Calibration of complex system models for the construction of the digital twin of the autonomous vehicle

Clara Carlier

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAG004

Thèse de doctorat

ENSAE

IP PARIS

Calibration of complex system models for the construction of the digital twin of the autonomous vehicle

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École Nationale de la Statistique et de l'Administration Économique

École doctorale n°574 École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat: Mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 30 septembre 2024, par

CLARA CARLIER

Composition du Jury :

Nicolas Chopin Professeur, CREST-ENSAE	Président du jury
Nicolas Bousquet Professeur associé, Sorbonne Université	Rapporteur
Pierre Pudlo Professeur, Aix-Marseille Université (I2M)	Rapporteur
Pierre Gaillard Chargé de Recherche, INRIA Grenoble (THOTH)	Examinateur
Christine Keribin Maîtresse de Conférences HDR, Université Paris-Saclay	Examinatrice
Matthieu Lerasle Professeur, CREST-ENSAE	Directeur de thèse
Arnaud Franju Ingénieur de Recherche, Groupe Renault (DEA-TDV)	Co-encadrant, Invité

Remerciements

Je tiens tout d'abord à remercier Matthieu sans qui cette thèse n'aurait littéralement jamais existé. Merci d'avoir sauvé mon M2 en me prenant en stage suite à l'annulation du mien, à une époque trouble et compliquée qu'était celle de la covid et des confinements. Merci pour ta bonne humeur et bienveillance, ton aide et tous tes conseils avisés.

Merci à Loïc qui m'a fait confiance en me recrutant, qui a créé et constitué le sujet de thèse et qui a été présent durant mon stage. Merci à Mathias d'avoir pris la relève de mon encadrement au pied levé suite au départ de Loïc et d'avoir pris le temps pour moi. Enfin, je tiens à remercier chaleureusement Arnaud, qui a également pris la relève de mon encadrement au pied levé suite au départ de Mathias. Tu m'as prouvé que je ne faisais pas fuir tous mes tuteurs industriels en restant à mes côtés jusqu'au bout et en t'investissant énormément. Merci pour tes qualités humaines et le partage de ton savoir (souvent autour de *bons* sandwichs !).

Je tiens à remercier sincèrement les membres de mon jury. Merci Pierre d'avoir accepté de rapporter ma thèse et d'avoir fait en sorte de vous libérer pour la soutenance, malgré votre mois de septembre bien chargé. Merci Nicolas, c'est un honneur pour moi que ma thèse soit la première que vous rapportiez. Je vous remercie particulièrement pour ce rapport détaillé et l'implication dont vous avez fait preuve. Merci Pierre d'avoir répondu présent pour faire partie de mon jury et de vous déplacer pour la présentation. Merci Christine, merci pour toutes ces années. Vous m'avez toujours soutenu et je n'en serais pas là aujourd'hui sans vous. C'est une grande fierté pour moi de vous avoir à mes côtés pour conclure ma thèse. Enfin, merci Nicolas pour ta grande gentillesse, tous tes précieux conseils, ces conversations très intéressantes et nos parties de ping-pong endiablées (surtout à cause du vent !). Je te suis reconnaissante de pouvoir compter sur toi, encore une fois, dans cette étape finale.

Merci au groupe Renault d'avoir permis la création de cette thèse. Merci à Didier Wautier de m'avoir accueilli dans son service. Merci à Baptiste pour son super management. Merci à mes collègues qui ont rendu cette aventure plus humaine, Laurence, Pascal, Matthias,

Hoda, Guillaume, Adda, Tarek, Delphine, Alexandre, Palvin, Marc, Mohamed-Amine et Florent.

