

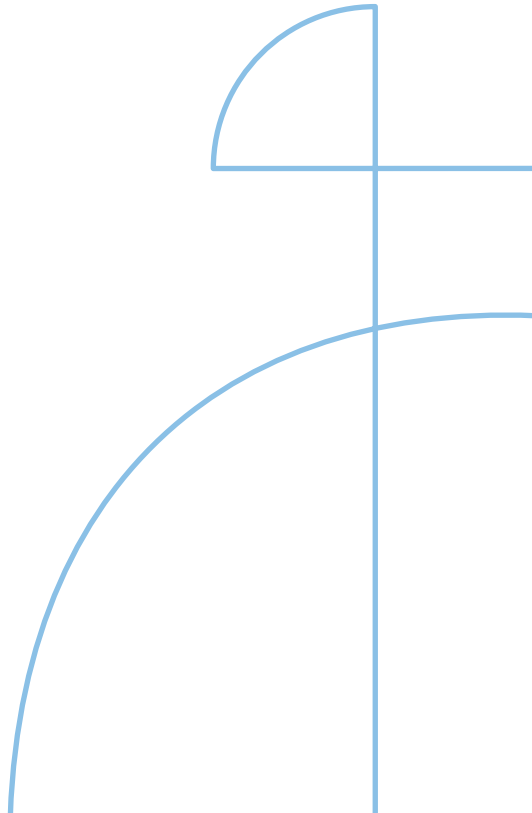


Doctoral Thesis in Electrical Engineering

Simulation-Driven Parameter Estimation with a Focus on Control and Privacy

BRAGHADEESH LAKSHMINARAYANAN

KTH ROYAL INSTITUTE OF TECHNOLOGY



Simulation-Driven Parameter Estimation with a Focus on Control and Privacy

BRAGHADEESH LAKSHMINARAYANAN

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Friday, the 8th May 2026, at 09:00 in F3, Lindstedtsvägen 26 & 28, Stockholm.

Doctoral Thesis in Electrical Engineering
KTH Royal Institute of Technology
Stockholm, Sweden 2026

© Braghadeesh Lakshminarayanan

TRITA-EECS-AVL-2026:24

ISBN 978-91-8106-566-4

Printed by: Universitetsservice US-AB, Sweden 2026

Dedicated to my grandparents.

Abstract

Parameter estimation is a pivotal task across various domains such as system identification, statistics, and machine learning. The literature presents numerous estimation procedures, many of which are backed by well-studied asymptotic properties. In the contemporary landscape, highly advanced digital twins (DTs) offer the capability to faithfully replicate real systems through proper tuning. Leveraging these DTs, simulation-driven estimators can alleviate challenges inherent in traditional methods, notably their computational cost and sensitivity to initializations. Furthermore, traditional estimators often rely on sensitive data, necessitating protective measures. In this thesis, we consider simulation-driven and privacy-preserving approaches to parameter estimation that overcome many of these challenges.

The first part of the thesis delves into an exploration of modern simulation-driven estimation techniques, focusing on the two-stage (TS) approach. Operating under the paradigm of inverse supervised learning, the TS approach simulates numerous samples across parameter variations and employs supervised learning methods to predict parameter values. Divided into two stages, the approach involves compressing data into a smaller set of auxiliary statistics, and the second stage utilizes these statistics to predict parameter values. The simplicity of the TS estimator underscores its interpretability, necessitating theoretical justification, which forms a core motivation for this thesis. We establish statistical frameworks for the TS estimator, yielding its Bayes and minimax versions, alongside developing an improved minimax TS variant based on gradient boosting that excels in computational efficiency. We conduct both asymptotic and non-asymptotic analyses of the TS estimator, establishing strong consistency, asymptotic normality, and finite-sample deviation bounds that characterize the estimation error in terms of the number of training samples and observation length. Finally, we address the question of generating diverse and

informative training samples by proposing a Metropolis-Adjusted Langevin Algorithm (MALA)-based scheme for sampling training parameters from Jeffreys prior, which reflects the intrinsic geometry of the parameter space via the Fisher Information Matrix.

The second part of the thesis considers the problem of adapting pretrained TS estimators to out-of-distribution scenarios. We introduce two fine-tuning methods: a supervised approach that combines feature-space anomaly detection with Fisher-information-guided retraining, and an unsupervised approach that minimizes the discrepancy between simulated and observed trajectories without requiring labeled data.

The third part of the thesis introduces applications of simulation-driven estimation methods in the design of tuning rules for PI controllers. Leveraging synthetic datasets generated from DTs, we train machine learning algorithms to meta-learn tuning rules, streamlining the calibration process without manual intervention. We explore both TS-based approaches with explicit parameter estimation and direct learning approaches using neural network architectures such as convolutional neural networks and WaveNet.

In the final part of the thesis, we tackle scenarios where estimation procedures must handle sensitive data. Here, we introduce differential privacy constraints into the Bayes point estimation problem to protect sensitive information. By proposing a unified approach, we integrate the estimation problem and differential privacy constraints into a single convex optimization objective, thereby optimizing the accuracy-privacy trade-off. In cases where both observations and parameter spaces are finite, this approach reduces to a tractable linear program which is solvable using off-the-shelf solvers.

In essence, this thesis endeavors to address theoretical foundations, robustness, and privacy concerns within the realm of simulation-driven parameter estimation.

Keywords

Parameter estimation, Two-Stage estimator, Neural networks, Differential privacy.

Sammanfattning

Parameterskattning är en central uppgift inom olika områden såsom systemidentifiering, statistik och maskininlärning. Bland befintlig litteratur finns många skattningsmetoder, varav flera har välstuderade asymptotiska egenskaper. Högteknologiska digitala tvillingar (DT) ger möjlighet att troget replikera verkliga system genom korrekt kalibrering. Genom att utnyttja dessa DT kan simuleringsbaserade skattare lösa problem utmaningar som är förknippade med traditionella metoder, framför allt deras beräkningskostnad och känslighet för initialvärden. Dessutom förlitar sig traditionella skattare ofta på känsliga data, vilket kräver skyddsåtgärder. I denna avhandling behandlar vi simuleringsbaserade och integritetsbevarande metoder för parameterskattning som övervinner många av dessa utmaningar.

Den första delen av avhandlingen utforskar moderna simuleringsbaserade skattningstekniker, med fokus på tvåstegsmetoden (TS). TS-metoden, som verkar under paradigmet invers övervakad inlärning, simulerar ett stort antal utfall över parametervariationer och använder övervakade inlärningsmetoder för att prediktera parametervärden. Metoden är uppdelad i två steg: det första steget komprimerar data till en mindre uppsättning hjälpstatistikor, och det andra steget använder dessa statistikor för att prediktera parametervärden. Enkelheten hos TS-skattaren understryker dess tolkningsbarhet, vilket motiverar en teoretisk underbyggnad som utgör en central motivation för denna avhandling. Vi etablerar statistiska ramverk för TS-skattaren, vilket ger upphov till dess Bayes- och minimaxversioner, samt utvecklar en förbättrad minimax-TS-variant baserad på gradientförstärkning som utmärker sig genom beräkningseffektivitet. Vi genomför både asymptotisk och icke-asymptotisk analys av TS-skattaren, och fastställer stark konsistens, asymptotisk normalitet och ändliga-sampel-gränser som karakteriserar skattningsfelet i termer av antalet datapunkter och observationslängd. Slutligen adresserar vi frågan om att gene-

rera diversifierad och informativ träningsdata genom att föreslå en Metropolis-justerad Langevin-algoritm (MALA)-baserad metod för att sampla träningsparametrar från Jeffreys prior, vilket återspeglar parameterrummets inneboende geometri via Fisher-informationsmatrisen.

Den andra delen av avhandlingen behandlar problemet att anpassa förtränade TS-skattare till situationer utanför träningsfördelningen. Vi introducerar två finjusteringsmetoder: en övervakad metod som kombinerar anomalidetektion i egenskapsrummet med Fisher-informationsstyrd omträning, och en självövervakad metod som minimerar diskrepansen mellan simulerade och observerade trajektorier utan att kräva märkt data.

Den tredje delen av avhandlingen introducerar tillämpningar av simuleringsbaserade skattningsmetoder för design av trimningsregler för PI-regulatorer. Genom att utnyttja syntetiska datamängder genererade från DT tränar vi maskininlärningsalgoritmer för att meta-lära trimningsregler, vilket effektiviserar kalibreringsprocessen utan manuellt ingripande. Vi utforskar både TS-baserade metoder med explicit parameterskattning och direkta inlärningsmetoder med neurala nätverksarkitekturer såsom falttningsneuronnät och WaveNet.

I den avslutande delen av avhandlingen behandlar vi scenarier där skattningsförfaranden måste hantera känsliga data. Här introducerar vi differentiell integritet som bivillkor i Bayes-punktskattningsproblemet för att skydda känslig information. Vi föreslår en enhetlig metod där skattningsproblemet och bivillkoren för differentiell integritet integreras i ett enda konvext optimeringsproblem, vilket optimerar avvägningen mellan noggrannhet och integritet. I fall där både observations- och parameterrummen är ändliga reduceras denna metod till ett lösbart linjärt program som kan lösas med standardverktyg.

Sammanfattningsvis syftar denna avhandling till att adressera teoretiska grunder, robusthet och integritetsfrågor inom området simuleringsbaserad parameterskattning.

Nyckelord

Parameterskattning, tvåstegsskattare, neurala nätverk, differentiell integritet.

Thesis Defence Committee

8 May 2026

Supervisor

Prof. Cristian R. Rojas, KTH Royal Institute of Technology, Sweden

Chairman for the Defense

Prof. Mikael Johansson, KTH Royal Institute of Technology, Sweden

Opponent

Prof. Maarten Schoukens, Eindhoven University of Technology, Netherlands

Members of the Grading Committee

Prof. Simone Garatti, Politecnico di Milano, Italy

Prof. Manon Kok, Delft University of Technology, Netherlands

Prof. Andrea Iannelli, University of Stuttgart, Germany

Substitute Member of the Grading Committee

Prof. Johan Karlsson, KTH Royal Institute of Technology, Sweden

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Cristian Rojas, for giving me the opportunity to pursue my PhD at DCS, KTH. His unwavering support and invaluable guidance have been instrumental throughout my doctoral studies. His vast knowledge and expertise have continually inspired me to broaden my horizons, and I am truly grateful for the trust he placed in me in both research and teaching. Thank you, Prof. Rojas, for being such a kind, supportive, and friendly mentor. I also sincerely thank my co-supervisor, Prof. Bo Wahlberg.

I would like to express my heartfelt appreciation to Prof. Håkan Hjalmarsson for his dedication to teaching and for the profound insights I gained from his lectures and lecture notes on System Identification. His enthusiastic approach to research and his unique perspectives have been a constant source of inspiration to me. I am also grateful to my master's thesis supervisor, Prof. Aditya Gopalan, for encouraging me to pursue a PhD. His exceptional teaching and his ability to formulate and address complex problems have been a lasting source of inspiration. I am fortunate to have collaborated with Federico Dettú and Prof. Simone Formentin, without whom Part III of this thesis would not have been possible. Thank you, Federico and Prof. Formentin. I also thank Matteo Cercola at Polimi for his kindness and collaboration during my research visit at Polimi, Milan.

I am immensely grateful to my close friends who made my PhD journey more meaningful through their companionship and support. Special thanks to Javad Parsa and Jiabao He for their friendship and for always being willing to listen to my concerns as fellow PhD students. Your hard work, humility, and deep interest in applied mathematics have inspired me greatly. I am thankful to Anubabh Ghosh for his invaluable collaboration during coursework. I am also grateful to Rodrigo Gonzalez for the weekly fikas and friendship. I would like to thank my group colleagues and friends Andrea Da Col, Margarita, and Yibo for their encouragement and support, and especially Margarita and Yibo for a wonderful collaboration.

I am grateful to Amir Karimi, Amir Daghestani, Miguel, Viswa Narayan, Venkatraman Renganathan, Farhad, Mahsa, Robert Bereza, Rebecka, Victor Molnő, Jacob, Philip Grünter, Andrea Da Col, Yibo, Margarita, Mayank, Dzenan, Xiao Chen, Adrian, Van-

dana, Tommaso, Jacopo, Gregorio, Kamil, Aneesh, Sri Balaji, and Pai-An for their friendship and kindness. I also admire Yuchao, Fei, Jiabao, Nana, Ying, Zifan, and Yibo for their kindness and hard work, which have been a constant source of inspiration to me. I would also like to thank my former and current office colleagues Mina, Fei, Joana, Rodrigo, Yuchao, Gustav, Le, Nana, Lisa, Yulun, Yichen, and Elliot for creating such a pleasant working environment. I am especially grateful to Jiabao, Ying, Miguel, Kamil, Yibo, Margarita, Andrea Da Col, Amir Karimi, Nana, and Philip for generously taking the time to proofread parts of my thesis. I sincerely thank Alice Harting for translating the abstract of this thesis into Swedish, and I also extend my special thanks to Alana and Christer for their administrative help. More broadly, I would like to acknowledge the enriching research environment provided by all my colleagues at DCS.

Lastly, I am deeply grateful to my friends and family in India, whose love and support have been a constant source of strength. Special thanks to my B.Tech friends, the “Hakuna-Matata” gang: Vignesh T., Vijay Anand, Viswesh, Prabhu Venkatesh, Chidi, Arunesh, Richard, PSS, Ravi, Pugazh, Vignesh K., P.B., and Malar, for their friendship and support. I also extend my heartfelt thanks to my school friend, Dr. Akshay Prashanth, and his parents, Dr. Giri and Dr. Booma, for their moral support during a crucial period in my life. I thank Sabarish for his friendship and support during my time at Wipro. I also warmly acknowledge my friends from IISc during my master’s studies: Datta Prasad, Deepak Paramasivan, Arun Rajkumar, Prabu Chandran, Raman Sankaran, Arvind Rameshwar, Varun Krishna, Chandrashekar Sriram, Rajkumar, Ramakrishnan, Sethupathy, Abhishek G., Sandesh Rao, Nagabhushan, Sai Kesava, Vinayak Killedar, Vighnesh Bhat, and my IISc colleagues Mohammadi Zaki and Sayak Ray, whose theoretical rigor constantly motivated me to work harder. I am also deeply thankful to both my maternal and paternal families, whose values, resilience, and words of encouragement have always inspired me and taught me important lessons about facing life’s struggles. The support and strength of this family system have greatly shaped my attitude toward life and helped me navigate challenges with patience and perspective. Finally, I extend my heartfelt thanks to my brother, Dr. Chandrashekar Lakshminarayanan, for his constant guidance, support, and understanding from my childhood until now. To my parents, thank you for your unconditional love and support. I am deeply grateful for your patience, for always listening to my concerns and frustrations, and for standing by me through every challenge, especially during my PhD. My mother and father have shown enormous love and strength in helping me move forward in life despite all the struggles I faced along the way. I also thank my mother and both my paternal and maternal grandmothers, from whom I learned cooking and household skills that became especially valuable after I moved abroad. A special thanks to my mother, who always believed in me and encouraged me to study hard. Both my parents were patient with my slow learning and saw potential in me, which, looking back, played a vital role in shaping who I am today. Finally, I thank my wife, Akshaya, for her love, affection, encouragement, and understanding during the demanding years of my PhD journey.



Braghadeesh Lakshminarayanan
Stockholm, April 8, 2026

Declaration of Use of Generative AI

Generative AI tools, including *ChatGPT*, *Claude* were used in a limited manner during the preparation of this thesis. Their use was restricted to language refinement and assistance with refining TikZ code for a small number of illustrative diagrams. All such outputs were critically reviewed, verified, and revised by the author before inclusion. No generative AI tool was used as a source of scientific content. The author takes full responsibility for the accuracy, originality, references, technical content, and conclusions of the thesis.

A handwritten signature in blue ink, reading "D. Braghadeesh". The signature is fluid and cursive, with the first letter 'D' being large and prominent.

Braghadeesh Lakshminarayanan
Stockholm, April 8, 2026

List of Included Papers

Journal Papers

- J1 **B. Lakshminarayanan** and C. R. Rojas. “On Asymptotic Analysis of the Two-Stage Approach: Towards Data-Driven Parameter Estimation”. Submitted to *Automatica*, 2024.
- J2 **B. Lakshminarayanan** and C. R. Rojas. “Finite Sample Analysis of a Data-Driven Estimator”. Submitted to *IEEE Signal Processing Letters*, 2026.
- J3 **B. Lakshminarayanan**, M. A. Guerrero, and C. R. Rojas. “Fine-Tuning a Simulation-Driven Estimator”. *IEEE Control Systems Letters*, vol. 9, pp. 2975–2980, 2025.
- J4 **B. Lakshminarayanan** and C. R. Rojas. “Trajectory-Level Self-Supervision for Simulation Driven Estimators”. *IFAC Journal of Systems and Control*, vol. 35, p. 100374, 2026.
- J5 **B. Lakshminarayanan**, F. Dettù, C. R. Rojas, and S. Formentin. “Inverse Supervised Learning of Controller Tuning Rules”. *Automatica*, vol. 178, p. 112356, 2025.

Conference Papers

- C1 **B. Lakshminarayanan** and C. R. Rojas. “A Statistical Decision-Theoretical Perspective on the Two-Stage Approach to Parameter Estimation”. In *Proc. 61st IEEE Conference on Decision and Control (CDC)*, pp. 5369–5374, 2022.

- C2 **B. Lakshminarayanan** and C. R. Rojas. “Minimax Two-Stage Gradient Boosting for Parameter Estimation”. In *Proc. 62nd IEEE Conference on Decision and Control (CDC)*, pp. 1189–1194, 2023.

- C3 Y. Shi, **B. Lakshminarayanan**, and C. R. Rojas. “A Metropolis-Adjusted Langevin Algorithm for Sampling Jeffreys Prior”. In *Proc. 64th IEEE Conference on Decision and Control (CDC)*, pp. 2359–2364, 2025.

- C4 F. Dettù, **B. Lakshminarayanan**, S. Formentin, and C. R. Rojas. “From Data to Control: A Two-Stage Simulation-Based Approach”. In *Proc. 2024 European Control Conference (ECC)*, pp. 3428–3433, 2024.

- C5 **B. Lakshminarayanan** and C. R. Rojas. “A Unified Approach to Differentially Private Bayes Point Estimation”. In *Proc. 22nd IFAC World Congress*, pp. 8375–8380, 2023.

Contents

Acknowledgment	xiii
List of Included Papers	xvii
Contents	xix
I Introduction	1
1 Introduction	3
1.1 Simulation-Driven Approaches to Parameter Estimation	4
1.2 Privacy in Point Estimation	6
1.3 Research Questions	7
1.4 Thesis Outline and Contributions	11
1.5 Author's Contributions	15
1.6 Contributions Outside the Thesis	16
2 Preliminaries	17
2.1 General Notation	17
2.2 Large Sample Theory and Concentration Inequalities	18
2.3 Parameter Estimation	20
2.3.1 Estimators and estimates	20
2.3.2 Risk and optimal estimators	21
2.3.3 Bayes estimation	21
2.3.4 Minimax estimation	22
2.3.5 Summary: From estimator to estimate	22
2.4 The Two-Stage Approach	22
2.5 Differential Privacy	25
2.5.1 Interpretation	25
2.5.2 Laplace mechanism	26
II Part 1: Analysis of the TS Estimator	29
Abbreviations	31
Notation and Symbols	35
3 Statistical Framework of the TS Estimator	37

3.1	Problem Setup	38
3.2	A Statistical Decision-Theoretical Formulation of TS	39
3.2.1	Bayes estimator	39
3.2.2	Minimax estimator	42
3.3	Two-Stage Approach for i.i.d. Data	44
3.3.1	First stage	44
3.3.2	Second stage	44
3.4	Simulations	45
3.4.1	Bayes estimator	46
3.4.2	Minimax estimator	48
3.5	Conclusion	50
4	Improved Minimax TS Estimator	51
4.1	Problem Setup	52
4.1.1	Gradient boosting	53
4.2	Minimax TSGBM	54
4.3	Simulation Study	56
4.3.1	Weibull distribution	56
4.3.2	Single parameter dynamical system	57
4.3.3	Stochastic volatility model	59
4.4	Conclusion	61
5	Asymptotic Analysis of TS	63
5.1	Setup for Asymptotic Analysis	63
5.1.1	Definitions of asymptotic properties	64
5.2	Asymptotic Analysis of TS Estimator	65
5.2.1	Main assumptions	66
5.2.2	Consistency of the TS estimator	67
5.2.3	Asymptotic normality of the TS estimator	70
5.2.4	Discussion on the non-i.i.d. case	73
5.3	Numerical Validation of Asymptotic Results	73
5.4	Conclusion	75
6	Non-Asymptotic Analysis of TS	77
6.1	Problem Formulation	77
6.2	TS Estimator	78
6.2.1	First stage of the TS estimator	78
6.2.2	Second stage of the TS estimator	79
6.3	Finite Sample Bounds	79
6.4	Numerical Validation	86
6.5	Conclusion	87
7	Sampling from Jeffreys Prior	89

7.1	Problem Statement	90
7.2	Proposed Method	91
7.2.1	Langevin-based Monte Carlo	91
7.2.2	Metropolis-Adjusted Langevin algorithm	92
7.2.3	Sampling with estimated FIM	93
7.3	Numerical Illustrations	94
7.3.1	Verification of the sampling procedure	95
7.3.2	Sampling Jeffreys Prior for a dynamical system	96
7.3.3	An application of Jeffreys prior to parameter estimation	98
7.4	Conclusion	100
III	Part 2: Fine-Tuning a Simulation-Driven Estimator	101
	Abbreviations	103
	Notation and Symbols	105
8	Supervised Fine-Tuning of a Simulation-Driven Estimator	107
8.1	Problem Statement	108
8.2	The TS Estimator	109
8.2.1	Stage 1 (Compression)	109
8.2.2	Stage 2 (Supervised mapping)	110
8.3	Fine-Tuning the TS Estimator	112
8.3.1	Step 1: Feature-space OOD detector	113
8.3.2	Step 2: Synthetic data generation	115
8.3.3	Step 3: Final-layer transfer on a targeted set	117
8.4	Theoretical Justification for Fine-Tuning Step	117
8.5	Simulation Study	121
8.5.1	Van der Pol oscillator	122
8.5.2	Cascaded water tanks	126
8.6	Conclusion	130
9	Unsupervised Fine-Tuning of a Simulation-Driven Estimator	131
9.1	Problem Statement	132
9.1.1	Two-Stage (TS) Estimator	133
9.2	Proposed Approach	134
9.2.1	Step 1: Feature-space OOD detector	135
9.2.2	Step 2: Trajectory-level unsupervised tuning	135
9.3	Numerical Study	137
9.3.1	Van der Pol oscillator	137
9.3.2	Cascaded water tanks	140
9.4	Conclusion	144

IV Part 3: Applications in Control	145
Abbreviations	147
Notation and Symbols	149
10 The Two-Stage Approach for Controller Tuning	151
10.1 Problem Statement	152
10.2 Existing Approaches	153
10.2.1 Modelling-based approaches	153
10.2.2 Virtual reference feedback tuning	153
10.2.3 Reinforcement learning based approaches	153
10.3 Motivation for Inverse Supervised Learning	154
10.4 The TS-ISL for Meta Learning a Controller Tuning Rule	155
10.4.1 First stage	155
10.4.2 Second stage	156
10.5 Case Study: The Yaw-Rate Tracking Problem	156
10.5.1 System modelling	157
10.5.2 Control design	159
10.6 Simulation Analysis	160
10.6.1 Dataset generation	160
10.6.2 Gray-box identification benchmark	160
10.6.3 Model compression details	161
10.6.4 Second stage training details	161
10.6.5 Controller parameters identification performance	162
10.7 Conclusion	165
11 Direct Learning of Controller Tuning Rules	167
11.1 Problem Statement	167
11.2 CNN-Based Meta-Learning of Controller Tuning Rule	168
11.3 WN-Based Meta-Learning of Controller Tuning Rule	169
11.4 Simulation Analysis	170
11.5 Conclusion	174
V Part 4: Privacy in Point Estimation	175
Abbreviations	177
Notation and Symbols	179
12 Differentially Private Point Estimation	181
12.1 Problem Formulation	182
12.2 Proposed Approach	183

12.2.1	Finite case	184
12.3	Simulations	185
12.3.1	Results	187
12.3.2	Discussion	191
12.4	Conclusion	192
VI	Closing Remarks	193
13	Summary and Future Research Directions	195
13.1	Conclusion: Analysis of the TS Estimator (Part 1)	195
13.1.1	Possible future work	196
13.2	Conclusion: Fine-Tuning a Simulation-Driven Estimator (Part 2) .	197
13.2.1	Possible future work	197
13.3	Conclusion: Applications in Control — Meta-Learning Con- troller Tuning Rules (Part 3)	198
13.3.1	Possible future work	198
13.4	Conclusion: Privacy in Point Estimation (Part 4)	199
13.4.1	Possible future work	199
	Bibliography	201

Introduction

1 Introduction

Mathematical models are essential for describing, predicting, and controlling real-world phenomena across engineering and science. Disciplines such as statistics [1], chemometrics [2], econometrics [3], machine learning [4], and system identification [5, 6] rely on such models to understand, modify, and control physical systems. For instance, in genomics [7], the Hardy–Weinberg principle describes the interaction between genetic markers and a particular trait; in chemometrics, the Beer–Lambert law relates the concentration of a substance to its light absorption; in econometrics, the Black–Scholes formula models financial assets depending on unknown drift and volatility parameters; and in system identification, parametric methods assume a model structure, such as output error (OE), Box–Jenkins (BJ), ARMAX, ARX, or NARX, and select a model that best describes the input-output relationship of a controlled system [8]. Similarly, model-based reinforcement learning requires estimating transition probabilities for each state-action pair in a suitable Markov Decision Process [9].

These mathematical models consist of unknown parameters that need to be estimated; once estimated, the model can be used for tasks such as controller design, simulation, and prediction of future system behavior. Over the years, several estimation procedures have been developed, such as the *prediction error method* (PEM) [10], *particle filter-based approaches* [11, 12, 13], and *instrumental variables* [14]. While these approaches are remarkable in terms of guaranteed convergence and implementation, several of them depend on good initialization since their cost functions are typically non-convex, so convergence to the global optimum is not guaranteed.

Furthermore, when the data used for estimation is collected from sensitive sources such as hospitals, banks, or universities, the resulting estimates can

inadvertently leak information about individual data points. For example, it was demonstrated in genomics [15] that, under some conditions, it is possible to identify whether or not the DNA sample of an individual was present in a dataset based on aggregate statistics alone. This raises a fundamental question: *can we design estimators that are both accurate and privacy-preserving?* This question is formalized in Section 1.2; the remainder of this chapter focuses on the main methodological theme of the thesis: simulation-driven parameter estimation, which addresses the computational and initialization limitations of the classical methods discussed above.

1.1 Simulation-Driven Approaches to Parameter Estimation

The use of simulators for parameter estimation has a long history, particularly in econometrics. Classical approaches include the *method of simulated moments* [16], which matches simulated and observed summary statistics, and *indirect inference* [3], in which an auxiliary model is fitted to both observed and simulated data, and the parameters are adjusted to minimize the discrepancy between the auxiliary model fits on the two datasets. While these methods make effective use of simulators, they require solving an optimization problem at estimation time for each new observation. This per-observation computational cost can be significant, particularly when estimation must be performed repeatedly across many systems or operating conditions.

A fundamentally different strategy is to shift the computational effort to an offline training phase: use the simulator to generate a large synthetic dataset of parameter-observation pairs, and then train a machine learning model that learns a direct mapping from observations to parameters. Once trained, such a *simulation-driven estimator* produces estimates via a single forward pass, with negligible computation time at deployment. This construction is made practical by recent advancements in high-fidelity modeling, which have led to the widespread availability of sophisticated simulators, popularly known as *digital twins* (DTs) [17]. Although a DT provides a detailed description of the system, it is itself parameterized by unknown quantities, e.g., friction coefficients and material properties, that vary across individual systems or operating conditions; estimating these parameters from observed data is necessary to calibrate the DT to a specific real system. Crucially, a DT can be queried offline to generate large volumes of synthetic data for many parameter configurations, making the construction of simulation-driven estimators feasible.

The general construction of a simulation-driven estimator is illustrated in Figure 1.1.1. Based on prior beliefs or domain expertise, a parameter space $\Theta \subseteq \mathbb{R}^d$ is chosen to reflect the plausible range of simulator parameters; for example, if the unknown quantities are friction coefficients and material properties as discussed above, Θ would encode the physically reasonable ranges for these quantities. Parameter configurations $\theta_1, \dots, \theta_M$ are then sampled from Θ , and for

each sampled parameter, the simulator generates corresponding input-output data. This synthetic dataset of parameter and input-output data pairs is used to train a machine learning model $T : \mathbb{R}^{2N} \rightarrow \mathbb{R}^d$ that learns a direct mapping from input-output data to parameters. Once trained, the estimator produces estimates via a single forward pass, with *negligible computation time*. In this sense, the approach *meta-learns* the estimation procedure, that is, it constructs a single estimator that is valid across a class of systems parameterized by the simulator.

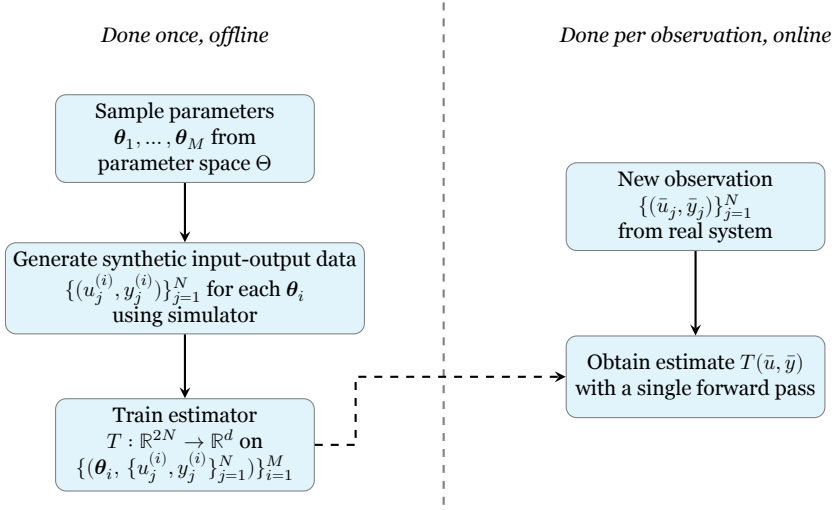


Figure 1.1.1: Construction of a simulation-driven estimator. Here, u and y denote the input and output signals, respectively, and the sequence $\{(u_j, y_j)\}_{j=1}^N$ forms input-output data of length N . The offline phase (left) trains the estimator T on parameter and input-output data pairs generated from the simulator for various parameter configurations sampled from the parameter space Θ . The online phase (right) produces estimates via a single forward pass with negligible computation time.

The principle of leveraging simulated data to construct estimators offline has gained renewed attention in recent years. For instance, *DeepBayes* [18] trains neural networks on synthetic datasets to produce conditional mean estimates for dynamical systems, *learning to estimate without bias* [19] constructs estimators from synthetic data with desirable statistical properties, and *TabPFN* [20] employs a transformer architecture pre-trained on millions of synthetic tasks, enabling it to solve new problems in a single forward pass. Related works leveraging synthetic data in identification tasks beyond parameter estimation are comprehensively covered in [21, 22].

A closely related line of research is *simulation based inference* [23], driven by advances in deep learning. Methods such as *sequential neural posterior estimation* and its variants [24, 25] also leverage simulators to generate synthetic

data; however, their goal is to approximate the full posterior distribution rather than to produce a single point estimate. This thesis focuses on the *point estimation* problem: given observed data, output a single parameter estimate. Unlike posterior inference methods, the simulation-driven estimators studied here do not necessarily require a differentiable simulator, since the simulator is used solely to generate labeled training data and the estimator learns a direct mapping from observations to parameters via supervised learning.

The motivation for studying simulation-driven estimators is threefold. First, from a *modeling* perspective, the availability of a DT provides a natural mechanism for generating abundant synthetic training data, enabling the construction of estimators without requiring an explicit likelihood function. Second, from a *computational* perspective, the offline training phase shifts the computational burden away from the deployment phase, allowing fast estimation via a single forward pass. Third, unlike classical methods such as PEM or *extended Kalman filter* (EKF) [26], simulation-driven estimators do not require user-specified initialization, since the estimator is trained offline and deployed directly on new observations. These advantages make simulation-driven point estimators attractive for applications where fast, repeated estimation is required across a family of systems. As a concrete illustration of these advantages, we refer the reader to [27], where a simulation-driven estimator is compared against PEM (with multiple random initializations) and the EKF (with varying initial covariance) for an autonomous stochastic system. The results therein demonstrate that the simulation-driven estimator accurately recovers the true parameter across the entire parameter range, without requiring initialization, while operating orders of magnitude faster than PEM since estimation reduces to a single forward pass.

1.2 Privacy in Point Estimation

As mentioned earlier, when estimation is performed on data collected from sensitive sources, there is a risk that the resulting estimates leak information about individual data points. A pressing awareness of this issue emerged in 2010 when Netflix, a popular on-demand video streaming company, canceled its well-known “Netflix Prize” competition due to privacy concerns [28]. When releasing the database of its users to competitors for building a better recommendation system [29], Netflix anonymized its users using standard anonymization tricks. However, such tricks could not safeguard the sensitive information of the users, as demonstrated by the authors of [30], where “reconstruction attacks” were proposed to establish that the leakage of sensitive information is possible simply by cross-referencing users in the IMDB database (a movie reviewing website).

These concerns are not limited to the release of anonymized datasets. Even when raw data are not disclosed, the output of an estimation procedure it-

self may still reveal information about individual samples. For example, it was demonstrated in genomics [15] that, under some conditions, it is possible to identify whether or not the DNA sample of an individual was present in a dataset based on aggregate statistics. For this reason, the National Institute of Health (NIH) removed access to some aggregate statistics such as p-values and chi-squared statistics, which were once openly available [31]. Figure 1.2.2 depicts a scenario where an intruder attempts to infer the sensitive information of individuals based on an aggregate statistic.

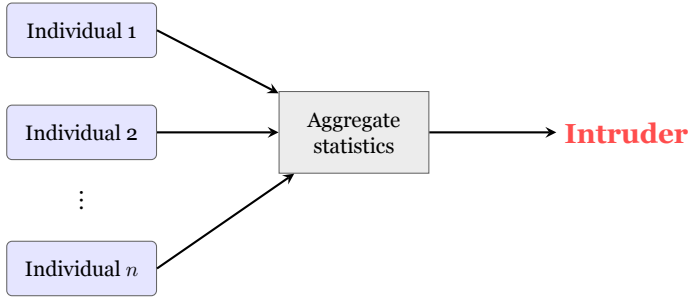


Figure 1.2.2: Privacy violation via aggregate statistics. Data from multiple individuals is combined into aggregate statistics (e.g., sample mean). An intruder observing these statistics may attempt to infer sensitive information about specific individuals, violating their privacy.

Several notions of privacy have been explored in the literature, most of which fall under the broader umbrella of *anonymization* tricks. However, for estimation problems, most of these anonymization tricks are not robust against privacy attacks. As a result, in the machine learning and statistics literature, another notion of privacy was revisited: *differential privacy* (DP), originally proposed in [32]. Here, the objective is to produce estimates as solutions to an optimization problem while satisfying DP constraints. Such an objective has proven to be successful in protecting the privacy of individuals in a database where the estimation task is carried out by an algorithm. This is traditionally achieved by the so-called *Laplace mechanism*, where the estimates are randomized by adding Laplace-distributed noise. DP ensures that by observing the output of an algorithm, almost no probabilistic inference can be made about the observations used to produce this output; the level of desired privacy can be tuned via a parameter $\epsilon > 0$. However, it remains unclear whether the Laplace mechanism is optimal in terms of the accuracy-privacy trade-off. In particular, increasing the amount of Laplace noise improves privacy at the expense of accuracy, whereas reducing it improves accuracy but weakens privacy.

1.3 Research Questions

Among the various simulation-driven estimators discussed in Section 1.1, this thesis focuses on the *Two-Stage* (TS) estimator [33, 34], which is described

in detail in Chapters 2 and 3. The TS estimator first compresses observations through a feature extractor and then maps the extracted features to parameter estimates via a machine learning model. Beyond parameter estimation, the thesis also explores how the same simulation-driven framework can be applied to controller tuning. The thesis addresses six research questions, organized below.

Statistical foundations of the TS estimator

While the TS estimator offers computational advantages by leveraging a simulator to produce abundant synthetic data, its precise statistical properties, such as consistency and asymptotic normality, have seldom been studied. This stands in contrast to classical estimators like PEM or maximum likelihood, for which theoretical results exist [5, 6]. Establishing such properties for the TS estimator is important for two reasons: first, consistency and asymptotic normality ensure that the estimator converges to the true parameter as the amount of training data or observation length increases; second, finite-sample guarantees quantify the estimation error for a given data budget, providing concrete guidance for practitioners.

Research Question 1

Can the consistency and asymptotic normality of the TS estimator be established?

Research Question 2

Can finite-sample (non-asymptotic) guarantees of TS be provided?

Training data diversity

Beyond theoretical guarantees, simulation-driven estimators face practical challenges related to their construction. As outlined earlier, the TS estimator is built by sampling parameter configurations from a predefined range, generating input-output trajectories from the simulator for each configuration, and training a machine learning model to map observations to parameter estimates. The quality of the resulting estimator depends critically on the *diversity* of the synthetic training data: if the sampled parameters do not adequately cover the informative regions of the parameter space, the resulting estimator may exhibit high variance or poor generalization.

Research Question 3

How can we generate diverse training samples that reflect the intrinsic geometry of the parameter space, thereby improving the TS estimator's performance?

Out-of-distribution adaptation

Furthermore, once trained, the TS estimator is deployed to estimate the true parameters of a real system based on observed data. However, when the true parameters lie outside the predefined range used during training, i.e., an *out-of-distribution* (OOD) scenario, the pretrained estimator can produce poor estimates. Addressing this challenge requires both theoretical understanding and principled algorithmic solutions.

Research Question 4

How can simulation-driven estimators be adapted to out-of-distribution scenarios at the implementation phase?

Meta-learning controller tuning rules

While the preceding research questions focus on the construction and adaptation of simulation-driven estimators, the scope of these methods is not limited to parameter estimation. When a high-fidelity simulator is available, the same framework can also be used for other tasks; one important example being the meta-learning of controller tuning rules. Traditionally, a control engineer designs a controller by first identifying a model of the plant and then applying a model-based design rule such as loop shaping [35] to compute the controller parameters. This process relies heavily on domain expertise and can be time-consuming, particularly when dealing with a family of plants with varying characteristics.

The same meta-learning philosophy underlying the simulation-driven estimator can be applied to automate this process, as illustrated in Figure 1.3.3.

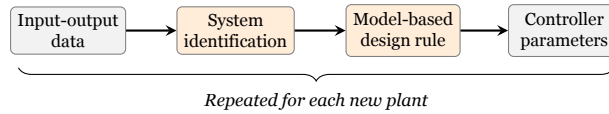
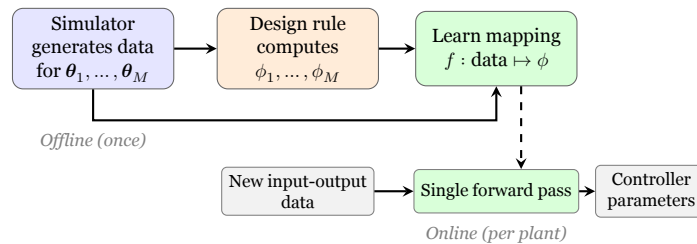
Traditional approach**Meta-learning approach**

Figure 1.3.3: Comparison of traditional controller design and meta-learning approach. **Top:** Traditional approach requires system identification and model-based design for each new plant. **Bottom:** Meta-learning approach trains offline on synthetic data from the simulator, then deploys the learned tuning rule via a single forward pass.

In the traditional approach [5, 6], a control engineer must perform system identification and apply a model-based design rule for each new plant encountered, a time-consuming procedure that must be repeated whenever the plant characteristics change. The meta-learning approach, by contrast, trains offline using synthetic data from the simulator: for various parameter configurations $\theta_1, \dots, \theta_M$, input-output data are generated and a model-based design rule computes the corresponding controller parameters $\phi_1, \dots, \phi_M$. A machine learning model is then trained to directly map input-output data to controller parameters, effectively *learning how a control engineer would tune the controller*. Once trained, this learned tuning rule can be applied to new input-output data in real-time via a single forward pass, outputting appropriate controller parameters without repeating the identification and design steps. In this sense, the approach meta-learns the controller tuning rule itself.

Research Question 5

Can simulation-driven estimators be leveraged to meta-learn controller tuning rules directly from data?

Privacy-optimal point estimation

As discussed in Section 1.2, when estimation is performed on sensitive data, the resulting estimates can leak individual information. While the Laplace mechanism is the standard tool for enforcing differential privacy, it remains unclear whether it achieves the best possible accuracy-privacy trade-off.

Research Question 6

Is the Laplace mechanism optimal in the sense of the accuracy-privacy trade-off?

These six research questions define the scope of this thesis. The chapters that follow develop the theoretical foundations of the TS estimator, propose principled methods for improving its OOD adaptation, extend the framework of a simulation-driven estimator to controller tuning, and study privacy-preserving approaches to parameter estimation.

1.4 Thesis Outline and Contributions

The materials presented in this thesis are the result of articles that are previously published, in preparation, or submitted for publication. The thesis is organized into four parts spanning thirteen chapters, each addressing specific aspects of simulation-driven estimation and its applications. The connections between these chapters and the research questions posed above are outlined below.

Chapter 2

Chapter 2 establishes notations and necessary background that are required to understand the rest of the thesis.

Part 1: Analysis of the TS Estimator

Chapter 3

Chapter 3 describes the statistical frameworks for the TS estimator that lead to the derivation of Bayes and Minimax estimators, and provides a theoretical justification for the TS approach. These frameworks lay the foundation for addressing **Research Questions 1 & 2** in subsequent chapters, where some of the statistical properties of these estimators are analyzed. The results presented in this chapter are based on the following paper:

- C1. **B. Lakshminarayanan** and C. R. Rojas. “A Statistical Decision-Theoretical Perspective on the Two-Stage Approach to Parameter Esti-

mation”. In *61st IEEE Conference on Decision and Control (CDC)*, pages 5369–5374, 2022.

Chapter 4

Chapter 4 addresses the computational difficulty of the Minimax TS estimator which was derived in Chapter 3. In particular, the min-max objective corresponding to the TS estimator is solved via gradient boosting. Also, numerical examples are provided to validate the minimax TS approach for the case when the data-generating mechanism is non-i.i.d. The findings of this chapter are based on the following paper:

C2. **B. Lakshminarayanan** and C. R. Rojas. “Minimax Two-Stage Gradient Boosting for Parameter Estimation”. In *62nd IEEE Conference on Decision and Control (CDC)*, pages 1189–1194, 2023.

Chapter 5

In Chapter 5, the asymptotic properties of the TS estimator are studied, addressing **Research Questions 1 & 2**. In particular, the strong consistency and asymptotic normality of the TS estimator are established under reasonable assumptions, and to establish these properties, the Bayes framework is considered. The theoretical analyses provided in this chapter are based on the following article:

J1. **B. Lakshminarayanan** and C. R. Rojas. “On Asymptotic Analysis of the Two-Stage Approach: Towards Data-Driven Parameter Estimation”. Submitted to *Automatica*, 2024.

Chapter 6

Chapter 6 provides a non-asymptotic analysis of the TS estimator, establishing finite-sample guarantees that complement the asymptotic results of Chapter 5. This chapter further addresses **Research Questions 1 & 2** by characterizing the estimation error in terms of the number of training samples M and the observation length N , providing practitioners with concrete guidance on the data requirements for achieving a desired estimation accuracy. The results presented in this chapter are based on the following paper:

J2. **B. Lakshminarayanan** and C. R. Rojas. “Finite Sample Analysis of a Data-Driven Estimator”. *IEEE Signal Processing Letters* (submitted for publication), 2026.

Chapter 7

Chapter 7 addresses **Research Question 3** on how to generate diverse training samples for the TS estimator. Chapter 7 proposes a Metropolis-Adjusted

Langevin Algorithm (MALA)-based scheme for sampling from Jeffreys prior, which reflects the intrinsic geometry of the parameter space via the Fisher Information Matrix (FIM). The chapter presents both the case where the FIM can be computed analytically and the case where it must be approximated using particle filtering. Numerical examples demonstrate that sampling according to Jeffreys prior improves the performance of a simulation-driven estimator compared to uniform sampling. The results presented in this chapter are based on the following paper:

C3. Y. Shi, **B. Lakshminarayanan**, and C. R. Rojas. “A Metropolis-Adjusted Langevin Algorithm for Sampling Jeffreys Prior”. In *64th IEEE Conference on Decision and Control (CDC)*, pages 2359–2364, 2025.

Part 2: Fine-Tuning a Simulation-Driven Estimator

Chapter 8

Chapter 8 addresses **Research Question 4** by considering the problem of adapting a pretrained TS estimator to OOD scenarios using a supervised fine-tuning approach. The method consists of three steps: (i) a feature-space OOD detector based on Wald’s test, (ii) a Gauss-Newton refinement combined with Fisher-information-guided sampling to generate a targeted synthetic dataset, and (iii) transfer learning that updates only the final layers of the second-stage neural network. Theoretical justification for the OOD detection and sampling procedures is provided, and numerical experiments on the Van der Pol oscillator and cascaded water tanks demonstrate the effectiveness of the approach. The results presented in this chapter are based on the following paper:

J3. **B. Lakshminarayanan**, M. A. Guerrero, and C. R. Rojas. “Fine-Tuning a Simulation-Driven Estimator”. *IEEE Control Systems Letters*, vol. 9, pages 2975–2980, 2025.

Chapter 9

Chapter 9 presents an alternative, unsupervised approach to fine-tuning the TS estimator under OOD scenarios, further addressing **Research Question 4**. Unlike the supervised approach in Chapter 8, this method updates all weights of the second-stage neural network by minimizing the simulation error—the discrepancy between trajectories generated by the simulator and the true system. This requires the simulator to be differentiable with respect to its parameters, enabling end-to-end backpropagation. The same OOD detection mechanism from Chapter 8 is employed to trigger fine-tuning only when necessary. Numerical experiments demonstrate that the unsupervised approach achieves competitive performance with the supervised method while being conceptually simpler. The results presented in this chapter are based on the following paper:

J4. **B. Lakshminarayanan** and C. R. Rojas. “Trajectory-Level Self-Supervision for Simulation Driven Estimators”. *IFAC Journal of Systems and Control*, vol. 35, page 100374, 2026.

Part 3: Applications in Control

Chapter 10

In Chapter 10, we address **Research Question 5** by demonstrating the application of the TS estimator to meta-learn a controller tuning rule by making use of an available DT. The findings of this chapter are based on the following paper:

C4. F. Dettù, **B. Lakshminarayanan**, S. Formentin, and C. R. Rojas. “From Data to Control: A Two-Stage Simulation-Based Approach”. In *Proceedings of the 2024 European Control Conference (ECC)*, pages 3428–3433, 2024.

Chapter 11

While Chapter 10 discussed the role of the TS approach to tune a controller that involved an identification step in the first stage, Chapter 11 further addresses **Research Question 5** by exploring the idea of directly learning a controller tuning rule via sequence models. This approach bypasses the explicit parameter estimation step and instead learns a direct mapping from observations to controller parameters using neural network architectures such as WaveNets (WNs) and convolutional neural networks (CNNs). The results presented in this chapter are based on the following article:

J5. **B. Lakshminarayanan**, F. Dettù, C. R. Rojas, and S. Formentin. “Inverse Supervised Learning of Controller Tuning Rules”. *Automatica*, vol. 178, p. 112356, 2025.

Part 4: Privacy in Point Estimation

Chapter 12

Chapter 12 addresses **Research Question 6** by considering the Bayes point estimation problem under differential privacy constraints. Interesting insights are drawn by proposing a unified approach for combining estimation and privacy into a single convex optimization program. The results presented in this chapter are based on the following paper:

C5. **B. Lakshminarayanan** and C. R. Rojas. “A Unified Approach to Differentially Private Bayes Point Estimation”. In *Proceedings of the 22nd IFAC World Congress (IFAC WC)*, pp. 8375–8380, 2023.

Chapter 13

Chapter 13 summarizes the main conclusions from the four parts of this thesis and outlines possible future directions.

1.5 Author's Contributions

In Chapters 3, 4, 5, 6, 9, and 12, the thesis author serves as the primary contributor of the related publications, with the co-authors significantly contributing to supervision, technical discussions, offering valuable suggestions, and making substantial manuscript improvements.

For Chapter 7, the first author (Y. S.) developed the MALA-based sampling scheme and implemented the numerical examples. The second author (**B. L.**) contributed to the problem formulation, and application to simulation-driven estimation. The third author (C. R. R.) provided supervision, technical guidance, and manuscript improvements. All authors contributed to writing the manuscript.

For Chapter 8, the thesis author (**B. L.**) developed the fine-tuning methodology and implemented the numerical experiments. The second author (M. A. G.) contributed to the manuscript writing. The third author (C. R. R.) provided supervision, technical guidance, and manuscript improvements. All authors contributed to writing the manuscript.

For Chapter 10, the second (**B. L.**) and last (C. R. R.) authors of the corresponding articles proposed utilizing the TS approach to design a controller. The second (**B. L.**) author further designed a gradient boosting algorithm for implementation. The first (F. D.) and third (S. F.) authors suggested employing a DT, designed a practical example, and the second author implemented a feed-forward network for the controller. All authors actively engaged in technical discussions, equally contributed to manuscript writing, and played significant roles in its improvement.

The publication related to Chapter 11 has equal contributions from the first (**B. L.**) and second (F. D.) authors in implementing the tuning rule for the controller. The first author (**B. L.**) introduced and implemented the idea of using a WaveNet, while the second author (F. D.) proposed and implemented a CNN. The third (C. R. R.) and fourth (S. F.) authors contributed significantly by making substantial manuscript improvements, providing suggestions, and supervising the first and second authors. All authors equally contributed to writing the manuscript.

1.6 Contributions Outside the Thesis

The following contributions are not included in the thesis:

- M. A. Guerrero, **B. Lakshminarayanan**, and C. R. Rojas. “Gain-Scheduled Data-Enabled Predictive Control: A DeePC Approach for Nonlinear Systems”. *IEEE Control Systems Letters*, vol. 9, pp. 3041–3046, 2025.
- M. A. Guerrero, **B. Lakshminarayanan**, and C. R. Rojas. “Data-Driven Estimation of Structured Singular Values”. *IEEE Control Systems Letters*, vol. 9, pp. 1976–1981, 2025.
- A. Gopalan, **B. Lakshminarayanan**, and V. Saligrama. “Bandit Quickest Changepoint Detection”. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 29064–29073, 2021.

2 Preliminaries

In this chapter, we specify the general notations that we use in the thesis. A simulation-driven parameter estimation procedure, as described in the first part of this thesis, is designed and analyzed using mathematical tools such as large sample theory and risk minimization, whose backgrounds are provided in this chapter. Finally, a background on DP is provided and the Laplace mechanism that guarantees DP is reviewed.

2.1 General Notation

This section summarizes the general notation used throughout the thesis. While we strive for consistency, some notation may be adapted in specific chapters or parts to align with conventions in the relevant literature; any such deviations are noted where they occur.

The set of real numbers is denoted by $\mathbb{R}$, the set of strictly positive real numbers by $\mathbb{R}^+$, and the set of non-negative real numbers by $\mathbb{R}_0^+$. The set of natural numbers is denoted by $\mathbb{N}$. For $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$ as the index set of positive integers from 1 to n . If A and B are sets, then $A \subseteq B$ denotes that A is a subset of B , and $|B|$ denotes the cardinality of B . For $x \in \mathbb{R}$, $\lfloor x \rfloor$ denotes the floor function, i.e., the largest integer z such that $z \leq x$, and $|x|$ denotes the absolute value of x . The indicator function of a set A is denoted by $\mathbb{1}_A\{x\}$, which takes the value 1 if $x \in A$ and 0 otherwise. The composition of a function $h: A \rightarrow B$ with a function $g: B \rightarrow C$ is denoted by $g \circ h$. The derivative of a differentiable function $h: \mathbb{R} \rightarrow \mathbb{R}$ is denoted by h' . Vectors are written in bold lowercase (e.g., $\mathbf{x}$, $\mathbf{y}$, $\boldsymbol{\theta}$) and matrices in bold uppercase (e.g., $\mathbf{A}$, $\mathbf{X}$). A vector $\mathbf{y} \in \mathbb{R}^N$ is written as $\mathbf{y} = (y_1, \dots, y_N)^\top$, where $(\cdot)^\top$ denotes the transpose. For a matrix $\mathbf{X}$, we denote by $\mathbf{X}^\top$ its transpose, by $\|\mathbf{X}\|_2$ its in-

duced 2-norm (i.e., maximum singular value), by $\|X\|_{\max}$ its maximum norm (i.e., the maximum of the absolute values of its entries), and by $\sigma_{\min}(X)$ its minimum singular value (i.e., $\|X^{-1}\|_2^{-1}$). Given matrices A and B of the same size, $A \succ B$ means that $A - B$ is positive definite. A diagonal matrix $A \in \mathbb{R}^{n \times n}$ is denoted by $\text{diag}(a_1, \dots, a_n)$, where a_i is the i -th diagonal entry. The trace of a square matrix A is denoted by $\text{tr}(A)$. For a vector $x \in \mathbb{R}^n$, we denote by $\|x\|_1 := \sum_{i=1}^n |x_i|$ its ℓ_1 -norm, by $\|x\|_2 := \sqrt{\sum_{i=1}^n x_i^2}$ its ℓ_2 -norm (Euclidean norm), and by $\|x\|_A^2 := x^\top A x$ its weighted norm with respect to a compatible positive definite matrix A . The Jacobian matrix of a function $J: \mathbb{R}^m \rightarrow \mathbb{R}^n$ with respect to $x = (x_1, \dots, x_m)^\top \in \mathbb{R}^m$ is denoted by $\nabla J(x) \in \mathbb{R}^{n \times m}$, with entries

$$[\nabla J(x)]_{i,j} = \frac{\partial J_i(x)}{\partial x_j}, \quad i = 1, \dots, n, \quad j = 1, \dots, m.$$

The triplet $(\Omega, \mathcal{F}, \mathbb{P})$ denotes a probability space, where Ω is the sample space, $\mathcal{F}$ is the associated σ -algebra, and $\mathbb{P}$ is the corresponding probability measure. $\mathbb{P}(A)$ denotes the probability of an event A , and $\mathbb{E}[\cdot]$ denotes the expectation. A random variable is denoted by X , its probability density function (pdf) by f , and its cumulative distribution function (cdf) by F . The symbol $\sim$ denotes “distributed according to”. $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2 , and $\text{Lap}(a, b)$ denotes the Laplace distribution with location parameter a and scale parameter b . The Hamming distance between two vectors is denoted by d_H .

2.2 Large Sample Theory and Concentration Inequalities

In this section, we recall definitions and results from probability theory that are used throughout the thesis, stated without proofs; for details, the reader is referred to [36, 37, 38, 39, 40]. We begin with the definition of random variables and probability spaces, at a level sufficient for the thesis. We then review four modes of convergence, the weak and strong laws of large numbers, the central limit theorem, the Delta method, and a few properties of almost sure convergence. Finally, we state Hoeffding’s inequality.

Let $(X_n)_{n \in \mathbb{N}}$ be a sequence of random variables, each defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

Definition 2.1 (Almost sure convergence [37, 36]). A sequence of r.v.s $(X_n)_{n \in \mathbb{N}}$ converges almost surely (a.s.) to the r.v. X if

$$\mathbb{P}(\{\omega \in \Omega: X_n(\omega) \rightarrow X(\omega) \text{ as } n \rightarrow \infty\}) = 1.$$

We denote the a.s. convergence as $X_n \xrightarrow{a.s.} X$.

Definition 2.2 (Convergence in mean square [37, 36]). A sequence of r.v.s $(X_n)_{n \in \mathbb{N}}$ converges in mean square sense to the r.v. X if

$$\mathbb{E}[|X_n - X|^2] \rightarrow 0 \text{ as } n \rightarrow \infty.$$

The mean square convergence is denoted as $X_n \xrightarrow{m.s.} X$.

Definition 2.3 (Convergence in probability [37, 36]). A sequence of r.v.s $(X_n)_{n \in \mathbb{N}}$ converges in probability to the r.v. X if

$$\text{for each } \varepsilon > 0, \mathbb{P}(\{\omega \in \Omega : |X_n(\omega) - X(\omega)| > \varepsilon\}) \rightarrow 0 \text{ as } n \rightarrow \infty.$$

We represent the convergence in probability as $X_n \xrightarrow{p} X$.

Definition 2.4 (Convergence in distribution [37, 36]). A sequence of r.v.s $(X_n)_{n \in \mathbb{N}}$ converges in distribution to the r.v. X if it holds for the sequence of respective CDFs F_{X_n} that

$$F_{X_n}(x) \rightarrow F_X(x) \text{ as } n \rightarrow \infty \text{ for all } x \text{ that are points of continuity of } F_X(x).$$

We write this compactly as $X_n \xrightarrow{d} X$.

Theorem 2.1 (Relation between convergences [37, 36]). *For the sequence of r.v.s $(X_n)_{n \in \mathbb{N}}$, the following holds:*

1. $X_n \xrightarrow{m.s.} X \implies X_n \xrightarrow{p} X$;
2. $X_n \xrightarrow{a.s.} X \implies X_n \xrightarrow{p} X$;
3. $X_n \xrightarrow{p} X \implies X_n \xrightarrow{d} X$.

Theorem 2.2 (Strong law of large numbers [37, 36, 41]). *Let $(X_n)_{n \in \mathbb{N}}$ be a sequence of i.i.d. r.v.s defined on $(\Omega, \mathcal{F}, \mathbb{P})$, with $\mathbb{E}[X_i] = b$ and $\text{Var}[X_i] = \sigma^2 < \infty$. Let $S_n := \sum_{i=1}^n X_i$. Then,*

$$\frac{S_n}{n} \xrightarrow{a.s.} b.$$

Theorem 2.3 (Weak law of large numbers [37, 36, 41]). *Let $(X_n)_{n \in \mathbb{N}}$ be a sequence of i.i.d. r.v.s defined on $(\Omega, \mathcal{F}, \mathbb{P})$, with $\mathbb{E}[X_i] = b$ and $\text{Var}[X_i] = \sigma^2 < \infty$. Let $S_n := \sum_{i=1}^n X_i$. Then,*

$$\frac{S_n}{n} \xrightarrow{p} b.$$

Theorem 2.4 (Central limit theorem [37, 36, 41]). *Let $(X_n)_{n \in \mathbb{N}}$ be a sequence of i.i.d. r.v.s defined on $(\Omega, \mathcal{F}, \mathbb{P})$, with $\mathbb{E}[X_i] = b$ and $\text{Var}[X_i] = \sigma^2 < \infty$. Let $S_n := \sum_{i=1}^n X_i$. Then,*

$$\sqrt{n} \left(\frac{S_n}{n} - b \right) \xrightarrow{d} \mathcal{N}(0, \sigma^2).$$

Theorem 2.5 (Hoeffding's inequality [42]). *Suppose the r.v.s $(X_i)_{i=1}^N$ defined on $(\Omega, \mathcal{F}, \mathbb{P})$ are independent and X_i has mean ν_i with $a_i \leq X_i \leq b_i$, where $a_i \leq b_i$ and $a_i, b_i \in \mathbb{R}$ are finite real numbers. Then, for all $t \geq 0$, we have*

$$\mathbb{P} \left[\frac{\sum_{i=1}^N (X_i - \nu_i)}{N} \geq t \right] \leq \exp \left\{ -\frac{2Nt^2}{\sum_{i=1}^N (b_i - a_i)^2} \right\}. \quad (2.2.1)$$

2.3 Parameter Estimation

Consider a physical, controlled *single-input-single-output* (SISO) system that produces outputs $\mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^N$ based on an input signal $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^N$, where $\mathcal{U}$ is called the *input space*, $\mathcal{Y}$ is called the *output space*, and $N \geq 1$ (typically large) denotes the number of samples at discrete time instants.

