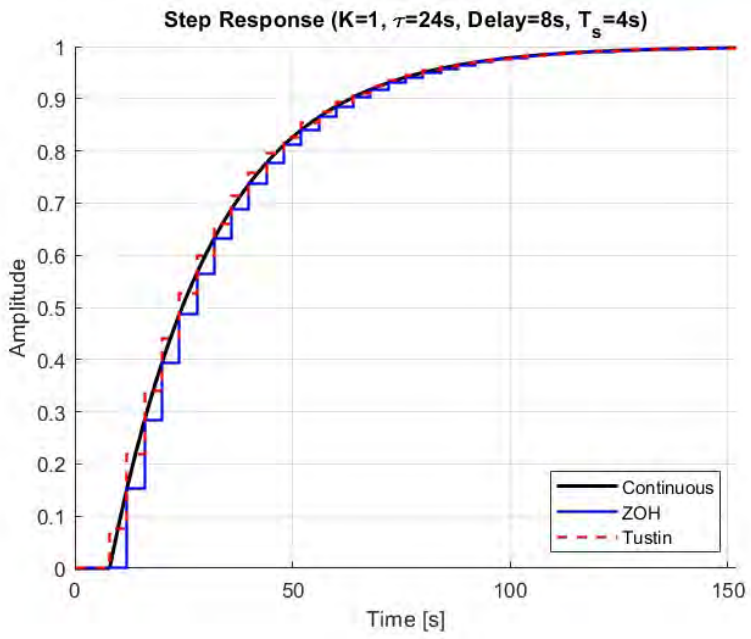


Figure 2.1: Comparison of ZOH and Tustin discretization for a FOPDT step response.

$$a_1 = -e^{-T_s/\tau_P}, \qquad b_1 = K\left(1 - e^{-T_s/\tau_P}\right) = K(1+a_1), \qquad \hat{n}_k = 1 + n_k. \tag{2.19}$$

Here, $n_k = \lceil L/T_s \rceil$ denotes the process delay in samples, and the additional "1" accounts for the one-sample delay introduced by ZOH. Then, the discrete transfer function of the first order plant model is defined as shown in Eq. (2.20). This representation corresponds to an ARX$(1,1,n_k)$ model, establishing a direct link between the physical parameters of FOPDT models and the coefficients of ARX models used in system identification. The original process parameters can be recovered as shown in Eq. (2.21).

$$G_p(z) = \frac{b_1 z^{-(n_k+1)}}{1 + a_1 z^{-1}}. \tag{2.20}$$

$$K = \frac{b_1}{1+a_1}, \qquad \tau_P = -\frac{T_s}{\ln(-a_1)}, \qquad L = n_k T_s \tag{2.21}$$

Section 2.3: Routine-Operating Data

Routine-operating data refers to closed-loop process data collected under normal industrial operation, when no deliberate external excitation is applied. In this case, the reference signal remains constant (sometimes even fixed at zero), and variability arises only from disturbances (Y. Shardt, 2012). Such data are common in practice, since operators avoid injecting test signals that would disrupt production . They are also readily available from process historians or distributed control systems (DCS), offering

practical advantages: they are nonintrusive, requiring no process interruptions, and cost-effective, since they are already passively recorded.

However, despite these advantages, this kind of data come with significant challenges for system identification. In closed-loop operation, the controller continuously adjusts the manipulated variable (control input) to keep the process output close to its reference value. This means that the manipulated variable becomes highly correlated with both the disturbance and the process output (Y. A. Shardt, 2022), due to the feedback and the fact that the disturbance is the only excitation signal for the system. In addition, routine-operating data are often downsampled for storage and can also be affected by measurement noise. These factors make it more difficult to study the true dynamics of the process, especially in fast processes or systems with short delays (Y. A. Shardt & Huang, 2011a, 2011b).

Identifiability with closed-loop and routine operating data case is still possible, but under specific conditions involving process delay, sampling rate, and model order (Y. A. Shardt & Huang, 2011a, 2011b; Y. A. Shardt, Huang, & Ding, 2015) as mentioned in Section 2.1. With short data records, however, the lack of persistent excitation typically leads to biased parameter estimates, as the information content is insufficient to guarantee consistency.

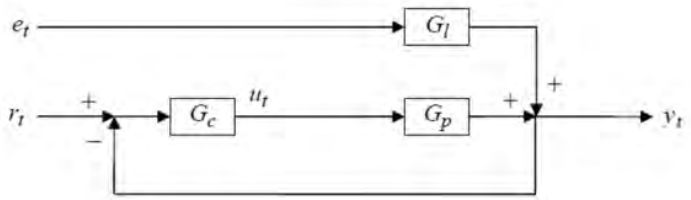


Figure 2.2: General closed-loop block diagram.

A graphical representation of a typical closed-loop configuration is shown in Fig. 2.2. The overall closed-loop relation in the Laplace domain is

$$Y(s) = \underbrace{\frac{G_c(s)G_p(s)}{1+G_c(s)G_p(s)}}_{T(s)} R(s) + \underbrace{\frac{G_l(s)}{1+G_c(s)G_p(s)}}_{S(s)} E(s), \tag{2.22}$$

where $G_c(s)$ is the controller, $G_p(s)$ is the process (FOPDT), and $G_l(s)$ captures the disturbance path dynamics.

As an illustrative example, consider a heated-tank temperature loop actuated by an electric heater and measured by a temperature sensor. The plant model maps the heater power input $u(t)$ [W] to the tank temperature $y(t)$ [°C], with transfer function defined in Eq. 2.15, where K [°C/W] is the static gain, τ_P [s] is the time constant, and L [s] is the input dead time.

For the disturbance model, the following simplified disturbance path for ARX consistency is used:

$$G_l(s) = \frac{1}{1+\tau_P s}, \tag{2.23}$$

The setpoint $r(t)$ [°C] = 50°C is compared to the measured temperature $y(t)$ [°C] to form the error $e_r(t) = r(t) - y(t)$ [°C]. The controller can be a PID as defined in Eq. 2.16, with K_p [W/°C], T_i [s], T_d [s], and filter factor $N > 0$. We can apply this modeling to other real cases as closed-loop setup for a refrigerated container cooled by a Peltier element or even a compressor, room temperature, and others.

For routine-operation conditions, the setpoint is constant and we can adopt $r(t) \equiv 0$ for simplification of the study. Setting $R(s) = 0$ in Eq. (2.22) yields

$$\boxed{Y(s) = \frac{G_l(s)}{1 + G_c(s)G_p(s)} E(s)} \tag{2.24}$$

From Fig. 2.2 we can obtain also a relation between the controller output U(s), noise innovation E(s) and output Y(s) as follows:

$$Y(s) = G_p(s)\,U(s) + G_l(s)\,E(s). \tag{2.25}$$

In practice, the corresponding discrete-time relation is obtained by substituting with the ZOH-discretized models $G_p(z)$ as in Eq. (2.18) and a discretized $G_l(z)$ into the same structure and obtain Y(z):

$$Y(z) = \underbrace{\frac{b_1 z^{-\hat{n}_k}}{1 + a_1 z^{-1}}}_{\text{process (FOPDT, ZOH)}} U(z) + \underbrace{\frac{1}{1 + a_1 z^{-1}}}_{\text{noise/disturbance model}} E(z), \tag{2.26}$$

$$U(s) = \frac{Y(s) - G_l(s)E(s)}{G_p(s)} \tag{2.27}$$

With Eq. (2.24) in Eq. (2.27) we obtain:

$$\boxed{U(s) = -\frac{G_c(s)\,G_l(s)}{1 + G_c(s)G_p(s)} E(s)} \tag{2.28}$$

Section 2.4: Asymptotic and Nonasymptotic Methods

Classical approaches to system identification rely on asymptotic theory. These methods assume that as the number of data points $N \to \infty$, parameter estimates obtained by maximum-likelihood or prediction-error methods converge in probability to the true system values (consistency) (Y. A. Shardt, 2022).

In a Monte Carlo simulation, the experiment is performed N_{exp} times using different noise seeds to assure randomness. For each repetition, the parameters of interest are estimated based on the data

generated. From the collection of N_{exp} estimates, mean value can be calculated and a confidence interval can be obtained using the following analytical procedure:

$$\bar{\alpha} \pm t_{1-\frac{\beta}{2},N_{exp}-1}\sqrt{\frac{\mathrm{var}(\bar{\alpha})}{N_{exp}}} \tag{2.29}$$

where $\bar{\alpha}$ is the mean value of the estimated variable α, β is the confidence value, which will be taken as 0.05, and $\mathrm{var}(\bar{\alpha})$ is the variance of the variable.

$$\frac{(N_{exp}-1)\hat{\sigma}_\alpha^2}{\chi^2_{1-\frac{\beta}{2},N_{exp}-1}} \leq \sigma_\alpha^2 \leq \frac{(N_{exp}-1)\hat{\sigma}_\alpha^2}{\chi^2_{\frac{\beta}{2},N_{exp}-1}} \tag{2.30}$$

where $\chi^2_{z,N_{exp}-1}$ is the χ^2-test with $N-1$ degrees of freedom, $\hat{\sigma}_\alpha$ is the estimated standard deviation of the parameter, and σ_α is the true standard deviation of the parameter. (Y. Shardt, 2012). In general, it is desired that as $N_{exp} \to \infty$, the estimated parameter values approach the theoretical or true parameter values, and the standard deviation approaches zero.

In the case of an asymptotic confidence region, the parameter-estimation error is often modeled as a multivariate normal distribution, so that the regions take the form of ellipsoids centered on the estimated parameter vector:

$$\mathscr{E}_\alpha = \left\{\theta \in \mathbb{R}^p : (\theta-\hat{\theta})^\top P^{-1}(\theta-\hat{\theta}) \leq \chi^2_{p,\alpha}\right\}, \tag{2.31}$$

where $\hat{\theta}$ is the estimated parameter vector, P is an estimate of the asymptotic covariance matrix, and $\chi^2_{p,\alpha}$ is the upper α-quantile of the chi-squared distribution with p degrees of freedom.

However, as emphasized by Csáji and Weyer (2015), such regions are only asymptotic approximations: they provide no guarantees at small sample sizes and may underestimate the true uncertainty. This limitation motivates the development of nonasymptotic methods, such as the SPS approach, which yield exact confidence regions with guaranteed coverage for any finite sample size.

Nonasymptotic methods address this gap by targeting exact, finite-sample guarantees (Csáji et al., 2012). The SPS approach is a leading example for linear regression models that include common identification structures such as FIR or ARX. Consider

$$y_t \;=\; \varphi_t^\top \theta^* + e_t, \qquad t = 1,\ldots,n,$$

with regressors $\varphi_t \in \mathbb{R}^d$, unknown parameter $\theta^* \in \mathbb{R}^d$, and noise innovations e_t that is independent and symmetric about zero. Without assuming a known distribution, SPS constructs a data-driven acceptance test with user-chosen coverage that is exact for any fixed n. At a candidate parameter θ, form the least-squares estimate and residuals

$$\hat{\theta}_n \;=\; \left(\sum_{t=1}^n \varphi_t\varphi_t^\top\right)^{-1}\sum_{t=1}^n \varphi_t y_t, \qquad r_t(\theta) \;=\; y_t - \varphi_t^\top\theta,$$

and the reference sum

$$S_0(\theta) \;=\; \sum_{t=1}^n \varphi_t\, r_t(\theta).$$

Generate $m-1$ sign sequences $\alpha_t^{(i)} \in \{\pm 1\}$ and build perturbed sums

$$S_i(\theta) \;=\; \sum_{t=1}^{n} \alpha_t^{(i)}\, \varphi_t\, r_t(\theta), \qquad i = 1,\dots,m-1.$$

With a chosen norm $\|\cdot\|$ (e.g., Euclidean, or $\|v\|_H = (v^\top H v)^{1/2}$ for some $H \succ 0$), rank the reference statistic among all m values and define

$$R(\theta) \;=\; 1 + \sum_{i=1}^{m-1} \mathbf{1}\big\{\, \|S_i(\theta)\| \le \|S_0(\theta)\| \,\big\}.$$

For an integer $q \in \{1,\dots,m-1\}$, the SPS confidence region is

$$\Theta \;=\; \Big\{\, \theta \in \mathbb{R}^d :\, R(\theta) \le m-q \,\Big\}, \qquad \text{which yields} \qquad \mathbb{P}\big(\theta^* \in \Theta\big) \;=\; 1 - \frac{q}{m}.$$

Two important consequences follow. Since $S_0(\hat{\theta}_n) = \sum_t \varphi_t r_t(\hat{\theta}_n) = 0$, the least-squares estimate $\hat{\theta}_n$ is always included, and the region is star-convex with $\hat{\theta}_n$ as a star center (Csáji et al., 2012; Csáji & Weyer, 2015). Concretely, at a candidate θ one compares $\|S_0(\theta)\|$ with the $m-1$ perturbed norms $\|S_i(\theta)\|$. The candidate is accepted if the reference ranks among the smallest $m-q$, which delivers the exact coverage $1-q/m$ by a uniform-ordering argument implied by the symmetry of the innovations. The asymptotic and SPS boundaries for a small dataset test is shown in Fig. 2.3. Both show similar shapes with the same inclination and contain the LSE and the true parameters. A summary of both asymptotic and nonasymptotic methods can be seen in Table 2.1.

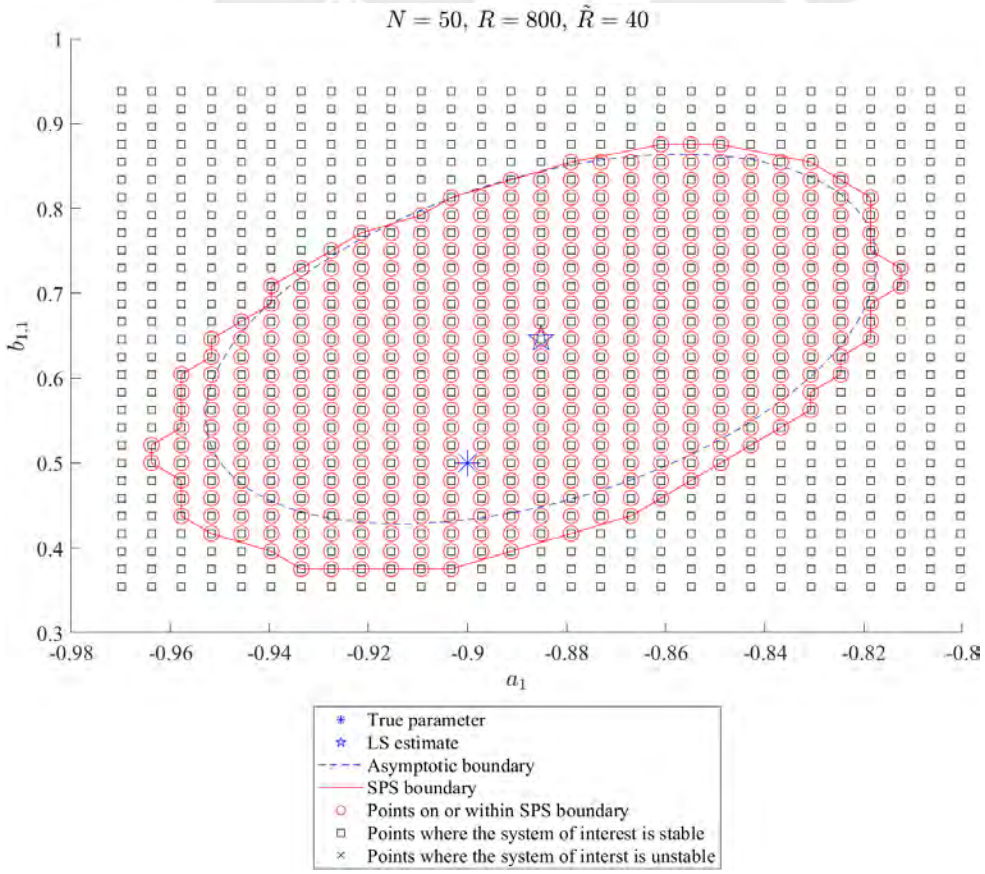


Figure 2.3: Confidence region of a open-loop ARX(1,1) model without delay

Table 2.1: Comparison of asymptotic and nonasymptotic identification methods.

Aspect	**Asymptotic methods**	**Nonasymptotic method (SPS)**
Guarantees	Valid in the limit $N \to \infty$; tight confidence regions	Exact, finite-sample coverage with user-chosen confidence
Assumptions	Often Gaussian (or CLT-based), mild dependence	Independent and symmetric distribution around zero for the noise term
Typical shape of regions	Ellipsoids	Star-convex (contain LS estimate), possibly non-elliptical
Computation	A lot of data processing and parameter estimations; long time	Requires randomized/perturbed statistics.
Data requirements	Perform best with long data records	Designed for short data (finite N)
Robustness to model mismatch	Not good at small N	Coverage holds at finite N under stated assumptions
Use cases	Routine large datasets	Safety-critical, short runs, commissioning tests

Until now we had some assumptions for the noise form according to the asymptotic method (gaussian noise) and nonasymptotic method (independent and symmetric around zero) developed in this work. It must also be mentioned a little about the identification under other type of noise: In real industrial processes, measurement and disturbance noise can often be neither white nor Gaussian. Colored noise introduces temporal correlation, while non-Gaussian noise may exhibit heavy tails or asymmetry, both of which can bias conventional least-squares or maximum-likelihood estimators that rely on Gaussian assumptions. Recent studies have addressed these challenges through robust and data-driven methods. For example, Sun, Zeng, and Yang (2024) studied system identification methods for Lévy (non-Gaussian) noise, while Elamin (2025) developed a particle-filter-based estimation for bilinear systems with colored noise. Similarly, Kanakeri and Mitra (2025) introduced boosting-enabled identification techniques for systems with heavy-tailed disturbances. These contributions extend the classical identification framework and make it more suitable for realistic industrial environments, where noise characteristics can deviate from ideal assumptions. This would be considered for further research.

Section 2.5: Kernel density estimation (KDE)

KDE is a nonparametric technique used for estimating the probability density function (PDF) of a random variable from a finite dataset (Silverman, 2018). In system identification and parameter estimation, KDE is helpful for examining the empirical distribution of parameter estimates obtained from repeated experiments, such as Monte Carlo simulations (Chen, 2017). Unlike traditional methods that rely on strong distributional assumptions, KDE allows a more flexible and data-driven assessment of uncertainty.

The KDE of a parameter's PDF is given by Eq. 2.32, where $\hat{\theta}_i$ are the parameter estimates, $K(\cdot)$ is the kernel function (commonly Gaussian), h is the bandwidth or smoothing parameter, and N is the

number of samples. The choice of bandwidth h is critical: a small h may result in noisy, overfitted estimates, while a large h may oversmooth important features (Chen, 2017; Silverman, 2018).

$$\hat{f}_h(\theta) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{\theta - \hat{\theta}_i}{h}\right) \tag{2.32}$$

Because KDE provides a smooth approximation of the parameter distributions, it is an effective visualization tool. It can highlight the most probable parameter values like mode, median, and empirical confidence intervals, such as central 95% mass regions. Furthermore, KDE allows assessment of estimation bias, skewness, and spread, and can be used to study how parameter uncertainty varies with simulation conditions such as noise variance, data size, or controller configuration. In the case of multivariate parameter estimation, KDE can be extended to two or more dimensions. For example, joint distributions of ARX parameters a_1 and b_1 can be visualized using 2D contour plots or heatmaps based on product kernels or multivariate Gaussian kernels (Chen, 2017).

In MATLAB, KDE can be computed using the built-in functions `ksdensity` for one-dimensional data and `mvksdensity` for multivariate cases (The MathWorks, Inc., 2024a, 2024b). The resulting plots typically show smooth 1D KDE curves with shaded confidence intervals and markers for statistics such as the median or mode. For the 2D KDE analyses, `mvksdensity` produces KDE contour plots with annotated confidence regions (e.g., 90

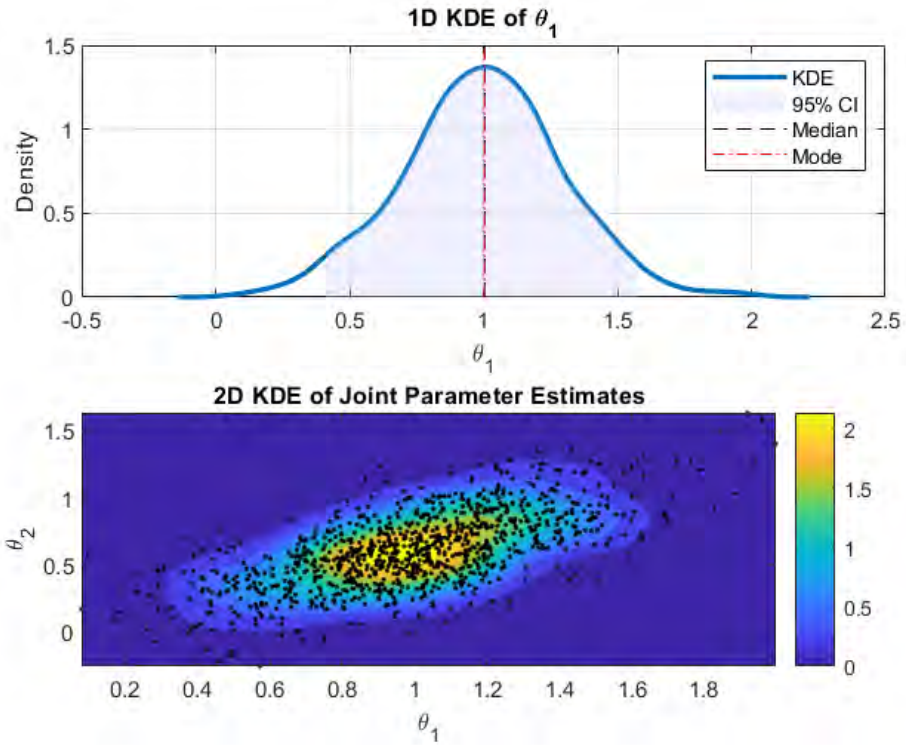


Figure 2.4: KDE in 1D and 2D

Section 2.6: Signal-Noise Ratio (SNR)

This metric describes how strong a useful signal is compared to background noise. In continuous time, it is defined as the ratio between the average power of the input signal $u(t)$ and the average power of the noise $n(t)$ (Schouten, 2010):

$$\mathrm{SNR} = \frac{\mathbb{E}\left[u(t)^2\right]}{\mathbb{E}\left[n(t)^2\right]}, \tag{2.33}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator. With finite data $\{u_t, n_t\}_{t=1}^{T}$, the empirical SNR is estimated by the ratio of sample powers,

$$\widehat{\mathrm{SNR}} \approx \frac{\frac{1}{T}\sum_{t=1}^{T} u_t^2}{\frac{1}{T}\sum_{t=1}^{T} n_t^2} = \frac{\sum_{t=1}^{T} u_t^2}{\sum_{t=1}^{T} n_t^2}, \tag{2.34}$$

where T is the number of data samples. A larger SNR means that the signal is dominant, while a small SNR indicates that the noise is dominating.

For interpretability, it is often convenient to express the SNR in decibels (dB). Since $\widehat{\mathrm{SNR}}$ in (2.34) is a power ratio, the logarithmic conversion is

$$\mathrm{SNR}_{\mathrm{dB}} = 10\log_{10}\left(\widehat{\mathrm{SNR}}\right). \tag{2.35}$$

Equivalently, one may express the ratio in terms of the root-mean-squared (RMS) values of the signal and noise,

$$\mathrm{SNR}_{\mathrm{dB}} = 20\log_{10}\left(\frac{\mathrm{RMS}(u)}{\mathrm{RMS}(n)}\right), \tag{2.36}$$

which is numerically identical because RMS^2 corresponds to average power. A positive value of $\mathrm{SNR}_{\mathrm{dB}}$ indicates that the signal level is stronger than the noise level. This distinction between $10\log_{10}$ for power ratios and $20\log_{10}$ for amplitude ratios is standard in signal processing and control engineering (Maher, 2014; Schouten, 2010).

This definition is consistent with the simulation code, where the empirical SNR is first obtained as a power ratio

$$\widehat{\mathrm{SNR}} = \frac{\mathrm{mean}(U_t^2)}{\mathrm{mean}(N_t^2)}, \tag{2.37}$$

and then converted to dB according to (2.35) or (2.36).

For a numerical example, consider $T = 5$ samples with input $u = [2, -1, 1, 2, 0]$ and additive noise $n = [0.2, -0.1, 0.0, 0.1, -0.2]$. The sample powers are

$$\sum u_t^2 = 2^2 + (-1)^2 + 1^2 + 2^2 + 0^2 = 4 + 1 + 1 + 4 + 0 = 10,$$

$$\sum n_t^2 = 0.2^2 + (-0.1)^2 + 0^2 + 0.1^2 + (-0.2)^2 = 0.10.$$

Using (2.34), the empirical SNR (power ratio) is

$$\widehat{\mathrm{SNR}} = \frac{\sum u_t^2}{\sum n_t^2} = \frac{10}{0.10} = 100.$$

In decibels, from (2.35),

$$\mathrm{SNR}_{\mathrm{dB}} = 10\log_{10}(100) = 10 \times 2 = \mathbf{20\ dB}.$$

Equivalently, if $\mathrm{RMS}(u) = \sqrt{10/5} = \sqrt{2}$ and $\mathrm{RMS}(n) = \sqrt{0.10/5} = \sqrt{0.02}$, then by (2.36)

$$\mathrm{SNR}_{\mathrm{dB}} = 20\log_{10}\left(\frac{\mathrm{RMS}(u)}{\mathrm{RMS}(n)}\right) = 20\log_{10}\left(\sqrt{\frac{2}{0.02}}\right) = 20\log_{10}(10) = 20\ \mathrm{dB}.$$

This example clarifies that the power ratio uses $10\log_{10}(\cdot)$, while the amplitude (RMS) ratio uses $20\log_{10}(\cdot)$, yielding the same dB value.

Chapter 3: Design of the Simulations

Section 3.1: Monte Carlo simulations

The three FOPDT plants shown in Table 3.1 were chosen to evaluate the parameter estimation across different dynamics. Plant 1 is the closed-loop identification case used by (Y. A. Shardt & Huang, 2011b) and serves as a baseline to reproduce comparable results before extending the analysis. Plants 2 and 3 were then selected to be, respectively, slower and faster than the baseline. With respect to the gains, slightly different values from the baseline were chosen. These three plants allow testing the same model ARX$(1,1,n_k)$ structure under different regimes and noise variances (Table 3.3) without changing the basic modeling assumptions summarized in Table 3.2.

Table 3.1: Plants used in the identification study

Plant	**Transfer function**	**Dynamics**	K	τ_P **(s)**	τ_s **(s)**
1	$\frac{1.54}{200s+1}e^{-20s}$	Baseline (Shardt)	1.54	200	20
2	$\frac{2}{300s+1}e^{-40s}$	Slower case	2.00	300	40
3	$\frac{1}{24s+1}e^{-8s}$	Faster case	1.00	24	8

Table 3.2: Sampling time and operating mode

Parameter	**Value**
Sampling time T_s	4 s
Operation: Closed loop, $r(t) = 0$	

Table 3.3: Noise settings

Variance	**Description**
0.5	Low noise
1	Medium noise
5	High noise
10	Higher noise

For Plant 3 (the fastest plant) with $\tau_p = 24$ s, the recommended range for the sampling time is $[2.4, 4.8]$ s. A value of $T_s = 4$ s lies within this range, mitigating aliasing while maintaining identifiability according to the theory described in Sec. (2.2). The discrete pole for Plant 3 is -0.8465. For Plants 1 and 2, the recommended sampling time ranges are $[20, 40]$ s and $[30, 60]$ s respectively. Then, for these two plants, a case of oversampling will be obtained with $T_s = 4$ s. The corresponding discrete poles lie very close to $z = -1$ ($a \approx -0.9802$for plant 1 and $a \approx -0.9868$) for Plant 2). This will make the parameter

estimation more difficult, and this will be interesting to review in the Results section. Then, the sampling time was decided to be fixed at $T_s = 4\,\text{s}$ for all the configurations (Table 3.2).

This choice yields discrete delays ($n_k = 5$, $n_k = 10$ and $n_k = 2$), computed from the continuous dead times as in Eq. (2.11). The value of n_k was checked to ensure that it is favorable for closed-loop identifiability from routine data. The specific delay choices satisfy the identifiability conditions summarized in Chapter 1 (simplified Theorem 1 for ARX shown in Eq. (2.14)) and summarized in Table 3.4.

Table 3.4: Verification of Theorem 1 for each plant and controller combination

Plant	**Controller**	n_k	**Theorem 1**	**Succeed?**
1	PI	5	$n_k \geq 1$	Yes
1	PID	5	$n_k \geq 1$	Yes
2	PI	10	$n_k \geq 1$	Yes
2	PID	10	$n_k \geq 1$	Yes
3	PI	2	$n_k \geq 1$	Yes
3	PID	2	$n_k \geq 1$	Yes

The experiments were carried out in closed-loop, routine operation with a fixed setpoint $r(t) = 0$. The analysis was extended to six controllers (PID1–PID3 and PI1–PI3), as shown in Table 3.5.

Table 3.5: Controller parameters used in the simulations

Label	$\mathbf{K_p}$	$\mathbf{K_i}$	$\mathbf{K_d}$	$\mathbf{T_i}$	$\mathbf{T_d}$	$\mathbf{N_f}$
PID1	1	1/70	1	70	1	10
PID2	1	1/150	1	150	1	10
PID3	2	1/70	10	70	10	10
PI1	1	1/70	0	70	0	–
PI2	1	1/150	0	150	0	–
PI3	2	1/70	0	70	0	–

Combining three plants, six controllers, and four noise levels yields 72 distinct configurations. This produces 132 sliding windows per configuration with capped counts $[20, 20, 20, 20, 20, 20, 11]$ plus the complete dataset (Table 3.6). Each configuration was run for $N = 1000$ Monte Carlo trials, and every sliding window is fit with an ARX$(1, 1, n_k)$ model. This gives 132,000 ARX fits per configuration, for a total of 9,504,000 fits across all 72 configurations. The discrete parameters (a_1, b_1) are then mapped to the continuous interpretable quantities K, τ_P using Eq. 2.21. The summary of the Monte Carlo simulations are given in Table 3.7.

During the tests, some invalid fits were detected and discarded according to rules embedded in the analysis code. The nonfinite or nonreal estimates, the gains outside the range $|K| > 50$, or time constants with values outside $|\tau_P| > 1000$ were discarded. The number of rejected experiments and the reasons were recorded for further analysis. Approved fits were then summarized by the medians of (a_1, b_1), K, and τ_P per configuration.

Table 3.6: Dataset construction per Monte Carlo run

Window size	Max. number of windows, % Overlap	
500	20	10%
1,000	20	10%
2,000	20	10%
5,000	20	10%
10,000	20	80%
20,000	20	80%
50,000	11	90%
100,000	1 (full dataset)	—

Table 3.7: Summary of the configuration of the experiments for the Monte Carlo simulation

Plant	Noise variance	Controller	Data size N	# of sliding windows	# of runs	Parameters
1 2 3	0.5 1 5 10	PI1 PI2 PI3 PID1 PID2 PID3	500 1,000 2,000 5,000 10,000 20,000 50,000 100,000	20 20 20 20 20 20 11 1	1,000 (per config.)	K_{estimate} $\tau_{P,\text{estimate}}$ $a_{1,\text{estimate}}$ $b_{1,\text{estimate}}$
3	**4**	**6**	**8**	**132**	**1,000**	**4**

Section 3.2: Record of one dataset per configuration

For each configuration, an additional base record of 100,000 samples was generated and stored as `full_K*_var_j*_ctrl_PI**_ZOH_nk*_SNR*.mat` files. The main idea is to re-use these data using a capped sliding window with different window sizes $\{500, 1000, 2000, 5000, 10{,}000, 20{,}000, 50{,}000\}$, different overlaps $\{20\%, 50\%, 70\%, 90\%\}$, and limits on the number of sliding windows per size $\{10, 20, 40\}$, plus the full dataset of 100,000 points. The goal is to evaluate whether, with very limited data, we can recover the parameter estimates, and benefit from sliding window ARX fits, to obtain insight into the relation between number of fits and data length.

Section 3.3: Overlay of SPS and asymptotic confidence regions according to data size and noise variance

The 95% nonasymptotic confidence region obtained with the SPS method is computed for the smallest data size and for the smallest and largest noise-variance levels using the main controller as summarized in Table 3.8. These regions are then overlain and compared with the asymptotic 2D KDE for the same configurations. The degree of overlap is quantified. The bias is computed and examined.