Merci à toutes les personnes travaillant au CREST à faire de ce laboratoire ce qu'il est. Merci aux administratifs pour leur gentillesse et efficacité, Leyla, Fanda, Nawel et Édith. Merci aux permanents d'être présents et de prendre le temps de nous aider et conseiller, Arnak, Anna, Azadeh, Cristina, Guillaume et Jaouad. Merci Victor-Emmanuel pour tout le temps que tu investis en tant que responsable de l'EDMH. Merci aux doctorants, postdoctorants et stagiaires, présents et passés, Nayel, Younès, Maria, Francesca, Jordan, Omar, Adrien, Flore, Julien, Amir, ... Je ne me risque pas à tous vous citer, ça me préservera d'un oubli maladroit.
Je tiens à remercier plus particulièrement mes colocataires du bureau 3010-3012 (désormais délocalisé) avec qui nous avons passé de supers moments scientifiques mais aussi (et surtout) humains. Hugo, Étienne, Nina, Clémentine, Sirine et Marius, merci ! J'aurais difficilement trouvé la motivation si je ne vous savais pas présents !

Je tiens à remercier mes amis qui ont su me montrer tout leur soutien et m'ont toujours encouragée. Bien que les mathématiques soient un mystère pour eux, notre amitié est, elle, bien réelle ! Merci Julia pour ton soutien infaillible, merci de me supporter. Merci Ambre, Aurore, Célin, Émelin, Juliette, Justin, Mathilde, Réann.

Merci à ma famille qui a toujours cru en moi et été très fière. *Gracias abuela, Carol y Kariniti.* Merci à ma grand-mère de prendre soin de moi et d'être une oreille si attentive. Merci à mes parents de m'avoir tant appris et si bien préparée au monde des "grands". Merci maman pour tes conseils éclairés en lesquels j'ai toute confiance. Merci papa, te rendre fier et heureux fait de moi la plus heureuse !

Enfin, pour terminer, je tiens à remercier mon cocon familial, un endroit sans lequel je n'aurais pas gardé la tête hors de l'eau ! Mes petits chats Rin et Tobi, mon chien Shoto et bien sûr, Paul, merci pour ton soutien à toute épreuve, merci pour ta joie de vivre et le soleil que tu amènes dans ma vie.

Abstract

The validation of autonomous vehicles and driving aids can no longer rely exclusively on real-life tests due to the diversity of use cases, systems involved, and reliability levels required. Digital simulation is emerging as a promising solution to complement these tests. However, for simulation to be usable in a certification framework, it is crucial to demonstrate its reliability and establish sufficient correlation with on-track results. This thesis addresses the issue of inverse problems, which involve deducing input parameters from real system observations. The objective is to identify parameters enabling simulations to reproduce real observations during testing.

The calibration step, or parameter inference, generally requires numerous simulations, which imposes significant computational costs. This thesis proposes developing a surrogate model that mimics the behavior of the original simulation while significantly reducing computational costs. This model represents a solution to the so-called direct problem, which involves predicting output results for specific inputs.

The overarching objective of this thesis is to develop a range of statistical methodologies to address these two problems, direct and inverse. It is particularly important to design approaches that can be automatically adapted to different contexts to facilitate the generalization of these methods to a wide range of situations.

Renault's digital simulator generated all the data used in this study. The manuscript is structured into four main parts, which are described below.

1. The initial chapter introduces the context and various objectives of this thesis. It also delineates the statistical tools fundamental to comprehending the manuscript. The metrics are comprehensively examined, and the corresponding outcomes are presented.

2. The second chapter is dedicated to the direct problem and the construction of the surrogate model, employing a range of state-of-the-art statistical learning techniques.

3. The third chapter presents a theoretical study of time series prediction using expert aggregation, the method employed in the previous chapter. A novel framework is constructed, considering time series as functions of time. As such, theoretical guarantees are not contingent on time discretization.

4. The fourth and final chapter addresses the inverse problem, considering a range of noise assumptions, from mild to highly restrictive. It employs optimization and sequential Monte Carlo methods to tackle these problems.