To estimate the unknown parameters of the system, we consider a *model structure*: a family of parametric models $\{\mathcal{M}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$, where $\Theta \subseteq \mathbb{R}^d$, $d \geq 1$, is called the *parameter space*. Each model $\mathcal{M}(\boldsymbol{\theta})$ takes an input signal $\mathbf{u}$ and produces the corresponding output signal $\mathbf{y}^{(\boldsymbol{\theta})}$, as shown in Figure 2.3.1.

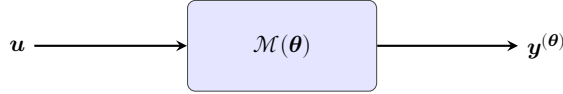


Figure 2.3.1: A model $\mathcal{M}(\boldsymbol{\theta})$ from the model structure. The input $\mathbf{u} = (u_1, \dots, u_N)^\top$ is fed to the model, which produces the output $\mathbf{y}^{(\boldsymbol{\theta})} = (y_1^{(\boldsymbol{\theta})}, \dots, y_N^{(\boldsymbol{\theta})})^\top$.

In general, the system under study need not share the same structure as $\mathcal{M}(\boldsymbol{\theta})$, nor is it required that any $\boldsymbol{\theta} \in \Theta$ reproduces it exactly; the parameter estimation framework developed here applies regardless. We assume that a high-fidelity simulator is available, so that the system can be represented as $\mathcal{M}(\boldsymbol{\theta}_0)$ for some unknown $\boldsymbol{\theta}_0 \in \mathbb{R}^d$. We denote it as the *real system*.

Given observations $(\mathbf{u}, \mathbf{y})$ from the real system $\mathcal{M}(\boldsymbol{\theta}_0)$, where $\mathbf{u} = (u_1, \dots, u_N)^\top$ and $\mathbf{y} = (y_1, \dots, y_N)^\top$, the goal is to estimate $\boldsymbol{\theta}_0$. We assume that the observations are generated according to a probability distribution $\mathbb{P}^N(\mathbf{y}|\mathbf{u}; \boldsymbol{\theta}_0)$ depending on $\boldsymbol{\theta}_0$.

2.3.1 Estimators and estimates

An *estimator* is a function $T: \mathcal{U} \times \mathcal{Y} \rightarrow \Theta$ that maps input-output observations to a parameter value. Given observations $(\mathbf{u}, \mathbf{y})$ from the real system, an *estimate* of $\boldsymbol{\theta}_0$ is the value

$$\hat{\boldsymbol{\theta}} = T(\mathbf{u}, \mathbf{y}) \in \Theta. \quad (2.3.2)$$

The key distinction is that T is a function (designed before observing data), while $\hat{\theta}$ is a value (obtained after applying T to specific observations). The quality of the estimate $\hat{\theta}$ as an approximation to θ_0 depends on how well the estimator T is designed.

2.3.2 Risk and optimal estimators

To quantify the performance of an estimator T , we introduce the *loss function* $\ell: \Theta \times \Theta \rightarrow \mathbb{R}_0^+$, where $\ell(\theta_0, \hat{\theta})$ measures the cost of obtaining the estimate $\hat{\theta}$ when the true parameter is θ_0 . A common choice is the squared error loss $\ell(\theta_0, \hat{\theta}) = \|\theta_0 - \hat{\theta}\|_2^2$.

Since the observations $\mathbf{y}$ are random, generated according to $\mathbb{P}^N(\cdot | \mathbf{u}; \theta_0)$, the estimate $\hat{\theta} = T(\mathbf{u}, \mathbf{y})$ is also random. We therefore consider the expected loss, called the *risk*:

$$R(\theta_0, T) = \mathbb{E}_{\mathbf{y} \sim \mathbb{P}^N(\cdot | \mathbf{u}; \theta_0)}[\ell(\theta_0, T(\mathbf{u}, \mathbf{y}))]. \quad (2.3.3)$$

The risk $R(\theta_0, T)$ quantifies the average estimation error when using estimator T to estimate the true parameter θ_0 .

The challenge in designing an optimal estimator is that the risk (2.3.3) depends on θ_0 , which is precisely what we wish to estimate. An estimator that performs well for one value of θ_0 may perform poorly for another. To address this, we consider two classical approaches [1, 43]: Bayes and minimax estimation.

2.3.3 Bayes estimation

The Bayesian approach addresses the dependence on the unknown θ_0 by assuming a *prior* distribution π on Θ that encodes our beliefs about θ_0 before observing data. Instead of minimizing the risk for a specific θ_0 , we minimize the *average risk* (also called Bayes risk):

$$r(\pi, T) = \mathbb{E}_{\theta \sim \pi}[R(\theta, T)] = \int_{\Theta} R(\theta, T) \pi(\theta) d\theta. \quad (2.3.4)$$

The *Bayes estimator* (or Bayes rule) with respect to π is defined as

$$T_{\text{Bayes}}^* := \arg \min_{T \in \mathcal{T}} r(\pi, T), \quad (2.3.5)$$

where $\mathcal{T} := \{T: \mathcal{U} \times \mathcal{Y} \rightarrow \Theta\}$ denotes the class of all estimators. The Bayes estimate of θ_0 is then obtained by applying T_{Bayes}^* to the observations:

$$\hat{\theta}_{\text{Bayes}}^* = T_{\text{Bayes}}^*(\mathbf{u}, \mathbf{y}). \quad (2.3.6)$$

2.3.4 Minimax estimation

The minimax approach does not assume any prior distribution on θ_0 . Instead, it seeks an estimator that minimizes the *worst-case risk* over all possible values of θ_0 :

$$T_{\text{minimax}}^* := \arg \min_{T \in \mathcal{T}} \max_{\theta \in \Theta} R(\theta, T), \quad (2.3.7)$$

where $\mathcal{T} = \{T: \mathcal{U} \times \mathcal{Y} \rightarrow \Theta\}$. The minimax estimator provides a robust guarantee: regardless of the true value θ_0 , the risk is bounded by $\max_{\theta \in \Theta} R(\theta, T_{\text{minimax}}^*)$. The minimax estimate of θ_0 is obtained as

$$\hat{\theta}_{\text{minimax}}^* = T_{\text{minimax}}^*(\mathbf{u}, \mathbf{y}). \quad (2.3.8)$$

2.3.5 Summary: From estimator to estimate

Figure 2.3.2 summarizes the relationship between the estimator T and the estimate $\hat{\theta}$ of the unknown parameter θ_0 .

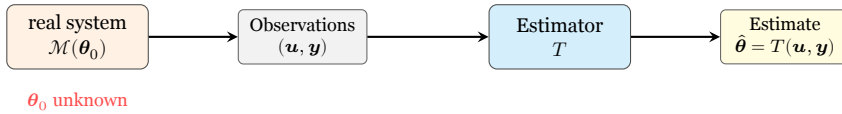


Figure 2.3.2: The estimation pipeline. Observations $(\mathbf{u}, \mathbf{y})$ from the real system $\mathcal{M}(\theta_0)$ are fed to an estimator T , which produces the estimate $\hat{\theta}$. The goal is to design T such that $\hat{\theta}$ is close to the unknown θ_0 .

Remark 2.1. The parameter estimation problem is also called *point estimation*. The value $\hat{\theta} = T(\mathbf{u}, \mathbf{y})$ is called the *point estimate* of θ_0 .

2.4 The Two-Stage Approach

The Two-Stage (TS) approach is a simulation-based parameter estimation procedure that works on the principle of *inverse supervised learning*. In standard supervised learning, one observes input-output pairs and learns a mapping from inputs to outputs; in the inverse setting, the roles are reversed: the parameters serve as labels and the observed data serve as inputs, so that the learned mapping predicts the parameters that generated a given data sample. The TS approach was originally proposed in [33] and an extended version appeared with applications to estimating the parameters of Pacejka’s tyre model in [34].

For a given model structure $\{\mathcal{M}(\theta): \theta \in \Theta\}$, a synthetic dataset is prepared by simulating the model for several parameter values. In particular, M parameter samples $\{\theta^{(i)}\}_{i=1}^M$ are drawn from Θ according to a distribution π . For each $\theta^{(i)}$, an input sequence $\mathbf{u}^{(i)} = (u_1^{(i)}, \dots, u_N^{(i)})^\top \in \mathbb{R}^N$ is applied to the simulator $\mathcal{M}(\theta^{(i)})$, which produces the output sequence $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_N^{(i)})^\top \in \mathbb{R}^N$.

Here, N denotes the number of samples in each input-output trajectory, and is typically large (in the order of 10^4 – 10^5). The input sequences are typically generated from a class of persistently exciting signals such as white noise or pseudo-random binary signals (PRBS).¹ A typical data-chart is shown in Table 2.4.1.

θ	$\mathbf{u}$	$\mathcal{M}(\theta)$	Data-chart
$\theta^{(1)}$	$\mathbf{u}^{(1)}$	$\mathbf{y}^{(1)} = (y_1^{(1)}, \dots, y_N^{(1)})^\top$	$(\theta^{(1)}, (\mathbf{u}^{(1)}, \mathbf{y}^{(1)}))$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\theta^{(M)}$	$\mathbf{u}^{(M)}$	$\mathbf{y}^{(M)} = (y_1^{(M)}, \dots, y_N^{(M)})^\top$	$(\theta^{(M)}, (\mathbf{u}^{(M)}, \mathbf{y}^{(M)}))$

Table 2.4.1: A typical data-chart in the TS approach. The synthetic dataset $\mathcal{D} = \{(\theta^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^M$ consists of M parameter-observation tuples, where each input-output pair $(\mathbf{u}^{(i)}, \mathbf{y}^{(i)})$ has length N .

Once this data-chart is prepared, an inverse map is constructed to learn an estimator. This inverse map is decomposed into two steps and hence the name “two-stage”: (i) data compression and (ii) function approximation.

First stage (data compression). The data compression step is achieved by a function $\varphi: \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^n$ which, upon receiving observations $\mathbf{y}^{(i)}$ from $\mathcal{M}(\theta^{(i)})$, compresses them into a smaller set of values called the *auxiliary statistic*:

$$\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \in \mathbb{R}^n, \quad \text{with } n \ll N. \quad (2.4.9)$$

The function φ depends implicitly on N through the dimension of its input arguments.² As remarked in [34], a well-chosen first-stage function makes the compressed representation approximately independent of the specific input realization. We thus derive a new data-chart $\{(\boldsymbol{\eta}^{(i)}, \theta^{(i)})\}_{i=1}^M$.

Second stage (function approximation). The derived data-chart is then fed to a supervised learning algorithm to obtain a function $\psi^* \in \mathcal{G}$, where $\mathcal{G}$ is a class of supervised learning models [4]. For example, $\mathcal{G}$ can be a class of (i) Deep Neural Networks (NNs) [44], (ii) Gradient Boosted Machines (GBMs) [45, 46], (iii) Linear regressors [4], (iv) Kernel regressors [4], and so on. In particular, ψ^* is obtained as the solution to the following optimization program:

$$\psi^* = \arg \min_{\psi \in \mathcal{G}} \frac{1}{M} \sum_{i=1}^M \ell(\theta^{(i)}, \psi(\boldsymbol{\eta}^{(i)})), \quad (2.4.10)$$

¹The superscript (i) in $\mathbf{u}^{(i)}$ indicates that different input realizations may be used for different parameter settings $\theta^{(i)}$. In practice, two common conventions are employed: (i) *common input*, where the same input sequence $\mathbf{u}$ is used for all M simulations, and (ii) *independent input*, where independent input realizations are generated for each simulation. The choice depends on the application; the TS framework accommodates both. When the same input is used, one may write $\mathbf{u}^{(i)} = \mathbf{u}$ for all i .

²When we wish to emphasize this dependence, we write φ_N instead of φ .

where ℓ denotes the loss function as introduced in (2.3.3). Eq. (2.4.10) is also known as Empirical Risk Minimization (ERM) [4].

The construction of ψ^* from the data-chart as shown in Table 2.4.1 constitutes the so-called *training phase* of the TS approach, as depicted in Fig. 2.4.3.

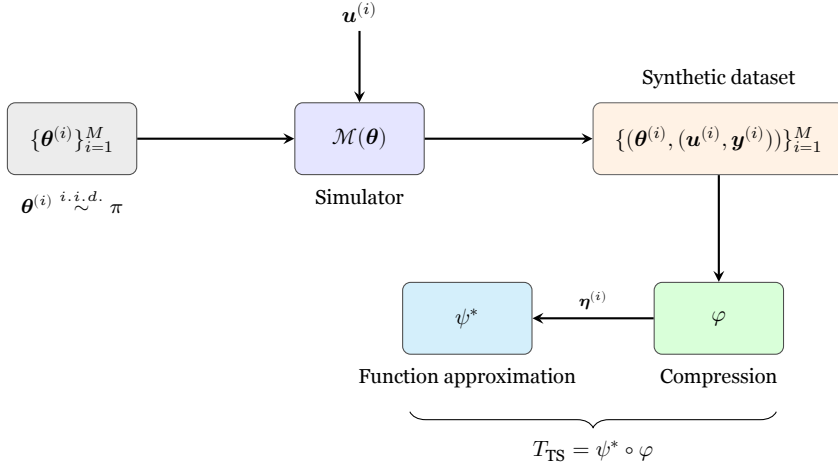


Figure 2.4.3: Training phase of the TS approach to parameter estimation. Parameters $\theta^{(i)}$ are sampled from Θ according to distribution π . For each $\theta^{(i)}$, the simulator $\mathcal{M}(\theta)$ receives input $u^{(i)}$ and generates output $y^{(i)}$, forming the synthetic dataset. The first-stage function φ compresses observations into auxiliary statistics $\eta^{(i)} = \varphi(u^{(i)}, y^{(i)})$, and the second-stage function ψ^* is obtained via function approximation. The resulting TS estimator is $T_{\text{TS}} = \psi^* \circ \varphi$.

The Two-Stage estimator. Once ψ^* is identified using (2.4.10), the TS estimator is obtained as the composition:

$$T_{\text{TS}} = \psi^* \circ \varphi, \quad (2.4.11)$$

where φ is fixed (chosen a priori) and ψ^* is learned from the synthetic data. The TS estimator $T_{\text{TS}}: \mathcal{U} \times \mathcal{Y} \rightarrow \Theta$ maps input-output pairs to the parameter space.

Implementation phase. Upon receiving new input and observations from the real system $\mathcal{M}(\theta_0)$, given by $u \in \mathcal{U}$ and $y = (y_1, \dots, y_N)^\top \in \mathcal{Y}$ respectively, the TS estimate of the true parameter θ_0 is computed as:

$$\hat{\theta}_{\text{TS}} = T_{\text{TS}}(u, y) = \psi^*(\varphi(u, y)). \quad (2.4.12)$$

This constitutes the *implementation phase* of the TS approach as shown in Fig. 2.4.4.

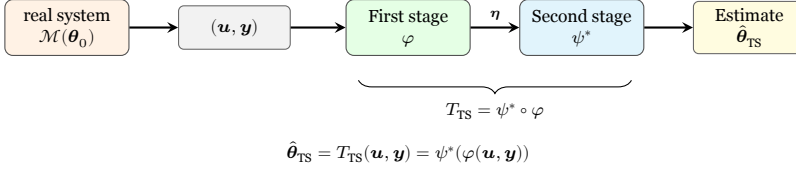


Figure 2.4.4: Implementation phase of the TS approach. Given new observations $(\mathbf{u}, \mathbf{y})$ from the real system $\mathcal{M}(\theta_0)$, the TS estimator $T_{\text{TS}} = \psi^* \circ \varphi$ is applied to obtain the estimate $\hat{\theta}_{\text{TS}}$ of the unknown parameter θ_0 .

2.5 Differential Privacy

One of the ways to impose privacy constraints in the point estimation problem is via differential privacy (DP). We shall precisely formulate the definition of DP, but to do so, we first need to define the notion of *neighbouring inputs*. For this, we let $\mathbf{y} = (y_1, \dots, y_N)^\top$ and $\mathbf{y}' = (y'_1, \dots, y'_N)^\top$ be elements of $\mathcal{Y} := \mathbb{R}^N$, called the *observation domain*.

Definition 2.5 (Hamming distance [47]). The Hamming distance between inputs $\mathbf{y}, \mathbf{y}'$ is defined as

$$d_H(\mathbf{y}, \mathbf{y}') := \sum_{i=1}^N \mathbb{1}\{y_i \neq y'_i\}.$$

Definition 2.6 (Neighbouring inputs [32]). $\mathbf{y}$ and $\mathbf{y}'$ are called *neighbouring inputs* if $d_H(\mathbf{y}, \mathbf{y}') = 1$, where d_H is the Hamming distance, i.e., if $y_i \neq y'_i$ for some unique $i \in \{1, \dots, N\}$, and $y_j = y'_j$ for all $j \in \{1, \dots, N\} \setminus \{i\}$.

Definition 2.7 (ε -Differential Privacy [32]). Let Θ be a subset of $\mathbb{R}^d$, $d \in \mathbb{N}$. For $\varepsilon > 0$, a randomized estimator $T: \mathcal{Y} \rightarrow \Theta$ is ε -*differentially private* (ε -DP) if for each pair of neighboring inputs $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$ and any measurable subset $A \subseteq \Theta$, it holds that

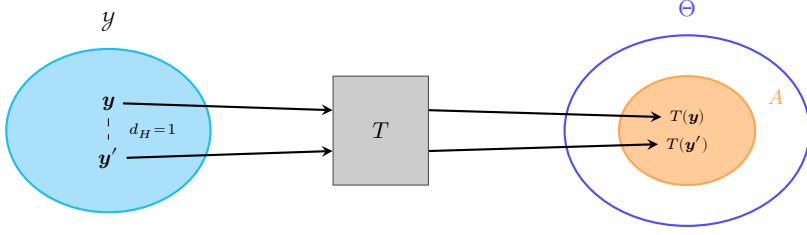
$$\mathbb{P}[T(\mathbf{y}) \in A] \leq e^\varepsilon \mathbb{P}[T(\mathbf{y}') \in A], \quad (2.5.13)$$

where $\mathbb{P}$ is the probability measure induced by the randomization in T .

Figure 2.5.5 provides a visual illustration of the ε -DP definition.

2.5.1 Interpretation

Definition 2.7 implies that for small values of ε , the probability distribution of the output $t = T(\mathbf{y})$ of the algorithm is almost the same (up to a multiplicative constant e^ε) for two neighbouring inputs $\mathbf{y}$ and $\mathbf{y}'$. Therefore, by looking at the output t , it is difficult to infer whether $\mathbf{y}$ or $\mathbf{y}'$ is its corresponding input, since the distribution of t is almost indistinguishable for $\mathbf{y}$ and $\mathbf{y}'$, thereby guaranteeing privacy.



$$\mathbb{P}[T(\mathbf{y}) \in A] \leq e^\epsilon \mathbb{P}[T(\mathbf{y}') \in A]$$

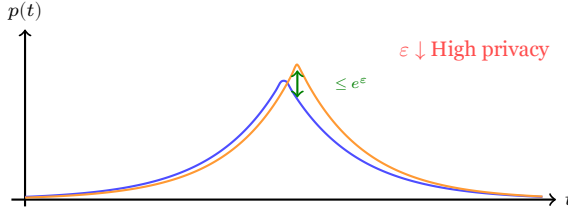


Figure 2.5.5: Illustration of ϵ -differential privacy. Top: Neighboring inputs $\mathbf{y}$ and $\mathbf{y}'$ (with Hamming distance $d_H = 1$) are processed by the randomized estimator T . The outputs $T(\mathbf{y})$ and $T(\mathbf{y}')$ lie in the parameter space Θ ; privacy requires that their probability of landing in any subset $A \subseteq \Theta$ differs by at most a factor of e^ϵ . Bottom: The output distributions for $\mathbf{y}$ and $\mathbf{y}'$ must be close, i.e., their ratio bounded by e^ϵ at every point. Smaller ϵ implies stronger privacy as the distributions become indistinguishable.

2.5.2 Laplace mechanism

A standard approach to achieve ϵ -DP is the *Laplace mechanism* [32], which we describe below. To this end, we need the notion of ℓ_1 -sensitivity.

Definition 2.8 (ℓ_1 -sensitivity [32]). The ℓ_1 -sensitivity of an estimator $T: \mathcal{Y} \rightarrow \Theta$ is

$$\sigma_T := \sup_{d_H(\mathbf{y}, \mathbf{y}')=1} \|T(\mathbf{y}) - T(\mathbf{y}')\|_1. \quad (2.5.14)$$

The ℓ_1 -sensitivity measures the maximum change in the estimate when a single observation is modified. This quantity determines the amount of noise required to guarantee privacy.

Definition 2.9 (Laplace mechanism [32]). Given a deterministic estimator $T: \mathcal{Y} \rightarrow \Theta$, the *Laplace mechanism* is a randomized algorithm that outputs the vector $\mathcal{A}(\mathbf{y})$ whose i^{th} component is distributed according to $[\mathcal{A}(\mathbf{y})]_i \stackrel{i.i.d.}{\sim} \text{Lap}([T(\mathbf{y})]_i, \frac{\sigma_T}{\epsilon})$, for $i = 1, \dots, d$. Here, $[T(\mathbf{y})]_i$ denotes the i^{th} component of the estimate $\hat{\theta} = T(\mathbf{y})$, and $\text{Lap}(a, b)$ is the Laplace distribution with probability

density function (pdf)

$$f_{\text{Lap}}(x; a, b) = \frac{1}{2b} \exp\left(-\frac{|x - a|}{b}\right), \quad (2.5.15)$$

where $b > 0$.

Notice from Definition 2.9 that $[\mathcal{A}(\mathbf{y})]_i = [T(\mathbf{y})]_i + \chi_i$, where $\chi_i \stackrel{i.i.d.}{\sim} \text{Lap}(0, \frac{\sigma_T}{\varepsilon})$, i.e., $\mathcal{A}(\mathbf{y}) = T(\mathbf{y}) + \boldsymbol{\chi}$, where $\boldsymbol{\chi} = (\chi_1, \dots, \chi_d)^\top$. Hence, DP is enforced by explicitly randomizing the estimate $\hat{\boldsymbol{\theta}} = T(\mathbf{y})$ via the addition of a Laplace noise vector $\boldsymbol{\chi}$ [32]. The resulting *private estimate* is $\hat{\boldsymbol{\theta}}_{\text{priv}} = \mathcal{A}(\mathbf{y}) = \hat{\boldsymbol{\theta}} + \boldsymbol{\chi}$. This process is illustrated in Figure 2.5.6.

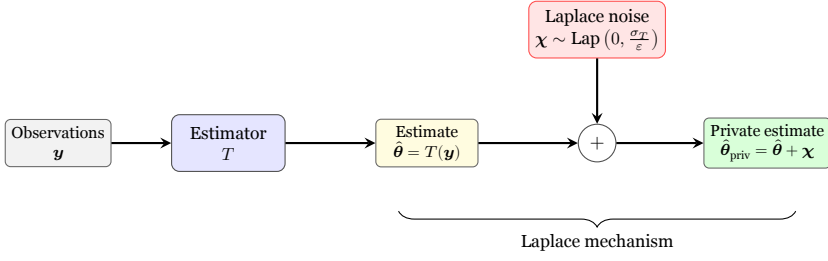


Figure 2.5.6: The Laplace mechanism for ε -differential privacy. The estimate $\hat{\boldsymbol{\theta}} = T(\mathbf{y})$ is randomized by adding Laplace noise $\boldsymbol{\chi}$ with scale σ_T/ε , where σ_T is the ℓ_1 -sensitivity of the estimator T . The resulting private estimate $\hat{\boldsymbol{\theta}}_{\text{priv}} = \hat{\boldsymbol{\theta}} + \boldsymbol{\chi}$ satisfies ε -DP.

Remark 2.2 (Privacy-accuracy trade-off). The scale of the Laplace noise is σ_T/ε . Thus, smaller ε (stronger privacy) requires larger noise, which degrades the accuracy of the estimate. Conversely, larger ε (weaker privacy) allows smaller noise and better accuracy. This fundamental trade-off between privacy and accuracy is a central theme in differential privacy.

Part 1: Analysis of the TS Estimator

Abbreviations

a.s.	almost surely
AR	autoregressive
ARMAX	autoregressive moving-average with exogenous input
ARX	autoregressive with exogenous input
BJ	Box-Jenkins
cdf	cumulative distribution function
CLT	central limit theorem
CRLB/CRB	Cramér-Rao lower bound
CVXPY	convex optimization library for Python
DNN	deep neural network
DT	digital twin
EKF	extended Kalman filter
FFNN	feed-forward neural network
FIM	Fisher information matrix
GBM	gradient boosted machine
GN	Gauss–Newton
LMC	Langevin-based Monte Carlo
i.i.d.	independent and identically distributed
LightGBM	Light Gradient Boosting Machine
MALA	Metropolis-adjusted Langevin algorithm
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MH	Metropolis–Hastings
MLE	maximum likelihood estimator

MSE	mean squared error
NLSS	nonlinear state-space
OE	output error
OOD	out-of-distribution
pdf	probability density function
PEM	prediction error method
PRBS	pseudo-random binary sequence
ResNet	residual network
r.v.	random variable
SDE	stochastic differential equation
SNR	signal-to-noise ratio
s.t.	subject to
TS	two-stage method
TSGBM	two-stage gradient boosted machine
w.r.t.	with respect to
ULA	Unadjusted Langevin Algorithm
XGBoost	eXtreme Gradient Boosting

Notation and Symbols

$\mathbb{R}^N$	The set of all N -dimensional real vectors
$\mathcal{N}(\mu, \sigma^2)$	The Gaussian distribution with mean μ and variance σ^2
$(\Omega, \mathcal{F}, \mathbb{P})$	Probability space
$\mathbb{E}[X]$	The expectation of a random variable X
$\mathbb{1}\{A\}$	The indicator function of an event A
$()^\top$	The transpose operator
$\ \cdot\ _2$	ℓ_2 norm
$\ \cdot\ _{\max}$	Max norm
∇f	The gradient of a function f
$\xrightarrow{p}$	Convergence in probability
$\xrightarrow{d}$	Convergence in distribution
$\xrightarrow{\text{a.s.}}$	Almost sure convergence
θ	A parameter vector
θ_0	The true parameter
$\hat{\theta}$	An estimate of the parameter θ
Θ	The parameter space
d	Dimension of the parameter space
$\mathcal{M}(\theta)$	The simulator/model parameterized by θ
$f(\cdot; \theta)$	The probability density function parameterized by θ
$\pi(\theta)$	Prior distribution over the parameter space
N	Length of observation sequence
M	Number of training samples

$\theta^{(i)}$	The i -th training parameter sample
$\mathbf{z}$	Input-output trajectory $(u_1, y_1, \dots, u_N, y_N)$
φ	First-stage compression function
n	Dimension of compressed features
ψ	Second-stage mapping function
T_{TS}	Two-Stage estimator
$\mathcal{J}_{\theta}$	Fisher Information Matrix
$\pi_J(\theta)$	Jeffreys prior
$\det(\cdot)$	The determinant of a matrix
$\ln(\cdot)$	The natural logarithm
$\exp(\cdot)$	The exponential function
$\mathcal{U}[a, b]$	The uniform distribution on $[a, b]$

3 Statistical Framework of the TS Estimator

Chapter 1 served as a motivation to explore simulation-driven parameter estimation and its potential for designing tuning rules for controllers. In Chapter 2, we provided essential background information on the TS approach, a modern simulation-driven estimator. The primary appeal of the TS approach is that its construction is entirely offline: once trained on synthetic data generated by a simulator, estimation at deployment reduces to a single forward pass, making it almost instantaneous. As illustrated in Section 2.4, the TS approach frames the estimation problem within an inverse supervised learning setup [4], wherein numerous samples are simulated for various parameter values, and any supervised learning method can be employed to predict parameter values for a given data sample. This supervised learning problem is divided into two stages: the first stage compresses the data into a concise set of values, while the second stage utilizes these compressed samples to predict parameter values. Despite its advantages over other approaches, TS relies on several user choices, whose impact on method performance is not yet fully understood.

In this chapter, we formalize TS by deriving it within a statistical decision theory framework. This enables us to recognize that the standard TS is essentially a Bayes estimator with respect to a given prior on the parameters, and we extend the approach to minimax decision rules. To facilitate understanding of the TS approach within the framework of statistical decision theory, we assume the data is independent and identically distributed (i.i.d.). We propose specific choices for the first and second stages corresponding to TS for i.i.d. data, leading to a simple convex optimization program that can be solved using off-the-shelf solvers.

3.1 Problem Setup

Recall from Section 2.3 that the goal in parameter estimation is to design an estimator T that minimizes the risk $R(\theta_0, T) = \mathbb{E}_{\mathbf{y} \sim \mathbb{P}^N(\cdot|\mathbf{u}; \theta_0)}[\ell(\theta_0, T(\mathbf{y}))]$, where $\mathbf{y} = (y_1, \dots, y_N)^\top \in \mathcal{Y} \subseteq \mathbb{R}^N$ are the observations, $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^N$ are the inputs corresponding to a physical, controlled SISO process with unknown system parameters θ_0 , and $\hat{\theta} = T(\mathbf{y})$ is an estimate of θ_0 . Here, $N \geq 1$ (typically large) denotes the number of samples at discrete time instants.

We consider parametric models $\mathcal{M}(\theta)$, where θ belongs to a parameter space $\Theta \subseteq \mathbb{R}^d$. The estimator is a mapping $T: \mathcal{Y} \rightarrow \Theta$ from the observation space to the parameter space.

In this chapter, we facilitate the analysis of the TS estimator by considering the case when the system is autonomous, i.e., it requires no input to produce an observation. We assume that the true system belongs to the model structure with $\theta_0 \in \Theta$, and hence observations are generated according to $\mathcal{M}(\theta_0)$. Further, we assume that $\mathbf{y} = (y_1, \dots, y_N)^\top$ is a random vector such that y_i 's are absolutely continuous [38], i.i.d. random variables with pdf $f(y; \theta)$, where $\theta \in \Theta$ is the parameter via which $\mathcal{M}(\theta)$ has generated y_i .

Given this setup, the risk of an estimator T at a parameter value $\theta \in \Theta$ is defined as

$$R(\theta, T) := \mathbb{E}_{\mathbf{y} \sim f^N(\mathbf{y}; \theta)}[\ell(\theta, T(\mathbf{y}))], \quad (3.1.1)$$

where $f^N(\mathbf{y}; \theta) = \prod_{i=1}^N f(y_i; \theta)$ due to the i.i.d. assumption. The goal of the TS approach is to produce an optimal estimator $T: \mathcal{Y} \rightarrow \Theta$ by minimizing either:

- **Bayes risk:** $\mathbb{E}_{\theta \sim f_\theta}[R(\theta, T)]$, where the expectation is taken with respect to a prior distribution f_θ supported on Θ ; or
- **Minimax risk:** $\max_{\theta \in \Theta} R(\theta, T)$, the worst-case risk over the parameter space.

As described in Section 2.4, the TS approach constructs the estimator T using a training dataset. During training, we draw M parameter samples $\theta^{(1)}, \dots, \theta^{(M)}$ from the parameter space Θ , and for each $\theta^{(i)}$, we simulate an N -length observation vector $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_N^{(i)})^\top$ from the model $\mathcal{M}(\theta^{(i)})$. This yields the training dataset $\{(\theta^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^M$, which is used to learn the estimator T .

In the following sections, we shall prove that the TS estimator reduces to either a Bayes or minimax estimator via the Bayes or minimax rule, respectively.

3.2 A Statistical Decision-Theoretical Formulation of TS

In this section, we will show, in a computational manner, that the TS estimator reduces to either a Bayes or minimax estimator, which then helps us to justify the structure of the TS estimator as a composition of a data compression step, followed by a function approximation step.

3.2.1 Bayes estimator

Expanding the expression for $R(\theta, T)$ in (3.1.1), we have

$$R(\theta, T) = \mathbb{E}_{\mathbf{y} \sim f^N(\mathbf{y}; \theta)} [\ell(\theta, T(\mathbf{y}))] = \int_{\mathbf{y}} \ell(\theta, T(\mathbf{y})) f^N(\mathbf{y}; \theta) d\mathbf{y}. \quad (3.2.2)$$

Now, we define the *weighted risk*

$$R_{\text{weighted}}(T) = \int_{\Theta} R(\theta, T) \pi(\theta) d\theta, \quad (3.2.3)$$

where $\pi: \Theta \rightarrow \mathbb{R}_0^+$ is a weighting function over Θ . Note that π need not be a probability density function (*pdf*) over Θ , that is, $\int_{\Theta} \pi(\theta) d\theta$ does not need to be equal to 1 (or even finite). However, if $\pi(\theta) = f_{\theta}$ happens to be a pdf over Θ , then we obtain the definition of *Bayes risk*,

$$R_{\text{Bayes}}(T) := \int_{\Theta} R(\theta, T) f_{\theta} d\theta, \quad (3.2.4)$$

where f_{θ} is called a *prior* on θ , supported on Θ . The minimizer of R_{Bayes} is called the *Bayes rule* associated with f_{θ} .

The standard approach to derive the Bayes rule is as follows. We first expand (3.2.4):

$$R_{\text{Bayes}}(T) = \int_{\Theta} \int_{\mathbf{y}} \ell(\theta, T(\mathbf{y})) f(\theta|\mathbf{y}) d\theta f^N(\mathbf{y}) d\mathbf{y}, \quad (3.2.5)$$

where,

$$f(\theta|\mathbf{y}) := \frac{f^N(\mathbf{y}|\theta) f_{\theta}}{f^N(\mathbf{y})} = \frac{f^N(\mathbf{y}|\theta) f_{\theta}}{\int_{\Theta} f^N(\mathbf{y}|\theta) f_{\theta} d\theta}. \quad (3.2.6)$$

It is clear from (3.2.5) that, in order to minimize R_{Bayes} we have to minimize the inner integral $\int_{\Theta} \ell(\theta, T(\mathbf{y})) f(\theta|\mathbf{y}) d\theta$ over T , since $f^N(\mathbf{y})$ is non-negative.

Therefore, the Bayes decision rule is given by

$$T_{\text{Bayes}}^* = \arg \min_{T: \mathbf{y} \rightarrow \Theta} \int_{\Theta} \ell(\theta, T(\mathbf{y})) f(\theta|\mathbf{y}) d\theta. \quad (3.2.7)$$

We now propose an alternative computational approach to derive the Bayes rule associated with a prior f_θ , which essentially corresponds to the TS method [33, 34]. To this end, note that

$$R_{\text{Bayes}}(T) \approx \int_{\Theta} \left[\frac{1}{M_{\mathbf{y}}} \sum_{j=1}^{M_{\mathbf{y}}} \ell(\theta, T(\mathbf{y}_j)) \right] f_\theta d\theta, \quad (3.2.8)$$

where $\mathbf{y}_1, \dots, \mathbf{y}_{M_{\mathbf{y}}}$ are independent random samples from $f^N(\mathbf{y}; \theta)$. We can further approximate the outer integral of (3.2.8) by Monte-Carlo sampling, by taking M independent samples $\theta^{(1)}, \dots, \theta^{(M)}$ from f_θ :

$$R_{\text{Bayes}}(T) \approx \frac{1}{M_{\mathbf{y}}} \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^{M_{\mathbf{y}}} \ell(\theta^{(i)}, T(\mathbf{y}_j^{(i)})), \quad (3.2.9)$$

where $\mathbf{y}_j^{(i)} \stackrel{i.i.d.}{\sim} f^N(\mathbf{y}; \theta^{(i)})$ and $\theta^{(i)} \stackrel{i.i.d.}{\sim} f_\theta$, for $i = 1, \dots, M$ and $j = 1, \dots, M_{\mathbf{y}}$.

From (3.2.9), it follows that one can approximate the Bayes rule by solving the optimization problem

$$\tilde{T}_{\text{Bayes}} := \arg \min_{T \in \mathcal{T}} \frac{1}{M_{\mathbf{y}}} \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^{M_{\mathbf{y}}} \ell(\theta^{(i)}, T(\mathbf{y}_j^{(i)})), \quad (3.2.10)$$

where $\mathcal{T} := \{\mathcal{Y} \rightarrow \Theta\}$ is a suitable function space. If $M_{\mathbf{y}}, M$ are sufficiently large, and $\mathcal{T}$ is sufficiently large, one should expect that $\tilde{T}_{\text{Bayes}} \approx T_{\text{Bayes}}^*$ in a suitable sense.

Choosing the function space

The next step is to choose a function space $\mathcal{T}$. To this end, notice that typically $\mathcal{Y}$ is a product space, i.e., $\mathcal{Y} = \tilde{\mathcal{Y}}^N$ due to the i.i.d. assumption. This means that $\mathbf{y} \in \mathcal{Y}$ is of the form $\mathbf{y} = (y_1, \dots, y_N)^\top$, where $y_i \in \tilde{\mathcal{Y}}$ ($i = 1, \dots, N$) are the individual *samples* in the data, and N is the *sample size*.

A motivation to consider a two-stage approach is to better isolate the information in $\mathbf{y}^{(i)}$ about $\theta^{(i)}$ from the effect of noise [34], so that the second-stage function can be learned while bypassing the bias-variance trade-off. In addition, directly learning a mapping from the raw observation vector $\mathbf{y} \in \mathbb{R}^N$ to the parameter space Θ using complex models presents certain challenges:

- **High dimensionality and overfitting:** The observation vector $\mathbf{y} = (y_1, \dots, y_N)^\top$ is N -dimensional, where N is typically large. Learning a mapping directly from such high-dimensional inputs to Θ requires a

very large number of training samples M to avoid overfitting, a manifestation of the curse of dimensionality. Compressing $\mathbf{y}$ into a low-dimensional auxiliary statistic $\boldsymbol{\eta} \in \mathbb{R}^n$ with $n \ll N$ mitigates this issue, making the second-stage learning problem substantially easier.

- **Computational expense:** Complex models like expressive deep neural networks have a large number of parameters that scale with the dimension of their inputs (as well as with their number of layers, and other factors). Thus, training such models directly on large sequences of data can be very computationally demanding. The first-stage compression reduces the input dimension from N to n , significantly lowering the computational cost of training the second stage.

This motivates a two-stage approach: first applying a low-dimensional representation, followed by a simpler model to balance the bias-variance trade-off optimally. Precisely, we express T as the composition of two functions,

$$T_{\text{TS}} = \psi \circ \varphi, \quad (3.2.11)$$

where $\varphi: \mathcal{Y} \rightarrow \mathbb{R}^n$ and $\psi: \mathbb{R}^n \rightarrow \Theta$, with $n \ll N$. For identifiability purposes [48], we require that N is larger than n .

The first-stage function φ can be chosen as a fixed function (i.e., not to be optimized) with the property that if $\mathbf{y} \sim f^N(\mathbf{y}; \boldsymbol{\theta})$, then $\varphi(\mathbf{y}) \rightarrow \bar{\boldsymbol{\eta}}$ a.s. as $N \rightarrow \infty$, where $\bar{\boldsymbol{\eta}}$ is a deterministic quantity that depends on $\boldsymbol{\theta}$. This means that the first-stage function φ handles the high-dimensional input $\mathbf{y}$ as N grows, while the second-stage function ψ can be designed independently of N on the low-dimensional auxiliary statistic $\phi(\mathbf{y})$.

A natural choice for φ is to use an estimator for a *misspecified model* of $\mathbf{y}$, i.e., a model which is not necessarily related to $f^N(\mathbf{y}; \boldsymbol{\theta})$, but which is nevertheless easy to estimate, for instance, using least squares [6]. The auxiliary statistic $\varphi(\mathbf{y})$ should serve as an *approximate sufficient statistic* [49] for $\boldsymbol{\theta}$, in the sense of containing, approximately, the same information about $\boldsymbol{\theta}$ as $\mathbf{y}$. This resembles the use of misspecified models in techniques such as Indirect Inference [50].

Once φ is chosen, we are left with the problem of determining ψ according to

$$\psi^* = \arg \min_{\psi \in \mathcal{G}} \frac{1}{M_{\mathbf{y}}} \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^{M_{\mathbf{y}}} \ell(\boldsymbol{\theta}^{(i)}, \psi(\boldsymbol{\eta}_j^{(i)})), \quad (3.2.12)$$

where $\boldsymbol{\eta}_j^{(i)} = \varphi(\mathbf{y}_j^{(i)})$. Here, $\mathcal{G}$ is a function space that can be chosen according to general functional approximation methods, such as Deep Neural Networks (DNNs), Gradient Boosted Machines (GBMs), radial basis functions, or finite-dimensional function spaces (e.g., polynomials, piecewise affine functions). We call the final estimator $T_{\text{TS}} := \psi^* \circ \varphi$ the *Bayes TS estimator*.

Under i.i.d. or stationarity assumptions, if N is sufficiently large, each sample will give a good estimate of the loss (due to a concentration of measure), so we can take $M_y = 1$ without significant loss in performance.

The approach given here to approximate T_{Bayes}^* is essentially identical to the TS approach given in [34], where the first stage consists in the computation of $\eta^{(i)} = \varphi(\mathbf{y}^{(i)})$, called the “compression stage”, and in the second stage ψ^* is determined. However, our formulation clarifies that the way how the values of θ are sampled via f_θ affects the resulting estimate, which is in fact an approximate Bayes estimator.

3.2.2 Minimax estimator

In this subsection we modify the previous formulation in order to arrive at a minimax estimator. The minimax estimator minimizes the worst-case risk over the parameter space Θ :

$$T_{\text{minimax}}^* = \arg \min_{T \in \mathcal{T}} \left[\max_{\theta \in \Theta} R(\theta, T) \right]. \quad (3.2.13)$$

By using the expression for risk $R(\theta, T)$ (cf. (3.2.2)), we have

$$\max_{\theta \in \Theta} R(\theta, T) = \max_{\theta \in \Theta} \int_{\mathbf{y}} \ell(\theta, T(\mathbf{y})) f^N(\mathbf{y}; \theta) d\mathbf{y}. \quad (3.2.14)$$

The integral in (3.2.14) can be approximated using Monte-Carlo sampling as

$$\int_{\mathbf{y}} \ell(\theta, T(\mathbf{y})) f^N(\mathbf{y}; \theta) d\mathbf{y} \approx \frac{1}{M_y} \sum_{j=1}^{M_y} \ell(\theta, T(\mathbf{y}_j)), \quad (3.2.15)$$

where $\mathbf{y}_j \stackrel{i.i.d.}{\sim} f^N(\mathbf{y}; \theta)$ for a fixed θ , and M_y is the number of samples generated according to $f^N(\mathbf{y}; \theta)$. Then, we can use (3.2.15) in (3.2.14) to obtain the approximate optimization problem

$$\min_{T \in \mathcal{T}} \max_{\theta \in \Theta} \left[\frac{1}{M_y} \sum_{j=1}^{M_y} \ell(\theta, T(\mathbf{y}_j)) \right]. \quad (3.2.16)$$

In contrast to Bayes rules, we will now apply the *scenario approach* [51] to transform (3.2.16) into a tractable optimization problem by replacing the (potentially infinite) parameter space Θ by a finite set of sampled points. To this end, let $s(\theta)$ be a sampling distribution over Θ , and consider M samples

$\theta^{(i)} \stackrel{i.i.d.}{\sim} s(\theta)$, $i = 1, \dots, M$. By ignoring the constant factor $1/M_y$ in (3.2.16), and taking $M_y = 1$, we can further approximate (3.2.16) as

$$\min_{T \in \mathcal{T}} \max_{i=1, \dots, M} \ell(\theta^{(i)}, T(\mathbf{y}^{(i)})). \quad (3.2.17)$$

By using the epigraph formulation of this optimization problem [52, page 134], we have that (3.2.17) is equivalent to

$$\begin{aligned} \min_{T \in \mathcal{T}, t \in \mathbb{R}} \quad & t \\ \text{s.t.} \quad & \ell(\theta^{(i)}, T(\mathbf{y}^{(i)})) \leq t, \quad i = 1, \dots, M. \end{aligned} \quad (3.2.18)$$

The solution to (3.2.18) is the *minimax TS estimator*, again denoted T_{TS} . Its form can be taken, as in the Bayesian case, as $T_{\text{TS}} = \psi^* \circ \varphi$.

Finally, we will consider a very special case of the minimax TS estimator, where $d = 1$, the loss function is the *mean square error* $\ell(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$, where $\hat{\theta}$ is an estimate of θ , and ψ is chosen from a class of functions that can be expressed as linear regression on nonlinear features.¹

Polynomial regression in the second stage

To construct a tractable second stage, we introduce a *feature map* $J: \mathbb{R}^n \rightarrow \mathbb{R}^p$ with $n < p \ll N$, which transforms the auxiliary statistic $\boldsymbol{\eta} := \varphi(\mathbf{y})$ into a higher-dimensional feature vector. A common choice is to let $J(\boldsymbol{\eta})$ consist of polynomial features, i.e., monomials of the components of $\boldsymbol{\eta}$ up to a specified degree. The second-stage function is then parameterized as

$$\psi(\boldsymbol{\eta}) = \beta^\top J(\boldsymbol{\eta}), \quad (3.2.19)$$

where $\beta \in \mathbb{R}^p$ is a vector of regression coefficients. This formulation corresponds to *polynomial regression* (or, more generally, *kernel ridge regression* when J is viewed as an implicit feature map). The optimization over the function class $\mathcal{G}$ thus reduces to optimizing over the finite-dimensional parameter β .

Furthermore, $T(\mathbf{y}^{(i)}) = \beta^\top J(\boldsymbol{\eta}^{(i)})$, where $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{y}^{(i)})$. Then, we have that

$$\begin{aligned} \ell(\theta^{(i)}, T(\mathbf{y}^{(i)})) &= (\theta^{(i)} - \beta^\top J(\boldsymbol{\eta}^{(i)}))^2 \\ &= \beta^\top (J(\boldsymbol{\eta}^{(i)})J(\boldsymbol{\eta}^{(i)})^\top) \beta - 2\theta^{(i)} J(\boldsymbol{\eta}^{(i)})^\top \beta + (\theta^{(i)})^2. \end{aligned} \quad (3.2.20)$$

¹This special case can be easily extended to more than one parameter, i.e., $d > 1$, by applying TS to estimate each parameter separately.

Substituting (3.2.20) in (3.2.18) we obtain the quadratically constrained quadratic program (QCQP):

$$\begin{aligned} \min_{\beta \in \mathbb{R}^p, t \in \mathbb{R}} \quad & t \\ \text{s.t.} \quad & \beta^\top (J(\boldsymbol{\eta}^{(i)})J(\boldsymbol{\eta}^{(i)})^\top) \beta - 2\theta^{(i)}J(\boldsymbol{\eta}^{(i)})^\top \beta + (\theta^{(i)})^2 \leq t, \\ & i = 1, \dots, M. \end{aligned} \quad (3.2.21)$$

This optimization problem is convex and can be solved using a standard solver, such as CVXPY [53].

3.3 Two-Stage Approach for i.i.d. Data

To properly account for the *i.i.d.* assumption, we notice that under it, reasonable decision rules are “permutation-invariant” [48], that is, their output should not vary if the entries of $\mathbf{y}$ are permuted. Such decision rules can be shown [48] to be functions of the *order statistics* of $y_1, \dots, y_N$, which correspond to the values of $y_1, \dots, y_N$ sorted in ascending order, and are denoted by $y_{(1)}, \dots, y_{(N)}$.

We suggest to compress the data starting from its order statistics. To this end, we will use some *quantiles* of the order statistics: If $0 < \gamma < 1$, then the (100γ) -th sample quantile is a linear interpolation between $y_{(\lfloor N\gamma \rfloor)}$ and $y_{(\lceil N\gamma \rceil)}$. The computation of a small and fixed number of quantiles as the compressed data constitutes the first stage of TS, and the description of the first stage is given below.

3.3.1 First stage

For the first stage, we set $\gamma = k/n$, where $k = 1, \dots, n$, and obtain n quantiles. We then collect those quantiles into the auxiliary statistic $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n)^\top = \varphi(\mathbf{y})$. The first stage for both Bayes TS and minimax TS is the same.

3.3.2 Second stage

For both formulations, the second stage starts by constructing nonlinear features $J(\boldsymbol{\eta})$ using the auxiliary statistic $\boldsymbol{\eta}$ and a fixed feature map $J: \mathbb{R}^n \rightarrow \mathbb{R}^p$ with $n < p \ll N$. The feature map J must be chosen by the user, and it depends on the type of parameter(s) that we want to estimate. In both formulations, we then parameterize ψ as in (3.2.19), i.e., $\psi(\boldsymbol{\eta}) = \beta^\top J(\boldsymbol{\eta})$, where β is determined as explained in Section 3.2. If $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)^\top$, where $d > 1$, is the parameter vector to be estimated, then we can design a separate second-stage function $\psi_k(\boldsymbol{\eta}) = \beta_k^\top J(\boldsymbol{\eta})$ for each $k = 1, \dots, d$. To avoid numerical issues, in both Bayes and minimax formulations we add a regularization term of the form $\lambda_r \|\beta\|_2^2$ to the optimization problem to be solved, where $\lambda_r > 0$ is a small constant (e.g., 10^{-8}).

Remark 3.1. Both n (the number of quantiles) and $J(\cdot)$ (the feature map) are user choices, which need to be selected according to the specific estimation problem.

3.4 Simulations

In this section, we illustrate the performance of the Bayes and minimax TS estimators derived in Section 3.2 on the estimation of the two parameters of a Weibull distribution. Weibull distributions are widely used in reliability engineering [54, 55] to model the probability of failure of equipment at a particular age, and are parameterized by: (i) a *scale* parameter, denoted by $\lambda > 0$, and (ii) a *shape* parameter, denoted by $\kappa > 0$. Here, λ and κ are unknown. Hence, this is a two-dimensional parameter estimation problem, with $\theta = (\lambda, \kappa)^\top$ being the unknown parameter. As described in Section 3.3.2, we design separate second-stage functions for λ and κ , yielding estimates denoted by $\hat{\lambda}$ and $\hat{\kappa}$ respectively.

The Weibull probability density function is

$$f_{\text{Weibull}}(x; \lambda, \kappa) = \frac{\kappa}{\lambda} \left(\frac{x}{\lambda}\right)^{\kappa-1} \exp\left[-\left(\frac{x}{\lambda}\right)^\kappa\right], \quad x \geq 0. \quad (3.4.22)$$

We follow the TS approach described in Section 3.3, where for the first stage we consider $n = 10$ quantiles, and for the second stage, we choose the feature map $J(\eta)$ according to the following:

- For scale parameter λ ,

$$J_i(\eta) = \begin{cases} \eta_i, & \text{if } 1 \leq i \leq n \\ \frac{\eta_{i-n+1}}{\eta_1}, & \text{if } n+1 \leq i \leq 2n-1. \end{cases} \quad (3.4.23)$$

The reason for choosing these combinations of quantiles to estimate λ is that quotients of the (theoretical) quantiles of a distribution are proportional to its scale parameter (when the other parameters are kept fixed), which suggests that the quantities $J_i(\eta)$ above can be used for estimating λ .

- For shape parameter κ , let

$$\xi_j(\eta) = \begin{cases} \eta_j, & \text{if } 1 \leq j \leq n \\ \frac{\eta_{j-n}}{\eta_n}, & \text{if } n+1 \leq j \leq 2n-1. \end{cases} \quad (3.4.24)$$

We let $J(\eta)$ consist of monomials of the $\xi_j(\eta)$'s up to order 2. This choice is suggested by the fact that a small increase in κ (for $\kappa > 1$) leads to a similar change of the Weibull distribution as a small decrease in λ .

We should remark that for the Weibull distribution there exist exact expressions for λ and κ in terms of any two quantiles [56]. However, using such expressions together with sample quantiles may lead to sub-optimal estimates, since they ignore the variability of the sample quantiles.

We will now look at results corresponding to both Bayes and minimax frameworks discussed in Section 3.2.

3.4.1 Bayes estimator

To illustrate the Bayes TS approach of Section 3.2.1, we study the effect of choosing different priors for the parameter θ . In particular, two choices of prior are considered:

1. Independent uniform priors $\text{unif}[1, 20]$ for λ and κ .
2. Independent *uninformative* priors for λ and κ with density [57]

$$f(x) = \begin{cases} \frac{c}{x}, & \text{if } a \leq x \leq b \\ 0, & \text{otherwise,} \end{cases} \quad (3.4.25)$$

where $c = 1/\ln(b/a)$. Here, we take $a = 1$ and $b = 20$ for both λ and κ .

For each of the above priors, we find the Bayes TS estimator by drawing 1000 values of $\theta = (\lambda, \kappa)^\top$ independently. For each sample of θ , 10000 observations are drawn according to the Weibull distribution parameterized by θ . Once an estimator is found for each of the above priors, we evaluate its performance by drawing 1000 new values of the true parameter $\theta_0 = (\lambda_0, \kappa_0)^\top$ (different from those used during training), and for each θ_0 , 10000 observations are drawn according to the Weibull distribution parameterized by θ_0 .

We show scatter plots of estimated values of λ and κ vs. their true values in Figures 3.4.1-3.4.4. Figures 3.4.1 and 3.4.2 correspond to the uniform prior, whereas Figures 3.4.3 and 3.4.4 correspond to the uninformative prior given by (3.4.25).

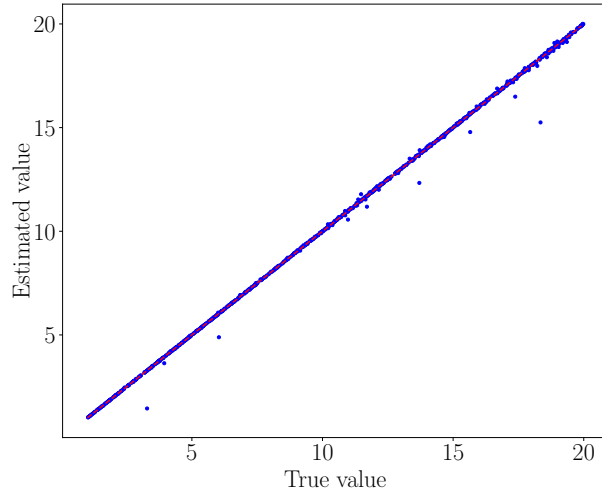


Figure 3.4.1: Scatter plot of Bayes TS estimates of λ vs. its true value for a uniform prior.

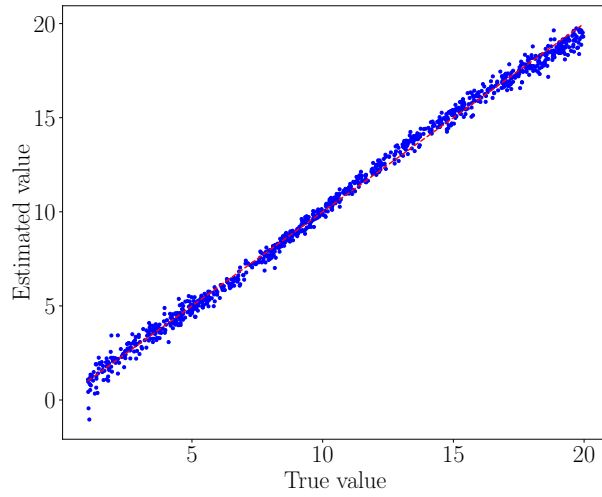


Figure 3.4.2: Scatter plot of Bayes TS estimates of κ vs. its true value for a uniform prior.

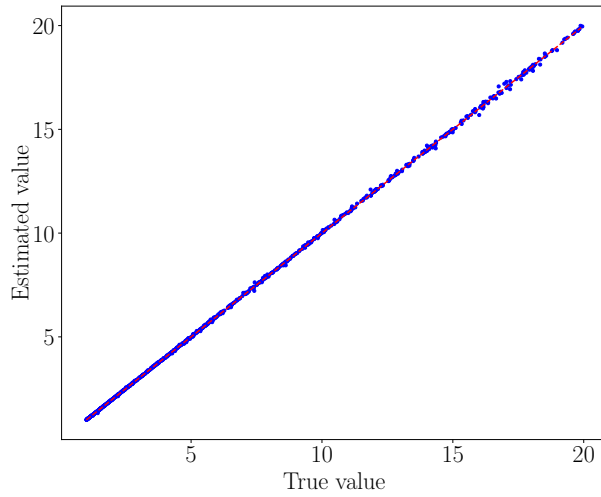


Figure 3.4.3: Scatter plot of Bayes TS estimates of λ vs. its true value for an uninformative prior.

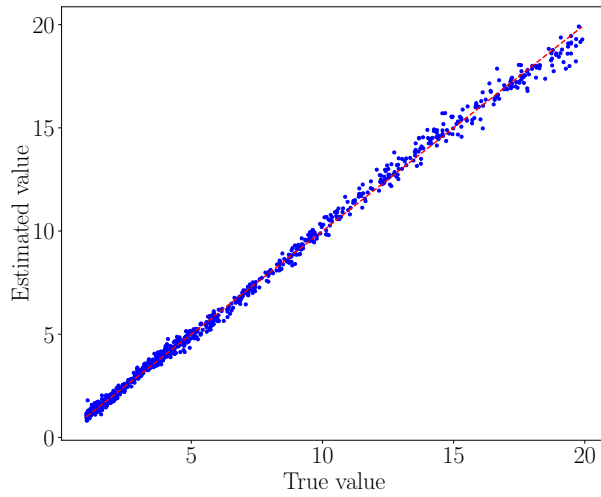


Figure 3.4.4: Scatter plot of Bayes TS estimates of κ vs. its true value for an uninformative prior.

3.4.2 Minimax estimator

To illustrate the minimax TS approach of Section 3.2.2, we find it by drawing 1000 values of $\theta = (\lambda, \kappa)^\top$ independently from a proposal uniform $\mathcal{U}[1, 20]$ distribution. For each sample of θ , 10000 observations are drawn according to the

Weibull distribution parameterized by θ . Once the estimator is found, we evaluate its performance by drawing 1000 new values of $\theta_0 = (\lambda_0, \kappa_0)^\top$, and for each θ_0 , 10000 observations are drawn according to the Weibull distribution parameterized by θ_0 . We show scatter plots of estimated values of λ and κ vs. their true values in Figures 3.4.5 and 3.4.6.

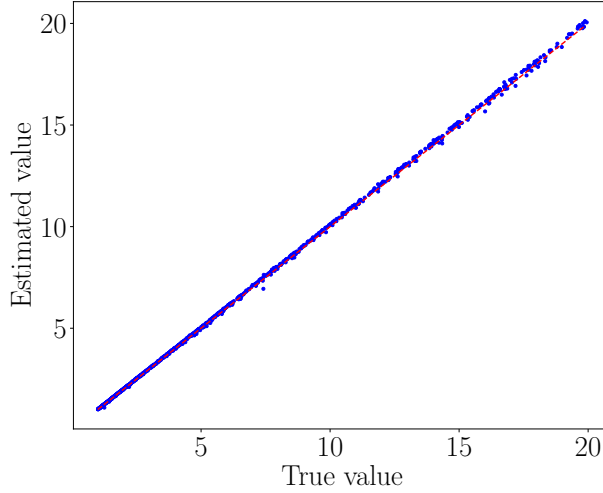


Figure 3.4.5: Scatter plot of minimax TS estimates of λ vs. its true value.

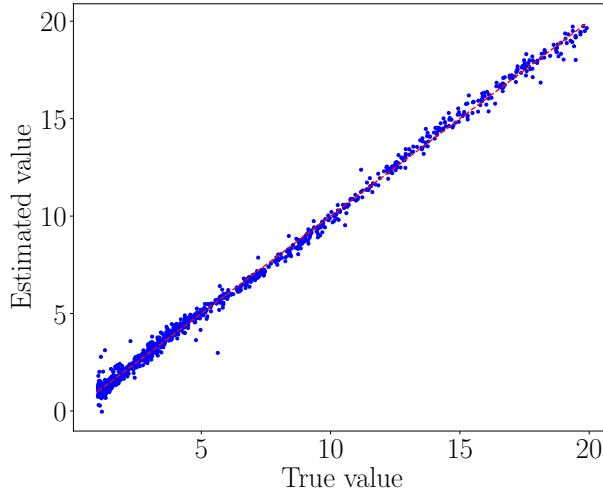


Figure 3.4.6: Scatter plot of minimax TS estimates of κ vs. its true value.