Table 3.8: Overlay configurations for SPS and KDE confidence regions (routine operation, $r(t) \equiv 0$)

Plant	**Noise variance**	**Controller**	**Data size N**	**Parameters**
1			500 1,000 2,000	
2	0.5		5,000	$a_{1,\text{estimate}}$
3	10	PID1	10,000	$b_{1,\text{estimate}}$
3	**2**	**2**	**4**	**4**

Section 3.3.1: Computational complexity and execution time

The flow diagram of the simulation and analysis process can be observed in Fig 3.1. The total computational effort grows linearly with the number of configurations and Monte Carlo runs and quadratically with the number of samples per dataset when computing confidence regions on grids. The LSE in each ARX fit has complexity according to the number of parameters, but this cost has not big impact compared with the number of runs. Hence, the dominant cost is determined by $N_{\text{conf}} = 72$ (number of configurations), $N_{\text{MC}} = 1000$ (number of Monte Carlo trials), and $N_{\text{win}} = 132$ (number of sliding windows applied).

The full simulation required approximately 9.5 million ARX fits. When executed on the university HPC cluster using 32 cores in parallel, the total runtime was about 3 days, including data generation, model estimation, and post-processing. The SPS region computation was performed only for selected cases and took about 2-15 hours in my personal computer, depending on data size.

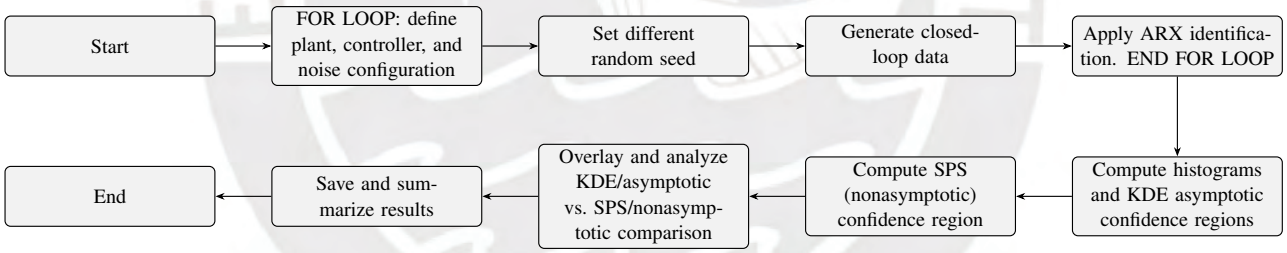


Figure 3.1: Flow diagram of the simulation and analysis process.

Chapter 4: Results

Section 4.1: Monte Carlo Simulation Results

The program `FOR_LOOP_V27_HPC.m`, which is detailed in Sec. A.1.1, realizes a Monte Carlo simulation that produced millions of estimated ARX parameters across different configurations. The data used for closed-loop system identification were the control signal u and the output signal y. The only excitation introduced to this system was the noise innovation e_t as shown in Fig. 2.2. These signals are shown in Fig. 4.1. As expected, the output y varies around zero.

Some experiments were rejected because their estimated parameters exceeded the limits $|K| > 50$ and $|\tau_P| > 1000$, as reported in Y. A. Shardt and Huang (2011b). The total rejection summary per plant is given in Table 4.1. All experiments for Plant 3 completed without rejections. In contrast Plant 1 and Plant 2 had rejection rates of 1.75% and 2.57%, respectively. A global summary of these results is shown in Table 4.2. In total, only 1.44% of the ARX fits were discarded for exceeding the established parameter limits. Additional details on the rejection statistics, such as those grouped by plant and noise variance, are provided in Appendix A.2.

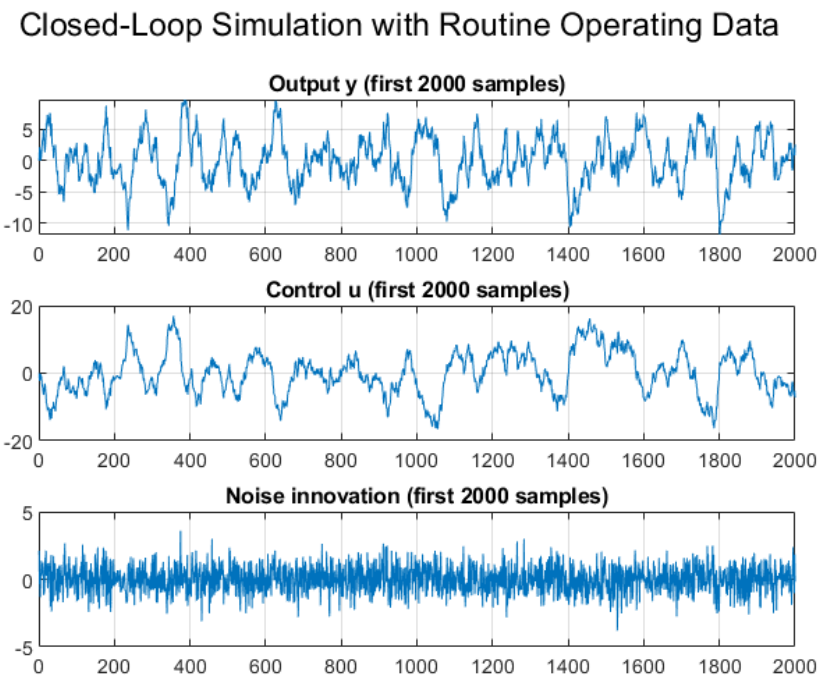


Figure 4.1: Closed-loop system with routine operating data: main signals.

Table 4.1: Totals per plant — all controllers, all noise variances.

Plant	Total Fits	Total Failed	Accepted Fits	% Discarded
$K = 1.54$	3,168,000	55,453	3,112,547	1.75%
$K = 2$	3,168,000	81,560	3,086,440	2.57%
$K = 1$	3,168,000	0	3,168,000	0.00%

Table 4.2: Global totals — all plants, controllers, and noise variances.

Total Fits	Total Failed	Accepted Fits	% Discarded
9,504,000	137,013	9,366,987	1.44%

All resulting data were stored in `.mat` files according to the corresponding configuration. For example, the file `results_K1_var_j1_ctrl_PI1_ZOH.mat` refers to the Monte Carlo experiment data of plant 1, noise variance 0.5, and the controller labeled `PI1`, with different seeds and data sizes. As mentioned in Sec. 3.1, there are 72 `.mat` files in total, corresponding to three plants, four noise variances, and six controllers.

Section 4.1.1: Mean and Median Values vs. Data Size According to Configuration

The program `FOR_LOOP_V27_HPC.m` also stores the mean, median, standard deviation, and detailed information for each configuration and data size in the files `MainSummary.mat` and `MainSummary.xlsx`. From this database, the following tables are created for a better visualization of the data: Tables 4.3, 4.4, 4.5, and 4.6 present the mean values, bias, and standard deviation for the controller PID1 and the three plants, whereas Tables 4.7, 4.8, 4.9, and 4.10 report the corresponding median values and their bias. From these results, it can be observed that the plant having the largest mean bias for the plant parameters is Plant 1. For this configuration, a bias less than or equal to 5% of the true parameter value is achieved when the sample size reaches $N = 5{,}000$. As expected, the standard deviation decreases as the data size used for ARX estimation increases. By definition, the median tends to yield estimates closer to the true parameter values. This can be confirmed by examining the bias values in the median tables, where a bias below 5% is already achieved for $N = 2{,}000$.

The main program also generates 2D and 3D plots of the mean and median of the estimated parameters, grouped by plant and controller, with the noise variance and data size as axes. Examples of the 3D plots for K and τ_P, corresponding to the three plants and controller PID1, are shown in Fig. 4.2 (mean) and Fig. 4.3 (median). The figures of Plants 1 and 2 show that in some cases, a data size larger than 1000 is required to have a more accurate estimation of the parameters, as shown in Fig. 4.4 and Fig. 4.5, respectively. In other hand, the Fig. 4.6 shows that Plant 3 has always accurate parameter estimation even with data size of 500. The 2D plots show not only the parameter estimation, but also the 2% and 5% error bands for better visual analysis of the bias. In this chapter, it is shown mainly the results of PID controller because the PI results are very similar. After examining the tables and all the plots, it can be observed that the noise variance did not show a significant impact on the parameter estimation results with the median.

As shown in these plots, a data length of $N = 500$ is clearly insufficient for reliable parameter estimation due to the bias observed in the results. In general, especially for Plant 1 and Plant 2. a data size of at least $N = 2{,}000$ points is recommended to have a more accurate estimation of the plant parameters.

Note: The complete set of 3D plots of the mean and median values (noise variance vs. data size) for all configurations can be found in the directory "`Thesis_2025_Final_Results_JADS/2. FOR_LOOP_Results/Plots_Mean_vs_N_fromResults`". The corresponding 2D plots of the mean and median values are available in "`Thesis_2025_Final_Results_JADS/2. FOR_LOOP_Results`" and "texttt`Thesis_2025_Final_Results_JADS/2. FOR_LOOP_Results/Plots_Median_vs_N_fromResults`", respectively.

Table 4.3: Summary metrics (mean) for **Controller = PID1_ZOH**, **N = 500**

ν	$\overline{K}$	**Bias** $_{\overline{K}}$	**std**(K)	$\overline{\tau_P}$	**Bias** $_{\overline{\tau_P}}$	**std**(τ_P)
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$						
0.5	1.8742	21.70%	1.6608	226.5639	13.28%	169.8639
1	1.8903	22.75%	1.6743	227.8564	13.93%	170.8991
5	1.8630	20.97%	1.6376	225.4912	12.75%	167.7776
10	1.8677	21.28%	1.6434	226.1786	13.09%	168.2779
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$						
0.5	2.1134	5.67%	1.3310	305.4970	1.83%	180.0678
1	2.1162	5.81%	1.3587	306.8178	2.27%	182.6417
5	2.1352	5.50%	1.3350	305.5002	1.83%	179.4815
10	2.1130	5.65%	1.3509	306.0079	2.00%	181.4375
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$						
0.5	1.0487	4.87%	0.3494	24.6596	2.75%	5.8061
1	1.0466	4.66%	0.3453	24.6443	2.68%	5.6854
5	1.0473	4.73%	0.3448	24.6555	2.73%	5.7076
10	1.0500	5.00%	0.3490	24.7178	2.99%	5.7584

Table 4.4: Summary metrics (mean) for **Controller = PID1_ZOH**, **N = 1000**

ν	$\overline{K}$	**Bias** $_{\overline{K}}$	**std**(K)	$\overline{\tau_P}$	**Bias** $_{\overline{\tau_P}}$	**std**(τ_P)
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$						
0.5	1.8362	19.24%	1.1780	227.8099	13.90%	126.0485
1	1.8438	19.73%	1.1812	228.1920	14.10%	125.9763
5	1.8317	18.94%	1.1856	226.7118	13.36%	125.2308
10	1.8336	19.06%	1.1840	227.0521	13.53%	125.8616
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$						
0.5	2.1595	7.98%	1.0999	317.4549	5.82%	152.4862
1	2.1525	7.62%	1.1032	316.5360	5.51%	152.2862
5	2.1462	7.31%	1.0950	315.7876	5.26%	150.3283
10	2.1625	8.12%	1.1044	317.7246	5.91%	151.1178
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$						
0.5	1.0213	2.13%	0.2320	24.2683	1.12%	3.8370
1	1.0226	2.26%	0.2321	24.3200	1.33%	3.8225
5	1.0248	2.48%	0.2331	24.3451	1.44%	3.8667
10	1.0240	2.40%	0.2312	24.3377	1.41%	3.8079

Table 4.5: Summary metrics (mean) for **Controller = PID1_ZOH**, **N = 2000**

ν	$\overline{K}$	**Bias** $_{\overline{K}}$	**std**(K)	$\overline{\tau_P}$	**Bias** $_{\overline{\tau_P}}$	**std**(τ_P)
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$						
0.5	1.7085	10.94%	0.7684	216.0385	8.02%	83.0902
1	1.7125	11.20%	0.7804	216.2589	8.13%	84.1414
5	1.6973	10.22%	0.7594	215.0427	7.52%	82.1659
10	1.6984	10.28%	0.7674	214.5268	7.26%	82.8088
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$						
0.5	2.1203	6.02%	0.8065	314.0187	4.67%	112.1744
1	2.1039	5.20%	0.7888	311.9002	3.97%	109.4009
5	2.0996	4.98%	0.8001	311.9478	3.98%	111.6892
10	2.1143	5.71%	0.7979	313.2841	4.43%	110.1930
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$						
0.5	1.0105	1.05%	0.1586	24.1337	0.56%	2.6358
1	1.0112	1.01%	0.1595	24.1388	0.58%	2.6309
5	1.0111	1.18%	0.1601	24.1619	0.67%	2.6429
10	1.0112	1.12%	0.1611	24.1836	0.76%	2.6649

Table 4.6: Summary metrics (mean) for **Controller = PID1_ZOH**, **N = 5000**

ν	$\overline{K}$	**Bias** $_{\overline{K}}$	**std**(K)	$\overline{\tau_P}$	**Bias** $_{\overline{\tau_P}}$	**std**(τ_P)
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$						
0.5	1.6014	3.99%	0.4046	205.6820	2.84%	43.9627
1	1.6041	4.16%	0.4134	206.0732	3.04%	45.0662
5	1.5983	3.78%	0.4037	205.4250	2.71%	43.9390
10	1.6002	3.91%	0.4076	205.4446	2.72%	44.4179
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$						
0.5	2.0461	2.30%	0.4703	305.6090	1.87%	65.6510
1	2.0453	2.26%	0.4679	305.2482	1.75%	65.2963
5	2.0442	2.21%	0.4698	305.3494	1.78%	65.1868
10	2.0470	2.35%	0.4709	305.3757	1.79%	65.2277
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$						
0.5	1.0035	0.35%	0.0992	24.0412	0.17%	1.6483
1	1.0037	0.37%	0.1002	24.0533	0.22%	1.6446
5	1.0048	0.48%	0.0993	24.0697	0.29%	1.6397
10	1.0048	0.48%	0.0996	24.0816	0.34%	1.6439

Table 4.7: Summary metrics (median) for **Controller = PID1_ZOH**, **N = 500**

ν	$\tilde{K}$	**Bias**$_{\tilde{K}}$	$\tilde{\tau}_P$	**Bias**$_{\tilde{\tau}_P}$
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$				
0.5	1.4289	7.21%	180.3361	10.90%
1	1.4325	6.98%	181.3011	10.31%
5	1.4260	7.40%	180.8076	10.61%
10	1.4231	7.59%	179.6969	11.30%
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$				
0.5	1.7523	12.39%	254.5828	17.84%
1	1.7431	12.84%	255.6698	17.34%
5	1.7485	12.57%	255.9848	17.19%
10	1.7389	13.05%	254.1550	18.04%
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$				
0.5	0.9939	0.61%	23.6994	1.27%
1	0.9951	0.49%	23.7333	1.12%
5	0.9944	0.56%	23.7247	1.16%
10	0.9968	0.32%	23.8518	0.62%

Table 4.8: Summary metrics (median) for **Controller = PID1_ZOH**, **N = 1000**

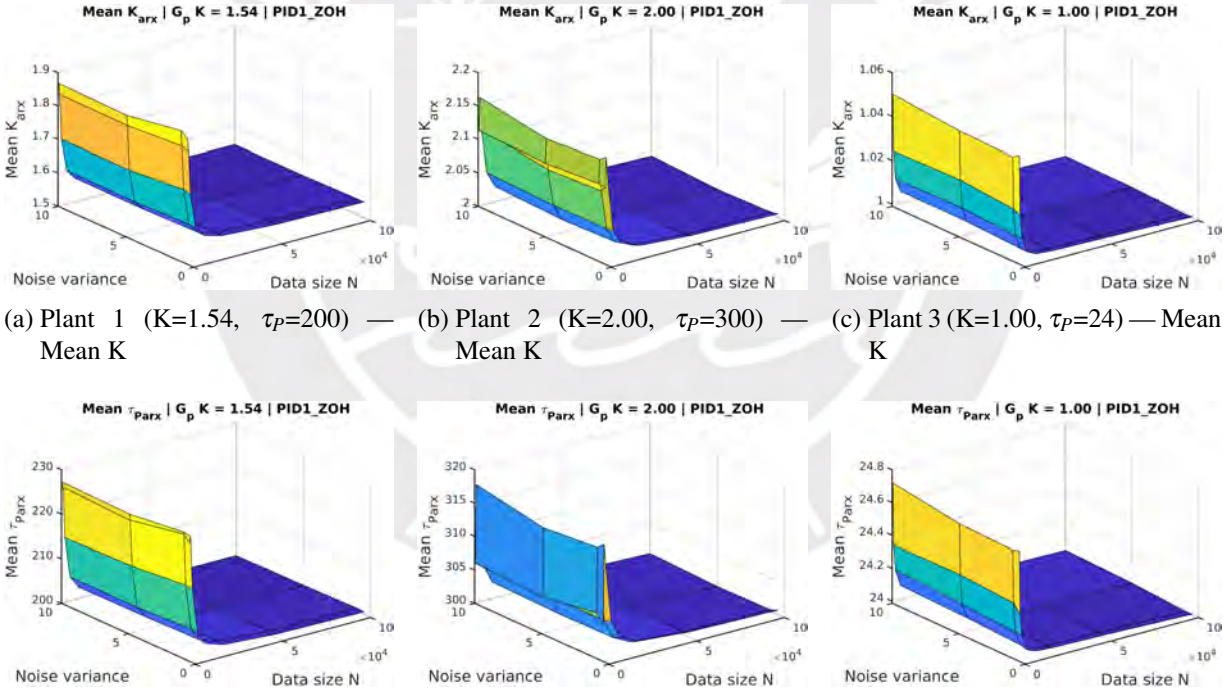
ν	$\tilde{K}$	**Bias**$_{\tilde{K}}$	$\tilde{\tau}_P$	**Bias**$_{\tilde{\tau}_P}$
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$				
0.5	1.5045	2.31%	192.7905	3.74%
1	1.5219	1.17%	194.6546	2.75%
5	1.5171	1.49%	192.8786	3.69%
10	1.5026	2.43%	192.1733	4.07%
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$				
0.5	1.8869	5.66%	279.7228	7.25%
1	1.8824	5.88%	279.8796	7.37%
5	1.8776	6.12%	277.3339	8.17%
10	1.8868	5.66%	280.1160	7.10%
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$				
0.5	0.9950	0.50%	23.8229	0.74%
1	0.9959	0.41%	23.8671	0.56%
5	0.9984	0.16%	23.8670	0.56%
10	0.9992	0.08%	23.8748	0.52%

Table 4.9: Summary metrics (median) for **Controller = PID1_ZOH**, **N = 2000**

ν	$\tilde{K}$	**Bias**$_{\tilde{K}}$	$\tilde{\tau}_P$	**Bias**$_{\tilde{\tau}_P}$
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$				
0.5	1.5375	0.17%	198.1865	0.92%
1	1.5353	0.30%	197.0743	1.48%
5	1.5228	1.12%	196.6924	1.68%
10	1.5296	0.68%	196.4247	1.82%
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$				
0.5	1.9573	2.13%	290.7363	3.30%
1	1.9449	2.76%	289.9024	3.48%
5	1.9374	3.13%	288.7847	3.88%
10	1.9543	2.29%	290.5726	3.24%
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$				
0.5	0.9969	0.31%	23.9250	0.31%
1	0.9971	0.29%	23.9320	0.28%
5	0.9985	0.15%	23.9462	0.22%
10	0.9994	0.06%	23.9754	0.10%

Table 4.10: Summary metrics (median) for **Controller = PID1_ZOH**, **N = 5000**

ν	$\tilde{K}$	**Bias**$_{\tilde{K}}$	$\tilde{\tau}_P$	**Bias**$_{\tilde{\tau}_P}$
Plant 1 (baseline): $K = 1.54$, $\tau_P = 200$, $\tau_d = 20$				
0.5	1.5378	0.14%	199.0102	0.50%
1	1.5403	-0.02%	198.9439	0.53%
5	1.5363	0.24%	198.8264	0.59%
10	1.5364	0.23%	198.5846	0.71%
Plant 2 (slow): $K = 2$, $\tau_P = 300$, $\tau_d = 40$				
0.5	1.9847	0.76%	296.4285	1.20%
1	1.9823	0.89%	296.6103	1.14%
5	1.9773	1.14%	296.3613	1.23%
10	1.9843	0.79%	296.6359	1.13%
Plant 3 (fast): $K = 1$, $\tau_P = 24$, $\tau_d = 8$				
0.5	0.9981	0.19%	23.9643	0.15%
1	0.9992	0.08%	23.9697	0.13%
5	1.0003	-0.03%	23.9748	0.10%
10	1.0001	0.00%	23.9971	0.01%



(a) Plant 1 (K=1.54, τ_P=200) — Mean K

(b) Plant 2 (K=2.00, τ_P=300) — Mean K

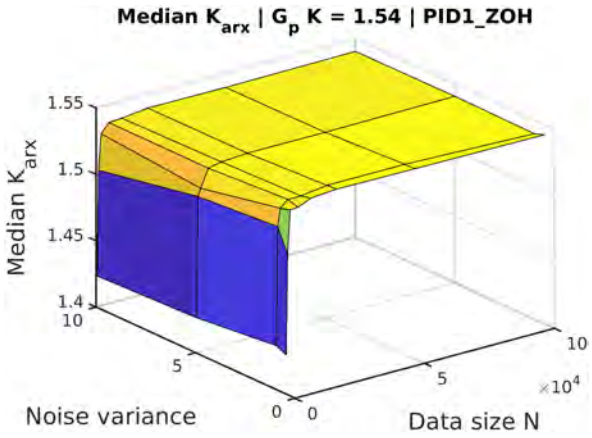
(c) Plant 3 (K=1.00, τ_P=24) — Mean K

(d) Plant 1 (K=1.54, τ_P=200) — Mean τ_P

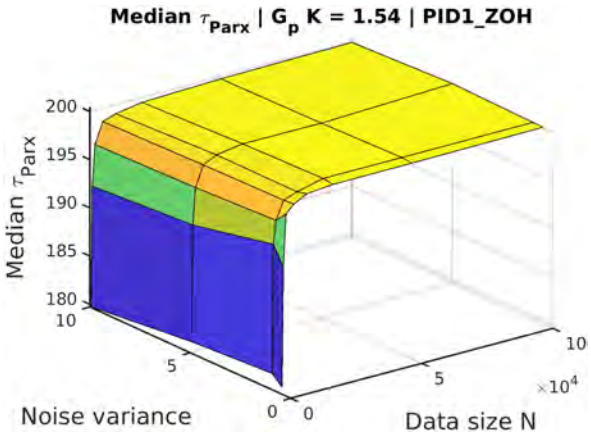
(e) Plant 2 (K=2.00, τ_P=300) — Mean τ_P

(f) Plant 3 (K=1.00, τ_P=24) — Mean τ_P

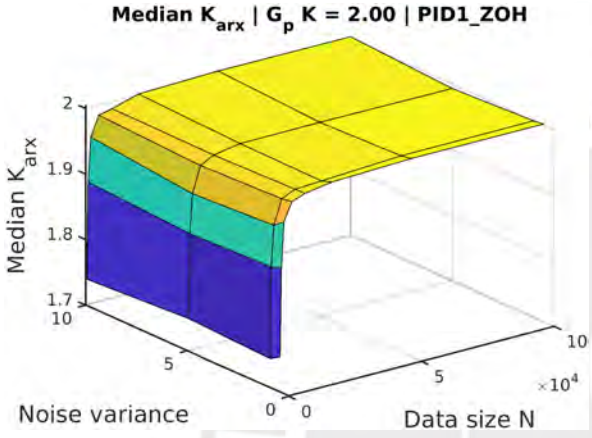
Figure 4.2: Mean K (top row) and Mean τ_P (bottom row) for PID1 across three plants.



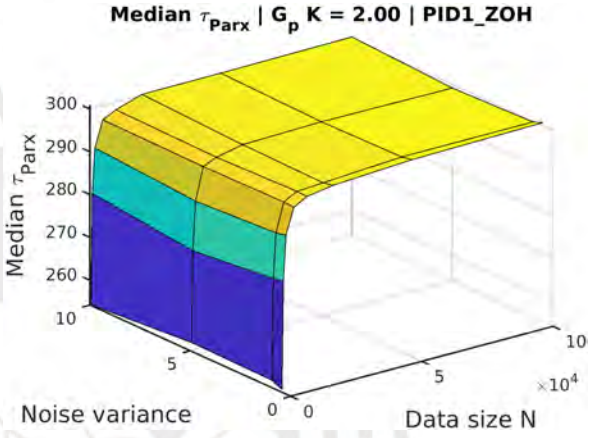
(a) Plant 1 (K=1.54, τ_P=200) — Median K



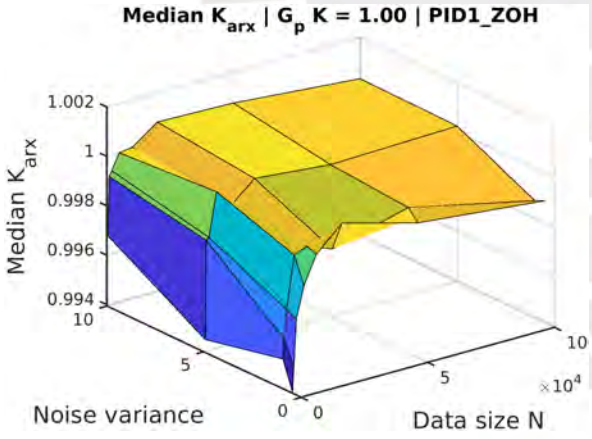
(b) Plant 1 (K=1.54, τ_P=200) — Median τ_P



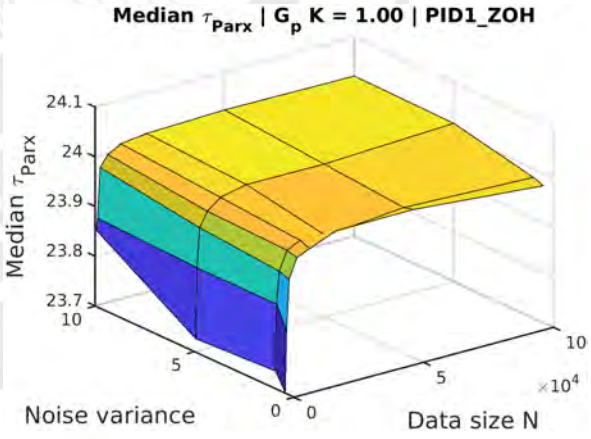
(c) Plant 2 (K=2.00, τ_P=300) — Median K



(d) Plant 2 (K=2.00, τ_P=300) — Median τ_P

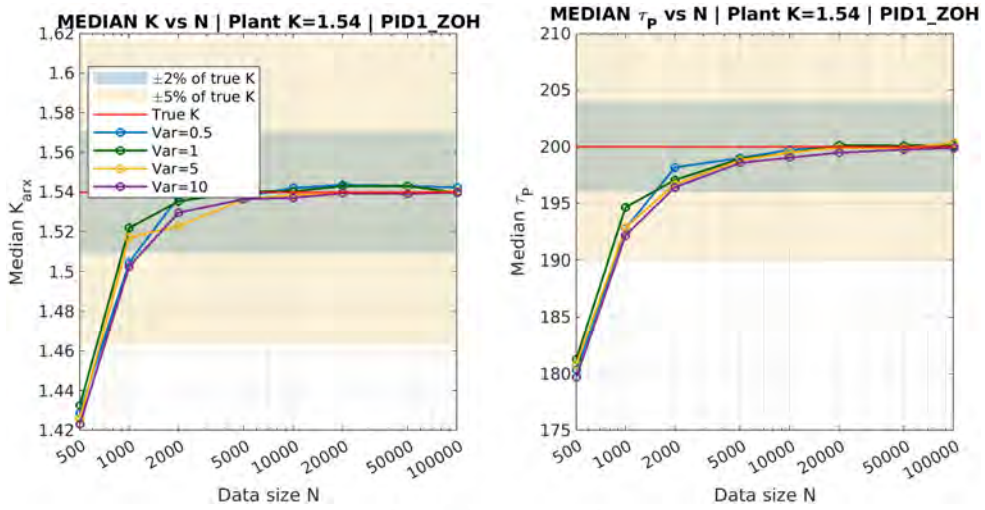


(e) Plant 3 (K=1.00, τ_P=24) — Median K

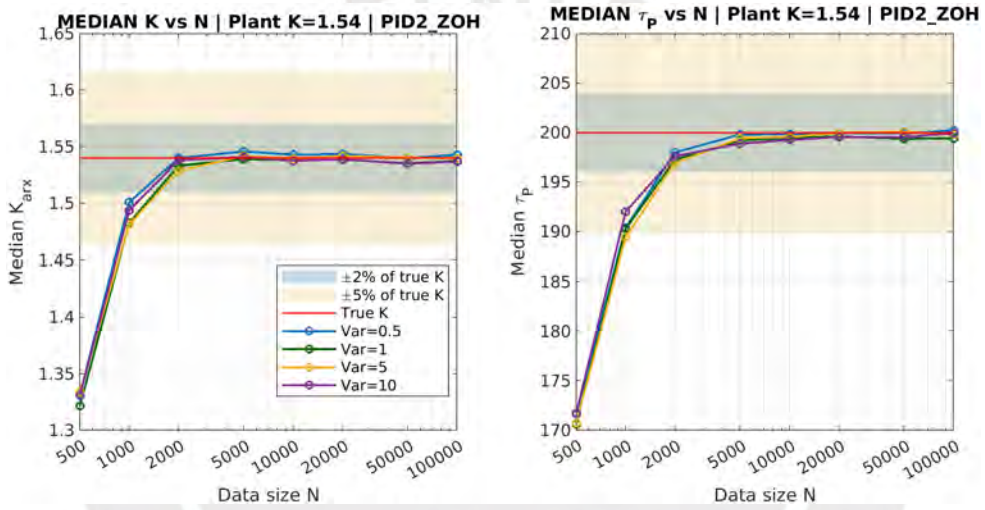


(f) Plant 3 (K=1.00, τ_P=24) — Median τ_P

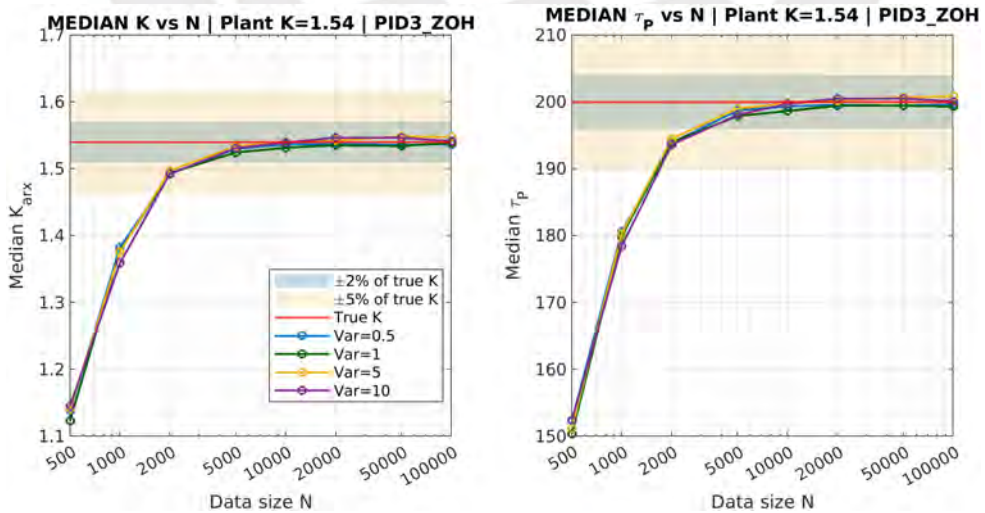
Figure 4.3: Median K (left column) and Median τ_P (right column) for PID1 across three plants.