Résumé

La validation des véhicules autonomes et des aides à la conduite ne peut plus reposer exclusivement sur des essais réels, en raison de la diversité des cas d'usage, des systèmes impliqués et des niveaux de fiabilité requis. La simulation numérique émerge comme une solution prometteuse pour compléter ces tests. Toutefois, pour que la simulation soit utilisable dans un cadre de certification, il est crucial de démontrer sa fiabilité et d'établir une corrélation suffisante avec les résultats obtenus sur piste. Cette thèse se concentre sur les problèmes dits inverses : à partir d'observations réelles du système, il faut déduire les paramètres d'entrée qui ont généré ces observations. L'objectif est d'identifier les paramètres qui permettront aux simulations de reproduire au mieux les observations réelles lors des essais.

Cette étape de calibration de modèle, ou inférence des paramètres, requiert généralement de nombreuses simulations, ce qui impose des côuts computationnels importants. Pour surmonter cette contrainte, cette thèse propose l'élaboration d'un modèle de substitution qui imite le comportement de la simulation originale tout en réduisant significativement les coûts computationnels. Ce modèle revient à résoudre ce que l'on appelle le problème direct, qui consiste à prédire les résultats de sortie pour des entrées spécifiques.

L'objectif global de cette thèse est de développer diverses méthodologies statistiques pour adresser ces deux problèmes, direct et inverse. Il est particulièrement crucial de concevoir des approches qui s'ajustent automatiquement à chaque contexte, afin de faciliter la généralisation de ces méthodes à un large éventail de situations.

Toutes les données utilisées ont été générées par le simulateur numérique de Renault. Le manuscrit est structuré en quatre parties principales décrites ci-après.

1. Le premier chapitre présente le contexte et les objectifs variés de cette thèse. Il expose également les outils statistiques essentiels à la compréhension du manuscrit. Une analyse détaillée des métriques utilisées est réalisée, et les résultats correspondants sont exposés.

2. Le deuxième chapitre se consacre à la résolution du problème direct et à la construction du modèle de substitution, en exploitant diverses méthodes issues de l'état de l'art de l'apprentissage statistique et des séries temporelles.

3. Le troisième chapitre propose une étude théorique de la prédiction de séries temporelles par aggrégation d'experts, méthode employée dans le chapitre précédent. Un tout nouveau cadre est construit considérant les séries temporelles comme des fonctions du temps. Ainsi les garanties théoriques ne dépendent pas de la discrétisation du temps.

4. Le quatrième et dernier chapitre traite de la résolution du problème inverse, en considérant diverses hypothèses sur le bruit, allant de peu à très contraignantes. Il met en œuvre des méthodes d'optimisation et de Monte Carlo séquentiel pour aborder ces problématiques.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

General introduction

Contents

1.1 Context and objectives of the thesis

Autonomous driving (AD) and advanced driver assistance systems (ADAS) are increasingly prominent globally, as numerous automotive companies are heavily investing and achieving breakthroughs. ADAS improves road safety for all by assisting drivers with various technologies. These systems are growing more complex and providing support at multiple levels, as outlined in recent reviews [Ziebinski et al., 2017]. ADAS is divided into two primary categories: passive and active. Passive ADAS systems furnish drivers with crucial information, enabling them to make safer and more informed decisions. For instance, forward collision warning (FCW) and lane change assist (LCA) systems alert drivers to potential hazards and suggest actions but do not control the vehicle. Conversely, active ADAS systems play a more direct role in vehicle operation by autonomously making decisions to prevent accidents or enhance driving comfort. Examples include automatic emergency braking (AEB), which autonomously initiates braking to avoid collisions, and adaptive cruise control, which automatically adjusts the vehicle's speed to maintain a safe distance from other cars. Active

ADAS not only enhances safety by intervening in potentially dangerous situations but also improves driving comfort by reducing the driver's workload. Passive and active ADAS are crucial for developing safer automotive technologies and represent an essential step toward fully autonomous vehicles, aiming to significantly reduce traffic accidents and improve the driving experience.