To assess the quality of the Bayes and minimax TS estimators of λ and κ , we have numerically evaluated the *mean square error* (MSE) of the TS estimators, based on 1000 Monte Carlo simulations, and compared them with a numerical evaluation of their corresponding Cramér-Rao lower bounds (CRLB) [58]. The results are shown in Table 3.4.1.

True Values		CRLB		MSE, Bayesian (Uniform Prior)		MSE, Bayesian (Uninformative Prior)		MSE, Minimax	
λ	κ	$\hat{\lambda}$	$\hat{\kappa}$	$\hat{\lambda}$	$\hat{\kappa}$	$\hat{\lambda}$	$\hat{\kappa}$	$\hat{\lambda}$	$\hat{\kappa}$
2	2	1.11×10^{-4}	2.43×10^{-4}	2.58×10^{-4}	5.77×10^{-2}	1.42×10^{-4}	2.06×10^{-2}	2.17×10^{-4}	16×10^{-2}
2	8	6.93×10^{-6}	3.89×10^{-3}	1.11×10^{-5}	5.61×10^{-2}	1.27×10^{-5}	4.44×10^{-2}	4.28×10^{-4}	13.29×10^{-2}
4	2	4.43×10^{-4}	2.43×10^{-4}	6.74×10^{-4}	1.05×10^{-1}	6.07×10^{-4}	2.43×10^{-2}	8.38×10^{-4}	14.67×10^{-2}
4	8	2.77×10^{-5}	3.89×10^{-3}	3.84×10^{-5}	6.40×10^{-2}	4.33×10^{-5}	3.96×10^{-2}	1.72×10^{-3}	8.59×10^{-2}
8	2	1.77×10^{-3}	2.43×10^{-4}	2.26×10^{-3}	1.89×10^{-1}	2.27×10^{-3}	2.35×10^{-2}	3.307×10^{-3}	18.605×10^{-2}
8	8	1.11×10^{-4}	3.89×10^{-3}	1.58×10^{-4}	7.901×10^{-2}	1.76×10^{-4}	4.51×10^{-2}	6.77×10^{-3}	8.39×10^{-2}

Table 3.4.1: MSE of Bayes and minimax TS estimators of the scale (λ) and shape (κ) parameters, and their corresponding CRLBs.

One can see from Table 3.4.1 the effect of changing priors in the Bayes estimator: For the *uninformative* priors, the estimates of λ and κ are relatively much closer to the true values than for the uniform priors. It is also interesting to note that, while no prior is needed for the minimax estimator, one can still get reliable estimates, as demonstrated in Figures 3.4.5 and 3.4.6.

By more careful “feature engineering” of the feature map J in the second stage for $\hat{\kappa}$, we could obtain more reliable estimates of the shape parameter without using more complex functional approximators such as DNNs.

3.5 Conclusion

In this chapter, we have provided a decision-theoretical framework for the TS paradigm to estimate the parameters of a data generating model. In particular, we have derived Bayes and minimax formulations of TS, and showed how to implement them for *i.i.d.* data, by considering nonlinear functions of the sample quantiles of the data as a first stage, and then optimizing for polynomial regression coefficients as the second stage. We have also evaluated the performance of these novel estimators on simulated data from a Weibull distribution.

4 Improved Minimax TS Estimator

The theoretical underpinnings of the TS approach to parameter estimation were discussed in Chapter 3, where two statistical frameworks were explored to derive the TS estimator: (i) Bayes and (ii) Minimax estimation. The derivations of these frameworks were conducted under the assumption of i.i.d. data samples, although they can be readily extended to parameter estimation of dynamical systems where the data is non-i.i.d. Under the i.i.d. assumption, the first stage can be constructed from quantiles of the observations, demonstrating that the TS approach provides reliable estimates for both Bayes and minimax estimates of the unknown parameters.

This chapter specifically focuses on the minimax TS framework, which is both practically and theoretically appealing because, unlike Bayes estimators, minimax estimators are robust to prior knowledge. However, the minimax TS approach outlined in Chapter 3 encounters challenges in terms of computational time required to obtain reliable estimates. This is primarily due to (i) the need to run an optimization algorithm for each sample of parameter values, with a large number of such samples, and (ii) the majority of constraints in the epigraph formulation, utilized in Chapter 3 to solve the minimax problem, being inactive, resulting in slow solver performance.

To mitigate these computational challenges, this chapter employs a Gradient Boosted Machine (GBM) as the supervised learning algorithm in the second stage of the TS approach. A GBM is a non-linear function approximator constructed based on decision or regression trees [59, 45, 60]. By carefully designing the cost function of the GBM and modifying the evaluation metric using a soft-max approximation, the minimax problem can be efficiently solved.

This chapter specifically adopts a GBM as the supervised learning algorithm in the second stage of the TS approach. Here, we adapt the evaluation metric of the

GBM using a soft-max approximation, enabling efficient solution of the minimax objective. The effectiveness of this proposed approach in implementing the minimax statistical framework is then numerically validated across several examples, including dynamical systems where the data generation mechanism is non-i.i.d.

4.1 Problem Setup

In this chapter, our focus is on the minimax TS estimator, which was derived in Chapter 3. Recall that the TS estimator has the form $T_{\text{TS}} = \psi \circ \varphi$, where $\varphi: \mathcal{Y} \rightarrow \mathbb{R}^n$ is the first-stage function that compresses the observations $\mathbf{y}^{(i)}$ into an auxiliary statistic $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{y}^{(i)})$, and $\psi: \mathbb{R}^n \rightarrow \Theta$ is the second-stage function that maps the auxiliary statistic to the parameter space.

Let us recall from Eq. (3.2.17) the approximate minimax optimization program that constructs the minimax TS estimator as

$$\min_{T \in \mathcal{T}} \max_{i=1, \dots, M} \ell(\boldsymbol{\theta}^{(i)}, T(\mathbf{y}^{(i)})). \quad (4.1.1)$$

Using an epigraph formulation [52, page 134], we obtain the equivalent optimization problem

$$\begin{aligned} \min_{T \in \mathcal{T}, t \in \mathbb{R}} \quad & t \\ \text{s.t.} \quad & \ell(\boldsymbol{\theta}^{(i)}, T(\mathbf{y}^{(i)})) \leq t, \quad i = 1, \dots, M. \end{aligned} \quad (4.1.2)$$

As we notice from Figure 3.4.6 in Chapter 3, the estimates have high variance for small values of the parameters. This might be due to the fact that the epigraph formulation is sensitive to the number of training samples M , and hence we need typically large M to obtain reliable estimates with low variance. To overcome this issue, we will directly solve objective (4.1.1) and the method to solve it is the main theme of this chapter.

To this end, we propose the Two-Stage Gradient Boosted Machine (TSGBM), which is a gradient boosting algorithm that solves (4.1.1) to yield the minimax TS estimator. In TSGBM, the first-stage function φ is fixed (as in the standard TS approach), while the second-stage function ψ is learned using gradient boosting on the training set $\{(\boldsymbol{\eta}^{(i)}, \boldsymbol{\theta}^{(i)})\}_{i=1}^M$, where $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{y}^{(i)})$. We borrow the same assumptions from Chapter 3 except that the observations are not necessarily i.i.d. Hence, we shall first review gradient boosting and then propose our method. While minimax TS was proposed in Chapter 3 for i.i.d. data generating mechanisms, this approach can also be applied when the data is non-i.i.d.

4.1.1 Gradient boosting

Gradient boosting is a machine learning paradigm for functional estimation that can be used to learn the second-stage function ψ in the TS approach. In this context, we have a training set $\{(\boldsymbol{\eta}^{(i)}, \boldsymbol{\theta}^{(i)})\}_{i=1}^M$, where $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{y}^{(i)})$ is the auxiliary statistic and $\boldsymbol{\theta}^{(i)}$ is the corresponding parameter value. The goal is to estimate a function ψ that maps $\boldsymbol{\eta}^{(i)} \in \mathbb{R}^n$ to $\boldsymbol{\theta}^{(i)} \in \Theta$. For simplicity, we assume in this section that $\Theta \subseteq \mathbb{R}$.

The estimated function, denoted by $\hat{\psi}$, is obtained by solving the optimization problem

$$\hat{\psi} = \arg \min_{H \in \mathcal{H}} \frac{1}{M} \sum_{i=1}^M \ell(\boldsymbol{\theta}^{(i)}, H(\boldsymbol{\eta}^{(i)})), \quad (4.1.3)$$

where ℓ is a loss function and $\mathcal{H}$ contains functions H of the additive form

$$H(\boldsymbol{\eta}) = \sum_{c=1}^C \omega_c h(\boldsymbol{\eta}; \alpha_c), \quad (4.1.4)$$

where $h(\cdot; \alpha_c)$ are *weak learners* (or base hypotheses) that are simple functions parameterized by α_c , e.g., $h(\boldsymbol{\eta}; \alpha_c) = \alpha_c^T \boldsymbol{\eta}$. Substituting (4.1.4) in (4.1.3) yields an optimization problem in $\{\omega_c\}_{c=1}^C$ and $\{\alpha_c\}_{c=1}^C$, which is usually solved in a forward “stage-wise” fashion, where we start with an initial guess H_0 and then for $c = 1, \dots, C$,

$$\omega_c, \alpha_c = \arg \min_{\omega, \alpha} \sum_{i=1}^M \ell(\boldsymbol{\theta}^{(i)}, H_{c-1}(\boldsymbol{\eta}^{(i)}) + \omega h(\boldsymbol{\eta}^{(i)}; \alpha)), \quad (4.1.5)$$

with $H_c(\boldsymbol{\eta}) = H_{c-1}(\boldsymbol{\eta}) + \alpha_c h(\boldsymbol{\eta}; \alpha_c)$.

A GBM is a tree-based gradient boosting algorithm that uses a regression tree as the weak learner $h(\boldsymbol{\eta}; \alpha)$. In particular, at each boosting iteration c in the stage-wise optimization (4.1.5), a regression tree of depth $D \in \mathbb{N}$ partitions the input space into $Q \in \mathbb{N}$ disjoint leaf regions, denoted by the set $\{R_{jc}\}_{j=1}^Q$, where the auxiliary statistic $\boldsymbol{\eta}$ falls into exactly one of these leaves. Thus, α_c is replaced by the set $\{R_{jc}\}_{j=1}^Q$. It has been shown in [60] that (4.1.5) can be equivalently solved by

$$\nu_{jc} = \arg \min_{\nu} \sum_{\boldsymbol{\eta}^{(i)} \in R_{jc}} \ell(\boldsymbol{\theta}^{(i)}, H_{c-1}(\boldsymbol{\eta}^{(i)}) + \nu), \quad (4.1.6)$$

$$H_c(\boldsymbol{\eta}) = H_{c-1}(\boldsymbol{\eta}) + \rho \nu_{jc} \mathbb{1}\{\boldsymbol{\eta} \in R_{jc}\},$$

where $\mathbb{1}\{A\}$ denotes the indicator function of the set A , and $0 < \rho \leq 1$ is the learning rate (step size).

There are several open source libraries such as XGBoost [46], CatBoost [61], and LightGBM [62] that implement GBM. Each library has its own hyperparameters that characterize the regression trees. An illustration of GBM workflow is given in Figure 4.1.1.

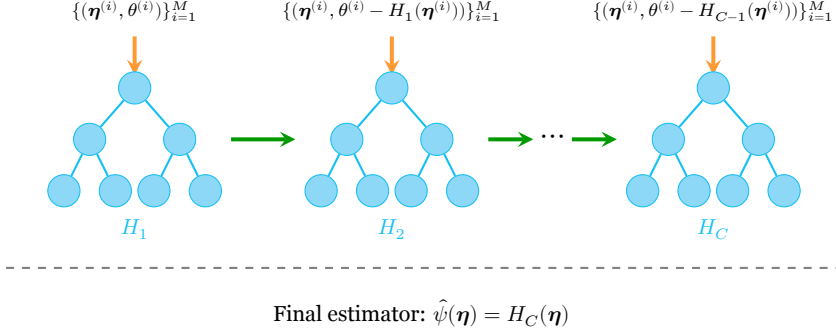


Figure 4.1.1: Illustration of GBM. At each iteration p , a regression tree is trained on the residuals $\theta^{(i)} - H_{p-1}(\eta^{(i)})$ from the previous iteration, yielding the cumulative model H_p . The final estimator $\hat{\psi} = H_C$ is the result after P boosting iterations.

4.2 Minimax TSGBM

In this section, we propose the TSGBM algorithm for parameter estimation. Towards that end, we use LightGBM in the second stage of TS, which is a gradient boosting library developed by Microsoft [62]. In particular, we are interested in using it to minimize $\max_{\theta \in \Theta} R(\theta, T)$, where $T = \psi^* \circ \varphi$, with ψ^* that will be delivered by LightGBM, and φ is a fixed compression function. Recall from (4.1.1) that the approximate minimax risk is given by

$$\min_{T \in \mathcal{T}} \max_{i=1, \dots, M} \ell(\theta^{(i)}, T(\mathbf{y}^{(i)})). \quad (4.2.7)$$

Using LightGBM in the context of the minimax TS approach to parameter estimation requires minimizing the “custom loss” $\max_{i=1, \dots, M} \ell(\theta^{(i)}, T(\mathbf{y}^{(i)}))$. Such a loss does not have a simple expression, hence it cannot be directly used for LightGBM. Therefore, we use a soft-max approximation [63] for $\max_{i=1, \dots, M} \ell(\theta^{(i)}, T(\mathbf{y}^{(i)}))$ which is given by

$$\max_{i=1, \dots, M} \ell(\theta^{(i)}, T(\mathbf{y}^{(i)})) \approx \frac{\log \left(\sum_{i=1}^M \exp(K \ell(\theta^{(i)}, T(\mathbf{y}^{(i)}))) \right)}{K}, \quad (4.2.8)$$

where K is chosen large enough (in the order of $10^3 - 10^4$).

Algorithm 4.1 describes the implementation of TSGBM. The inputs to this algorithm are the proposal distribution s and a collection of tuning parameters of LightGBM, which includes: learning rate, number of iterations, maximum depth of a regression tree, number of leaves, fraction of data used, minimum number of data points in a leaf, and amount of ℓ_1 regularization. The output

Algorithm 4.1 TSGBM Estimator

```

1: Input:  $s, K$ , tuning parameters
2: Generate  $\theta^{(i)} \sim s(\cdot)$ ,  $i = 1, \dots, M$ 
3: for  $i = 1, \dots, M$  do
4:    $\mathbf{y}^{(i)} \leftarrow \mathcal{M}(\theta^{(i)})$  ▷ Model  $\mathcal{M}(\theta)$  as data simulator
5:    $\boldsymbol{\eta}^{(i)} \leftarrow \varphi(\mathbf{y}^{(i)})$  ▷ Data compression step
6: end for
7: Construct nonlinear features  $J(\boldsymbol{\eta}^{(i)})$ , where  $J: \mathbb{R}^n \rightarrow \mathbb{R}^m$ , with  $n < m \ll N$ 
8: Define the soft-max loss as in (4.2.8)
9: Train LightGBM with  $\{(J(\boldsymbol{\eta}^{(i)}), \theta^{(i)})\}_{i=1}^M$  as the training set and (4.2.8) as
   loss function
10: Output:  $\psi^*$  ▷ Function learned by LightGBM
11: Final estimator:  $T := \psi^* \circ \varphi$ .

```

of Algorithm 4.1 is $T = \psi^* \circ \varphi$, which is the desired TSGBM estimator. Here, φ is the data compression function and ψ^* is the function approximation provided by LightGBM. To evaluate the performance of this estimator, its MSE is computed via Algorithm 4.2. For simplicity, Algorithm 4.2 is presented for a scalar parameter $\theta \in \Theta \subseteq \mathbb{R}$. The inputs are the estimator T , a test parameter value θ^* at which the MSE is evaluated, and the number of Monte Carlo runs. If the parameter vector $\boldsymbol{\theta}$ is d -dimensional (with $d > 1$), Algorithm 4.2 is applied separately to each coordinate of $\boldsymbol{\theta}$, yielding a corresponding MSE for each component.

Algorithm 4.2 MSE Evaluation for TSGBM

```

1: Input: Estimator  $T$ , test parameter  $\theta^*$ , number of Monte Carlo runs  $N_{\text{mc}}$ 
2:  $S \leftarrow 0$ 
3: for  $m = 1, \dots, N_{\text{mc}}$  do
4:    $\mathbf{y} \leftarrow \mathcal{M}(\theta^*)$  ▷ Simulate data from model
5:    $\boldsymbol{\eta} \leftarrow \varphi(\mathbf{y})$  ▷ Data compression
6:   Construct nonlinear features  $J(\boldsymbol{\eta})$ 
7:    $\hat{\theta} \leftarrow T(\mathbf{y})$  ▷ Compute estimate
8:    $S \leftarrow S + (\hat{\theta} - \theta^*)^2$ 
9: end for
10:  $\text{MSE} \leftarrow S/N_{\text{mc}}$ 
11: Output: MSE

```

Remark 4.1. To account for nonlinearities in the true system, we can use a nonlinear transformation (monomials of degree up to a given order) of the compressed data, which will then be a part of the training set to GBM. However, the dimension of the transformed data should be much smaller than N .

4.3 Simulation Study

In this section we demonstrate the performance of TSGBM via numerical simulations that include both i.i.d. and non-i.i.d. data generating mechanisms. A detailed description of these mechanisms is provided in Sections 4.3.1, 4.3.2 and 4.3.3.

For the numerical simulations, the following setup is considered: The TSGBM estimator (T) is implemented using Algorithm 4.1, with $K = 10^3$. For each $\theta^{(i)}$ in Algorithm 4.1, we generate observations $\mathbf{y}^{(i)}$ of dimension $N \times 1$. Unless otherwise stated, we do not construct nonlinear features $J(\boldsymbol{\eta}^{(i)})$ in Step 7 of Algorithm 4.1, in which case the training set in Step 9 of Algorithm 4.1 becomes $\{(\boldsymbol{\eta}^{(i)}, \theta^{(i)})\}_{i=1}^M$. For TSGBM, the function approximator ψ is defined via a regression tree. To generate scatter plots for the estimated vs. true values of the parameter, we draw fresh samples of θ from the proposal distribution s , and collect them in a test set. We compute the MSE according to Algorithm 4.2, where the values of θ^* used are listed in Tables 4.3.1-4.3.4 as “True Values”.

4.3.1 Weibull distribution

We first demonstrate the performance of TSGBM on an i.i.d. data generating mechanism. Specifically, we consider a Weibull distribution whose probability density function is parameterized by two parameters, namely, *scale* (λ) and *shape* (κ). It has been shown in Chapter 3 that the minimax TS approach gives reliable estimates for λ . However, estimates for the shape parameter κ when its true value is small are not reliable, in the sense that its MSE is significantly larger when compared to the asymptotic Cramér-Rao lower bound (CRLB).

True Values		CRLB		MSE, Minimax		MSE, TSGBM	
η	$\tilde{\alpha}$	η	$\tilde{\alpha}$	$\hat{\eta}$	$\hat{\tilde{\alpha}}$	$\hat{\eta}$	$\hat{\tilde{\alpha}}$
2	2	1.11×10^{-4}	2.43×10^{-4}	2.17×10^{-4}	16×10^{-2}	0.62×10^{-4}	2.17×10^{-4}
2	8	6.93×10^{-6}	3.89×10^{-3}	4.28×10^{-4}	13.29×10^{-2}	0.36×10^{-5}	0.44×10^{-2}
4	2	4.43×10^{-4}	2.43×10^{-4}	8.38×10^{-4}	14.67×10^{-2}	3.76×10^{-4}	2.11×10^{-4}
4	8	2.77×10^{-5}	3.89×10^{-3}	1.72×10^{-3}	8.59×10^{-2}	3.86×10^{-5}	0.52×10^{-2}
8	2	1.77×10^{-3}	2.43×10^{-4}	3.307×10^{-3}	18.605×10^{-2}	2.37×10^{-3}	2.16×10^{-4}

Table 4.3.1: MSE of the “actual” minimax TS (see Section 3.4.2) and TSGBM estimators of the scale and shape parameters, and their corresponding CRLBs.

However, using TSGBM, *i.e.*, with LightGBM in the second stage, shows significant improvement in the MSE for κ , and also for the scale parameter λ the MSE is comparable to that outlined in Chapter 3. This has been highlighted in Table 4.3.1. Also, the training time of TSGBM is considerably smaller compared to the minimax TS algorithm developed in Chapter 3, which runs CVXPY [53] for each training sample of θ (and the number of such training samples required for the original minimax TS method to be reliable was fairly large). TSGBM, on

the other hand, does not suffer from this issue, since it directly uses a soft-max maximization objective as the custom loss during its training and minimizes this loss for the entire training sample, *i.e.*, it performs batch training.

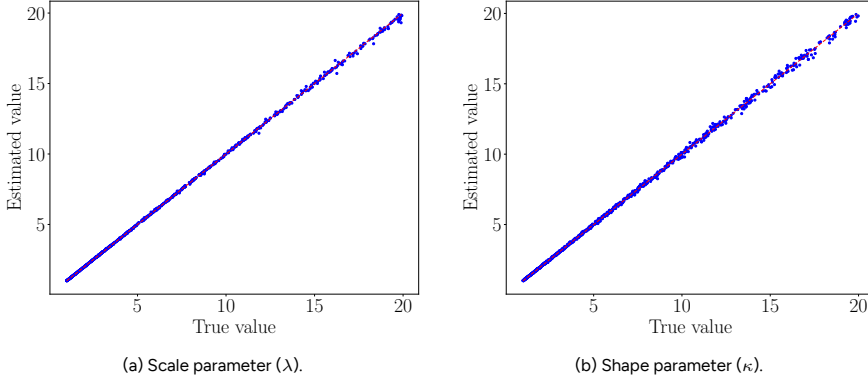


Figure 4.3.2: Scatter plots of estimated vs. true parameter values. (a) Weibull scale parameter λ (Section 4.3.1), (b) Weibull shape parameter κ (Section 4.3.1).

For the numerical simulations, the following choices are made: (i) φ computes quantiles from order statistics of the observations ($n = 5$), for both the original TS and TSGBM-based estimators. As nonlinear features, J was taken as described in Chapter 3 for λ and κ , (ii) The functional approximator ψ was taken to be linear regression for the original TS estimator, while for TSGBM, we use LightGBM with parameters: $l_r = 0.1$, $itr = 10^3$, $B_max = 5$, $l_num = 16$, $bg_fr = 1$, $d_l_min = 20$, and $\rho = 10^{-4}$, (iii) For both TS estimators, the proposal distribution for λ and κ is an uninformative prior over $[1, 20]$. Also, $N_{mc} = 1000$ for MSE computations and (iv) For the original minimax TS estimator, $M = 1000$ samples are considered, while for TSGBM, $M = 10^4$ since it is more computationally efficient and it gives more reliable estimates compared to the original approach.

The scatter plots of estimated vs. true parameter shown in Figures 4.3.2a, 4.3.2b complement the results given in Table 4.3.1, demonstrating that TSGBM yields reliable estimates.

4.3.2 Single parameter dynamical system

We now consider a single parameter discrete-time state-space model from [33] that is described as

$$\begin{aligned} x_1(k+1) &= \bar{a}x_1(k) + v_{11}(k), \\ x_2(k+1) &= x_1(k) + \bar{a}^2x_2(k) + v_{12}(k), \\ y(k) &= \bar{a}x_1(k) + x_2(k) + v_2(k), \end{aligned} \tag{4.3.9}$$

where $x_1(k)$ and $x_2(k)$ are hidden states and $y(k)$ is the output at time k . $v_{11}(k)$ and $v_{12}(k)$ are additive process noises and $v_2(k)$ is the observation noise at time k . $v_{11} \sim \mathcal{N}(0, 1)$, $v_{12} \sim \mathcal{N}(0, 1)$ and $v_2 \sim \mathcal{N}(0, 0.01)$ are mutually independent Gaussian noises.

For the above state-space model (4.3.9), the parameter of interest is $\bar{a}$ and we use TSGBM to estimate it. For this purpose, we set $M = 10^4$, $N = 10^4$, φ as the coefficients of an estimated AR(5) model, J as the collection of all monomials of degree at most 2, s as a uniform distribution over $[-1, 1]$, and $\theta = \bar{a}$. LightGBM is used with parameters: $l_r = 0.05$, $itr = 10^3$, $B_{max} = 4$, $l_{num} = 8$, $bg_{fr} = 0.9$, $d_{l_min} = 30$, and $\rho = 10^{-3}$.

Figure 4.3.3 shows a scatter plot of estimated vs. true values of $\bar{a}$. We see that the estimates are quite reliable, in the sense that the relation between the true and estimated values is almost linear.

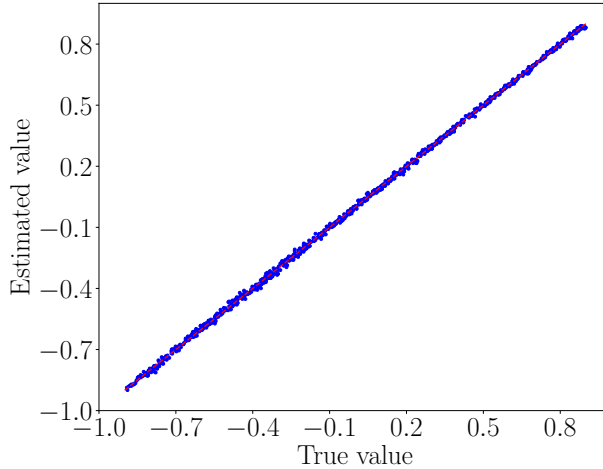


Figure 4.3.3: Scatter plots of estimated vs. true values of $\bar{a}$.

Next, we compute the MSE of TSGBM by evaluating the trained estimator for 5 different values of $\bar{a}$ over 1000 Monte Carlo runs in Table 4.3.2.

True Values ($\bar{a}$)	MSE
0.1	1.08×10^{-4}
0.2	1.01×10^{-4}
0.3	8.12×10^{-5}
0.4	7.24×10^{-5}
0.5	1.02×10^{-4}

Table 4.3.2: MSE of TSGBM for different values of $\bar{a}$; see Section 4.3.2.

Based on the results of this section, we can conclude that the minimax framework of the TS approach is indeed applicable for dynamical systems, where the data generating process is not i.i.d, and TSGBM gives very reliable estimates.

4.3.3 Stochastic volatility model

In this subsection, we consider a discrete-time stochastic volatility model from [64] that is described by the equations

$$\begin{aligned} x(k+1) &= a_v + b_v x(k) + v_k, \\ y(k) &= \sqrt{e^{x(k)}} r_k, \end{aligned} \quad (4.3.10)$$

where $v_k \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1)$, $r_k \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1)$ and v_k, r_k are mutually independent. In this model, a_v, b_v are the unknown parameters that we need to estimate. Let $\theta = (a_v \ b_v)^T$ be the parameter vector of model (4.3.10). As one may notice, model (4.3.10) has multiplicative noise in its observation. Hence, using an estimated AR(n) process as the compression stage of the TS approach is not a good idea, since an AR(n) model assumes that noise is added to the output. In order to obtain an additive noise structure for (4.3.10), we transform the observations by first squaring $y(k)$ and then applying the logarithm. This leads to the transformed observations

$$\bar{y}(k) = x(k) + \bar{r}_k,$$

where $\bar{r}(k) = \log(r_k^2)$.

Now, $\bar{y}(k)$ has an additive noise structure, and hence an AR(n) model can be used in the first stage of TS. Therefore, the final state-space model equivalent to (4.3.10) is

$$\begin{aligned} x(k+1) &= a_v + b_v x(k) + v_k \\ \bar{y}(k) &= x(k) + \bar{r}_k. \end{aligned} \quad (4.3.11)$$

We will use this model, instead of (4.3.10), to estimate $\theta = (a_v \ b_v)^T$ using the TS approach. For estimating a_v and b_v , we consider ranges of a_v and b_v to be (i) $a_v \in [-0.5, 0.5]$ and $b_v \in [0.45, 0.9]$ and (ii) $a_v \in [0.1, 0.5]$ and $b_v \in [0.1, 0.9]$.

Figures 4.3.4-4.3.7 show scatter plots of estimated vs. true values of the unknown parameters a_v and b_v of the stochastic volatility model. Tables 4.3.4, 4.3.3 list the MSE of TSGBM for both a_v and b_v .

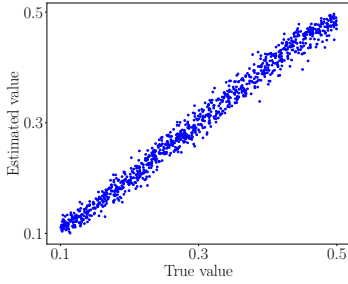


Figure 4.3.4: Scatter plot of estimated vs. true values for $a_v \in [0.1, 0.5]$; see Section 4.3.3.

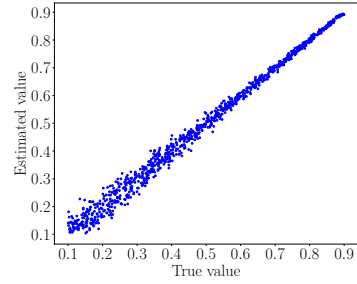


Figure 4.3.5: Scatter plot of estimated vs. true values for $b_v \in [0.1, 0.9]$; see Section 4.3.3.

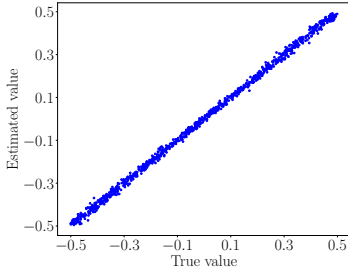


Figure 4.3.6: Scatter plot of estimated vs. true values for $a_v \in [-0.5, 0.5]$; see Section 4.3.3.

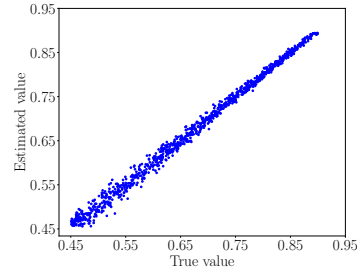


Figure 4.3.7: Scatter plot of estimated vs. true values for $b_v \in [0.45, 0.9]$; see Section 4.3.3.

True Values		MSE	
a_v	b_v	$\hat{a}_v$	$\hat{b}_v$
0.1	0.5	1.74×10^{-4}	4.05×10^{-4}
0.2	0.6	1.22×10^{-4}	1.6×10^{-4}
0.3	0.7	2.12×10^{-4}	8.68×10^{-5}
0.4	0.8	2.21×10^{-4}	4.78×10^{-5}
0.5	0.9	1.27×10^{-3}	1.42×10^{-5}

Table 4.3.3: MSE of TSGBM for different values of $a_v \in [0.1, 0.5]$ and $b_v \in [0.1, 0.9]$; see Section 4.3.3.

True Values		MSE	
a_v	b_v	$\hat{a}_v$	$\hat{b}_v$
-0.5	0.45	6.13×10^{-4}	3.83×10^{-4}
-0.3	0.55	2.59×10^{-4}	2.89×10^{-4}
-0.1	0.65	1.19×10^{-4}	1.13×10^{-4}
0.1	0.75	7.07×10^{-5}	4.75×10^{-5}
0.3	0.85	1.89×10^{-4}	3.57×10^{-5}

Table 4.3.4: MSE of TSGBM for different values of $a_v \in [-0.5, 0.5]$ and $b_v \in [0.45, 0.9]$; see Section 4.3.3.

For these figures and tables, we have set $M = 10^4$, $N = 3 \times 10^4$, s as a uniform distribution over the ranges of a_v, b_v considered (see below), and φ as the coefficients of an estimated AR(5) model. Also, we have considered a nonlinear feature map J consisting of all monomials of degree at most 2, and the following parameters for LightGBM: $l_r = 0.05$, $itr = 10^4$, $B_{max} = 6$, $l_{num} = 8$, $bg_{fr} = 0.9$, $d_{l_{min}} = 30$, and $\rho = 10^{-4}$. From Figures 4.3.4-4.3.7 and Tables 4.3.4-4.3.3, we see that TSGBM provides fairly reliable estimates of a_v and b_v with minimal training effort, as it might not be the case with DNNs or recurrent neural networks.

4.4 Conclusion

In this chapter, we have developed a computationally efficient minimax implementation of the TS approach, and we have numerically demonstrated that it is applicable even for dynamical systems with non-i.i.d. observations. In particular, the new algorithm, called TSGBM, uses gradient boosting as the supervised learning algorithm in the second stage of TS, and it provides reliable estimates of unknown parameters with minimal training effort, in contrast to using, say, deep neural networks.

5 Asymptotic Analysis of TS

Building on the statistical frameworks developed in Chapters 3 and 4, this chapter establishes the asymptotic properties of the TS estimator. Specifically, we prove that, under suitable conditions, the TS estimator is strongly consistent and asymptotically normal.

5.1 Setup for Asymptotic Analysis

We consider the setting introduced in Chapter 3: a high-fidelity simulator $\mathcal{M}(\theta)$ parameterized by $\theta \in \Theta \subseteq \mathbb{R}^d$, $d \geq 1$, generates observations $\mathbf{y}^{(\theta)} = (y_1, \dots, y_N)^\top$. The true system is $\mathcal{M}(\theta_0)$ with unknown $\theta_0 \in \Theta$, and real-world observations are denoted $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_N^{(0)})^\top$. In the sequel, observations for $\theta^{(i)} \in \Theta$ and $\theta \in \Theta$ are denoted $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_N^{(i)})^\top$ and $\mathbf{y}^{(\theta)} = (y_1^{(\theta)}, \dots, y_N^{(\theta)})^\top$ respectively, and for the true system, $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_N^{(0)})^\top$.

Recall that the TS estimator operates in two phases:

1. **Training Phase:** Sample parameters $\{\theta^{(i)}\}_{i=1}^M, \theta^{(i)} \stackrel{\text{i.i.d.}}{\sim} \pi$. For each $\theta^{(i)}$, $\mathcal{M}(\theta^{(i)})$ generates observations $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_N^{(i)})^\top$, forming $\mathcal{D}_{\text{tr}} = \{(\boldsymbol{\eta}^{(i)}, \theta^{(i)})\}_{i=1}^M$, where $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{y}^{(i)})$ is the auxiliary statistic. The TS estimator learns a mapping

$$T_{\text{TS}}(\cdot) = \arg \min_{T(\cdot) \in \mathcal{T}} \frac{1}{M} \sum_{i=1}^M \ell(T(\mathbf{y}^{(i)}), \theta^{(i)}), \quad (5.1.1)$$

where $\mathcal{T} = \{T: \mathbb{R}^N \rightarrow \mathbb{R}^d\}$. It has a composite structure:

$$T_{\text{TS}} = \psi^* \circ \varphi, \quad (5.1.2)$$

with $\varphi: \mathbb{R}^N \rightarrow \mathbb{R}^n$, $n \ll N$, a compression function (e.g., sample quantiles), and $\psi^*: \mathbb{R}^n \rightarrow \mathbb{R}^d$ solving

$$\psi^* = \arg \min_{\psi \in \mathcal{G}} \frac{1}{M} \sum_{i=1}^M \ell(\psi(\boldsymbol{\eta}^{(i)}), \boldsymbol{\theta}^{(i)}), \quad (5.1.3)$$

where $\mathcal{G}$ is a class of supervised learning models (e.g., kernel regressors).

2. **Implementation Phase:** Once the mapping T_{TS} has been trained, it is applied directly to new, real-world observations $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_N^{(0)})^\top$ to produce an estimate of the true parameter:

$$\hat{\boldsymbol{\theta}}_0 = T_{\text{TS}}(\mathbf{y}^{(0)}) = \psi^*(\boldsymbol{\eta}^{(0)}), \quad (5.1.4)$$

where $\boldsymbol{\eta}^{(0)} = \varphi(\mathbf{y}^{(0)})$. No optimization is performed at this stage; the mapping learned offline in the training phase is used to map the auxiliary statistic to a parameter estimate.

The training phase of TS typically involves large values of M and N , and in our asymptotic analysis we consider regimes in which N (and subsequently M) grows. In the limit, we will establish that, under suitable conditions, the estimator T_{TS} converges to

$$\bar{T}_{\text{TS}}(\cdot) = \bar{\psi} \circ \varphi(\cdot), \quad (5.1.5)$$

where

$$\bar{\psi} = \arg \min_{\psi \in \mathcal{G}} \lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M \ell(\psi(\boldsymbol{\eta}^{(i)}), \boldsymbol{\theta}^{(i)}). \quad (5.1.6)$$

The limit $N \rightarrow \infty$ being taken before $M \rightarrow \infty$ reflects the assumption that the observation length grows faster than the number of parameter samples. Consistency and normality will be proved for $\bar{T}_{\text{TS}}(\mathbf{y}^{(0)})$ as $N \rightarrow \infty$.

Remark 5.1. The need for three limits ($M \rightarrow \infty$ and $N \rightarrow \infty$ in training, and $N \rightarrow \infty$ in implementation) to define the asymptotic properties of TS distinguishes it from standard point estimation, requiring a tailored analysis.

5.1.1 Definitions of asymptotic properties

Definition 5.1 (Consistency). The TS estimator is said to be *consistent* if, when observations $\mathbf{y}^{(0)}$ are generated from the true system $\mathcal{M}(\boldsymbol{\theta}_0)$,

$$\bar{T}_{\text{TS}}(\mathbf{y}^{(0)}) \xrightarrow[N \rightarrow \infty]{\text{a.s.}} \boldsymbol{\theta}_0, \quad \text{for all } \boldsymbol{\theta}_0 \in \Theta. \quad (5.1.7)$$

Definition 5.2 (Asymptotic Normality). The TS estimator is said to be *asymptotically normal* if there exists a positive definite matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ such that, when observations $\mathbf{y}^{(0)}$ are generated from $\mathcal{M}(\boldsymbol{\theta}_0)$,

$$\sqrt{N} [\bar{T}_{\text{TS}}(\mathbf{y}^{(0)}) - \boldsymbol{\theta}_0] \xrightarrow[N \rightarrow \infty]{d} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad \text{for all } \boldsymbol{\theta}_0 \in \Theta. \quad (5.1.8)$$

5.2 Asymptotic Analysis of TS Estimator

In this section, we state the choice of the first and second stages of the TS estimator, followed by the main assumptions for asymptotic analysis.

First stage of the TS estimator. The first stage is a data compression function $\varphi: \mathbb{R}^N \rightarrow \mathbb{R}^n$, where $n \ll N$ is fixed. We choose φ as the function that computes sample quantiles.

Definition 5.3 (Sample quantiles). Let $y_{(1)}^{(\theta)} \leq \dots \leq y_{(N)}^{(\theta)}$ denote the order statistics of a sample $y_1^{(\theta)}, \dots, y_N^{(\theta)}$. Fix $n \ll N$ and define levels $\gamma_k = \frac{k}{n+1}$ for $k = 1, \dots, n$. The empirical γ_k -quantile is $y_{(\lfloor \gamma_k N \rfloor)}^{(\theta)}$, and the first-stage compression function collects these quantiles into the auxiliary statistic

$$\boldsymbol{\eta}^{(\theta)} := \varphi(\mathbf{y}^{(\theta)}) = (y_{(\lfloor \gamma_1 N \rfloor)}^{(\theta)}, \dots, y_{(\lfloor \gamma_n N \rfloor)}^{(\theta)})^\top \in \mathbb{R}^n. \quad (5.2.9)$$

As shown later in Lemma 5.1, as $N \rightarrow \infty$, the sample quantiles converge to the population quantiles.

Definition 5.4 (Population quantiles). For a parameter value $\theta \in \Theta$ and a level $\gamma \in [0, 1]$, the γ -quantile of the distribution F_θ is defined by

$$q^{(\theta)}(\gamma) := \inf\{x \in \mathbb{R} : F_\theta(x) \geq \gamma\}.$$

For the vector of quantile levels $(\gamma_1, \dots, \gamma_n)^\top$, define the population quantiles as

$$\bar{\boldsymbol{\eta}}^{(\theta)} := (q^{(\theta)}(\gamma_1), \dots, q^{(\theta)}(\gamma_n))^\top \in \mathbb{R}^n. \quad (5.2.10)$$

The vector $\bar{\boldsymbol{\eta}}^{(\theta)}$ collects the population quantiles corresponding to the order statistics extracted in the first stage.

Second stage of the TS estimator. The second stage is a supervised learning model with functional representation $\psi: \mathbb{R}^n \rightarrow \mathbb{R}^d$. We choose ψ such that

$$\psi(\boldsymbol{\eta}) = \boldsymbol{\beta}^\top J(\boldsymbol{\eta}), \quad (5.2.11)$$

where $J: \mathbb{R}^n \rightarrow \mathbb{R}^p$ is a continuously differentiable feature map (e.g., monomials of fixed degree) and $\boldsymbol{\beta}$ is a $p \times d$ coefficient matrix. In the scalar-parameter case $d = 1$, $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\psi(\boldsymbol{\eta}) = \boldsymbol{\beta}^\top J(\boldsymbol{\eta})$ yields a scalar output.

Thus, ψ receives the auxiliary statistic from the first stage, as given by (5.2.9), and outputs an estimate of θ as

$$\begin{aligned} T_{\text{TS}}(\mathbf{y}^{(\theta)}) &= \psi(\boldsymbol{\eta}^{(\theta)}) \\ &= \boldsymbol{\beta}^\top J(\boldsymbol{\eta}^{(\theta)}). \end{aligned} \quad (5.2.12)$$

Therefore, combining (5.2.9) and (5.2.12), the TS estimator is a function of the form

$$T_{\text{TS}} = \boldsymbol{\beta}^\top (J \circ \varphi). \quad (5.2.13)$$

5.2.1 Main assumptions

Assumption 5.1. The model structure $\mathcal{M}(\theta)$ and true system are autonomous, i.e., do not have input signals.

Assumption 5.2. For each $\theta \in \Theta$, $\mathcal{M}(\theta)$ generates observations $\mathbf{y}^{(\theta)} = (y_1^{(\theta)}, \dots, y_N^{(\theta)})^\top$ on $(\Omega, \mathcal{F}, \mathbb{P})$, where, given θ , $y_j^{(\theta)} \stackrel{\text{i.i.d.}}{\sim} f_\theta$ with conditional distribution $\mathbb{P}_\theta(\mathbf{y}^{(\theta)}) = \prod_{j=1}^N f_\theta(y_j^{(\theta)})$, whose pdf f_θ and cdf F_θ are continuous.

Assumption 5.3. The true system is $\mathcal{M}(\theta_0)$, with $\theta_0 \in \Theta$.

Assumption 5.4. There exists a $\beta^* \in \mathbb{R}^{p \times d}$ such that $\beta^{*\top} J(\bar{\eta}^{(\theta)}) = \theta$ for each $\theta \in \Theta$, where $J: \mathbb{R}^n \rightarrow \mathbb{R}^p$ is a continuously differentiable function, and $\bar{\eta}^{(\theta)} \in \mathbb{R}^n$ (cf. Eq. (5.2.10)).

Assumption 5.5. $\mathbb{E}_\theta[J(\bar{\eta}^{(\theta)})J^\top(\bar{\eta}^{(\theta)})]$ is invertible.

Assumption 5.6. $F_\theta(\cdot)$ has connected support, and $F_\theta^{-1}(\cdot)$ is differentiable for each $\theta \in \Theta$.

Remark 5.2 (On the assumptions).

1. Autonomous models arise naturally when inputs are constant or unobserved, as is typical in reliability or vibration problems. Section 5.2.4 shows that, appealing to standard prediction–error arguments [5], the theory extends to systems with external inputs.
2. The i.i.d. hypothesis is the classical baseline for large-sample analysis. The same consistency and normality conclusions typically hold for weakly dependent, stationary data by replacing independence with common mixing [65] or m -dependence [10]. The continuity of f_θ and F_θ ensures that the quantiles are smooth functions of θ .
3. Assuming that $\mathcal{M}(\theta_0)$ belongs to the model class defines a clear target parameter.
4. Assumption 5.4 is the TS analogue of global identifiability, as considered in PEM: it requires that θ be a linear functional of the feature vector. The condition can be relaxed if J is replaced by a universal approximator with controlled error.
5. Assumption 5.5 prevents degeneracy in the training regression by ensuring the feature covariance matrix is nonsingular, mirroring the requirement of local identifiability in classical estimation.
6. Assumption 5.6, on the connected support and a differentiable quantile map, is a standard regularity condition that justifies the Delta-method step; virtually all continuous distributions used in practice satisfy it.

5.2.2 Consistency of the TS estimator

We establish consistency by showing that the first-stage compression converges a.s., which propagates through the continuous second stage.

Note: For ease of exposition, proofs are presented for the scalar parameter case ($d = 1$). In this setting, β and its finite-sample estimate $\hat{\beta}$ are vectors in $\mathbb{R}^p$. The results extend to the multivariate case by treating β as a $p \times d$ matrix.

5.2.2.1 Almost sure convergence of empirical quantiles

Lemma 5.1. *Under Assumptions 5.1–5.2, for each $\gamma \in [0, 1]$ and $\theta \in \Theta$, $y_{(\lfloor \gamma N \rfloor)}^{(\theta)} \xrightarrow{\text{a.s.}} q^{(\theta)}(\gamma)$ as $N \rightarrow \infty$.*

Proof. For each $\varepsilon > 0$, let $A_{k,\varepsilon} := \{|y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma)| \leq \varepsilon \text{ for every } N \geq k\}$ be an event in $\mathcal{A}$. To show that $y_{(\lfloor \gamma N \rfloor)}^{(\theta)} \xrightarrow{\text{a.s.}} q^{(\theta)}(\gamma)$ as $N \rightarrow \infty$, we need to establish that for every $\varepsilon > 0$, $\mathbb{P}(A_{k,\varepsilon}) \rightarrow 1$ as $k \rightarrow \infty$.

Notice that

$$\mathbb{P}(A_{k,\varepsilon}^c) = \mathbb{P}\left[|y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma)| > \varepsilon \text{ for some } N \geq k\right].$$

Then,

$$\begin{aligned} \mathbb{P}(A_{k,\varepsilon}^c) &= \mathbb{P}(\cup_{N=k}^{\infty} \{|y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma)| > \varepsilon\}) \\ &\stackrel{(a)}{\leq} \sum_{N=k}^{\infty} \mathbb{P}(|y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma)| > \varepsilon) \\ &\stackrel{(b)}{=} \sum_{N=k}^{\infty} \mathbb{P}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma) > \varepsilon) + \mathbb{P}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma) < -\varepsilon), \end{aligned} \tag{5.2.14}$$

where (a) follows from the union bound and (b) follows from the fact that the events are disjoint.

Now, let us evaluate $\mathbb{P}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma) > \varepsilon)$. Observe that

$$\begin{aligned} y_{(l)}^{(\theta)} \leq x \text{ if and only if } \sum_{j=1}^N \mathbb{1}\{y_j^{(\theta)} \leq x\} \geq l, \\ y_{(l)}^{(\theta)} \geq x \text{ if and only if } \sum_{j=1}^N \mathbb{1}\{y_j^{(\theta)} \geq x\} \geq N - l + 1. \end{aligned} \tag{5.2.15}$$

Then $\mathbb{P}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma) > \varepsilon)$ can be bounded as

$$\begin{aligned}
 & \mathbb{P}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma) > \varepsilon) \\
 & \stackrel{(c)}{=} \mathbb{P}\left(\frac{1}{N} \sum_{j=1}^N \mathbb{1}\{y_j^{(\theta)} \geq q^{(\theta)}(\gamma) + \varepsilon\} \geq 1 - \frac{\lfloor \gamma N \rfloor}{N} + \frac{1}{N}\right) \\
 & \leq \mathbb{P}\left(\frac{1}{N} \sum_{j=1}^N \mathbb{1}\{y_j^{(\theta)} \geq q^{(\theta)}(\gamma) + \varepsilon\} \geq 1 - \frac{\lfloor \gamma N \rfloor}{N}\right) \\
 & = \mathbb{P}\left(\frac{1}{N} \sum_{j=1}^N (\mathbb{1}\{y_j^{(\theta)} \geq q^{(\theta)}(\gamma) + \varepsilon\} - \mathbb{E}[\mathbb{1}\{y_j^{(\theta)} \geq q^{(\theta)}(\gamma) + \varepsilon\}]) \right. \\
 & \quad \left. \geq 1 - \frac{\lfloor \gamma N \rfloor}{N} - \mathbb{E}[\mathbb{1}\{y_j^{(\theta)} \geq q^{(\theta)}(\gamma) + \varepsilon\}]\right) \\
 & \stackrel{(d)}{\leq} \exp\left(-2N\left(F_\theta(q^{(\theta)}(\gamma) + \varepsilon) - \frac{\lfloor \gamma N \rfloor}{N}\right)^2\right) \\
 & \stackrel{(e)}{\leq} \exp[-2N(F_\theta(q^{(\theta)}(\gamma) + \varepsilon) - \gamma)^2],
 \end{aligned}$$

where (c) follows from (5.2.15), (d) follows from Hoeffding's inequality [42], and (e) follows from the inequality $\lfloor \gamma N \rfloor < \gamma N$. Note that $F_\theta(q^{(\theta)}(\gamma) + \varepsilon) - \gamma > 0$ due to Assumption 5.6, according to which $F_\theta(\cdot)$ has connected support.

Similarly, $\mathbb{P}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma) < -\varepsilon) \leq \exp[-2N(F_\theta(q^{(\theta)}(\gamma) + \varepsilon) - \gamma)^2]$ by applying Hoeffding's inequality in the opposite direction.

Combining this with (5.2.14), we see that $\mathbb{P}(A_{k,\varepsilon}^c) \rightarrow 0$ as $k \rightarrow \infty$, which means $\mathbb{P}(A_{k,\varepsilon}) \rightarrow 1$. $\square$

Based on this lemma, $y_{(\lfloor \gamma N \rfloor)}^{(\theta)} \xrightarrow{\text{a.s.}} q^{(\theta)}(\gamma)$ as $N \rightarrow \infty$ for each fixed $\gamma \in [0, 1]$. It immediately follows that the auxiliary statistic of the first stage of TS converges a.s. to the population quantiles:

$$\boldsymbol{\eta}^{(\theta)} \xrightarrow{\text{a.s.}} \bar{\boldsymbol{\eta}}^{(\theta)} \quad \text{as } N \rightarrow \infty. \quad (5.2.16)$$

5.2.2.2 Strong consistency of the TS estimator

We study the TS estimator on a fresh test sample $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_N^{(0)})^\top$, $y_j^{(0)} \stackrel{\text{i.i.d.}}{\sim} f_{\theta_0}$, generated by the true system $\mathcal{M}(\theta_0)$.

The finite-sample estimate is $\hat{\theta}_0 = \hat{\beta}^\top J(\boldsymbol{\eta}^{(0)})$, with $\hat{\beta}$ obtained by solving

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^p} \frac{1}{M} \sum_{i=1}^M (\theta^{(i)} - \beta^\top J(\boldsymbol{\eta}^{(i)}))^2. \quad (5.2.17)$$

The solution to (5.2.17) is given by

$$\hat{\beta} = \left[\sum_{i=1}^M J(\boldsymbol{\eta}^{(i)}) J^\top(\boldsymbol{\eta}^{(i)}) \right]^{-1} \sum_{i=1}^M J(\boldsymbol{\eta}^{(i)}) \theta^{(i)}. \quad (5.2.18)$$

Assuming the limit exists, which will be established in Theorem 5.2, the limiting estimate $\bar{T}_{\text{TS}}(\mathbf{y}^{(0)})$ (cf. Eqs. (5.1.5) and (5.1.6)) is given by

$$\bar{\theta}_0 = \bar{\beta}^\top J(\boldsymbol{\eta}^{(0)}), \quad \text{where } \bar{\beta} = \lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} \hat{\beta}.$$

Our aim is to prove:

$$(i) \quad \bar{\theta}_0 \xrightarrow[N \rightarrow \infty]{\text{a.s.}} \theta_0. \quad (5.2.19)$$

$$(ii) \quad \sqrt{N}(\bar{\theta}_0 - \theta_0) \xrightarrow[N \rightarrow \infty]{d} \mathcal{N}(0, \Sigma_{\text{TS}}). \quad (5.2.20)$$

To this end, the first step is to show that the feature map J , applied to the first-stage output, preserves almost sure convergence. This is guaranteed by the following standard result on continuous mappings of convergent sequences.

Lemma 5.2. *Let $J: \mathbb{R}^n \rightarrow \mathbb{R}^p$ be a continuous function and $x_1, \dots, x_N$ be i.i.d. random variables defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ with $x_i: \Omega \rightarrow \mathbb{R}^n$. Then, if $x_N \xrightarrow{\text{a.s.}} x$, it holds that $J(x_N) \xrightarrow{\text{a.s.}} J(x)$.*

Proof. See [66, p. 8]. □

As a result of Lemma 5.2,

$$J(\boldsymbol{\eta}^{(\theta)}) \xrightarrow{\text{a.s.}} J(\bar{\boldsymbol{\eta}}^{(\theta)}) \quad \text{for each } \theta \in \Theta. \quad (5.2.21)$$

To establish strong consistency (5.2.19), we first show in Theorem 5.1 together with Eq. (5.2.21) that, as $N \rightarrow \infty$ and $M \rightarrow \infty$, $\hat{\beta} \rightarrow \bar{\beta} \in \mathbb{R}^p$ with probability 1, where

$$\bar{\beta} = \mathbb{E}_\theta [J(\bar{\boldsymbol{\eta}}^{(\theta)}) J^\top(\bar{\boldsymbol{\eta}}^{(\theta)})]^{-1} \mathbb{E}_\theta [\theta J(\bar{\boldsymbol{\eta}}^{(\theta)})]. \quad (5.2.22)$$

Theorem 5.1. *Under Assumptions 5.1–5.2 and Lemma 5.2, $\lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} \hat{\beta} \xrightarrow{\text{a.s.}} \bar{\beta}$.*

Proof. The limit can be established as follows:

$$\begin{aligned}
 & \lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} \hat{\beta} \\
 & \stackrel{(b)}{=} \lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} \left[\sum_{i=1}^M J(\boldsymbol{\eta}^{(i)}) J^\top(\boldsymbol{\eta}^{(i)}) \right]^{-1} \sum_{i=1}^M J(\boldsymbol{\eta}^{(i)}) \theta^{(i)} \\
 & \stackrel{(c)}{=} \lim_{M \rightarrow \infty} \left[\frac{1}{M} \sum_{i=1}^M J(\bar{\boldsymbol{\eta}}^{(i)}) J^\top(\bar{\boldsymbol{\eta}}^{(i)}) \right]^{-1} \frac{1}{M} \sum_{i=1}^M J(\bar{\boldsymbol{\eta}}^{(i)}) \theta^{(i)} \\
 & \stackrel{(d)}{=} \mathbb{E}_\theta [J(\bar{\boldsymbol{\eta}}^{(\theta)}) J^\top(\bar{\boldsymbol{\eta}}^{(\theta)})]^{-1} \mathbb{E}_\theta [\theta J(\bar{\boldsymbol{\eta}}^{(\theta)})] \\
 & = \bar{\beta}.
 \end{aligned}$$

Here, (b) follows from (5.2.18), (c) follows from Lemmas 5.1 and 5.2, and (d) follows from the strong law of large numbers [41]. $\square$

Theorem 5.2 (Strong Consistency). *Under Assumptions 5.1–5.5, $\bar{\theta}_0 = \bar{T}_{\text{TS}}(\mathbf{y}^{(0)}) = \bar{\beta}^\top J(\boldsymbol{\eta}^{(0)}) \xrightarrow{\text{a.s.}} \theta_0$ as $N \rightarrow \infty$.*

Proof. We have that

$$\begin{aligned}
 \lim_{N \rightarrow \infty} \hat{\theta}_0 &= \lim_{N \rightarrow \infty} \bar{\beta}^\top J(\boldsymbol{\eta}^{(0)}) \\
 & \stackrel{(a)}{=} \lim_{N \rightarrow \infty} \mathbb{E}_\theta [\theta J^\top(\bar{\boldsymbol{\eta}}^{(\theta)})] \mathbb{E}_\theta [J(\bar{\boldsymbol{\eta}}^{(\theta)}) J^\top(\bar{\boldsymbol{\eta}}^{(\theta)})]^{-1} J(\boldsymbol{\eta}^{(0)}) \\
 & \stackrel{(b)}{=} \lim_{N \rightarrow \infty} \beta^{*\top} \mathbb{E}_\theta [J(\bar{\boldsymbol{\eta}}^{(\theta)}) J^\top(\bar{\boldsymbol{\eta}}^{(\theta)})] \mathbb{E}_\theta [J(\bar{\boldsymbol{\eta}}^{(\theta)}) J^\top(\bar{\boldsymbol{\eta}}^{(\theta)})]^{-1} J(\boldsymbol{\eta}^{(0)}) \\
 & \stackrel{(c)}{=} \lim_{N \rightarrow \infty} \beta^{*\top} J(\boldsymbol{\eta}^{(0)}) \\
 & = \beta^{*\top} \lim_{N \rightarrow \infty} J(\boldsymbol{\eta}^{(0)}) \\
 & = \beta^{*\top} J(\bar{\boldsymbol{\eta}}^{(0)}) = \theta_0,
 \end{aligned}$$

where (a) follows from (5.2.22), whereas (b) and (c) follow from Assumptions 5.4 and 5.5 respectively.

Therefore, $\lim_{N \rightarrow \infty} \hat{\theta}_0 = \theta_0$ a.s. $\square$

5.2.3 Asymptotic normality of the TS estimator

Having established consistency, we now establish asymptotic normality in the form of Eq. (5.2.20). The argument proceeds in three steps:

1. derive a central-limit theorem (CLT) [41] for the empirical quantiles that form the first stage of TS;

2. apply the Delta method (Lemma 5.3) together with Lemma 5.4 to transfer the CLT to the auxiliary statistic $\eta^{(\theta)}$;
3. propagate this limit through the continuous second stage to obtain the asymptotic distribution of the TS estimate.

For each $x \in \mathbb{R}$, define the empirical cdf

$$F_N^{(\theta)}(x) = \frac{1}{N} \sum_{i=1}^N \mathbb{1}\{y_i^{(\theta)} \leq x\} = \frac{1}{N} \sum_{i=1}^N Z_i^{(\theta)}, \quad (5.2.23)$$

where $Z_i^{(\theta)} := \mathbb{1}\{y_i^{(\theta)} \leq x\}$. Since $y_1^{(\theta)}, \dots, y_N^{(\theta)}$ are i.i.d. under Assumption 5.2, the $Z_i^{(\theta)}$'s are also i.i.d. Each $Z_i^{(\theta)}$ takes the value 1 with probability $\mathbb{P}(y_i^{(\theta)} \leq x) = F_\theta(x)$ and the value 0 otherwise, so $Z_i^{(\theta)} \sim \text{Bernoulli}(F_\theta(x))$ with mean $\mathbb{E}[Z_i^{(\theta)}] = F_\theta(x)$ and variance $\text{Var}(Z_i^{(\theta)}) = F_\theta(x)(1 - F_\theta(x))$. By the CLT,

$$\sqrt{N}(F_N^{(\theta)}(x) - F_\theta(x)) \xrightarrow{d} \mathcal{N}(0, F_\theta(x)(1 - F_\theta(x))). \quad (5.2.24)$$

Lemma 5.3 (Delta Method). *Let $g: \mathbb{R} \rightarrow \mathbb{R}$ be a differentiable function. Let $x_N, N \in \mathbb{N}$, be i.i.d. random variables such that $N^b(x_N - a) \xrightarrow{d} x$ for some constants $b > 0$ and a . Then,*

$$N^b(g(x_N) - g(a)) \xrightarrow{d} g'(a) \cdot x. \quad (5.2.25)$$

Proof. See [66, p. 26, Theorem 3.1]. □

Under Assumption 5.6, we can compute the derivative of F_θ^{-1} : Let $t = F_\theta^{-1}(z)$. Then $z = F_\theta(t)$ implies $dz = f_\theta(t) dt$, hence

$$(F_\theta^{-1})'(z) = \frac{1}{f_\theta(F_\theta^{-1}(z))}. \quad (5.2.26)$$

Using (5.2.26) with $b = 1/2$ in Lemma 5.3 and (5.2.24) yields

$$\sqrt{N}(F_\theta^{-1}(F_N^{(\theta)}(x)) - F_\theta^{-1}(F_\theta(x))) \xrightarrow{d} \mathcal{N}\left(0, \frac{F_\theta(x)(1 - F_\theta(x))}{f_\theta^2(x)}\right). \quad (5.2.27)$$

Lemma 5.4. $y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - F_\theta^{-1}(F_N^{(\theta)}(x)) \xrightarrow{\text{a.s.}} 0$ for $\gamma = F_\theta(x)$.