(a) PID1

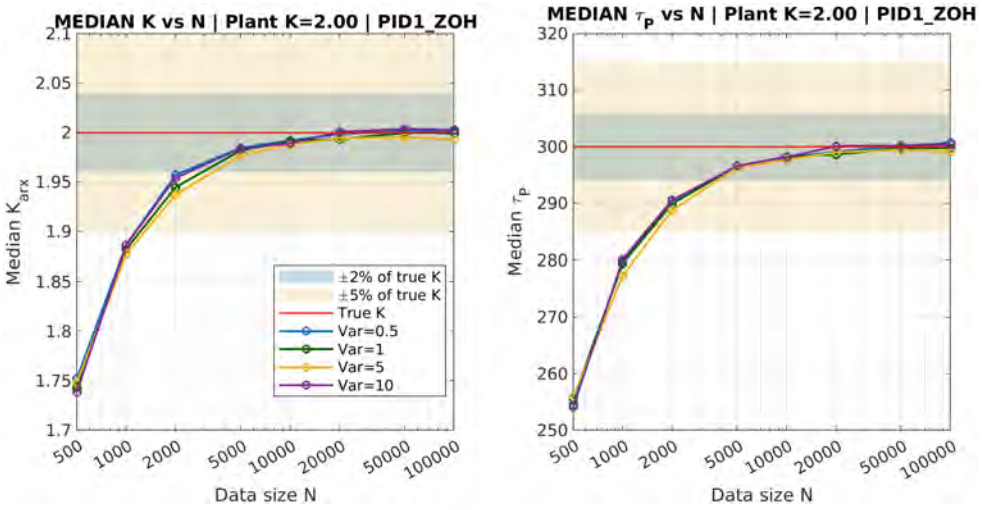


(b) PID2

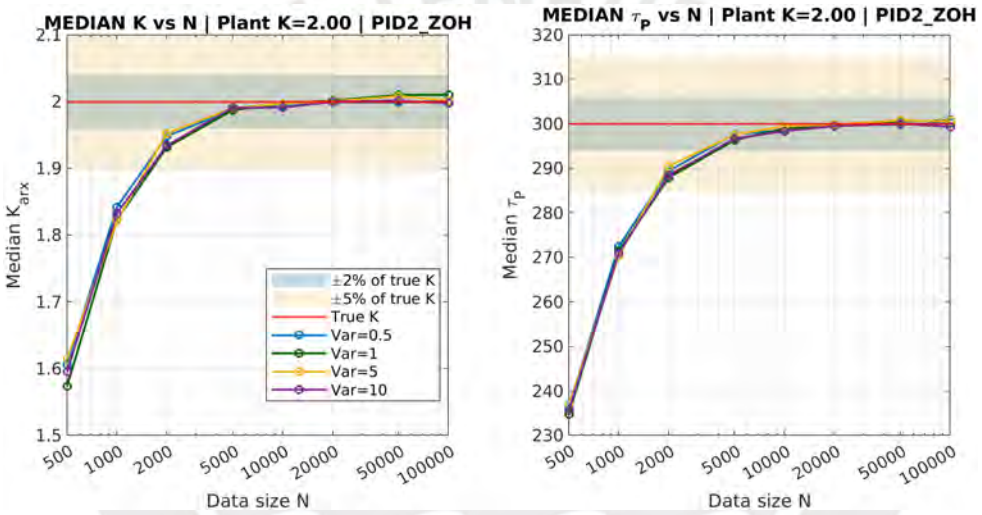


(c) PID3

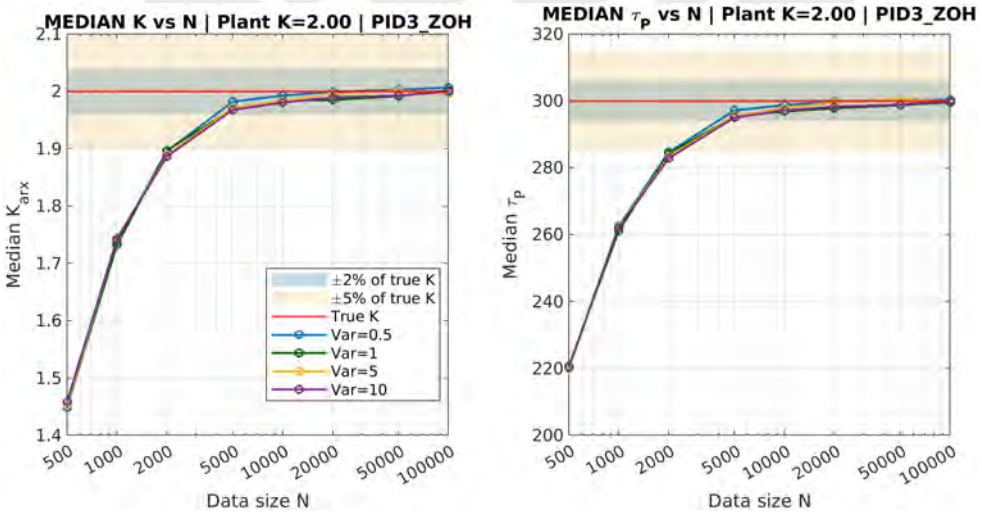
Figure 4.4: Median vs. N (LR) for Plant $K = 1.54$ using PID controllers (ZOH).



(a) PID1

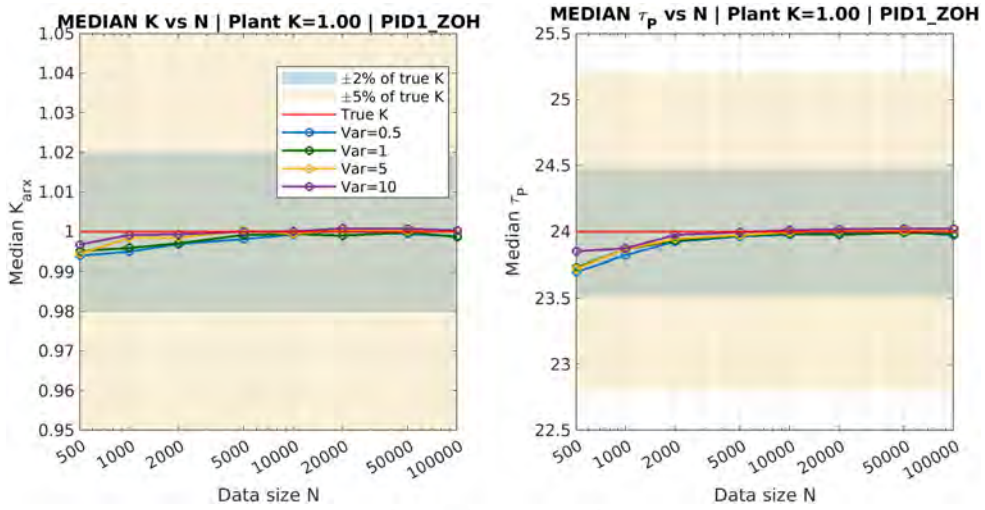


(b) PID2

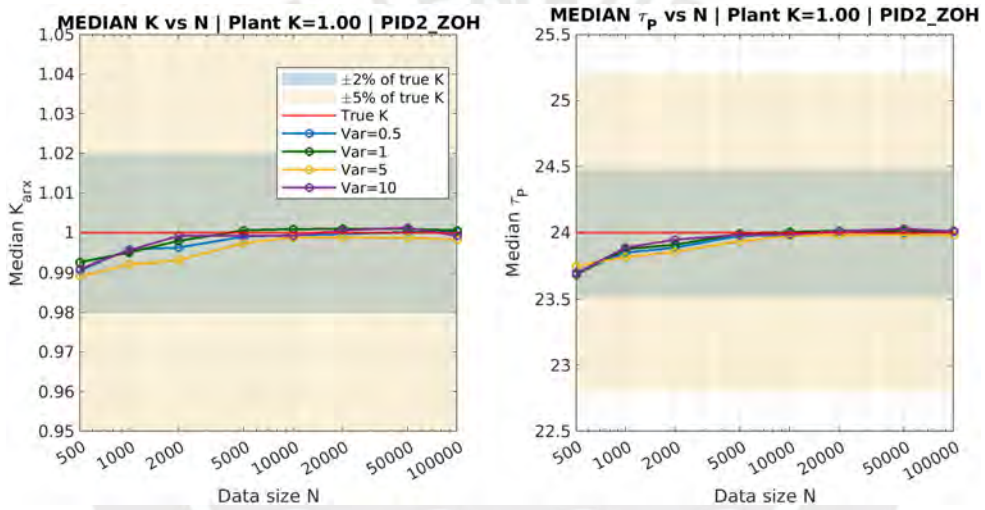


(c) PID3

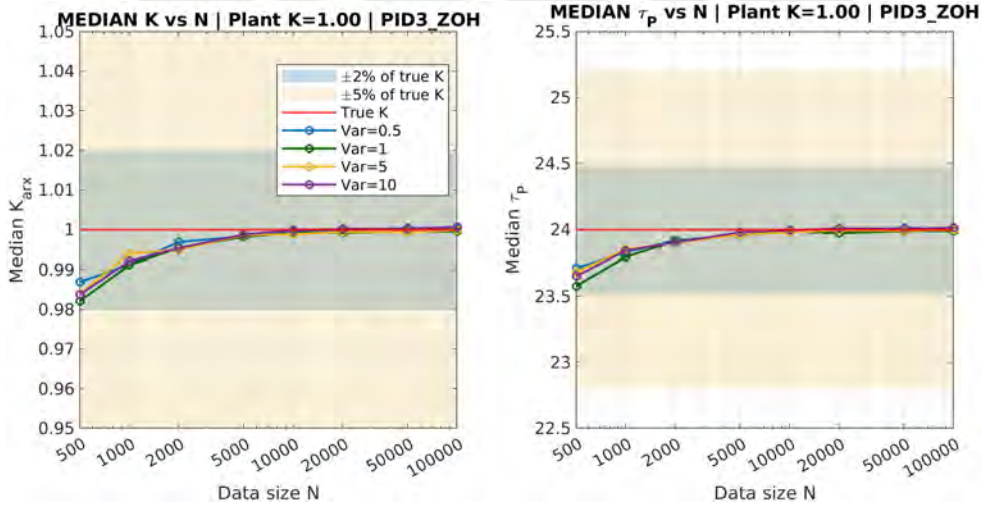
Figure 4.5: Median vs. N (LR) for Plant $K = 2.00$ using PID controllers (ZOH).



(a) PID1



(b) PID2



(c) PID3

Figure 4.6: Median vs. N (LR) for Plant $K = 1.00$ using PID controllers (ZOH).

Section 4.1.2: Histograms

The histogram plots of a_1 & b_1 for Plant 1 with the PID1 controller (Figures 4.7–4.10) show that for small data sizes ($N = 500$ and $N = 1{,}000$), the distributions are slightly skewed and irregular, indicating non-normal behavior with deviations from the fitted Gaussian curve shown in red. A small bias between the true parameter and the sample median can also be observed for these data sizes. As the sample size increases ($N = 2{,}000$ and $N = 5{,}000$), the distributions become smoother, more symmetric, and closely aligned with the normal fit, suggesting convergence toward normality. For $N \geq 10{,}000$, the histograms are nearly perfectly bell-shaped and symmetric, with the sample medians coinciding with the true parameter values, indicating that the estimation errors are approximately Gaussian. At $N = 100{,}000$, the histograms of a_1 and b_1 appear less smooth and slightly irregular compared to smaller data sizes. This behavior does not indicate a loss of normality but rather results from the reduced number of Monte Carlo estimates at this largest data size. Since each experiment uses the entire dataset of 100,000 samples without segmentation, the total number of realizations contributing to the histogram is smaller than for shorter datasets. In these histograms is not seen skewness and the values with more frequency are coincident with the median and true parameters.

In the other hand, the histograms of K_{ARX} and $\tau_{P,ARX}$ are shown in the Figures 4.11–4.14. The small sample size cases ($N = 500$ and $N = 1{,}000$ can be seen in Figure 4.11), where both parameters show strongly right-skewed and dispersed distributions. Particularly K_{ARX}, presents a long right tail. The Gaussian fit does not adequately represent the data in this regime, and the sample medians deviate noticeably from the true plant parameters, indicating estimation bias and non-normality. The spread of $\tau_{P,ARX}$ estimates is also wide, although the skewness is slightly less pronounced than for K_{ARX}. As N increases to 2,000 and 5,000 (Figure 4.12), the distributions become narrower and more symmetric, even less skewness, reflecting reduced estimation variance and improved alignment with the normal fit. For larger data sets ($N = 10{,}000$ and $N = 20{,}000$; Figure 4.13), the histograms exhibit nearly perfect bell shapes, with the red Gaussian curves almost matching the empirical distributions. The second histogram is better centered around the true parameter values and its distribution can be considered approximately normal.

Finally, for $N = 50{,}000$ and $N = 100{,}000$ (Figure 4.14), the histograms are sharply peaked and highly symmetric, with almost complete overlap between the empirical and Gaussian curves. The confidence intervals are very tight, and the medians coincide with the true parameters, confirming the asymptotic consistency of the ARX estimates. Although the histograms for $N = 100{,}000$ appear slightly less smooth, this effect is again due to the reduced number of Monte Carlo experiments for this case compared to those with other data size.

Overall, the results demonstrate a trend of the parameter estimation distributions getting a gaussian form as N increases. This is consistent with the central limit theorem. These observations were also tested by the Lilliefors normality test as done according to (Y. A. Shardt & Huang, 2011b). This test confirmed that the parameter estimation have a more normal distribution as larger the data size, and it can be seen in Table 4.11 that the parameters a1 & b1 had normal distribution even for smaller data sizes, but for the plant parameters of interest K_{ARX} and $\tau_{P,ARX}$ this test passed only for data size larger than $N = 50{,}000$. The details of all the Lilliefor tests done are in the excel file "NORMAL_SUM.xlsx" in the shared folder and also the program "analyze_normal_sum.m" that reads this file and makes the statistics to obtain the shown tables.

Table 4.11: Percentage of Approved Lilliefor Tests by Data Size and Parameter Pair

Data Size	K&τ_P (%)	a_1&b_1 (%)
500	0.0	9.7
1000	0.0	14.6
2000	0.0	25.7
5000	0.0	29.9
10000	0.0	39.6
20000	0.0	50.0
50000	13.2	63.2
100000	68.8	97.9

Table 4.12: Approvals of Lilliefor tests by Plant and Parameter Pair

Plant	Parameter Pair	Total N	Approved N	Approved (%)	True a_1	True b_1
K=1, τ_P=24	K&τ_P	384	67	17.4	−0.8465	0.1535
K=1, τ_P=24	a_1&b_1	384	233	60.7	−0.8465	0.1535
K=1.54, τ_P=200	K&τ_P	384	25	6.5	−0.9802	0.0305
K=1.54, τ_P=200	a_1&b_1	384	119	31.0	−0.9802	0.0305
K=2, τ_P=300	K&τ_P	384	26	6.8	−0.9868	0.0265
K=2, τ_P=300	a_1&b_1	384	124	32.3	−0.9868	0.0265

It is important to note that although the discrete-time ARX parameters (a_1, b_1) tend to follow an approximately normal distribution for large N, the corresponding continuous-time parameters K_{ARX} and $\tau_{P,ARX}$, obtained through nonlinear transformations of these coefficients, are not necessarily normally distributed. The mapping from discrete to continuous time involves logarithmic and ratio operations, such as shown in Eq. 2.21, which introduce nonlinearities into the parameters. Even if (a_1, b_1) are Gaussian, these transformations lead to skewed or heavy-tailed distributions, particularly for small sample sizes or when a_1 is close to unity like in Plant 1 and 2 cases, as can be seen in Tables 4.11 and 4.12.

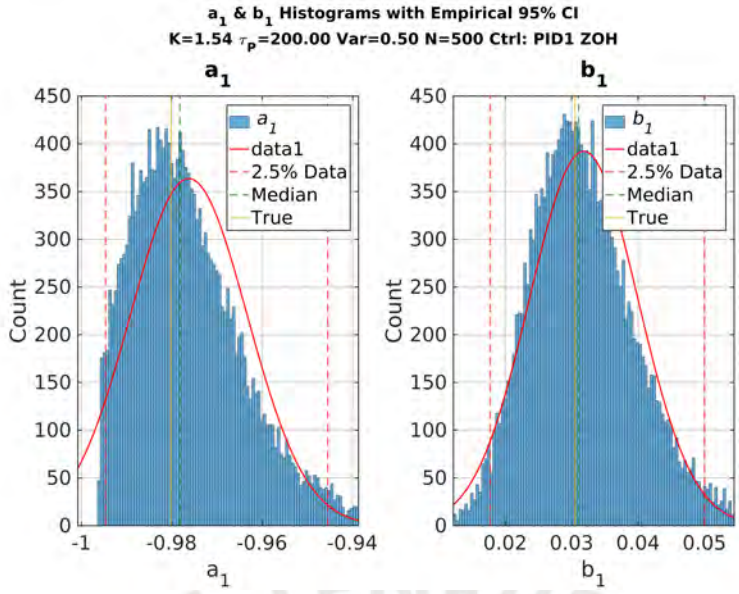
Beyond the previous histograms, the joint behaviour of the plant parameters is shown by the 2D histogram/heatmap "top views" of $(K_{ARX}, \tau_{P,ARX})$. For Plant $K = 1.54,\ \tau_P = 200\,\mathrm{s}$, the sequence in Fig. 4.15 ($\sigma^2 = 0.50$) shows a clear transition from a diffuse, noisy cloud at $N = 500$–1000 to a tight, nearly elliptical cluster by $N = 50{,}000$–$100{,}000$. The dominant shape is a positively sloped ridge, indicating strong positive correlation between the K_{ARX} and $\tau_{P,ARX}$ estimators. This ridge reflects the well-known identifiability coupling of gain and time constant in closed loop: combinations of slightly larger K with slightly larger τ_P can produce similar input–output behaviour, so uncertainty is largest along this "identification manifold," while it is much smaller in the orthogonal direction. Increasing the noise level to $\sigma^2 = 10.00$ (Fig. 4.16) broadens the cloud substantially, especially along the ridge, and the tails become more pronounced at the smallest N; nevertheless, with large N the cluster again contracts toward an almost Gaussian ellipse centred on the true parameter pair.

The same qualitative pattern is observed for the other plants at $\sigma^2 = 0.50$. For Plant $K = 2.00,\ \tau_P = 300\,\mathrm{s}$ (Fig. 4.17), the ridge is slightly more elongated, indicating stronger coupling and a slower reduction of uncertainty along the principal direction as N grows. For Plant $K = 1.00,\ \tau_P = 24\,\mathrm{s}$ (Fig. 4.18), the joint distribution also tightens markedly with N, but the ellipse becomes compact earlier, consistent with the faster underlying time scale. Across plants, the heatmaps confirm three robust features: (i) strong positive correlation of $(K_{ARX}, \tau_{P,ARX})$ for small and moderate N; (ii) anisotropic uncer-

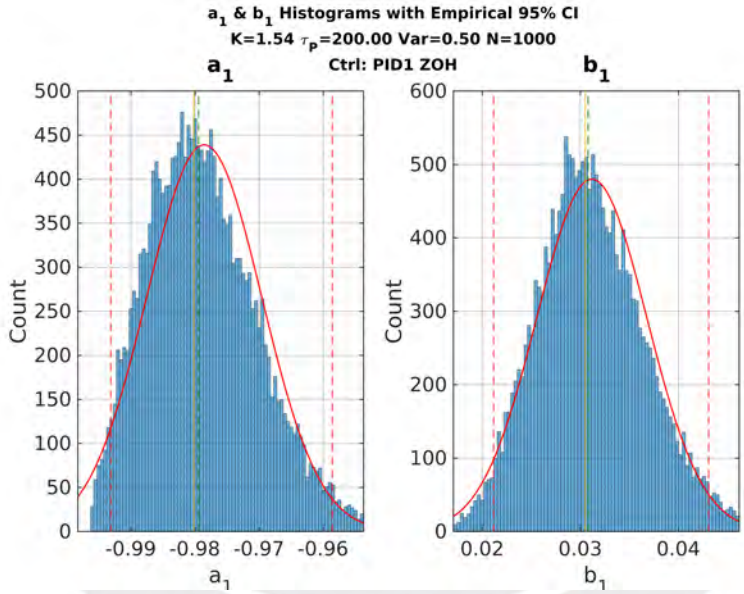
tainty that shrinks fastest in the direction orthogonal to the ridge; and (iii) eventual concentration of the joint distribution around the true parameters as N becomes large, consistent with asymptotic normality of the estimators.

These joint plots complement the 1D histograms and the normality testing in Tables 4.11 and 4.12. Even when the 1D marginals for K_{ARX} and $\tau_{P,ARX}$ appear approximately Gaussian at large N, the 2D heatmaps show that their *dependence structure* is non-negligible: uncertainty remains highly anisotropic and aligned with the model's sensitivity directions. This explains why marginal CIs can be narrow while the joint spread is still noticeable along the ridge, and it motivates the use of joint (elliptical) confidence regions or reporting the empirical covariance of $(K_{ARX}, \tau_{P,ARX})$ in addition to the univariate summaries.

All the histograms of a_1 & b_1 and K & τ_P for each configuration and data size can be found in the results folder, located at "`Thesis_2025_Final_Results_JADS/ 3. Histograms and other plots/Histograms_by_Plant_Noise_DataSize`".

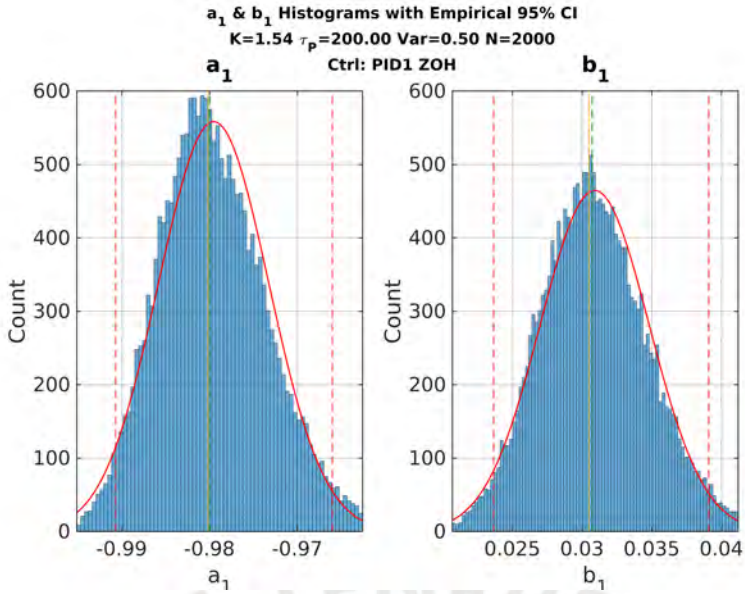


(a) $N = 500$

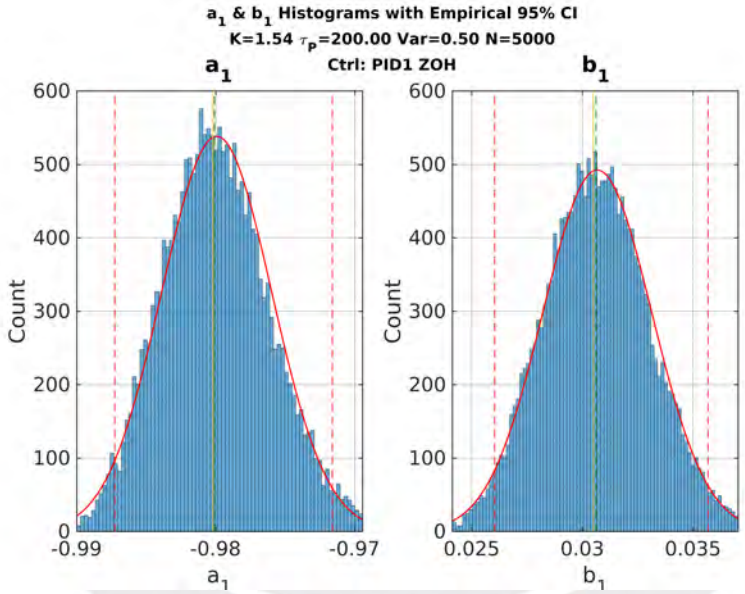


(b) $N = 1000$

Figure 4.7: Histograms of a_1 and b_1 for $N = 500$ and $N = 1000$ (Plant $K = 1.54$, $\tau_P = 200\,\mathrm{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

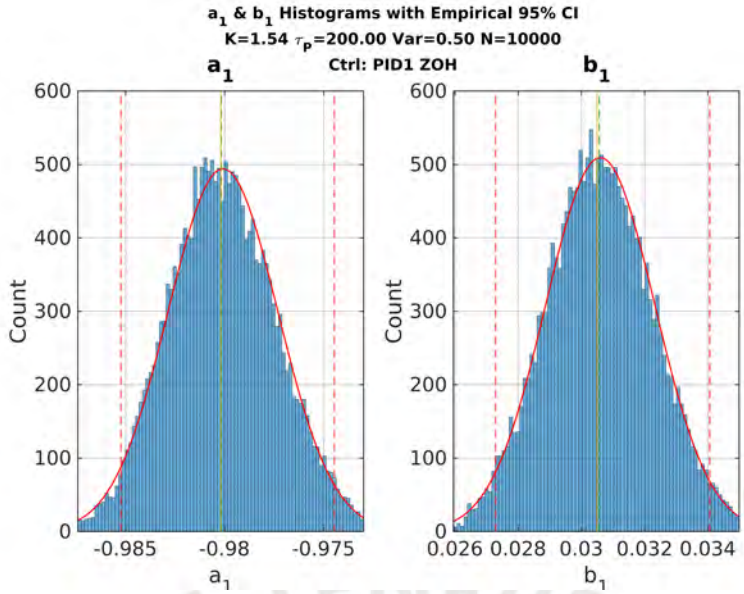


(a) $N = 2000$

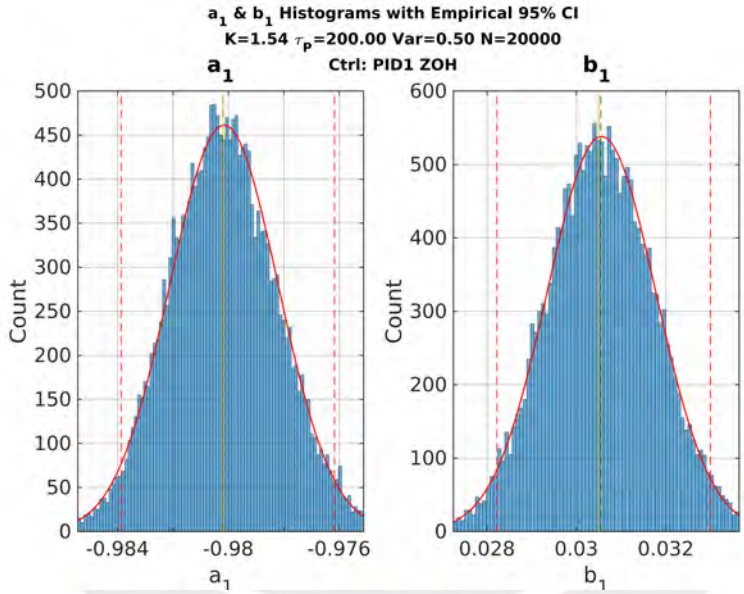


(b) $N = 5000$

Figure 4.8: Histograms of a_1 and b_1 for $N = 2000$ and $N = 5000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

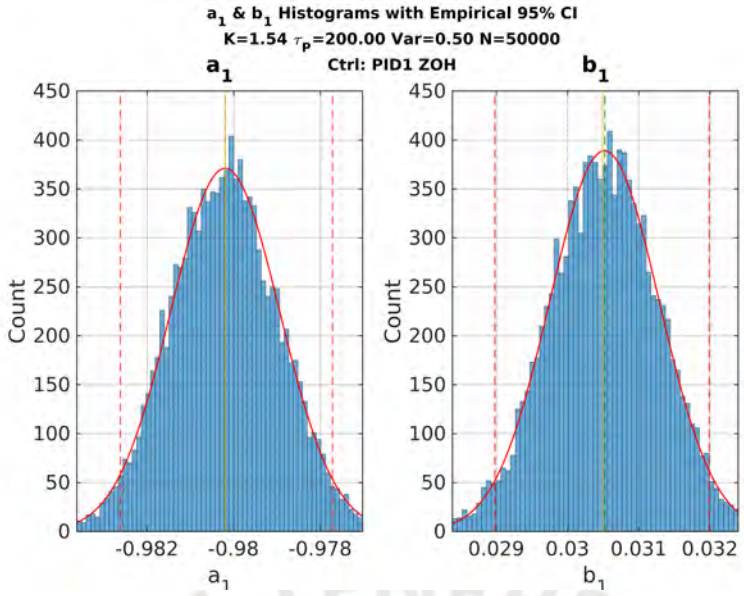


(a) $N = 10{,}000$

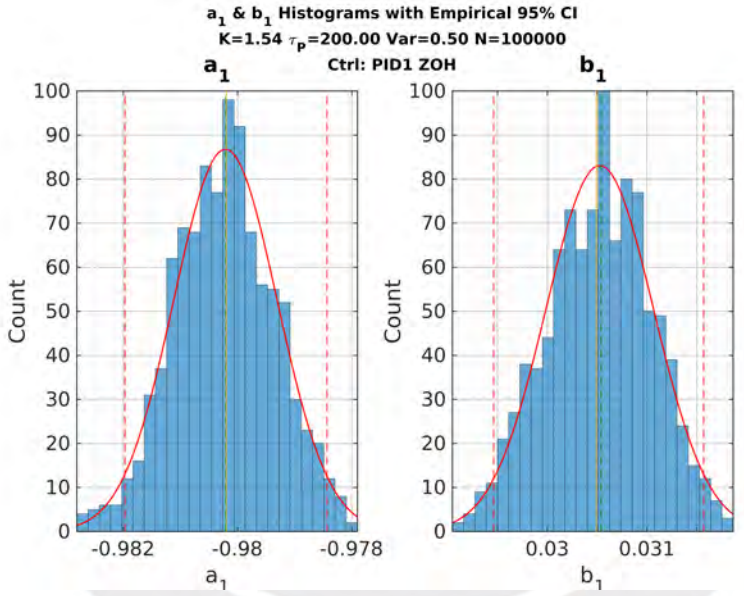


(b) $N = 20{,}000$

Figure 4.9: Histograms of a_1 and b_1 for $N = 10{,}000$ and $N = 20{,}000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

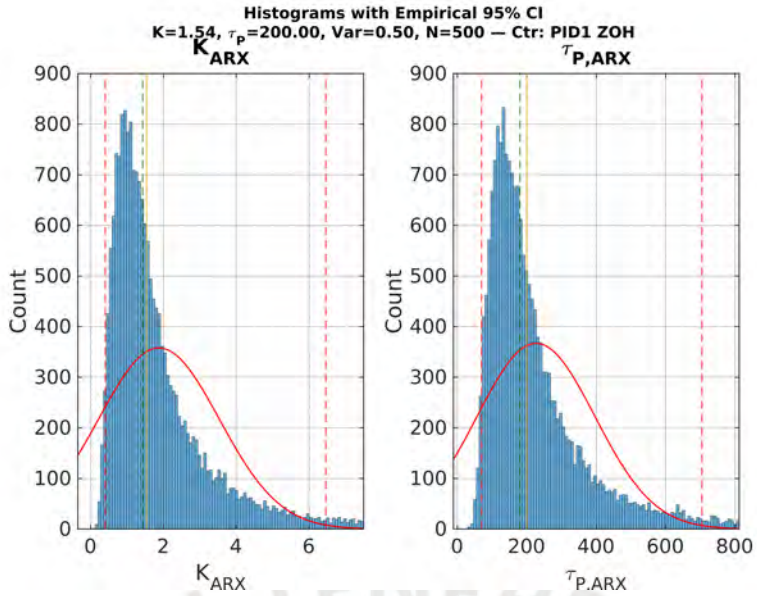


(a) $N = 50{,}000$

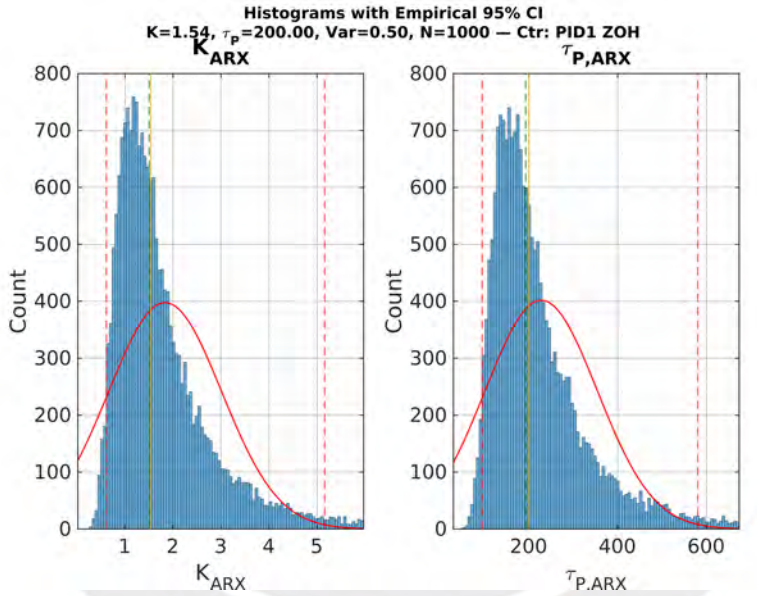


(b) $N = 100{,}000$

Figure 4.10: Histograms of a_1 and b_1 for $N = 50{,}000$ and $N = 100{,}000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