A recent public opinion survey in the USA indicates that the use of ADAS has increased rapidly. According to survey data from 2020, 96% of new vehicles are equipped with at least one ADAS function. What's more, 58% of those surveyed said they would be interested in ADAS when buying their next vehicle, indicating growing consumer interest and acceptance. Despite this enthusiasm, users are highly skeptical, with 86% expressing doubts about their reliability. Similarly, 80% of participants expressed the wish that these systems would work more efficiently. These doubts and mistrust underline the importance of strict regulation and real reliability of these technologies, guaranteeing their proper functioning and safety and enabling consumer confidence in them to be strengthened. The increasing complexity of these systems and the integration of a growing number of sensors, as indicated by Kim et al. [2018], necessitates extensive and comprehensive testing. As ADAS evolve, potential use cases multiply, complicating national and international regulations. This expanding regulatory landscape calls for more rigorous and extensive testing for vehicle certification, as Kalra and Paddock [2016] points out.

Certification processes for ADAS and autonomous vehicles are notoriously complicated, time-consuming, incomplete, and costly due to the need for real-world, on-track testing. One of the major challenges of on-track testing is to reproduce specific environmental conditions. For example, certain weather scenarios are difficult to simulate accurately due to the absence of snow cannons or wind tunnels on test tracks. Furthermore, it is impossible to control the outside temperature. What's more, some specific scenarios present too high a risk for everyone's safety and are therefore not feasible in the real world. For example, assessing the behavior of a vehicle at high speed in a run-off-road scenario. However, as Lakomicki [2018] points out, the automotive industry is developing more optimal test methods, moving further and further away from traditional track testing. Manufacturers like Renault are turning to digital and physical simulators as complementary tools to refine the testing process. These simulators provide a safe, controlled environment in which various driving scenarios and environmental conditions can be accurately reproduced and tested. This approach not only reduces costs and test times but also improves the rigor and safety of the certification process. By integrating these innovative technologies, the automotive industry aims to improve the reliability and efficiency of vehicle testing, ultimately leading to safer, more efficient ADAS technologies.

Renault simulator. Renault's digital simulator is based on the SCANeR^{TM} studio software suite. It is developed by AVSimulation [2023] and is specially designed for automotive simulations. Its primary function is to support the development and testing of autonomous vehicles and ADAS. The simulator provides a set of essential tools to facilitate the creation of a realistic virtual environment. It enables users to define complex road environments, vehicle dynamics, traffic patterns, and variable weather conditions, providing a comprehensive framework for rigorous testing. To launch a simulation, we need to define various **input parameters** to set up the desired scenario. These parameters may include, for example, initial vehicle speed, braking efficiency, and other vehicle-specific dynamics. Once these parameters have been defined, the simulator generates **time series**, which describes the behavior over time of the vehicles involved, such as the evolution of speed and acceleration. A **scenario** is defined by a combination of input parameters and the resulting time series. This data enables precise analysis of how different parameters affect vehicle performance under the desired conditions. For further details on the data generated by these simulations, refer to Section 1.1.1, which provides a deeper insight into the structure and utility of the generated data.

Before simulations can be officially integrated into the vehicle homologation process, it is imperative to verify that the simulator operates correctly, as demonstrated in the work by Nabhan [2020]. It is essential to establish that the outputs from simulations are consistent with data obtained from real on-track tests. While existing methods can achieve this validation, they often lack automation and require extensive implementation time, making them less efficient for frequent or large-scale use. To address these challenges, we propose an automated methodology to **calibrate the simulator**. This innovative approach aims to ensure that the data generated by the simulator is sufficiently correlated with the real on-track data. Our objective is to establish a procedure that allows for the continuous assessment of simulation quality by direct comparison with on-track data, followed by necessary adjustments to enhance accuracy. This methodology simplifies the process of aligning simulated time series with real ones and thus significantly improves the reliability and usefulness of the simulator. Automating the calibration process reduces required time and labor, thereby increasing efficiency and enabling more frequent calibrations as needed. Ultimately, this leads to more accurate and trustworthy data, which is crucial for developing and certifying AD and ADAS.