Proof. The proof is similar to that of Lemma 5.1. □

Lemma 5.4 implies that for each $x \in \mathbb{R}$, there is a $\gamma = F_\theta(x)$ such that $y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - F_\theta^{-1}(F_N^{(\theta)}(x)) \xrightarrow{\text{a.s.}} 0$. Furthermore, $y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - F_\theta^{-1}(F_N^{(\theta)}(F_\theta^{-1}(\gamma))) \xrightarrow{\text{a.s.}} 0$ for each $\gamma \in [0, 1]$.

Applying Eq. (5.2.27) with Slutsky's theorem [41], we obtain

$$\sqrt{N}(y_{(\lfloor \gamma N \rfloor)}^{(\theta)} - q^{(\theta)}(\gamma)) \xrightarrow{d} \mathcal{N}\left(0, \frac{\gamma(1-\gamma)}{(f_\theta(q^{(\theta)}(\gamma)))^2}\right). \quad (5.2.28)$$

Since $y_1^{(\theta)}, \dots, y_N^{(\theta)}$ are i.i.d. with distribution F_θ , Eq. (5.2.28) implies

$$\sqrt{N}(\boldsymbol{\eta}^{(\theta)} - \bar{\boldsymbol{\eta}}^{(\theta)}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad (5.2.29)$$

where

$$\boldsymbol{\Sigma} = \text{diag}\left(\frac{\gamma_1(1-\gamma_1)}{(f_\theta(q^{(\theta)}(\gamma_1)))^2}, \dots, \frac{\gamma_n(1-\gamma_n)}{(f_\theta(q^{(\theta)}(\gamma_n)))^2}\right). \quad (5.2.30)$$

Therefore, the auxiliary statistic $\boldsymbol{\eta}^{(\theta)}$, the output of the first stage of TS, is asymptotically normal. It remains to show that the TS estimator is also asymptotically normal.

Theorem 5.3 (Asymptotic Normality). *Under Assumptions 5.1–5.6,*

$$\sqrt{N}(\hat{\theta}_0 - \theta_0) \xrightarrow{d} \mathcal{N}(0, \Sigma_{\text{TS}}),$$

where $\Sigma_{\text{TS}} = \bar{\boldsymbol{\beta}}^\top \nabla J(\bar{\boldsymbol{\eta}}^{(0)}) \boldsymbol{\Sigma} \nabla J(\bar{\boldsymbol{\eta}}^{(0)})^\top \bar{\boldsymbol{\beta}}$, and $\nabla J(\bar{\boldsymbol{\eta}}^{(0)}) \in \mathbb{R}^{p \times n}$ is the Jacobian matrix of J evaluated at $\bar{\boldsymbol{\eta}}^{(0)}$.

Proof. Recall that $\hat{\theta}_0 = \bar{T}_{\text{TS}}(\mathbf{y}^{(0)}) = \bar{\boldsymbol{\beta}}^\top J(\boldsymbol{\eta}^{(0)})$. From (5.2.29), we know that

$$\sqrt{N}(\boldsymbol{\eta}^{(0)} - \bar{\boldsymbol{\eta}}^{(0)}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \quad (5.2.31)$$

where $\boldsymbol{\Sigma}$ is obtained by substituting θ by θ_0 in (5.2.30).

Using Lemma 5.3 with $b = 1/2$, $N = N$, and $g(\boldsymbol{\eta}^{(0)}) = \bar{\boldsymbol{\beta}}^\top J(\boldsymbol{\eta}^{(0)})$ in (5.2.31), we finally obtain

$$\sqrt{N}(\bar{\boldsymbol{\beta}}^\top J(\boldsymbol{\eta}^{(0)}) - \bar{\boldsymbol{\beta}}^\top J(\bar{\boldsymbol{\eta}}^{(0)})) \xrightarrow{d} \mathcal{N}(0, \Sigma_{\text{TS}}),$$

where $\Sigma_{\text{TS}} = \bar{\boldsymbol{\beta}}^\top \nabla J(\bar{\boldsymbol{\eta}}^{(0)}) \boldsymbol{\Sigma} \nabla J(\bar{\boldsymbol{\eta}}^{(0)})^\top \bar{\boldsymbol{\beta}}$ is the asymptotic covariance of the TS estimator, and $\nabla J(\bar{\boldsymbol{\eta}}^{(0)}) \in \mathbb{R}^{p \times n}$ is the Jacobian matrix of J evaluated at $\bar{\boldsymbol{\eta}}^{(0)}$. $\square$

Remark 5.3. The asymptotic normality has been established in Theorem 5.3 for the scalar case. The analysis extends to the multivariate case by noting that $\bar{\boldsymbol{\beta}} \in \mathbb{R}^{p \times d}$ if $\boldsymbol{\theta} \in \mathbb{R}^d$. In this case, Σ_{TS} is a $d \times d$ matrix.

5.2.4 Discussion on the non-i.i.d. case

When Assumptions 5.1–5.2 are relaxed to allow external inputs and stationary dependent data, a natural first stage is given by the least-squares coefficients of an ARX(n_a, n_b) model fitted to each input–output record.

Let $n = n_a + n_b$ be the total number of ARX coefficients. For each training pair, collect the input–output record in $\mathbf{z}_N^{(i)} = (u_1^{(i)}, y_1^{(i)}, \dots, u_N^{(i)}, y_N^{(i)})^\top$ and define

$$\begin{aligned} \boldsymbol{\eta}^{(i)} &= \varphi(\mathbf{z}_N^{(i)}) = \arg \min_{\boldsymbol{\xi}^{(i)} \in \mathbb{R}^n} \sum_{t=n_a+1}^N (y_t^{(i)} - \boldsymbol{\phi}_t^\top \boldsymbol{\xi}^{(i)})^2, \\ \boldsymbol{\phi}_t &= (y_{t-1}^{(i)}, \dots, y_{t-n_a}^{(i)}, u_{t-1}^{(i)}, \dots, u_{t-n_b}^{(i)})^\top. \end{aligned} \quad (5.2.32)$$

Standard prediction-error theory [5], together with a persistently exciting input of order n for every $\boldsymbol{\theta}^{(i)} \in \Theta$, implies

$$\boldsymbol{\eta}^{(i)} \xrightarrow[\text{a.s.}]{N \rightarrow \infty} \bar{\boldsymbol{\eta}}^{(i)},$$

where $\bar{\boldsymbol{\eta}}^{(i)}$ denotes the limiting ARX coefficients.

Combining this limit with Assumption 5.4 and the continuity of the second-stage map ψ , the proof of strong consistency in Section 5.2.2 carries over verbatim, with Lemma 5.1 replaced by the ARX limit above.

Analogously, prediction-error results give the central limit theorem

$$\sqrt{N}(\boldsymbol{\eta}^{(0)} - \bar{\boldsymbol{\eta}}^{(0)}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sigma^2(\mathbb{E}[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^\top])^{-1}),$$

which, via the Delta method, yields the asymptotic normality of the TS estimator under non-i.i.d. assumptions.

Remark 5.4. Alternatively, we can replace independence with weak-dependence assumptions (mixing/ m -dependence) so that the asymptotic theory for order statistics and our quantile-based first stage still apply [65, 67, 68].

5.3 Numerical Validation of Asymptotic Results

We corroborate the asymptotic results with a numerical illustration. Consider the problem of estimating the signal-to-noise ratio (SNR) of a model structure $\mathcal{M}(\theta)$, where $\theta = \mu^2/\sigma^2$ is the SNR, with known mean $\mu = 5$ and true parameter $\theta_0 = 5$ (so $\sigma_0^2 = \mu^2/\theta_0 = 5$). Observations $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_N^{(0)})^\top$ are generated according to

$$y_t = \mu + e_t, \quad e_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2), \quad (5.3.33)$$

with a uniform prior $\pi(\theta) \sim \mathcal{U}[2, 10]$.

The TS estimator uses sample quantiles with $n = 5$ as the compression function φ , selecting order statistics at indices $s = \lfloor j \cdot \frac{N}{n+1} \rfloor$, $j = 1, \dots, n$. Training involves $M \in \{100, 1000, 10000\}$ parameter samples and observation lengths $N \in \{100, 1000, 10000\}$. The mapping ψ is learned via polynomial regression of degree = 2. In the implementation phase, the trained TS estimator is evaluated on test sequences whose length matches the training observation length N . The mean squared error (MSE), $\frac{1}{R} \sum_{r=1}^R (\hat{\theta}^{(r)} - \theta_0)^2$, is computed over $R = 100$ Monte Carlo runs.

The Fisher information for θ is $I(\theta) = N\theta^2/(4\mu^4)$, giving $\text{CRB} = 4\mu^4/(N\theta^2)$. For $\theta_0 = 5$ and $\mu = 5$, $\text{CRB}(\theta) = 500/N$.

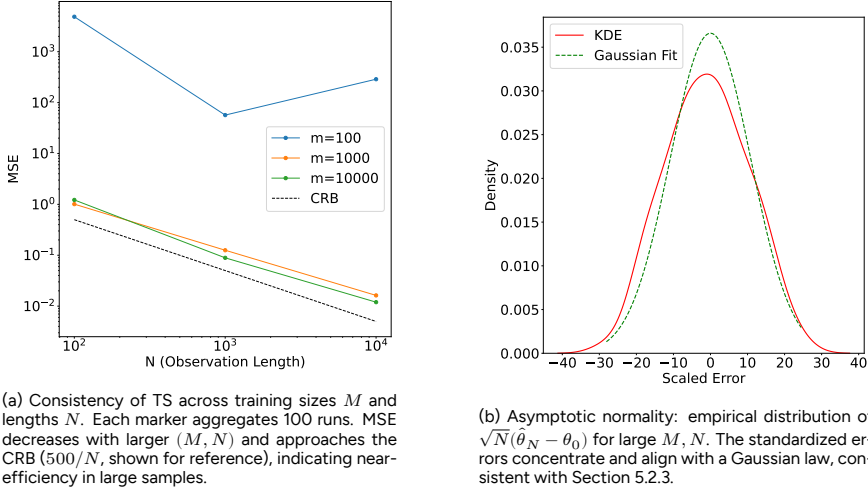


Figure 5.3.1: Numerical validation of the asymptotic properties of the TS estimator: consistency (left) and asymptotic normality (right).

The consistency plot 5.3.1a validates our result on the consistency of TS under i.i.d. assumptions. As M and N increase to 10^4 , the MSE decreases, approaching the CRB, which indicates near-efficiency and consistency as $N \rightarrow \infty$, $M \rightarrow \infty$. The normality plot 5.3.1b confirms that for large M and N , the distribution of $\sqrt{N}(\hat{\theta}_N - \theta_0)$ is approximately Gaussian, corroborating our theoretical results from Section 5.2.3.

5.4 Conclusion

In this chapter, we have established that the TS estimator is *consistent* and *asymptotically normal* in the i.i.d. setting with quantile-based compression and a linear-in-features second stage, and we have derived a closed-form expression for its asymptotic covariance. We have also briefly outlined how our analysis can be extended to stationary, weakly dependent data.

6 Non-Asymptotic Analysis of TS

The asymptotic properties of the TS estimator have been established in Chapter 5. In this chapter, we derive non-asymptotic results for the TS estimator, specifically where the first stage (data compression) utilizes sample quantiles and the second stage employs an instrumental variable (IV) regressor.

The rest of the chapter is structured as follows: In Section 6.1, we define the problem setup. Section 6.2 provides the structure of the TS estimator, while in Section 6.3, we derive finite sample bounds for the TS estimator, followed by a numerical validation in Section 6.4. Finally, we conclude the chapter in Section 6.5.

6.1 Problem Formulation

Consider a data generating mechanism capable of producing sequences of random real samples of any desired length N and with a prescribed joint distribution $p_N(\cdot; \theta)$, depending on a parameter $\theta \in \Theta \subseteq \mathbb{R}^d$. Given a finite sequence $(y_1^{(\theta)}, \dots, y_N^{(\theta)})$ generated by this mechanism, where $\theta \in \Theta$ is unknown, the goal of parameter estimation is to approximate such parameter vector.

Simulation-driven estimators address this problem by sampling a large number of parameter vectors $\theta^{(i)}$ ($i \in [M]$) according to a prescribed distribution on Θ , and for each $\theta^{(i)}$ generate, using the mechanism, a long sequence $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_N^{(i)})$. Then, using these parameter vectors and the synthetic sequences, an estimator $T: \mathbb{R}^N \rightarrow \Theta$ is fitted using a supervised learning approach. For example, the TS approach determines this estimator as the mapping T_{TS} that

minimizes the empirical loss

$$\frac{1}{M} \sum_{i=1}^M \ell(T(\mathbf{y}^{(i)}), \boldsymbol{\theta}^{(i)})$$

over a restricted class of mappings T , to be defined in the next section, where $\ell: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_0^+$ is a predefined loss function. In this chapter, we focus on a variant of TS estimators based on instrumental variables (cf. Section 6.3).

The goal of this chapter is to derive a finite sample deviation bound for $T_{\text{TS}}(\mathbf{y}^{(\boldsymbol{\theta})})$ from $\boldsymbol{\theta} \in \Theta$. That is, given any $\varepsilon > 0$, we aim to find a $\delta \in (0, 1)$ such that

$$\mathbb{P} [\|T_{\text{TS}}(\mathbf{y}^{(0)}) - \boldsymbol{\theta}_0\|_2 \leq \varepsilon] \geq 1 - \delta.$$

Our finite sample bound for the TS approach holds under general conditions. A special case when the data generating mechanism is i.i.d. and without exogenous inputs will be discussed in the next section, and later it will be shown that the assumptions needed for the finite sample bound hold in such a setup.

It is important to note that a simulation-driven estimator, such as TS, can be constructed for each parameter separately, by restricting the loss function ℓ to the domain $\mathbb{R} \times \mathbb{R}$, while keeping the fictitious samples as being drawn for the full parameter vector. Therefore, in the sequel we restrict ourselves to the estimation of scalar parameters, i.e., $d = 1$.

6.2 TS Estimator

In this section, we revisit the TS approach, and discuss how to specifically choose its stages to estimate parameters of models producing i.i.d. data.

6.2.1 First stage of the TS estimator

The first stage of the TS estimator is a data compression function, denoted by $\varphi: \mathbb{R}^N \rightarrow \mathbb{R}^n$, where $n \ll N$ is fixed. For example, in the case of i.i.d. scalar data, we can choose φ as a function that outputs sample quantiles. The auxiliary statistic is then

$$\boldsymbol{\eta}^{(\boldsymbol{\theta})} := \varphi(\mathbf{y}^{(\boldsymbol{\theta})}) = \left(y_{(\lfloor N \frac{1}{n+1} \rfloor)}^{(\boldsymbol{\theta})}, \dots, y_{(\lfloor N \frac{n}{n+1} \rfloor)}^{(\boldsymbol{\theta})} \right)^\top, \quad (6.2.1)$$

where the sample quantiles are defined next.

Definition 6.1 (Sample quantiles). The (100γ) -th sample quantile $y_{(N\gamma)}^{(\boldsymbol{\theta})}$, where $\gamma \in [0, 1]$ is such that $N\gamma \in [N]$, is the $(N\gamma)$ -th smallest number in the set $\{y_1^{(\boldsymbol{\theta})}, \dots, y_N^{(\boldsymbol{\theta})}\}$.

6.2.2 Second stage of the TS estimator

The second stage of the TS estimator is a supervised learning model with functional representation $\psi: \mathbb{R}^n \rightarrow \mathbb{R}$. In this chapter, we choose ψ to be a kernel regressor, of the form

$$\psi(\boldsymbol{\eta}) = \boldsymbol{\beta}^\top J(\boldsymbol{\eta}).$$

Here, $J: \mathbb{R}^n \rightarrow \mathbb{R}^p$, where $n \leq p \ll N$, is a fixed continuous function, for example, one that computes monomials of a given order, and $\boldsymbol{\beta} \in \mathbb{R}^p$. Thus, ψ receives the auxiliary statistic from the first stage, as given by (6.2.1), and then estimates θ as

$$T_{\text{TS}}(\mathbf{y}^{(\theta)}) = \psi(\boldsymbol{\eta}^{(\theta)}) = \boldsymbol{\beta}^\top J(\boldsymbol{\eta}^{(\theta)}). \quad (6.2.2)$$

Therefore, combining (6.2.1) and (6.2.2), the TS estimator is a functional of the form

$$T_{\text{TS}} = \hat{\boldsymbol{\beta}}^\top (J \circ \varphi),$$

where the choice of $\hat{\boldsymbol{\beta}}$ will be introduced later.

6.3 Finite Sample Bounds

In this section, we derive finite sample bounds for a specific version of the TS approach. For the analysis, let $g: \mathbb{R}^N \rightarrow \mathbb{R}^p$ denote the composition $g := J \circ \varphi$, and let $g_*: \Theta \rightarrow \mathbb{R}^p$ be a given function. For a matrix $\mathbf{A}$, we denote by $\sigma_{\min}(\mathbf{A})$ its minimum singular value and by $\|\mathbf{A}\|_{\max} := \max_{i,j} |A_{ij}|$ its max-norm. We make the following assumptions on g and g_* :

Assumption 6.1. There is an $\alpha > 0$ such that the functions g satisfy $\|g(x_1, \dots, x_N)\|_2 \leq \alpha$ for all $N \in \mathbb{N}$ and $x_1, \dots, x_N \in \mathbb{R}$.

Assumption 6.2. There is a function $\mu: \mathbb{N} \rightarrow \mathbb{R}_0^+$ such that $\lim_{N \rightarrow \infty} \mu(N) = 0$ and, for all $N \in \mathbb{N}$, $\|\mathbb{E}[g(\mathbf{y}^{(\theta)})|\theta] - g_*(\theta)\|_2 \leq \mu(N)$ for all $\theta \in \Theta$.

Assumption 6.3. There is a function $\delta: \mathbb{N} \times \mathbb{R}^+ \rightarrow \mathbb{R}_0^+$ such that $\lim_{N \rightarrow \infty} \delta(N, \varepsilon) = 0$ for every $\varepsilon > 0$, and, for all $N \in \mathbb{N}$ and $\varepsilon > 0$, $\mathbb{P}[\|g(\mathbf{y}^{(\theta)}) - g_*(\theta)\|_2 \geq \varepsilon] \leq \delta(N, \varepsilon)$ with probability at most $\delta(N, \varepsilon)$.

Assumption 6.4. There exists a vector $\boldsymbol{\beta}^* \in \mathbb{R}^p$ and a small number $\zeta > 0$ such that

$$\theta = \boldsymbol{\beta}^{*\top} g_*(\theta) + r(\theta), \quad \text{for all } \theta \in \Theta,$$

where $|r(\theta)| \leq \zeta$ for all $\theta \in \Theta$.

Assumption 6.5. The function $g_* : \Theta \rightarrow \mathbb{R}^p$ satisfies

$$\mathbb{E}_\theta \{g_*(\theta) g_*^\top(\theta)\} \succ 0.$$

Assumption 6.1 requires that the range of g is uniformly bounded. Assumption 6.2 states that g should be an asymptotically unbiased estimator of g_* , which is a very mild requirement, due to the definition of g_* . Assumption 6.3 is, in a sense, a stronger version of Assumption 6.2, as it requires not only asymptotic unbiasedness, but consistency in probability of g . Note, in particular, that as long as the first and second stages satisfy Assumptions 6.2 and 6.3, the results derived in this section are *not* restricted to i.i.d. data.

Assumption 6.4 requires that the structure of the estimator is flexible enough to lead to a sufficiently small bias (described by the function r). Since quantiles fully describe a distribution, if the distributions are locally identifiable, then, by Weierstrass' theorem [69, Theorem 7.26] (on the uniform approximability of continuous functions by polynomials), Assumption 6.4 holds with ζ as small as desired if p is chosen sufficiently large and J contains all monomials up to a sufficiently large degree.

Finally, Assumption 6.5 can be interpreted as an identifiability requirement [70, 71], in the sense that the limit data compression stage and fixed monomials, as codified in g_* , should span $\mathbb{R}^p$ over the support of the prior distribution over $\theta \in \Theta$.

For the choice of sample quantiles as a first stage of the TS estimator, and mild assumptions on the data distribution and the choice of J , it can be shown that several of the previous assumptions hold.

Theorem 6.1. *Assume that the first stage of the TS estimator is given by (6.2.1), and that J is Lipschitz continuous and bounded. Also, assume that the distribution of the samples has connected and bounded support, and their cumulative distribution function, F_θ , has a differentiable inverse for each $\theta \in \Theta$. Then, Assumptions 6.1–6.3 hold.*

Proof. Assumption 6.1 follows directly from the boundedness of J . To establish Assumptions 6.2 and 6.3, we use the finite-sample concentration bound for sample quantiles derived in the proof of Lemma 5.1 in Chapter 5. Specifically, for each $k \in [n]$ and $\varepsilon > 0$,

$$\mathbb{P} \left(\left| y_{(\lfloor \frac{kN}{n+1} \rfloor)}^{(\theta)} - q^{(\theta)} \left(\frac{k}{n+1} \right) \right| > \varepsilon \right) \leq 2 \exp \left[-2N \left(F_\theta \left(q^{(\theta)} \left(\frac{k}{n+1} \right) + \varepsilon \right) - \frac{k}{n+1} \right)^2 \right], \quad (6.3.3)$$

where $q^{(\theta)}(\gamma)$ denotes the γ -quantile of F_θ (cf. Definition 5.4).

Let the population quantiles be denoted by

$$\bar{\eta}^{(\theta)} = \left(q^{(\theta)} \left(\frac{1}{n+1} \right), \dots, q^{(\theta)} \left(\frac{n}{n+1} \right) \right)^\top,$$

so that $g_*(\theta) = J(\bar{\eta}^{(\theta)})$. If J has Lipschitz constant $L > 0$, then by the union bound over $k = 1, \dots, n$ and (6.3.3),

$$\|g(\mathbf{y}^{(\theta)}) - g_*(\theta)\|_2 \leq Ln \max_{k=1, \dots, n} \left| y_{\lfloor \frac{kN}{n+1} \rfloor}^{(\theta)} - q^{(\theta)}(k/[n+1]) \right| \leq Ln\varepsilon$$

with probability at least

$$1 - 2 \sum_{k=1}^n \exp \left[-2N \left(F_\theta \left(q^{(\theta)} \left(\frac{k}{n+1} \right) + \varepsilon \right) - \frac{k}{n+1} \right)^2 \right].$$

Thus, Assumption 6.3 holds with

$$\delta(N, \varepsilon) = 2 \sum_{k=1}^n \exp \left[-2N \left(F_\theta \left(q^{(\theta)} \left(\frac{k}{n+1} \right) + \varepsilon \right) - \frac{k}{n+1} \right)^2 \right].$$

Finally, to establish Assumption 6.2, Jensen's inequality and the identity $\mathbb{E}[X] = \int_0^M \mathbb{P}[X \geq x] dx$ (where X is a non-negative random variable with support in $[0, M]$) yield

$$\|\mathbb{E}[g(\mathbf{y}^\theta) | \theta] - g_*(\theta)\|_2 \leq \mathbb{E} [\|g(\mathbf{y}^\theta) - g_*(\theta)\|_2 | \theta] \leq \int_0^{2LnY} \delta(N, \varepsilon) d\varepsilon,$$

where Y is an upper bound on the $y_i^{(\theta)}$'s. The integral is finite and tends to 0 as $N \rightarrow \infty$ by the monotone convergence theorem [72]. Therefore, Assumption 6.2 holds with $\mu(N) = \int_0^{2LnY} \delta(N, \varepsilon) d\varepsilon$. $\square$

In Chapter 5, the second-stage coefficients $\hat{\beta}$ were obtained via ordinary least squares. Here, we instead employ an instrumental variables (IV) estimator [14]. For each parameter value $\theta^{(i)}$, two independent data sequences $\mathbf{y}^{(i)}$ and $\mathbf{y}'^{(i)}$ are generated. In the least-squares normal equations, the covariance matrix is formed from products $g(\mathbf{y}^{(i)})g^\top(\mathbf{y}^{(i)})$, for which $\mathbb{E}[g(\mathbf{y}^{(i)})g^\top(\mathbf{y}^{(i)}) | \theta^{(i)}] \neq g_*(\theta^{(i)})g_*^\top(\theta^{(i)})$ for finite N . The IV formulation replaces this with $g(\mathbf{y}^{(i)})g^\top(\mathbf{y}'^{(i)})$; since $\mathbf{y}^{(i)}$ and $\mathbf{y}'^{(i)}$ are conditionally independent given $\theta^{(i)}$, we have $\mathbb{E}[g(\mathbf{y}^{(i)})g^\top(\mathbf{y}'^{(i)}) | \theta^{(i)}] = \mathbb{E}[g(\mathbf{y}^{(i)}) | \theta^{(i)}] \mathbb{E}[g^\top(\mathbf{y}'^{(i)}) | \theta^{(i)}]$, which converges to $g_*(\theta^{(i)})g_*^\top(\theta^{(i)})$ by Assumption 6.2. The IV estimator is given by

$$\hat{\beta}_{\text{IV}} = \left[\frac{1}{M} \sum_{i=1}^M g(\mathbf{y}'^{(i)}) g^\top(\mathbf{y}^{(i)}) \right]^{-1} \frac{1}{M} \sum_{i=1}^M \theta^{(i)} g(\mathbf{y}'^{(i)}).$$

Here, for each value $\theta^{(i)}$, two independent data sequences $\mathbf{y}^{(i)}$ and $\mathbf{y}'^{(i)}$ are generated. The resulting estimator can then be decomposed, under Assumption 6.4, as

$$\begin{aligned} \hat{\beta}_{\text{IV}} = \beta^* &+ \left[\frac{1}{M} \sum_{i=1}^M g(\mathbf{y}'^{(i)}) g^\top(\mathbf{y}^{(i)}) \right]^{-1} \cdot \frac{1}{M} \sum_{i=1}^M g(\mathbf{y}'^{(i)}) [g_*(\theta^{(i)}) - g(\mathbf{y}^{(i)})]^\top \beta^* \\ &+ \left[\frac{1}{M} \sum_{i=1}^M g(\mathbf{y}'^{(i)}) g^\top(\mathbf{y}^{(i)}) \right]^{-1} \frac{1}{M} \sum_{i=1}^M r(\theta^{(i)}) g(\mathbf{y}'^{(i)}). \end{aligned} \quad (6.3.4)$$

Let us define

$$\begin{aligned} \mathbf{B}_{N,M} &:= \frac{1}{M} \sum_{i=1}^M g(\mathbf{y}'^{(i)}) g^\top(\mathbf{y}^{(i)}), \\ \mathbf{S}_{N,M} &:= \frac{1}{M} \sum_{i=1}^M g(\mathbf{y}'^{(i)}) [g_*(\theta^{(i)}) - g(\mathbf{y}^{(i)})]^\top, \\ \mathbf{e}_{N,M} &:= \frac{1}{M} \sum_{i=1}^M r(\theta^{(i)}) g(\mathbf{y}'^{(i)}). \end{aligned}$$

Then, (6.3.4) becomes

$$\hat{\beta}_{\text{IV}} - \beta^* = \mathbf{B}_{N,M}^{-1} \mathbf{S}_{N,M} \beta^* + \mathbf{B}_{N,M}^{-1} \mathbf{e}_{N,M}. \quad (6.3.5)$$

Note that the entries of $\mathbf{B}_{N,M}$ are averages of i.i.d. random variables of bounded support, hence by Hoeffding's inequality we have that

$$\lim_{M \rightarrow \infty} \lim_{N \rightarrow \infty} \mathbf{B}_{N,M} = \mathbb{E}_\theta \{g_*(\theta) g_*^\top(\theta)\} =: \mathbf{B}_*.$$

Note that, under Assumption 6.5, $\mathbf{B}_*$ is non-singular.

Furthermore, we have the following finite-sample bound:

Lemma 6.1. *Under Assumptions 6.1, 6.2 and 6.5, given an N large enough so that $2\alpha\mu(N) + \mu^2(N) \leq \sigma_{\min}(\mathbf{B}_*)/[4p]$, the matrix $\mathbf{B}_{N,M}$ satisfies*

$$\sigma_{\min}(\mathbf{B}_{N,M}) \geq \frac{1}{2} \sigma_{\min}(\mathbf{B}_*)$$

with probability at least

$$1 - 4p^2 \exp \left(-\frac{M \sigma_{\min}^2(\mathbf{B}_*)}{128 p^2 \alpha^4} \right).$$

Proof. Let $\mathbf{B}_N := \mathbb{E}[\mathbf{B}_{N,M}]$, $g_i := \mathbb{E}[g(\mathbf{y}^{(i)})|\theta^{(i)}]$, and $\hat{\mathbf{B}}_* := \frac{1}{M} \sum_{i=1}^M g_*(\theta^{(i)})g_*^\top(\theta^{(i)})$. We first note that $\mathbf{B}_{N,M} - \mathbf{B}_* = [\mathbf{B}_{N,M} - \mathbf{B}_N] + [\mathbf{B}_N - \hat{\mathbf{B}}_*] + [\hat{\mathbf{B}}_* - \mathbf{B}_*]$.

Then,

$$\begin{aligned} & |\sigma_{\min}(\mathbf{B}_{N,M}) - \sigma_{\min}(\mathbf{B}_*)| \\ & \leq p\|\mathbf{B}_{N,M} - \mathbf{B}_N\|_{\max} + p\|\mathbf{B}_N - \hat{\mathbf{B}}_*\|_{\max} + p\|\hat{\mathbf{B}}_* - \mathbf{B}_*\|_{\max}, \end{aligned} \quad (6.3.6)$$

where the inequality follows from [73, Corollary 7.3.5 and Problem 5.6.P23] and the triangle inequality.

Therefore, a high probability lower bound on $\sigma_{\min}(\mathbf{B}_{N,M})$ can be obtained by bounding each term appearing in (6.3.6). To this end, we note that

$$\begin{aligned} \mathbf{B}_N - \hat{\mathbf{B}}_* &= \frac{1}{M} \sum_{i=1}^M ([g_i - g_*(\theta^{(i)})]g_*^\top(\theta^{(i)}) + g_*(\theta^{(i)})[g_i - g_*(\theta^{(i)})]^\top \\ & \quad + [g_i - g_*(\theta^{(i)})][g_i - g_*(\theta^{(i)})]^\top). \end{aligned}$$

From Assumptions 6.1 and 6.2, $|(\mathbf{B}_N)_{ij} - (\hat{\mathbf{B}}_*)_{ij}| \leq 2\alpha\mu(N) + \mu^2(N)$ for all $i, j \in [p]$, so $\|\mathbf{B}_N - \hat{\mathbf{B}}_*\|_{\max} \leq 2\alpha\mu(N) + \mu^2(N)$. Choosing $\mu(N)$ large enough so that $2\alpha\mu(N) + \mu^2(N) \leq \sigma_{\min}(\mathbf{B}_*)/(4p)$ yields

$$\|\mathbf{B}_N - \hat{\mathbf{B}}_*\|_{\max} \leq \frac{\sigma_{\min}(\mathbf{B}_*)}{4p}. \quad (6.3.7)$$

Further, since the entries of g and g_* lie in $[-\alpha, \alpha]$, the entries of the zero-mean matrices $\mathbf{B}_{N,M} - \mathbf{B}_N$ and $\hat{\mathbf{B}}_* - \mathbf{B}_*$ are bounded by $2\alpha^2$. Applying Hoeffding's inequality and the union bound over the $2p^2$ total entries, the probability that either $\|\mathbf{B}_{N,M} - \mathbf{B}_N\|_{\max}$ or $\|\hat{\mathbf{B}}_* - \mathbf{B}_*\|_{\max}$ exceeds ε is at most $4p^2 \exp(-M\varepsilon^2/[2\alpha^4])$. Choosing $\varepsilon = \sigma_{\min}(\mathbf{B}_*)/(8p)$ and combining this with (6.3.7) and (6.3.6) yields the result

$$\mathbb{P} \left[\sigma_{\min}(\mathbf{B}_{N,M}) \geq \frac{1}{2} \sigma_{\min}(\mathbf{B}_*) \right] \geq 1 - 4p^2 \exp \left[-\frac{M\sigma_{\min}^2(\mathbf{B}_*)}{128p^2\alpha^4} \right]. \quad (6.3.8)$$

□

A similar bound can be established for $\mathbf{S}_{N,M}$:

Lemma 6.2. *Under Assumptions 6.1 and 6.2, given any $\varepsilon > 0$, and N large enough so that $p\alpha\mu(N) \leq \varepsilon$, the matrix $\mathbf{S}_{N,M}$ satisfies $\|\mathbf{S}_{N,M}\|_2 \leq \varepsilon$ with probability at least*

$$1 - 2p^2 \exp \left(-\frac{M\varepsilon^2}{32\alpha^4} \right).$$

Proof. Note that, if we write $g_i := \mathbb{E}[g(\mathbf{y}^{(i)})|\theta^{(i)}]$, then $\mathbb{E}[\mathbf{S}_{N,M}] = M^{-1} \sum_{i=1}^M g_i [g_*(\theta^{(i)}) - g_i]^\top$, so by Assumptions 6.1 and 6.2, $|(\mathbb{E}[\mathbf{S}_{N,M}])_{ij}| \leq \alpha \mu(N)$ for all $i, j \in [p]$. In other words, $\|\mathbb{E}[\mathbf{S}_{N,M}]\|_{\max} \leq \alpha \mu(N)$.

The entries of g lie in $[-\alpha, \alpha]$, so the entries of $g(\mathbf{y}'^{(i)}) [g_*(\theta^{(i)}) - g(\mathbf{y}^{(i)})]^\top$ lie in $[-2\alpha^2, 2\alpha^2]$. Hence, Hoeffding's inequality implies that $|(\mathbf{S}_{N,M})_{ij} - (\mathbb{E}[\mathbf{S}_{N,M}])_{ij}| \geq \varepsilon$ with probability at most $2 \exp(-M\varepsilon^2/[8\alpha^4])$, for each $i, j \in [p]$. Therefore, by the union bound over i, j , we have that $\|\mathbf{S}_{N,M} - \mathbb{E}[\mathbf{S}_{N,M}]\|_{\max} \geq \varepsilon$ with probability at most $2p^2 \exp(-M\varepsilon^2/[8\alpha^4])$.

Combining these bounds yields

$$\begin{aligned} \|\mathbf{S}_{N,M}\|_2 &\leq \|\mathbf{S}_{N,M} - \mathbb{E}[\mathbf{S}_{N,M}]\|_2 + \|\mathbb{E}[\mathbf{S}_{N,M}]\|_2 \\ &\leq p\|\mathbf{S}_{N,M} - \mathbb{E}[\mathbf{S}_{N,M}]\|_{\max} + p\|\mathbb{E}[\mathbf{S}_{N,M}]\|_{\max} \\ &\leq p\|\mathbf{S}_{N,M} - \mathbb{E}[\mathbf{S}_{N,M}]\|_{\max} + p\alpha\mu(N). \end{aligned}$$

This implies that the result of the lemma holds if N is large enough so that $p\alpha\mu(N) \leq \varepsilon/2$. $\square$

Finally, we can obtain a simple, deterministic upper bound on $\|e_{N,M}\|_2$: from Assumptions 6.1 and 6.4, we have that

$$\|e_{N,M}\|_2 \leq \frac{1}{M} \sum_{i=1}^M |r(\theta^{(i)})| \|g(\mathbf{y}'^{(i)})\|_2 \leq \alpha\zeta. \quad (6.3.9)$$

Combining Lemmas 6.1 and 6.2 with (6.3.5) and (6.3.9), and the bound

$$\|\hat{\beta}_{\text{IV}} - \beta^*\|_2 \leq \|\mathbf{B}_{N,M}^{-1}\|_2 \|\mathbf{S}_{N,M}\|_2 \|\beta^*\|_2 + \|\mathbf{B}_{N,M}^{-1}\|_2 \|e_{N,M}\|_2,$$

immediately produces the following result:

Theorem 6.2. *Under Assumptions 6.1, 6.2, 6.4 and 6.5, for any $\varepsilon > 0$,*

$$\|\hat{\beta}_{\text{IV}} - \beta^*\|_2 \leq \frac{\varepsilon \|\beta^*\|_2 + \alpha\zeta}{2\sigma_{\min}(\mathbf{B}_*)}$$

holds with probability at least

$$1 - 4p^2 \exp\left(-\frac{M\sigma_{\min}^2(\mathbf{B}_*)}{128p^2\alpha^4}\right) - 2p^2 \exp\left(-\frac{M\varepsilon^2}{32\alpha^4}\right),$$

as long as N is large enough so that $2\alpha\mu(N) + \mu^2(N) \leq \sigma_{\min}(\mathbf{B}_)/[4p]$ and $p\alpha\mu(N) \leq \varepsilon$.*

In case Θ is a bounded set, the following upper bound on the norm of β^* can be derived:

Lemma 6.3. Assume that $\Theta \subseteq [-R, R]$, and that Assumptions 6.1, 6.4 and 6.5 hold. Then,

$$\|\beta^*\|_2 \leq \frac{\alpha(R + \zeta)}{\sigma_{\min}(\mathbf{B}_*)}.$$

Proof. By Assumption 6.4, we have that $\mathbb{E}[\theta g_*(\theta)] = \mathbb{E}[g_*(\theta)g_*^\top(\theta)]\beta^* + \mathbb{E}[r(\theta)g_*(\theta)]$, or $\beta^* = \mathbf{B}_*^{-1} \{\mathbb{E}[\theta g_*(\theta)] - \mathbb{E}[r(\theta)g_*(\theta)]\}$. Therefore, by Assumptions 6.1, 6.4 and 6.5,

$$\|\beta^*\|_2 \leq \|\mathbf{B}_*^{-1}\|_2 (R\alpha + \zeta\alpha) = \frac{\alpha(R + \zeta)}{\sigma_{\min}(\mathbf{B}_*)}.$$

□

Once the estimator $\hat{\beta}_{\text{IV}}$ is built, it is applied on a new set of observations $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_N^{(0)})^\top$ which are drawn from $\mathcal{M}(\theta_0)$, and outputs an estimate of θ_0 as $\hat{\theta} = \hat{\beta}_{\text{IV}}^\top g(\mathbf{y}^{(0)})$. Thus, we would like to obtain a probabilistic bound on $\hat{\theta} - \theta_0$. Such a bound is provided by the following theorem.

Theorem 6.3. Under Assumptions 6.1–6.5, for any $\varepsilon > 0$,

$$|\hat{\theta} - \theta_0| \leq \zeta + \varepsilon \|\beta^*\|_2 + \alpha \frac{\varepsilon \|\beta^*\|_2 + \alpha \zeta}{2 \sigma_{\min}(\mathbf{B}_*)}$$

holds except on a set of probability at most

$$4p^2 \exp \left[-\frac{M \sigma_{\min}^2(\mathbf{B}_*)}{128 p^2 \alpha^4} \right] + 2p^2 \exp \left[-\frac{M \varepsilon^2}{32 \alpha^4} \right] + \delta(N, \varepsilon),$$

as long as N is large enough so that $2\alpha\mu(N) + \mu^2(N) \leq \sigma_{\min}(\mathbf{B}_*)/[4p]$ and $p\alpha\mu(N) \leq \varepsilon$.

Proof. Note that, under Assumption 6.4,

$$\begin{aligned} \hat{\theta} &= g(\mathbf{y}^{(0)})^\top \hat{\beta}_{\text{IV}} \\ &= g_*^\top(\theta_0) \beta^* + [g^\top(\mathbf{y}^{(0)}) - g_*^\top(\theta_0)] \beta^* + g^\top(\mathbf{y}^{(0)}) [\hat{\beta}_{\text{IV}} - \beta^*] \\ &= \theta_0 - r(\theta_0) + [g^\top(\mathbf{y}^{(0)}) - g_*^\top(\theta_0)] \beta^* + g^\top(\mathbf{y}^{(0)}) [\hat{\beta}_{\text{IV}} - \beta^*]. \end{aligned}$$

Then, by the Cauchy–Schwarz inequality,

$$\begin{aligned} |\hat{\theta} - \theta_0| &\leq |r(\theta_0)| + \|g(\mathbf{y}^{(0)}) - g_*(\theta_0)\|_2 \|\beta^*\|_2 + \|g(\mathbf{y}^{(0)})\|_2 \|\hat{\beta}_{\text{IV}} - \beta^*\|_2 \\ &\leq \zeta + \|g(\mathbf{y}^{(0)}) - g_*(\theta_0)\|_2 \|\beta^*\|_2 + \alpha \|\hat{\beta}_{\text{IV}} - \beta^*\|_2. \end{aligned}$$

Assumptions 6.3–6.4 and Theorem 6.2 imply the result.

□

Remark 6.1. The following observations are in order:

1. The bound naturally separates the effects of the first and second stages: indeed, the performance of the first stage only appears explicitly in the last term, $\delta(N, \varepsilon)$, of the probability bound, corresponding to the two factors ε appearing in the bound. Thanks to this separation, Theorems 6.2 and 6.3 directly apply to setups with non-i.i.d. data and exogenous inputs, as long as the required assumptions hold for the first stage of TS.
2. α is just a normalization constant. By normalizing g so that $\alpha = 1$ (which implicitly affects β^* and B_* too), the bound of Theorem 6.3 simplifies to

$$|\hat{\theta} - \theta_0| \leq (\zeta + \varepsilon \|\beta^*\|_2) \left[1 + \frac{1}{2\sigma_{\min}(B_*)} \right].$$

3. The term ζ represents the bias of the TS estimator. Indeed, as $M \rightarrow \infty$ and $N \rightarrow \infty$, one can gradually decrease ε so that the bound approaches $|\hat{\theta} - \theta_0| \leq \zeta$ with probability tending to 1.
4. The factor $\sigma_{\min}(B_*)$ measures the sensitivity of the data distribution to the parameter values: as $\sigma_{\min}(B_*)$ decreases, the data distribution is less dependent on θ , which implies that the estimation accuracy decreases, as exhibited in the derived bound, because $\sigma_{\min}(B_*)$ appears in the denominator of the bound.
5. The probabilistic bound is exponential in M : this is due to the use of Hoeffding's inequality, and it is typical of concentration bounds relying on boundedness or sub-Gaussian assumptions on the data [4, Appendix B]. In our case, it follows from Assumption 6.1.
6. The dependence on p (the number of features, and hence the dimension of β) appears in the probability of the bound, which deteriorates as p increases. On the other hand, as p increases, one would expect that the bias bound ζ should decrease, as more features become available to learning an estimator from the data to the parameter value. This conflict corresponds to the standard *bias-variance tradeoff* of statistical inference.

6.4 Numerical Validation

We corroborate Theorem 6.3 numerically using the simulation-generating mechanism $y_k = \sqrt{\theta_0} s_k + w_k$, where $s_k, w_k \sim \mathcal{N}(0, 1)$ and $\theta_0 = 2$. The TS estimator is constructed from a synthetic dataset $\{\mathbf{y}^{(i)}, \mathbf{y}'^{(i)}, \theta^{(i)}\}_{i=1}^M$, with $\theta^{(i)} \sim \mathcal{U}[0.1, 5]$ and $N = 2000$. Two independent observation vectors per $\theta^{(i)}$

ensure the independence required for concentration bounds. The feature mapping g uses 5 sample quantiles with a degree-2 polynomial expansion ($p = 21$). To isolate stochastic error from finite-sample bias, we set the violation threshold τ to the 90-th percentile of empirical error at $M = 100$, then estimate $\mathbb{P}[|\hat{\theta} - \theta_0| > \tau]$ via Monte Carlo as M increases.

Figure 6.4.1 shows that the log of error probability decays linearly, confirming the exponential concentration predicted by Theorem 6.3.

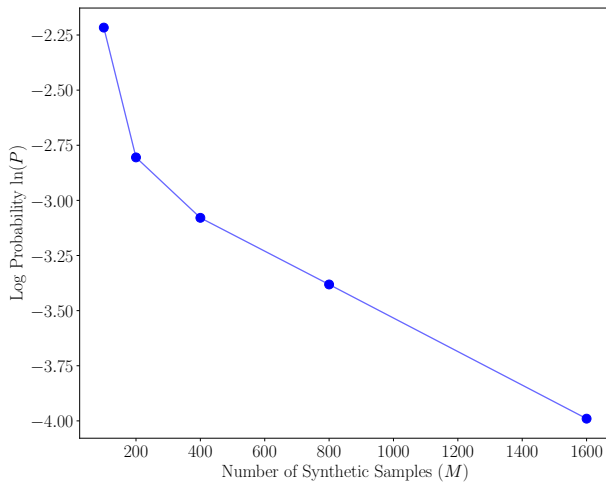


Figure 6.4.1: Probability of violation as a function of M .

6.5 Conclusion

In this chapter, we have derived finite-sample bounds for the Two-Stage estimator. The derived bounds hold under fairly general conditions, and we have discussed how they can be applied to i.i.d. data by making specific choices for the first and second stages, in particular, by compressing the data using sample quantiles in the first stage and then employing kernel regression in the second stage. The bounds clearly exhibit the effects of hyperparameters chosen by the user in the construction of this estimator, and we have shown that they present the standard bias-variance tradeoff prevalent in statistical estimation.

7 Sampling from Jeffreys Prior

In the preceding chapters, we established the statistical foundations of the Two-Stage (TS) estimator and analyzed both its asymptotic (Chapter 5) and finite-sample (Chapter 6) properties. A key insight from these analyses is that the performance of the TS estimator depends critically on the quality of the synthetic training data used to construct it. Recall that the TS estimator is constructed from a training dataset $\{(\boldsymbol{\eta}^{(i)}, \boldsymbol{\theta}^{(i)})\}_{i=1}^M$, where each $\boldsymbol{\theta}^{(i)} \in \Theta$ is a sampled parameter value, $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_N^{(i)})^\top$ is a corresponding observation sequence generated from the model $\mathcal{M}(\boldsymbol{\theta}^{(i)})$, and $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{y}^{(i)})$ is the auxiliary statistic.

A natural question arises: *how should the training parameters $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^M$ be sampled?* While uniform sampling over Θ is a common baseline, it does not account for the varying informativeness of different parameter regions. This chapter addresses the question of diverse and informative training sample generation by proposing a sampling scheme based on *Jeffreys prior*. Jeffreys prior is proportional to the square root of the determinant of the Fisher Information Matrix (FIM), which encodes the intrinsic geometric structure of the parameter space. By sampling parameters according to this distribution, we concentrate training samples in regions where the model is most sensitive to parameter variations, thereby improving the TS estimator's performance.

This chapter proposes a Metropolis-Adjusted Langevin Algorithm (MALA)-based sampling scheme for Jeffreys prior and extends it to cases where the FIM must be approximated numerically via particle filtering. Numerical examples demonstrate that sampling according to Jeffreys prior improves the performance of the TS estimator compared to uniform sampling.

7.1 Problem Statement

Consider a family of probability distributions $\{f(\cdot; \theta) : \theta \in \Theta\}$ defined on a sample space $\mathcal{Y}$, where $\Theta \subseteq \mathbb{R}^d$ is the parameter space. The FIM at a given parameter value θ , denoted by $\mathcal{J}_\theta$, is defined as

$$\mathcal{J}_\theta = \mathbb{E}_{y \sim f(\cdot; \theta)} \left[\nabla_\theta \ln f(y; \theta) \nabla_\theta^\top \ln f(y; \theta) \right], \quad (7.1.1)$$

where $y \in \mathcal{Y}$ denotes the observations, ∇_θ denotes the gradient with respect to θ , and $\mathbb{E}_{y \sim f(\cdot; \theta)}[\cdot] = \int_{\mathcal{Y}} [\cdot] f(y; \theta) dy$.

Intuitively, $\mathcal{J}_\theta$ measures the local sensitivity of the log-likelihood to changes in θ and can be viewed as a Riemannian metric on the parameter space Θ [74].

In practice, computing $\mathcal{J}_\theta$ for many complex or high-dimensional models can be analytically intractable or computationally prohibitive, especially when the distributions lack closed-form expressions (see, e.g., [75] for an example in nonlinear dynamical systems). Consequently, one often resorts to approximate or Monte Carlo methods to estimate $\mathcal{J}_\theta$. For instance, given a finite dataset $\{y_i\}_{i=1}^N \subset \mathcal{Y}$, one can compute sample-based estimates $\hat{\mathcal{J}}_\theta$ by replacing the expectation in (7.1.1) with an empirical average [75].

Jeffreys prior [76] is an uninformative prior distribution on the parameter space Θ . Concretely, Jeffreys prior $\pi(\theta)$ is defined (up to a constant factor) by

$$\pi(\theta) \propto \sqrt{\det(\mathcal{J}_\theta)}, \quad (7.1.2)$$

where $\det(\cdot)$ denotes the determinant function.

One key advantage of Jeffreys prior over alternative uninformative priors is its reparameterization invariance. If one changes variables from θ to $\phi = g(\theta)$ via a smooth invertible transformation, the associated Jeffreys prior adjusts automatically, preserving the same degree of “non-informativeness” in the new parameter space. This property makes Jeffreys prior a canonical choice when one wishes to impose as little subjective structure as possible.

Most importantly, Jeffreys prior’s explicit dependence on the FIM $\mathcal{J}_\theta$ has a Riemannian geometric interpretation. Indeed, as mentioned above, a suitable Riemannian metric on Θ is given by the FIM $\mathcal{J}_\theta$. In particular, $\Delta\theta^\top \mathcal{J}_\theta \Delta\theta$ is a good measure of how “different” the probability distributions $p(\cdot; \theta)$ and $p(\cdot; \theta + \Delta\theta)$ are, for $\Delta\theta$ sufficiently small [77]. Since such a metric induces a natural volume element of the form $d\text{vol} = \sqrt{\det(\mathcal{J}_\theta)} d\theta^1 \wedge \cdots \wedge d\theta^d$ [74, pp. 233], it helps us to distribute samples from Θ according to this volume element so that these samples are well distributed in the parameter space.

Sampling from Jeffreys prior is thus highly relevant in system identification, machine learning, and statistics, but poses two main challenges: (i) for complex models, computing or approximating the FIM can introduce uncertainty,

and (ii) the normalizing constant of Jeffreys prior is rarely known, complicating direct sampling. The objective of this chapter is to address these challenges by proposing an LMC approach that generates Markovian samples whose stationary distribution corresponds precisely to Jeffreys prior $\pi(\theta)$.

7.2 Proposed Method

In this section, we first describe two standard sampling methods: Langevin-based Monte Carlo (LMC), and its improved variant, MALA. Then, in Subsection 7.2.3, we introduce a novel MALA-based method for sampling from Jeffreys prior, which accommodates two scenarios: one where the FIM can be computed analytically, and the other where it must be approximated.

7.2.1 Langevin-based Monte Carlo

LMC provides an efficient, gradient-based framework for exploring the Jeffreys prior distribution. In this setting, we generate samples distributed according to

$$\theta \sim \pi(\theta) \propto \exp(-V(\theta)) \quad (7.2.3)$$

by simulating the following Langevin stochastic differential equation (SDE) [78]:

$$d\theta_t = -\nabla_{\theta} V(\theta_t) dt + \sqrt{2} d\mathbf{w}_t, \quad (7.2.4)$$

where $\mathbf{w}_t$ denotes standard Brownian motion in $\mathbb{R}^d$, and $V: \mathbb{R}^d \rightarrow \mathbb{R}$ is a differentiable potential function.

A standard numerical method to simulate (7.2.4) is the Euler–Maruyama method [79]. Discretizing time in steps of size τ , we obtain

$$\begin{aligned} \theta_{k+1} &= \theta_k - \tau \nabla_{\theta} V(\theta_k) + \sqrt{2\tau} (\mathbf{w}_{k+1} - \mathbf{w}_k) \\ &= \theta_k - \tau \nabla_{\theta} V(\theta_k) + \sqrt{2\tau} \xi_k, \end{aligned} \quad (7.2.5)$$

where $\xi_k \sim \mathcal{N}(0_d, \mathbf{I}_d)$. The discretization step size τ can be fixed or adapted during simulation.

To verify that the SDE in (7.2.4) samples from the Jeffreys prior defined in (7.1.2), we use the corresponding Fokker–Planck equation [78]. For simplicity of illustration, we consider the one-dimensional case with parameter θ . The Fokker–Planck equation associated with the Langevin SDE

$$d\theta_t = -\frac{dV(\theta_t)}{d\theta} dt + \sqrt{2} d\mathbf{w}_t$$

describes the evolution of the probability density $\varrho(\theta, t)$ as

$$\frac{\partial \varrho(\theta, t)}{\partial t} = -\frac{\partial}{\partial \theta} \left[\varrho(\theta, t) \frac{dV(\theta)}{d\theta} \right] + \frac{\partial^2 \varrho(\theta, t)}{\partial \theta^2}.$$

In the steady-state regime $\partial \varrho(\theta, t)/\partial t = 0$, the probability density reaches a stationary distribution denoted by $\varrho_\infty(\theta)$. Imposing zero probability flux at the boundaries (or as $|\theta| \rightarrow \infty$), we obtain $\varrho_\infty(\theta) \propto \exp(-V(\theta))$. Hence, the normalized stationary distribution is

$$\varrho_\infty(\theta) = \frac{\exp(-V(\theta))}{Z}, \quad (7.2.6)$$

where $Z \in \mathbb{R}^+$ is a normalizing constant.

If we specifically define the potential function as

$$V(\theta) = -\frac{1}{2} \ln(\mathcal{I}_\theta), \quad (7.2.7)$$

then it follows from (7.2.6) that $\varrho_\infty(\theta) \propto \pi(\theta)$. Thus, the stationary distribution associated with the Langevin SDE (7.2.4) coincides exactly with Jeffreys prior.

The update rule in (7.2.5) to sample from $\pi(\theta)$ leads to the Unadjusted Langevin Algorithm (ULA), which iterates (7.2.5) to generate the samples θ_k for $k = 0, 1, 2, \dots$, whose distribution approximates $\pi(\theta)$ after a sufficient burn-in period. However, ULA is highly sensitive to the choice of step size τ . If τ is excessively large, discretization errors introduce bias into the invariant distribution or even cause divergence; if too small, the chain suffers slow mixing, resulting in computational inefficiency. Moreover, one often requires the sampled parameter θ_k to stay in a constrained parameter space $\Theta_c \subset \Theta$ due to physical limitations or prior knowledge. ULA lacks a mechanism to enforce these constraints. In the following subsection, we describe the MALA framework, which addresses these limitations.

7.2.2 Metropolis-Adjusted Langevin algorithm

MALA [80] improves upon ULA by introducing a Metropolis–Hastings accept/reject step that compensates for the errors induced by the discretization. Note that in (7.2.5), the proposed variable θ_{k+1} follows the Gaussian distribution

$$\theta_{k+1} \sim \mathcal{N}(\theta_k - \tau \nabla_\theta V(\theta_k), 2\tau \mathbf{I}_d). \quad (7.2.8)$$

The proposal density can thus be explicitly written as

$$q(\theta_{k+1} \mid \theta_k) \propto \exp\left(-\frac{\|\theta_{k+1} - \theta_k + \tau \nabla_\theta V(\theta_k)\|_2^2}{4\tau}\right). \quad (7.2.9)$$

MALA accepts a given proposed sample θ' with probability

$$\varrho^{\text{MALA}}(\theta', \theta_k) = \min\left\{1, \frac{\exp(-V(\theta')) q(\theta_k \mid \theta')}{\exp(-V(\theta_k)) q(\theta' \mid \theta_k)}\right\}.$$

If the proposal is accepted, the chain advances as $\theta_{k+1} = \theta'$; otherwise, it remains at the current position, $\theta_{k+1} = \theta_k$. This ensures that the chain remains exactly invariant with respect to $\pi(\theta)$ under mild conditions, yielding more robust sampling compared to ULA [80]. Furthermore, MALA inherently accommodates constrained parameter spaces $\theta \in \Theta_c$ by modifying the acceptance probability as follows:

$$\varrho_c^{\text{MALA}}(\theta', \theta_k) = \begin{cases} \varrho^{\text{MALA}}(\theta', \theta_k), & \text{if } \theta' \in \Theta_c, \\ 0, & \text{otherwise.} \end{cases}$$

7.2.3 Sampling with estimated FIM

If an analytical form of the FIM is available, we can directly define the multivariate potential function

$$V(\theta) = -\frac{1}{2} \ln \det(\mathcal{J}_\theta), \quad (7.2.10)$$

which is a generalization of (7.2.7), with gradient

$$\nabla_\theta V(\theta) = -\frac{1}{2} \text{tr} \left[\mathcal{J}_\theta^{-1} \frac{\partial \mathcal{J}_\theta}{\partial \theta} \right], \quad (7.2.11)$$

to sample from Jeffreys prior via MALA.

For many problems, such as nonlinear state-space (NLSS) models, closed-form expressions for FIMs are not available. In such cases, one must resort to numerical approximations. In this chapter, we adopt a Particle Filter (PF)-based approach to estimate the FIM, following [75], whereby Forward Filtering–Backward Smoothing (FFBSm) provides an unbiased Monte Carlo approximation of the score function, and consequently, the FIM. Despite the inherent stochasticity of particle filters, the resulting estimates are consistent and converge to the true FIM under standard regularity conditions as the number of particles increases.

Remark 7.1. For a concrete illustration of PF-based FIM estimation within our framework, we refer the reader to the NLSS model simulation in Section 7.3.

Given the estimated FIM $\hat{\mathcal{J}}_\theta$, the gradient $\partial \mathcal{J}_\theta / \partial \theta$ can be approximated by the one-point unbiased estimate. Specifically, we introduce a random perturbation $\nu \sim \mathcal{N}(0_d, \mathbf{I}_d)$ and approximate the derivative as

$$\frac{\partial \mathcal{J}_\theta}{\partial \theta_j} \approx \frac{\nu_j}{h} \left(\hat{\mathcal{J}}_{\theta+h\nu} - \hat{\mathcal{J}}_\theta \right), \quad j = 1, \dots, d, \quad (7.2.12)$$

where $h > 0$ is a small step size, and $\hat{\mathcal{J}}_{\theta+h\nu}$ is computed using the same estimation procedure as for $\hat{\mathcal{J}}_\theta$. This estimator is unbiased (for $h \rightarrow 0$) and substantially faster compared to coordinate-wise finite differences.

Remark 7.2. Although the use of the approximate gradient in (7.2.12) might introduce errors, the Metropolis–Hastings step compensates for these inaccuracies and ensures that the target distribution $\pi(\theta)$ is still preserved [81]. Standard results in MCMC theory [82, 80] guarantee that, as long as the gradient estimator is unbiased and its noise is properly accounted for by the acceptance step, the resulting Markov chain will have $\pi(\theta)$ as its stationary distribution.

Algorithm 7.1 integrates the PF-based FIM estimation with the one-point gradient approximation within MALA. In this way, even when the FIM is only approximately available, our approach guarantees convergence to Jeffreys prior.