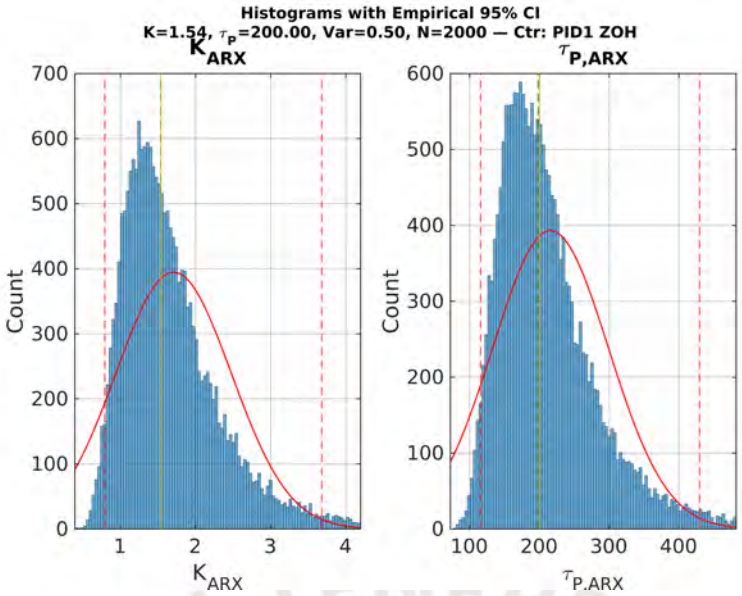


(a) $N = 500$

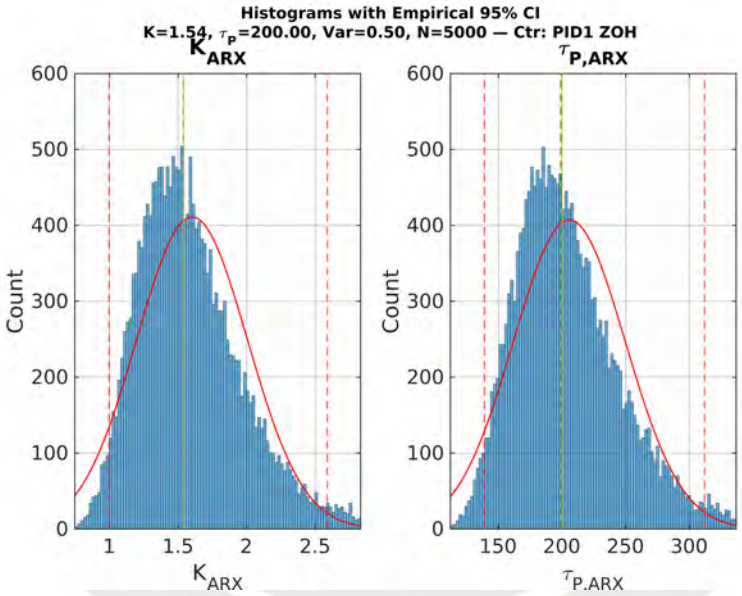


(b) $N = 1000$

Figure 4.11: Histograms for data sizes $N = 500$ and $N = 1000$ (Plant $K = 1.54$, $\tau_P = 200\,\mathrm{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

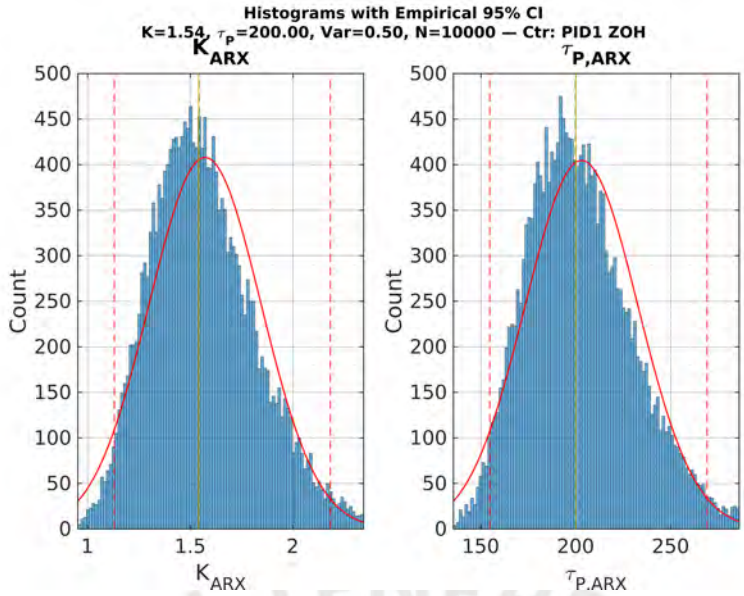


(a) $N = 2000$

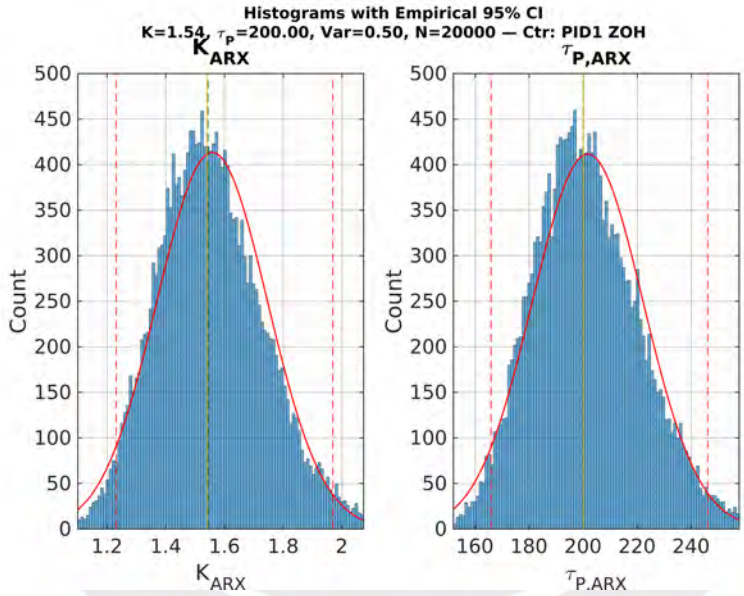


(b) $N = 5000$

Figure 4.12: Histograms for data sizes $N = 2000$ and $N = 5000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

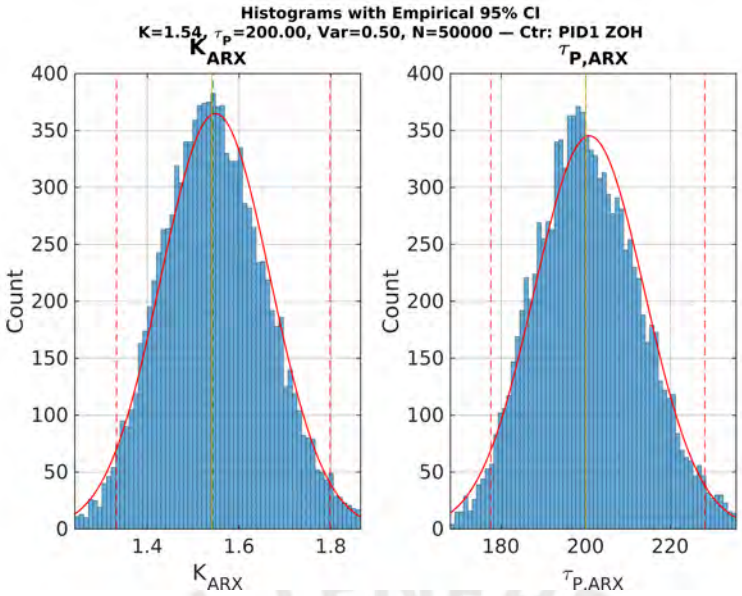


(a) $N = 10{,}000$

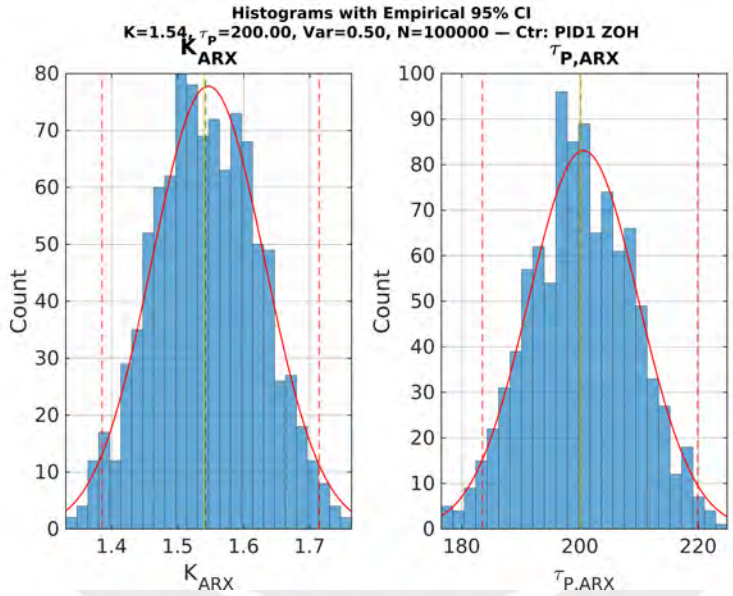


(b) $N = 20{,}000$

Figure 4.13: Histograms for data sizes $N = 10{,}000$ and $N = 20{,}000$ (Plant $K = 1.54$, $\tau_P = 200\,\mathrm{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

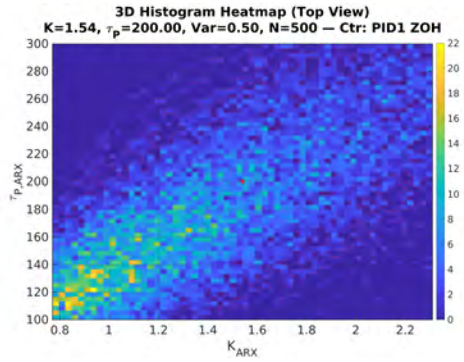


(a) $N = 50{,}000$

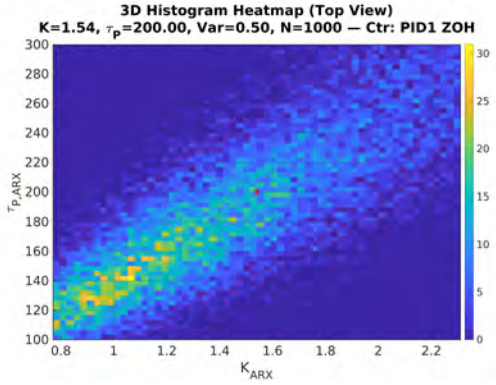


(b) $N = 100{,}000$

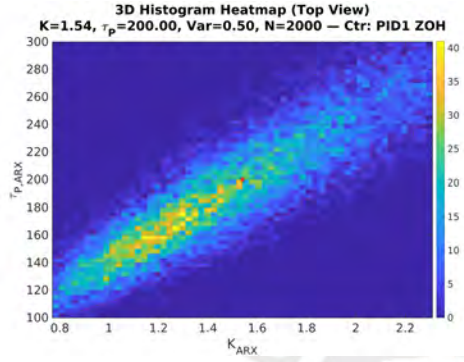
Figure 4.14: Histograms for data sizes $N = 50{,}000$ and $N = 100{,}000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).



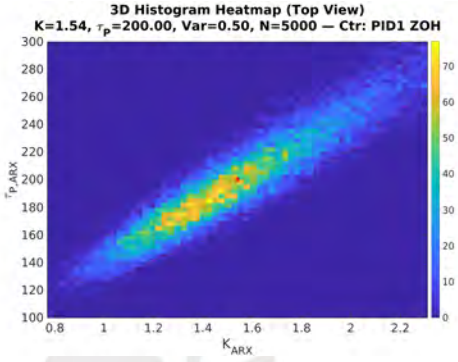
(a) $N = 500$



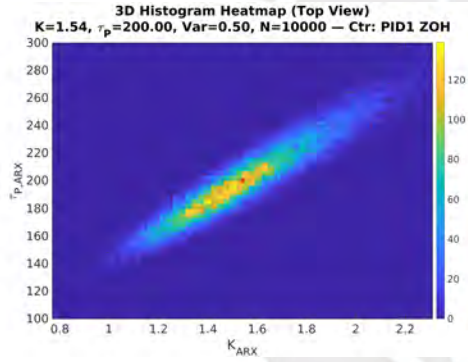
(b) $N = 1000$



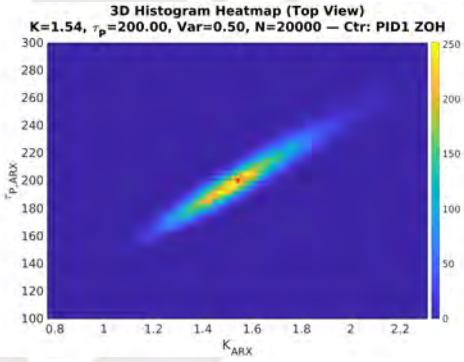
(c) $N = 2000$



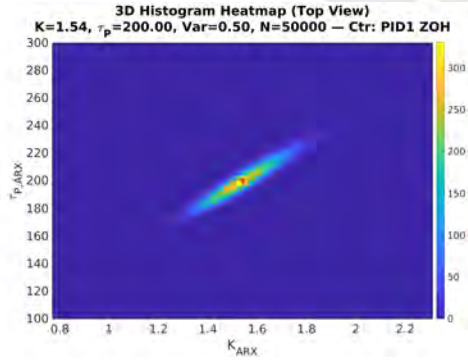
(d) $N = 5000$



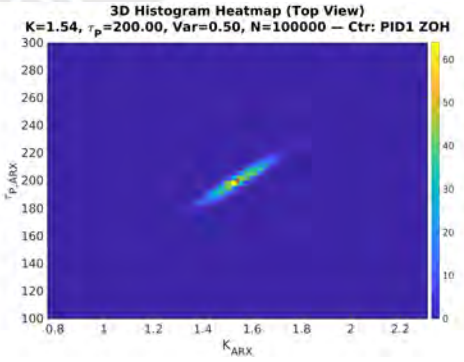
(e) $N = 10{,}000$



(f) $N = 20{,}000$

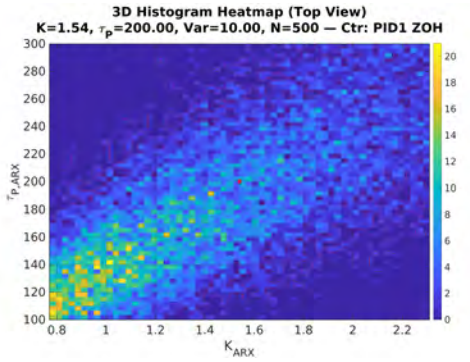


(g) $N = 50{,}000$

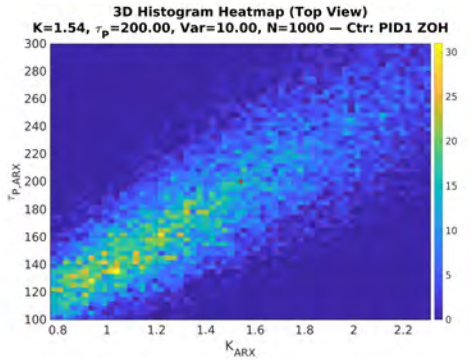


(h) $N = 100{,}000$

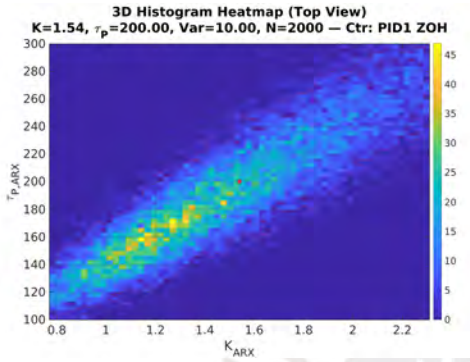
Figure 4.15: 2D histograms/heatmaps (top view) of $(K_{ARX}, \tau_{P,ARX})$ for noise variance $\sigma^2 = 0.50$, Plant $K = 1.54,\ \tau_P = 200\,\text{s}$, controller PID1 (ZOH), across all data sizes N.



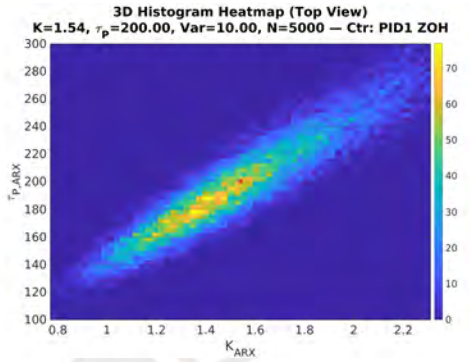
(a) $N = 500$



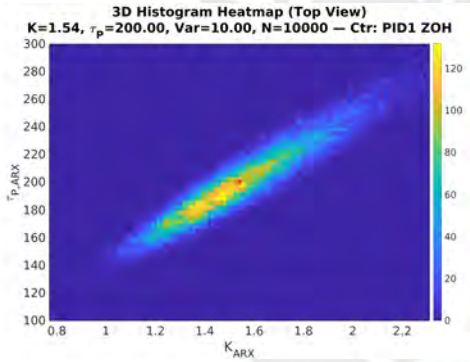
(b) $N = 1000$



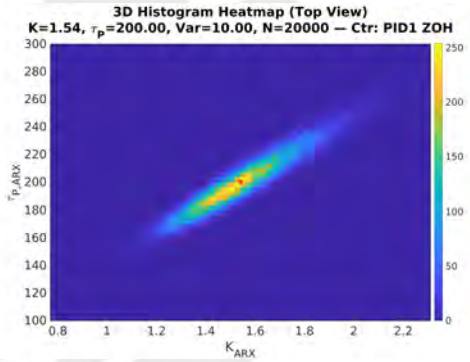
(c) $N = 2000$



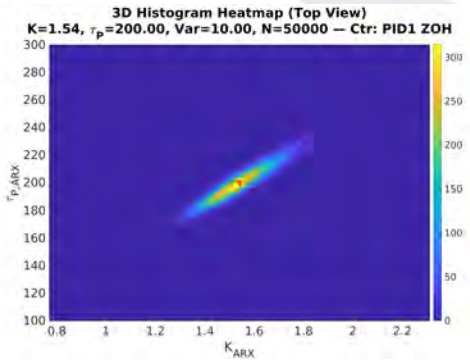
(d) $N = 5000$



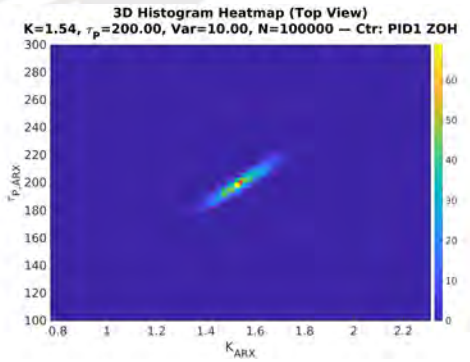
(e) $N = 10{,}000$



(f) $N = 20{,}000$

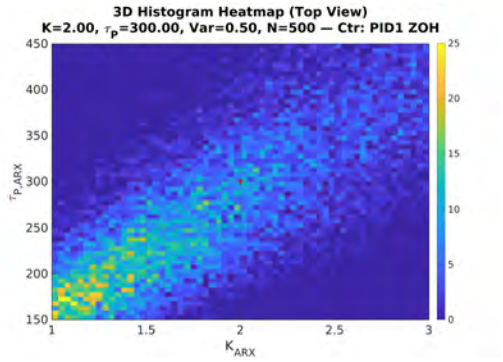


(g) $N = 50{,}000$

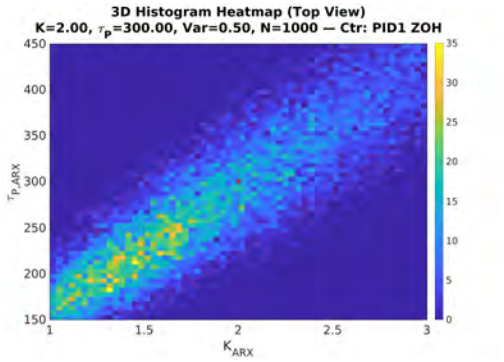


(h) $N = 100{,}000$

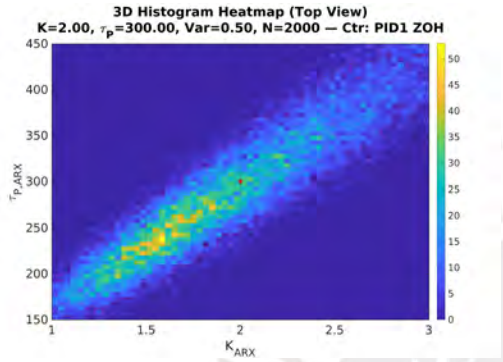
Figure 4.16: 2D histograms/heatmaps (top view) of $(K_{ARX}, \tau_{P,ARX})$ for noise variance $\sigma^2 = 10.00$, Plant $K = 1.54,\ \tau_P = 200\,\text{s}$, controller PID1 (ZOH), across all data sizes N.



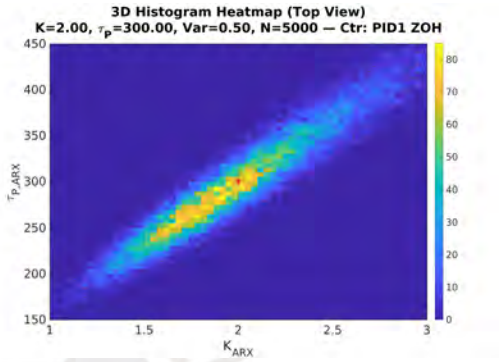
(a) $N = 500$



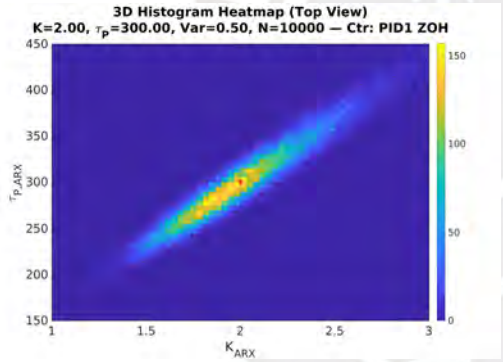
(b) $N = 1000$



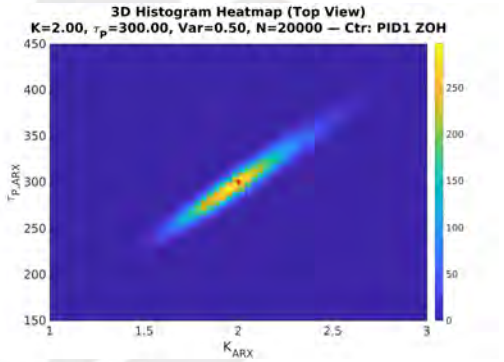
(c) $N = 2000$



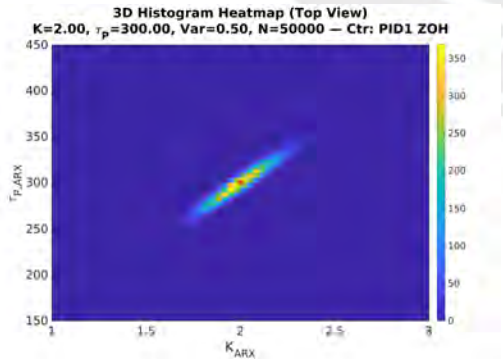
(d) $N = 5000$



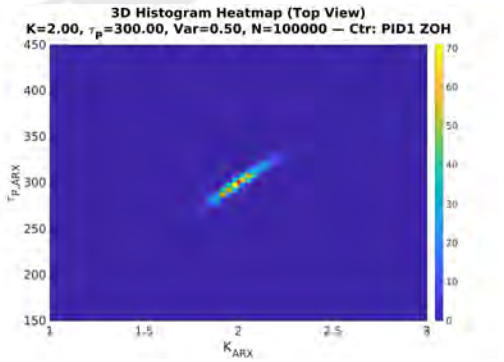
(e) $N = 10{,}000$



(f) $N = 20{,}000$

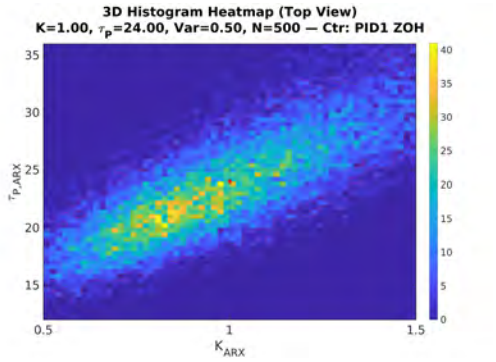


(g) $N = 50{,}000$

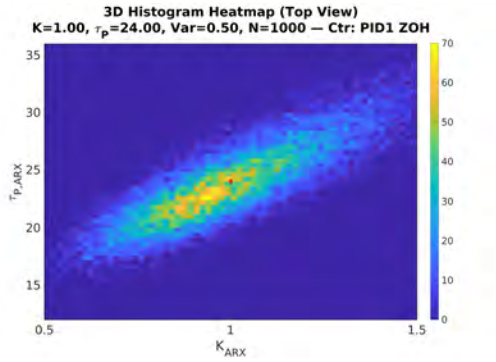


(h) $N = 100{,}000$

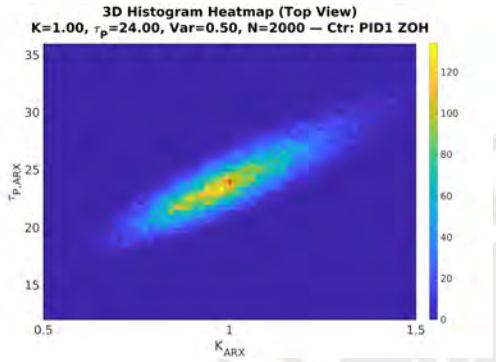
Figure 4.17: 2D histograms/heatmaps (top view) of $(K_{ARX}, \tau_{P,ARX})$ for noise variance $\sigma^2 = 0.50$, Plant $K = 2.00,\ \tau_P = 300\,\text{s}$, controller PID1 (ZOH), across all data sizes N.



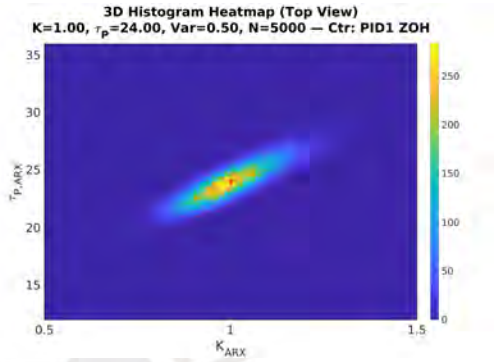
(a) $N = 500$



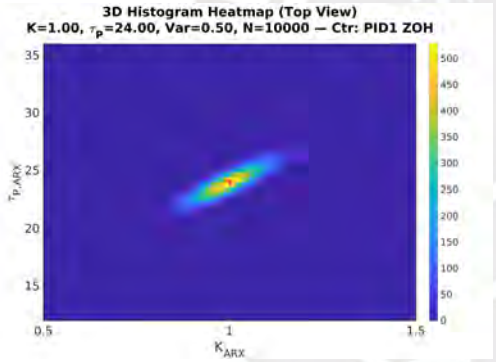
(b) $N = 1000$



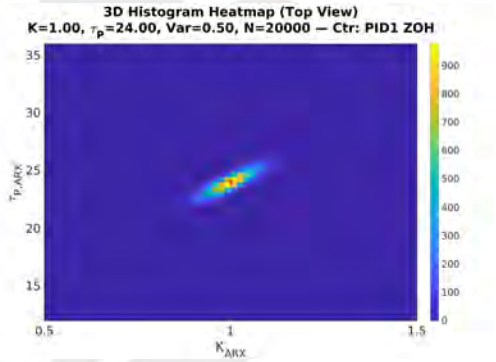
(c) $N = 2000$



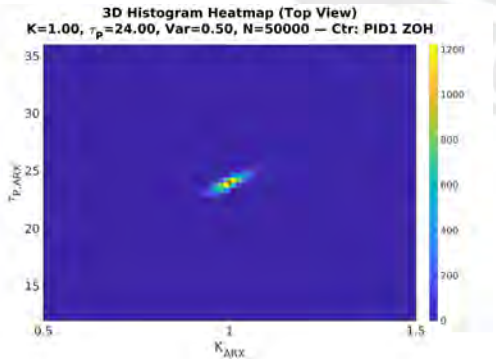
(d) $N = 5000$



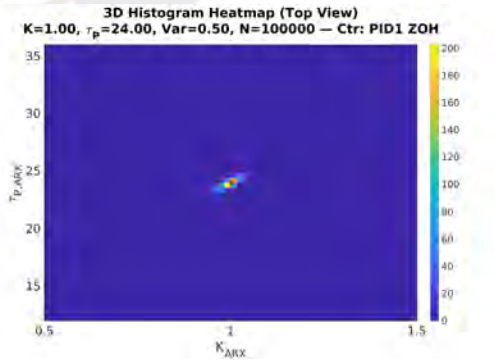
(e) $N = 10{,}000$



(f) $N = 20{,}000$



(g) $N = 50{,}000$



(h) $N = 100{,}000$

Figure 4.18: 2D histograms/heatmaps (top view) of $(K_{ARX}, \tau_{P,ARX})$ for noise variance $\sigma^2 = 0.50$, Plant $K = 1.00,\ \tau_P = 24\,\text{s}$, controller PID1 (ZOH), across all data sizes N.

Section 4.1.3: 1D KDE plots

Figures 4.19–4.20 and 4.21–4.22 present the one-dimensional Kernel Density Estimation (1D KDE) results for the estimated ARX parameters K_{ARX} and $\tau_{P,\mathrm{ARX}}$, obtained under constant plant and noise conditions ($K = 1.54$, $\tau_P = 200\,\mathrm{s}$, $\sigma^2 = 0.5$) using the PID1 controller with zero-order-hold (ZOH) discretization. Each plot shows the empirical probability density of the estimated parameter (gray area), along with the empirical 95% confidence interval (red dashed lines), the median (green line), the mode (blue line), and the true parameter value (orange dashed line).

For the smallest datasets, shown in Figures 4.19 and 4.20, the parameter distributions are relatively wide and slightly skewed. Both K_{ARX} and $\tau_{P,\mathrm{ARX}}$ exhibit noticeable bias, with peaks displaced from the true values. This bias arises from limited excitation and the higher influence of measurement noise in short data records. The confidence intervals are correspondingly broad, indicating low estimator precision.

In contrast, Figures 4.21 and 4.22 show the distributions for the largest data sizes. Here, both K_{ARX} and $\tau_{P,\mathrm{ARX}}$ converge closely to their true plant values, and the KDEs become sharply peaked and nearly symmetric. The empirical confidence intervals are substantially narrower, quantifying the marked improvement in estimation precision. These results clearly show the transition from biased, high-variance estimates at small N to consistent, near-Gaussian estimates at large N, in agreement with the asymptotic properties of least-squares estimators.