The following section presents the data used to enhance understanding of the subject. These data will be used exclusively to calibrate our methodology and facilitate informed decision-making. Furthermore, we aim to develop a calibration method that is as automatic as possible, enabling it to go beyond the current framework and be generalized to various types of data. This wider applicability underlines the versatility and relevance of our approach, suggesting that our results could be useful for similar challenges in related fields.

1.1.1 Data description

In this thesis, we are interested in an automatic emergency braking (AEB) scenario involving two vehicles: the **target** vehicle, which leads, and the **ego** vehicle, which follows. Initially, both vehicles travel at predetermined constant speeds. The scenario unfolds as the target vehicle suddenly applies its brakes, prompting the ego vehicle to activate its emergency braking system. This response is crucial for avoiding a collision, and the dynamics of this interaction are key to assessing the effectiveness of the AEB system under study.

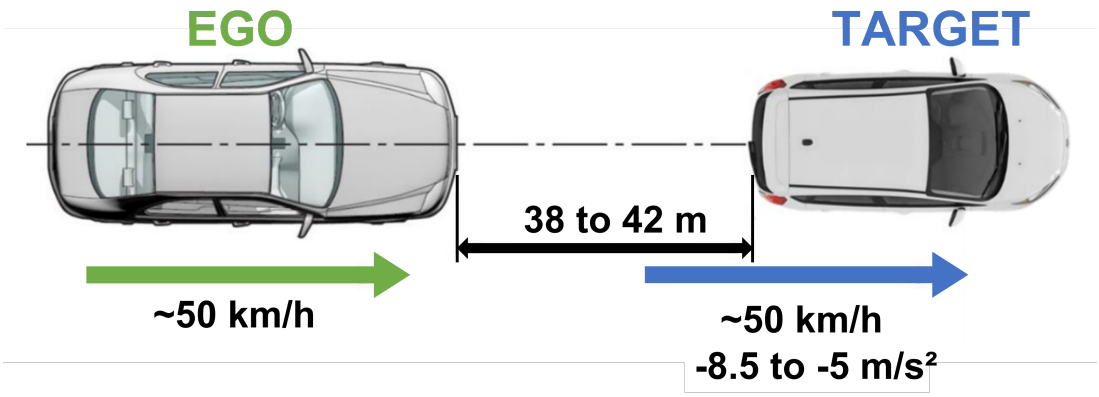


Figure 1.1: Automatic emergency braking scenario involving ego and target vehicles

A scenario is defined as a combination of two main components.

- **Input parameters.** These variables establish the necessary conditions for conducting the simulations. They include parameters that set the initial conditions of the vehicles, such as their speed and the braking force. Other parameters concern internal configurations like the reaction time of the AEB system and its braking efficiency, which directly influence the vehicle's response during critical scenarios. Environmental factors like the ground adhesion coefficient are also specified, as they significantly impact vehicle dynamics under various road conditions.

- **Output time series.** This data captures the dynamic behavior of the vehicles throughout the simulation experience. It includes detailed measurements such as the ego vehicle's speed and acceleration, the target vehicle's speed, and the changing distance between them. These outputs are crucial for analyzing the effectiveness of the AEB system and other safety features under simulated emergency conditions, providing insights into how the vehicles interact and respond to the defined inputs.

Two primary types of scenarios are used in testing and developing vehicle safety systems: simulated and real on-track scenarios. Each type serves a distinct purpose.

- **Simulated scenarios.** They are generated using the digital simulator, offering great flexibility in experimental design. Users can define various input parameters to tailor scenarios to specific test requirements. This also enables experiments to be reproduced under the same conditions, which is sometimes necessary for testing and validation. Moreover, simulated scenarios are cost-effective, eliminating many logistical and financial burdens associated with physical testing environments.
- **Real on-track scenarios.** These scenarios are physical tests performed on specialized tracks. Due to the high costs associated with setting up and running these tests, they are infrequently performed. Real track tests generally follow a well-defined context with precisely fixed parameters, called nominal parameters, to ensure that the scenarios faithfully reproduce potential real-world events. The data collected during these tests are varied time series. They are essential for validating the accuracy and reliability of simulations.