Algorithm 7.1 Sample from Jeffreys Prior Distribution

Require: Initial guess θ_0 , constraint set Θ_c , step size τ , number of iterations

N_{iter} , small step size $h > 0$

- 1: **for** $k = 0, 1, \dots, N_{\text{iter}} - 1$ **do**
- 2: Compute or estimate $\mathcal{J}_{\theta_k}$
- 3: Compute $\nabla_{\theta} V(\theta_k)$ using (7.2.11) **or**:
- 4: Draw a random direction $\nu \sim \mathcal{N}(0_d, I_d)$
- 5: Estimate $\mathcal{J}_{\theta_k + h\nu}$
- 6: Estimate $\nabla_{\theta} V(\theta_k)$ using (7.2.12) and (7.2.11)
- 7: Sample $\xi_k \sim \mathcal{N}(0_d, I_d)$ and $U \sim \mathcal{U}(0, 1)$
- 8: **Propose:**

$$\theta' \leftarrow \theta_k - \tau \nabla_{\theta} V(\theta_k) + \sqrt{2\tau} \xi_k$$

$$\varrho_k \leftarrow \begin{cases} \varrho^{\text{MALA}}(\theta', \theta_k), & \text{if } \theta' \in \Theta_c, \\ 0, & \text{otherwise.} \end{cases}$$
- 9: **Accept/reject step:**

$$\theta_{k+1} \leftarrow \begin{cases} \theta', & \text{if } U < \varrho_k, \\ \theta_k, & \text{otherwise.} \end{cases}$$
- 10: **end for**
- 11: **return** $\{\theta_k\}_{k=1}^{N_{\text{iter}}}$

7.3 Numerical Illustrations

In this section, we present three numerical examples:

1. Sanity check: a simple example in which Jeffreys prior can be computed exactly, verifying that Algorithm 7.1 converges to the correct distribution;

2. NLSS system: a sampling example where the FIM is estimated via the PF method, showing that Algorithm 7.1 remains effective with estimated $\mathcal{I}_\theta$;
3. Parameter estimation: a practical setting in which Jeffreys prior sampling outperforms a uniform prior in data-driven parameter estimation performance.

7.3.1 Verification of the sampling procedure

7.3.1.1 Experiment setup

In this experiment, we apply Algorithm 7.1 to sample from Jeffreys prior for a Coin-Bending Model [83], where the probability $q(a)$ of obtaining heads depends on the bending angle a according to

$$q(a) = \frac{1}{2} + \frac{1}{2} \left(\frac{a}{\pi} \right)^3.$$

Each coin toss yields a binary random variable Y , where $Y = 1$ with probability $\frac{1}{2} + \frac{1}{2}(a/\pi)^3$, and $Y = 0$ otherwise, with 1 denoting heads. Its score function is

$$\nabla_a \ln f(Y; a) = \mathbb{1}\{Y = 1\}p_1 - \mathbb{1}\{Y = 0\}p_0,$$

with

$$p_1 = \frac{\frac{3}{\pi} \left(\frac{a}{\pi} \right)^2}{1 + \left(\frac{a}{\pi} \right)^3}, \quad p_0 = \frac{\frac{3}{\pi} \left(\frac{a}{\pi} \right)^2}{1 - \left(\frac{a}{\pi} \right)^3}.$$

The parameter to be estimated is a . Given N independent coin tosses resulting in observations $\mathbf{y} \in \{0, 1\}^N$, the information $\mathcal{I}_a$ per sample can be approximated, if N is sufficiently large, by

$$\hat{\mathcal{J}}_a = \frac{1^\top \mathbf{y}}{N} p_1^2 + \frac{N - 1^\top \mathbf{y}}{N} p_0^2.$$

Assuming consistency as $N \rightarrow \infty$, we treat $\pi(a) \propto \sqrt{\hat{\mathcal{J}}_a}$ as the ground truth, and define the potential $V(a) = -\frac{1}{2} \ln \hat{\mathcal{J}}_a$ whose gradient is given explicitly by

$$\nabla V(a) = -\frac{1}{2} \hat{\mathcal{J}}_a^{-1} \frac{d\hat{\mathcal{J}}_a}{da}.$$

We run Algorithm 7.1 to generate samples from $\pi(a)$ within the constrained set $a \in \Phi_c = [2, 3]$. Three experiments are performed with $N_{\text{iter}} \in \{100, 1000, 10000\}$ samples, each repeated for 10 independent realizations. The resulting empirical distributions are then compared with the estimated Jeffreys prior to verify convergence and consistency.

7.3.1.2 Simulation results

The simulation results are shown in Figure 7.3.1, which shows that the empirical distribution of samples obtained by our algorithm matches the theoretical Jeffreys prior as the number of samples increases (from Row 1 to Row 3).

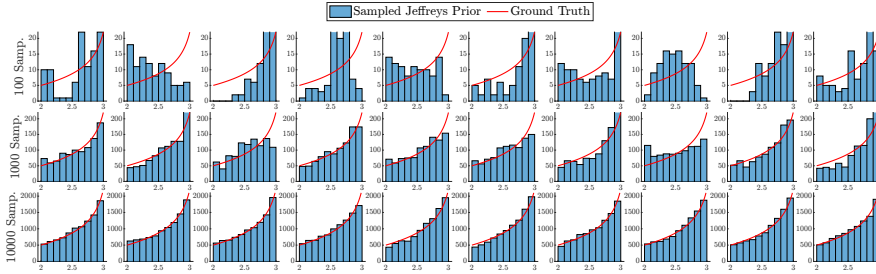


Figure 7.3.1: Histograms of 100 (top), 1000 (middle), and 10000 (bottom) samples of Jeffreys prior and the shape of the true Jeffreys prior (red curve) from the Coin-Bending Model. Each set of sampling experiments contains ten different realizations.

Moreover, only a relatively small sample size (e.g., 1000 samples) yields a very good match to the exact Jeffreys prior, indicating rapid convergence. Additionally, the difference in the sampled realizations (for $N_{\text{iter}} = 100$) reveals the stochastic nature of the chain, demonstrating good mixing behavior and adequate exploration of the parameter space.

This example provides a clear validation of our proposed MALA-based sampler, as it allows direct comparison between the empirical and exact prior distributions. Furthermore, this one-dimensional scenario enables straightforward assessment of mixing quality and acceptance rates.

7.3.2 Sampling Jeffreys Prior for a dynamical system

7.3.2.1 Experiment setup

In this example, we apply Algorithm 7.1 on an NLSS model, where the FIM is numerically approximated using a PF. Specifically, we consider the Hull–White stochastic volatility (SV) model [84, 85]

$$\begin{aligned} x_{t+1} &| x_t \sim \mathcal{N}(ax_t + \varrho u_t, \sigma_v^2), \\ y_t &| x_t \sim \mathcal{N}(0, \beta^2 e^{x_t}), \end{aligned}$$

where the control input u_t is white noise of standard normal distribution. Our goal is to sample the Jeffreys prior $\pi(a)$ for the system parameter a corresponding to the system dynamics, constrained to the feasible region $\Phi_c = [0.3, 0.9]$, ensuring system stability. For simulation, the remaining model parameters are

set as $[\varrho, \sigma_v, \beta] = [0.2, 0.5, 0.7]$, and the system is simulated for $T = 1000$ time steps.

Since the FIM $\mathcal{J}_a$ lacks an analytical form for this model, we estimate it using a PF-based method from [75], as described in Section 7.2, employing $N_p = 1000$ particles and averaging multiple Monte Carlo runs to obtain a reliable estimate $\hat{\mathcal{J}}_a$. Accordingly, we compute the estimated shape of Jeffreys prior $\pi(a)$ and compare it with the distribution of the generated samples.

We define the potential function as $V(a) = -\frac{1}{2} \ln \hat{\mathcal{J}}_a$ and approximate its gradient $\hat{\nabla} V(a)$ using (7.2.11) and (7.2.12).

Finally, Algorithm 7.1 is run for $N_{\text{iter}} = 10000$ iterations with step size $\tau = 0.05$, and the generated samples are compared against the estimated shape of Jeffreys prior.

7.3.2.2 Simulation results

A histogram of the 10000 generated samples is shown in Figure 7.3.2, overlaid with the estimated shape of Jeffreys prior $\pi(a)$ computed from the PF estimation.

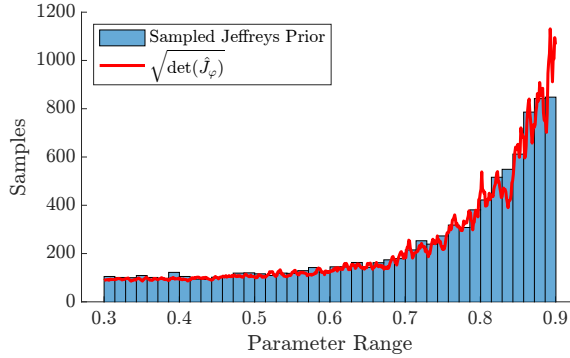


Figure 7.3.2: Histogram of 10000 samples of Jeffreys prior and the shape of the PF-estimated Jeffreys prior (red curve) from the SV Model.

As expected, the estimated Jeffreys prior assigns higher probability near the boundary $a = 0.9$, indicating greater parameter sensitivity, and lower probability near $a = 0.3$.

Around $a = 0.3$, the influence of x_t on the evolution of the state is relatively small; the dynamics are then dominated by the input u_t and the process noise v_t , making the likelihood less sensitive to changes in a . In contrast, around $a = 0.9$, a small perturbation in a produces significant changes in the predicted

states and, hence, in the likelihood function, leading to a higher FIM. Thus, the PF-based FIM estimation is consistent with theoretical expectations.

Moreover, the close agreement observed in Figure 7.3.2 between the histogram of generated samples and the estimated Jeffreys prior verifies that our proposed algorithm reliably samples from Jeffreys prior even when the FIM is numerically approximated. Overall, the results from experiments 1 and 2 demonstrate the validity of our sampling framework.

7.3.3 An application of Jeffreys prior to parameter estimation

7.3.3.1 Experiment setup

In this example, we illustrate the advantages of sampling from Jeffreys prior within the Two-Stage (TS) estimation framework, using parameter estimation for a Weibull distribution as a test case. TS provides an effective validation scenario, although the framework itself is well-established. In TS, given a parametric model $f(\cdot; \theta)$ with parameters $\theta \in \Theta \subseteq \mathbb{R}^d$, one constructs a synthetic training dataset of parameter-data pairs $\{(\theta^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^M$, where data samples $\mathbf{y}^{(i)}$ are generated from $f(\cdot; \theta^{(i)})$. A supervised learning method then builds an estimator of the form $T_{\text{TS}}(\mathbf{y}) = \psi(\varphi(\mathbf{y}))$, with a fixed compression function φ and a regression-based function ψ . For additional details, see Chapter 3.

In the context of TS, the distribution of the synthetic parameter samples is crucial. We advocate sampling θ according to Jeffreys prior, since $\pi(\theta)$ naturally reflects the local identifiability structure of the model, thus producing synthetic data which is most informative about the parameters. In contrast, a uniform prior may not account for how informative different regions of the parameter space are, potentially leading to suboptimal training datasets.

As a numerical example, we consider the Weibull distribution, widely used in reliability engineering, whose probability density function, parameterized by $\theta = [\lambda, \kappa]^\top$, is given by

$$f(A; \lambda, \kappa) = \frac{\kappa}{\lambda} \left(\frac{A}{\lambda} \right)^{\kappa-1} \exp \left[- \left(\frac{A}{\lambda} \right)^\kappa \right], \quad A \geq 0,$$

where $\lambda > 0$ and $\kappa > 0$ are the scale and shape parameters, respectively.

7.3.3.2 Simulation results

We validate the TS estimators trained under uniform and Jeffreys priors using a validation set consisting of 1000 parameter points $\theta_\ell = [\lambda_\ell, \kappa_\ell]^\top$ uniformly sampled from $[1, 20] \times [1, 20]$. For each θ_ℓ , synthetic data $\{\mathbf{y}_\ell^{(i)}\}_{i=1}^M$ is generated from the Weibull model. We evaluate two classes of TS estimators based on the samples from the uniform and Jeffreys priors, respectively.

Figure 7.3.3(a) shows the sampled Jeffreys distribution in the (κ, λ) parameter space. Notably, Jeffreys prior emphasizes lower values of κ while remaining relatively uniform along λ , aligning well with the Weibull model’s FIM structure, which indicates higher sensitivity at smaller κ .

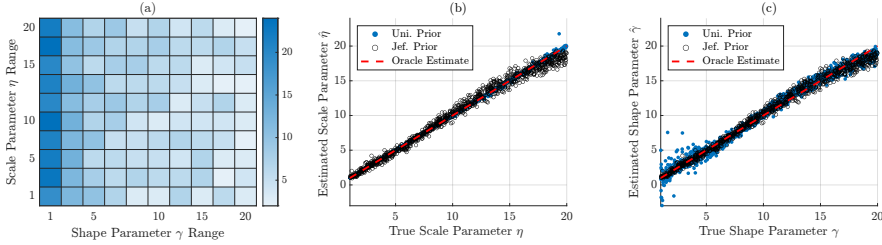


Figure 7.3.3: (a) Heatmap of 1000 samples from Jeffreys prior distribution of the scale (λ) and shape (κ) parameters from a Weibull distribution; (b) Scatter plot of estimated scale parameter $\hat{\lambda}$ based on the uniform prior and Jeffreys prior vs. its true value; (c) Scatter plot of estimated shape parameter $\hat{\kappa}$ based on the uniform prior and Jeffreys prior vs. its true value. The red dashed line corresponds to an oracle estimate, which knows the true value of the parameter.

Estimation performances for λ and κ are compared in Figures 7.3.3(b) and 7.3.3(c), respectively. In Figure 7.3.3(b), both uniform and Jeffreys-based estimators produce accurate and similar results for the scale parameter across its entire range, because Jeffreys prior is nearly uniform in λ and thus provides coverage comparable to the uniform prior.

However, notable differences arise for the shape parameter κ , as seen in Figure 7.3.3(c). For $\kappa < 5$, the uniform-based estimator exhibits significant variance and bias. In contrast, the Jeffreys-based estimator has considerably higher accuracy in this region, since Jeffreys prior focuses sampling efforts where the FIM is greatest. Remarkably, despite allocating fewer training samples to higher κ regions, the Jeffreys-based estimator maintains effective generalization in those regions, showing the advantage of information-driven weighting in the parameter space.

Overall, these experiments demonstrate that sampling according to Jeffreys prior within the TS framework substantially enhances estimation accuracy and robustness for shape-dominated parameters, while maintaining comparable performance to the uniform-based estimator along directions of weaker sensitivity. This also suggests that our approach may have broader applicability in synthetic data generation and parameter estimation for complex models.

7.4 Conclusion

In this chapter, we have proposed a MALA-based scheme designed to generate samples from Jeffreys prior. Our approach imposes Jeffreys prior as the stationary distribution of a Langevin-based MCMC, where each update step of the Markov chain is determined by the gradient of the logarithm of the determinant of the FIM. Furthermore, for the case when this matrix cannot be computed analytically, we have employed a PF algorithm to approximate the score function, and thus the gradient of the log determinant of the FIM. We have validated our sampling scheme through several numerical examples, including one that has demonstrated how the diversity inherent in Jeffreys prior can enhance the performance of a data-driven estimator.

These results provide a principled approach to training data generation that leverages the statistical structure of the estimation problem. In the following chapters, we explore complementary approaches to improving TS estimator performance: supervised and self-supervised fine-tuning methods that adapt pretrained estimators to out-of-distribution scenarios.

Part 2: Fine-Tuning a Simulation-Driven Estimator

Abbreviations

Adam	Adaptive Moment Estimation (optimizer)
AR	autoregressive
ARMAX	autoregressive moving-average with exogenous input
ARX	autoregressive with exogenous input
DT	digital twin
EKF	extended Kalman filter
FIM	Fisher information matrix
GN	Gauss–Newton
i.i.d.	independent and identically distributed
MC	Monte Carlo
MLP	multi-layer perceptron
MSE	mean squared error
OOD	out-of-distribution
PEM	prediction error method
PRBS	pseudo-random binary sequence
ReLU	rectified linear unit
ResNet	residual network
SISO	single-input single-output
SNR	signal-to-noise ratio
TS	two-stage method
VDP	Van der Pol
w.r.t.	with respect to

Notation and Symbols

$\mathcal{M}(\theta)$	Simulator/model parameterized by θ
θ_0	True parameter
Θ	Parameter space
Θ_p	Calibration set
d	Dimension of the parameter space
N	Length of observation sequence
M	Number of training samples
$\mathbf{z}$	Input-output trajectory $(u_1, y_1, \dots, u_N, y_N)$
$\mathbf{z}_0$	Observed trajectory from the true system
φ	First-stage compression function
n	Dimension of compressed features
$\boldsymbol{\eta}$	Auxiliary statistics (compressed features)
$\boldsymbol{\eta}_{\text{obs}}$	Observed features
ψ_{pre}	Pretrained second-stage neural network
$T_{\text{TS}}^{\text{pre}}$	Pretrained TS estimator
$T_{\text{TS}}^{\text{FT}}$	Fine-tuned TS estimator
$\hat{\boldsymbol{\theta}}_{\text{init}}$	Initial estimate from pretrained TS
$\hat{\boldsymbol{\theta}}_{\text{GN}}$	Gauss–Newton refined estimate
$\mathbf{w}_T$	Trunk parameters
$\mathbf{w}_H$	Head parameters
$\hat{\mathbf{w}}_H^{\text{FT}}$	Fine-tuned head parameters

$\mathbf{S}$	Feature covariance matrix
$\mathbf{D}$	Whitened Jacobian
$\mathbf{G}$	Approximate FIM
s_{obs}	Whitened test statistic for OOD detection
$q_{1-\alpha}$	Bootstrap quantile threshold
$\mathcal{H}_0$	Null hypothesis (in-distribution)
$\mathcal{E}$	Confidence ellipsoid
$\mathcal{D}_{\text{FT}}$	Fine-tuning dataset
M_{FT}	Number of fine-tuning samples
$\mathbf{\Omega}$	All trainable weights of the ResNet
$\mathbf{\Omega}_{\text{FT}}$	Fine-tuned ResNet weights
ℓ_{sim}	Simulation error / trajectory MSE
K	Number of simulations for OOD detection
K'	Number of parallel simulations for unsupervised fine-tuning
$\mathcal{U}[a, b]$	The uniform distribution on $[a, b]$

8 Supervised Fine-Tuning of a Simulation-Driven Estimator

The preceding chapters established the theoretical foundations of the Two-Stage (TS) estimator: its statistical framework in Chapter 3, asymptotic properties in Chapter 5, finite-sample guarantees in Chapter 6, and training data generation via Jeffreys prior sampling in Chapter 7. A recurring assumption throughout that development is that the training parameters $\{\theta^{(i)}\}_{i=1}^M$ are drawn from the full parameter space Θ , and that the true parameter θ_0 also belongs to Θ . Under this assumption, the pretrained TS estimator enjoys all the consistency, asymptotic normality, and finite-sample guarantees derived earlier.

In practice, however, one typically does not sample from the entirety of Θ . The parameter space $\Theta \subseteq \mathbb{R}^d$ is defined by physical or structural constraints of the model and may be considerably larger than the region of practical interest. Instead, the practitioner constructs a *calibration set* $\Theta_p \subseteq \Theta$, which is a bounded subset shaped by prior domain knowledge, historical operating data, or engineering judgment about the plausible regime of the true system, and draws training samples exclusively from Θ_p . The choice of Θ_p is inherently subjective: it reflects what is *believed* about the system rather than what is *known*, and there is no guarantee that the true parameter θ_0 falls within it.

The distinction between Θ and Θ_p is the source of the problem addressed in this chapter. When the true parameter θ_0 lies inside Θ_p , the pretrained estimator $T_{\text{TS}}^{\text{pre}}$ benefits from the guarantees established in earlier chapters, since the training distribution covers a neighborhood of θ_0 . However, if the prior knowledge encoded in Θ_p is incomplete because the system has drifted, been reconfigured, or was simply less well understood than assumed, then θ_0 may

fall outside Θ_p while still belonging to the physically valid space Θ . In this *out-of-distribution (OOD)* scenario, the pretrained TS estimator faces an extrapolation problem: the second stage ψ_{pre} has never seen features corresponding to parameters near θ_0 , and its predictions can degrade significantly, even though the underlying model $\mathcal{M}(\theta)$ and the first-stage compressor φ remain perfectly valid at θ_0 .

This chapter addresses the OOD problem through *supervised fine-tuning*: a transfer learning approach that adapts the pretrained TS estimator using the model structure and new observations from the true system. In previous chapters, the second stage of the TS estimator was implemented using kernel regression and gradient boosting; more broadly, any supervised learner is a valid choice for this stage. In this chapter, we specifically choose ψ_{pre} to be a neural network, as its layered parametric structure enables a fine-tuning strategy, selectively updating subsets of weights, that is not readily available with other supervised learning methods. The key insight is that the two-stage structure of the TS estimator enables efficient adaptation: we keep the first-stage compression function φ fixed and fine-tune only the final layers of ψ_{pre} , following standard transfer learning practice where early layers capture generic structure and are preserved [86].

The contributions of this chapter are fourfold: we derive a feature-space OOD test based on DT-derived feature statistics and whitening; we propose a fine-tuning approach that combines Gauss–Newton iterations with a confidence set to generate a fresh synthetic dataset for retraining the second stage of TS; we design a minimal update strategy that freezes the trunk and adjusts only the final head layers; and we numerically validate the proposed approach against standard baselines including PEM and dual-EKF.

8.1 Problem Statement

We consider a data-generating mechanism $\mathcal{M}(\theta)$, a high-fidelity simulator or DT, parameterized by $\theta \in \Theta \subseteq \mathbb{R}^d$. For notational simplicity, we assume a single-input single-output (SISO) setting: given an N -length *scalar* input sequence $\mathbf{u} = (u_1, \dots, u_N)^\top \in \mathbb{R}^N$, the DT produces a (possibly stochastic) *scalar* output sequence $\mathbf{y} = (y_1, \dots, y_N)^\top \in \mathbb{R}^N$. Concretely, $\mathbf{y} = \mathbf{y}(\theta; \omega)$, where ω denotes the simulator’s noise seed (or any internal randomness). Assume that a length- N trajectory

$$\mathbf{z}_0 = (u_1, y_{0,1}, \dots, u_N, y_{0,N}) \in \mathbb{R}^{2N}$$

has been measured from the DT at some unknown parameter $\theta_0 \in \Theta$. The goal is to estimate θ_0 from $\mathbf{z}_0$ by constructing a *simulation-driven parameter estimator*.

Given a *calibration set* $\Theta_p \subseteq \Theta$, constructed from prior domain knowledge and possible operating regimes, we draw independent and identically distributed

(i.i.d.) parameter settings $\{\theta^{(i)}\}_{i=1}^M$ from Θ_p . For each draw $\theta^{(i)}$, we run the simulator with (possibly) its own noise seed ω_i and the chosen input sequence¹ $\mathbf{u}$, obtaining $\mathbf{z}^{(i)} = (u_1, y_{i,1}, \dots, u_N, y_{i,N})$. This yields a synthetic, labeled dataset $\mathcal{D} = \{(\mathbf{z}^{(i)}, \theta^{(i)})\}_{i=1}^M$.

An estimator $T_{\text{TS}}^{\text{pre}}: \mathbb{R}^{2N} \rightarrow \mathbb{R}^d$ is then fitted by solving the following minimization problem over a prescribed function class $\mathcal{F}$ (e.g., a neural network):

$$T_{\text{TS}}^{\text{pre}} \in \arg \min_{F \in \mathcal{F}} \frac{1}{M} \sum_{i=1}^M \ell(F(\mathbf{z}^{(i)}), \theta^{(i)}), \quad (8.1.1)$$

where $\ell: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$ is a suitable loss function. Following Eq. (8.1.1), an estimate of θ_0 is obtained as

$$\hat{\theta} = T_{\text{TS}}^{\text{pre}}(\mathbf{z}_0), \quad (8.1.2)$$

where $\mathbf{z}_0$ are the observations (input-output data) collected from the system $\mathcal{M}(\theta_0)$.

When $\theta_0 \in \Theta_p$, suitable choices of $\mathcal{F}$ and ℓ can lead to an accurate estimate $T_{\text{TS}}^{\text{pre}}(\mathbf{z}_0)$ of θ_0 . However, because Θ_p is user-specified and may omit plausible regimes, an *out-of-distribution* (OOD) situation may occur if $\theta_0 \notin \Theta_p$, or if there are not enough samples $\theta^{(i)}$ in the neighborhood of θ_0 . Hence, performance can degrade under an OOD, motivating mechanisms that refine $T_{\text{TS}}^{\text{pre}}$ at implementation phase. In this chapter, we focus on a special class of simulation-driven estimators $T_{\text{TS}}^{\text{pre}}$, known as the TS estimator.

8.2 The TS Estimator

The TS estimator [34] decomposes (8.1.1) into two-stages: a fixed *compression* of $\mathbf{z}$ into low-dimensional features, followed by a supervised machine learning model that maps these features to the parameters.

8.2.1 Stage 1 (Compression)

Choose a deterministic feature map

$$\varphi: \mathbb{R}^{2N} \rightarrow \mathbb{R}^n, \quad n \ll 2N, \quad (8.2.3)$$

e.g., the estimated coefficients of an AR/ARX/ARMAX model fitted to the data. Let $\boldsymbol{\eta}^{(i)} = \varphi(\mathbf{z}^{(i)})$ and form the synthetic dataset

$$\mathcal{D}_{\text{tr}} = \{(\boldsymbol{\eta}^{(i)}, \theta^{(i)})\}_{i=1}^M = \{(\varphi(\mathbf{z}^{(i)}), \theta^{(i)})\}_{i=1}^M \subset \mathbb{R}^n \times \mathbb{R}^d. \quad (8.2.4)$$

¹The input sequence is considered fixed, even though the treatment can be extended to random inputs.

8.2.2 Stage 2 (Supervised mapping)

Let $\boldsymbol{\eta} = \varphi(\mathbf{z}) \in \mathbb{R}^n$ be the Stage-1 feature vector. The Stage-2 function $\psi_{\text{pre}}: \mathbb{R}^n \rightarrow \mathbb{R}^d$ is a fully connected deep neural network with *trunk* and *head* layers.

8.2.2.1 Trunk Layers

The trunk of the second stage neural network is mathematically described as follows: Let $\mathbf{c}^{(0)} = \boldsymbol{\eta}$ be the input layer and for $r = 1, \dots, R-1$, the output of the r^{th} hidden layer is given by

$$\mathbf{c}^{(r)} = \sigma_r(\mathbf{W}_r \mathbf{c}^{(r-1)}), \quad \mathbf{W}_r \in \mathbb{R}^{p_r \times p_{r-1}},$$

where $\mathbf{W}_r$ denotes the weights of the hidden layer r , σ_r denotes an activation function (e.g., $\text{ReLU}(\cdot)$), $p_0 = n$ and $p_{R-1} = p$. These R layers constitute the *trunk* of ψ_{pre} . The trunk output is $\mathbf{t}(\boldsymbol{\eta}) := \mathbf{c}^{(R-1)} \in \mathbb{R}^p$ and its parameters are

$$\mathbf{w}_T := \{\mathbf{W}_r\}_{r=1}^{R-1}.$$

8.2.2.2 Head Layers

To finally predict the targets $\boldsymbol{\theta}^{(i)} \in \Theta_p \subseteq \mathbb{R}^d$, which can be multidimensional ($d > 1$), we append $V-1$ subsequent hidden layers for each parameter dimension, and these $V-1$ layers corresponding to each individual parameter constitute the *head* of ψ_{pre} , denoted as $\mathbf{h}_j^{(v)}$, where $v = 1, \dots, V-1$. We additionally impose that the weights of the first of these $V-1$ layers are *least correlated*, which is needed to ensure that each parameter is correctly identified. We precisely formalize the structure of the head as follows:

For each parameter index $j \in \{1, \dots, d\}$, starting from $\mathbf{h}_j^{(0)} = \mathbf{t}(\boldsymbol{\eta})$ and for $v = 1, \dots, V-1$ define

$$\mathbf{h}_j^{(v)} = \sigma_{j,v}(\mathbf{U}_{j,v} \mathbf{h}_j^{(v-1)}), \quad \mathbf{U}_{j,v} \in \mathbb{R}^{m_{j,v} \times m_{j,v-1}},$$

where $\mathbf{U}_{j,v}$ corresponds to the weights of hidden layer v of head j , and $\sigma_{j,v}$ corresponds to the activation function of v^{th} hidden layer of head j , with $m_{j,0} = p$. The scalar output for head j is linear:

$$\hat{\theta}_j(\boldsymbol{\eta}) = \mathbf{a}_j^\top \mathbf{h}_j^{(V-1)}, \quad \mathbf{a}_j \in \mathbb{R}^{m_{j,V-1}}.$$

The head parameters are then defined as

$$\boldsymbol{\xi}_j := \{(\mathbf{U}_{j,v})_{v=1}^{V-1}, \mathbf{a}_j\},$$

and $\mathbf{w}_H := (\xi_1, \dots, \xi_d)$. Stacking the d heads yields the second stage output as

$$\psi_{\text{pre}}(\boldsymbol{\eta}; \mathbf{w}_T, \mathbf{w}_H) := \begin{bmatrix} \hat{\theta}_1(\boldsymbol{\eta}) \\ \vdots \\ \hat{\theta}_d(\boldsymbol{\eta}) \end{bmatrix} \in \mathbb{R}^d, \quad [\psi_{\text{pre}}(\boldsymbol{\eta}; \mathbf{w}_T, \mathbf{w}_H)]_j := \hat{\theta}_j(\boldsymbol{\eta}). \quad (8.2.5)$$

Given the training dataset $\mathcal{D}_{\text{tr}} = \{(\boldsymbol{\eta}^{(i)}, \boldsymbol{\theta}^{(i)})\}_{i=1}^M$, we learn $(\mathbf{w}_T, \mathbf{w}_H)$ by solving the following optimization problem via stochastic gradient descent:

$$\begin{aligned} (\hat{\mathbf{w}}_T, \hat{\mathbf{w}}_H) \in \arg \min_{\mathbf{w}_T, \mathbf{w}_H} \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^d \rho_j \tilde{\ell}(\hat{\theta}_j(\boldsymbol{\eta}^{(i)}), [\boldsymbol{\theta}^{(i)}]_j) \\ + \lambda_{\text{orth}} \sum_{1 \leq j < k \leq d} \|\mathbf{U}_{j,1} \mathbf{U}_{k,1}^\top\|_F^2, \end{aligned} \quad (8.2.6)$$

where $\rho_j \geq 0$ are weights chosen as $\rho_j = 1/\hat{\text{Var}}([\theta]_j)$, where

$$\hat{\text{Var}}([\theta]_j) := M^{-1} \sum_{i=1}^M ([\boldsymbol{\theta}^{(i)}]_j - \bar{\theta}_j)^2, \quad \bar{\theta}_j := M^{-1} \sum_{i=1}^M [\boldsymbol{\theta}^{(i)}]_j.$$

Here, $\tilde{\ell}$ is the scalar Huber loss [87], given by

$$\tilde{\ell}(x, y) = \begin{cases} \frac{1}{2}(x - y)^2, & \text{if } |x - y| \leq \delta, \\ \delta \left(|x - y| - \frac{1}{2}\delta \right), & \text{otherwise,} \end{cases}$$

with threshold $\delta > 0$. We choose this loss because it behaves like the squared error for small discrepancies, which makes it appropriate for measuring accuracy in low noise scenarios and like the absolute error for large discrepancies, thus making it robust to outliers. $[\boldsymbol{\theta}^{(i)}]_j$ denotes the j^{th} coordinate of $\boldsymbol{\theta}^{(i)}$, $\lambda_{\text{orth}} \geq 0$ is a regularization coefficient, and the last term encourages diversity among the heads by penalizing the correlations between their first-layer weights. Notice that the overall loss function ℓ in Eq. (8.1.1) is related to $\tilde{\ell}$ as

$$\ell(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}) := \sum_{j=1}^d \rho_j \tilde{\ell}([\hat{\boldsymbol{\theta}}]_j, [\boldsymbol{\theta}]_j) + \lambda_{\text{orth}} \sum_{1 \leq j < k \leq d} \|\mathbf{U}_{j,1} \mathbf{U}_{k,1}^\top\|_F^2.$$

The final TS estimator is the composition

$$T_{\text{TS}}^{\text{pre}}(z) = \psi_{\text{pre}}(\varphi(z); \hat{\mathbf{w}}_T, \hat{\mathbf{w}}_H) \in \mathbb{R}^d.$$

At implementation phase, given a new trajectory z_0 generated by $\mathcal{M}(\boldsymbol{\theta}_0)$, the estimate is

$$\hat{\boldsymbol{\theta}}_0 = T_{\text{TS}}^{\text{pre}}(z_0) = \psi_{\text{pre}}(\varphi(z_0); \hat{\mathbf{w}}_T, \hat{\mathbf{w}}_H). \quad (8.2.7)$$

If $\boldsymbol{\theta}_0 \notin \Theta_p$, then $(z_0, \boldsymbol{\theta}_0)$ is OOD w.r.t. $\mathcal{D}_{\text{tr}}$, and Eq. (8.2.7) may yield a poor estimate of $\boldsymbol{\theta}_0$. We shall now state a fine-tuning approach to overcome this problem.

8.3 Fine-Tuning the TS Estimator

We propose a *simulation-driven transfer learning* procedure to improve a pre-trained TS estimator under OOD observations. We keep the first stage feature extractor φ unchanged and *fine-tune only the last layers* of the second stage $\psi_{\text{pre}}(\cdot; w_T, w_H)$: we *freeze* the pretrained w_T and *update* only the head layers w_H using a small, targeted synthetic dataset from the DT in a neighborhood of the current TS estimate, following standard transfer learning practice i.e., earlier layers capture generic structure and are preserved, so only the final layers are tuned [86]. In summary, the procedure consists of the following steps:

- **Step 1 — Feature-space OOD detector.** Test whether the observed features $\eta_{\text{obs}} := \varphi(z_0)$ are statistically consistent with features generated by the simulator at the initial TS prediction $\hat{\theta}_{\text{init}} = \psi_{\text{pre}}(\varphi(z_0); \hat{w}_T, \hat{w}_H)$. If consistent, skip fine-tuning and deploy $T_{\text{TS}}^{\text{pre}}$.
- **Step 2 — Synthetic data generation.** If an OOD is detected, perform a few Gauss–Newton (GN) iterations in feature space, regularized toward $\hat{\theta}_{\text{init}}$, to obtain an improved estimate $\hat{\theta}_{\text{GN}}$. Then, derive a confidence region around $\hat{\theta}_{\text{GN}}$; sample an informative cloud of parameters from this region; and simulate the DT at these samples to form a small synthetic dataset of feature–parameter pairs.
- **Step 3 — Final-layer transfer.** With $\hat{w}_T$ frozen, update only $\hat{w}_H$ on the small synthetic dataset using a weighted loss. The result is an improved estimator $\psi_{\text{pre}}(\cdot; \hat{w}_T, \hat{w}_H^{\text{FT}})$ under the OOD, where $\hat{w}_H^{\text{FT}}$ are updated weights $\hat{w}_H$.

The fine-tuning procedure is illustrated in Figures 8.3.1–8.3.2, and the architecture of the second stage is shown in Fig. 8.3.3.

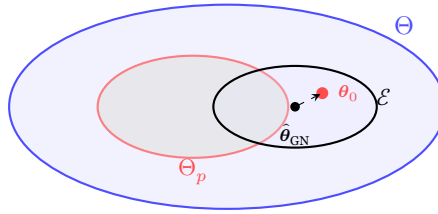


Figure 8.3.1: Illustration of an OOD scenario. The calibration set $\Theta_p \subseteq \Theta$ (shaded) is the parameter region over which the TS estimator was pretrained. The true parameter θ_0 lies outside Θ_p but within the confidence ellipsoid $\mathcal{E}$ centered at $\hat{\theta}_{\text{GN}}$.

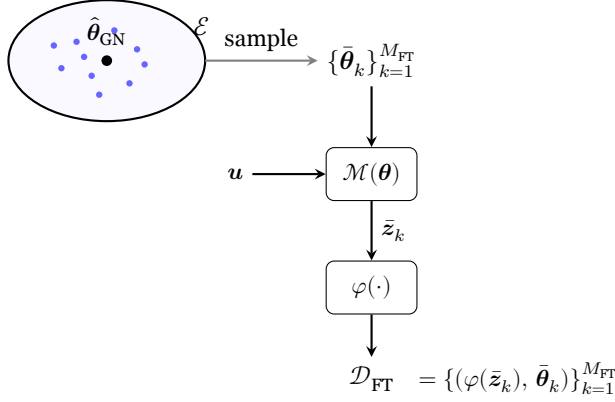


Figure 8.3.2: Retraining data generation for supervised fine-tuning. Parameters $\{\bar{\theta}_k\}_{k=1}^{M_{\text{FT}}}$ are sampled from the confidence ellipsoid $\mathcal{E}$ centered at $\hat{\theta}_{\text{GN}}$ (blue dots). Each sampled parameter is passed through the simulator $\mathcal{M}(\theta)$ with input u to produce trajectories $\bar{z}_k$, which are compressed by $\varphi(\cdot)$ to form $\mathcal{D}_{\text{FT}}$.

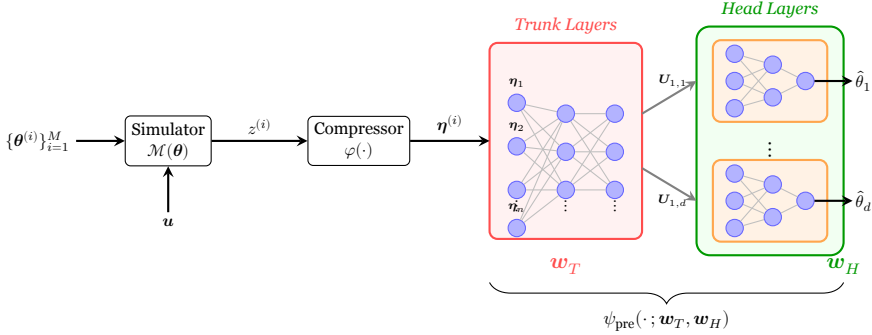


Figure 8.3.3: Architecture of the pretrained TS estimator $\psi_{\text{pre}}(\cdot; w_T, w_H)$. Training parameters $\{\theta^{(i)}\}_{i=1}^M$ are fed to the simulator $\mathcal{M}(\theta)$ along with input u to generate trajectories $z^{(i)}$. The compressor $\varphi(\cdot)$ produces features $\eta^{(i)} \in \mathbb{R}^n$, which are processed by the trunk layers parameterized by w_T and d independent head layers parameterized by $w_H = (\xi_1, \dots, \xi_d)$ to produce estimates $\hat{\theta}_1, \dots, \hat{\theta}_d$ for each parameter component. During fine-tuning, only the head parameters w_H are updated while w_T remains frozen.

8.3.1 Step 1: Feature-space OOD detector

We now provide a procedure to detect an OOD scenario. To this end, we appeal to a bootstrapped version of Wald's test [88, 89, 90]. The null hypothesis, $\mathcal{H}_0$, is that the observed features, η_{obs} , are statistically consistent with the features generated by the simulator when its parameter is set to the initial estimate, $\theta_{\text{init}} := \psi_{\text{pre}}(\eta_{\text{obs}}; \hat{w}_T, \hat{w}_H)$.

The procedure begins by computing the initial parameter estimate from the new observations, z_0 , generated by the system $\mathcal{M}(\theta_0)$, where θ_0 is unknown. This is done by first extracting its features, $\eta_{\text{obs}} := \varphi(z_0)$, and then applying the pre-trained second-stage estimator, outputting initial estimate of θ_0 as

$$\hat{\theta}_{\text{init}} := \psi_{\text{pre}}(\eta_{\text{obs}}; \hat{w}_T, \hat{w}_H).$$

Next, we fix the simulator's parameter at $\hat{\theta}_{\text{init}}$ and run K simulations, each with a different random seed ω_k . Applying the feature extractor φ to each of the K resulting trajectories yields a $K \times n$ matrix of simulated features:

$$H := ((\eta^{(1)})^\top, \dots, (\eta^{(K)})^\top)^\top,$$

where $\eta^{(k)} = \varphi(z^{(k)})$, and $z^{(k)}$ is the input–output trajectory generated by the simulator $\mathcal{M}(\hat{\theta}_{\text{init}})$ at noise seed ω_k , for $k = 1, \dots, K$.

This matrix, H , characterizes the expected variability of the features if the true parameter were indeed $\hat{\theta}_{\text{init}}$. To construct a test statistic, we compute a whitened discrepancy measure. From the simulated features H (at parameter $\hat{\theta}_{\text{init}}$), estimate the feature mean and regularized covariance as functions of θ :

$$\begin{aligned} \bar{\eta} &:= \frac{1}{K} \sum_{k=1}^K H_{k:}^\top, \\ S &:= \frac{1}{K-1} \sum_{k=1}^K [\eta^{(k)} - \bar{\eta}][\eta^{(k)} - \bar{\eta}]^\top + \varepsilon I_n, \quad \varepsilon > 0, \end{aligned} \tag{8.3.8}$$

where $H_{k:} = (\eta^{(k)})^\top$, and I_n is an $n \times n$ identity matrix. The covariance matrix is then used to define a whitening transform, $S^{-1/2}$, which normalizes the feature space by accounting for correlations and different scales among the features. The test statistic for our observed data is the whitened squared residual norm:

$$s_{\text{obs}} = \|S^{-1/2}(\eta_{\text{obs}} - \bar{\eta})\|_2^2. \tag{8.3.9}$$

Finally, we establish a critical value for this statistic using the bootstrap principle. We compute the same whitened statistic for each of the K simulated feature vectors:

$$s_k = \|S^{-1/2}(H_{k:}^\top - \bar{\eta})\|_2^2. \tag{8.3.10}$$

The set $\{s_k\}_{k=1}^K$ provides an empirical distribution of the test statistic under $\mathcal{H}_0$. We define our decision threshold, $q_{1-\alpha}$, as the empirical $(1 - \alpha)$ -quantile of this distribution.

Decision rule: If $s_{\text{obs}} > q_{1-\alpha}$, we reject the null hypothesis $\mathcal{H}_0$, conclude that an OOD scenario is likely, and proceed to Step 2. Otherwise, the observation is deemed statistically consistent with the variability at $\hat{\theta}_{\text{init}}$, and we retain the pretrained estimator without fine-tuning.

8.3.2 Step 2: Synthetic data generation

If $s_{\text{obs}} > q_{1-\alpha}$, then the compressed features from the first-stage of TS evaluated at $\hat{\theta}_{\text{init}}$ fail to explain η_{obs} in the whitened space. In that case, we refine $\hat{\theta}_{\text{init}}$ by running GN iterations that minimize the whitened feature residual between the observed feature and the first-stage generated feature. Because this residual is random, the GN estimate, denoted $\hat{\theta}_{\text{GN}}$, still carries parameter uncertainty. To effectively fine-tune the second stage of the TS estimator, we need to generate a cloud of synthetic data that covers the region of plausible parameter values around this estimate. We shall now describe precisely the synthetic dataset generation for fine-tuning the second stage of TS:

GN update We compute the GN update starting from the TS initial estimate $\hat{\theta}_{\text{init}}$. We define a seed-averaged simulator feature map $\bar{f}(\theta)$ and its corresponding whitened residual $r(\theta)$:

$$\begin{aligned}\bar{f}(\theta) &:= \frac{1}{K_{\text{avg}}} \sum_{\omega \in \mathcal{S}_{\text{avg}}} \varphi(z(\theta, \omega; \mathbf{u})), \\ r(\theta) &:= S^{-1/2}(\bar{f}(\theta) - \eta_{\text{obs}}) \in \mathbb{R}^n,\end{aligned}\tag{8.3.11}$$

where $z(\theta, \omega; \mathbf{u})$ is the simulator trajectory for parameter θ and seed ω . We then minimize the following regularized objective function using iterative GN steps (with a backtracking line search):

$$v(\theta) = \frac{1}{2} \|r(\theta)\|_2^2 + \frac{\gamma}{2} \|\theta - \hat{\theta}_{\text{init}}\|_2^2, \quad \gamma > 0.\tag{8.3.12}$$

A few iterations of minimizing this objective yield the refined estimate $\hat{\theta}_{\text{GN}}$. Let $\partial \bar{f}(\theta) / \partial \theta \in \mathbb{R}^{n \times d}$ denote the Jacobian of the seed-averaged feature map, and let $S(\theta)$ be the feature covariance estimated from simulator draws at parameter θ , analogous to Eq. (8.3.8). Define the whitened Jacobian

$$\mathbf{D} := S^{-1/2} \frac{\partial \bar{f}(\theta)}{\partial \theta}.$$

We re-compute $\bar{\eta}$, S and $\mathbf{D}$ at $\hat{\theta}_{\text{GN}}$, and set

$$\mathbf{G} := \mathbf{D}^\top \mathbf{D} + \lambda \mathbf{I}_d,\tag{8.3.13}$$

where $\lambda = 10^{-3}$ is a small regularization constant ensuring $\mathbf{G}$ is invertible.

Sampling procedure $\hat{\theta}_{\text{GN}}$ is a point estimate derived from a single, noisy observation and is therefore random. To effectively fine-tune the TS estimator, we must generate a cloud of synthetic data that explores the region of plausible

parameter values around this estimate. We employ a hybrid sampling strategy that combines a *confidence-ellipsoid* based method with an alternative sampling scheme.

Our primary approach is to sample from the approximate $1 - \beta$ confidence ellipsoid for the true parameter θ_0 . This region is defined by the set of all θ satisfying:

$$(\theta - \hat{\theta}_{\text{GN}})^\top \mathbf{G}(\theta - \hat{\theta}_{\text{GN}}) \leq \chi_d^2(\beta), \quad (8.3.14)$$

where $\chi_d^2(\beta)$ is the chi-squared quantile and $\mathbf{G}$ is the approximate Fisher Information Matrix (FIM). This ellipsoid represents a high-confidence region, and sampling from it is statistically efficient for exploring well-identified parameters. However, this primary method can fail if the model is insensitive to one or more parameters. In such cases, the FIM, $\mathbf{G}$, becomes ill-conditioned, causing the confidence ellipsoid to be stretched along the insensitive directions, and thereby making the sampling procedure numerically unstable. To overcome this, we alternate between this sampling approach and *sensitivity* sampling. The latter sampling procedure handles the ill-conditioned directions. It works by generating samples along individual parameter axis with a step size that is inversely proportional to the simulator's sensitivity to that parameter. The sensitivity of coordinate j is defined as $s'_j := \|\mathbf{D}\mathbf{e}_j\|_2$. To set the sampling scale, we define two quantities. First, $\Delta_{\text{misfit}} := \|\mathbf{r}(\hat{\theta}_{\text{GN}})\|_2$, measures the discrepancy between the observed summary statistic and the one generated by the simulator at the point estimate $\hat{\theta}_{\text{GN}}$. Second, we compute $d_{\text{med}} := \text{median}_k d_k$, based on the spread of pre-computed summary features, where each $d_k := \|\mathbf{S}^{-1/2}(\mathbf{H}_{k,:} - \bar{\eta})\|_2$ represents the whitened residual. We then set coordinate scales, r_j , that increase exploration for less sensitive parameters (small s'_j) while also adapting to the simulator's noise level and the current feature mismatch as

$$r_j = \frac{d_{\text{med}}}{s'_j} \Delta_{\text{misfit}}. \quad (8.3.15)$$

Our sampling procedure combines a confidence ellipsoid and sensitivity sampling, yielding a hybrid sampling scheme, given by $\bar{\theta}_k = \hat{\theta}_{\text{GN}} + \Delta\theta_k$, where $\Delta\theta_k$ is drawn as

$$\begin{aligned} \text{(confidence set sampling)} \quad \Delta\theta_k &= \mathbf{B}z, \quad z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad \mathbf{B}\mathbf{B}^\top \approx \mathbf{G}^{-1}, \\ \text{(sensitivity sampling)} \quad \text{pick } i &\sim \text{Unif}\{1, \dots, d\}, \quad \zeta \sim \mathcal{N}(0, r_i^2), \quad \Delta\theta_k = \zeta \mathbf{e}_i, \end{aligned} \quad (8.3.16)$$

where $\mathbf{e}_i \in \mathbb{R}^d$ is the i -th standard basis vector (with a 1 at position i , zeros elsewhere), and $\mathbf{B}$ is the Cholesky factor of $\mathbf{G}^{-1}$.

To create the *fine-tuning dataset*, $\mathcal{D}_{\text{FT}}$, we generate a synthetic observation $\bar{z}_k$ from the simulator for each parameter sample $\bar{\theta}_k$. Each observation is then compressed using φ , forming a dataset of pairs: $\mathcal{D}_{\text{FT}} = \{(\varphi(\bar{z}_k), \bar{\theta}_k)\}_{k=1}^{M_{\text{FT}}}$, where M_{FT} is the number of new synthetic samples.

Remark 8.1. The quantity Δ_{misfit} can be artificially small due to noise induced by the randomness of the simulator $\mathcal{M}$, so the naive radius $\Delta_{\text{misfit}}/s'_j$ would under-explore. Multiplying by d_{med} rescales the step to the *typical* feature fluctuation at $\hat{\theta}_{\text{init}}$, giving $r_j = (d_{\text{med}}/s'_j) \Delta_{\text{misfit}}$. This preserves the desired inverse-sensitivity $(1/s'_j)$ while making the samples noise-aware.

Remark 8.2 (Theoretical justification of Steps 1 and 2). For Step 1, under the null hypothesis (in-distribution), the observed features η_{obs} and the simulated features $\eta^{(k)}$ are asymptotically normal with the same mean and covariance. This implies that their whitened residuals (s_{obs} and s_k) converge to the same χ^2 distribution, validating the bootstrap test. For Step 2, the GN objective $v(\theta)$ is an M-estimator whose expectation is minimized at the true θ_0 . Therefore, the GN estimate $\hat{\theta}_{\text{GN}}$ is consistent [66, Ch. 5], and the FIM $\mathbf{G}$, as a consistent estimator of the asymptotic covariance, defines an asymptotically valid confidence ellipsoid for sampling. More details follow in Section 8.4.

8.3.3 Step 3: Final-layer transfer on a targeted set

Finally, we freeze the early layers of $\psi_{\text{pre}}(\cdot; \mathbf{w}_T, \mathbf{w}_H)$ i.e., keep $\mathbf{w}_T$ fixed, and modify only the last layers, yielding $\psi_{\text{FT}}(\cdot; \mathbf{w}_T, \mathbf{w}_H^{\text{FT}})$. We fit $\mathbf{w}_H^{\text{FT}}$ on $\mathcal{D}_{\text{FT}}$ with a small number of gradient steps using a weighted objective (cf. (8.2.6)):

$$\begin{aligned} \hat{\mathbf{w}}_H^{\text{FT}} \in \arg \min_{\mathbf{w}_H} \frac{1}{M_{\text{FT}}} \sum_{i=1}^{M_{\text{FT}}} \sum_{j=1}^d \rho_j \tilde{\ell}([\psi_{\text{pre}}(\varphi(\bar{\mathbf{z}}_i); \mathbf{w}_T, \mathbf{w}_H)]_j, [\bar{\theta}_i]_j) \\ + \lambda_{\text{orth}} \sum_{1 \leq j < k \leq d} \|\mathbf{U}_{j,1} \mathbf{U}_{k,1}^\top\|_F^2, \end{aligned} \quad (8.3.17)$$

where $\rho_j = d [\mathbf{G}]_{jj} / \sum_{k=1}^d [\mathbf{G}]_{kk}$, with $[\mathbf{G}]_{jj}$ denoting the j -th diagonal entry of $\mathbf{G}$, and $\tilde{\ell}$ is the Huber loss function.

The fine-tuned TS estimator is then given by

$$T_{\text{TS}}^{\text{FT}}(\cdot) = \psi_{\text{pre}}(\varphi(\cdot); \mathbf{w}_T, \hat{\mathbf{w}}_H^{\text{FT}}). \quad (8.3.18)$$

8.4 Theoretical Justification for Fine-Tuning Step

We outline a proof sketch for the asymptotic validity of Wald's bootstrap test for OOD detection. Under the null hypothesis $\mathcal{H}_0$ (in-distribution),

$$\mathbb{P}_{\mathcal{H}_0}[s_{\text{obs}} > q_{1-\alpha} \mid \{s_k\}] \rightarrow \alpha \quad \text{as } K \rightarrow \infty. \quad (8.4.19)$$

The implication is that, under $\mathcal{H}_0$, with probability $1 - \alpha$ it is necessary that $s_{\text{obs}} \leq q_{1-\alpha}$; otherwise we declare an OOD.

To prove (8.4.19), we consider the asymptotic properties of the first stage of TS. Under standard prediction-error assumptions [5], the first stage features, chosen as AR/ARX coefficients fitted to the observations, are asymptotically consistent and normal, as discussed in Chapter 5. That is,

$$\boldsymbol{\eta}^{(k)} = \varphi(\mathbf{z}) \xrightarrow{a.s.} \bar{\boldsymbol{\eta}}^{(\hat{\boldsymbol{\theta}}_{\text{init}})} \quad \text{as } N \rightarrow \infty, \quad (8.4.20)$$

where $\mathbf{z} \sim \mathcal{M}(\hat{\boldsymbol{\theta}}_{\text{init}})$, and $\bar{\boldsymbol{\eta}}^{(\boldsymbol{\theta}_{\text{init}})} \in \mathbb{R}^n$ are the limit values of the AR/ARX coefficients, which are computed from the observations generated by $\mathcal{M}$ at $\hat{\boldsymbol{\theta}}_{\text{init}}$. Moreover, by standard prediction-error-method theory [5],

$$\sqrt{N}(\boldsymbol{\eta}^{(k)} - \bar{\boldsymbol{\eta}}^{(\hat{\boldsymbol{\theta}}_{\text{init}})}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \Sigma^{(\hat{\boldsymbol{\theta}}_{\text{init}})}) \quad \text{as } N \rightarrow \infty. \quad (8.4.21)$$

Furthermore, in Chapter 5, we showed that the TS estimator is consistent. Thus, when the observations $\mathbf{z}_0$ are generated by $\mathcal{M}(\boldsymbol{\theta}_0)$ and are in-distribution (i.e., $\boldsymbol{\theta}_0 \in \Theta_p$), asymptotic consistency implies that

$$\hat{\boldsymbol{\theta}}_{\text{init}} \xrightarrow{a.s.} \boldsymbol{\theta}_0 \quad \text{as } M, N \rightarrow \infty. \quad (8.4.22)$$

Hence, by the asymptotic normality of TS [27],

$$\begin{aligned} \boldsymbol{\eta}^{(k)} &\xrightarrow{a.s.} \bar{\boldsymbol{\eta}}^{(\boldsymbol{\theta}_0)} \quad \text{as } N \rightarrow \infty, \\ \sqrt{N}(\boldsymbol{\eta}^{(k)} - \bar{\boldsymbol{\eta}}^{(\boldsymbol{\theta}_0)}) &\xrightarrow{d} \mathcal{N}(\mathbf{0}, \Sigma^{(\boldsymbol{\theta}_0)}) \quad \text{as } N \rightarrow \infty. \end{aligned} \quad (8.4.23)$$

In addition, as $N \rightarrow \infty$, followed by $K \rightarrow \infty$ and $\varepsilon \rightarrow 0$,

$$\begin{aligned} \bar{\boldsymbol{\eta}} &:= \frac{1}{K} \sum_{k=1}^K \mathbf{H}_{k:}^\top \xrightarrow{a.s.} \bar{\boldsymbol{\eta}}^{(\boldsymbol{\theta}_0)}, \\ \mathbf{S} &:= \frac{1}{K-1} \sum_{k=1}^K [\boldsymbol{\eta}^{(k)} - \bar{\boldsymbol{\eta}}][\boldsymbol{\eta}^{(k)} - \bar{\boldsymbol{\eta}}]^\top + \varepsilon \mathbf{I}_n \xrightarrow{a.s.} \Sigma^{(\boldsymbol{\theta}_0)}. \end{aligned} \quad (8.4.24)$$

Combining (8.4.24) with Slutsky's theorem [41] yields

$$\mathbf{S}^{-1/2}(\boldsymbol{\eta}_{\text{obs}} - \bar{\boldsymbol{\eta}}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \mathbf{I}_n).$$

Similarly, by (8.4.23) and Slutsky's theorem,

$$\mathbf{S}^{-1/2}(\boldsymbol{\eta}^{(k)} - \bar{\boldsymbol{\eta}}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \mathbf{I}_n).$$

Thus, asymptotically, $s_{\text{obs}} = \|S^{-1/2}(\eta_{\text{obs}} - \bar{\eta})\|_2^2$ and $s_k = \|S^{-1/2}(\eta^{(k)} - \bar{\eta})\|_2^2$ have the same distribution, namely $\chi^2(n)$ with n degrees of freedom. Define the $(1 - \eta)$ empirical quantile

$$q_{1-\alpha} = \inf \left\{ t : F_K(t) := \frac{1}{K} \sum_{k=1}^K \mathbb{1}\{s_k \leq t\} \geq 1 - \eta \right\}.$$

By the Glivenko–Cantelli theorem [41, 66], as $N \rightarrow \infty$ and $K \rightarrow \infty$, $F_K(t) \xrightarrow{a.s.} F(t)$. Hence, assuming continuity of F , the empirical quantiles satisfy $q_{1-\alpha} \xrightarrow{a.s.} F^{-1}(1 - \eta)$ [66]. Therefore, under $\mathcal{H}_0$,

$$\mathbb{P}[s_{\text{obs}} > q_{1-\alpha}] \rightarrow \eta.$$

Analysis of the sampling procedure (Step 2).

Assuming that the regularizer γ in Eq. (8.3.12) is negligible, the Gauss–Newton objective is approximately

$$\begin{aligned} v(\theta) &= \frac{1}{2} \|\mathbf{r}(\theta)\|_2^2 \\ &= \frac{1}{2} [\bar{\mathbf{f}}(\theta) - \eta_{\text{obs}}]^\top S^{-1/2} [\bar{\mathbf{f}}(\theta) - \eta_{\text{obs}}]. \end{aligned} \tag{8.4.25}$$

By the law of large numbers, $S \xrightarrow{a.s.} \Sigma^{(\hat{\theta}_{\text{init}})}$ and $\bar{\mathbf{f}}(\theta) \xrightarrow{a.s.} \bar{\eta}(\theta)$. Hence, $\mathbb{E}[S] = \Sigma^{(\hat{\theta}_{\text{init}})}$, $\mathbb{E}[\bar{\mathbf{f}}(\theta)] = \bar{\eta}(\theta)$, and $\mathbb{E}[\bar{\mathbf{f}}(\theta_0)] = \bar{\eta}(\theta_0)$. It is then natural to view $v(\theta)$ as an M-estimator [66, Ch. 5].

Define $\mathcal{J}(\theta) = \mathbb{E}[v(\theta)]$. Then,

$$\begin{aligned} \mathcal{J}(\theta) &= \frac{1}{2} \mathbb{E}_{\eta_{\text{obs}}, \bar{\mathbf{f}}(\theta), S} [(\bar{\mathbf{f}}(\theta) - \eta_{\text{obs}})^\top S^{-1} (\bar{\mathbf{f}}(\theta) - \eta_{\text{obs}})] \\ &= \frac{1}{2} \mathbb{E}_{\eta_{\text{obs}}} [(\bar{\eta}(\theta) - \eta_{\text{obs}})^\top (\Sigma^{(\theta_{\text{init}})})^{-1} (\bar{\eta}(\theta) - \eta_{\text{obs}})] \\ &= \frac{1}{2} \left[\bar{\eta}(\theta)^\top (\Sigma^{(\theta_{\text{init}})})^{-1} \bar{\eta}(\theta) - 2\bar{\eta}(\theta)^\top (\Sigma^{(\theta_{\text{init}})})^{-1} \bar{\eta}(\theta_0) + \mathbb{E}_{\eta_{\text{obs}}} [\eta_{\text{obs}}^\top (\Sigma^{(\theta_{\text{init}})})^{-1} \eta_{\text{obs}}] \right], \end{aligned} \tag{8.4.26}$$

where

$$\mathbb{E}_{\eta_{\text{obs}}} [\eta_{\text{obs}}^\top (\Sigma^{(\theta_{\text{init}})})^{-1} \eta_{\text{obs}}] = \text{tr}((\Sigma^{(\theta_{\text{init}})})^{-1} \Sigma^{(\theta_0)}) + \bar{\eta}(\theta_0)^\top (\Sigma^{(\theta_{\text{init}})})^{-1} \bar{\eta}(\theta_0). \tag{8.4.27}$$

Thus

$$\mathcal{J}(\theta) = \frac{1}{2} (\bar{\eta}(\theta) - \bar{\eta}(\theta_0))^\top (\Sigma^{(\theta_{\text{init}})})^{-1} (\bar{\eta}(\theta) - \bar{\eta}(\theta_0)) + c, \tag{8.4.28}$$

where $c = \text{tr}((\Sigma^{(\theta_{\text{init}})})^{-1} \Sigma^{(\theta_0)})$ is a constant, and $\text{tr}(\cdot)$ denotes the trace of a square matrix.