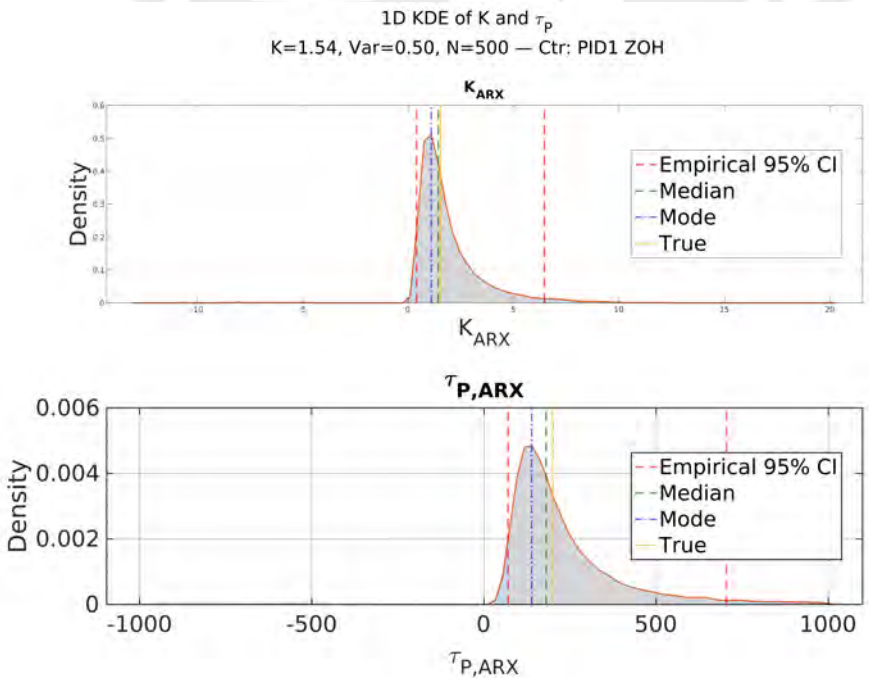


Figure 4.19: 1D KDE for $N = 500$ (Plant $K = 1.54$, $\tau_P = 200\,\mathrm{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

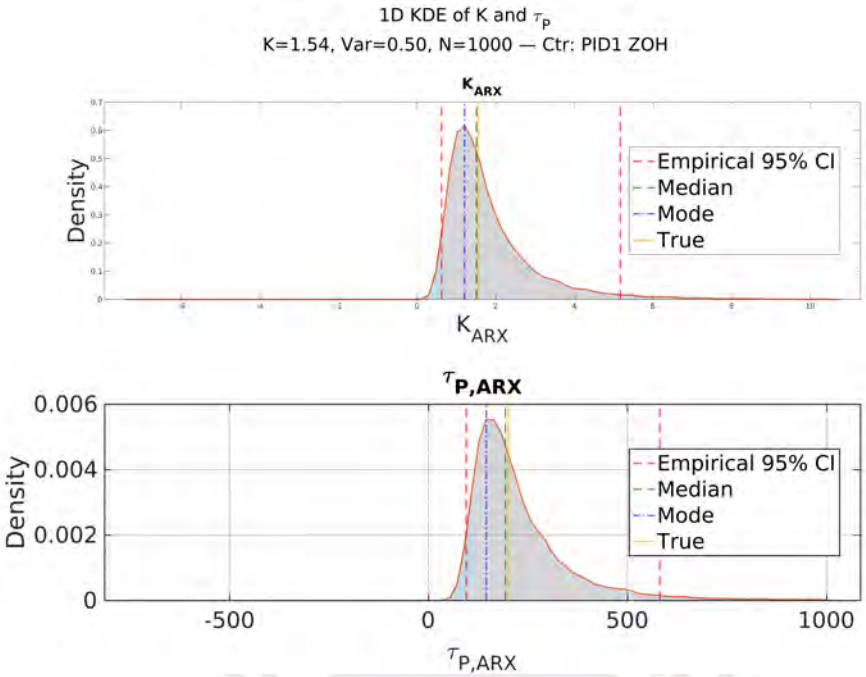


Figure 4.20: 1D KDE for $N = 1000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

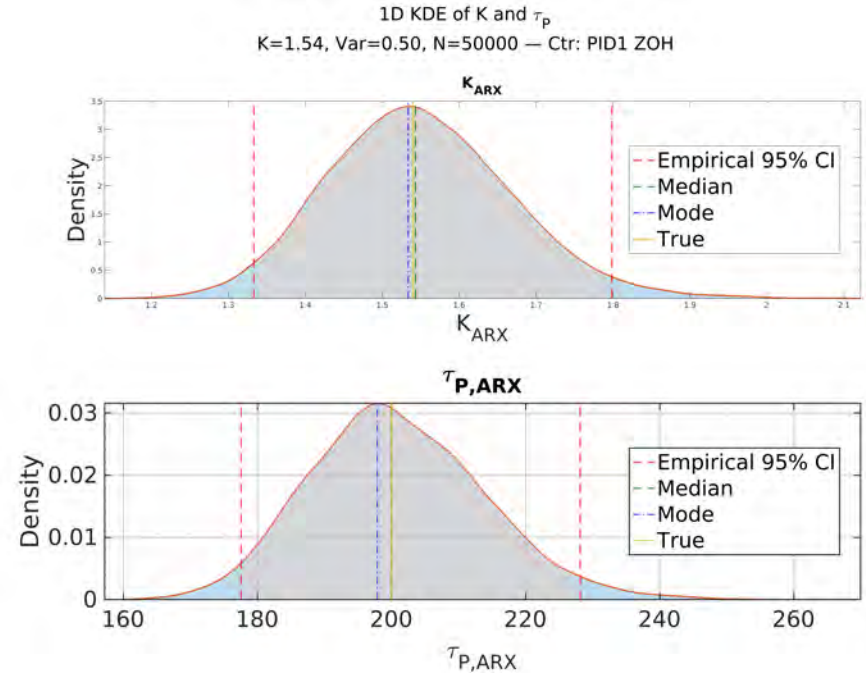


Figure 4.21: 1D KDE for $N = 50{,}000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

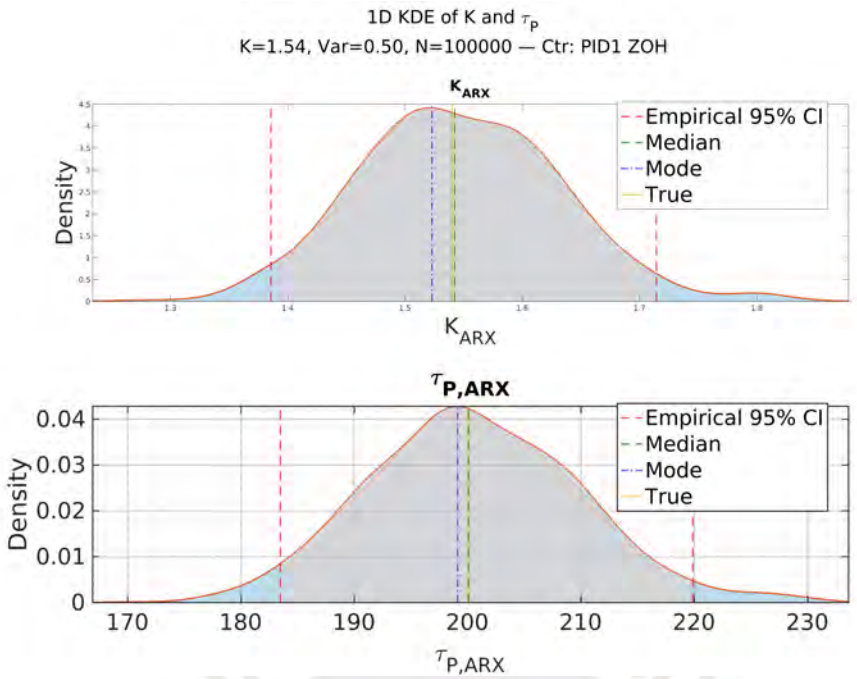
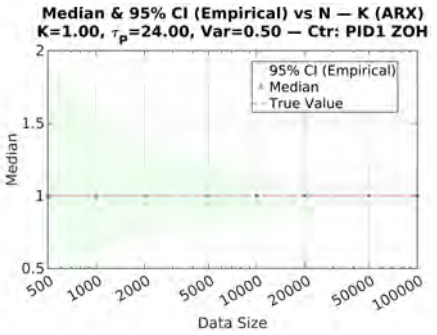


Figure 4.22: 1D KDE for $N = 100{,}000$ (Plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $\sigma^2 = 0.50$; PID1, ZOH).

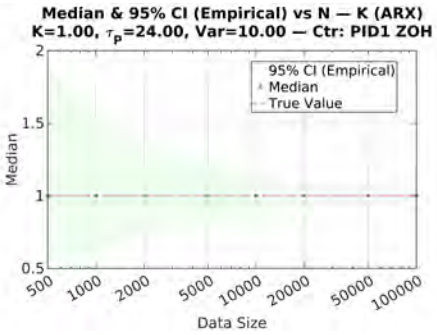
Section 4.1.4: Median estimates vs. data size and confidence intervals

Figures 4.23–4.25 summarize the *empirical* confidence intervals (CIs) obtained from Monte Carlo trials for the parameters K_{ARX} (top row of each panel) and $\tau_{P,\text{ARX}}$ (bottom row), under two noise levels $\sigma^2 \in \{0.50, 10.00\}$ and PID1 with ZOH. Across all three plants, the empirical CIs visibly shrink as the data length N increases, reflecting the reduction in sampling variability. Importantly, the empirical CIs consistently include the true parameter value in our experiments, and their width decreases roughly at the rate expected for $\sqrt{N}$-consistent estimators. The tightening is more pronounced for K_{ARX}; $\tau_{P,\text{ARX}}$ displays wider intervals—especially for small N and higher noise—indicating greater sensitivity of the time-constant estimate to measurement noise and closed-loop dynamics.

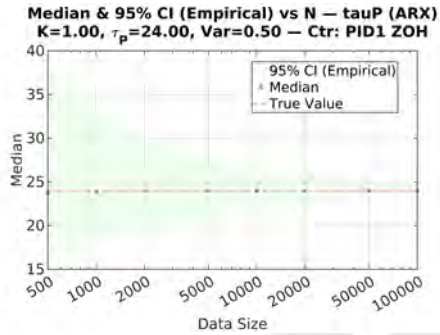
Figures 4.26–4.28 report the *analytical* CIs computed with the closed-form expression in (2.29) applied to the median curves (top: K_{ARX}, bottom: $\tau_{P,\text{ARX}}$). While these intervals also narrow with increasing N, they do not always cover the true value—occasionally even at the largest N. This under-coverage is attributable to finite-sample effects that violate the assumptions behind (2.29): the sampling distribution of the estimator can be skewed for moderate N (see the KDEs in Sec. 4.1.3), and the variance model used by the analytical CI relies on asymptotic normality and a locally linear approximation that may be inaccurate when noise is high or when the mapping from data to parameters is strongly nonlinear. The effect is more evident for $\tau_{P,\text{ARX}}$, whose sampling distribution remains asymmetric longer than that of K_{ARX}. Taken together, the empirical CIs (Figs. 4.23–4.25) provide reliable finite-sample coverage at the cost of simulation effort, whereas the analytical CIs (Figs. 4.26–4.28, via (2.29)) are fast and interpretable but can be mis-calibrated in small or noisy samples. In practice, accuracy improves with N; for large datasets the analytical and empirical bands converge, consistent with asymptotic theory.



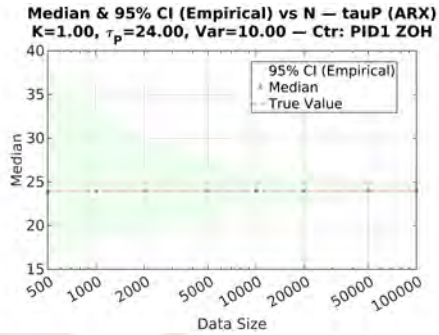
(a) $K_{\text{ARX}},\ \sigma^2 = 0.50$



(b) $K_{\text{ARX}},\ \sigma^2 = 10.00$

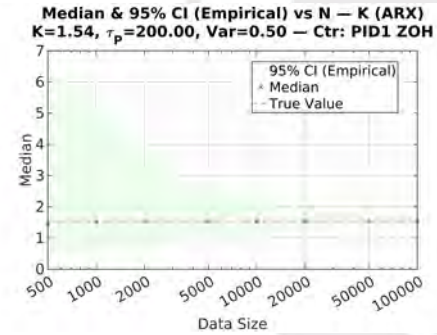


(c) $\tau_{P,\text{ARX}},\ \sigma^2 = 0.50$

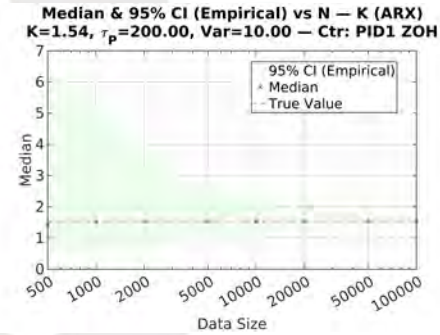


(d) $\tau_{P,\text{ARX}},\ \sigma^2 = 10.00$

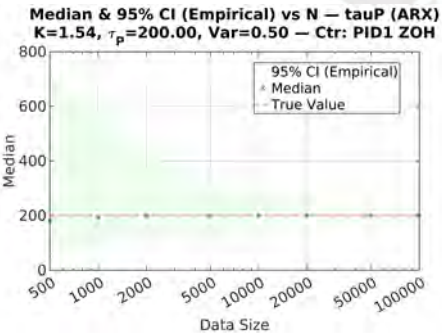
Figure 4.23: Median vs. N with *empirical* confidence intervals for for plant 3, controller PID1 (ZOH).



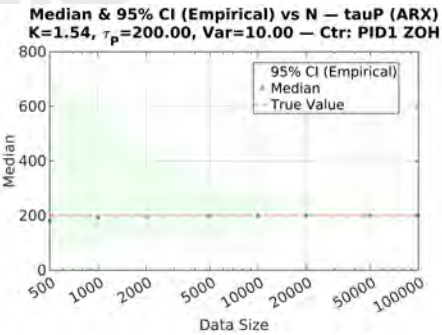
(a) $K_{\text{ARX}},\ \sigma^2 = 0.50$



(b) $K_{\text{ARX}},\ \sigma^2 = 10.00$

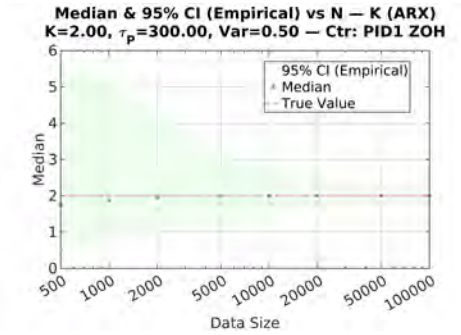


(c) $\tau_{P,\text{ARX}},\ \sigma^2 = 0.50$

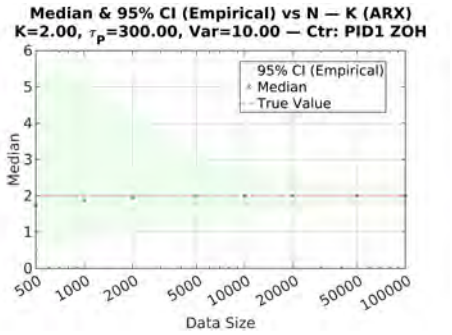


(d) $\tau_{P,\text{ARX}},\ \sigma^2 = 10.00$

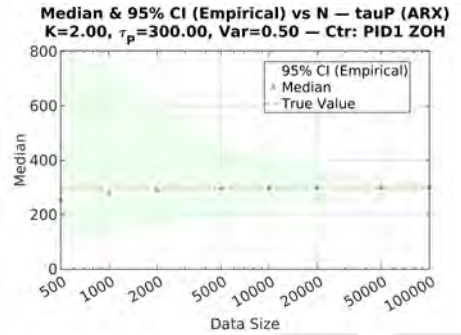
Figure 4.24: Median vs. N with *empirical* confidence intervals for plant 1, controller PID1 (ZOH).



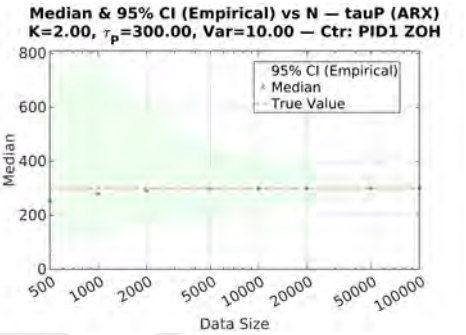
(a) $K_{\mathrm{ARX}},\ \sigma^2 = 0.50$



(b) $K_{\mathrm{ARX}},\ \sigma^2 = 10.00$

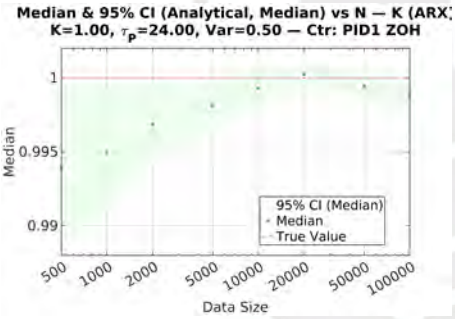


(c) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 0.50$

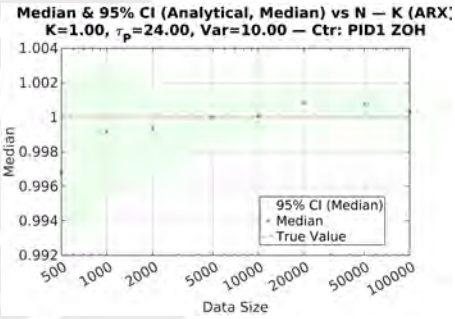


(d) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 10.00$

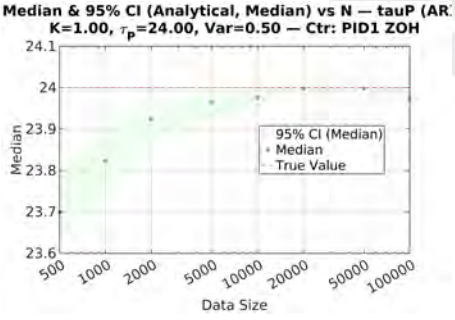
Figure 4.25: Median vs. N with *empirical* confidence intervals for plant 2, controller PID1 (ZOH).



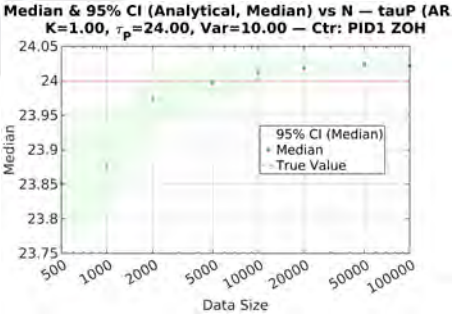
(a) $K_{\mathrm{ARX}},\ \sigma^2 = 0.50$



(b) $K_{\mathrm{ARX}},\ \sigma^2 = 10.00$

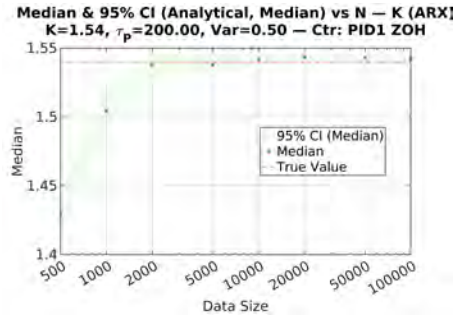


(c) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 0.50$

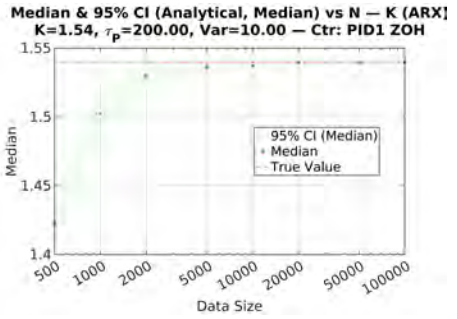


(d) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 10.00$

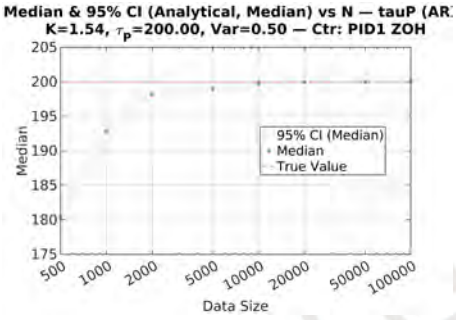
Figure 4.26: Median vs. N with *analytical* confidence intervals for K_{ARX} and $\tau_{P,\mathrm{ARX}}$, plant $K = 1.00$, $\tau_P = 24\,\mathrm{s}$; controller PID1 (ZOH).



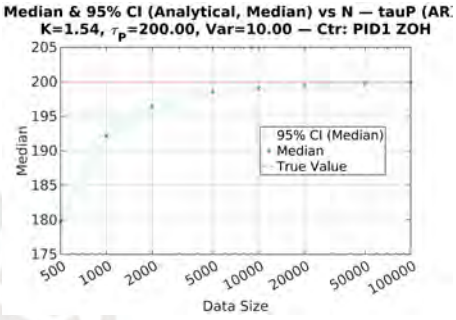
(a) $K_{\mathrm{ARX}},\ \sigma^2 = 0.50$



(b) $K_{\mathrm{ARX}},\ \sigma^2 = 10.00$

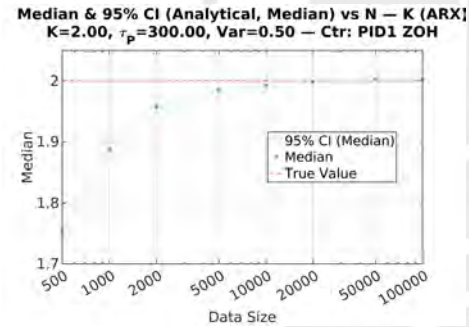


(c) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 0.50$

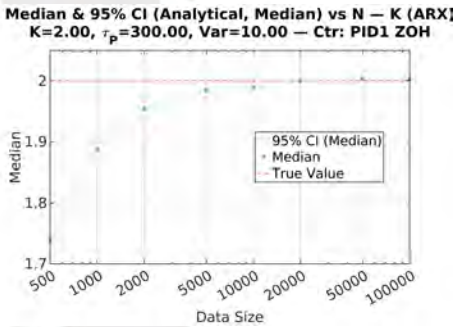


(d) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 10.00$

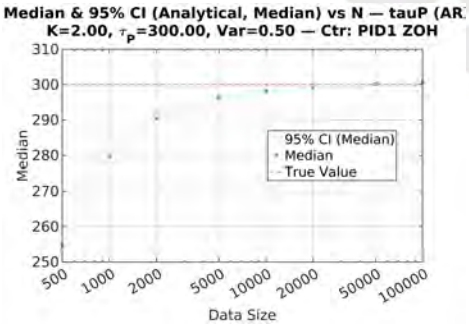
Figure 4.27: Median vs. N with *analytical* confidence intervals for K_{ARX} and $\tau_{P,\mathrm{ARX}}$, plant $K = 1.54$, $\tau_P = 200\,\mathrm{s}$; controller PID1 (ZOH).



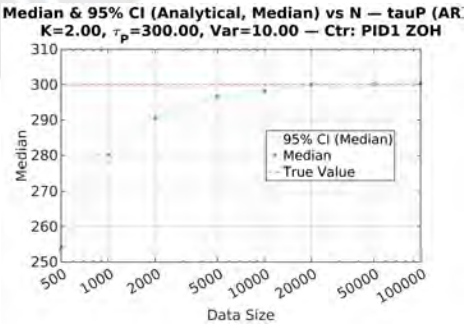
(a) $K_{\mathrm{ARX}},\ \sigma^2 = 0.50$



(b) $K_{\mathrm{ARX}},\ \sigma^2 = 10.00$



(c) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 0.50$



(d) $\tau_{P,\mathrm{ARX}},\ \sigma^2 = 10.00$

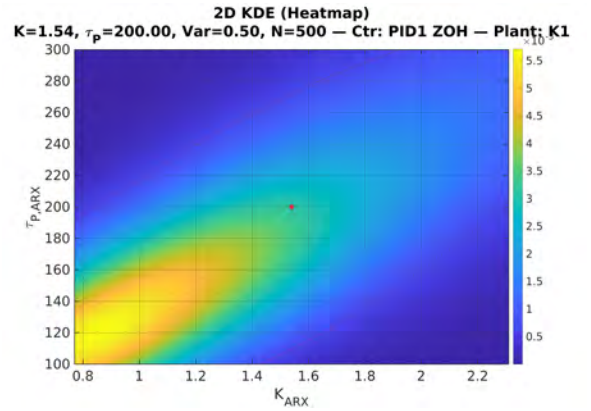
Figure 4.28: Median vs. N with *analytical* confidence intervals for K_{ARX} and $\tau_{P,\mathrm{ARX}}$, plant $K = 2.00$, $\tau_P = 300\,\mathrm{s}$; controller PID1 (ZOH).

Section 4.1.5: Empirical 95% confidence region for different data size

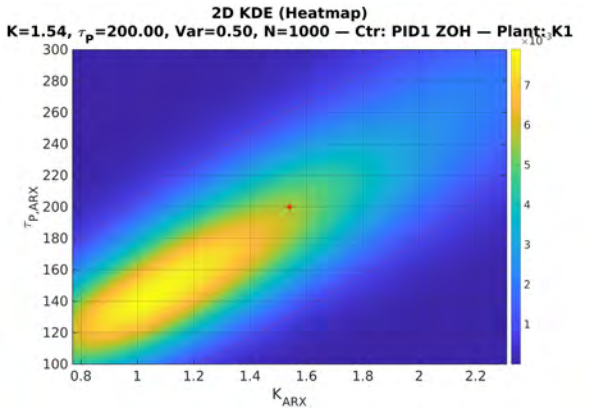
Figures 4.29–4.32 show the evolution of the empirical 95% confidence regions obtained from the two-dimensional kernel density estimates (KDEs) of the estimated parameters (K, τ_P) and (a_1, b_1). In each plot, the red contour line delineates the empirical 95% confidence boundary derived from Monte Carlo simulations, while the color intensity represents the probability density and the black marker denotes the true parameter values.

For small datasets ($N \leq 1000$), the red confidence regions are broad and slightly distorted, indicating high estimation variance and noticeable bias relative to the true parameter location. As the number of samples increases ($N = 2000$–5000), these regions contract and gradually align with the true values, showing reduced dispersion and improved accuracy. For large datasets ($N \geq 10^4$), the contours become tight and nearly elliptical, centered around the true parameters, and the densities exhibit a Gaussian-like symmetry.

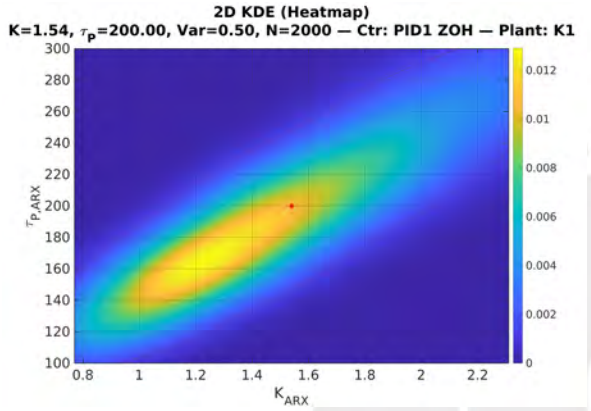
Overall, the progression of the red 95% contours across increasing data sizes highlights the expected finite-sample behavior: as more data become available, the empirical confidence regions shrink approximately with $1/\sqrt{N}$, the estimates converge to their true values, and parameter correlation weakens, confirming the asymptotic consistency of the identification method even under closed-loop operation.



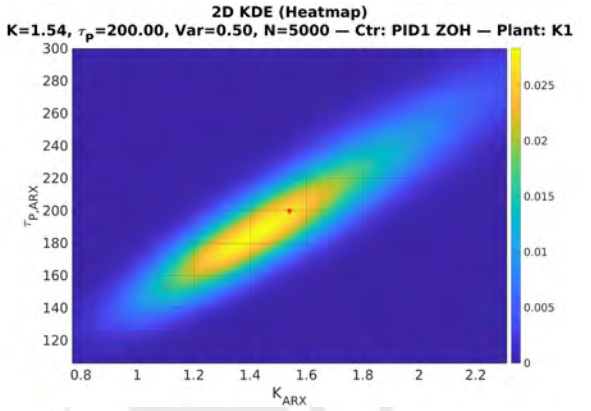
(a) $N = 500$



(b) $N = 1000$

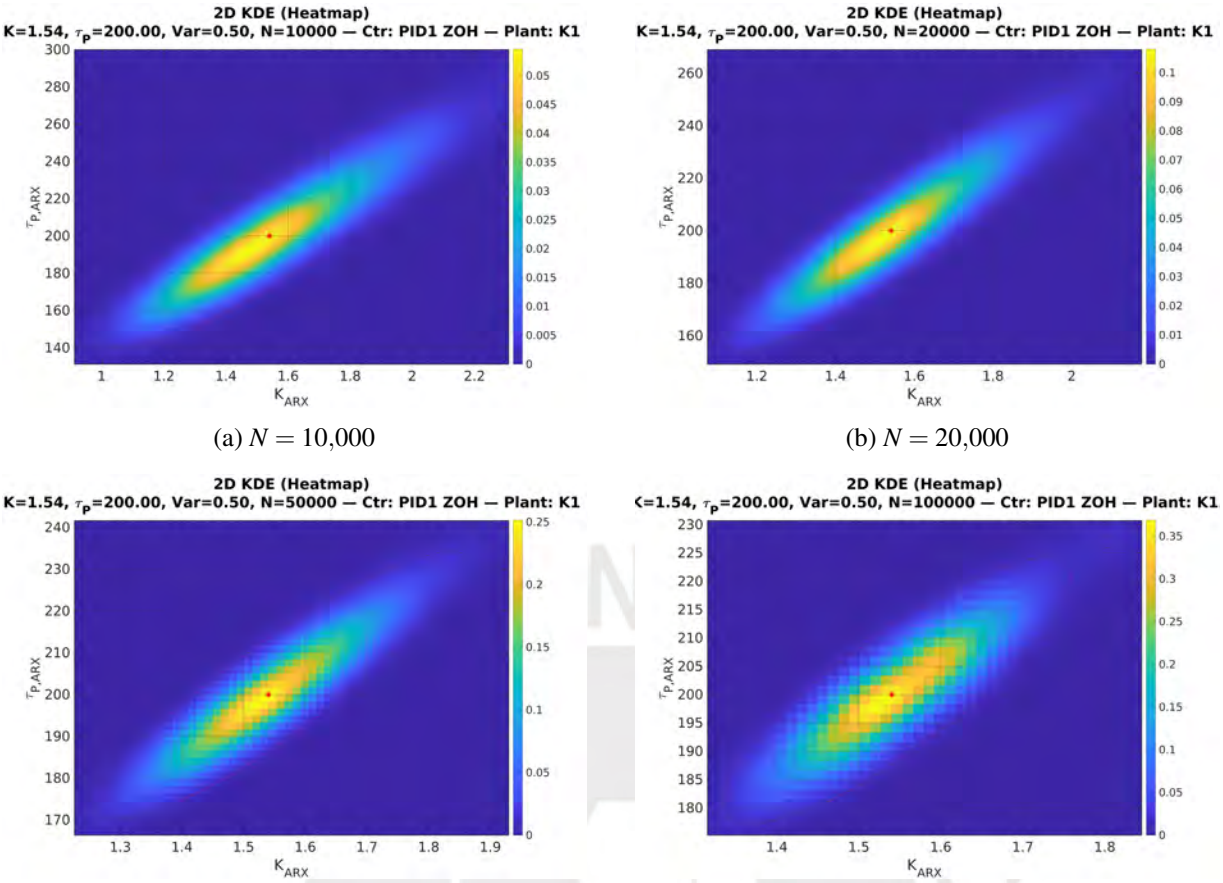


(c) $N = 2000$



(d) $N = 5000$

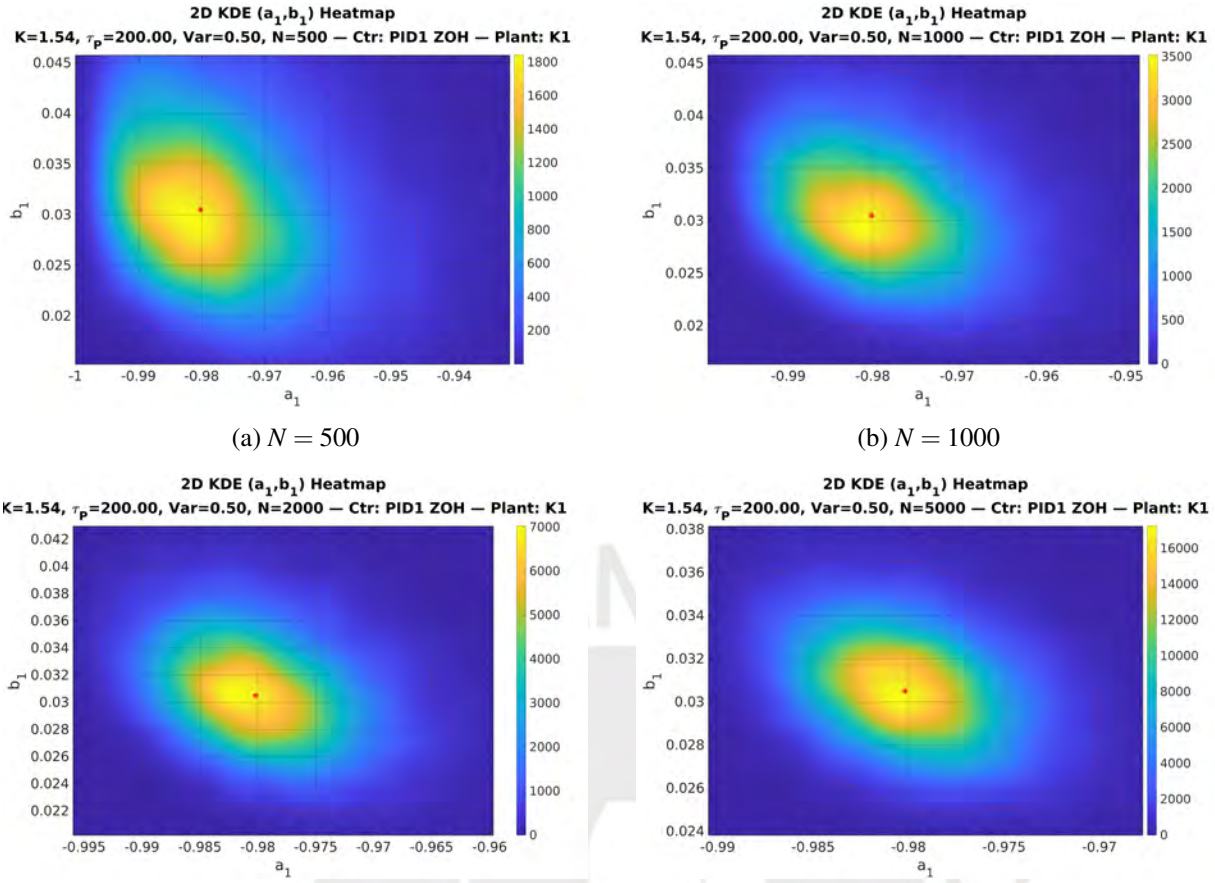
Figure 4.29: 2D KDE of (K, τ_P) estimates for plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $L = 20\,\text{s}$; controller PID1 (ZOH); noise variance $\sigma^2 = 0.5$. Panels: (a) $N = 500$, (b) $N = 1000$, (c) $N = 2000$, (d) $N = 5000$.