Nominal parameters. The nominal parameters define the reference scenario to be tested and are used as initial values during track tests. The main objective is to approximate these on-track reference time series through simulation. The challenge is to identify and adjust the simulator's input parameters in order to obtain simulated time series that resemble the real reference data as closely as possible. This process involves fine-tuning the input parameters, starting from nominal values and readjusting them to obtain the most realistic simulations possible.

Type	Parameter	Nominal value	Interval
Scenario	ego initial speed	50	[48, 52]
	target initial speed	50	[48, 52]
	target braking force	-6	[-7, -5]
	initial distance between the two	40	[38, 42]
MADA	front braking efficiency	1	[0.5, 1.5]
(ego)	rear braking efficiency	1	[0.5, 1.5]
	AEB latency parameter	1	[0.9, 1.1]
	braking system parameter	1	[0.9, 1.1]

Table 1.1: Input parameters definition

The nominal parameter values are given in Table 1.1. We will explore intervals defined by simulator-authorized values for this specific experiment.

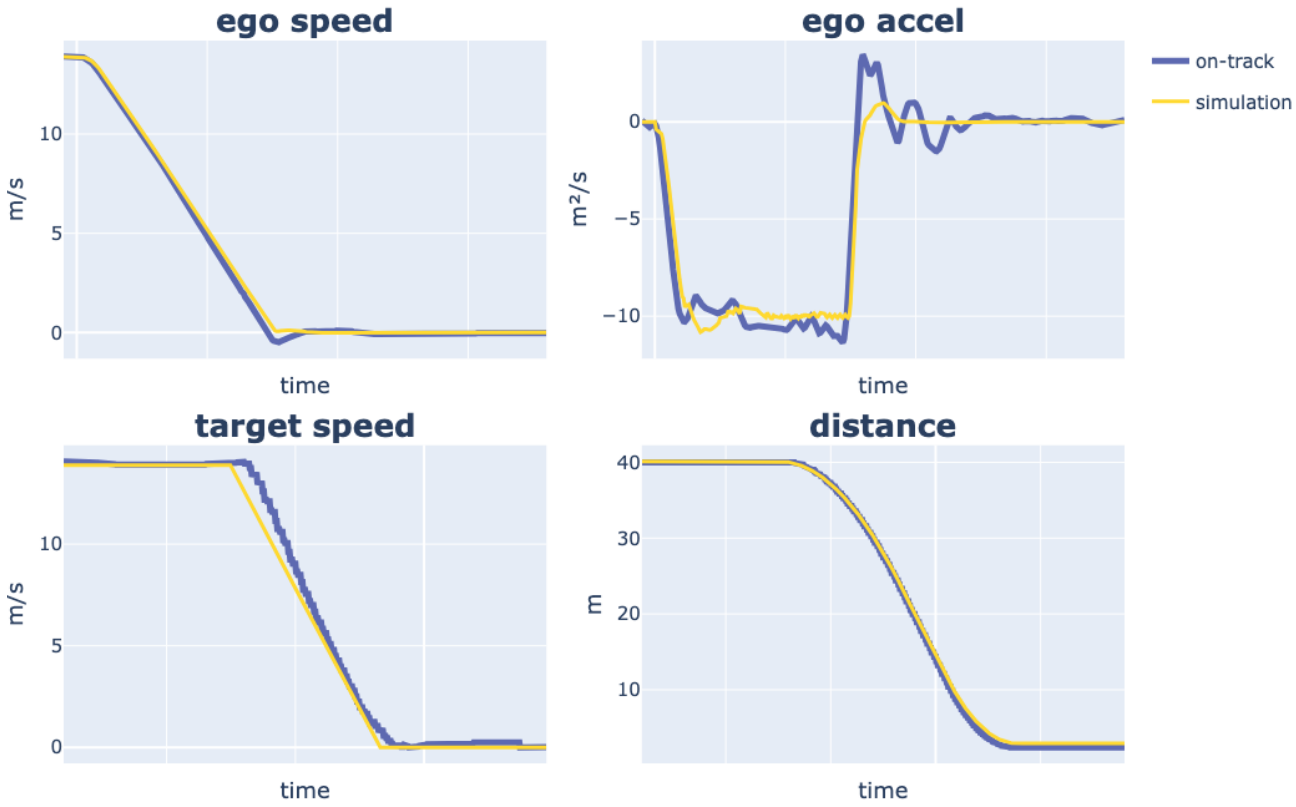


Figure 1.2: Complete reference on-track time series and an example of simulation

Figure 1.2 shows four references on-track time series and time series simulated with the nominal values. The simulated nominal time series appears similar to the reference ones. Although they are quite close, our aim is to refine the simulator parameters to obtain an even closer simulation. Figure 5.3 demonstrates the effectiveness of our approach. This figure illustrates how, by fine-tuning the parameters, it is possible to improve the simulated time series, aligning them even more closely with the values obtained on the track.

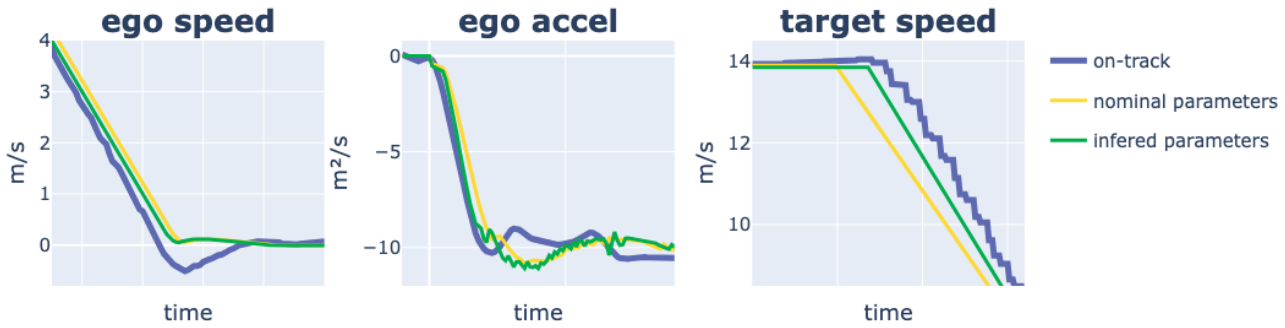


Figure 1.3: Comparison of the time series simulated with nominal parameters vs. inferred parameters