Since the right-hand side of Eq. (8.4.28) is quadratic in $\bar{\eta}^{(\theta)}$, it is clear that θ_0 is the unique minimizer of $\mathcal{J}(\theta)$. Combining this with the fact that $\hat{\theta}_{\text{GN}}$ minimizes $v(\theta)$, we observe from standard M-estimation theory [66] that

$$\hat{\theta}_{\text{GN}} \xrightarrow{p} \theta_0.$$

That is, $\hat{\theta}_{\text{GN}}$ is a consistent estimator of θ_0 . It also follows that

$$\sqrt{N}(\hat{\theta}_{\text{GN}} - \theta_0) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \Sigma_{\text{GN}}). \quad (8.4.29)$$

(Here, the estimate $\hat{\theta}_{\text{GN}}$ depends on three limits $N \rightarrow \infty$, $K_{\text{avg}} \rightarrow \infty$, and $K \rightarrow \infty$; we implicitly assume in (8.4.29) that N grows faster than K_{avg} and K .)

The asymptotic covariance in (8.4.29) is

$$\Sigma_{\text{GN}} = \mathbf{D}(\theta_0)^{-1} \Sigma^{(\theta_0)} \mathbf{D}(\theta_0), \quad (8.4.30)$$

where $\mathbf{D}(\theta_0) = \frac{\partial \mathbb{E}[\varphi(z)]}{\partial \theta} \Big|_{\theta=\theta_0}$ and $z \sim \mathcal{M}(\theta)$; see [66].

Again, by the law of large numbers,

$$\mathbf{D}(\hat{\theta}_{\text{GN}}) \xrightarrow{p} \mathbf{D}(\theta_0), \quad S(\hat{\theta}_{\text{GN}}) \xrightarrow{p} S^{(\theta_0)},$$

where

$$S(\hat{\theta}_{\text{GN}}) = \frac{1}{K-1} \sum_{k=1}^K [\eta^{(k)} - \bar{\eta}][\eta^{(k)} - \bar{\eta}]^\top,$$

with $\eta^{(k)} = \varphi(z)$, $z \sim \mathcal{M}(\hat{\theta}_{\text{GN}})$ simulated with noise seed ω_k , and $\bar{\eta} = \frac{1}{K} \sum_{k=1}^K \eta^{(k)}$. Therefore, by the continuous mapping theorem,

$$\mathbf{G} = \mathbf{D}(\hat{\theta}_{\text{GN}})^{-1} S(\hat{\theta}_{\text{GN}}) \mathbf{D}(\hat{\theta}_{\text{GN}}) \xrightarrow{p} \Sigma_{\text{GN}}.$$

Hence, by Slutsky's theorem,

$$\sqrt{N} \mathbf{G}^{1/2} (\hat{\theta}_{\text{GN}} - \theta_0) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \mathbf{I}_d),$$

so $(\hat{\theta}_{\text{GN}} - \theta_0)^\top \mathbf{G} (\hat{\theta}_{\text{GN}} - \theta_0)$ is asymptotically χ_d^2 . Therefore,

$$\mathbb{P}(\theta_0 \in \mathcal{E}) \rightarrow 1 - \beta \quad \text{as } N \rightarrow \infty, K \rightarrow \infty, K_{\text{avg}} \rightarrow \infty,$$

where the confidence ellipsoid is

$$\mathcal{E} = \{\theta: (\hat{\theta}_{\text{GN}} - \theta)^\top \mathbf{G} (\hat{\theta}_{\text{GN}} - \theta) \leq \chi_d^2(\beta)\}.$$

8.5 Simulation Study

We evaluate the proposed fine-tuning scheme on two nonlinear systems: (i) the Van der Pol (VDP) oscillator and (ii) cascaded water tanks. In both cases, the model structure is known but the true parameters may lie outside the calibration set Θ_p used to pretrain the TS estimator.

Common settings For all simulations we set $K = 150$ in (8.3.8). The whitened Jacobian $\mathbf{D}$ is obtained via finite differences. To compute $\hat{f}(\theta)$ in (8.3.11) and the Jacobians, we use common random numbers with the seed set $\mathcal{S}_{\text{avg}} = \{8000, 8001, 8002\}$. Both examples are simulated in discrete time using forward Euler with a fixed step size Δt (VDP: 0.05 s; tanks: 0.5 s). The Gauss–Newton regularization parameter in (8.3.12) is $\gamma = 10^{-3}$ (VDP) and $\gamma = 10^{-4}$ (tanks). The orthogonality regularizer λ_{orth} on the first head layer is 0 for VDP (single output) and 5×10^{-4} for the tanks. We set $\alpha \in [0.05, 0.10]$; this range was selected from a preliminary sensitivity analysis to balance false positives on in-distribution data with reliable OOD detection. Unless otherwise noted, we use $\alpha = 0.10$ in all experiments.

Feature compression and stage-2 network For VDP we use AR(5) coefficients fitted to y ; for the tanks we use ARX(64, 64) coefficients fitted to (y, u) . The stage-2 regressor ψ_{pre} is a fully connected network with a shared trunk and one head per parameter, and the following settings:

- **VDP:** trunk (256, 128) with ReLU+Dropout(0.1); one head (depth 2, hidden 128, Tanh+Dropout(0.1)). Adam with lr 10^{-3} , batch size 64, 200 epochs. Offline sample size $M = 10000$; fine-tune cloud $M_{\text{FT}} = 400$.
- **Tanks:** trunk (256, 256) with LayerNorm+LeakyReLU+Dropout(0.15); four heads (depth 3, hidden 64, LayerNorm+Tanh+Dropout(0.15)). Adam with lr 10^{-3} , batch size 64, 250 epochs. Offline sample size $M = 10000$; fine-tune cloud $M_{\text{FT}} = 1000$.

In the fine-tuned TS estimator (TS-fine), we *freeze the trunk* and *retrain only the head layers* on the synthetic cloud around the GN update; for the tanks we also use a smaller head (hidden 64, depth 1, dropout 0.20) with early stopping. We fix all hyperparameters (K , M_{FT} , γ , λ_{orth} , architectures, and training choices) *a priori* across all experiments.

Baselines We compare TS-fine against two classical methods: the prediction error method (PEM) and the dual extended Kalman filter (dual-EKF). Both methods process the observed trajectory and require an initial estimate of θ_0 to begin their updates; when we refer to “initialization” below, we mean this starting value.

PEM iteratively refines the parameter estimate by minimizing the squared error between the noise-free simulator output and the observed trajectory. At each iteration, the Jacobian of the simulated trajectory with respect to θ is computed via forward finite differences, and the parameter update is obtained from a regularized Gauss–Newton step with a backtracking line search. For the tanks system, the iterates are additionally projected onto a feasible region to prevent the simulator from encountering unphysical states.

The dual-EKF treats the unknown parameters as additional state variables, appending them to the dynamical state to form a joint state vector. The dynamical states are initialized from the first observation, and the parameter components are set to the chosen starting value. The initial error covariance is diagonal, with larger entries for the parameter components than for the dynamical states, reflecting the greater uncertainty in the parameters at the start. The filter then processes the observations sequentially: at each time step it predicts the joint state forward using the discretized dynamics with the parameters modeled as constant, and corrects both the states and parameters based on the prediction error. The process noise on the parameter components is set much smaller than on the dynamical states, encoding the assumption that the parameters change slowly relative to the system dynamics.

Both baselines are evaluated under three choices of the initial parameter estimate. *Good Init (GI)* initializes at a small Gaussian perturbation of the true value; this serves as an oracle benchmark since it requires knowledge of θ_0 and is unavailable in practice. *TS-Init (TI)* initializes at the pretrained TS output $T_{\text{TS}}^{\text{pre}}(z_0)$, which is available without any knowledge of θ_0 . *Bad Init (BI)* initializes at the corner of the global parameter space farthest from θ_0 , representing the most adversarial feasible starting point.

8.5.1 Van der Pol oscillator

The VDP dynamics are simulated using the discrete-time state-space model

$$\begin{aligned} x_{1,k+1} &= x_{1,k} + \Delta t x_{2,k}, \\ x_{2,k+1} &= x_{2,k} + \Delta t (\mu(1 - x_{1,k}^2)x_{2,k} - x_{1,k}) + w_k, \\ y_k &= x_{1,k}, \end{aligned} \tag{8.5.31}$$

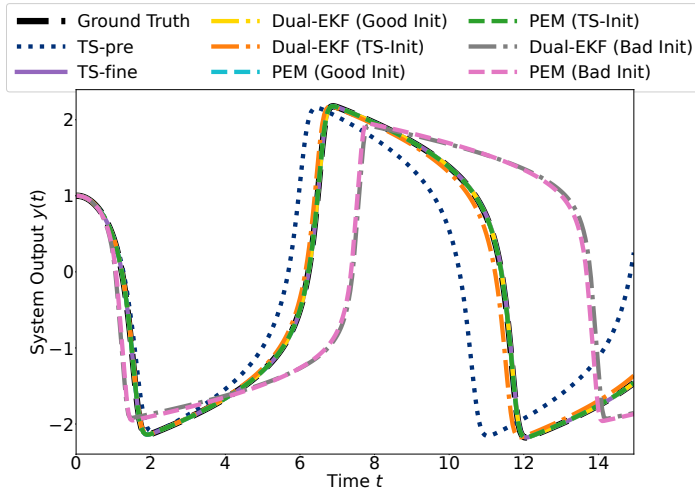
where μ is the unknown damping coefficient, $\Delta t = 0.05$ s is the sampling period, and $w_k \sim \mathcal{N}(0, 0.02)$ is process noise. We collect $N = 300$ length observations for each $\mu_i \sim \text{Unif}[0, 2.5]$ to form the offline training set $\mathcal{D}_{\text{tr}}$.

OOD tests (VDP) We evaluate at $\mu = 3.0$ (OOD 1, mild OOD) and $\mu = 5.0$ (OOD 2), both outside the pretraining range. Over $n_{\text{mc}} = 100$ Monte Carlo runs we report MSE and average computation time in Table 8.5.1.

Table 8.5.1: VDP results: MSE and avg. computation time (ms) over 100 MC runs.

Method	OOD 1		OOD 2	
	MSE	ms	MSE	ms
TS-pre	0.320 400	1.885	7.783 00	1.893
TS-fine	0.000 550	13 087.631	0.046 08	12 779.925
Dual-EKF (GI)	0.003 269	8.290	0.397 80	8.177
Dual-EKF (TI)	0.006 730	8.402	1.498 00	8.327
PEM (GI)	0.001 154	138.950	0.075 30	150.616
PEM (TI)	0.001 154	115.815	0.632 50	81.269
Dual-EKF (BI)	15.780 000	8.369	5.017 00	8.273
PEM (BI)	14.670 000	83.473	4.531 00	108.913

Average trajectories and residuals are shown in Figures 8.5.4–8.5.7, and box plots of the squared-error distribution appear in Figure 8.5.8. The trajectory and residual plots show the noise-free simulator output evaluated at the mean parameter estimate across 100 MC runs, i.e., the simulator is run once at $\bar{\theta} = \frac{1}{n_{mc}} \sum_{i=1}^{n_{mc}} \hat{\theta}^{(i)}$ for each method, rather than averaging the individual trajectories.

**Figure 8.5.4:** VDP (OOD 1): ground-truth vs. mean predicted trajectories (noise-free) over 100 MC runs.

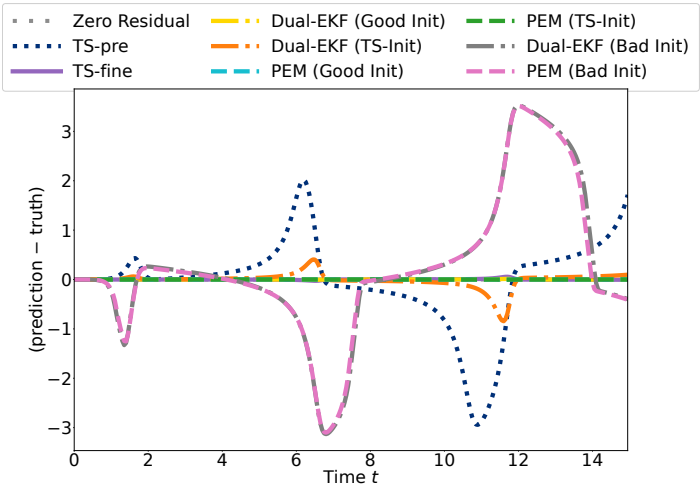


Figure 8.5.5: VDP (OOD 1): residuals between estimated and true output trajectories over 100 MC runs.

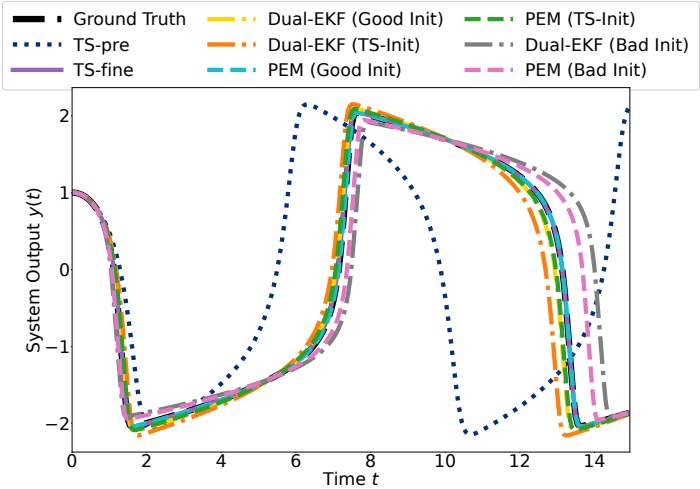


Figure 8.5.6: VDP (OOD 2): ground-truth vs. mean predicted trajectories (noise-free) over 100 MC runs.

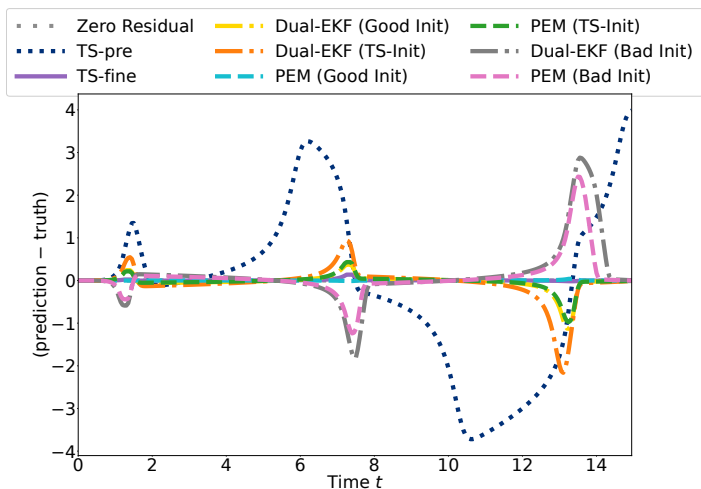


Figure 8.5.7: VDP (OOD 2): residuals between estimated and true output trajectories over 100 MC runs.

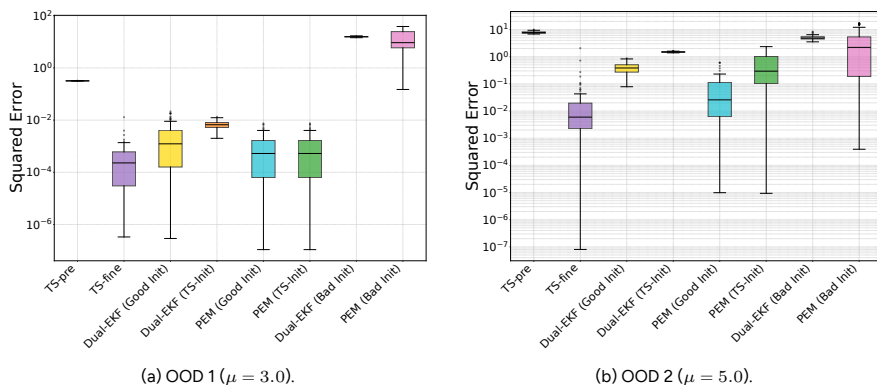


Figure 8.5.8: VDP: box plots of squared-error distribution over 100 MC runs.

TS-fine substantially improves upon TS-pre in both OOD settings, confirming that the fine-tuning step corrects the pretrained estimator’s extrapolation error. PEM and dual-EKF achieve good accuracy under favorable initialization but degrade sharply otherwise. TS-fine requires no user-supplied initialization and performs reliably across both scenarios. The box plots reinforce this: the squared-error distribution of TS-fine is concentrated at low values, whereas the poorly initialized baselines exhibit heavy tails. Regarding computation time, TS-pre requires only a forward pass (under 2 ms), and this is all that is needed when the OOD detector determines the observation is in-distribution; the fine-tuning overhead of TS-fine is incurred only upon OOD detection.

8.5.2 Cascaded water tanks

The cascaded water tanks are simulated using the discrete-time model

$$\begin{aligned} h_{1,k+1} &= h_{1,k} + \Delta t(-k_1\sqrt{|h_{1,k}|} + k_4 u_k) + w_k^{(1)}, \\ h_{2,k+1} &= h_{2,k} + \Delta t(k_2\sqrt{|h_{1,k}|} - k_3\sqrt{|h_{2,k}|}) + w_k^{(2)}, \\ y_k &= h_{2,k} + v_k, \end{aligned} \quad (8.5.32)$$

with parameter vector $\theta = (k_1, k_2, k_3, k_4)^\top$, sampling period $\Delta t = 0.5$ s, process noises $w_k^{(1)}, w_k^{(2)} \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.01)$, and measurement noise $v_k \sim \mathcal{N}(0, 0.02)$. The parameters k_1, k_2, k_3, k_4 are lumped physical coefficients that enter the mass-balance and flow relations; collectively, they represent the tank cross-sectional areas together with the effluent areas and associated conversion constants in the input-flow relationship [91].

Offline pretraining set We draw samples $\theta^{(i)} \sim [0.4, 0.8] \times [0.4, 0.8] \times [0.4, 0.8] \times [0.8, 1.2]$ uniformly ($M = 10000$) and for each $\theta^{(i)}$ collect $N = 400$ length observations. The inputs are PRBS signals [91]: the level of binary signal is drawn i.i.d. from $[0, 3]$ and held for 30 samples.

OOD tests (Tanks) We consider $\theta = (1.2, 1.2, 0.9, 1.0)^\top$ (OOD 1) and $\theta = (1.3, 1.3, 0.6, 0.7)^\top$ (OOD 2), both outside the pretraining set. For evaluation we fix the same block-uniform input across runs, simulate with the noise model above, and average over $n_{\text{mc}} = 100$ MC runs. MSE and computation time are reported in Table 8.5.2, trajectories and residuals in Figures 8.5.9–8.5.12, and box plots in Figure 8.5.13. The trajectory and residual plots are generated by simulating the noise-free system at each method’s mean parameter estimate (averaged over 100 MC runs) under a common block-uniform input signal. This input realization is generated with a different random seed than the one used during the Monte Carlo estimation runs, but is shared across all methods and the ground truth within each plot.

Table 8.5.2: Tanks results: MSE and avg. computation time (ms) over 100 MC runs.

Method	OOD 1		OOD 2	
	MSE	ms	MSE	ms
TS-pre	0.118 900	3.995	0.192 10	3.981
TS-fine	0.070 510	50 854.724	0.163 10	51 366.442
Dual-EKF (GI)	6.112 000	18.851	2.854 00	18.787
Dual-EKF (TI)	8.345 000	19.107	6.015 00	19.081
PEM (GI)	0.073 670	8564.444	0.061 05	8635.810
PEM (TI)	0.186 000	8783.368	0.328 10	8410.821
Dual-EKF (BI)	8.978 000	19.034	7.445 00	18.915
PEM (BI)	0.392 400	7086.185	0.180 00	6761.688

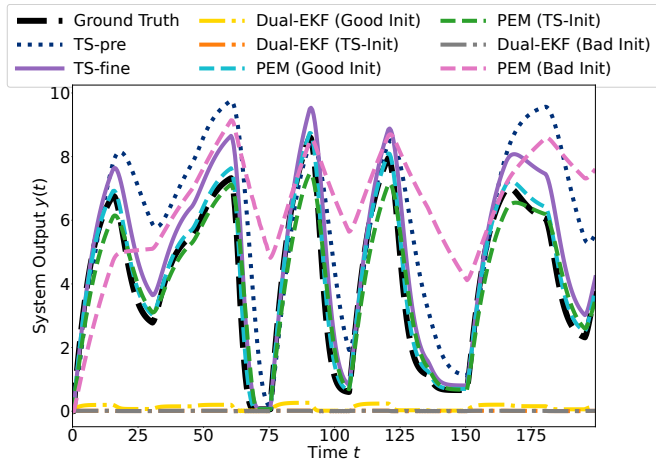


Figure 8.5.9: Tanks (OOD 1): ground-truth vs. mean predicted trajectories (noise-free) over 100 MC runs.

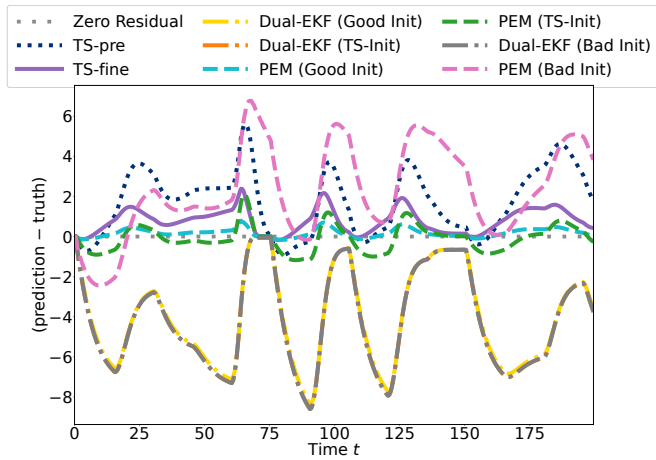


Figure 8.5.10: Tanks (OOD 1): residuals between estimated and true output trajectories over 100 MC runs.

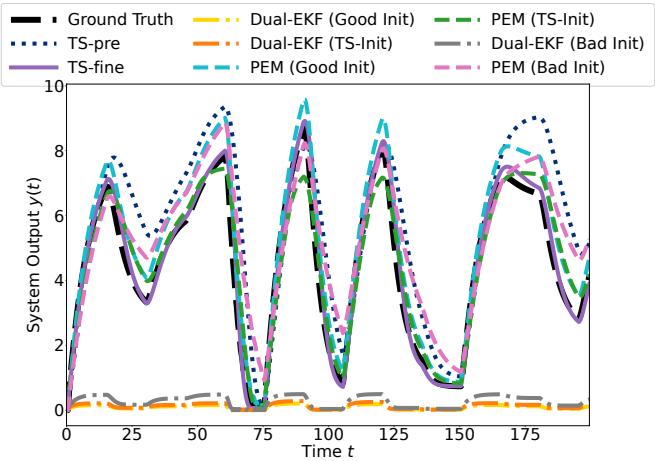


Figure 8.5.11: Tanks (OOD 2): ground-truth vs. mean predicted trajectories (noise-free) over 100 MC runs.

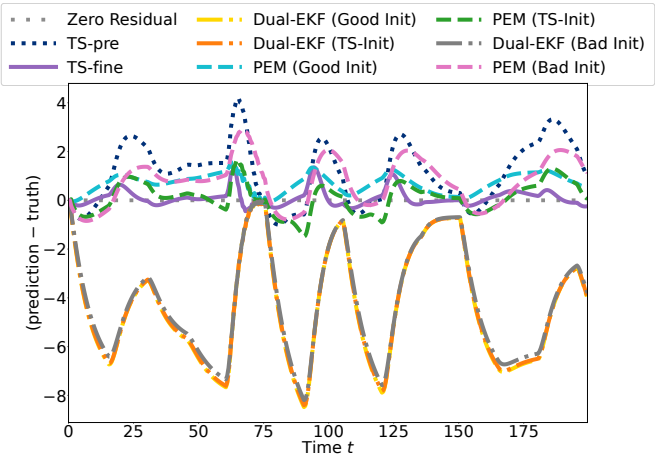


Figure 8.5.12: Tanks (OOD 2): residuals between estimated and true output trajectories over 100 MC runs.

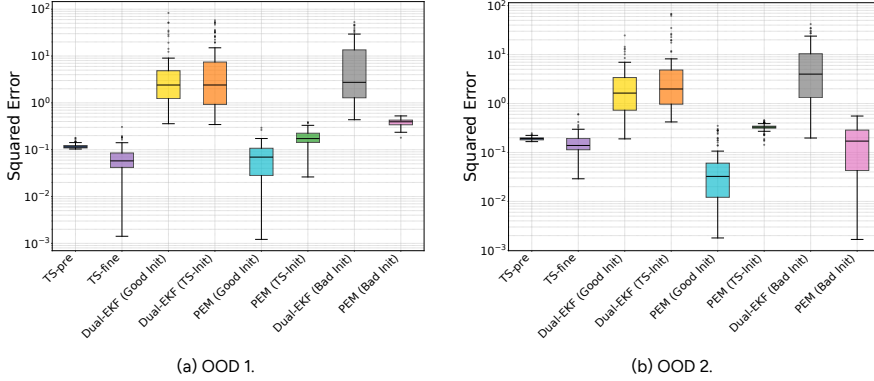


Figure 8.5.13: Cascaded water tanks: box plots of squared-error distribution over 100 MC runs for OOD 1 and OOD 2.

The trends observed for VDP carry over to this higher-dimensional setting, with one notable difference: the dual-EKF struggles with the four-dimensional parameter vector, performing poorly across all initializations. PEM remains sensitive to its starting point: accurate under Good Init but unreliable otherwise. TS-fine again improves upon TS-pre and delivers competitive accuracy without requiring any knowledge of θ_0 . The box plots confirm that TS-fine maintains a tighter error distribution and fewer outliers than the baselines. The computational profile is similar to VDP: TS-pre is the cheapest method, and the fine-tuning cost of TS-fine is only incurred when the OOD detector fires.

Remark 8.3. The accuracy of TS-pre and TS-fine depends on both stages. Under high process/measurement noise (low SNR), the Stage 1 compressor should yield statistics from which the parameters can be inferred irrespective of the noise level, while Stage 2 (MLP) can become unreliable during training. Our aim here is to fine-tune TS, not to optimize the underlying first/second-stage architectures for noise robustness. Consequently, in low SNR regimes, the standard baselines may outperform our current setup. This can be addressed by (a) designing a noise-robust Stage 1 compressor and (b) strengthening Stage 2 via adversarial training.

Remark 8.4. Fine-tuning adds a computation overhead from three sources: (i) computing $\mathbf{D}$ and solving the $d \times d$ system $\mathbf{G} = \mathbf{D}^\top \mathbf{D} + \lambda \mathbf{I}_d$, which has $\mathcal{O}(d^3)$ time complexity, (ii) generating a synthetic cloud of M_{FT} parameter samples and their features, and (iii) short, head-only retraining over this cloud. This computation overhead can be overcome via *unsupervised* fine-tuning that updates only the final-layer weights by directly shrinking the discrepancy between simulated and observed trajectories/features, thereby avoiding large synthetic clouds.

8.6 Conclusion

In this chapter, we have proposed a transfer-learning approach to fine-tune a pretrained, simulation-driven (TS) estimator for OOD scenarios. We first apply a statistical detection test on the compressed features to flag OOD; if the test detects an OOD, we modify only the second stage, a neural network, by freezing the shared trunk and retraining the last, per-parameter layers using a newly synthesized local dataset. Numerical examples demonstrate that the fine-tuned TS delivers improved performance over the pretrained TS and fares well against initialization-dependent baselines such as dual-EKF and PEM.

9 Unsupervised Fine-Tuning of a Simulation-Driven Estimator

Chapter 8 addressed the OOD problem through supervised fine-tuning: upon detecting an OOD, a Gauss–Newton refinement produces a point estimate around which a targeted synthetic dataset is generated from the simulator, and the head layers of the second-stage neural network are retrained on this dataset. That approach requires only the ability to run the simulator forward, but involves several intermediate steps: computing the whitened Jacobian, constructing a confidence ellipsoid, generating a synthetic cloud, and retraining the head layers.

This chapter presents an alternative approach: *unsupervised fine-tuning* via trajectory matching. Rather than constructing a labeled synthetic dataset, the idea is to directly minimize the discrepancy between the simulator’s output trajectory and the observed trajectory from the true system. This is achieved by backpropagating through a differentiable simulator to update the second-stage neural network weights, bypassing the synthetic data generation step entirely. The resulting pipeline directly optimizes the quantity of ultimate interest, which is a trajectory fit, rather than an intermediate surrogate loss in feature space.

The key distinction from Chapter 8 lies in what is assumed about the simulator. Chapter 8 requires only forward simulation, whereas the present chapter requires the simulator to be *differentiable* with respect to its parameters, so that gradients of the trajectory can be computed via automatic differentiation. The two approaches are complementary rather than competing: the supervised method applies to any black-box simulator, while the unsupervised method offers a simpler pipeline when a differentiable simulator is available. Neither consistently dominates the other in accuracy or computation time.

The contributions of this chapter are threefold: we employ the same feature-space OOD detector from Chapter 8 to initiate the fine-tuning procedure; we design an unsupervised fine-tuning objective that updates the second-stage network weights by minimizing the squared error between simulated and observed output trajectories; and we numerically validate the approach against the same baselines considered in Chapter 8.

9.1 Problem Statement

Consider a high-fidelity simulator $\mathcal{M}(\theta)$ with parameters $\theta \in \Theta \subset \mathbb{R}^d$. We consider a single-input-single-output (SISO) setting: for a fixed input sequence $\mathbf{u} = (u_1, \dots, u_N)^\top \in \mathbb{R}^N$, the simulator produces an output sequence $\mathbf{y} = (y_1, \dots, y_N)^\top \in \mathbb{R}^N$ that may depend on internal randomness ω , i.e., $\mathbf{y} = \mathbf{y}(\theta; \omega)$. From the physical system, represented by the simulator for an unknown θ_0 , we observe a length- N trajectory

$$\mathbf{z}_0 = (u_1, y_{0,1}, \dots, u_N, y_{0,N}) \in \mathbb{R}^{2N},$$

and our goal is to estimate θ_0 from $\mathbf{z}_0$.

Using prior knowledge of plausible regions for θ_0 , we specify a calibration set $\Theta_p \subseteq \Theta$ and draw i.i.d. parameter samples $\{\theta^{(i)}\}_{i=1}^M \subset \Theta_p$. For each draw, we run the simulator with the fixed input $\mathbf{u}$ (but possibly a different noise seed ω_i) to obtain

$$\mathbf{z}^{(i)} = (u_1, y_{i,1}, \dots, u_N, y_{i,N}), \quad i = 1, \dots, M,$$

yielding a labeled synthetic set $\mathcal{D} = \{(\mathbf{z}^{(i)}, \theta^{(i)})\}_{i=1}^M$. Additionally, we make the following assumption about the simulator, which will be used later.

Assumption 9.1 (Differentiable Simulator). The simulator $\mathcal{M}(\theta)$ is differentiable with respect to θ .

In simulation-driven estimation procedures, we train a function $T_{\text{TS}}^{\text{pre}}: \mathbb{R}^{2N} \rightarrow \mathbb{R}^d$ from a model class $\mathcal{F}$ by empirical risk minimization:

$$T_{\text{TS}}^{\text{pre}}(\cdot) \in \arg \min_{F \in \mathcal{F}} \frac{1}{M} \sum_{i=1}^M \ell(F(\mathbf{z}^{(i)}), \theta^{(i)}), \quad (9.1.1)$$

where $\ell: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$ is a loss function (e.g., the squared or Huber losses [87]). In the implementation phase, the estimate for the observed trajectory is

$$\hat{\theta}_0 = T_{\text{TS}}^{\text{pre}}(\mathbf{z}_0).$$

When θ_0 lies well within Θ_p and $\mathcal{D}$ sufficiently covers its neighborhood, (9.1.1) typically yields accurate estimates. In practice, however, since Θ_p is user-specified and may omit plausible regions, an OOD situation may occur if $\theta_0 \notin$

Θ_p , or if there are not enough samples $\theta^{(i)}$ in the neighborhood of θ_0 . Hence, the pretrained TS estimator might yield poor estimates, motivating the search for mechanisms that refine $T_{\text{TS}}^{\text{pre}}(\cdot)$ at the implementation phase. In this chapter, we focus on a special class of simulation-driven estimators $T_{\text{TS}}^{\text{pre}}(\cdot)$, known as the Two-Stage (TS) estimator.

9.1.1 Two-Stage (TS) Estimator

The TS estimator [33, 34] decomposes (9.1.1) into two stages: a fixed *compression* of z into low-dimensional features, followed by a supervised machine learning model that maps these features to the parameters. We shall now formulate the stages of the TS estimator in more detail.

9.1.1.1 First Stage – Data Compression

In the first stage, we compress the input-output data $z^{(i)}$ generated by the simulator $\mathcal{M}(\theta^{(i)})$ using the function

$$\varphi: \mathbb{R}^{2N} \rightarrow \mathbb{R}^n, \quad n \ll 2N, \quad (9.1.2)$$

e.g., the estimated coefficients of an AR/ARX/ARMAX model fitted to the data. Let $\eta^{(i)} = \varphi(z^{(i)})$ and generate the synthetic dataset

$$\mathcal{D}_{\text{tr}} = \{(\eta^{(i)}, \theta^{(i)})\}_{i=1}^M = \{(\varphi(z^{(i)}), \theta^{(i)})\}_{i=1}^M \subset \mathbb{R}^n \times \mathbb{R}^d. \quad (9.1.3)$$

9.1.1.2 Second Stage – Supervised Learning via ResNet

Once we obtain the synthetic dataset $\mathcal{D}_{\text{tr}}$, we fit a neural network that takes $\varphi(z^{(i)})$ as inputs and $\theta^{(i)}$ as the corresponding targets. In this chapter, the second stage is a function $\psi_{\text{pre}}: \mathbb{R}^n \rightarrow \mathbb{R}^d$ implemented as a residual neural network (ResNet) with d output neurons. The ResNet consists of three components: an input projection, R residual blocks, and a linear output layer.

Input projection

$$s^{(0)} = \sigma_0(W_0\eta + b_0), \quad W_0 \in \mathbb{R}^{p \times n}, \quad b_0 \in \mathbb{R}^p. \quad (9.1.4)$$

Residual blocks For each residual block $r = 1, \dots, R$,

$$u^{(r)} = \rho_r(W_1^{(r)}s^{(r-1)} + b_1^{(r)}), \quad \tilde{u}^{(r)} = W_2^{(r)}u^{(r)} + b_2^{(r)}, \quad (9.1.5)$$

$$s^{(r)} = \sigma_r(s^{(r-1)} + \tilde{u}^{(r)}),$$

where $\rho_r(\cdot)$ and $\sigma_r(\cdot)$ are element-wise activation functions (e.g., tanh, ReLU), and $s^{(r)} \in \mathbb{R}^p$ are hidden vectors.

Linear output layer The d -dimensional output is

$$\hat{\theta}(\boldsymbol{\eta}) = \psi_{\text{pre}}(\boldsymbol{\eta}; \boldsymbol{\Omega}) = \mathbf{W}_{\text{out}} \mathbf{s}^{(R)} + \mathbf{b}_{\text{out}} \in \mathbb{R}^d, \quad (9.1.6)$$

with $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d \times p}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^d$. Here, $\boldsymbol{\Omega}$ denotes all trainable parameters of the ResNet:

$$\boldsymbol{\Omega} = \left\{ \mathbf{W}_0, \mathbf{b}_0, (\mathbf{W}_1^{(r)}, \mathbf{b}_1^{(r)}, \mathbf{W}_2^{(r)}, \mathbf{b}_2^{(r)})_{r=1}^R, \mathbf{W}_{\text{out}}, \mathbf{b}_{\text{out}} \right\}. \quad (9.1.7)$$

Remark 9.1. The choice of neural network architecture for the second stage is problem-dependent and not prescribed by the TS framework. In this chapter, we adopt a standard residual neural network (ResNet), which simplifies the second-stage architecture used in Chapter 8.

These trainable parameters are optimized via backpropagation by solving:

$$\hat{\boldsymbol{\Omega}} \in \arg \min_{\boldsymbol{\Omega}} \frac{1}{M} \sum_{i=1}^M \|\psi_{\text{pre}}(\boldsymbol{\eta}^{(i)}; \boldsymbol{\Omega}) - \boldsymbol{\theta}^{(i)}\|_2^2. \quad (9.1.8)$$

The final TS estimator is

$$T_{\text{TS}}^{\text{pre}}(\mathbf{z}) = \psi_{\text{pre}}(\varphi(\mathbf{z}); \hat{\boldsymbol{\Omega}}) \in \mathbb{R}^d. \quad (9.1.9)$$

Given a new trajectory $\mathbf{z}_0$ generated by $\mathcal{M}(\boldsymbol{\theta}_0)$, the point estimate is

$$\hat{\boldsymbol{\theta}}_0 = T_{\text{TS}}^{\text{pre}}(\mathbf{z}_0) = \psi_{\text{pre}}(\varphi(\mathbf{z}_0); \hat{\boldsymbol{\Omega}}). \quad (9.1.10)$$

If $\boldsymbol{\theta}_0 \notin \Theta_p$, then $(\mathbf{z}_0, \boldsymbol{\theta}_0)$ is OOD w.r.t. $\mathcal{D}_{\text{tr}}$, and (9.1.10) may degrade; in such cases we can adjust $\psi_{\text{pre}}(\cdot; \boldsymbol{\Omega})$ via fine-tuning by updating all parameters $\boldsymbol{\Omega}$ through backpropagation, as discussed next.

9.2 Proposed Approach

To improve a pretrained TS estimator when it encounters OOD data, we employ a unsupervised learning procedure. We keep the first stage φ fixed and, during the implementation phase, fine-tune the second stage $\psi_{\text{pre}}(\cdot; \boldsymbol{\Omega})$ by updating all weights $\boldsymbol{\Omega}$ based on new observations from the system. Fine-tuning is only performed when necessary, i.e., when an OOD is detected via a statistical test based on Wald's test [88, 89, 90]. If an OOD is detected, the weights are updated by minimizing the simulation error, which is the squared difference between trajectories generated by the simulator and trajectories from the true system.

The unsupervised approach consists of two steps: an OOD detector (Step 1) and a trajectory-level weight update (Step 2).

9.2.1 Step 1: Feature-space OOD detector

The OOD detection procedure is identical to that described in Chapter 8, Section 8.3.1. We provide a brief summary here for completeness.

The null hypothesis, $\mathcal{H}_0$, is that the observed features $\boldsymbol{\eta}_{\text{obs}} = \varphi(\mathbf{z}_0)$ are statistically consistent with features generated by the simulator at the initial estimate $\hat{\boldsymbol{\theta}} = \psi_{\text{pre}}(\boldsymbol{\eta}_{\text{obs}}; \boldsymbol{\Omega})$. The procedure involves running K simulations at $\hat{\boldsymbol{\theta}}$ with different noise seeds to obtain features $\{\boldsymbol{\eta}^{(k)}\}_{k=1}^K$, computing the feature mean $\bar{\boldsymbol{\eta}}$ and regularized covariance $\mathbf{S}$ as in Eq. (8.3.8) of Chapter 8, and computing the whitened test statistics:

$$s_{\text{obs}} = \|\mathbf{S}^{-1/2}(\boldsymbol{\eta}_{\text{obs}} - \bar{\boldsymbol{\eta}})\|_2^2, \quad s_k = \|\mathbf{S}^{-1/2}(\boldsymbol{\eta}^{(k)} - \bar{\boldsymbol{\eta}})\|_2^2. \quad (9.2.11)$$

The decision threshold $q_{1-\alpha}$ is the empirical $(1-\alpha)$ -quantile of $\{s_k\}_{k=1}^K$.

Decision rule: If $s_{\text{obs}} > q_{1-\alpha}$, we reject $\mathcal{H}_0$, conclude that an OOD scenario is likely, and proceed to Step 2. Otherwise, the pretrained estimator is retained without modification.

9.2.2 Step 2: Trajectory-level unsupervised tuning

If an OOD is detected, the second-stage network is fine-tuned using an unsupervised approach that does not require the ground-truth parameters. Starting from the pretrained estimate $\hat{\boldsymbol{\theta}} = \psi_{\text{pre}}(\varphi(\mathbf{z}_0); \boldsymbol{\Omega})$, the simulator $\mathcal{M}$ is run K' times with different noise seeds to produce output trajectories $\tilde{\mathbf{y}}_k = (\tilde{y}_{1,k}, \dots, \tilde{y}_{N,k})^\top$, $k = 1, \dots, K'$. The fine-tuning objective minimizes the simulation error between these trajectories and the observed output $\mathbf{y} = (y_1, \dots, y_N)^\top$:

$$\ell_{\text{sim}}(\boldsymbol{\Omega}) = \frac{1}{K'N} \sum_{k=1}^{K'} \sum_{t=1}^N (\tilde{y}_{t,k} - y_t)^2. \quad (9.2.12)$$

This loss is a function of $\boldsymbol{\Omega}$ through the chain $\boldsymbol{\Omega} \rightarrow \hat{\boldsymbol{\theta}} \rightarrow \tilde{\mathbf{y}}_k$. The gradient with respect to $\boldsymbol{\Omega}$ is computed via the chain rule:

$$\nabla_{\boldsymbol{\Omega}} \ell_{\text{sim}} = \frac{2}{K'N} \sum_{k=1}^{K'} \sum_{t=1}^N (\tilde{y}_{t,k} - y_t) \frac{\partial \tilde{\mathbf{y}}_k}{\partial \hat{\boldsymbol{\theta}}} \frac{\partial \psi_{\text{pre}}(\varphi(\mathbf{z}_0); \boldsymbol{\Omega})}{\partial \boldsymbol{\Omega}}, \quad (9.2.13)$$

which requires the gradient of the simulator output with respect to its parameters (Assumption 9.1) and the gradient of the network output with respect to its weights (standard backpropagation). The weights are then updated iteratively, e.g., $\boldsymbol{\Omega}_{i+1} \leftarrow \boldsymbol{\Omega}_i - \tau \nabla_{\boldsymbol{\Omega}_i} \ell_{\text{sim}}$, where $\tau > 0$ is the step size. The procedure is visualized in Figure 9.2.1 and outlined in Algorithm 9.1. The output is the fine-tuned TS estimator:

$$T_{\text{TS}}^{\text{FT}}(\cdot) = \psi_{\text{pre}}(\varphi(\cdot); \boldsymbol{\Omega}_{\text{FT}}). \quad (9.2.14)$$

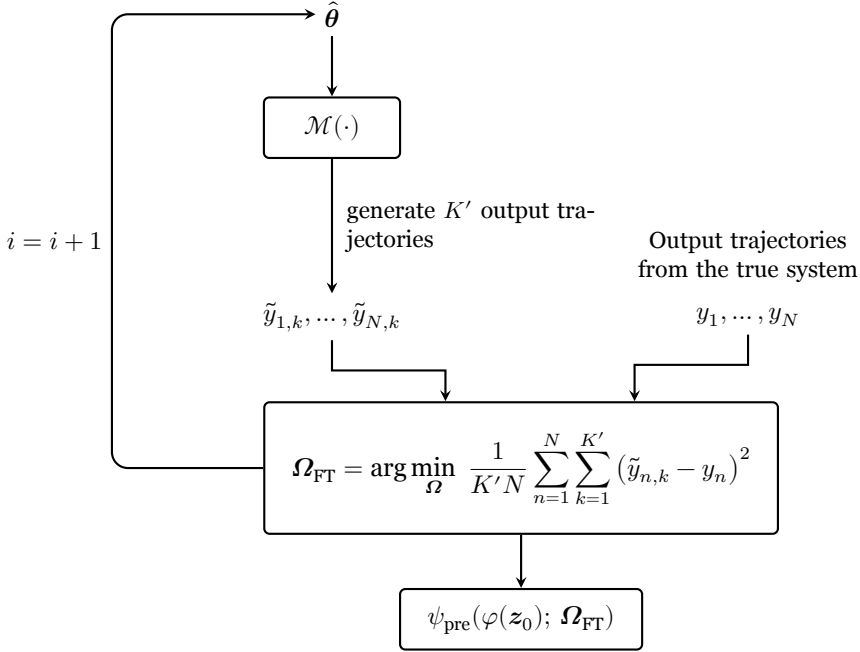


Figure 9.2.1: Flowchart of the unsupervised fine-tuning procedure. Starting from the current estimate $\hat{\theta} = \psi_{\text{pre}}(\varphi(z_0); \Omega)$, the simulator $\mathcal{M}(\cdot)$ generates K' output trajectories. The network weights Ω are updated by minimizing the simulation error ℓ_{sim} between simulated and observed trajectories.

Algorithm 9.1 Fine-Tuning TS Estimator via Simulation Error Minimization

Require: Pretrained network $\psi_{\text{pre}}(\cdot; \Omega)$, $\varphi(\cdot)$, learning rate η , number of epochs E , number of parallel simulations K' .

- 1: Collect trajectory z_0 from the true system.
- 2: Initialize weights Ω from pretrained $\hat{\Omega}$.
- 3: **for** $i = 1$ to E **do**
- 4: Compute parameter estimate: $\hat{\theta} = \psi_{\text{pre}}(\varphi(z_0); \Omega)$.
- 5: Generate K' simulated trajectories $\{\tilde{y}_k\}_{k=1}^{K'}$ from $\mathcal{M}(\hat{\theta})$.
- 6: Compute simulation error: $\ell_{\text{sim}} = \frac{1}{K'N} \sum_{k=1}^{K'} \sum_{t=1}^N (\tilde{y}_{t,k} - y_t)^2$.
- 7: Compute gradient $\nabla_{\Omega} \ell_{\text{sim}}$ and update: $\Omega \leftarrow \Omega - \tau \nabla_{\Omega} \ell_{\text{sim}}$.
- 8: **end for**

Ensure: Fine-tuned weights Ω_{FT} .

Remark 9.2. The fine-tuning objective ℓ_{sim} in (9.2.12) is designed for OOD deployments. In settings where the new data are in distribution, adding a small regularizer can help prevent unnecessary drift in the estimates. A systematic study of such safeguards is left for future work.

Remark 9.3. The number of simulated trajectories K' in Algorithm 9.1 can be fairly small. In practice, setting $K' = 1$ often suffices and keeps the compute overhead low. In our numerical study (Section 9.3), we fix $K' = 1$.

Remark 9.4. Our fine-tuning approach is related to indirect inference [50], in that both methods match simulated behavior to observed data. However, they differ in what is matched and what is optimized: indirect inference aligns *auxiliary* parameters and optimizes the *model parameters* explicitly, whereas we align *trajectories* directly and optimize the *network weights* Ω ; the simulator parameters are adjusted implicitly via the network's output.

9.3 Numerical Study

We evaluate the proposed fine-tuning scheme on the same two nonlinear systems as Chapter 8: the Van der Pol (VDP) oscillator and cascaded water tanks. We compare against TS-pre, PEM, and dual-EKF under the same three initializations described in Chapter 8. Both examples are simulated in discrete time using forward Euler with a fixed step size Δt ($= 0.05$ s for VDP and $= 0.5$ s for the tanks). We set $\alpha = 0.10$ for the OOD detector in all experiments. Since the focus of this chapter is on the unsupervised fine-tuning methodology rather than a head-to-head comparison with Chapter 8, we report results via trajectory plots and squared-error box plots; quantitative MSE tables and computation times follow the same trends as in Chapter 8 and are omitted for brevity.

9.3.1 Van der Pol oscillator

The VDP dynamics are governed by (8.5.31) with unknown $\mu > 0$ and process noise $w_k \sim \mathcal{N}(0, 0.02)$.

Offline pretraining set We collect $N = 300$ observations for each $\mu_i \sim \text{Unif}[0, 2.5]$ ($M = 6000$). The first stage uses AR(5) coefficients; the second stage is a ResNet with $R = 4$ blocks of hidden dimension $p = 32$ with LeakyReLU activations, trained with Adam ($\text{lr} = 10^{-3}$, 100 epochs).

OOD tests We evaluate at $\mu = 3.0$ (OOD 1) and $\mu = 5.0$ (OOD 2). Upon OOD detection, the differentiable simulator

$$\begin{aligned} x_{1,k+1} &= x_{1,k} + \Delta t x_{2,k}, \\ x_{2,k+1} &= x_{2,k} + \Delta t (\mu(1 - x_{1,k}^2)x_{2,k} - x_{1,k}), \\ y_k &= x_{1,k}, \end{aligned} \tag{9.3.15}$$

is used to compute $\nabla_{\Omega} \ell_{\text{sim}}$ via automatic differentiation.

Over $n_{\text{mc}} = 100$ Monte Carlo runs, trajectories and residuals are shown in Figures 9.3.2–9.3.3, and squared-error distributions in Figure 9.3.4.

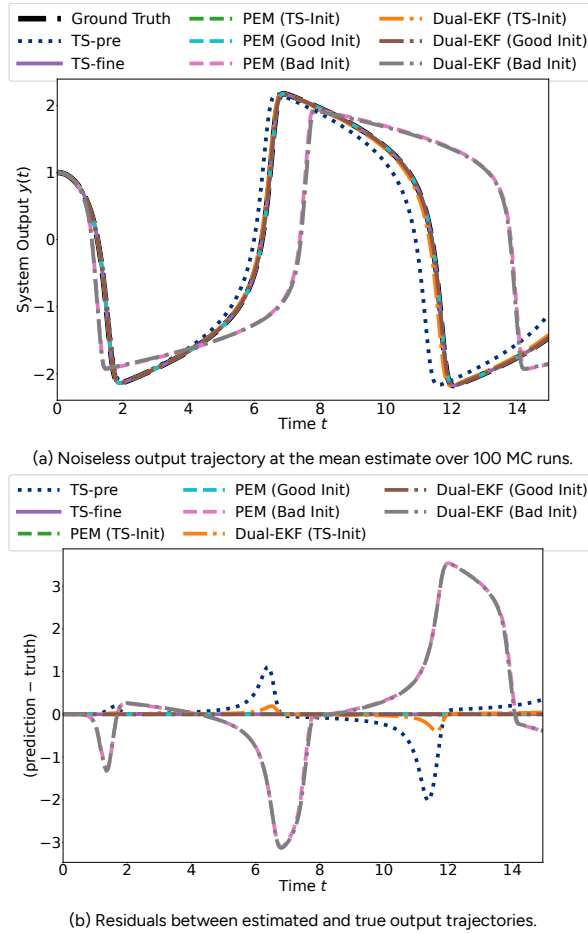


Figure 9.3.2: VDP (OOD 1, $\mu = 3.0$): trajectory comparison and residuals over 100 MC runs.

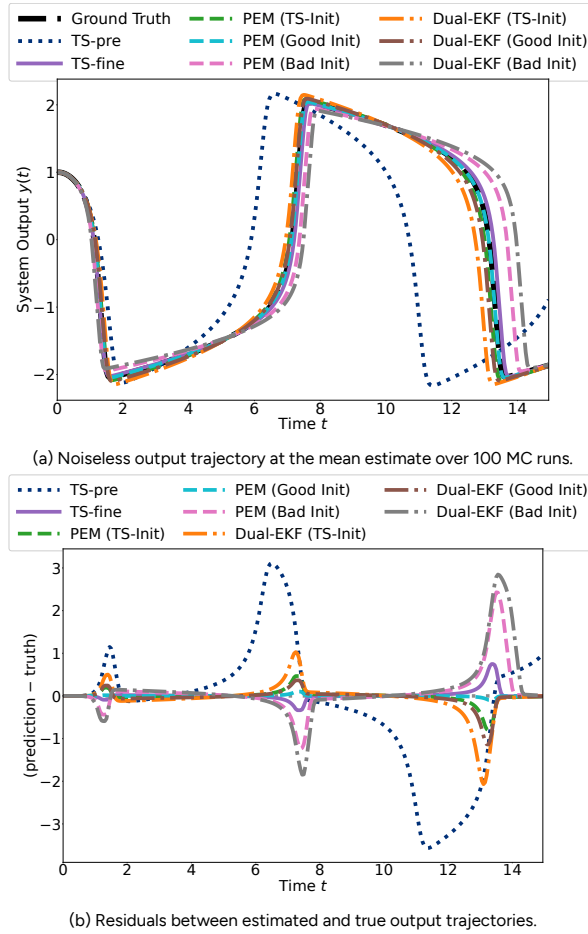


Figure 9.3.3: VDP (OOD 2, $\mu = 5.0$): trajectory comparison and residuals over 100 MC runs.

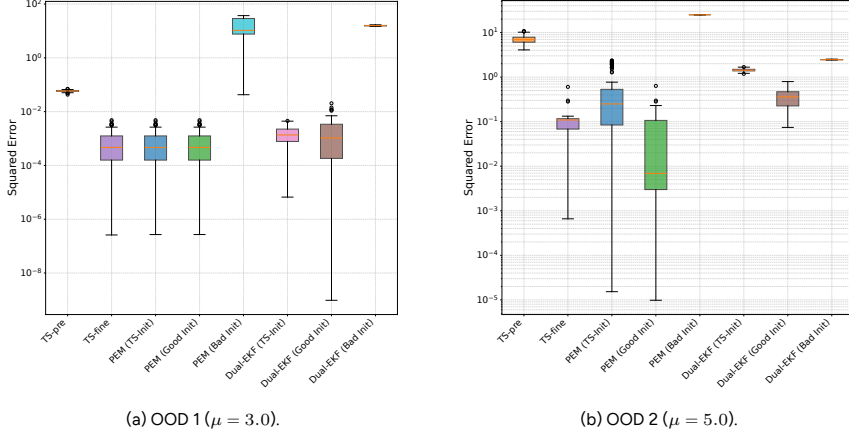


Figure 9.3.4: VDP: Box plots evaluating squared error distribution corresponding to different estimators over 100 MC runs. In the mild OOD setting ($\mu = 3.0$), the TS-pre estimate is already close to the true parameter, so both PEM (TS-Init) and PEM (Good Init) converge to the same local minimum, and TS-fine makes only a negligible refinement.

The trajectory and residual plots are generated by running the noise-free simulator at the mean parameter estimate over 100 MC runs for each method, rather than averaging individual trajectories.

TS-fine improves upon TS-pre in tracking the true output trajectory, with residuals substantially lower than those of the pretrained estimator and competitive with PEM and dual-EKF under favorable initialization. The box plots confirm that TS-fine achieves a lower squared error than TS-pre, while PEM and dual-EKF degrade under poor initialization.

9.3.2 Cascaded water tanks

The tanks dynamics are governed by (8.5.32) with $\theta = (k_1, k_2, k_3, k_4)^\top$.

Offline pretraining set We draw $\theta^{(i)} \sim [0.4, 0.8]^3 \times [0.8, 1.2]$ uniformly ($M = 10000$) and collect $N = 400$ observations per sample. The input is a block-uniform signal with amplitude from $[0, 3]$ held for 30 samples. The first stage uses ARX(32, 32) coefficients; the second stage is a ResNet with $R = 2$ blocks of hidden dimension $p = 64$ with tanh activations.

OOD tests We consider $\theta = (1.2, 1.2, 0.9, 1.0)^\top$ (OOD 1) and $\theta = (1.3, 1.3, 0.6, 0.7)^\top$ (OOD 2). Upon OOD detection, the differentiable simu-

lator

$$\begin{aligned}
 y_k &= h_{2,k}, \\
 h_{1,k+1} &= h_{1,k} + \Delta t (-k_1 \sqrt{|h_{1,k}| + \varepsilon} + k_4 u_k), \\
 h_{2,k+1} &= h_{2,k} + \Delta t (k_2 \sqrt{|h_{1,k}| + \varepsilon} - k_3 \sqrt{|h_{2,k}| + \varepsilon}),
 \end{aligned} \tag{9.3.16}$$

is used for gradient computation, where the $\sqrt{|\cdot| + \varepsilon}$ safeguard preserves differentiability. For evaluation, the same fixed block-uniform input is used across all MC runs and all methods. Trajectories and residuals are shown in Figures 9.3.5–9.3.6, and squared-error distributions in Figure 9.3.7.

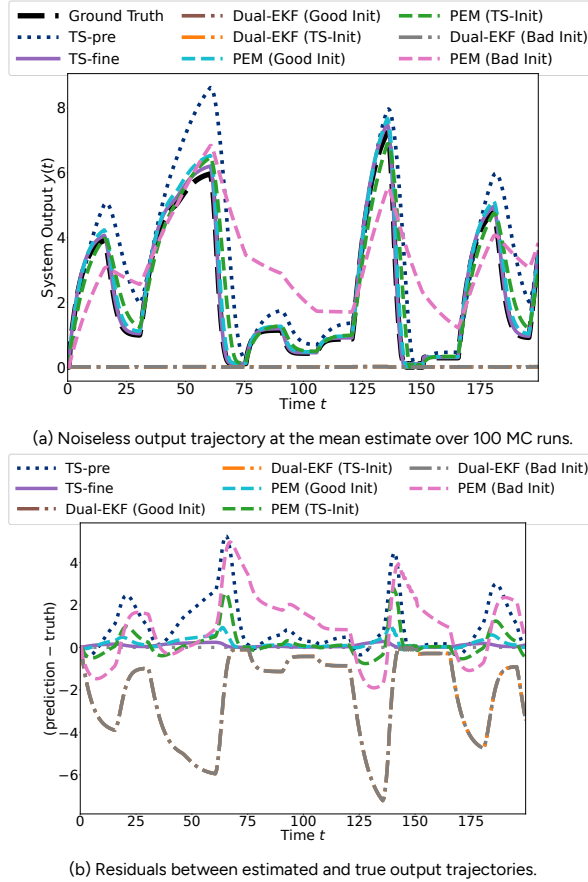


Figure 9.3.5: Cascaded water tanks (OOD 1): trajectory comparison and residuals over 100 MC runs.

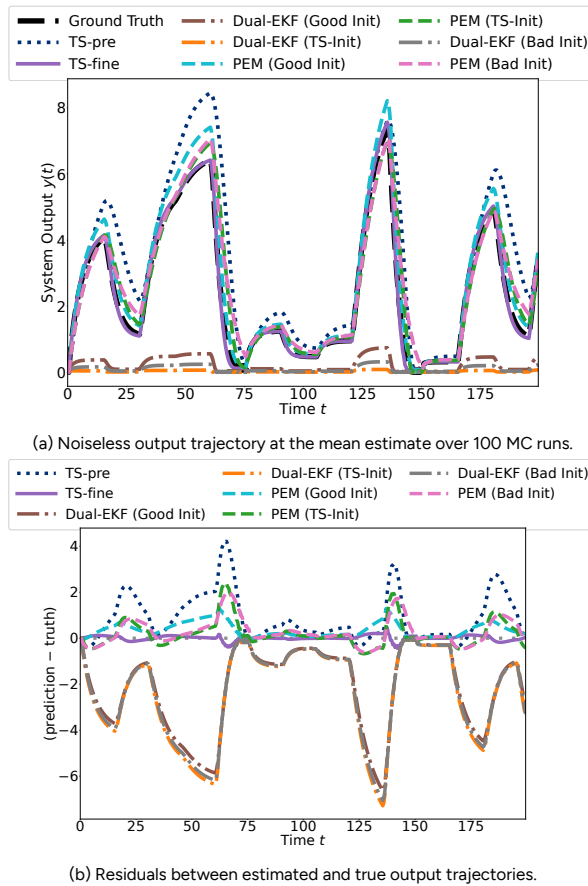


Figure 9.3.6: Cascaded water tanks (OOD 2): trajectory comparison and residuals over 100 MC runs.