(a) $N = 10{,}000$ (b) $N = 20{,}000$

(c) $N = 50{,}000$ (d) $N = 100{,}000$

Figure 4.30: 2D KDE of (K, τ_P) estimates for plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $L = 20\,\text{s}$; controller PID1 (ZOH); noise variance $\sigma^2 = 0.5$. Panels: (a) $N = 10{,}000$, (b) $N = 20{,}000$, (c) $N = 50{,}000$, (d) $N = 100{,}000$.



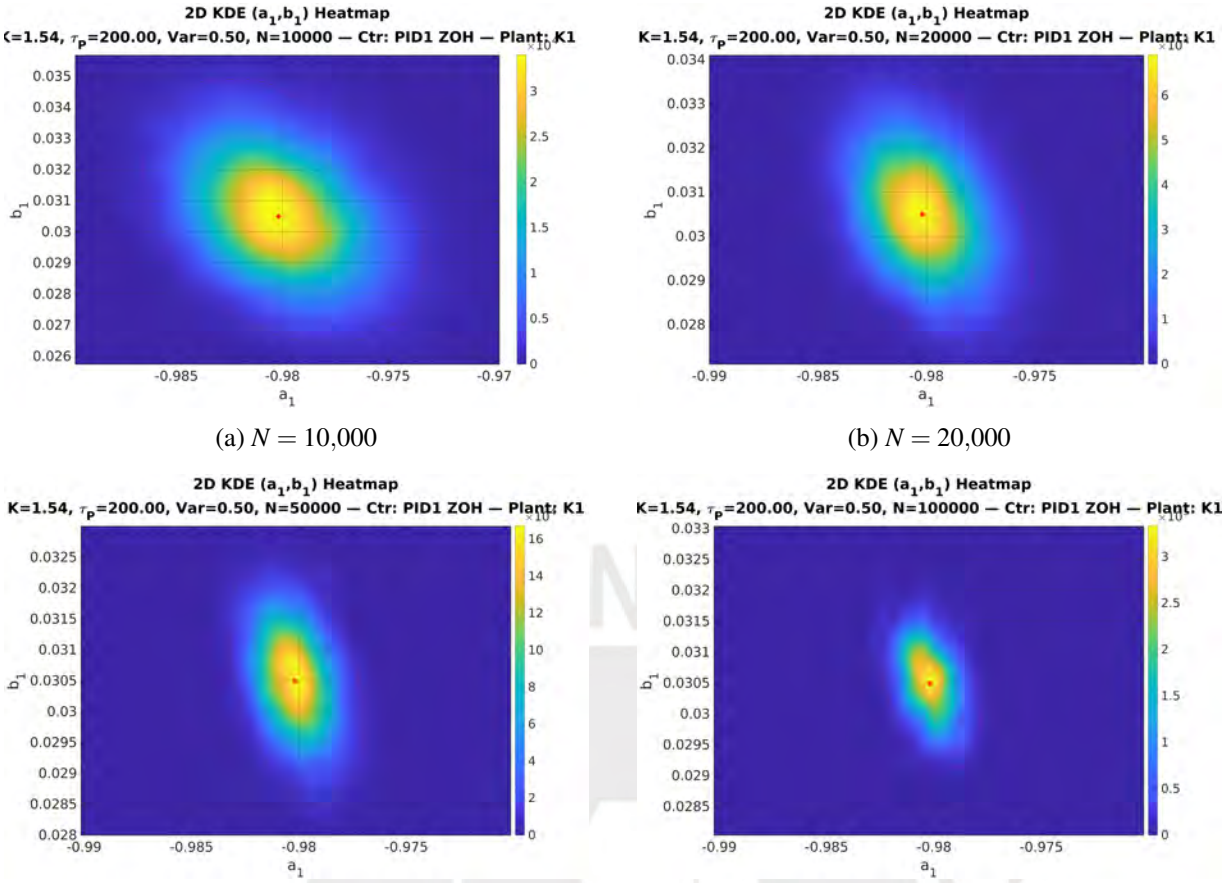
(a) $N = 500$

(b) $N = 1000$

(c) $N = 2000$

(d) $N = 5000$

Figure 4.31: 2D KDE of (a_1, b_1) estimates for plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $L = 20\,\text{s}$; controller PID1 (ZOH); noise variance $\sigma^2 = 0.5$. Panels: (a) $N = 500$, (b) $N = 1000$, (c) $N = 2000$, (d) $N = 5000$.



(a) $N = 10{,}000$ (b) $N = 20{,}000$

(c) $N = 50{,}000$ (d) $N = 100{,}000$

Figure 4.32: 2D KDE of (a_1, b_1) estimates for plant $K = 1.54$, $\tau_P = 200\,\text{s}$, $L = 20\,\text{s}$; controller PID1 (ZOH); noise variance $\sigma^2 = 0.5$. Panels: (a) $N = 10{,}000$, (b) $N = 20{,}000$, (c) $N = 50{,}000$, (d) $N = 100{,}000$.

Section 4.2: Overlay of empirical 95% KDE confidence region and SPS boundary

Figure 4.33 shows the complete dataset (output and input) for the closed-loop PID1 case with $K = 1.54$, $\tau_P = 200\,\text{s}$, and noise variance $\sigma^2 = 0.5$ used in the overlays below. In Figs. 4.34–4.39, we superimpose the SPS boundary on the empirical 95% KDE confidence region in the $(a_1, b_{1,1})$ plane for three data sizes ($N = 500$, $N = 1000$, and $N = 2000$) and two initializations (start@FULL and start@KDE). Across all panels, the SPS and KDE regions have very similar orientation and both include the true parameters (star marker), indicating agreement between the sampling-based SPS construction and the empirical distribution of estimates. Small portions of the KDE region extend beyond the SPS boundary (visible as KDE color outside the magenta contour), which reflects sampling variability and model/estimator nonlinearity. As N increases from 500 to 1000 and then to 2000, the joint confidence region tightens further, consistent with the asymptotic variance reduction of least-squares estimators. For the full set of overlays, see the folder “5. Overlay of SPS and KDE plots per configuration”.

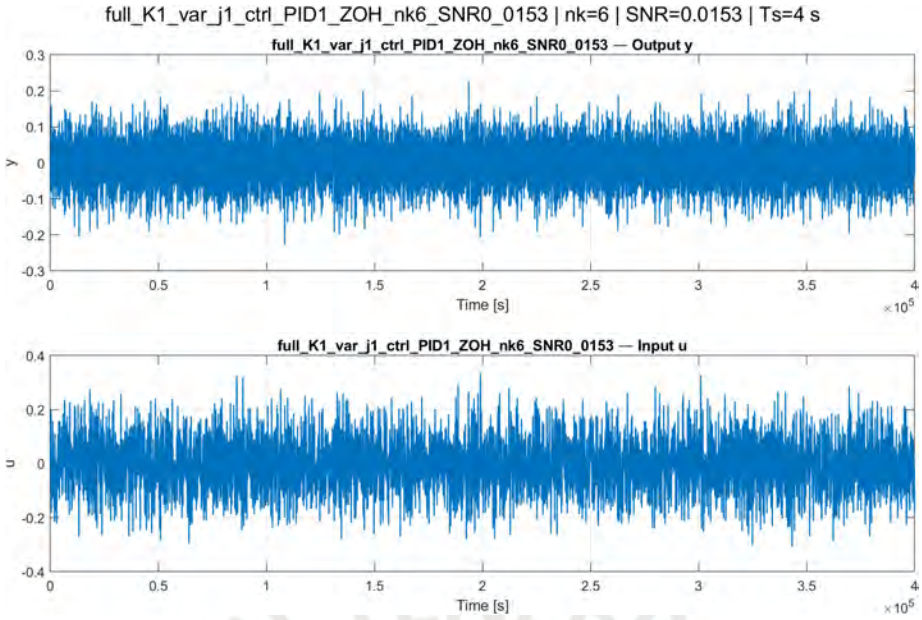


Figure 4.33: Complete dataset of one experiment with plant $K = 1.54$, noise variance 0.5, PID1, and $T_s = 4\,\mathrm{s}$.

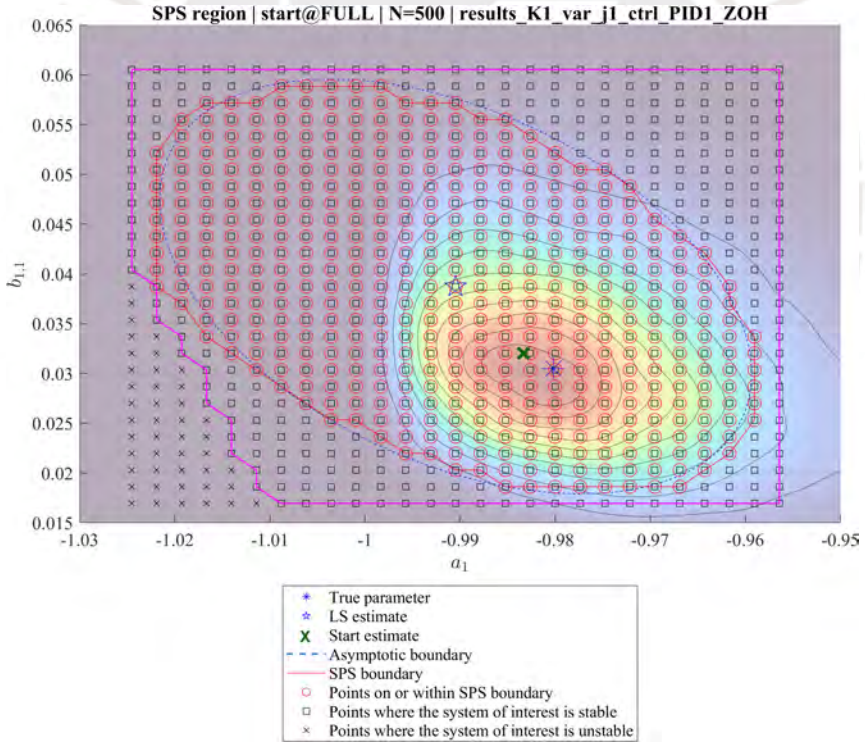


Figure 4.34: Overlay (start@FULL), $N = 500$: SPS boundary and empirical 95% KDE region in $(a_1, b_{1,1})$.

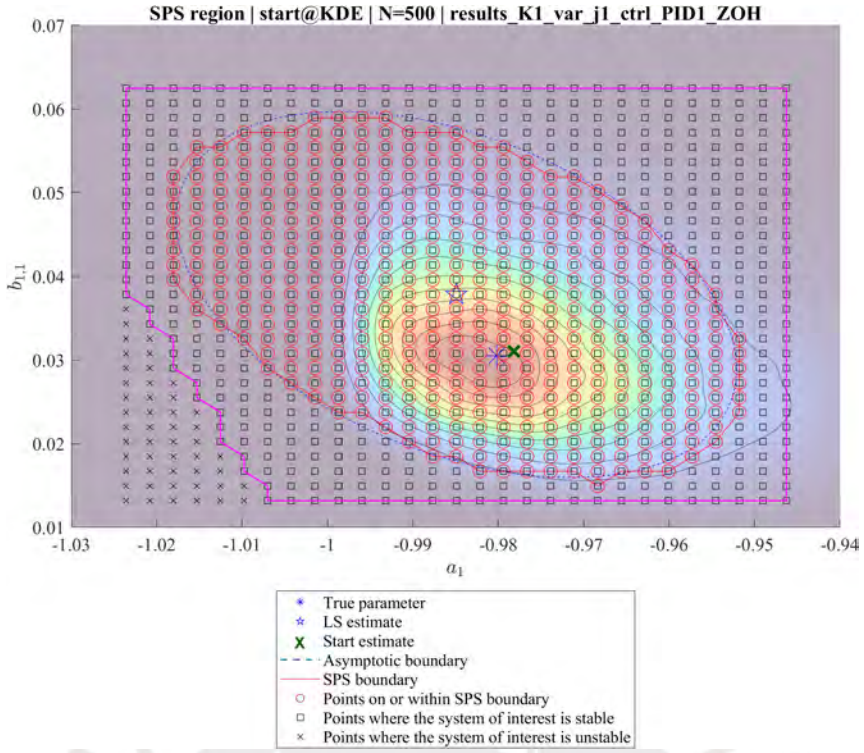


Figure 4.35: Overlay (start@KDE), $N = 500$: SPS boundary and empirical 95% KDE region in $(a_1, b_{1,1})$.

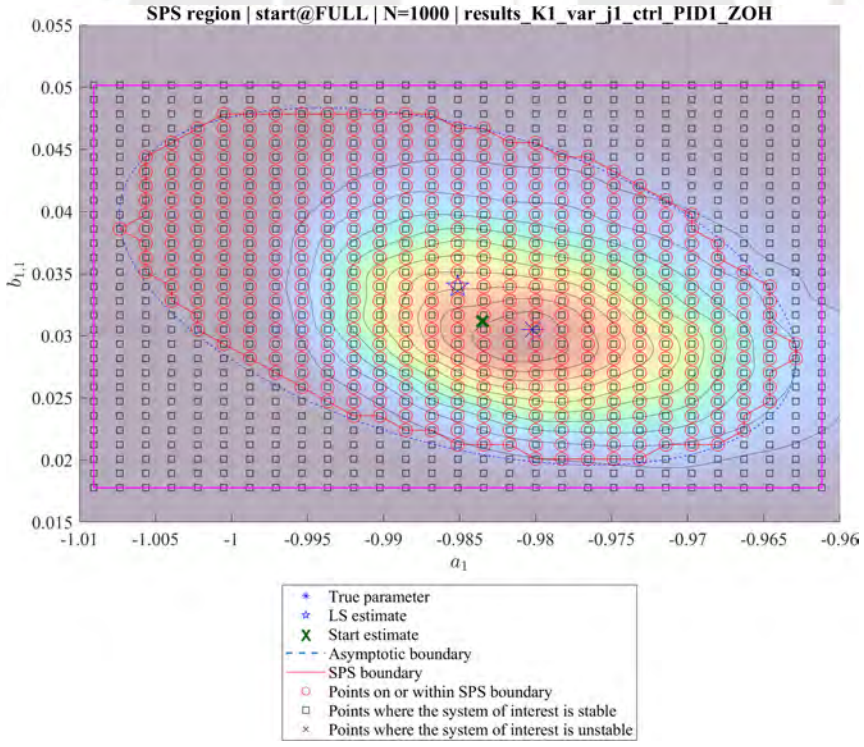


Figure 4.36: Overlay (start@FULL), $N = 1000$: SPS boundary and empirical 95% KDE region in $(a_1, b_{1,1})$.

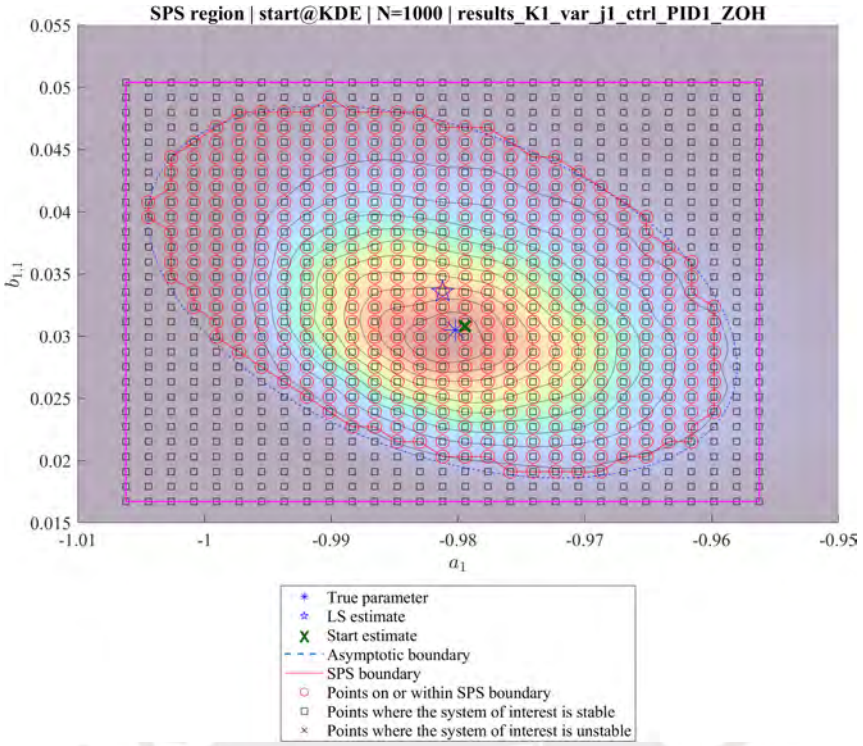


Figure 4.37: Overlay (start@KDE), $N = 1000$: SPS boundary and empirical 95% KDE region in $(a_1, b_{1,1})$.

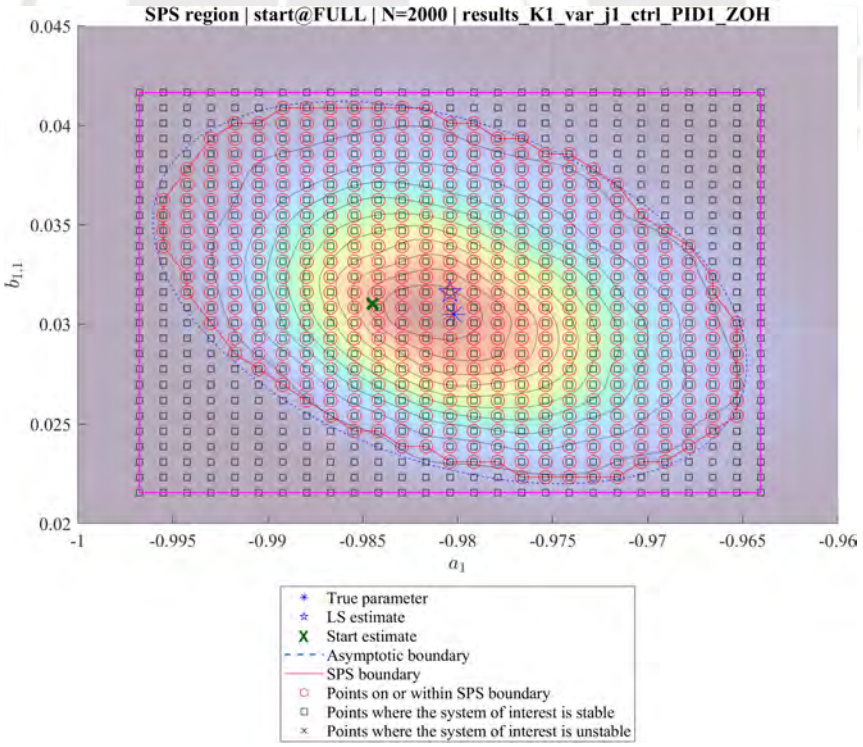


Figure 4.38: Overlay (start@FULL), $N = 2000$: SPS boundary and empirical 95% KDE region in $(a_1, b_{1,1})$.

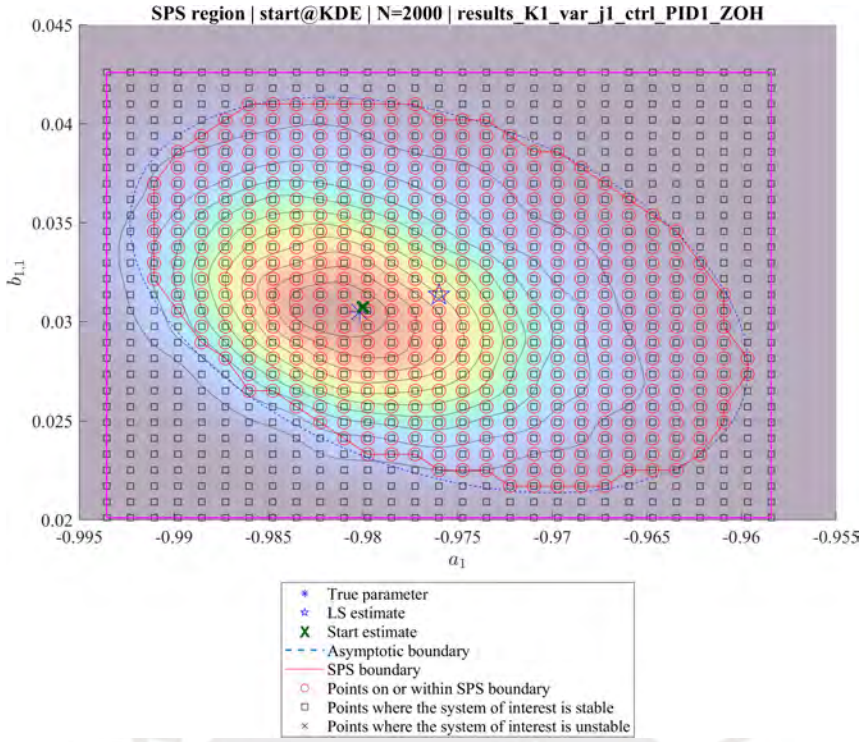


Figure 4.39: Overlay (start@KDE), $N = 2000$: SPS boundary and empirical 95% KDE region in $(a_1, b_{1,1})$.

Section 4.2.1: SNR Results

The signal-to-noise ratio (SNR) values obtained for $N = 100{,}000$ samples are summarized in Table 4.13 and shown in Figure 4.40. The results show that the SNR remains nearly constant across different noise variances, forming compact clusters for each plant-controller combination. This indicates that the impact of the additive noise level on the overall closed-loop SNR is minimal under the tested conditions. Instead, the SNR predominantly depends on the plant dynamics and the controller tuning, consistent with the theoretical expression in Eq. 2.28 presented in Chapter 2, where the SNR for routine operating data is determined by the closed-loop transfer functions G_p, G_c, and G_l.

As observed in Figure 4.40, the SNR values form well-separated groups according to the plant gain and time constant, while variations between controllers mainly reflect differences in loop bandwidth and disturbance attenuation capability. Overall, these results confirm that, in closed-loop operation, the achievable SNR is an inherent property of the plant–controller interaction rather than a direct function of the measurement noise variance.

Table 4.13: Configuraciones de simulación por planta, controlador y nivel de ruido.

Plant	τ_P	L	n_k	Var	Controlador	N	SNR
1	24	8	2	0.5	PI1_ZOH	100000	3.22
1	24	8	2	1	PI1_ZOH	100000	3.22
1	24	8	2	5	PI1_ZOH	100000	3.22
1	24	8	2	10	PI1_ZOH	100000	3.21
1	24	8	2	0.5	PID1_ZOH	100000	3.44
1	24	8	2	1	PID1_ZOH	100000	3.44
1	24	8	2	5	PID1_ZOH	100000	3.44
1	24	8	2	10	PID1_ZOH	100000	3.44
1.54	200	20	5	0.5	PI1_ZOH	100000	37.85
1.54	200	20	5	1	PI1_ZOH	100000	37.85
1.54	200	20	5	5	PI1_ZOH	100000	37.85
1.54	200	20	5	10	PI1_ZOH	100000	37.83
1.54	200	20	5	0.5	PID1_ZOH	100000	37.53
1.54	200	20	5	1	PID1_ZOH	100000	37.56
1.54	200	20	5	5	PID1_ZOH	100000	37.55
1.54	200	20	5	10	PID1_ZOH	100000	37.48
2	300	40	10	0.5	PI1_ZOH	100000	77.54
2	300	40	10	1	PI1_ZOH	100000	77.49
2	300	40	10	5	PI1_ZOH	100000	77.46
2	300	40	10	10	PI1_ZOH	100000	77.43
2	300	40	10	0.5	PID1_ZOH	100000	76.20
2	300	40	10	1	PID1_ZOH	100000	76.18
2	300	40	10	5	PID1_ZOH	100000	76.16
2	300	40	10	10	PID1_ZOH	100000	76.17

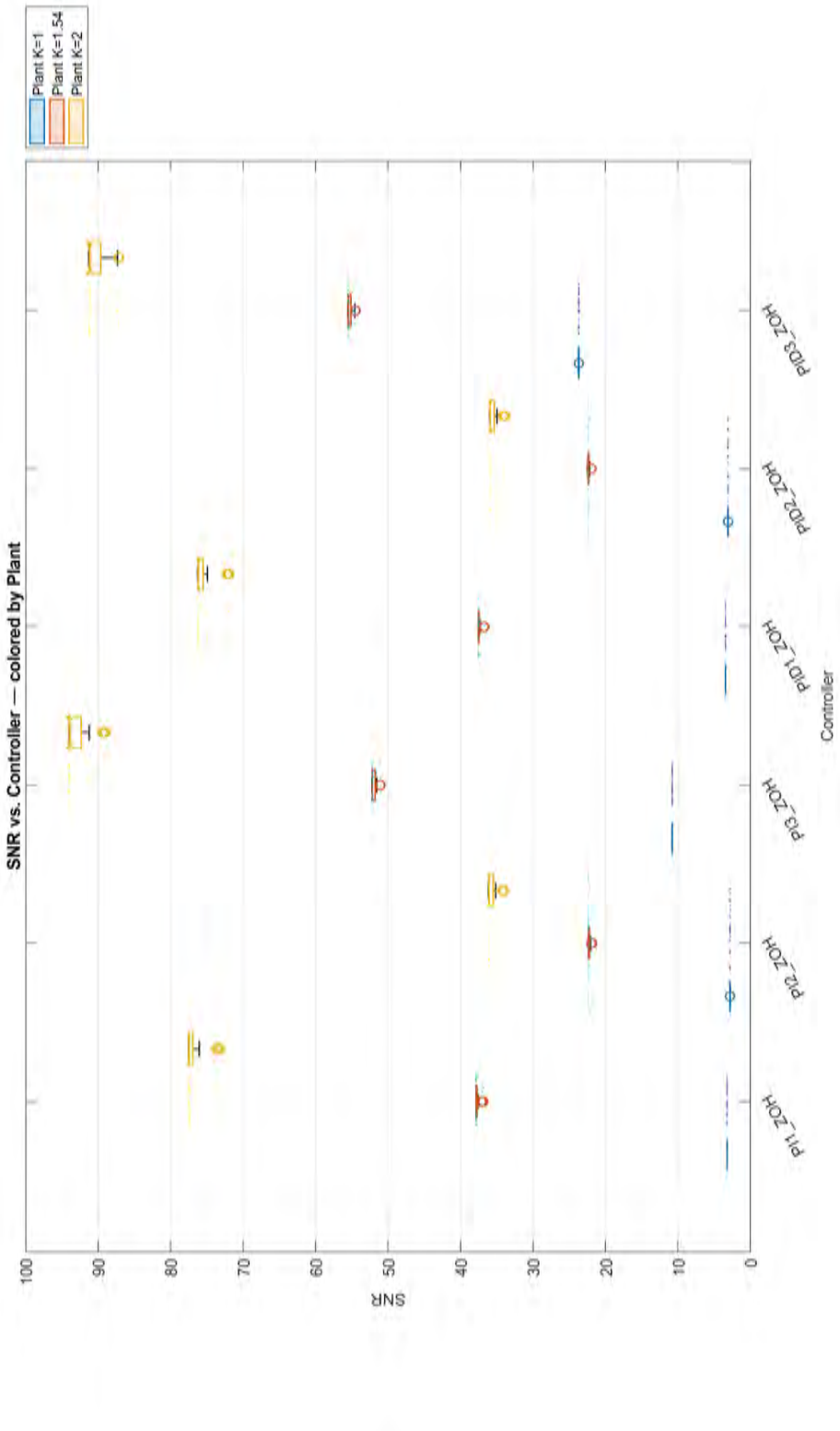


Figure 4.40: SNR vs. Controller by Plant (rotated for better visualization).

Chapter 5: Conclusions

In the closed-loop ARX system identification using routine operating data, the accuracy of the estimated parameters is strongly determined by the available data length N. For short datasets, the identified parameters $(K_{\mathrm{ARX}}, \tau_{P,\mathrm{ARX}})$ deviate noticeably from the true plant values, reflecting the limited excitation and higher influence of stochastic disturbances. As the number of data samples increases, these deviations progressively decrease. Once the data length reachs approximately $N = 5000$, both the mean and median estimates stabilize and converge toward the true values. The median estimates, in particular, provide a more robust representation of the central tendency: its errors fall below 5% for $N \geq 2000$ and below 2% for $N \geq 5000$.

The results show a clear transition in the shape of the estimation error distributions as N increases. At small data sizes, the distributions are visibly non-Gaussian and often skewed, but they gradually evolve toward an approximately Gaussian form in accordance with the central limit theorem. This behavior is supported by the Lilliefors normality test, which consistently rejects normality for $N \leq 1000$ but accepts it for $N \geq 5000$ for a1 and b1. However, the situation is different for the continuous-time parameters obtained from these coefficients: they do not follow necessarily a normal distribution, even when the discrete parameters (a_1, b_1) do. The logarithmic and ratio operations used in the discrete-to-continuous mapping introduce nonlinearity and distortion in the parameter space, leading to skewed and heavy-tailed distributions—especially at small N or for slow systems where a_1 approaches unity. This explains the asymmetric shapes and elongated tails observed in the estimated K_{ARX} and $\tau_{P,\mathrm{ARX}}$ distributions for small data sizes.

The nonlinear transformations required to compute

The minimum data length required for reliable estimation was also analyzed. The discrete ARX parameters begin to exhibit nearly normal distributions from $N = 5000$ onward, whereas the transformed continuous-time parameters only approach normality at $N \geq 20000$. For datasets smaller than $N = 50000$, none of the K_{ARX} or $\tau_{P,\mathrm{ARX}}$ estimates passed the Lilliefors normality test, indicating that classical parametric statistics relying on Gaussian assumptions cannot be applied safely at lower data sizes. Nevertheless, the median estimates remain close to the true parameters even for moderately small datasets. The comparison between the Sign-Perturbed Sums (SPS) confidence regions and the empirical 95% KDE regions shows that, for $N = 2000$, the SPS region already encompasses most of the KDE region, confirming consistency between both methods. Based on these findings, a minimum data size of about 2000 samples is recommended to limit estimation bias and achieve meaningful confidence regions under routine closed-loop operation.

The effect of noise variance on estimation accuracy was also examined. Varying the disturbance variance $\sigma_e^2 \in \{0.5, 1, 5, 10\}$ caused only minor shifts in the estimated parameters compared with the effect of the data length. This weak dependency arises from the closed-loop configuration: when the noise amplitude increases, the controller reacts proportionally, increasing the variance of the control input $u(t)$. Consequently, both the input and output variances scale with the disturbance level, leading to a nearly

constant signal-to-noise ratio (SNR) across noise conditions. The effective information content of the data is therefore governed primarily by N, not by the raw noise variance. Differences in the SNR observed in some simulations can be attributed instead to variations in controller parameterization, loop sensitivity, or disturbance filtering, rather than to the magnitude of the noise itself.

Finally, it should be noted that the rejection of extreme outliers in postprocessing further reduces the apparent influence of noise variance. In summary, for closed-loop identification under routine operating conditions and without external excitation, the data length N is the dominant factor determining estimator precision and bias, whereas the noise variance plays only a secondary role. Increasing N substantially improves both the accuracy and the normality of the estimated parameters, allowing reliable application of asymptotic and non-asymptotic confidence region methods such as SPS and KDE.