Note that we consider AEB system scenarios. In these cases, the simulator, operating with the nominal parameters, performed satisfactorily, closely matching the expected on-track outcomes. However, it is important to recognize this is not always the case. In other scenarios, the simulator can exhibit suboptimal behavior using the same nominal settings. Therefore, our efforts to optimize the parameters become even more crucial. By fine-tuning these parameters, we can significantly enhance the reliability of the simulated time series, ensuring that they accurately reflect real-world conditions and behaviors. That's why this calibration process is key to improving the validity and utility of the simulated data across various test cases.

1.1.2 Inverse problem

To **calibrate the simulator**, we aim to solve the following **inverse problem**: we have a reference experiment described by on-track time series, and we want to recover the input parameters that simulate the time series closest to the reference ones.

Inverse problem definition

In machine learning, problems are fundamentally characterized by an input-output relationship encapsulated within a model. For a specific input x, the output y is derived using the model expressed by the equation $y = f(x)$. The primary objective of machine learning is to thoroughly understand and characterize the relationship between the input x and the output y. This understanding is pivotal as it dictates how the model behaves and performs under various conditions. Depending on the nature of the problem and underlying modeling assumptions, this relationship can be either deterministic or probabilistic. A deterministic model yields the same output for a given input under consistent conditions, whereas a probabilistic model incorporates randomness, allowing for output variability even with the same input.