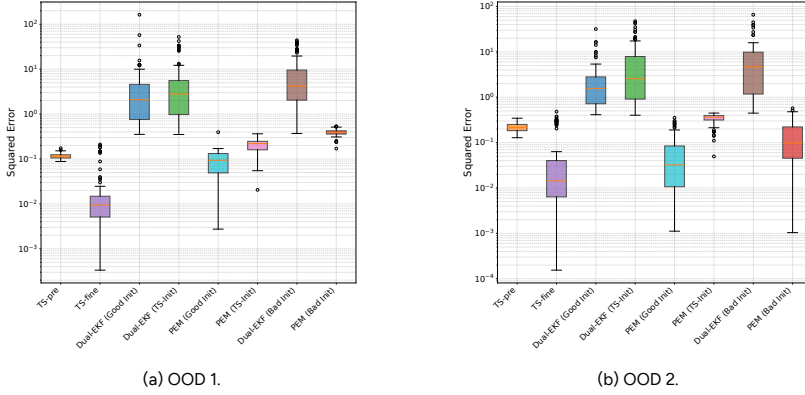


Figure 9.3.7: Cascaded water tanks: box plots of squared-error distribution over 100 MC runs.

The trajectory and residual plots are generated by simulating the noise-free system at each method's mean parameter estimate (averaged over 100 MC runs). The input signal used for these plots is the same fixed block-uniform realization that was used during the Monte Carlo estimation runs. This differs from Chapter 8, where the trajectory plots use a separate input realization; in both cases, however, all methods and the ground truth share the same input within each plot, so the comparisons are fair.

The trends observed for VDP carry over: TS-fine improves upon TS-pre and is competitive with PEM and dual-EKF under favorable initialization, while remaining initialization-free. The residual plots confirm that TS-fine tracks the true trajectory well, and the box plots show a tighter error distribution than the pretrained estimator and the poorly initialized baselines.

Remark 9.5. With the availability of a high-fidelity simulator, the structural model mismatch is negligible and the parameters are assumed to be identifiable. If structural mismatch dominates, parameters may be non-identifiable and trajectory-matching refinements can lead to biased estimates; addressing this is left for future work.

Remark 9.6. Trajectory-level fine-tuning inherits the standard identifiability limitations of prediction-error methods: if the deployed experiment is not sufficiently informative, i.e., the input/initial condition does not excite the relevant dynamics, then multiple parameter vectors can produce nearly identical output trajectories, and fine-tuning may converge to an output-equivalent parameter that is not unique.

Remark 9.7. The fine-tuning objective is generally nonconvex for nonlinear simulators and further parameterized through a neural network, hence, global convergence may not be guaranteed; rather, fine-tuning should be interpreted as a local refinement that empirically converges to a stationary point.

Remark 9.8. Fine-tuning is executed only when an OOD is detected, and its runtime scales roughly linearly with the number of optimization steps, about E forward/backward simulator passes. Thus, the overhead can be mitigated by amortizing updates and running fine-tuning periodically or event-driven instead of at every time step.

9.4 Conclusion

In this chapter, we have proposed a trajectory-matching scheme to fine-tune a simulation-driven estimator under OOD scenarios. The approach leverages a differentiable simulator to enable end-to-end backpropagation through the second-stage ResNet, updating the network weights to directly reduce the simulation error ℓ_{sim} between simulated and observed trajectories. Combined with the Wald-type feature-space OOD detector from Chapter 8, the fine-tuning is triggered only when needed. Across simulations on both VDP and cascaded water tanks, the fine-tuned TS estimator consistently outperforms its pretrained counterpart under OOD and achieves competitive accuracy relative to PEM and dual-EKF, without requiring any initialization.

Part 3: Applications in Control

Abbreviations

ARMAX	autoregressive moving-average with exogenous input
ARX	autoregressive with exogenous input
BB	black-box
CNN	convolutional neural network
DT	digital twin
FFNN	feed-forward neural network
GB	grey-box
GBM	gradient boosted machine
ISL	inverse supervised learning
PI	proportional-integral
PRBS	pseudo-random binary sequence
RL	reinforcement learning
RMS	root mean square
SNR	signal-to-noise ratio
TS	two-stage method
VRFT	virtual reference feedback tuning
WN	WaveNet

Notation and Symbols

$\mathcal{M}(\boldsymbol{\theta})$	Simulator parameterized by $\boldsymbol{\theta}$
$\boldsymbol{\theta}$	Parameter vector
$\mathbf{x}_k$	State vector
u_k, y_k	Scalar input and output of the plant
N	Length of each input-output sequence
M	Number of training samples
D_i^N	i -th input-output dataset of length N
$C(\boldsymbol{\Psi})$	Parametric controller
$\boldsymbol{\Psi}$	Controller parameters
k_p	Proportional gain
T_i	Integral time
$\mathcal{R}$	Model-based control design rule
Ξ	Mapping from input-output data to controller parameters
g	Second-stage function
h	First-stage compression function
$\tilde{D}_i^n$	Compressed data (ARMAX coefficients)
n	Dimension of compressed features
μ_s	Tyre-road friction scaling coefficient
m_δ	Vehicle mass variation
φ_m	Phase margin
ω_c	Cutoff frequency
$\mathcal{U}[a, b]$	The uniform distribution on $[a, b]$

10 The Two-Stage Approach for Controller Tuning

In Chapters 3–9, we studied a particular class of simulation-driven estimators, the TS estimator, its statistical frameworks, and asymptotic properties. Due to its computational advantage with respect to other estimation methods, in this chapter, we investigate the application of the TS approach to meta-learning the tuning rule of a controller. The underlying motivation for considering such an application is the availability of highly sophisticated DTs that are often used for control design [92], e.g. in the automotive industry. These models can be used to cheaply generate large amounts of data, and are typically used for testing and calibrating controllers.

However, although DTs are extremely faithful, the parameters of a real system might change over time, and robustness or control adaptation is necessary. In this chapter, we leverage the availability of large amounts of available simulated data, and the choice of some control design rule, to *learn* a law directly mapping fresh data to the control parameters. In a sense, we *meta-learn* the control design process.

The key idea of this chapter is to turn an existing model-based controller design rule—one that requires a calibrated model to compute controller parameters—into a data-driven rule that produces the same controller parameters directly from input-output data, without requiring an intermediate modelling step at deployment. To achieve this, we leverage the TS approach. Offline, using the DT, we generate input-output data for many realizations of the parameters and compress each dataset into Auto-Regressive Moving-Average with Exogenous-Input (ARMAX) coefficients (first stage). In parallel, for each realization, we apply the model-based design rule to the known model to compute the corresponding controller parameters. The ARMAX coefficients and the controller

parameters together form the training set for a supervised learning algorithm (second stage), which learns the mapping from compressed data to controller parameters. Once trained, this mapping can be applied to fresh input-output data from the real system to produce controller parameters directly, bypassing the need to identify a model or to have access to the design rule at run time.

10.1 Problem Statement

Consider a discrete-time single-input single-output model $\mathcal{M}(\theta)$

$$\mathcal{M}: \begin{cases} \mathbf{x}_{k+1} = \tilde{f}(\mathbf{x}_k, u_k, \theta), \\ y_k = \tilde{g}(\mathbf{x}_k, \theta), \end{cases} \quad (10.1.1)$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}$, $y_k \in \mathbb{R}$ are the state, input and output of the model, respectively. $\tilde{f}$ and $\tilde{g}$ are possibly non-linear functions. $\theta \in \mathbb{R}^p$ is a vector of parameters, $\theta = (\theta_1, \dots, \theta_p)^\top$, each endowed with a known prior probability density function (*pdf*) $\theta_i \sim \mathcal{D}_i$; this Bayesian assumption reflects the practitioner's prior belief about the plausible range of each parameter. We assume that we are able to obtain input-output sequences, say, $D_1^N, \dots, D_M^N$, of length N , from an available simulator of $\mathcal{M}$, given a specific value of the parameter vector.

Further, consider a parametric controller $C(\Psi)$, where $\Psi \in \mathbb{R}^{n_\Psi}$ is the vector of controller parameters, and assume that a model-based control design rule $\mathcal{R}: \mathcal{M}(\theta) \mapsto \Psi$ is available.

The assumptions made above are realistic, and indeed are motivated by the widespread use of high-fidelity simulators. These models are very good replicas of the system, apart from a few key parameters that are inherently prone to variations and cannot be known in advance. Under these considerations, we are able to generate a set of data, depicted in Table 10.1.1. Each row of the table shows how a realization of the parameters is mapped onto a collection of input-output time-domain data—the pairs $(u_i^{(j)}, y_i^{(j)})$, $i = 1, \dots, N$, $j = 1, \dots, M$ —and a parametric controller $C(\Psi_j)$, $j = 1, \dots, M$.

$\theta^{(1)}$	$D_1^N = \left\{ \left(u_1^{(1)}, y_1^{(1)} \right), \dots, \left(u_N^{(1)}, y_N^{(1)} \right) \right\}$	$C(\Psi_1)$
$\theta^{(2)}$	$D_2^N = \left\{ \left(u_1^{(2)}, y_1^{(2)} \right), \dots, \left(u_N^{(2)}, y_N^{(2)} \right) \right\}$	$C(\Psi_2)$
$\vdots$	$\vdots$	$\vdots$
$\theta^{(M)}$	$D_M^N = \left\{ \left(u_1^{(M)}, y_1^{(M)} \right), \dots, \left(u_N^{(M)}, y_N^{(M)} \right) \right\}$	$C(\Psi_M)$

Table 10.1.1: Collection of parameter samples, input-output trajectories, and designed controllers.

Given this framework, we are able to design a controller for a given realization of the model $\mathcal{M}(\theta)$ which is optimal—in the sense of satisfying the tuning rule $\mathcal{R}$. In practice, the parameters in θ are subject to variations, in which case a classical solution is to re-identify the system parameters, which can be computationally intensive. The *goal* is to leverage the available simulator to find a *direct* solution to obtain the controller parameters, *de facto* learning the tuning rule $\mathcal{R}$.

10.2 Existing Approaches

The above goal can be achieved through various existing solutions. We briefly explain them below and outline the difficulties involved, before proposing our approach in Section 10.4.

10.2.1 Modelling-based approaches

A standard procedure to tune the controller is to first derive a model for the system based on input-output data and domain knowledge, and then design the controller parameters for the identified model via a model-based control design rule. Two such modeling based approaches are: (i) grey box (GB) and (ii) black box (BB) modelling. Identification based approaches typically involve large costs in both time and effort. In addition, every time the system parameters change significantly, the modelling procedure [5] has to be carried out from scratch, which often takes more time.

10.2.2 Virtual reference feedback tuning

A different paradigm for data-driven control design is Virtual Reference Feedback Tuning (VRFT) [93]. VRFT is a *direct* method that allows the user to obtain a parametric controller via input-output data, without going through an identification step. It minimizes the difference between the closed-loop behaviour of the system under control and a reference model. While VRFT can provide a controller for which the closed loop is close to a desired reference model, this controller does not optimize a specific performance criterion. Thus, VRFT often requires a trial-and-error approach to adjust the virtual reference model. This can be time-consuming, especially when the system requires a precise level of tuning or when the desired performance is difficult to achieve.

10.2.3 Reinforcement learning based approaches

Several *reinforcement learning* (RL) approaches have been proposed for tuning parametric controllers in reference tracking. For example, [94] employs *twin delayed deep deterministic policy gradient* for a Proportional-Integral-Derivative (PID) controller tuning, [95] uses *deterministic policy gradient*,

[96] explores policy gradient methods for Proportional-Integral (PI) controllers, and [97] applies *proximal policy optimization*. Alternatively, RL can directly generate online control signals [98]. However, policy gradient methods may converge only to local optima, and there is no guarantee that the resulting PI/PID controllers are stabilizing. Moreover, RL requires real-time interaction with the environment, posing safety risks in safety-critical systems such as the automotive example in Section 10.5. While DTs can help in learning an optimal RL policy, ensuring stability remains a challenge. Inverse reinforcement learning and imitation learning, which rely on expert demonstrations, require formulating and learning a reward function [99], a task complicated by time-varying system parameters. These approaches also suffer from local minima issues and lack stability guarantees.

10.3 Motivation for Inverse Supervised Learning

The existing approaches outlined in Section 10.2 have several shortcomings. To overcome them, we assume in this work that a DT, or high-fidelity model structure capable of accurately capturing the dynamics of the real system, is available. This assumption is reasonable in many applications, where a significant modelling effort has already been spent. For example, car manufacturers commonly use DTs for tasks such as designing, sizing, and testing [100, 101, 17].

However, such DTs may not be suitable models for control design, because control models are typically developed ad hoc as simple, low-order representations specifically tailored for model-based control approaches. In addition, some of the parameters of the DT need to be fine-tuned to the specific system or its current operating conditions by collecting data from the real system [5]. For instance, as discussed in Section 10.5 in an automotive context, properties such as friction coefficients or the total mass of a vehicle can vary over time, so accounting for them requires re-calibrating the DT. Once the DT has been fine-tuned, one could design a controller using a model-based approach, which might involve performing a preliminary model reduction step to obtain a simpler model suitable for control design.

We instead propose a data-driven approach which avoids the explicit fine-tuning of the DT and its use for model-based control design during production, by using it solely to generate, offline, artificial time series data and controller parameters using model reduction and model-based control design methods. Leveraging this data, we then train a machine learning model that directly maps the time-series data to controller parameters. Once the optimal mapping from data to controller parameters is learned offline, it can be directly used to tune the controller used on the real system.

Our approach offers several advantages over a traditional model-based control design method: (1) it requires less time and computational effort, by avoid-

ing the modelling effort needed to fine-tune a DT and reducing it to a control-oriented model, (2) it meta-learns a tuning rule based on classical control theory, ensuring stability, and (3) leverages a DT for offline learning. We call this approach *Inverse Supervised Learning* (ISL) for controller tuning.

In this chapter, we present a Two-Stage (TS) implementation of ISL, the TS-ISL method, where the first stage compresses input-output data via ARMAX identification, and the second stage learns a mapping from compressed features to controller parameters.

10.4 The TS-ISL for Meta Learning a Controller Tuning Rule

The TS approach is a methodology for parameter estimation in white-box models, where a simulator of the plant is available. The goal in TS is that of finding a function Ξ , mapping the set of measured data to the parameter. In this case, we shift the problem from estimating an unknown model parameter to estimating the controller parameters Ψ , thus embedding the control design rule $\mathcal{R}$ within Ξ , that is,

$$\Xi^* = \arg \min_{\Xi \in \mathcal{A}} \frac{1}{M} \sum_{i=1}^M \left\| \Psi_i - \Xi \left(u_1^{(i)}, y_1^{(i)}, \dots, u_N^{(i)}, y_N^{(i)} \right) \right\|_2^2. \quad (10.4.2)$$

There are two ways to approach this problem. One way is to use a neural sequence model that operates directly on the raw input-output data, learning Ξ end-to-end without an explicit compression step; this is the subject of Chapter 11. A second approach is to decompose Ξ into two stages: a fixed compression of the data into low-dimensional features, followed by a supervised learning model that maps these features to the controller parameters. In this chapter, we take the latter approach, following the TS framework, and extend it to the controller tuning problem.

10.4.1 First stage

The collected data $D_1^N, \dots, D_M^N$ overall form a set of $M \times 2N$ numbers, as each input and output sequence consists of N values. This number can explode in case several unknown parameters or large sample sizes are considered; unfortunately, this is a typical case in real systems.

Dimensionality reduction can be employed to obtain a manageable amount of data, retaining the necessary information with a lower number of samples. Specifically, we aim to obtain a reduced data-set $\tilde{D}_i^n$, $i = 1, \dots, M$, where $n \ll N$, by employing a compression function $g: (D_1^N, \dots, D_M^N) \mapsto (\tilde{D}_1^n, \dots, \tilde{D}_M^n)$. Leveraging the fact that the D_i^N 's are generated by a dynamical system, [34] considered the identification of a black-box model as such compression function, which captures each time series with a few parameters. Different possibilities have been employed in the literature [102], we however consider in

the following the ARMAX polynomial model, widely versatile and easy to analyse [5]. A generic ARMAX model reads

$$\begin{aligned} y_k + \alpha_1 y_{k-1} + \dots + \alpha_{n_a} y_{k-n_a} = & \alpha_{n_a+1} u_{k-n_d} + \dots + \alpha_{n_a+n_b} u_{k-n_d-n_b+1} \\ & + \alpha_{n_a+n_b+1} e_{k-1} + \dots + \alpha_{n_a+n_b+n_e} e_{k-n_e} + e_k. \end{aligned} \quad (10.4.3)$$

The set of the polynomial model parameters $\tilde{D}_i^n = (\alpha_1, \dots, \alpha_n)^T \in \mathbb{R}^{n=n_a+n_b+n_e}$ is then collected and associated with the corresponding value of the parameter θ_i .

10.4.2 Second stage

Having compressed the data samples, we now aim at finding some function $g^* \in \mathcal{G}$ such that $\Xi^* = g^* \circ h$, by solving

$$\begin{aligned} g^* = \arg \min_{g \in \mathcal{G}} \frac{1}{M} \sum_{i=1}^M \left\| \Psi_i - g \left(h \left(u_1^{(i)}, y_1^{(i)}, \dots, u_N^{(i)}, y_N^{(i)} \right) \right) \right\|_2^2 \\ = \arg \min_{g \in \mathcal{G}} \frac{1}{M} \sum_{i=1}^M \left\| \Psi_i - g \left(\tilde{D}_1^n, \dots, \tilde{D}_M^n \right) \right\|_2^2, \end{aligned} \quad (10.4.4)$$

where $\mathcal{G}$ is a prescribed space of functions, or *hypothesis class* [4]. Some commonly used classes $\mathcal{G}$ that provide a reliable solution to Eq. (10.4.4) are Gradient Boosted Machines (GBMs) [60] and Feed-Forward Neural Networks (FFNNs), which we employ in the second stage of the TS approach. The functions $g \in \mathcal{G}$ can in turn be parameterized in terms of hyper-parameters $w \in \mathbb{R}^{N_w}$. In other words, $g^* = g(w^*)$, where

$$w^* = \arg \min_{w \in \mathbb{R}^{N_w}} \frac{1}{M} \sum_{i=1}^M \left\| \Psi_i - g(w) \left(\tilde{D}_1^n, \dots, \tilde{D}_M^n \right) \right\|_2^2. \quad (10.4.5)$$

Implementation details for GBM and FFNN are given in Section 10.6.

10.5 Case Study: The Yaw-Rate Tracking Problem

As a case study to test the methodology described in Section 10.4, we consider a yaw-rate tracking problem via steer-by-wire actuation, being of high relevance within the automotive context. We focus on the movement dynamics of a car, depicted in Figure 10.5.1.

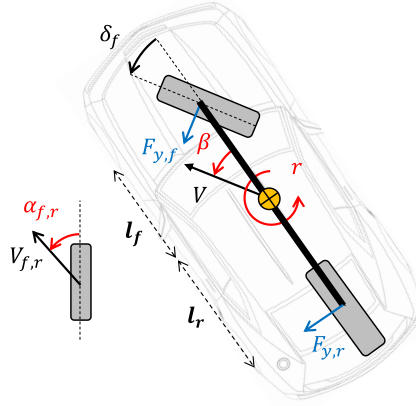


Figure 10.5.1: A schematic describing the dynamics of a vehicle.

Specifically, our interest lies in controlling the car's movement around sharp turns on the road, a critical concern given the varying road friction and vehicle mass, influenced by weather conditions and passenger load respectively.

Our objective is to achieve a desired yaw-rate to ensure safe turns, accounting for these variable parameters. We proceed by developing a mathematical model that relates yaw-rate to vehicle dynamics, considering variations in parameters such as mass and friction coefficient. This variability necessitates recalibration of the controller to maintain an effective control via tracking the desired yaw rate.

To this end, we propose a control design procedure employing a PI controller to regulate the yaw-rate reference. Subsequently, we test the efficacy of the data-driven recalibration procedure using the TS approach through simulation analysis. We compare the performance of this approach with that of a standard grey-box method.

10.5.1 System modelling

Upon the assumptions of small steering angle δ_f and front/rear wheels side-slip angles $\alpha_{f,r}$, the following set of equations describes the side-slip β and yaw-rate r dynamics [103]:

$$\begin{aligned}\dot{\beta} &= -r - \frac{C_f \alpha_f}{m_{veh} v_x} - \frac{C_r \alpha_r}{m_{veh} v_x}, \\ \dot{r} &= -\frac{l_f C_f \alpha_f}{J_z} + \frac{l_r C_r \alpha_r}{J_z},\end{aligned}\tag{10.5.6}$$

where l_f and l_r are the distances between the vehicle center-of-gravity and the front/rear wheels respectively, C_f and C_r are the cornering stiffnesses, m_{veh} is the vehicle mass, and J_z the yaw-axis vehicle inertia. The differential equations above can be simply obtained by balances of moments and forces in Figure 10.5.1, and they represent the well known single-track or bicycle vehicle model.

The tyre-road contact forces are approximated as $F_{y,f,r} = -C_{f,r}\alpha_{f,r}$, for small slip angles. In Equation (10.5.6), v_x is the vehicle longitudinal speed: it is indeed not a physical parameter, but rather a time-varying state of the system. However, it is common practice to consider it as a constant in the yaw-rate control problem [103, 104].

The wheel lateral slip angles $\alpha_{f,r}$ are computed via the kinematic equations

$$\alpha_f^{kin} = -\delta_f + \beta + \frac{L_f}{v_x}r, \quad \alpha_r^{kin} = \beta - \frac{L_r}{v_x}r. \quad (10.5.7)$$

However, it is well known that the tyre lag phenomenon introduces a phase shift between theoretical slips and actual ones [105, 106]. The tyre dynamics is thus introduced:

$$\dot{\alpha}_f = -\frac{v_x}{l_{rel,f}}(\alpha_f - \alpha_f^{kin}), \quad \dot{\alpha}_r = -\frac{v_x}{l_{rel,r}}(\alpha_r - \alpha_r^{kin}), \quad (10.5.8)$$

where $l_{rel,f/r}$ governs the lag phenomenon, and is known as *relaxation length*. Finally, the steer-by-wire actuator dynamics could be well within the range of the system dynamics. It is thus necessary to model it. A second order system is here considered, with two complex-conjugate poles (typical for this kind of actuator [107]):

$$\begin{aligned} \dot{z} &= -\omega_n^2\delta_f + \omega_n^2\delta_f^{cmd}, \\ \dot{\delta}_f &= z - 2\xi\omega_n\delta_f. \end{aligned} \quad (10.5.9)$$

Combining Eqs. (10.5.6) to (10.5.9) eventually yields a 6-th order Linear and Time-Invariant System, with state vector $x = (\beta \quad r \quad \alpha_f \quad \alpha_r \quad z \quad \delta_f)^T$, input $u = \delta_f^{cmd}$, and output $y = r$.

The cornering stiffness coefficient is highly dependent on the type of terrain, the tyre characteristics, temperature and so on [108]. For this reason, it is to be considered as an uncertain parameter, prone to significant variation throughout the vehicle functioning:

$$C_f = C_f^{nom} \cdot \mu_s, \quad C_r = C_r^{nom} \cdot \mu_s. \quad (10.5.10)$$

where μ_s is considered as a scaling coefficient for the tyre-road friction.

Another typical source of uncertainty is the vehicle mass, which usually changes about some nominal value, typically including the chassis and tyre payload, as well as the driver, i.e.,

$$m_{veh} = m_0 + m_\delta, \quad (10.5.11)$$

where m_δ is the mass variation.

For the considered model, we have identified two typical sources of uncertainty, the mass and the tyre-road friction, which can be collected in the vector $\theta \in \mathbb{R}^2$. We assume the following prior distributions for θ :

$$\theta = (\mu_s \quad m_\delta)^T, \quad \mu_s \sim \mathcal{U}(0.3, 1.1), \quad m_\delta \sim \mathcal{U}(0, 300), \quad (10.5.12)$$

where $\mathcal{U}(a, b)$ denotes the uniform distribution with support $[a, b]$. Finally, the system defined above is discretized with sampling period $T_s = 0.01$ s (*e.g.* also used in [103]). The physical parameters used in the simulations are given in Table 10.5.2.

Parameter	Value	Unit
m_{veh}	1895	kg
l_f, l_r	1.18, 1.53	m
J_z	2400	kg·m ²
$l_{\text{rel},f}, l_{\text{rel},r}$	0.4, 0.8	m
$C_f^{\text{nom}}, C_r^{\text{nom}}$	124.9, 166	N/(rad $\times 10^3$)
ω_n	$2\pi \cdot 5$	rad/s
ξ	0.9	—

Table 10.5.2: Vehicle model physical parameters.

The vehicle and actuator parameters are taken from [107], and the tyre relaxation lengths from [106].

10.5.2 Control design

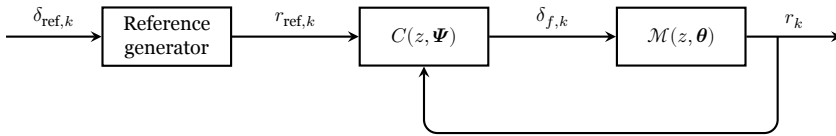


Figure 10.5.2: Control scheme for the yaw-rate tracking problem.

The objective in yaw-rate tracking problems is to regulate a reference behavior, which is generated using a driver's steering request based on ad-hoc logic. In practice, electronic systems track the driver's intent, thereby preventing unsafe situations [109].

One common method for controlling a linear time-invariant system is the loop shaping approach [35]. This method ensures the desired phase margin φ_m and cutoff frequency ω_c . These two parameters subsequently define the closed-loop

behavior of the system, in accordance with established principles of control theory. A PI controller is sufficient and is represented as follows:

$$C(z, \Psi) = \frac{k_p}{1 + 2T_i/T_s} \frac{z + 1}{z + \frac{T_s - 2T_i}{T_s + 2T_i}}. \quad (10.5.13)$$

The controller is discretized using the Tustin method at a sampling time of T_s , with the tuning parameters being the proportional gain and integral time, denoted as $\Psi = (k_p \ T_i)^T$.

By appropriately choosing the regulator parameters, it is possible to achieve the desired frequency-domain performance. We aim for $\varphi_m \geq 60^\circ$ as a fundamental threshold for ensuring an asymptotically stable and well-damped closed-loop response, and cutoff frequency $\omega_c \geq 1.5$ Hz. To implement the control tuning strategy, the loop shaping approach is employed, denoted as $\mathcal{R}(\mathcal{M}(\theta))$, and this is carried out using the MATLAB function `piddtune`.

10.6 Simulation Analysis

In this section, we apply the Two-Stage approach described in Section 10.4 to the case study of Section 10.5, showing its performance in terms of learning the control design rule.

10.6.1 Dataset generation

In order to test the Two-Stage approach, we generate a set of data from the yaw-rate dynamics model. We set $M = 1500$, so as to obtain a meaningful number of examples; they are randomly split between 80% for training and 20% for validation. Each dataset lasts 10 seconds, and it is sampled at a frequency of 1000 Hz, so $N = 10000$ data points in the time domain are available for each $D_i^N, i = 1, \dots, 1500$.

Further, we generate $M = 1500$ more scenarios, to evaluate the model on a different dataset (testing set). The testing set is also used to evaluate the benchmark identification and control tuning approach that we are going to describe in the following. The steer input $\delta_{f,k}, k = 1, \dots, N$, is selected to be a Pseudo-Random-Binary signal, so as to provide a meaningful excitation to the system; the output yaw-rate $r_k, k = 1, \dots, N$, is perturbed with Gaussian white noise, so as to achieve a signal-to-noise ratio of ≈ 30 dB.

10.6.2 Gray-box identification benchmark

As a benchmark to compare the TS approach to, we consider gray-box (GB) identification. This benchmark is possibly the best solution to be carried out

when the model structure and some initial value of the parameters are approximately known, and only some refinement is necessary. Given the initial model $\mathcal{M}(\theta_{\text{init}})$, we refine it using available data, obtaining $\mathcal{M}_{gb}$. The prediction error minimization approach can be utilized to adjust the model parameters (we consider here the MATLAB implementation `pem`).

Once the refined model has been identified, one can apply the loop shaping tuning rule described in Section 10.4.

10.6.3 Model compression details

For the model compression phase, we consider an ARMAX model of order $(2, 2, 2, 1)$. Such an ARMAX model is a 2nd order system. Compared to the data generator, which is a 6-th order one, we significantly reduce the model dimensionality.

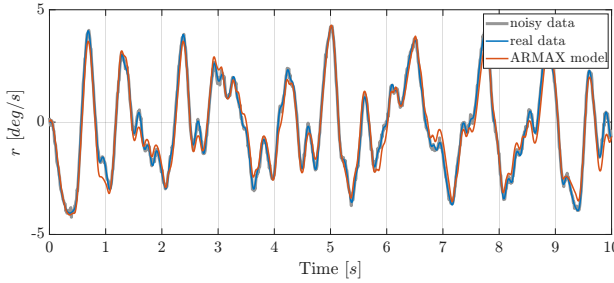


Figure 10.6.3: Time-domain representation of noiseless data samples, noisy ones, and ARMAX fitting performance.

An example of the ARMAX fitting performance is shown in Fig. 10.6.3, where noise-perturbed data are shown too. As one can notice, the reduced order model exhibits some prediction error; nevertheless, we will show that the extracted ARMAX features are enough to obtain a good learning of the loop-shaping rule.

10.6.4 Second stage training details

We try GBM and FFNN in the second stage to estimate the controller parameters k_p and T_i . The reason to choose these methods as function approximators is due to their capabilities to capture non-linearities in the data (corresponding to the ARMAX coefficients that we obtain from the first stage of the TS approach). One can simply use FFNN alone as a function approximator, but more recently it has been demonstrated empirically that GBMs outperform neural networks for tabular data and the training time is remarkably short in GBMs [110]. Hence, for the sake of fair comparison, we use GBM in the second stage.

To implement GBM, we use XGBoost, a Python package developed by [46]. More details on using the XGBoost package and the implications of its hyperparameters can be found in [46]. FFNN is implemented in the MATLAB Deep Learning Toolbox [111]. Their training details are as follows:

- **XGBoost:** We have selected, via cross-validation, `n_estimators = 2000`, `max_depth = 7`, `max_leaves = 10`, `learning_rate = 0.1`, `reg_lambda = 4`, and `min_child_weight = 25` for the tuning parameters of XGBoost. We run two separate XGBoost algorithms in parallel with the stated choices of parameters to predict k_p and T_i .
- **FFNN:** Model using `feedforwardnet` function, with 10 neurons in the hidden layer, `tanh` activation, and a linear output layer. The Levenberg-Marquardt (`trainlm`) algorithm is considered for training. `epochs` is set to 1000; `mu` is set to 0.001 (a parameter regulating gradient descent in `trainlm`); `Mu Decrease Ratio` and `Mu Increase Ratio` are respectively set to 0.1 and 10. `max_fail` – the maximum number of failed validation checks before stopping – is set to 5. The other parameters are left at MATLAB default.

10.6.5 Controller parameters identification performance

Figures 10.6.4 and 10.6.5 respectively depict the proportional gain k_p and integral time T_i estimation performance, for the three different methods.

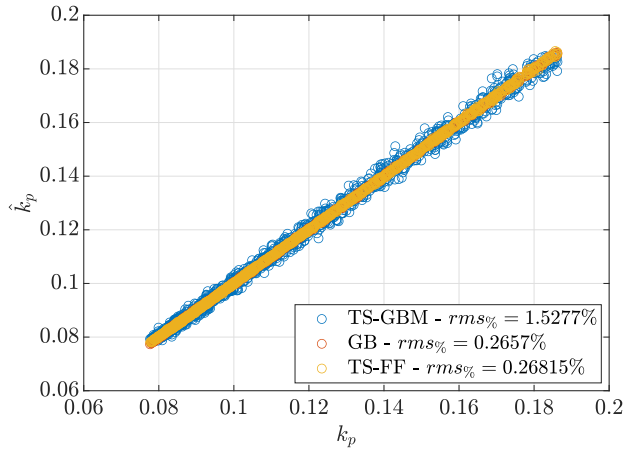


Figure 10.6.4: Regression plot showing the fitting of k_p in the testing case, for both considered methods.

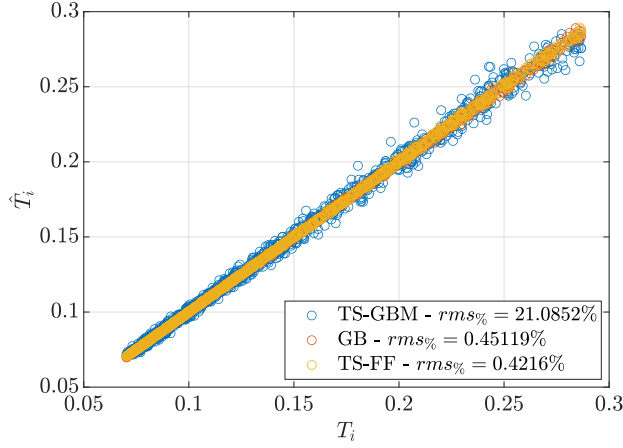


Figure 10.6.5: Regression plot showing the fitting of T_i in the testing case, for both considered methods.

As one can notice, good performance is achieved for all methods, and the average value is properly estimated, although TS-GBM exhibits some variance, especially for high values of T_i . The figures also give the $rms\%$ error metric, which is defined – for a generic parameter ν and its estimated $\hat{\nu}$ – as

$$rms\%(\nu, \hat{\nu}) = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} \left(100 \cdot \frac{\nu_i - \hat{\nu}_i}{\nu_i} \right)^2}. \quad (10.6.14)$$

Fitting the controller parameters is one way to assess the estimators goodness: however, it is interesting to look at some closed-loop system metrics to see what the impact of an error in fitting the integral reset time is. Table 10.6.3 provides statistics about the phase margin and cutoff frequency of the open-loop transfer function $C(z) \cdot \mathcal{M}(z)$, obtained by using the estimated controller parameters.

	φ_m [deg]		ω_c [Hz]	
	Mean	Std.	Mean	Std.
GB	59.98	0.2385	1.50	0.0037
TS-FF	59.99	0.1714	1.50	0.0032
TS-GBM	60.01	0.69	1.50	0.0143

Table 10.6.3: Loop-shaping metrics statistics. “Mean” and “Std.” respectively stand for average value and standard deviation, over a set of experiments.

One can note that all methods satisfy the control performance requirements of Section 10.5.2, yielding a phase margin $\varphi_m \approx 60$ deg and $\omega_c \approx 1.5$ Hz. In

fact, the phase margin achieved with TS-GBM reflects the increased variance of the estimator, but the impact on closed-loop performance is absolutely irrelevant.

¹Figure 10.6.6 provides a computational time analysis for the proposed approach and the benchmark. It can be seen that the TS approaches (TS-FF and TS-GBM) take less time in computing the controller estimates compared to the benchmark. Among the proposed approaches, TS-GBM has slightly better computational advantage over TS-FF.

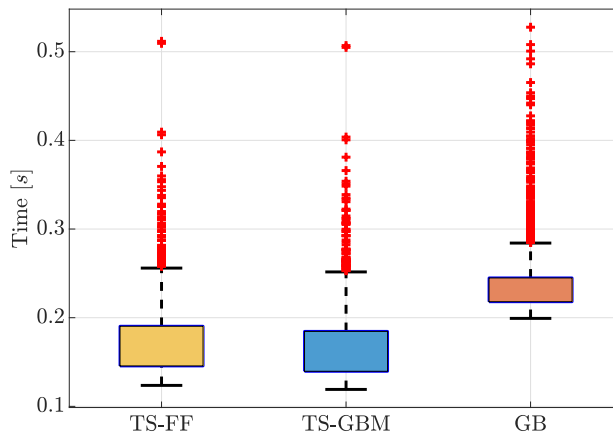


Figure 10.6.6: Analysis of computational times necessary for executing all methods.

Figures 10.6.7 shows the controller performance in the time-domain. The three controllers yield practically the same results, and this confirms the almost equivalence between the proposed approaches. Indeed, the reference tracking per-se can be improved, *e.g.* by selecting a more complex controller, which is however not in the scope of this work.

¹In a box plot, red spots show outliers, the center line marks the median, and the box spans the 25th to 75th percentiles.

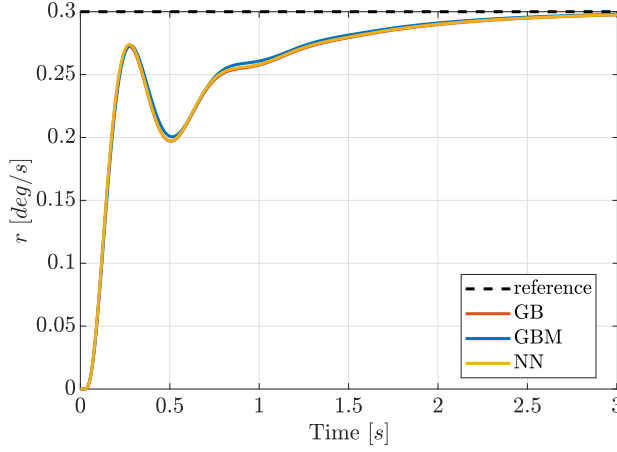


Figure 10.6.7: Time domain closed-loop behaviour of the system, for $\theta = (0.60, 44.14)^\top$.

10.7 Conclusion

This chapter has presented a novel methodology for meta-learning in control systems: we employ a Two-Stage approach, with a first stage consisting in model order reduction, and a second one consisting in direct estimation via black-box models. Specifically, we see how a possibly unknown control tuning rule can be learned from simulated data, and then applied to generate new suitable controllers just from fresh data. Simulation analyses have shown that the presented approach can achieve performance similar to an identification-for-control method, while providing an improvement in computational time.

11 Direct Learning of Controller Tuning Rules

In Chapter 10, we leveraged the TS approach to *meta-learn* a controller tuning rule, thereby reducing the computational time to output the right controller parameters once the respective functionals are trained using an available DT. However, this involved an identification step i.e., the first stage which was an ARMAX model and its coefficients acted as the compressed data. In this chapter, we shall take an alternate route by focusing directly on meta-learning the tuning rule of a PI controller, bypassing the identification step via sequence models. In particular, we use a Convolutional Neural Network (CNN) and WaveNet (WN) to serve this purpose. Furthermore, we demonstrate the use of WN, originally proposed in [112] as a generative neural network architecture for text to speech synthesis, to learn the tuning rule. In particular, the causal nature of WNs helps us capture long term dependencies in the input-output data generated by the DT.

11.1 Problem Statement

As in Chapter 10, we consider a single-input single-output discrete-time system $\mathcal{M}(\boldsymbol{\theta})$

$$\mathcal{M}: \begin{cases} \mathbf{x}_{k+1} = \tilde{f}(\mathbf{x}_k, u_k, \boldsymbol{\theta}), \\ y_k = \tilde{g}(\mathbf{x}_k, \boldsymbol{\theta}), \end{cases} \quad (11.1.1)$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}$ and $y_k \in \mathbb{R}$ are the state vector, input and output of the system. $\tilde{f}$ and $\tilde{g}$ are state and output functions, possibly non-linear. $\boldsymbol{\theta} \in \mathbb{R}^p$ is a vector of parameters, $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)^\top$, each endowed with a known prior probability density function (pdf) $\theta_i \sim \mathcal{D}_i$; this Bayesian assumption reflects the

practitioner's prior belief about the plausible range of each parameter. Assuming that we have a DT, we generate input-output sequences, say, $D_1^N, \dots, D_M^N$, of length N , from an available simulator of $\mathcal{M}$, given a specific value of the parameter vector and collect a set of data as depicted in Table 10.1.1.

Given the available information in Table 10.1.1, we want to find the best control parameters satisfying tuning rule $\mathcal{R}$. At run-time, a mere series of time-domain information is available. We can exploit these data so as to estimate the controller parameters directly, in a data-driven fashion. Precisely, we want to learn function $\Xi: D_i^N \mapsto \hat{\Psi}$ as

$$\Xi^* = \arg \min_{\Xi \in \mathcal{A}} \frac{1}{M} \sum_{i=1}^M \|\Psi_i - \Xi(D_i^N)\|_2^2, \quad (11.1.2)$$

where $\mathcal{A}$ is a suitable functional space. This implies that we are seeking for the best estimator of the controller parameters given the available data. In general, finding Ξ in a general setting is not an easy task, as this means extracting some meaningful features from time-domain data, and mapping these features – that have to intrinsically contain some information about the model parameters – onto the best controller parameters. The problem thus translates into the selection of a proper class of functions, for which neural networks are suitable candidates. In this chapter, we consider CNNs and WNs to solve Eq. (11.1.2).

11.2 CNN-Based Meta-Learning of Controller Tuning Rule

CNNs are a particular class of non-linear functions with an extremely high approximation capability; in fact, they have been used for a long time in image classification tasks, but applications with time-domain data recently emerged [113, 114]. The considered CNN architecture is displayed in Fig. 11.2.1. The model input is $\mathcal{J} \in \mathbb{R}^{2 \times N}$: in practice, the system input and output time-series data are collected row-wise. Each column of $\mathcal{J}$ represents the k -th time step for the input-output pair.

Three series of 2-D convolution filters are applied to $\mathcal{J}$; a *ReLU* transformation is applied after each bank of filters¹. Details on this kind of neural networks are given in [115]. The last layer is a Fully Connected one, and provides the estimate of the controller parameters $\hat{\Psi}$. The number of filters and kernel sizes in the actual implementation are discussed later in Section 11.4. For training purposes, other layers are added, following common practice in CNN design, namely batch normalization and dropout.

¹The ReLU function is defined as $\text{ReLU}(x) := \max(0, x)$.

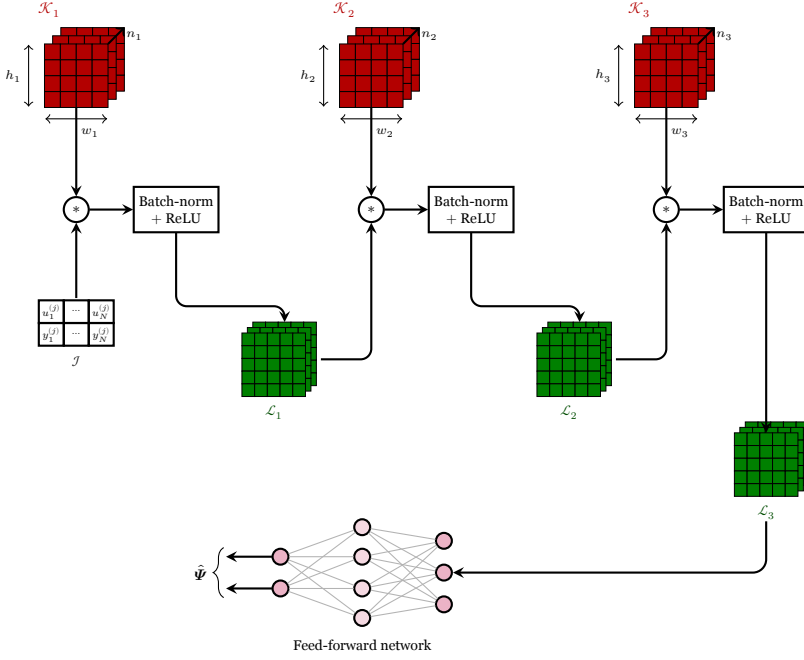


Figure 11.2.1: Architecture of the considered CNN. $\mathcal{K}_i$ denotes a filter bank that consists of kernels of dimensions $h_i \times w_i \times n_i$, followed by batch-norm and ReLU operations. h_i and w_i denote the height and width of a kernel respectively and n_i denotes the number of filters. $\mathcal{L}_i$ is the output corresponding to $\mathcal{K}_i$.

11.3 WN-Based Meta-Learning of Controller Tuning Rule

WNs [112] are auto-regressive neural-network models that capture long term dependencies in the data. These models were originally proposed for audio signal processing to predict the next word in the audio or translate text to speech. A WN model is based on the principle of causal 1-D convolution: The outputs of a given hidden or output layer depend only on the present and past values of the outputs of the preceding layer. The range of such dependencies is pre-specified by a kernel of size K , which performs the convolution. This is different from the vanilla CNN where the outputs have non-causal dependencies. To capture the long-term dependencies, dilations are introduced of order of 2^l , where $l = 0, 1, \dots, L - 1$ and L is the number of layers. A schematic of the WN architecture is shown in Fig. 11.3.2.

The input layer of a WN architecture has input-output data from a dynamical system arranged as an alternating sequence, denoted as $\{(u_i, y_i)\}_{i=1}^N$, where u_i represents the input at time i and y_i is the corresponding output.

Each neuron in a hidden layer of a WN is an activation function that is com-

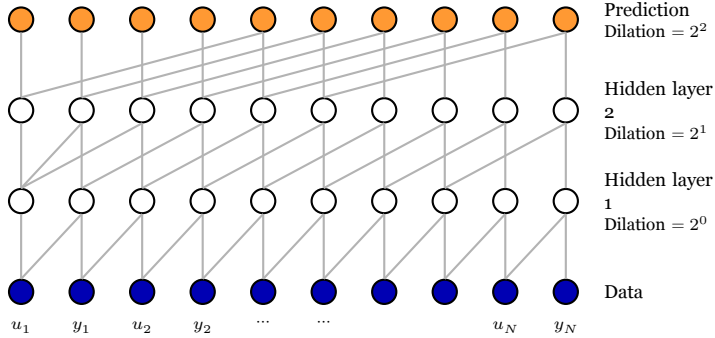


Figure 11.3.2: WaveNet architecture for prediction. The network consists of stacked hidden layers with exponentially increasing dilation factors ($2^0, 2^1, 2^2, \dots$), enabling the receptive field to grow exponentially with depth while maintaining a manageable number of parameters.

posed of several mathematical operations as shown in Fig. 11.3.3. During the training phase of a WN, the causal 1-D convolution is performed on each input-output data sequence $(u_i^{(j)}, y_i^{(j)})$, whose corresponding controller parameters are Ψ_j (see Table 10.1.1). The outputs of the convolution are then fed to the L stacked hidden layers as shown in Fig. 11.3.3.

Each hidden layer consists of a dilated convolution followed by two parallel operations: (i) the outputs of the dilated convolution are first mapped by *tanh* and *sigmoid* (σ) operations, and then multiplied, followed by a 1-D convolution (denoted by the 1×1 box in Fig. 11.3.3), and (ii) a residual operation, where the outputs of the causal convolution are added with the outcomes of the 1×1 block corresponding to (i). The first operation is known as a *skip connection*. The output layer of the WN architecture is the sum of skip-connections of L stacked hidden layers followed by a 1-D convolution. The input $(u_i^{(j)}, y_i^{(j)})$ to the WN has length $2N$, whereas its output is another sequence of length $2N$, which is aggregated to yield $\hat{\Psi}_j$ (the estimate of the controller parameters Ψ_j). For example, the last entry of the sequence is taken as the estimate of Ψ_j .

By choosing appropriate values for K and L , the estimate of Ψ_j can capture the long-term dynamics of time domain input-output data $(u_i^{(j)}, y_i^{(j)})$, $i = 1, \dots, N$, $j = 1, \dots, M$.

11.4 Simulation Analysis

In this section, we apply the ISL (cf. (11.1.2)) to the case study in Section 10.5, demonstrating its effectiveness in learning the control design policy. To train the neural networks (NNs), we generate data from the yaw-rate dynamics model. We set $M = 40000$ and sample parameters $\{\tilde{\theta}_i\}_{i=1}^M$. The data is split

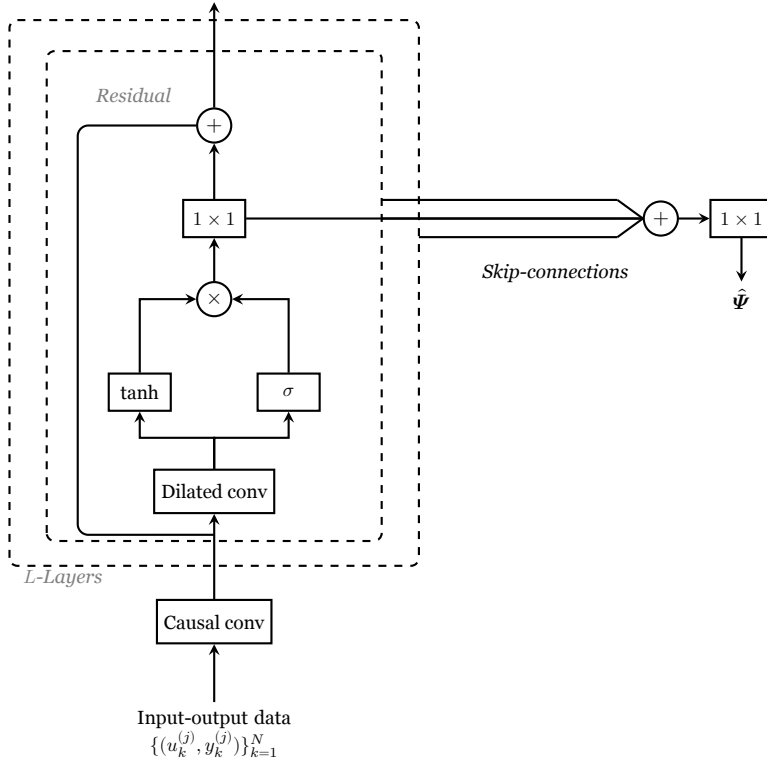


Figure 11.3.3: Operations performed in the WaveNet.

90% for training and 10% for validation. For each parameter sample, the steer input $\delta_{f,k}$, $k = 1, \dots, N$, is a Pseudo-Random Binary Signal (PRBS) to ensure meaningful system excitation, while the yaw-rate output r_k is perturbed with Gaussian noise to achieve a signal-to-noise ratio of ≈ 30 dB. The steer input, yaw-rate, and parameter samples form time-domain input-output data (see Table 10.1.1). The control design rule is as specified in Section 10.5 ($C(z, \Psi)$). To test the trained NNs, we generate an additional $M = 1000$ scenarios, forming a testing set. This set is also used to evaluate benchmark identification and control tuning methods, which will be discussed next.

Identification benchmarks. In order to benchmark the inverse supervised learning approach against, we consider gray-box (GB) identification, where the initial model $\mathcal{M}(\theta_{\text{init}})$ is refined using available data, obtaining $\mathcal{M}_{gb}$. The prediction error minimization approach can be utilized to adjust the model parameters (using the MATLAB command `pem`). Once the refined model has been identified, one can apply the loop shaping rule. Black-box (BB) identification infers a generic model structure from input-output data. As described in [102],

the process involves: (i) selecting the best ARX model based on prediction loss; (ii) training models like ARX, ARMAX, BJ, and OE; (iii) validating on a separate dataset to pick the best model; and (iv) tuning the controller using the optimal model $\mathcal{M}_{bb}$ and the loop-shaping rule $\mathcal{R}$. Further details are in the reference.

Virtual Reference Feedback Tuning. In order to compare VRFT to the other approaches, which are based on loop shaping, we select as reference model a (discrete-time) low-pass filter with unitary gain and a single pole at ω_c (the cutting frequency); we know, from basic control theory, that if the phase margin $\varphi_m \geq 80^\circ$, we can safely approximate the closed-loop behaviour in this way.

TS-ISL: We compare our proposed method against the TS-ISL (see Section 10.4 of Chapter 10), which relies on an intermediate step. In particular, TS gradient boosting machine (TSGBM) and TS Feed-Forward (TSFF) are considered. The implementation details are given in Chapter 10.

Direct ISL. The time-series for training and validation consist of 250 samples with two channels (input and output), making the CNN input $\mathcal{I}$ of size 250×2 . The first convolution uses a 2×2 kernel with 96 filters to capture short-term effects, while the next two layers capture long-term effects with kernel sizes 120×1 and 20×1 , and 48 and 96 filters, respectively. Kernel sizes are selected based on input size, and the filter numbers were determined via trial and error. The network functions by extracting features analogous to identification and applying control design rules. For the WN, the training and validation sets contain 500 samples each, with input-output data pairs. We use a kernel size of $K = 5$ and $L = 10$ layers. Each causal 1-D convolution has 4 output channels, and the final convolution for $\hat{\phi}$ has 1 channel. Two WNs are trained in parallel to estimate k_p and T_i . The Adam optimizer [116] is used with learning rate $\text{lr} = 1e^{-3}$ and $\beta = (\beta_1, \beta_2) = (0.9, 0.999)$ over 50 epochs.

Discussion. Figures 11.4.4 and 11.4.5 show that ISL-based approaches achieve lower root mean square errors compared to BB and VRFT. Fig. 11.4.6 confirms that both TS-based ISL and direct ISL methods are faster than BB and GB. However, our proposed direct ISL approaches (WN, CNN) outperform TS ISL methods (TSGBM, TSFF) by avoiding the preprocessing step, such as the compression stage required in TS-based ISL schemes. Notably, WaveNet (WN) outperforms CNN due to its ability to efficiently capture long- and short-term dependencies with fewer operations. Fig. 11.4.7 shows that all methods produce controllers with acceptable phase margins, with minor differences (e.g., 78° vs. 80°) being practically negligible.

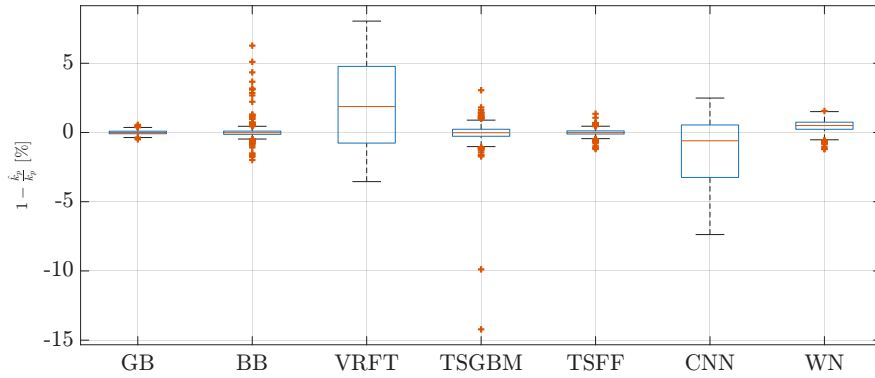
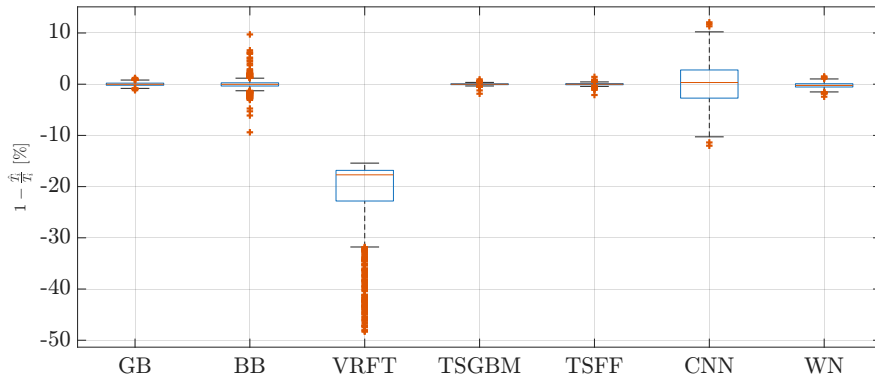
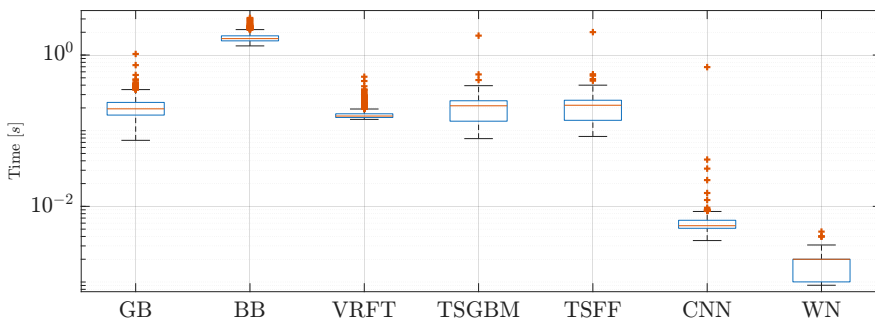
Figure 11.4.4: Box-plot of the errors in k_p .Figure 11.4.5: Box-plot of the errors in T_i .

Figure 11.4.6: Computational time comparison.

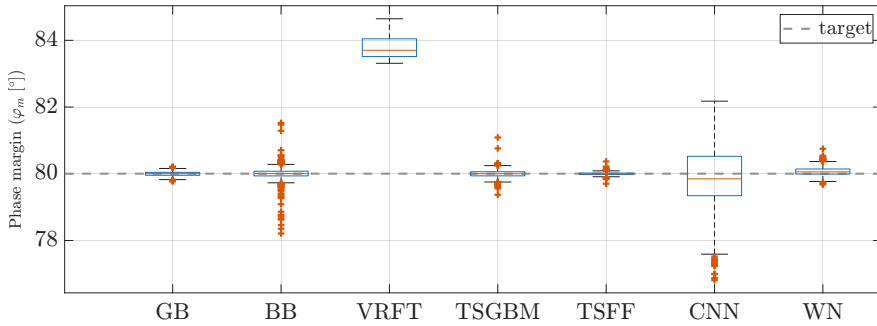


Figure 11.4.7: Phase margin comparison.

11.5 Conclusion

In this chapter, we have shown how a control design rule can be learned by means of neural networks. Leveraging the use of a Digital Twin, we generate large amounts of artificial data. For each batch of data, we design a controller. These data-controller pairs are then used to train a black-box model via supervised learning; the model is then directly used on fresh data for finding the controller satisfying the learned rule. Compared to the TS-based approaches in Chapter 10, the direct ISL methods, in particular WN, achieve competitive accuracy while requiring shorter computation times.

Part 4: Privacy in Point Estimation

Abbreviations

CVXPY	convex optimization library for Python
DP	differential privacy
DT	digital twin
i.i.d.	independent and identically distributed
LBaPP	Laplace Bayes Private Point
MSE	mean squared error
pdf	probability density function
pmf	probability mass function
r.v.	random variable
s.t.	subject to
TS	two-stage method
UBaPP	Unified Bayes Private Point

Notation and Symbols

θ	Parameter vector
Θ	Parameter space
$\mathbf{y}$	Observation vector, $\mathbf{y} = (y_1, \dots, y_N)^\top$
$\mathcal{Y}$	Observation space
$f(\cdot; \theta)$	Parametric distribution of each observation
π	Prior distribution on Θ
$\ell(\cdot, \cdot)$	Loss function
$R(\theta, \delta)$	Risk of estimator δ at θ
$r(\pi, \delta)$	Bayes risk of estimator δ under prior π
δ_B^*	Non-private Bayes point estimator
ε	Privacy parameter
$d_H(\cdot, \cdot)$	Hamming distance
$\sigma_{\delta_B^*}$	ℓ_1 -sensitivity of the Bayes estimator
$\delta_{\text{lpb}, \varepsilon}$	LBaPP estimator
$\delta_{p, \varepsilon}$	Randomized private Bayes estimator
$\delta_{p, \varepsilon}^*$	UBaPP estimator
S	Sufficient statistic
$\mathcal{T}$	Space of sufficient statistics
t	Value of sufficient statistic
ν_θ	Probability kernel of t given θ
P	Probability transition matrix
L	Loss matrix
V	Likelihood matrix
$\mathcal{U}[a, b]$	The uniform distribution on $[a, b]$

12 Differentially Private Point Estimation

Part 1 of the thesis explored simulation-driven parameter estimation, with a particular focus on analyzing statistical perspectives and asymptotic properties of one of the earliest data-driven estimators, the TS estimator. Simulation-driven estimation has broad applications, including control systems, as demonstrated in Part 3 of the thesis. Leveraging the abundance of data through the generation of synthetic data using a DT can significantly reduce inference time, as showcased in the control system application discussed in Part 3.