Bibliography

Bernhardsson, B., & Åström, K. J. (2016). *Control system design – pid control.* Lecture notes, Department of Automatic Control, Lund University. Retrieved from `https://www.control.lth.se/fileadmin/control/Education/DoctorateProgram/ControlSystemsSynthesis/2016/PIDControl.pdf`

Campi, M. C., Ko, S., & Weyer, E. (2009). Non-asymptotic confidence regions for model parameters in the presence of unmodelled dynamics. *Automatica*, *45*(10), 2175–2186.

Chen, Y.-C. (2017). A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, *1*(1), 161–187.

Csáji, B. C., Campi, M. C., & Weyer, E. (2012). Sign-perturbed sums (sps): A method for constructing exact finite-sample confidence regions for general linear systems. In *2012 ieee 51st ieee conference on decision and control (cdc)* (pp. 7321–7326).

Csáji, B. C., & Weyer, E. (2015). Closed-loop applicability of the sign-perturbed sums method. In *2015 54th ieee conference on decision and control (cdc)* (pp. 1441–1446).

Elamin, K. A. E. M. H. (2025). Particle filtering for enhanced parameter estimation in bilinear systems under colored noise. *arXiv preprint arXiv:2505.12041*.

Franklin, G. F., Powell, J. D., & Emami-Naeini, A. (2002). *Feedback control of dynamic systems* (Vol. 4). Prentice hall Upper Saddle River.

Kanakeri, V., & Mitra, A. (2025). Boosting-enabled robust system identification of partially observed lti systems under heavy-tailed noise. *arXiv preprint arXiv:2504.18444*.

Ljung, L. (1999). *System identification: Theory for the user*. Prentice Hall.

Maher, R. M. (2014). *The decibel scale* (Tech. Rep.). Montana State University. Retrieved from `https://www.montana.edu/rmaher/eele417_fl14/decibel_scale_eele417.pdf` (Accessed: 2025-09-28)

Ogata, K. (2010). *Modern control engineering*. Prentice hall.

Oshima, M., Kim, S., Shardt, Y. A., & Sotowa, K.-I. (2024). Experiment design taking nonasymptotic properties of the model into consideration. *IFAC-PapersOnLine*, *58*(15), 550–555.

Schouten, A. C. (2010, March). *Perturbation signal design.* Lecture notes, Delft University of Technology, Systems Identification & Parameter Estimation course. Retrieved from `https://ocw.tudelft.nl/wp-content/uploads/Lecture_4_-_Perturbation_Signal_Design.pdf` (Lecture 4, SIPE (System Identification & Parameter Estimation))

Shardt, Y. (2012). *Data quality assessment for closed-loop system identification and forecasting with application to soft sensors* (Ph.D. thesis). University of Alberta, Edmonton, Alberta, Canada.

Shardt, Y. A. (2022). *Statistics for chemical and process engineers*. Springer.

Shardt, Y. A. (2024). *At.215 automation engineering.* Lecture notes, TU Ilmenau. (Course material, AT.215)

Shardt, Y. A., & Huang, B. (2011a). Closed-loop identification condition for armax models using routine operating data. *Automatica*, *47*(7), 1534–1537.

Shardt, Y. A., & Huang, B. (2011b). Closed-loop identification with routine operating data: Effect of time delay and sampling time. *Journal of Process Control*, *21*(7), 997–1010.

Shardt, Y. A., Huang, B., & Ding, S. X. (2015). Minimal required excitation for closed-loop identifi-

cation: Some implications for data-driven, system identification. *Journal of Process Control*, *27*, 22–35.

Silverman, B. W. (2018). *Density estimation for statistics and data analysis*. Routledge.

Sun, Y.-H., Zeng, Y.-H., & Yang, Y.-G. (2024). Identification of hybrid energy harvesting systems with non-gaussian process. *Acta Mechanica Sinica*, *40*(2), 523154.

The MathWorks, Inc. (2024a). *ksdensity — estimate 1-d probability density function.* Retrieved from `https://www.mathworks.com/help/stats/ksdensity.html` (Accessed: 2025-07-06)

The MathWorks, Inc. (2024b). *mvksdensity — multivariate kernel density estimation.* Retrieved from `https://www.mathworks.com/help/stats/mvksdensity.html` (Accessed: 2025-07-06)

Yusuf, A. (2019). Model predictive control from routine plant data. *IFAC Journal of Systems and Control*, *8*, 100050.

Appendix A: Appendix

Section A.1: Programs

The programs in this section are here in a reduced version. To see the complete programs and results, must check the folder in the cloud-Ilmenau.

Section A.1.1: Initial program to generate the data of experiments (`FOR_LOOP_V27_HPC.m`): Closed-loop ARX experiments with Monte Carlo, `parfor` command, sliding windows, and ARX identification

This MATLAB program performs a large-scale Monte Carlo study of ARX model identification in closed-loop operation. Its aim is to assess how accurately ARX(1,1) models recover process dynamics from routine operating data. The simulation includes different plant dynamics, controller design, noise levels, and data length, and runs in parallel using `parfor`. The parallel pool is initialized robustly: the code requests up to 32 workers and automatically clamps to what the active profile, host core count, and SLURM settings allow.

Three FOPDT plants are considered, defined by $(K,T,L) \in \{(1.54,200,20),\ (2,300,40),\ (1,24,8)\}$. All models are discretized with a fixed sampling time of $T_s = 4\,\text{s}$. Closed-loop control is provided by six zero-order-hold (ZOH) controllers: three PID variants with a derivative filter and three PI variants. Four noise variances are tested $\{0.5,\ 1,\ 5,\ 10\}$, yielding $3 \times 6 \times 4 = 72$ unique configurations. Each configuration runs 1000 Monte Carlo trials with distinct seeds. For every configuration, the first full-length data set is saved once with detailed metadata (including SNR) for use in the construction of the confidence region based on SPS 95%.

Within each trial, data are generated by simulating the closed-loop response to an external disturbance only. To probe the effect of data length without repeatedly regenerating signals, the program uses an overlapping sliding-window scheme rather than fixed splits. Window sizes are $\{500, 1000, 2000, 5000, 10000, 20000, 50000\}$ samples with 50% overlap, capped at 20 windows per size; the full record is included in the analysis. This produces dozens of datasets per trial. For every window, the signal-to-noise ratio (SNR) is computed and stored.

Each window is identified with an ARX$(1,1,n_k)$ model using a fixed input delay $n_k = \text{round}(L/T_s) + 1$, derived from the plant delay. The relationship between the discrete-time ARX representation and the continuous-time first-order plus dead-time (FOPDT) model is given in Eq.(2.21). Using these mappings, the discrete-time parameters (a_1, b_1) are converted to physically interpretable quantities: the process gain K_{arx} and the process time constant $\tau_{P,\text{arx}}$. Estimates are screened for reliability and

rejected if non-finite, non-real, or outside admissible ranges

$$K_{\text{arx}} \in (-500, 500], \qquad \tau_{P,\text{arx}} \in (-1000, 1000].$$

Results (valid fits, counts, and metadata) are saved incrementally per configuration using an append/merge strategy, enabling long runs to resume without overwriting. Optional Excel exports (disabled by default to save memory) automatically split large tables across multiple files. After processing all configurations, the program aggregates results into summary tables. A compact table reports the mean and median of K_{arx} and $\tau_{P,\text{arx}}$ for each combination of plant, noise variance, window length, and controller. A long-form "tidy" table is also produced; grouped statistics (mean, median, and mode) are computed and written out for downstream analysis.

Finally, the program generates 3D surface and 2D plots showing how the mean and median of K_{arx} and $\tau_{P,\text{arx}}$ vary with data length and noise variance, for each plant–controller pair. In total, the simulation covers 72 configurations, each with 1000 trials and many sliding-window datasets per trial, resulting in millions of ARX fits. Thanks to systematic design, reproducible data handling, and efficient parallel execution, the program provides a scalable framework for studying closed-loop identification under routine operating conditions.

Section A.1.1.1: Matlab code: `FOR_LOOP_V27_HPC.m`

```
% FOR LOOP V27 WITH SLIDING WINDOW INSTEAD OF JUST SPLITTING THE MAIN
    DATASET
% Test with ARX and PARFOR & ZOH-only
% Short version for report
clear; clc; close all;

save_folder = 'FOR_LOOP_V27_INFO';
if ~exist(save_folder,'dir'); mkdir(save_folder); end
RUN_ID = datestr(now,'yyyymmdd_HHMMSS');
ANALYZE_INVALID = true;

APPROVAL = struct();
APPROVAL.K_bounds = [-500, 500];
APPROVAL.tauP_bounds = [-1000, 1000];
APPROVAL.require_finite = true;
APPROVAL.require_real = true;

WIN_SIZES = [500 1000 2000 5000 10000 20000 50000];
WIN_OVERLAP_SMALL = 0.10;
WIN_OVERLAP_MID = 0.80;
WIN_OVERLAP_50K = 0.90;
INCLUDE_FULL_WINDOW = true;
MAX_WINDOWS_PER_SIZE = 20;

K = [1.54 2 1];
L = [20 40 8];
T = [200 300 24];
Ts = 4;
```

```
s = tf('s');

for_noise_variance_SNR = [0.5 1 5 10];
for_numb_experiments_N = 1000;
for_amounts_data_Q = 100000 - 1;

Kp = 1; Ti = 70; Td = 1; N_cont = 10;
controllers_PID_ZOH = { pid(Kp, 1/Ti, Td, N_cont); pid(1, 1/150, 1,
   N_cont); pid(2, 1/70, 10, N_cont) };
controllers_PI_ZOH = { pid(Kp, 1/Ti); pid(1, 1/150); pid(2, 1/70) };
controllers = [controllers_PID_ZOH; controllers_PI_ZOH];
controller_labels = {'PID1','PID2','PID3','PI1','PI2','PI3'};
method = 'zoh';

try, c = parcluster("Processes"); catch, c = parcluster("local"); end
n = min([32, c.NumWorkers, feature('numcores')]);
p = gcp('nocreate'); if ~isempty(p) && p.NumWorkers~=n, delete(p); end
if isempty(gcp('nocreate')), parpool(c,n); end

Ntot = for_amounts_data_Q + 1;
counts = zeros(size(WIN_SIZES));
for ksz = 1:numel(WIN_SIZES)
    w = WIN_SIZES(ksz);
    if w <= Ntot
        if w == 50000, ov = WIN_OVERLAP_50K; elseif w > 5000 && w <
           50000, ov = WIN_OVERLAP_MID; else, ov = WIN_OVERLAP_SMALL;
           end
        step = max(1, floor(w*(1-ov)));
        counts(ksz) = numel(1:step:(Ntot - w + 1));
    end
end
est_windows = sum(min(counts, MAX_WINDOWS_PER_SIZE)) +
   double(INCLUDE_FULL_WINDOW);
est_total = est_windows * for_numb_experiments_N * numel(controllers) *
   numel(K) * numel(for_noise_variance_SNR);

for cidx = 1:numel(controllers)
    Gc_d = c2d(controllers{cidx}, Ts, method);
    [bc, ac] = tfdata(Gc_d, 'v');
    params = get(controllers{cidx});
    is_PID = isfield(params,'D') && params.D~=0;
    method_tag = ternary(is_PID,'ZOH_with_N','ZOH');
    ctrl_label = sprintf('%s_%s', controller_labels{cidx}, method_tag);

    for i = 1:length(K)
        Gp = K(i) * exp(-L(i)*s) / (T(i)*s + 1);
        Gp_d = c2d(Gp, Ts, 'zoh');
        p_disc = exp(-Ts/T(i));
        a1_arx_true = -p_disc;
        H_arx = tf(1,[1 a1_arx_true],Ts);
        CL_d = minreal(H_arx/(1+Gp_d*Gc_d));
        [bcl, acl] = tfdata(CL_d,'v');
        Gp_d_zpk = zpk(Gp_d); Gp_d_zpk.Variable = 'z^-1';
        beta1 = Gp_d_zpk.K;
        alpha1 = -pole(Gp_d_zpk);
```

```
if isprop(Gp_d_zpk,'OutputDelay') &&
   ~isempty(Gp_d_zpk.OutputDelay), nk_orig =
   abs(Gp_d_zpk.OutputDelay); else, nk_orig = NaN; end

t_end = for_amounts_data_Q * Ts;
t = (0:Ts:t_end)';
Ntot = numel(t);
idx_pairs = {};
labels = {};
if INCLUDE_FULL_WINDOW
    idx_pairs{end+1} = [1, Ntot];
    labels{end+1} = 'full';
end
for wlen = WIN_SIZES
    if wlen > Ntot, continue; end
    if wlen == 50000, ov = WIN_OVERLAP_50K; elseif wlen > 5000
       && wlen < 50000, ov = WIN_OVERLAP_MID; else, ov =
       WIN_OVERLAP_SMALL; end
    step = max(1, floor(wlen*(1-ov)));
    starts = 1:step:(Ntot - wlen + 1);
    if isfinite(MAX_WINDOWS_PER_SIZE) && numel(starts) >
       MAX_WINDOWS_PER_SIZE
        lin_idx = round(linspace(1, numel(starts),
           MAX_WINDOWS_PER_SIZE));
        starts = starts(lin_idx);
    end
    for kWin = 1:numel(starts)
        a = starts(kWin); b = a + wlen - 1;
        idx_pairs{end+1} = [a, b];
        labels{end+1} = sprintf('slide_%d_of_%d_N%d', kWin,
           numel(starts), wlen);
    end
end

maxFits = numel(idx_pairs);
nk_process = floor(L(i)/Ts);
nk_i = nk_process + 1;

for j = 1:length(for_noise_variance_SNR)
    nv = for_noise_variance_SNR(j);
    results_par = cell(1, for_numb_experiments_N);

    seed_base = (((cidx-1)*3 + (i-1))*4 + (j-1))*100000;
    seeds = uint32(1:for_numb_experiments_N) + uint32(1 +
       seed_base);

    parfor kexp = 1:for_numb_experiments_N
        rng(double(seeds(kexp)), 'twister');
        nvec = sqrt(nv) * randn(numel(t),1);
        y = filter(bcl, acl, nvec);
        u = filter(bc,  ac,  -y);
        cu = [0; cumsum(u.^2)];
        cn = [0; cumsum(nvec.^2)];
        segMean = @(cs,a,b) (cs(b+1)-cs(a)) / (b-a+1);
        tmpl = struct('experiment',[],'Gp_K',[],'Gp_tau',[], ...
```

```
        'Gp_L',[],'Var',[],'Seed',[],'N_data_sam',[],...
        'Controller',[],'Sampling_Time',[],'Data_label',[],...
        'alpha1',[],'beta1',[],'nk_orig',[],'K_arx_31',[],...
        'tauD_arx_27',[],'b1_arx',[],'a1_arx',[],'nk_arx',[],...
        'SNR',[],'tauP_arx_31',[]);
        local_results = repmat(tmpl, maxFits, 1);
        nValid = 0;
        for w = 1:numel(idx_pairs)
            idxw = idx_pairs{w}; a = idxw(1); b = idxw(2);
            segN = b - a + 1; if segN < (nk_i + 5), continue;
               end
            numP = segMean(cu,a,b); denP = segMean(cn,a,b);
            if isfinite(numP) && isfinite(denP) && denP > 0,
               SNR_val = numP/denP; else, SNR_val = NaN; end
            data = iddata(y(a:b), u(a:b), Ts);
            arx_model = arx(data, [1 1 nk_i]);
            b1_arx = arx_model.B(end);
            a1_arx = arx_model.A(end);
            nk_arx = arx_model.nk;
            if a1_arx < 0, tauP_arx_31 = -Ts/log(-a1_arx);
               else, tauP_arx_31 = NaN; end
            K_arx_31 = b1_arx/(1+a1_arx);
            okfinite = isfinite(K_arx_31) &&
               isfinite(tauP_arx_31) && isreal(K_arx_31) &&
               isreal(tauP_arx_31);
            inbounds = okfinite &&
               K_arx_31>=APPROVAL.K_bounds(1) &&
               K_arx_31<=APPROVAL.K_bounds(2) &&
               tauP_arx_31>=APPROVAL.tauP_bounds(1) &&
               tauP_arx_31<=APPROVAL.tauP_bounds(2);
            if ~inbounds, continue; end
            nValid = nValid + 1;
            r = tmpl;
            r.experiment = (kexp-1)*numel(idx_pairs) + w;
            r.Gp_K = K(i); r.Gp_tau = T(i); r.Gp_L = L(i);
            r.Var = nv; r.Seed = double(seeds(kexp));
            r.N_data_sam = segN; r.Controller = ctrl_label;
            r.Sampling_Time = Ts; r.Data_label = labels{w};
            r.alpha1 = alpha1; r.beta1 = beta1; r.nk_orig =
               nk_orig;
            r.K_arx_31 = K_arx_31; r.tauD_arx_27 =
               (nk_arx-1)*Ts;
            r.b1_arx = b1_arx; r.a1_arx = a1_arx; r.nk_arx =
               nk_arx;
            r.SNR = SNR_val; r.tauP_arx_31 = tauP_arx_31;
            local_results(nValid) = r;
        end
        if nValid>0, results_par{kexp} =
           local_results(1:nValid); else, results_par{kexp} =
           []; end
    end

    nonempty_results =
       results_par(~cellfun('isempty',results_par));
```

```
            if isempty(nonempty_results), all_results_var = struct([]);
               else, all_results_var = vertcat(nonempty_results{:}); end
            mat_file = fullfile(save_folder,
               sprintf('results_K%d_var_j%d_ctrl_%s_%s.mat', i, j,
               controller_labels{cidx}, method_tag));
            total_estimates = numel(all_results_var);
            if ANALYZE_INVALID
                total_invalid = 0; total_NaN = 0; total_K = 0;
                   total_tauP = 0; total_K_only = 0; total_tauP_only =
                   0; total_both = 0;
            else
                total_invalid = 0; total_NaN = 0; total_K = 0;
                   total_tauP = 0; total_K_only = 0; total_tauP_only =
                   0; total_both = 0;
            end
               save(mat_file,'all_results_var','total_invalid',...
            'total_estimates','total_NaN','total_K','total_tauP',...
            'total_K_only','total_tauP_only',...
            'total_both','APPROVAL','RUN_ID','-v7.3');
        end
    end
end

files = dir(fullfile(save_folder, 'results_K*_var_j*_ctrl_*.mat'));
MainSummary = table();
for f = 1:numel(files)
    S = load(fullfile(save_folder, files(f).name));
    if ~isfield(S,'all_results_var') || isempty(S.all_results_var),
       continue; end
    R = S.all_results_var; T = struct2table(R);
    if isempty(T), continue; end
    Kmask = isfinite(T.K_arx_31) & isreal(T.K_arx_31) &
       T.K_arx_31>=APPROVAL.K_bounds(1) &
       T.K_arx_31<=APPROVAL.K_bounds(2);
    Tmask = isfinite(T.tauP_arx_31) & isreal(T.tauP_arx_31) &
       T.tauP_arx_31>=APPROVAL.tauP_bounds(1) &
       T.tauP_arx_31<=APPROVAL.tauP_bounds(2);
    validMask = Kmask & Tmask;
    T = T(validMask,:);
    if isempty(T), continue; end
    T2 = T(:,
       {'Gp_K','Gp_tau','Gp_L','Var','Controller','N_data_sam','SNR',...
    'K_arx_31','tauP_arx_31','a1_arx','b1_arx'});
    T2.Properties.VariableNames =
       {'PlantK','PlantTau','PlantL','Var','Controller','N','SNR',...
    'K','tauP','a1','b1'};
    G = groupsummary(T2,
       {'PlantK','PlantTau','PlantL','Var','Controller','N'},
       {'numel','mean','std','median'}, {'K','tauP','a1','b1'});
    G.Properties.VariableNames =
       {'PlantK','PlantTau','PlantL','Var','Controller','N',...
    'GroupCount','mean_K','mean_tauP','mean_a1','mean_b1','std_K',...
    'std_tauP','std_a1','std_b1','median_K','median_tauP',...
    'median_a1','median_b1'};
    if any(strcmp('SNR', T2.Properties.VariableNames))
```

```
        GS = groupsummary(T2,
            {'PlantK','PlantTau','PlantL','Var','Controller','N'},
            'mean', 'SNR');
        G = innerjoin(G, GS(:,[1:6 7]), 'Keys',
            {'PlantK','PlantTau','PlantL','Var','Controller','N'});
        G.Properties.VariableNames(end) = {'SNR'};
    else
        G.SNR = NaN(height(G),1);
    end
    MainSummary = [MainSummary; G];
end

MainSummary = sortrows(MainSummary,
   {'PlantK','PlantTau','PlantL','Controller','Var','N'});
save(fullfile(save_folder,'MainSummary.mat'), 'MainSummary','APPROVAL');
writetable(MainSummary, fullfile(save_folder,'MainSummary.xlsx'));
```

Section A.1.2: Program for analysis of Monte Carlo and Nested FOR Loop results (`analyze_arx_results_V7.m`): Processing of ARX results for histograms, KDE, and confidence intervals and regions

The program implements a fully automated, parallel post-processing pipeline for Monte Carlo ARX identification results (sampling time $T_s = 4$ s) stored in files matching `results_K*_var_j*_ctrl_*.mat`. It executes headless with unified, print-friendly graphics defaults to ensure legible high-DPI figures suitable for six-per-page layouts, and writes all logs and outputs under `FOR_LOOP_V27_INFO/Analysis ARX`. During ingestion, it scans all result files, performs basic sanity checks (finite, real estimates), and consolidates valid entries into a table that records plant parameters (K, τ_P), the noise variance, the data length N, and controller labels robustly parsed from heterogeneous naming conventions. A plant map provides the nominal plants and, together with T_s, defines consistent analysis limits (±50% around the nominal values) for (K, τ_P) and for the implied discrete ARX(1,1) parameters $a_1 = -e^{-T_s/\tau_P}$ and $b_1 = K(1 - e^{-T_s/\tau_P})$, which are used both for validation and for plant-anchored plotting ranges.

Building on these metadata, the pipeline enumerates all unique combinations $(K, \tau_P, \text{Var}, N, \text{Controller}$ and, for each, produces a coordinated family of histograms and density estimates. One-dimensional histograms for K_{ARX} and $\tau_{P,\text{ARX}}$ use the Freedman–Diaconis rule (bounded between 25 and 100 bins), apply data-aware x-axis zooming so that the empirical 95% central interval is visible, and overlay a Normal fit scaled to counts, together with vertical markers for the median, the empirical 95% interval, and the nominal plant value. The same strategy is replicated after converting estimates to (a_1, b_1), thereby offering parallel views in continuous (K, τ_P) and discrete (a_1, b_1) parameterizations. In two dimensions, the program renders bivariate histograms for $(K_{\text{ARX}}, \tau_{P,\text{ARX}})$ and for (a_1, b_1) using plant-fixed bin edges to make empty regions explicit, exports both "bar3" and smooth surface visualizations, and adds a top-view heatmap. In all 2D/3D views, the nominal plant is indicated by a red star and a red dotted contour delineates the smallest threshold that encloses 95% of the joint histogram mass, providing a visually consistent notion of joint coverage across figures.

To complement histograms with smooth estimates, the program computes one- and two-dimensional kernel density estimates. For each group, 1D KDEs of K_{ARX} and $\tau_{P,\text{ARX}}$ (and, separately, of a_1 and b_1) highlight the KDE mode, the sample median, the empirical 95% band, and the nominal value.

Two-dimensional KDEs over both $(K_{\mathrm{ARX}}, \tau_{P,\mathrm{ARX}})$ and (a_1, b_1) are evaluated on plant-anchored grids with MATLAB's normal-approximation bandwidth, and a data-aware zoom (inner quantile box with padding) ensures the principal density region remains fully visible while respecting hard plant limits. Surfaces, filled contours, and heatmaps are exported consistently, each overlaying the 95% highest-density contour and the plant marker, thereby linking smoothly to the histogram-based joint region notion used earlier.

Beyond visualization, the program compiles controller-aware summary statistics that enable inference and reporting. Grouped summaries provide means, minima, and maxima for K_{ARX} and $\tau_{P,\mathrm{ARX}}$ (optionally $\tau_{D,\mathrm{ARX}}$ when available), KDE-based descriptors including the mean, the KDE mode, the trimmed median within the empirical 95% band, and the empirical 95% interval with its length, and distributional diagnostics using Kolmogorov–Smirnov and Lilliefors tests aggregated into a normality flag. Classical analytical intervals are also computed per group: Student–t confidence intervals for the mean and chi-square intervals for the variance, as well as an exact order-statistic 95% interval for the median. To study the effect of data length, the program generates "median versus N" panels on a logarithmic x-axis for each plant, noise level, controller, and parameter; these panels are provided both with analytical 95% median intervals as shaded bands and, for compact overviews, in a no-CI variant, always including the nominal value as a horizontal reference.

All computationally intensive loops, including plotting and density estimation, are parallelized with a process-based pool so that workers can create and save figures independently while inheriting the global graphics defaults for typographic consistency. The pipeline is robust to missing or small groups, gracefully skipping plots or tests when sample thresholds are not met, and includes a fallback loader for Excel chunk files if `.mat` results are unavailable. The outputs comprise a structured directory of high-resolution PNG figures (1D histograms, 3D bar and surface histograms, heatmaps, 1D/2D KDEs, and median-versus-N panels) for both (K, τ_P) and (a_1, b_1), together with `.mat` files that store the unique analysis sets and all summary tables, thereby supporting reproducibility and downstream inclusion into the written report.

Section A.1.3: Matlab code: `analyze_arx_results_V9.m`

```
% Short version for report
clear; clc;

folder = 'FOR_LOOP_V27_INFO';
use_timestamped_run  = false;
wipe_existing_outdir = true;

base_outdir = fullfile(folder, 'Analysis ARX_V2');
if use_timestamped_run
    outdir = fullfile(base_outdir, datestr(now,'yyyymmdd_HHMMSS'));
    [ok,msg] = mkdir(outdir); if ~ok, error(msg); end
else
    outdir = base_outdir;
    if wipe_existing_outdir && exist(outdir,'dir'), try
        rmdir(outdir,'s'); catch, end, end
    if ~exist(outdir,'dir'), [ok,msg] = mkdir(outdir); if ~ok,
        error(msg); end, end
end
```

```
results_folder = folder;
data_dir  = outdir;

result_files = dir(fullfile(results_folder,
   'results_K*_var_j*_ctrl_*.mat'));
selected_results = {};
for k = 1:numel(result_files)
    S = load(fullfile(results_folder, result_files(k).name));
    cand = fieldnames(S); res = [];
    for c = 1:numel(cand)
        val = S.(cand{c});
        if isstruct(val) && all(isfield(val, {'Gp_K','Gp_tau'})), res =
           val; break; end
    end
    if isempty(res), continue; end
    for idx = 1:numel(res)
        r = res(idx);
        if isfield(r,'K_arx_31') && isfield(r,'tauP_arx_31')
            if isreal(r.K_arx_31)&&isfinite(r.K_arx_31)&&...
            isreal(r.tauP_arx_31)&&isfinite(r.tauP_arx_31)
                selected_results{end+1} = r; %#ok<AGROW>
            end
        end
    end
end
if isempty(selected_results), error('No valid ARX results found.'); end
selected_results = [selected_results{:}];

T = struct2table(selected_results);
keepVars = intersect({'Gp_K','Gp_tau','Var','N_data_sam','K_arx_31',...
'tauP_arx_31','tauD_arx_27','Ctrl_Label','Controller',...
'Controller_Label','ctrl_label'}, T.Properties.VariableNames, 'stable');
T = T(:, keepVars);

ctrlCol = '';
for nm = string(T.Properties.VariableNames)
    if contains(lower(nm),'ctrl') || contains(lower(nm),'controller')
        ctrlCol = char(nm); break
    end
end

unique_plants     = unique(T{:,{'Gp_K','Gp_tau'}}, 'rows');
unique_noise_vars = unique(T.Var);
unique_data_sizes = unique(T.N_data_sam);
save(fullfile(data_dir,'unique_plants.mat'),'unique_plants');
save(fullfile(data_dir,'unique_noise_vars.mat'),'unique_noise_vars');
save(fullfile(data_dir,'unique_data_sizes.mat'),'unique_data_sizes');

keys_group = {'Gp_K','Gp_tau','Var','N_data_sam'};
if ~isempty(ctrlCol), keys_group{end+1} = ctrlCol; end
combos = unique(T(:, keys_group), 'rows');
nC = height(combos);

Ts = 4;
```

```
makeKey = @(K,tauP) sprintf('K%.6f_tau%.6f', K, tauP);
PLIMS = containers.Map;
for i = 1:size(unique_plants,1)
    Kp = unique_plants(i,1); Taup = unique_plants(i,2);
    lam = exp(-Ts/Taup);
    a1 = -lam; b1 = Kp*(1 - lam);
    limK   = [0.5*Kp,   1.5*Kp];
    limTau = [0.5*Taup, 1.5*Taup];
    limA1  = [max(-1, a1 - 0.5*abs(a1)), min(0, a1 + 0.5*abs(a1))];
    limB1  = [max(0, 0.5*b1), 1.5*b1];
    PLIMS(makeKey(Kp,Taup)) =
        struct('K',Kp,'tauP',Taup,'a1',a1,'b1',b1,'limK',limK,...
    'limTau',limTau,'limA1',limA1,'limB1',limB1);
end

hasTauD = ismember('tauD_arx_27', T.Properties.VariableNames);
colsToSumm = {'K_arx_31','tauP_arx_31'}; if hasTauD, colsToSumm{end+1}
   = 'tauD_arx_27'; end
groupKeys = {'Gp_K','Gp_tau','Var','N_data_sam'};
Gparts = cell(1, numel(colsToSumm));
for k = 1:numel(colsToSumm)
    col = colsToSumm{k};
    Gk = grpstats(T(:, [groupKeys, {col}]), groupKeys,
        {'mean','min','max'}, 'DataVars', col);
    vn = Gk.Properties.VariableNames;
    Gk.Properties.VariableNames{strcmp(vn, ['mean_' col])} = ['Mean_'
        col];
    Gk.Properties.VariableNames{strcmp(vn, ['min_'  col])} = ['Min_'
        col];
    Gk.Properties.VariableNames{strcmp(vn, ['max_'  col])} = ['Max_'
        col];
    Gk = Gk(:, [groupKeys, {['Mean_' col], ['Min_' col], ['Max_'
        col]}]);
    Gparts{k} = Gk;
end
G1 = Gparts{1};
for k = 2:numel(Gparts), G1 = outerjoin(G1, Gparts{k}, 'Keys',
   groupKeys, 'MergeKeys', true); end
save(fullfile(data_dir,'summary_results_by_noise_fast.mat'),...
'G1','-v7.3');
summary_table_simple = G1;
save(fullfile(data_dir,'summary_kde_modes.mat'),'summary_table_simple');

rows = cell(nC, 2);
Tc = parallel.pool.Constant(T);
Cc = parallel.pool.Constant(combos);
parfor c = 1:nC
    Tloc = Tc.Value; comb = Cc.Value;
    Kval = comb.Gp_K(c); tauPval = comb.Gp_tau(c); Varval =
        comb.Var(c); Nval = comb.N_data_sam(c);
    ctrlVal = ""; if ~isempty(ctrlCol) && ismember(ctrlCol,
        comb.Properties.VariableNames), ctrlVal =
        string(comb.(ctrlCol)(c)); end
    mask = Tloc.Gp_K==Kval & Tloc.Gp_tau==tauPval & Tloc.Var==Varval &
        Tloc.N_data_sam==Nval;
```

```
    if ctrlVal ~= "", mask = mask & strcmp(string(Tloc.(ctrlCol)),
        ctrlVal); end
    if ~any(mask), continue; end
    valsCell = {Tloc.K_arx_31(mask), Tloc.tauP_arx_31(mask)};
    labels   = {'K␣(ARX)','tauP␣(ARX)'};
    for which = 1:2
        vals = valsCell{which}; vals = vals(isfinite(vals));
        if isempty(vals), continue; end
        mean_val   = mean(vals);
        median_val = median(vals);
        if numel(unique(vals)) > 1
            xgrid = linspace(min(vals), max(vals), 1000);
            [f_kde, xi_kde] = ksdensity(vals, xgrid);
            [~,mx] = max(f_kde);
            mode_val = xi_kde(mx);
        else
            mode_val = vals(1);
        end
        s = sort(vals); n = numel(s);
        loIdx = max(1, ceil(0.025*n)); hiIdx = min(n, floor(0.975*n));
        lo = s(loIdx); hi = s(hiIdx);
        med_trim = median(s(loIdx:hiIdx));
        ctrlStr = string(ctrlVal); if ctrlStr==""; ctrlStr="ALL"; end
        rows{c, which} = {Kval, tauPval, Varval, Nval, ctrlStr,
           labels{which}, mean_val, mode_val, lo, hi, hi-lo,
           median_val, med_trim};
    end
end
rows = rows(:); rows = rows(~cellfun('isempty', rows));
kdeVarNames =
   {'Plant_K','Plant_tauP','Noise_Var','Data_Size','Controller',...
'Parameter','Mean','Mode_KDE','CI_Lower_95_kde_based',...
'CI_Upper_95_kde_based','Range_Length','Median','Median_Trimmed_95'};
if isempty(rows)
    KDE_SUM = cell2table(cell(0, numel(kdeVarNames)), 'VariableNames',
       kdeVarNames);
else
    KDE_SUM = cell2table(vertcat(rows{:}), 'VariableNames',
       kdeVarNames);
end
save(fullfile(data_dir, 'summary_kde_confidence_intervals.mat'),
   'KDE_SUM');

rowsN = cell(nC, 2);
parfor c = 1:nC
    Tloc = Tc.Value; comb = Cc.Value;
    Kval = comb.Gp_K(c); tauPval = comb.Gp_tau(c); Varval =
       comb.Var(c); Nval = comb.N_data_sam(c);
    ctrlVal = ""; if ~isempty(ctrlCol) && ismember(ctrlCol,
       comb.Properties.VariableNames), ctrlVal =
       string(comb.(ctrlCol)(c)); end
    mask = Tloc.Gp_K==Kval & Tloc.Gp_tau==tauPval & Tloc.Var==Varval &
       Tloc.N_data_sam==Nval;
    if ctrlVal ~= "", mask = mask & strcmp(string(Tloc.(ctrlCol)),
       ctrlVal); end
```

```
    if nnz(mask) < 20, continue; end
    valsCell = {Tloc.K_arx_31(mask), Tloc.tauP_arx_31(mask)};
    labels   = {'K␣(ARX)','tauP␣(ARX)'};
    for j = 1:2
        v = valsCell{j}; v = v(isfinite(v));
        if numel(v) < 20, continue; end
        mu = mean(v); sg = std(v);
        if sg == 0
            pks  = 1; plil = 1;
        else
            [~, pks]  = kstest( (v - mu) / max(eps, sg) );
            [~, plil] = lillietest(v);
        end
        normality = (pks > 0.05 && plil > 0.05) * "YES" + ~(pks > 0.05
           && plil > 0.05) * "NO";
        ctrlStr = string(ctrlVal); if ctrlStr==""; ctrlStr="ALL"; end
        rowsN{c, j} = {Kval, tauPval, Varval, Nval, ctrlStr, labels{j},
           pks, plil, normality};
    end
end
rowsN = rowsN(:); rowsN = rowsN(~cellfun('isempty', rowsN));
normVarNames =
   {'Plant_K','Plant_tauP','Noise_Var','Data_Size','Controller',...
'Parameter','KS_p_Value','Lilliefors_p_Value','Normal_Distribution'};
if isempty(rowsN)
    NORMAL_SUM = cell2table(cell(0, numel(normVarNames)),
       'VariableNames', normVarNames);
else
    NORMAL_SUM = cell2table(vertcat(rowsN{:}), 'VariableNames',
       normVarNames);
end
save(fullfile(data_dir,'summary_normality_table.mat'),'NORMAL_SUM');

alpha = 0.05;
rowsM = cell(nC, 2);
parfor c = 1:nC
    Tloc = Tc.Value; comb = Cc.Value;
    Kval = comb.Gp_K(c); tauPval= comb.Gp_tau(c); Varval = comb.Var(c);
       Nval = comb.N_data_sam(c);
    ctrlVal = ""; if ~isempty(ctrlCol) && ismember(ctrlCol,
       comb.Properties.VariableNames), ctrlVal =
       string(comb.(ctrlCol)(c)); end
    mask = Tloc.Gp_K==Kval & Tloc.Gp_tau==tauPval & Tloc.Var==Varval &
       Tloc.N_data_sam==Nval;
    if ctrlVal ~= "", mask = mask & strcmp(string(Tloc.(ctrlCol)),
       ctrlVal); end
    valsCell = {Tloc.K_arx_31(mask), Tloc.tauP_arx_31(mask)};
    labels   = {'K␣(ARX)','tauP␣(ARX)'};
    for j = 1:2
        v = valsCell{j}; v = v(isfinite(v)); n = numel(v);
        if n < 2, continue; end
        v = sort(v);
        Kb = binoinv(alpha/2, n, 0.5);
        loIdx = max(1, Kb+1); hiIdx = min(n, n-Kb); if loIdx > hiIdx,
           continue; end
```

```
        median_val = median(v); lo = v(loIdx); hi = v(hiIdx);
        ctrlStr = string(ctrlVal); if ctrlStr==""; ctrlStr="ALL"; end
        rowsM{c, j} = {Kval, tauPval, Varval, Nval, ctrlStr, labels{j},
            median_val, lo, hi};
    end
end
rowsM = rowsM(:); rowsM = rowsM(~cellfun('isempty', rowsM));
medVarNames =
   {'Plant_K','Plant_tauP','Noise_Var','Data_Size','Controller',...
'Parameter','Median','Median_CI_Lower_95','Median_CI_Upper_95'};
if isempty(rowsM)
    MEDIAN_SUM = cell2table(cell(0, numel(medVarNames)),
        'VariableNames', medVarNames);
else
    MEDIAN_SUM = cell2table(vertcat(rowsM{:}), 'VariableNames',
        medVarNames);
end
save(fullfile(data_dir,'summary_median_conf_int.mat'),'MEDIAN_SUM');

paramsCI  = {'K_arx_31','tauP_arx_31','tauD_arx_27'};
paramsCI  = paramsCI(ismember(paramsCI, T.Properties.VariableNames));
groupKeys = {'Gp_K','Gp_tau','Var','N_data_sam'};
if ~isempty(ctrlCol), groupKeys{end+1} = ctrlCol; end
CI_rows = {};
for pidx = 1:numel(paramsCI)
    col = paramsCI{pidx};
    G = grpstats(T(:, [groupKeys, {col}]), groupKeys,
        {'mean','var','numel'}, 'DataVars', col);
    vn = G.Properties.VariableNames;
    iMean = strcmp(vn, ['mean_' col]); iVar  = strcmp(vn, ['var_'
        col]); iN = strcmp(vn, ['numel_' col]);
    if any(iMean), G.Properties.VariableNames{iMean} = 'Mean'; end
    if any(iVar),  G.Properties.VariableNames{iVar}  = 'VarStat'; end
    if any(iN),    G.Properties.VariableNames{iN}    = 'N'; end
    keep = [groupKeys, {'N','Mean','VarStat'}];
    G = G(:, keep);
    G = G(G.N>=2 & isfinite(G.Mean) & isfinite(G.VarStat), :);
    conf = 0.95; beta = 1 - conf;
    tcrit  = tinv(1 - beta/2, G.N - 1);
    chi_lo = chi2inv(beta/2,   G.N - 1);
    chi_hi = chi2inv(1-beta/2, G.N - 1);
    se = sqrt(G.VarStat ./ G.N);
    CImean_lo = G.Mean - tcrit .* se;
    CImean_hi = G.Mean + tcrit .* se;
    CIvar_lo  = (G.N - 1).*G.VarStat ./ chi_hi;
    CIvar_hi  = (G.N - 1).*G.VarStat ./ chi_lo;
    if ~isempty(ctrlCol) && ismember(ctrlCol,
        G.Properties.VariableNames)
        CtrlVec = string(G.(ctrlCol));
    else
        CtrlVec = repmat("ALL", height(G), 1);
    end
    tag = repmat(string(col), height(G), 1);
```

```
    CI_rows{end+1} = table(G.Gp_K, G.Gp_tau, G.Var, G.N_data_sam,
       CtrlVec, tag, G.Mean, CImean_lo, CImean_hi, G.VarStat, CIvar_lo,
       CIvar_hi, ...
        'VariableNames',
           {'Plant_K','Plant_tauP','Noise_Var','Data_Size',...
        'Controller','Parameter','Mean','CI_Lower','CI_Upper',...
        'Variance','Var_CI_Lower','Var_CI_Upper'});
end
if isempty(CI_rows)
    CI_SUM = cell2table(cell(0,12), 'VariableNames',
       {'Plant_K','Plant_tauP','Noise_Var','Data_Size','Controller',...
    'Parameter','Mean','CI_Lower','CI_Upper','Variance',...
    'Var_CI_Lower','Var_CI_Upper'});
else
    CI_SUM = vertcat(CI_rows{:});
end
save(fullfile(data_dir,'confidence_intervals_arx.mat'),'CI_SUM');
```

Section A.1.4: Program for SPS analysis: `Final_Overlay_V3.m`

The program compares kernel–density summaries from previously computed Monte Carlo results with exact finite-sample SPS confidence regions for a fixed, long data length ($N = 10{,}000$) in a closed-loop ARX(1,1) setting. It targets Plant 1 with $K = 1.54$, $\tau_P = 200$ s, a transport delay $L = 20$ s (hence $n_k \approx 5$ at $T_s = 4$ s), and uses the ZOH-discretized true parameters $a_1^\star = -e^{-T_s/\tau_P}$ and $b_1^\star = K(1-e^{-T_s/\tau_P})$ as a reference. After locating the most recent Monte Carlo result file `results_K1_var_j1_ctrl_PID1_ZOH*` and a matching "full" time-series dataset `full_K1_var_j1_ctrl_PID1_ZOH*`, it extracts the ARX estimates (a_1, b_1), filters them to $N = 10{,}000$, and computes the sample medians. In parallel, it loads the full dataset and performs a sliding-window ARX(1,1) estimation with window size 10,000 and 25% overlap across the trajectory, from which it obtains another pair of medians. These two median pairs ("start@KDE" from the results table and "start@FULL" from the sliding windows) serve as alternative initialization points for SPS.

With the experimental context fixed to closed loop, no deliberate excitation, and white measurement noise (PID controller with $K_p = 1$, $T_i = 70$ s, $T_d = 0$, $N = 10$; noise standard deviation $\sqrt{0.5}$), the script constructs an SPS region for each start point. For reproducibility, it generates $m = 800$ sign sequences with target coverage 0.95 (thus $q = \lfloor m(1-0.95) \rfloor$) and uses the "Normal" SPS mode with a 2D region output. For each start, it first simulates data under a start-dependent "simulation plant" (only for data generation) while keeping the plot's "true parameter" at the real plant $(a_1^\star, b_1^\star)$. It then computes the least-squares center, forms the SPS perturbations, and evaluates the region via `calc_ConfRegion`. The resulting depiction includes the SPS boundary, interior/on-boundary points, stable and unstable sets, the asymptotic boundary for reference, the LS estimate, the chosen start estimate (as a dark-green "X"), and the true parameter (as a blue star), with a standardized legend layout.

To place the SPS region in the context of the Monte Carlo distribution, the program overlays a 2D KDE of the (a_1, b_1) sample behind the SPS figure, using a normal-approximation bandwidth and the current SPS axes limits so the density "backdrop" neither clips nor distorts the region. In addition, it maps both the SPS boundary and the stable set from (a_1, b_1) to physical parameters (K, τ_P) via $K = b_1/(1+a_1)$ and $\tau_P = -T_s/\log(-a_1)$ (with domain checks to avoid singularities), and draws a companion figure in

the (K, τ_P) plane. There as well, it overlays a KDE constructed from the Monte Carlo-derived (K, τ_P) values to compare the highest-density area with the mapped SPS boundary; a tight polygonal boundary of the mapped region is also sketched to aid visual interpretation. The delay index n_k is parsed from the dataset name when available and otherwise defaults to L/T_s; a one-step offset is applied to match the ARX structure used in estimation.

The output consists of four high-resolution PNG figures saved to `fig/`: two overlays in (a_1, b_1) and two in (K, τ_P), each pair produced once from the "start@FULL" medians and once from the "start@KDE" medians. Alongside the figures, the console log prints the true discrete parameters and both sets of start medians. The workflow depends on in-path routines for data generation, least-squares estimation, ARX parameter/transfer-function conversions, and SPS region construction (with CasADi required by the SPS engine). In sum, the script provides a controlled, apples-to-apples comparison between distributional evidence from Monte Carlo (via KDE) and exact finite-sample uncertainty sets from SPS, viewed in both the discrete parameter space (a_1, b_1) and the physically interpretable (K, τ_P) space, with sensitivity illustrated by the two plausible initialization points.

Section A.1.4.1: Matlab code: `Final_Overlay_V3.m`

```
% Data size 500. The data size was changed according to the determined
    tests
%Short version for report

clear; clc;

dt = 4; Ts = dt;
T_all = 500;
sigma_noise = sqrt(0.5);

results_file = 'results_K1_var_j1_ctrl_PID1_ZOH.mat';
full_file    = 'full_K1_var_j1_ctrl_PID1_ZOH.mat';

K_true = 1.54; TauP_true = 200; L_true = 20; Ts_true = 4;
a1_true_d = -exp(-Ts_true/TauP_true);
b1_true_d = K_true * (1 - exp(-Ts_true/TauP_true));

Sres = load(results_file);
cand_vars = {'all_results_var','R','Results','results_table'};
R = []; for c = 1:numel(cand_vars), if isfield(Sres,cand_vars{c}), R =
    Sres.(cand_vars{c}); break; end, end
if istable(R), R = table2struct(R); end
a1_all = [R.a1_arx]; b1_all = [R.b1_arx];

Nvec = []; fns = {'N_data_sam','N','data_size','T','len'};
for k=1:numel(fns), if isfield(R, fns{k}), Nvec = [R.(fns{k})]; break;
    end, end
if isempty(Nvec), selN = true(size(a1_all)); else, selN =
    (Nvec(:)==500); end

a1 = a1_all(selN); b1 = b1_all(selN);
ok = isfinite(a1) & isfinite(b1); a1 = a1(ok); b1 = b1(ok);
```

```
start_median_a1_KDE = median(a1,'omitnan');
start_median_b1_KDE = median(b1,'omitnan');

Sfull = load(full_file);
cand_t = {'t','time','Time','T'}; cand_y = {'y','Y','output','Y_t'};
    cand_u = {'u','U','input','U_t'}; cand_Ts = {'Ts','dt','TS','ts'};
t = []; y = []; u = [];
for i=1:numel(cand_t), if isfield(Sfull,cand_t{i}), t =
    Sfull.(cand_t{i}); break; end, end
for i=1:numel(cand_y), if isfield(Sfull,cand_y{i}), y =
    Sfull.(cand_y{i}); break; end, end
for i=1:numel(cand_u), if isfield(Sfull,cand_u{i}), u =
    Sfull.(cand_u{i}); break; end, end
for i=1:numel(cand_Ts), if isfield(Sfull,cand_Ts{i}), Ts =
    Sfull.(cand_Ts{i}); break; end, end
t = t(:); y = y(:); u = u(:);

nk_true = round(L_true/Ts_true);
nk = nk_true - 1;

win = 500; overlap = 0.25; step = max(1, floor(win*(1-overlap)));
Nsig = numel(y); last_start = Nsig - win + 1; w_starts =
    1:step:last_start;
if w_starts(end) ~= last_start, w_starts(end+1) = last_start; end
w_starts = unique(w_starts);

a1_win = nan(numel(w_starts),1); b1_win = nan(numel(w_starts),1);
for k = 1:numel(w_starts)
    i1 = w_starts(k); i2 = i1 + win - 1;
    D = iddata(y(i1:i2), u(i1:i2), Ts);
    try
        sys = arx(D, [1 1 nk]);
        a1_win(k) = sys.A(2);
        b1_win(k) = sys.B(end);
    catch
    end
end
start_median_a1_full = median(a1_win,'omitnan');
start_median_b1_full = median(b1_win,'omitnan');

fprintf('Start␣medians␣(N=500)\nKDE␣:␣a1=%.6g␣␣b1=%.6g\nFULL:␣a1=%.6g␣␣
    b1=%.6g\n', ...
    start_median_a1_KDE, start_median_b1_KDE, start_median_a1_full,
        start_median_b1_full);

Plant = "K1"; Controller = "PID1"; Discretiz = "ZOH"; DataSize = 500;
    NoiseVar = sigma_noise^2; True_a1 = a1_true_d; True_b1 = b1_true_d;
rows = {
    'start@FULL', start_median_a1_full, start_median_b1_full;
    'start@KDE' , start_median_a1_KDE , start_median_b1_KDE
};
Tstarts = cell2table(rows, 'VariableNames',
    {'StartTag','Start_a1','Start_b1'});
```

```
Tinfo = table(Plant, Controller, Discretiz, DataSize, NoiseVar,
   True_a1, True_b1);
Tout = [repmat(Tinfo, height(Tstarts), 1) Tstarts];

xlsx_out = fullfile(pwd, 'starts_500.xlsx');
if exist(xlsx_out,'file')
    Tprev = readtable(xlsx_out);
    Tout = [Tprev; Tout];
end
writetable(Tout, xlsx_out);
disp(Tout);
```

Section A.2: Rejected experiments

This annex summarizes the number of discarded identification results for each simulation configuration. Rejected fits correspond to ARX estimates that failed numerical convergence or produced non-finite, non-real, or physically implausible parameter values. Tables A.1-A.4 report the total number of identification attempts, the number of failed fits, the accepted ones, and the corresponding discard percentage for each controller and noise level. A complete rejection summary for all configurations is available in the file `RejectSummary_by_config.xlsx`, located in the folder `3. Rejected Experiments`.

Table A.1: Discard summary (Plant $K = 1.54$, $\tau_P = 200$, $L = 20$; noise variance 0.5)

Controller	Var	Total Fits	Total Failed	Accepted Fits	%Discarded
PI1	0,5	132000	1210	130790	0,92%
PI2	0,5	132000	2379	129621	1,80%
PI3	0,5	132000	2946	129054	2,23%
PID1	0,5	132000	1319	130681	1,00%
PID2	0,5	132000	2520	129480	1,91%
PID3	0,5	132000	3452	128548	2,62%

Table A.2: Discard summary (Plant $K = 1.54$, $\tau_P = 200$, $L = 20$; noise variance 10)

Controller	Var	Total Fits	Total Failed	Accepted Fits	%Discarded
PI1	10	132000	1219	130781	0,92%
PI2	10	132000	2514	129486	1,90%
PI3	10	132000	3095	128905	2,34%
PID1	10	132000	1295	130705	0,98%
PID2	10	132000	2488	129512	1,88%
PID3	10	132000	3401	128599	2,58%

Table A.3: Discard summary (Plant $K = 2$, $\tau = 300$, $L = 40$; noise variance 0,5)

ControllerBase	Var	TotalFits	TotalFailed	AcceptedFits	%Discarded
PI1	0,5	132000	1508	130492	1,14%
PI2	0,5	132000	4070	127930	3,08%
PI3	0,5	132000	4347	127653	3,29%
PID1	0,5	132000	1574	130426	1,19%
PID2	0,5	132000	4046	127954	3,07%
PID3	0,5	132000	5029	126971	3,81%

Table A.4: Discard summary (Plant $K = 2$, $\tau = 300$, $L = 40$; noise variance 10)

ControllerBase	Var	TotalFits	TotalFailed	AcceptedFits	%Discarded
PI1	10	132000	1414	130586	1,07%
PI2	10	132000	4015	127985	3,04%
PI3	10	132000	4203	127797	3,18%
PID1	10	132000	1597	130403	1,21%
PID2	10	132000	4092	127908	3,10%
PID3	10	132000	4964	127036	3,76%

Note: The complete summary table for the results can be found in the folder with the name "RejectSummary_by_config.xlsx" in the folder "3. Rejected Experiments".

Section A.3: Results - More tables

Tables A.5, A.8, and A.10 summarize the median estimates and biases of the ARX model parameters obtained under controller PID1. Each table corresponds to a different process dynamics: the plant with $\tau_P = 24$ (Table A.5) is the fastest, the plant with $\tau_P = 200$ (Table A.8) represents an intermediate dynamics, and the plant with $\tau_P = 300$ (Table A.10) is the slowest.

In general, the results show that the bias decreases as the data length N increases. The fastest plant exhibits the smallest biases for small datasets, since its dynamics contain more information per unit time. On the other hand, the slowest plant tends to display larger biases, particularly for small N. These trends appear in both the gain parameter K and the time constant τ_P, confirming that identification accuracy improves with longer datasets and when the underlying process dynamics are faster.

Tables A.8–A.10 summarise the median ARX estimates, percentage bias, data size N, and empirical SNR for the three plant configurations. For the fastest plant ($K = 1,\ \tau_P = 24,\ L = 8$), the estimator converges very quickly: already at $N = 2000$ the gain and time-constant biases fall below 0.5%, and for larger datasets the deviations are essentially negligible. The SNR remains roughly constant across N (around 3.4 for Var = 0.5 and 3.44 for Var = 10).

Table A.5: Median ARX estimates and biases for controller PID1.

K	τ_P	L	Var	N	med(K)	Bias$_K$ [%]	σ_K	medn(τ_P)	Bias$_{\tau_P}$ [%]	σ_{τ_P}
1	24	8	0.5	500	0.994	0.61	0.3113	23.6994	1.27	5.0681
1	24	8	0.5	1000	0.995	0.50	0.2184	23.8229	0.74	3.5512
1	24	8	0.5	2000	0.997	0.31	0.1555	23.9250	0.31	2.5408
1	24	8	0.5	5000	0.998	0.19	0.0979	23.9643	0.15	1.6238
1	24	8	0.5	10000	0.999	0.07	0.0688	23.9757	0.10	1.1481
1	24	8	0.5	20000	1.000	-0.02	0.0494	23.9973	0.01	0.8182
1	24	8	0.5	50000	0.999	0.06	0.0312	23.9977	0.01	0.5158
1	24	8	0.5	100000	0.999	0.11	0.0215	23.9751	0.10	0.3702
1	24	8	10	500	0.997	0.32	0.3065	23.8518	0.62	5.0335
1	24	8	10	1000	0.999	0.08	0.2164	23.8748	0.52	3.5392
1	24	8	10	2000	0.999	0.06	0.1555	23.9754	0.10	2.5665
1	24	8	10	5000	1.000	0.00	0.0984	23.9971	0.01	1.6144
1	24	8	10	10000	1.000	-0.01	0.0694	24.0120	-0.05	1.1467
1	24	8	10	20000	1.001	-0.08	0.0496	24.0186	-0.08	0.8200
1	24	8	10	50000	1.001	-0.07	0.0312	24.0237	-0.10	0.5193
1	24	8	10	100000	1.000	-0.03	0.0217	24.0215	-0.09	0.3777

Table A.6: Median ARX estimates and biases for Plant ($K = 1.54,\ \tau_P = 200,\ L = 20$).

K	τ_P	L	Var	N	med(K)	Bias$_K$ [%]	σ_K	med(τ_P)	Bias$_{\tau_P}$ [%]	σ_{τ_P}
1.54	200	20	0.5	500	1.429	7.21%	0.8986	180.3361	10.90%	96.7003
1.54	200	20	0.5	1000	1.504	2.31%	0.7397	192.7905	3.74%	81.2760
1.54	200	20	0.5	2000	1.537	0.17%	0.5594	198.1865	0.92%	60.7909
1.54	200	20	0.5	5000	1.538	0.14%	0.3617	199.0102	0.50%	39.2359
1.54	200	20	0.5	10000	1.542	-0.13%	0.2562	199.7302	0.14%	28.2129
1.54	200	20	0.5	20000	1.544	-0.24%	0.1818	199.9736	0.01%	20.0380
1.54	200	20	0.5	50000	1.543	-0.20%	0.1169	199.9992	0.00%	12.9319
1.54	200	20	0.5	100000	1.542	-0.15%	0.0867	200.0089	-0.04%	9.4279
1.54	200	20	10	500	1.423	7.59%	0.9000	179.6699	11.30%	95.7320
1.54	200	20	10	1000	1.503	2.43%	0.7312	192.1733	4.07%	79.7333
1.54	200	20	10	2000	1.530	0.68%	0.5583	196.4247	1.82%	60.8420
1.54	200	20	10	5000	1.536	0.23%	0.3629	198.5846	0.71%	39.6140
1.54	200	20	10	10000	1.537	0.19%	0.2575	199.0465	0.48%	28.2868
1.54	200	20	10	20000	1.539	0.04%	0.2183	199.4934	0.25%	19.9159
1.54	200	20	10	50000	1.539	0.06%	0.1146	199.7556	0.12%	12.6227
1.54	200	20	10	100000	1.540	0.01%	0.0809	199.8967	0.05%	8.7717

Table A.7: Median ARX estimates and biases for Plant ($K = 2,\ \tau_P = 300,\ L = 40$).

K	τ_P	L	Var	N	med(K)	Bias$_K$ [%]	σ_K	med(τ_P)	Bias$_{\tau_P}$ [%]	σ_{τ_P}
2	300	40	0.5	500	1.752	12.39%	1.0110	254.5828	17.84%	136.1261
2	300	40	0.5	1000	1.887	5.66%	0.8404	279.7228	7.25%	117.1135
2	300	40	0.5	2000	1.957	2.13%	0.6568	290.4287	3.30%	90.7789
2	300	40	0.5	5000	1.985	0.76%	0.4320	296.4285	1.20%	59.9155
2	300	40	0.5	10000	1.992	0.39%	0.3061	298.1538	0.62%	42.5578
2	300	40	0.5	20000	1.999	0.07%	0.2138	299.1296	0.29%	29.9299
2	300	40	0.5	50000	2.002	-0.12%	0.1354	300.1973	-0.07%	19.0066
2	300	40	0.5	100000	2.002	-0.09%	0.1002	300.6862	-0.23%	13.5568
2	300	40	10	500	1.739	13.05%	1.0065	254.1550	18.04%	135.4563
2	300	40	10	1000	1.887	5.66%	0.6577	280.1160	7.10%	118.8241
2	300	40	10	2000	1.954	2.29%	0.6516	290.5726	3.24%	90.0892
2	300	40	10	5000	1.984	0.79%	0.4273	296.6360	1.13%	59.5290
2	300	40	10	10000	1.989	0.54%	0.3046	298.1668	0.61%	42.3133
2	300	40	10	20000	2.001	-0.04%	0.2180	300.1366	-0.05%	30.1988
2	300	40	10	50000	2.004	-0.19%	0.1358	300.1790	-0.06%	18.6420
2	300	40	10	100000	2.003	-0.14%	0.0987	300.4481	-0.15%	13.6142

The baseline plant ($K = 1.54,\ \tau_P = 200,\ L = 20$) displays noticeably larger finite-sample bias for small N: at $N = 500$, the median gain and time-constant biases exceed 7% and 10%, respectively. However, both quantities fall below 2% once $N \geq 2000$. As in the fastest plant, the SNR is essentially constant across N (about 37), so the reduction in bias is again attributable to increased data length rather than a change in noise level.

The slowest plant ($K = 2,\ \tau_P = 300,\ L = 40$) constitutes the most demanding scenario. For $N = 500$, the gain and time-constant biases reach 12.4% and 17.8%, and convergence is clearly slower than for the first two plants. Even so, the estimates improve steadily with N: at $N = 2000$, the biases drop to 2.1% (gain) and 3.3% (time constant), and fall below 1% once $N \geq 5000$. Although this plant has a significantly larger time constant and delay, its SNR is much higher (around 72–76), yet almost independent of N. The SNR values are approximately 76.2 for $N > 1000$.

Overall, the results consistently show that longer datasets substantially reduce estimator bias, while plants with larger time constants and delays require much larger N to achieve the same level of accuracy. The empirical SNR remains effectively constant as N increases, meaning that improvements in estimation performance stem from averaging over more samples rather than changes in signal-to-noise ratio.

Table A.8: Median ARX estimates, biases, data size N, and SNR for Plant ($K = 1,\ \tau_P = 24,\ L = 8$).

K	τ_P	L	Var	N	SNR	med(K)	Bias$_K$ [%]	med(τ_P)	Bias$_{\tau_P}$ [%]
1	24	8	0.5	500	3.4368	0.994	0.61%	23.6994	1.27%
1	24	8	0.5	1000	3.4359	0.995	0.50%	23.8229	0.74%
1	24	8	0.5	2000	3.4381	0.997	0.31%	23.9250	0.31%
1	24	8	0.5	5000	3.4383	0.998	0.19%	23.9643	0.15%
1	24	8	0.5	10000	3.4390	0.999	0.07%	23.9757	0.10%
1	24	8	0.5	20000	3.4393	1.000	-0.02%	23.9973	0.01%
1	24	8	0.5	50000	3.4390	0.999	0.06%	23.9977	0.10%
1	24	8	0.5	100000	3.4390	0.999	0.11%	23.9751	0.10%
1	24	8	10	500	3.4437	0.997	0.32%	23.8518	0.62%
1	24	8	10	1000	3.4427	0.999	0.08%	23.8748	0.52%
1	24	8	10	2000	3.4436	0.999	0.06%	23.9754	0.10%
1	24	8	10	5000	3.4419	1.000	0.00%	23.9971	0.01%
1	24	8	10	10000	3.4424	1.000	-0.01%	24.0102	-0.05%
1	24	8	10	20000	3.4424	1.001	-0.08%	24.0186	-0.01%
1	24	8	10	50000	3.4426	1.001	-0.07%	24.0237	-0.10%
1	24	8	10	100000	3.4422	1.000	-0.03%	24.0215	-0.09%

Table A.9: Median ARX estimates, biases, data size N, and SNR for Plant ($K = 1.54,\ \tau_P = 200,\ L = 20$).

K	τ_P	L	Var	N	SNR	med(K)	Bias$_K$ [%]	med(τ_P)	Bias$_{\tau_P}$ [%]
1.54	200	20	0.5	500	36.6843	1.429	7.21%	180.3361	10.90%
1.54	200	20	0.5	1000	37.4867	1.504	2.31%	192.7905	3.74%
1.54	200	20	0.5	2000	37.5741	1.537	0.17%	198.1865	0.92%
1.54	200	20	0.5	5000	37.5349	1.538	0.14%	199.0128	0.50%
1.54	200	20	0.5	10000	37.5355	1.542	-0.13%	199.7302	0.14%
1.54	200	20	0.5	20000	37.5363	1.544	-0.24%	199.9736	0.01%
1.54	200	20	0.5	50000	37.5523	1.543	-0.20%	199.9992	-0.00%
1.54	200	20	0.5	100000	37.5316	1.542	-0.15%	200.0889	-0.04%
1.54	200	20	10	500	36.7019	1.423	7.59%	179.6969	11.30%
1.54	200	20	10	1000	37.3008	1.503	2.43%	192.1733	4.07%
1.54	200	20	10	2000	37.3825	1.536	0.68%	196.4247	1.82%
1.54	200	20	10	5000	37.4728	1.537	0.23%	198.5846	0.71%
1.54	200	20	10	10000	37.4818	1.539	0.19%	199.0465	0.48%
1.54	200	20	10	20000	37.4855	1.539	0.04%	199.4934	0.25%
1.54	200	20	10	50000	37.4909	1.539	0.06%	199.7556	0.12%
1.54	200	20	10	100000	37.4767	1.540	0.01%	199.8967	0.05%

Table A.10: Median ARX estimates, biases, data size N, and SNR for Plant ($K = 2,\ \tau_P = 300,\ L = 40$).

K	τ_P	L	Var	N	SNR	med(K)	Bias$_K$ [%]	med(τ_P)	Bias$_{\tau_P}$ [%]
2	300	40	0.5	500	71.9030	1.752	12.39%	254.5828	17.84%
2	300	40	0.5	1000	75.0500	1.887	5.66%	279.7228	7.25%
2	300	40	0.5	2000	76.0377	1.957	2.13%	290.4287	3.30%
2	300	40	0.5	5000	76.2292	1.985	0.76%	296.4285	1.20%
2	300	40	0.5	10000	76.2227	1.992	0.39%	298.1538	0.62%
2	300	40	0.5	20000	76.1935	2.002	-0.07%	299.1296	0.43%
2	300	40	0.5	50000	76.2008	2.002	-0.12%	300.1793	-0.07%
2	300	40	0.5	100000	76.2005	2.002	-0.09%	300.6862	-0.23%
2	300	40	10	500	72.1322	1.739	13.05%	254.1550	18.04%
2	300	40	10	1000	75.1561	1.887	5.66%	280.1160	7.10%
2	300	40	10	2000	76.0731	1.954	2.29%	290.5726	3.34%
2	300	40	10	5000	76.1744	1.984	0.79%	296.6360	1.31%
2	300	40	10	10000	76.1713	1.989	0.54%	298.1668	0.61%
2	300	40	10	20000	76.2176	2.001	-0.04%	300.1336	-0.05%
2	300	40	10	50000	76.2365	2.004	-0.19%	300.1790	-0.06%
2	300	40	10	100000	76.1739	2.003	-0.14%	300.4481	-0.15%