However, in certain contexts, the abundance of data poses challenges. Particularly in sectors like medicine, finance, and education, where inference tasks rely on individual-specific data, privacy concerns become paramount. Given that such data may contain sensitive information such as age, sex, gender, and ZIP code, it is imperative to protect this information before releasing it to the public.

In Chapter 1, we underscored the importance of privacy in estimation problems and provided some contexts where it is especially relevant. In this chapter, we introduce an alternative approach, termed the Unified Bayes Private Point (UBaPP) estimator, where we incorporate DP constraints into a Bayes parameter estimation problem. As outlined in the subsequent sections of the chapter, we directly integrate DP constraints into the Bayes risk minimization objective.

To this end, we precisely formulate our problem in Section 12.1. Following this, we propose our approach in Section 12.2, and provide numerical demonstrations of our approach in Section 12.3. Finally, we draw conclusions in Section 12.4.

12.1 Problem Formulation

We consider the problem of Bayes point estimation under a differential privacy (DP) constraint. Recall from Chapter 2 that the Bayes point estimator minimizes the Bayes risk

$$\delta_B^* := \arg \min_{\delta: \mathcal{Y} \rightarrow \Theta} r(\pi, \delta), \quad r(\pi, \delta) = \int_{\Theta} R(\theta, \delta) \pi(\theta) d\theta, \quad (12.1.1)$$

where $R(\theta, \delta)$ is the risk of estimator δ at θ , and π is a prior distribution on Θ . We assume that the observations $\mathbf{y} = (y_1, \dots, y_N)^\top \in \mathcal{Y} \subseteq \mathbb{R}^N$ are i.i.d. draws from a distribution $f(\cdot; \theta)$ parameterized by an unknown $\theta \in \Theta \subseteq \mathbb{R}^d$. In addition to minimizing the Bayes risk, we require the estimator δ to be ε -differentially private (see Definition 2.7 in Chapter 2), i.e., for each pair of neighbouring observations $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$ with Hamming distance $d_H(\mathbf{y}, \mathbf{y}') = 1$ and any measurable set $A \subseteq \Theta$,

$$\mathbb{P}[\delta(\mathbf{y}) \in A] \leq e^\varepsilon \mathbb{P}[\delta(\mathbf{y}') \in A]. \quad (12.1.2)$$

In the standard Laplace mechanism approach, this problem is solved in two steps. First, given observations $\mathbf{y} = (y_1, \dots, y_N)^\top \in \mathcal{Y} \subseteq \mathbb{R}^N$, where $y_i \stackrel{i.i.d.}{\sim} f(\cdot; \theta)$ and $\mathcal{Y}$ is the observation space, a non-private Bayes point estimate of θ , denoted by $\hat{\theta}_B^* = \delta_B^*(\mathbf{y})$, is obtained via (12.1.1). Second, Laplace noise is added to this non-private estimate to obtain the *Laplace Bayes Private Point* (LBaPP) estimate

$$\hat{\theta}_{\text{lpb}, \varepsilon} = \delta_{\text{lpb}, \varepsilon}(\mathbf{y}) = \delta_B^*(\mathbf{y}) + \text{Lap}\left(0, \frac{\sigma_{\delta_B^*}}{\varepsilon}\right), \quad (12.1.3)$$

which satisfies the privacy constraint (12.1.2) (see Figure 12.1.1). Here,

$$\sigma_{\delta_B^*} = \sup_{\substack{\mathbf{y}, \mathbf{y}' \in \mathcal{Y} \\ d_H(\mathbf{y}, \mathbf{y}') = 1}} \|\delta_B^*(\mathbf{y}) - \delta_B^*(\mathbf{y}')\|_1 \quad (12.1.4)$$

is the ℓ_1 -sensitivity (see Definition 2.8), and $\mathbf{y}, \mathbf{y}'$ are neighbouring observations (see Definition 2.6) with Hamming distance $d_H(\mathbf{y}, \mathbf{y}') = 1$.

Although $\delta_{\text{lpb}, \varepsilon}$ satisfies the privacy constraint, two important questions need to be addressed:

- Is the Laplace noise addition “optimal” in the sense of satisfying (12.1.1) subject to a DP constraint?
- What if we do not have a closed-form expression for $\delta_B^*(\mathbf{y})$ that is required to compute $\sigma_{\delta_B^*}$?

These two questions motivate the development, in the next section, of an alternative approach to find the optimal Bayes point estimator under a DP constraint.

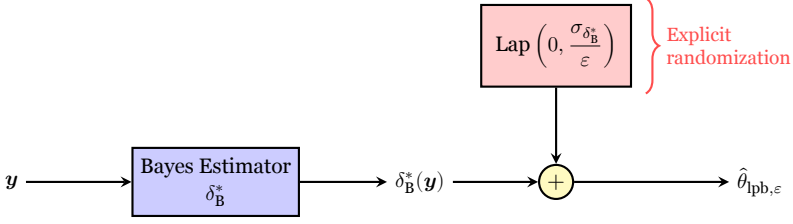


Figure 12.1.1: Standard approach to DP Bayes point estimation (LBaPP). The Laplace mechanism adds explicit randomization to the non-private Bayes estimate $\delta_B^*(\mathbf{y})$.

12.2 Proposed Approach

The DP constraint (12.1.2) requires the estimator to be randomized: a deterministic mapping from observations to parameters cannot satisfy (12.1.2) unless it is trivially constant. Rather than adding explicit noise to a non-private estimate as in the Laplace mechanism, we allow the randomization to be *implicit* by replacing the deterministic estimator $\delta(\cdot)$ with a *randomized private Bayes estimator* $\delta_{p,\epsilon}(\cdot)$, which is a probability density over the parameter space Θ for each $\mathbf{y} \in \mathcal{Y}$.

To formalize this, let $(\Theta, \mathcal{B}_\Theta)$ be the measurable space of parameters, $(\mathcal{Y}, \mathcal{B}_\mathcal{Y})$ the measurable space of observations endowed with the Hamming distance d_H , and π a prior distribution on Θ . Let $S: \mathcal{Y} \rightarrow \mathcal{T}$ be a sufficient statistic, where $(\mathcal{T}, \mathcal{B}_\mathcal{T})$ is a measurable space. We assume that S is *locally injective*, meaning $S(\mathbf{y}) \neq S(\mathbf{y}')$ whenever $d_H(\mathbf{y}, \mathbf{y}') = 1$, so that the DP constraint can be expressed in terms of $\mathbf{t} = S(\mathbf{y})$. Let $(\nu_\theta)_{\theta \in \Theta}$ denote the probability kernel of $\mathbf{t}$ given θ , and let $\ell: \Theta \times \Theta \rightarrow \mathbb{R}_{\geq 0}$ be a loss function. The randomized estimator is now conditioned on $\mathbf{t}$ rather than $\mathbf{y}$, and its Bayes risk is

$$R(\delta_{p,\epsilon}, \pi) = \int_{\theta \in \Theta} \int_{\mathbf{t} \in \mathcal{T}} \int_{\tilde{\theta} \in \Theta} \ell(\theta, \tilde{\theta}) \delta_{p,\epsilon}(\tilde{\theta} | \mathbf{t}) \nu_\theta(\mathbf{t}) \pi(\theta) d\theta d\mathbf{t} d\tilde{\theta}. \quad (12.2.5)$$

Note that $R(\delta_{p,\epsilon}, \pi)$ is a linear function of $\delta_{p,\epsilon}$. In standard Bayes point estimation, the estimate is considered as a deterministic function of the samples, but here, due to the DP constraint, the estimate will have to rely on some type of randomization mechanism. This is contrary to the standard Laplace mechanism, where the randomization is explicit. We will construct an estimator defined in terms of $\delta_{p,\epsilon}$ that is “optimal” in the sense of satisfying (12.1.1) subject to a DP constraint.

From the definition of DP (cf. (2.5.13)), and that S is locally injective, it follows that for each $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$ with $d_H(\mathbf{y}, \mathbf{y}') = 1$, there exist $\mathbf{t} = S(\mathbf{y})$ and $\mathbf{t}' = S(\mathbf{y}')$ such that

$$\delta_{p,\epsilon}(\tilde{\theta} | \mathbf{t}) \leq e^\epsilon \delta_{p,\epsilon}(\tilde{\theta} | \mathbf{t}'), \quad \text{for each } \tilde{\theta} \in \Theta. \quad (12.2.6)$$

Note that (12.2.6) is *equivalent* to DP, i.e., (12.2.6) holds if and only if (12.1.2) holds. Also, note that (12.2.6) a linear constraint on $\delta_{p,\varepsilon}$.

By combining (12.2.5) and (12.2.6), we can define the optimal ε -DP Bayes estimator $\delta_{p,\varepsilon}^*$ as the minimizer of the following optimization program:

$$\begin{aligned}
 \min_{\delta_{p,\varepsilon} \in \mathcal{P}(\mathcal{T}, \Theta)} \quad & \int_{\Theta} \int_{\mathcal{T}} \int_{\Theta} \ell(\theta, \tilde{\theta}) \delta_{p,\varepsilon}(\tilde{\theta}|\mathbf{t}) \nu_{\theta}(\mathbf{t}) \pi(\theta) d\theta d\mathbf{t} d\tilde{\theta} \\
 \text{s.t.} \quad & \delta_{p,\varepsilon}(\tilde{\theta}|S(\mathbf{y})) \leq e^{\varepsilon} \delta_{p,\varepsilon}(\tilde{\theta}|S(\mathbf{y}')), \text{ for each } \tilde{\theta} \in \Theta \\
 & \text{and } \mathbf{y}, \mathbf{y}' \in \mathcal{Y} \text{ s.t. } d_H(\mathbf{y}, \mathbf{y}') = 1, \\
 & \int_{\Theta} \delta_{p,\varepsilon}(\tilde{\theta}|\mathbf{t}) d\tilde{\theta} = 1, \quad \text{for each } \mathbf{t} \in \mathcal{T}, \\
 & \delta_{p,\varepsilon}(\tilde{\theta}|\mathbf{t}) \geq 0, \quad \text{for each } \mathbf{t} \in \mathcal{T}, \tilde{\theta} \in \Theta,
 \end{aligned} \tag{12.2.7}$$

where $\mathcal{P}(\mathcal{T}, \Theta)$ is the set of probability densities on Θ conditioned on $\mathcal{T}$.

From (12.2.7), it is evident that $\delta_{p,\varepsilon}^*$ corresponds to the optimal Bayes estimator (in the sense of minimizing the Bayes risk) satisfying the DP constraint (12.2.6). We denote the optimization program (12.2.7) as *Unified Bayes Private Point* (UBaPP) estimator (Figure 12.2.2) and call its solution $\delta_{p,\varepsilon}^*$ the UBaPP estimator.

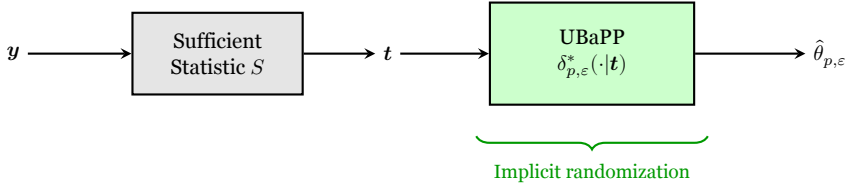


Figure 12.2.2: UBaPP estimator. Unlike the Laplace mechanism that imposes explicit randomization (Figure 12.1.1), UBaPP uses implicit randomization through the probability density $\delta_{p,\varepsilon}^*(\cdot|\mathbf{t})$.

In the following subsection, we specialize the UBaPP estimator for the case where Θ and $\mathcal{T}$ are both finite, thus arriving at a tractable convex optimization program.

12.2.1 Finite case

Consider the finite case when both $\Theta = \{\theta_1, \dots, \theta_{|\Theta|}\}$ and $\mathcal{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_{|\mathcal{T}|}\}$ are finite. Then, the randomized private Bayes estimator $\delta_{p,\varepsilon}(\cdot|\mathbf{t})$ (where $\mathbf{t} = S(\mathbf{y})$ for some $\mathbf{y} \in \mathcal{Y}$) can be described in terms of a matrix $\mathbf{P} \in \mathbb{R}^{|\Theta| \times |\mathcal{T}|}$ such that

$$\mathbf{P}_{i,j} := \mathbb{P}[\hat{\theta} = \theta_i | \mathbf{t}_j] = \delta_{p,\varepsilon}(\theta_i | \mathbf{t}_j), \tag{12.2.8}$$

for $i \in \{1, \dots, |\Theta|\}$, $j \in \{1, \dots, |\mathcal{T}|\}$, and $t_j = S(\mathbf{y}_j)$ for some $\mathbf{y}_j \in \mathcal{Y}$. Combining (12.2.8) with (12.2.7), we obtain the optimization program for the finite case as

$$\begin{aligned}
 \min_{\mathbf{P} \in \mathbb{R}^{|\Theta| \times |\mathcal{T}|}} \quad & \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^{|\Theta|} \sum_{k=1}^{|\Theta|} L(\theta_j, \theta_k) P_{k,i} \mathbb{P}[\mathbf{t}_i | \theta = \theta_j] \pi(\theta_j) \\
 \text{s.t.} \quad & P_{k,i} \leq e^\varepsilon P_{k,i'}, \text{ for all } k \in \{1, \dots, |\Theta|\} \\
 & \text{and } i, i' \in \{1, \dots, |\mathcal{T}|\} \text{ s.t. } d_H(\mathbf{y}_i, \mathbf{y}_{i'}) = 1, \\
 & \mathbf{1}^\top \mathbf{P} = \mathbf{1}^\top, \\
 & \mathbf{P} \geq 0,
 \end{aligned} \tag{12.2.9}$$

where $\pi: \Theta \rightarrow [0, 1]$ is a prior probability mass function on Θ , and $\mathbf{1} = (1, \dots, 1)^\top \in \mathbb{R}^{|\Theta|}$.

By defining $\mathbf{L} \in \mathbb{R}^{|\Theta| \times |\Theta|}$ as $L_{j,k} := L(\theta_j, \theta_k)$, $\mathbf{V} \in \mathbb{R}^{|\mathcal{T}| \times |\Theta|}$ as $V_{i,j} := \mathbb{P}[\mathbf{t}_i | \theta = \theta_j] = \nu_{\theta_j}(\mathbf{t}_i)$, and $\boldsymbol{\pi} \in \mathbb{R}^{|\Theta|}$ as $\pi_j := \pi(\theta_j)$, we can rewrite the optimization problem as

$$\begin{aligned}
 \min_{\mathbf{P} \in \mathbb{R}^{|\Theta| \times |\mathcal{T}|}} \quad & \text{tr}(\mathbf{V} \text{diag}(\boldsymbol{\pi}) \mathbf{L} \mathbf{P}) \\
 \text{s.t.} \quad & P_{k,i} \leq e^\varepsilon P_{k,i'}, \text{ for all } k \in \{1, \dots, |\Theta|\}, \\
 & \text{and } i, i' \in \{1, \dots, |\mathcal{T}|\} \text{ s.t. } d_H(\mathbf{y}_i, \mathbf{y}_{i'}) = 1, \\
 & \mathbf{1}^\top \mathbf{P} = \mathbf{1}^\top, \\
 & \mathbf{P} \geq 0.
 \end{aligned} \tag{12.2.10}$$

The minimizer of (12.2.10) gives us the UBaPP estimator $\delta_{p,\varepsilon}^*$ for the finite case, and it can be obtained using CVXPY [53]. We summarize the computation of $\delta_{p,\varepsilon}^*$ and its MSE in Algorithm 12.1.

12.3 Simulations

In this section, we compare our approach with the standard Laplace mechanism: the *mean squared error* (MSE) of the UBaPP estimator is compared with that of the LBaPP estimator $\delta_{\text{lpb},\varepsilon}$ (12.1.3), based on the Laplace mechanism, whose MSE is computed as in Algorithm 12.2.

Algorithm 12.1 UBaPP Estimator

-
- 1: **Input:** $\mathcal{Y}, \varepsilon > 0, \pi, \Theta$
 - 2: Compute $\mathbf{L}$ by evaluating $L(\theta_j, \theta_k)$ for $j, k \in \{1, \dots, |\Theta|\}$
 - 3: Compute $\mathbf{V}$ by evaluating $\mathbb{P}[t_i | \theta = \theta_j]$ for $i \in \{1, \dots, |\mathcal{T}|\}, j \in \{1, \dots, |\Theta|\}$
 - 4: Solve (12.2.10) using CVXPY and store the minimizer as $\delta_{p,\varepsilon}^*$
 - 5: **Output:** $\delta_{p,\varepsilon}^*, \text{MSE} = \text{tr}(\mathbf{V} \text{diag}(\pi) \mathbf{L} \delta_{p,\varepsilon}^*)$
-

Algorithm 12.2 MSE Evaluation for LBaPP Estimator

-
- 1: **Input:** $\mathcal{Y}, \varepsilon > 0, \pi$, number of Monte Carlo runs n_{mc}, Θ
 - 2: $S \leftarrow 0$
 - 3: **for** $i = 1, \dots, n_{\text{mc}}$ **do**
 - 4: Sample $\theta^{(i)} \sim \pi$
 - 5: Sample $\mathbf{y}^{(i)} \sim f^N(\cdot; \theta^{(i)})$
 - 6: Compute $\delta_{\mathbf{B}}^*(\mathbf{y}^{(i)})$ from (12.1.1)
 - 7: Compute $\sigma_{\delta_{\mathbf{B}}^*}$ from (12.1.4)
 - 8: $\hat{\theta}^{(i)} \leftarrow \delta_{\mathbf{B}}^*(\mathbf{y}^{(i)}) + \text{Lap}(0, \sigma_{\delta_{\mathbf{B}}^*}/\varepsilon)$
 - 9: $S \leftarrow S + (\hat{\theta}^{(i)} - \theta^{(i)})^2$
 - 10: **end for**
 - 11: **Output:** $\text{MSE} = S/n_{\text{mc}}$
-

Remark 12.1. While Algorithm 12.2 requires the explicit computation of $\delta_{\mathbf{B}}^*$ for numerical comparison, Algorithm 12.1 does not require computing such an estimator.

In particular, we demonstrate that the accuracy-privacy trade-off for our approach is better than for the Laplace mechanism, especially in the high privacy regime (i.e., for small values of ε). For this purpose, we consider the simple numerical example outlined next.

Setup: Suppose we are interested in estimating the unknown parameter $\theta \in \Theta = [0, 1]$ of a Bernoulli distribution $\text{Ber}(\theta)$ from which K independent samples are generated in $\{0, 1\}$. The simulation setup is as follows: Let π be the uniform distribution over Θ . In this case, $\mathcal{Y} \subseteq \mathbb{R}^N$, where $N = K$. Let $y_1, \dots, y_K \stackrel{i.i.d.}{\sim} \text{Ber}(\theta)$ be outcomes of Bernoulli trials, and $\mathbf{y} = (y_1, \dots, y_K)^\top$. Then, define the sufficient statistic S as $t = S(\mathbf{y}) = \sum_{i=1}^K y_i$. This means that $t \in \mathcal{T} = \{0, \dots, K\}$. Let ℓ be the squared loss, i.e., $\ell(\theta, \theta') = (\theta - \theta')^2$. For this setup, we first derive the LBaPP estimator $\delta_{\text{lpb},\varepsilon}$ and then describe the procedure to obtain the UBaPP estimator $\delta_{p,\varepsilon}^*$.

LBaPP estimator: Let $\mathbb{P}(\theta|t)$ denote the posterior distribution of θ . Then, it can be shown that $\mathbb{P}(\theta|t) \propto \theta^t (1 - \theta)^{K-t}$, which is a Beta distribution. Further,

it can be shown that

$$\delta_{\mathbf{B}}^*(\mathbf{y}) = \frac{\sum_{i=1}^K y_i + 1}{K + 2} \quad \text{and} \quad \sigma_{\delta_{\mathbf{B}}^*} = \frac{1}{K + 2}.$$

Therefore,

$$\delta_{\text{lpb},\varepsilon}(\mathbf{y}) = \frac{\sum_{i=1}^K y_i + 1}{K + 2} + \text{Lap}\left(0, \frac{1}{(K + 2)\varepsilon}\right).$$

We compute the MSE of the LBaPP estimator using Algorithm 12.2, where we take $n_{\text{mc}} = 5000$.

UBaPP estimator: To obtain the UBaPP estimator $\delta_{p,\varepsilon}^*$, we use formulation (12.2.10), for which we only need to discretize Θ , as $\mathcal{T}$ is already finite. For this purpose, we consider a grid of M equally spaced points and denote the set of such equally spaced points by $\tilde{\Theta}$. Due to the discretization, we note that there should be an additional factor $1/M$ in the objective function of (12.2.10), but this factor can be ignored since it does not depend on $\delta_{p,\varepsilon}$. Also, π is now the uniform distribution over $\tilde{\Theta}$, and hence, the j -th entry of $\text{diag}(\pi)$ is $\pi_j = 1/M$ for $j \in \{1, \dots, |\tilde{\Theta}|\}$. For this simulation setup, it is easy to see that $\mathbf{V}$ is Binomial, i.e.,

$$V_{i,j} = \mathbb{P}[t_i | \theta_j] = \binom{K}{t_i} \theta_j^{t_i} (1 - \theta_j)^{K - t_i},$$

where $\theta_j \in \tilde{\Theta}$ and $t_i \in \mathcal{T} = \{0, \dots, K\}$, for $j \in \{1, \dots, |\tilde{\Theta}|\}$ and $i \in \{0, \dots, |\mathcal{T}|\}$ respectively. For the simulation, we take $M = 5000$.

12.3.1 Results

We now study the performance of our approach to differentially private Bayes point estimation, UBaPP. In particular, we focus on (i) the effect of the privacy parameter ε on the MSE, for a fixed number of Bernoulli trials, and (ii) the effect of the number of Bernoulli trials K on the MSE, for a fixed ε .

To analyze the MSE of these two estimators for different values of ε , we consider two ranges of ε , classified as “high privacy regime” and “moderate-low privacy regime.” The high privacy regime corresponds to low values of ε , i.e., $\varepsilon \in \{0.001, 0.002, \dots, 0.010\}$, while the moderate-low privacy regime corresponds to high values of ε , i.e., $\varepsilon \in \{1, 1.5, \dots, 5\}$. The resulting plots of MSE vs. ε for the LBaPP and UBaPP estimators in the high and moderate-low privacy regimes are shown in Figures 12.3.3 and 12.3.4, respectively.

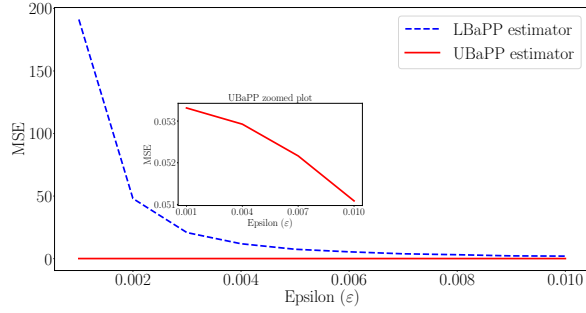


Figure 12.3.3: Plot of MSE vs. ε in the high privacy regime.

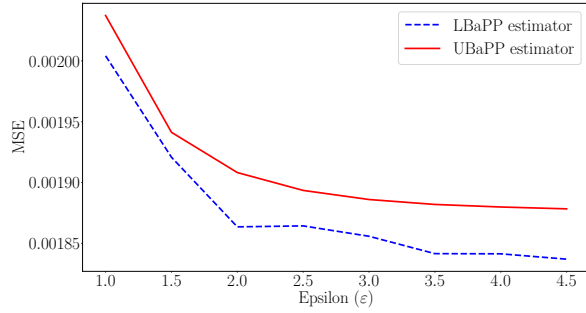


Figure 12.3.4: Plot of MSE vs. ε in the moderate-low privacy regime.

Finally, to understand the randomization induced by the UBaPP estimator, we plot in Figures 12.3.5–12.3.7 heat maps of the UBaPP estimator $\delta_{p,\varepsilon}^*$ for different values of ε .

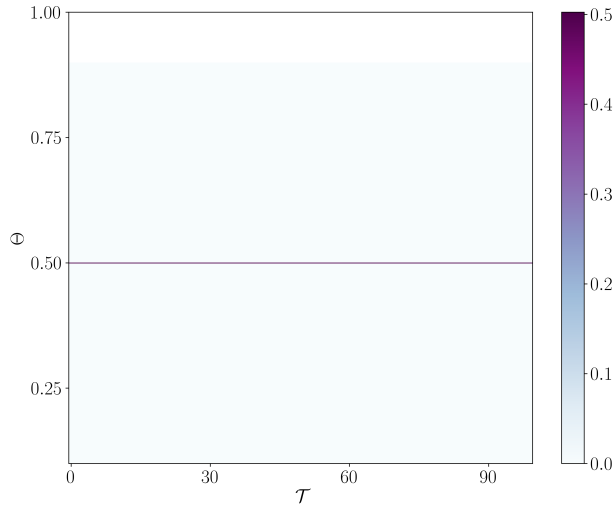


Figure 12.3.5: Heat map of the UBaPP estimator $\delta_{p,\epsilon}^*$ for $\epsilon = 0.0001$.

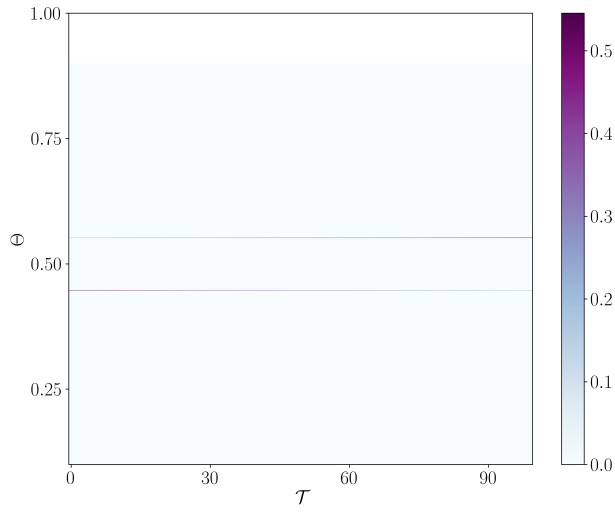


Figure 12.3.6: Heat map of the UBaPP estimator $\delta_{p,\epsilon}^*$ for $\epsilon = 0.01$.

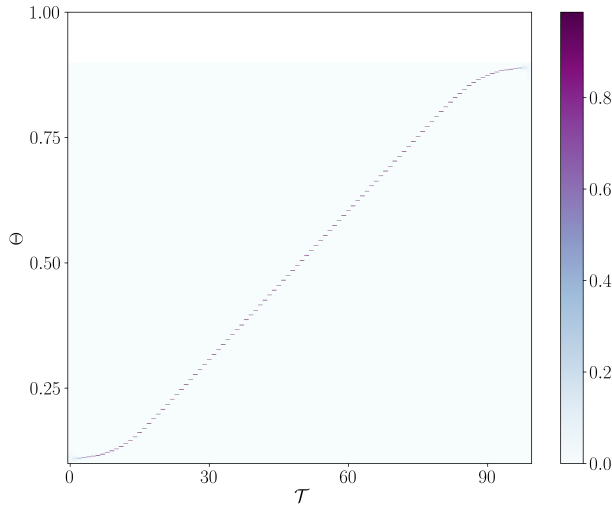


Figure 12.3.7: Heat map of the UBaPP estimator $\delta_{p,\varepsilon}^*$ for $\varepsilon = 5$.

Next, to study the effect of the number of Bernoulli trials on the MSE of both estimators, we plot the MSE of LBaPP and UBaPP for different values of K . We consider two different values of ε , namely $\varepsilon = 0.001$ and $\varepsilon = 5$, corresponding to the high privacy and moderate-low privacy regimes, respectively. The corresponding plots are shown in Figures 12.3.8 and 12.3.9, respectively.

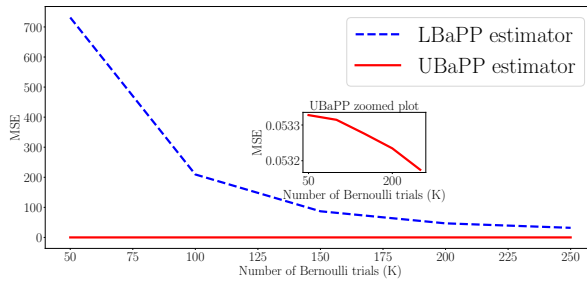


Figure 12.3.8: Plot of MSE vs. number of Bernoulli trials K in the high privacy regime ($\varepsilon = 0.001$).

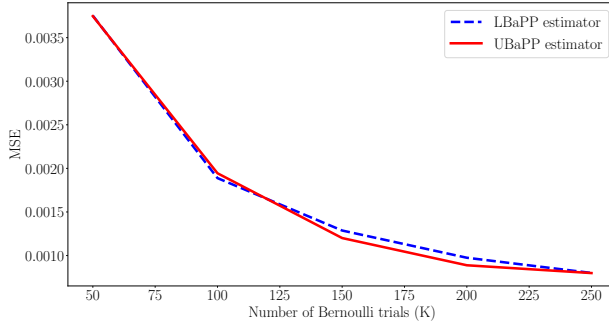


Figure 12.3.9: Plot of MSE vs. number of Bernoulli trials K in the moderate-low privacy regime ($\varepsilon = 5$).

12.3.2 Discussion

We observe from Figure 12.3.3 that UBaPP provides high accuracy (i.e., low MSE) in the high privacy regime, when compared to LBaPP. Also, Figure 12.3.4 shows that in the moderate-low privacy regime, the accuracy of UBaPP is very similar to that of LBaPP: even though the MSE of UBaPP is higher than that of LBaPP, the difference is only of order approximately 10^{-3} . This difference is due to the discretization of the parameter space Θ used in the implementation of UBaPP, whereas the computation of the LBaPP estimates is exact (i.e., no discretization is employed).

Regarding the dependence on the number of trials K , from Figure 12.3.8 it is clear that, in the high privacy regime, the MSE of UBaPP is less dependent on K than the MSE of LBaPP. This means that, for low values of K , UBaPP has significantly higher accuracy than LBaPP. On the other hand, according to Figure 12.3.9, the accuracy of LBaPP and UBaPP is similar in the moderate-low privacy regime.

To understand the difference between our approach, UBaPP, and the Laplace mechanism LBaPP, notice from Figure 12.3.5 that, for low values of ε , UBaPP outputs a deterministic estimate around $\theta = 0.5$ (which corresponds to the mean of the prior distribution on θ) for every $t \in \mathcal{T}$; since low values of ε imply high privacy, no valid inference about the observations $\mathbf{y}$ can be made based on the estimate of θ , so the estimator becomes independent of the data by outputting the same deterministic estimate for all $t \in \mathcal{T}$. Also, as seen in Figures 12.3.5–12.3.7, when ε is increased (which implies a shift from high to moderate levels of privacy), UBaPP introduces some level of randomization, because for a given value of $t \in \mathcal{T}$ the probability distribution $\delta_{p,\varepsilon}^*$ is not concentrated at a single value of $\theta \in \Theta$. Finally, as ε is further increased to very low privacy levels, Figure 12.3.7 shows that UBaPP becomes deterministic again, as it tends to the standard (non-private) Bayes point estimator of θ , which is known to be deterministic for convex loss functions [117]. Also, note from Fig-

ure 12.3.7 that the deterministic estimate varies with $t \in \mathcal{T}$, which suggests that the UBaPP estimator becomes strongly dependent on the data, and hence it becomes non-private.

In contrast, the Laplace mechanism adds randomization to the standard non-private Bayes point estimator of θ , with a variance that increases as $\varepsilon \rightarrow 0$. Thus, for large values of ε it coincides, like UBaPP, with the standard (non-private) Bayes point estimator of θ (shown in Figure 12.3.7). However, for very small values of ε (i.e., very high privacy), the variance of the estimator grows unbounded, which implies that its MSE tends to infinity, as shown in Figure 12.3.3, whereas the MSE of UBaPP tends to a constant as $\varepsilon \rightarrow 0$.

In conclusion, we see from the numerical study that UBaPP is more accurate than LBaPP in the high privacy regime, while for moderate-low privacy constraints both estimators yield similar performance (save for the discretization needed to implement UBaPP).

12.4 Conclusion

In this chapter, we have studied the problem of Bayes point estimation under differential privacy. We have argued that the standard approach based on the Laplace mechanism may not give accurate estimates under high privacy constraints. We have then proposed an optimal approach that combines risk minimization (minimum MSE) and differential privacy into a single convex optimization program, and specialized this approach to the case of finite parameter and observation spaces. Via a simple numerical study, we have shown that our approach yields more accurate estimates in the high privacy regime than the Laplace mechanism, and that both approaches have similar performance under low privacy constraints.

Closing Remarks

13 Summary and Future Research Directions

This thesis has developed theoretical foundations, algorithmic extensions, and applications of simulation-driven parameter estimation, alongside privacy-preserving approaches to point estimation. The thesis is structured into four parts. We now summarize the contributions of each part and outline potential avenues for future research.

13.1 Conclusion: Analysis of the TS Estimator (Part 1)

The TS estimator was chosen in this thesis over other simulation-driven approaches primarily because of its simple structure, which leads to computational advantages and facilitates the study of its statistical properties. A central contribution of Part 1 was the development of statistical frameworks for the TS estimator in Chapter 3. A central contribution of Part 1 was the development of statistical frameworks for the TS estimator in Chapter 3, which formalize what it means for a simulation-driven estimator to be *good* by yielding principled Bayes and minimax formulations of the TS approach. The observation that the minimax variant exhibited high variance in practice motivated the gradient-boosting-based reformulation in Chapter 4, which substantially improved reliability, particularly under non-i.i.d. data-generating mechanisms.

With these frameworks in place, Chapters 5 and 6 provided the theoretical justification that practitioners require: consistency and asymptotic normality ensure that the TS estimator converges to the true parameter, while finite-sample bounds characterize the estimation error for a given data budget. Together, these results place the TS estimator on a rigorous statistical footing compa-

rable to classical methods such as PEM and maximum likelihood. Finally, Chapter 7 addressed the practical question of training data quality, showing that geometry-aware sampling via Jeffreys prior can improve estimator performance over naive uniform sampling.

Despite these results, several challenges remain for future exploration, some of which are outlined below.

13.1.1 Possible future work

Several directions remain open for strengthening the theoretical foundations developed in Part 1.

Non-asymptotic guarantees beyond autonomous systems. The finite-sample analysis in Chapter 6 considers only autonomous systems, where no input is present. Extending these guarantees to input-driven systems requires translating existing non-asymptotic results for the first stage, such as finite-sample bounds for AR and ARX models [118, 119], and propagating them through the second stage. How errors in the first-stage compression affect the second-stage in terms of estimation accuracy remains an open problem.

Tightness of the finite-sample bounds. The bounds derived in Chapter 6 rely on Hoeffding’s inequality followed by a union bound, which can be conservative. Sharper results may be obtained by employing matrix concentration inequalities, such as the matrix Bernstein inequality [120]. Alternatively, PAC-Bayes bounds [121] offer a framework that can potentially yield tighter guarantees by directly addressing the dependence on the number of training samples M .

Impact of the first-stage choice on the second stage. In practice, the first-stage compressor is typically chosen from a fixed class (e.g., AR, ARX, or sample quantiles), and there is an intuitive understanding that compression reduces the effect of noise. However, a precise characterization of how the choice of first-stage compressor impacts the accuracy of the second-stage estimator is still lacking. Formalizing this relationship, for instance by quantifying how the information loss due to compression propagates through the second stage, is an important direction for future work.

Diverse sampling via Jeffreys prior. Chapter 7 demonstrated that sampling training parameters according to Jeffreys prior improves the performance of the TS estimator by reflecting the intrinsic geometry of the parameter space. However, the numerical validation was limited to a single example (the Weibull

distribution). Extending the experimental evaluation to a broader range of systems, including input-driven dynamical models, is necessary to assess the generality of the approach. Furthermore, investigating alternative geometry-aware sampling strategies beyond Jeffreys prior, and understanding their impact on the second-stage estimator, remains an open direction.

13.2 Conclusion: Fine-Tuning a Simulation-Driven Estimator (Part 2)

A pretrained TS estimator is only as reliable as the calibration set over which it was trained. Part 2 addressed the practically important scenario where the true parameters lie outside this calibration set, rendering the pretrained estimator unreliable. Chapter 8 demonstrated that a supervised fine-tuning pipeline, combining OOD detection with targeted retraining, can restore competitive accuracy without requiring a differentiable simulator. Chapter 9 showed that a conceptually simpler alternative, minimizing the discrepancy between simulated and observed trajectories, achieves comparable performance when a differentiable simulator is available. Taken together, these two chapters offer complementary fine-tuning strategies that extend the practical reach of the TS estimator beyond its original training domain.

13.2.1 Possible future work

Several directions remain open for improving the fine-tuning procedures developed in Part 2.

Noise robustness. The accuracy of both the pretrained and fine-tuned estimators depends on both stages of the TS construction. Under high process or measurement noise (low SNR), the Stage 1 compressor should yield statistics from which the parameters can be estimated irrespective of the noise level, while the Stage 2 neural network can become unreliable during training. In low SNR regimes, classical baselines may therefore outperform the current setup. Addressing this limitation requires (a) designing a noise-robust Stage 1 compressor and (b) strengthening Stage 2 via techniques such as adversarial training.

Computational overhead of supervised fine-tuning. The supervised approach in Chapter 8 incurs computation overhead from three sources: (i) computing the whitened Jacobian and solving the $d \times d$ system $G = J^\top J + \lambda I_d$ (cf. (8.3.13)), which has $\mathcal{O}(d^3)$ time complexity, (ii) generating a synthetic cloud of M_{FT} parameter samples and their corresponding features, and (iii) head-only retraining over this cloud. While this overhead is incurred only upon OOD detection, reducing it would broaden the applicability of the method, particularly for high-dimensional parameter spaces.

Theoretical guarantees for unsupervised fine-tuning. The unsupervised approach in Chapter 9 minimizes the simulation error as a proxy for the parameter estimation error. However, no formal guarantee is currently provided for why trajectory matching yields accurate parameter estimates. Establishing conditions under which minimizing the simulation error leads to consistent parameter recovery is an important open question.

Computational overhead of unsupervised fine-tuning. The unsupervised approach requires backpropagation through the simulator, which can be computationally expensive for complex or long-horizon systems. Investigating strategies to reduce this cost is a direction for future work.

13.3 Conclusion: Applications in Control — Meta-Learning Controller Tuning Rules (Part 3)

Part 3 demonstrated that the simulation-driven philosophy underlying the TS estimator extends naturally to controller design. Rather than repeating the traditional cycle of system identification followed by model-based design for each new plant, Chapters 10 and 11 showed that tuning rules for a PI controller can be meta-learned offline from synthetic data generated by a high-fidelity model structure and deployed in real time via a single forward pass. The two chapters explored complementary strategies: Chapter 10 retained an explicit identification stage within the TS framework, while Chapter 11 bypassed identification entirely by learning a direct mapping from input-output data to controller parameters using sequence models. Both approaches achieved the prescribed closed-loop performance with substantially reduced computation time compared to traditional grey-box methods, with the WaveNet-based approach in Chapter 11 exhibiting the best overall performance.

13.3.1 Possible future work

Advanced deep neural network architectures such as attention networks [122] have shown an exceptional ability to capture long-term dependencies within data and have found utility in state estimation [123]. Additionally, transformers have been explored for learning Kalman filters [124]. These architectural advancements suggest potential improvements over WaveNets in capturing extensive dependencies, thereby offering the possibility of achieving enhanced closed-loop performance through training on synthetic data generated using a DT.

A particularly important direction is the development of *safe* tuning rules. The learned tuning rules developed in Part 3 map input-output data to controller parameters, but provide no guarantees on the resulting closed-loop behavior. In safety-critical applications, it is essential that such rules also quantify uncer-

tainty in the predicted controller parameters and translate this uncertainty into control-relevant safety certificates. One avenue is to develop uncertainty-aware tuning rules that output both controller parameters and an associated confidence region. A complementary direction is to incorporate safety directly during training by formulating controller tuning as a constrained learning problem, where stability and performance constraints are enforced through constrained empirical risk minimization or scenario-based constraint handling. The goal would be to obtain tuning rules that are safe by construction across operating regimes and plant uncertainty.

13.4 Conclusion: Privacy in Point Estimation (Part 4)

Part 4 addressed a fundamentally different challenge: protecting individual privacy when estimation is performed on sensitive data. Chapter 12 introduced a unified approach that integrates Bayes risk minimization and differential privacy constraints into a single convex optimization problem, yielding an estimator that is optimal by construction with respect to the accuracy-privacy trade-off. A key finding was that the standard Laplace mechanism is not always optimal, and that the proposed approach eliminates the need to know the sensitivity of the statistic in advance. When both the parameter and observation spaces are finite, the optimization problem reduces to a tractable linear program, making the approach practical for discrete settings. Extending this framework to continuous or high-dimensional spaces remains the principal open challenge.

13.4.1 Possible future work

A significant challenge in combining estimation and DP into a unified optimization problem lies in the complexity of evaluating its objective function, which involves multiple integrals and operates in a high-dimensional (potentially infinite) space. Although our approach utilizes a Monte Carlo approximation, this method may not efficiently compute the integral over the joint observation and parameter space. To address this challenge, one potential avenue is to formulate the dual problem and explore the structure of its solution, investigating whether it lends itself to computational tractability. Another direction is the utilization of kernel mean embeddings [125] to tackle this high-dimensional optimization problem.

Bibliography

- [1] G. Casella and R. Berger. *Statistical Inference*. Chapman and Hall/CRC, 2024.
- [2] S.N. Deming, Y. Michotte, D. L. Massart, L. Kaufman, and B. G. M. Vandeginste. *Chemometrics: A textbook*. Vol. 2. Elsevier, 1988.
- [3] C. Gouriéroux and A. Monfort. *Statistics and Econometric Models*. Cambridge University Press, 1995.
- [4] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
- [5] L. Ljung. *System Identification: Theory for the User*. 2nd. Prentice-Hall, 1999.
- [6] T. Söderström and P. Stoica. *System Identification*. Prentice Hall, 1989.
- [7] D.P. Snustad and M.J. Simmons. *Principles of Genetics*. John Wiley & Sons, 2015.
- [8] U. Forssell and L. Ljung. “Closed-loop identification revisited”. In: *Automatica* 35.7 (1999), pp. 1215–1241.
- [9] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker. “Model-based reinforcement learning: A survey”. In: *Foundations and Trends in Machine Learning* 16.1 (2023), pp. 1–118.
- [10] L. Ljung. “On the consistency of prediction error identification methods”. In: *Mathematics in Science and Engineering*. Vol. 126. Elsevier, 1976, pp. 121–164.
- [11] T. B. Schön, A. Wills, and B. Ninness. “System identification of nonlinear state-space models”. In: *Automatica* 47.1 (2011), pp. 39–49.
- [12] G. Poyiadjis, A. Doucet, and S. S. Singh. “Maximum likelihood parameter estimation in general state-space models using particle methods”. In: *Proceedings of the American Statistical Association* (2005), p. 117.
- [13] A. Doucet and V. B. Tadić. “Parameter estimation in general state-space models using particle methods”. In: *Annals of the Institute of Statistical Mathematics* 55 (2003), pp. 409–422.

- [14] T. Söderström and P. Stoica. *Instrumental Variable Methods for System Identification*. Springer-Verlag, 1983.
- [15] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J.V. Pearson, D.A. Stephan, S.F. Nelson, and D.W. Craig. “Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays”. In: *PLOS Genetics* 4.8 (2008), e1000167.
- [16] C. Gouriéroux and A. Monfort. *Simulation-Based Econometric Methods*. Oxford University Press, 1996.
- [17] E. VanDerHorn and S. Mahadevan. “Digital Twin: Generalization, characterization and implementation”. In: *Decision support systems* 145 (2021), p. 113524.
- [18] A. Ghosh, M. Abdalmoaty, S. Chatterjee, and H. Hjalmarsson. “DeepBayes—An estimator for parameter estimation in stochastic nonlinear dynamical models”. In: *Automatica* 159 (2024), p. 111327.
- [19] T. Diskin, Y. C. Eldar, and A. Wiesel. “Learning to estimate without bias”. In: *IEEE Transactions on Signal Processing* 71 (2023), pp. 2162–2171.
- [20] N. Hollmann, S. Müller, K. Eggersperger, and F. Hutter. “TabPFN: A Transformer that solves small tabular classification problems in a second”. In: *The Eleventh International Conference on Learning Representations*. 2023.
- [21] D. Piga, M. Rufolo, G. Maroni, M. Mejari, and M. Forgione. “Synthetic data generation for system identification: leveraging knowledge transfer from similar systems”. In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. IEEE. 2024, pp. 6383–6388.
- [22] M. Forgione, F. Pura, and D. Piga. “From system models to class models: An in-context learning paradigm”. In: *IEEE Control Systems Letters* 7 (2023), pp. 3513–3518.
- [23] K. Cranmer, J. Brehmer, and G. Louppe. “The frontier of simulation-based inference”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062.
- [24] M. Deistler, P. J. Goncalves, and J. H. Macke. “Truncated proposals for scalable and hassle-free simulation-based inference”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23135–23149.
- [25] J. Zeghal, F. Lanusse, A. Boucaud, B. Remy, and E. Aubourg. “Neural posterior estimation with differentiable simulators”. In: *ICML 2022 Workshop on Machine Learning for Astrophysics*. 2023.
- [26] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Courier Corporation, 2012.
- [27] B. Lakshminarayanan and C. R. Rojas. “On asymptotic analysis of the two-stage approach: Towards data-driven parameter estimation”. In: *arXiv preprint arXiv:2508.18201* (2025).
- [28] *Netflix Cancels Contest After Concerns Are Raised About Privacy*. <https://www.nytimes.com/2010/03/13/technology/13netflix.html>. Accessed: 2010-03-12. 2010.
- [29] Y. Koren, R. Bell, and C. Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

- [30] I. Dinur and K. Nissim. “Revealing information while preserving privacy”. In: *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2003, pp. 202–210.
- [31] NIH. *NIH Genomic Data Sharing Policy*. <https://tinyurl.com/bdd6se45>. Accessed: 2014-08-27. 2014.
- [32] C. Dwork and A. Roth. “The algorithmic foundations of differential privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3–4 (2014), pp. 211–487.
- [33] S. Garatti and S. Bittanti. “Estimation of white-box model parameters via artificial data generation: a two-stage approach”. In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 11409–11414.
- [34] S. Garatti and S. Bittanti. “A new paradigm for parameter estimation in system modeling”. In: *International Journal of Adaptive Control and Signal Processing* 27.8 (2013), pp. 667–687.
- [35] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.
- [36] R. Durrett. *Probability: Theory and Examples*. Vol. 49. Cambridge university press, 2019.
- [37] T. Koski. *Lecture Notes: Probability and Random Processes at KTH*. 2017. URL: <https://www.math.kth.se/matstat/gru/sf2940/lectnotemat5.pdf>.
- [38] S. Resnick. *A Probability Path*. Birkhäuser, 2003.
- [39] P. Bremaud. *An Introduction to Probabilistic Modeling*. Springer, 2012.
- [40] J.N. McDonald and N.A. Weiss. *A Course in Real Analysis*. Elsevier, 2013.
- [41] T. S. Ferguson. *A Course in Large Sample Theory*. Routledge, 2017.
- [42] W. Hoeffding. “Probability inequalities for sums of bounded random variables”. In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30.
- [43] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis, 2nd Ed*. Springer, 1985.
- [44] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press, 2016.
- [45] J. H. Friedman. “Greedy function approximation: A gradient boosting machine”. In: *Annals of Statistics* (2001), pp. 1189–1232.
- [46] T. Chen and C. Guestrin. “XGBoost: A scalable tree boosting system”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.
- [47] R.W. Hamming. “Error detecting and error correcting codes”. In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160.
- [48] E. L. Lehmann and G. Casella. *Theory of Point Estimation, 2nd Ed*. Springer-Verlag, 1998.
- [49] L. Le Cam. “Sufficiency and approximate sufficiency”. In: *The Annals of Mathematical Statistics* 35.4 (1964), pp. 1419–1455.
- [50] C. Gouriéroux, A. Monfort, and E. Renault. “Indirect inference”. In: *Journal of applied econometrics* 8.S1 (1993), S85–S118.
- [51] G. C. Calafiore and M. C. Campi. “The scenario approach to robust control design”. In: *IEEE Transactions on Automatic Control* 51(5) (2006), pp. 742–753.

- [52] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [53] S. Diamond and S. Boyd. “CVXPY: A Python-embedded modeling language for convex optimization”. In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.
- [54] P. R. Tadikamalla. “Age replacement policies for Weibull failure times”. In: *IEEE Transactions on Reliability* R-29.1 (1980), pp. 88–90.
- [55] W. Li. “Incorporating aging failures in power system reliability evaluation”. In: *IEEE Transactions on Power Systems* 17.3 (2002), pp. 918–923.
- [56] J. D. Cook. *Determining distribution parameters from quantiles*. https://www.johndcook.com/quantiles_parameters.pdf. Jan. 2010.
- [57] D. Sun. “A note on noninformative priors for Weibull distributions”. In: *Journal of Statistical Planning and Inference* 61.2 (1997), pp. 319–338.
- [58] V. K. Murthy and G. B. Swartz. “Estimation of Weibull parameters from two-order statistics”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 37.1 (1975), pp. 96–102.
- [59] R. E. Schapire. “The boosting approach to machine learning: An overview”. In: *Nonlinear estimation and classification* (2003), pp. 149–171.
- [60] J. H. Friedman. “Stochastic gradient boosting”. In: *Computational Statistics & Data Analysis* 38.4 (2002), pp. 367–378.
- [61] A. N. Dorogush, V. Ershov, and A. Gulin. “CatBoost: gradient boosting with categorical features support”. In: *arXiv preprint arXiv:1810.11363* (2018).
- [62] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu. “LightGBM: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems*. 2017.
- [63] J. D. Cook. *Soft Maximum*. <https://www.johndcook.com/blog/2010/01/13/soft-maximum/>. Jan. 2010.
- [64] J. Courts, A. G. Wills, T. B. Schön, and B. Ninness. “Variational system identification for nonlinear state-space models”. In: *Automatica* 147 (2023), p. 110687.
- [65] P. E. Caines. *Linear stochastic systems*. SIAM, 2018.
- [66] A. W. Van der Vaart. *Asymptotic Statistics*. Vol. 3. Cambridge University Press, 2000.
- [67] M. Mohri and A. Rostamizadeh. “Rademacher complexity bounds for non-iid processes”. In: *Advances in neural information processing systems* 21 (2008).
- [68] M. Vidyasagar. *Learning and Generalisation: With Applications to Neural Networks*. Springer, 2013.
- [69] W. Rudin. *Principles of Mathematical Analysis, 3rd Edition*. McGraw-Hill, 1976.
- [70] T. J. Rothenberg. “Identification in parametric models”. In: *Econometrica* (1971), pp. 577–591.
- [71] P. Stoica and T. Söderström. “On non-singular information matrices and local identifiability”. In: *International Journal of Control* 36.2 (1982), pp. 323–329.
- [72] W. Rudin. *Real and Complex Analysis, 3rd Edition*. McGraw-Hill, 1987.
- [73] R. A Horn and C. R. Johnson. *Matrix Analysis, 2nd Ed*. Cambridge University Press, 2013.

- [74] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry, Revised 2nd Ed.* Acad. Press, 2003.
- [75] P. E. Valenzuela, J. Dahlin, C. R. Rojas, and T. B. Schön. “On robust input design for nonlinear dynamical models”. In: *Automatica* 77 (2017), pp. 268–278.
- [76] H. Jeffreys. “An invariant form for the prior probability in estimation problems”. In: *Proc. Royal Soc. London (Ser. A)* 186(1007) (1946), pp. 453–461.
- [77] S. Amari. *Information Geometry and its Applications*. Springer, 2016.
- [78] I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus, 2nd Ed.* Springer-Verlag, 1991.
- [79] D. Higham and P. E. Kloeden. *An Introduction to the Numerical Simulation of Stochastic Differential Equations*. SIAM, 2021.
- [80] Gareth O Roberts and Osnat Stramer. “Langevin diffusions and Metropolis-Hastings algorithms”. In: *Meth. Comp. Appl. Prob.* 4 (2002), pp. 337–357.
- [81] C. Andrieu and G. O. Roberts. “The Pseudo-Marginal Approach for Efficient Monte Carlo Computations”. In: *Ann. Stat.* 37(2) (2009), pp. 697–725.
- [82] G. O. Roberts and R. L. Tweedie. “Exponential convergence of Langevin distributions and their discrete approximations”. In: *Bernoulli* 2(3) (1996), pp. 341–363.
- [83] A. Ly, M. Marsman, J. Verhagen, R. P. P. P. Grasman, and E.-J. Wagenmakers. “A tutorial on Fisher information”. In: *J. Math. Psych.* 80 (2017), pp. 40–55.
- [84] J. Hull and A. White. “The pricing of options on assets with stochastic volatilities”. In: *J. Finance* 42(2) (1987), pp. 281–300.
- [85] J. Dahlin and F. Lindsten. “Particle filter-based Gaussian process optimisation for parameter inference”. In: *IFAC Proc. Vol.* 47(3) (2014), pp. 8675–8680.
- [86] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems* 27 (2014).
- [87] L. Ciampiconi, A. Elwood, M. Leonardi, A. Mohamed, and A. Rozza. *A survey and taxonomy of loss functions in machine learning*. 2024. arXiv: 2301.05579 [cs.LG].
- [88] A. Wald. “Tests of statistical hypotheses concerning several parameters when the number of observations is large”. In: *Transactions of the American Mathematical society* 54.3 (1943), pp. 426–482.
- [89] L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.
- [90] W. Greene. “Econometric analysis”. In: *Prentice Hall* (2003).
- [91] T. Wigren and J. Schoukens. “Three free data sets for development and benchmarking in nonlinear system identification”. In: *2013 European control conference (ECC)*. IEEE, 2013, pp. 2933–2938.
- [92] F. Dettù, S. Formentin, and S. M. Savaresi. “The Twin-in-the-Loop Approach for Vehicle Dynamics Control”. In: *IEEE/ASME Transactions on Mechatronics* (2023), pp. 1–12.
- [93] M. C. Campi, A. Lecchini, and S. M. Savaresi. “Virtual reference feedback tuning: a direct method for the design of feedback controllers”. In: *Automatica* 38.8 (2002), pp. 1337–1346.

- [94] M. A. Chowdhury, S. Al-Wahaibi, and Q. Lu. "Entropy-maximizing TD3-based reinforcement learning for adaptive PID control of dynamical systems". In: *Computers & Chemical Engineering* 178 (2023), p. 108393.
- [95] A. Lakhani, M. A. Chowdhury, and Q. Lu. "Stability-preserving automatic tuning of PID control with reinforcement learning". In: *arXiv preprint arXiv:2112.15187* (2021).
- [96] W. J. Shipman and L. C. Coetzee. "Reinforcement Learning and Deep Neural Networks for PI Controller Tuning". In: *IFAC-PapersOnLine* 52.14 (2019).
- [97] P. K. Kumar and K. P. Detroja. "Design of Reinforcement Learning based PI controller for nonlinear Multivariable System". In: *2023 European Control Conference (ECC)*. 2023, pp. 1–6.
- [98] J. Hall, C. E. Rasmussen, and J. Maciejowski. "Reinforcement learning with reference tracking control in continuous state spaces". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 2011.
- [99] A. Gupta, A. Pacchiano, Y. Zhai, S. Kakade, and S. Levine. "Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15281–15295.
- [100] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn. "Digital Twin in manufacturing: A categorical literature review and classification". In: *IFAC-PapersOnline* 51.11 (2018), pp. 1016–1022.
- [101] M. Grieves and J. Vickers. "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems". In: *Transdisciplinary perspectives on complex systems: New findings and approaches* (2017), pp. 85–113.
- [102] S. Formentin, K. Van Heusden, and A. Karimi. "A comparison of model-based and data-driven controller tuning". In: *International Journal of Adaptive Control and Signal Processing* 28.10 (2014), pp. 882–897.
- [103] A. Lucchini, S. Formentin, M. Corno, D. Piga, and S. M. Savaresi. "Torque vectoring for high-performance electric vehicles: an efficient MPC calibration". In: *IEEE Control Systems Letters* 4.3 (2020), pp. 725–730.
- [104] A. Lucchini, F. P. Azza, M. Corno, S. Formentin, and S. M. Savaresi. "Design and Implementation of a MPC-based Rear-Wheel Steering Controller for Sports Cars". In: *2021 IEEE Conference on Control Technology and Applications (CCTA)*. 2021.
- [105] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelmann, and J. C. Gerdes. "Neural network vehicle models for high-performance automated driving". In: *Science robotics* 4.28 (2019), eaaw1975.
- [106] J. S. Loeb, D. A. Guenther, H. F. Chen, and J. R. Ellis. "Lateral stiffness, cornering stiffness and relaxation length of the pneumatic tire". In: *SAE transactions* (1990), pp. 147–155.
- [107] M. Corno, G. Panzan, F. Roselli, M. Giorelli, D. Azzolini, and S. M. Savaresi. "An LPV approach to autonomous vehicle path tracking in the presence of steering actuation nonlinearities". In: *IEEE Transactions on Control Systems Technology* 29.4 (2020), pp. 1766–1774.
- [108] C. E. Beal and J. C. Gerdes. "Model predictive control for vehicle stabilization at the limits of handling". In: *IEEE Transactions on Control Systems Technology* 21.4 (2012), pp. 1258–1269.

- [109] R. Rajamani. “Electronic Stability Control”. In: *Vehicle Dynamics and Control*. Springer, 2012, pp. 201–240.
- [110] R. Shwartz-Ziv and A. Armon. “Tabular data: Deep learning is not all you need”. In: *Information Fusion* 81 (2022), pp. 84–90.
- [111] *MATLAB Deep Learning Toolbox*. <https://se.mathworks.com/products/deep-learning.html>. Accessed: October 4th 2023.
- [112] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).
- [113] D. Linaro, F. Bizzarri, D. Del Giudice, C. Pisani, G. M. Giannuzzi, S. Grillo, and A. M. Brambilla. “Continuous estimation of power system inertia using convolutional neural networks”. In: *Nature Communications* 14.1 (2023), p. 4440.
- [114] R. Karlsson and G. Hendeby. “Speed estimation from vibrations using a deep learning CNN approach”. In: *IEEE Sensors Letters* 5.3 (2021), pp. 1–4.
- [115] J. Wu. “Introduction to convolutional neural networks”. In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.
- [116] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [117] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis, 2nd Ed.* Springer-Verlag, 1985.
- [118] R. A. González and C. R. Rojas. “A finite-sample deviation bound for stable autoregressive processes”. In: *Learning for Dynamics and Control*. PMLR, 2020, pp. 191–200.
- [119] A. Alomar, M. Dahleh, S. Mann, and D. Shah. “SAMoSSA: Multivariate singular spectrum analysis with stochastic autoregressive noise”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 8442–8486.
- [120] J. A. Tropp. “An introduction to matrix concentration inequalities”. In: *Foundations and Trends® in Machine Learning* 8.1-2 (2015), pp. 1–230.
- [121] P. Alquier. “User-friendly introduction to PAC-Bayes bounds”. In: *Foundations and Trends® in Machine Learning* 17.2 (2024), pp. 174–303.
- [122] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in Neural Information Processing Systems* 30 (2017).
- [123] R. Busetto, V. Breschi, M. Forgione, D. Piga, and S. Formentin. “In-context learning of state estimators”. In: *IFAC-PapersOnLine* 58.15 (2024), pp. 145–150.
- [124] G. Goel and P. Bartlett. “Can a Transformer represent a Kalman filter?” In: *6th Annual Learning for Dynamics & Control Conference*. 2024, pp. 1502–1512.
- [125] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. “Kernel Mean Embedding of Distributions: A Review and Beyond”. In: *Foundations and Trends® in Machine Learning* 10.1-2 (2017), pp. 1–141.