We encounter two fundamental types of problems represented in Figure 1.4 and defined below.

- **Forward problem.** We need to predict the output y based on a specific input x. To do this, we need to build an estimator $\widehat{f}$ that reproduces the behavior of f as closely as possible, thus enabling faithful predictions of y for any given x. The accuracy and performance of $\widehat{f}$ must be assessed and are crucial as they directly impact the reliability of the predictions made by the model.
- **Inverse problem.** In contrast to the forward problem, it focuses on estimating the input x given an observed output y. It involves inferring the possible values of x that

could have led to the observed y. Effectively, the task is to reverse-engineer the model f to determine which inputs would produce the output seen in the real world when processed through the model. This process is often complex and requires sophisticated statistical methods or algorithms to address the potential non-uniqueness and instability of the solution, especially in cases where multiple inputs might yield similar outputs.

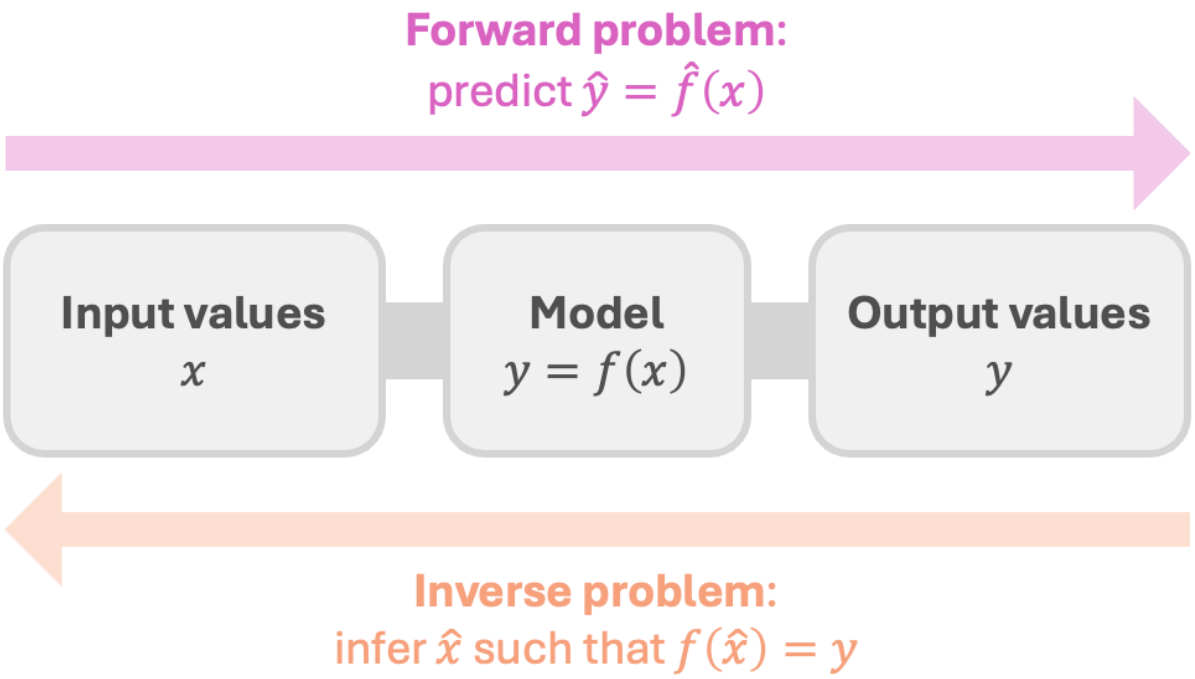


Figure 1.4: Forward and inverse problem diagram

Solving the inverse problem

The aim of solving our inverse problem is briefly outlined in Figure 1.5. We will systematically adjust the simulator's input parameters based on the discrepancies identified between the simulated outcomes and the actual on-track data.

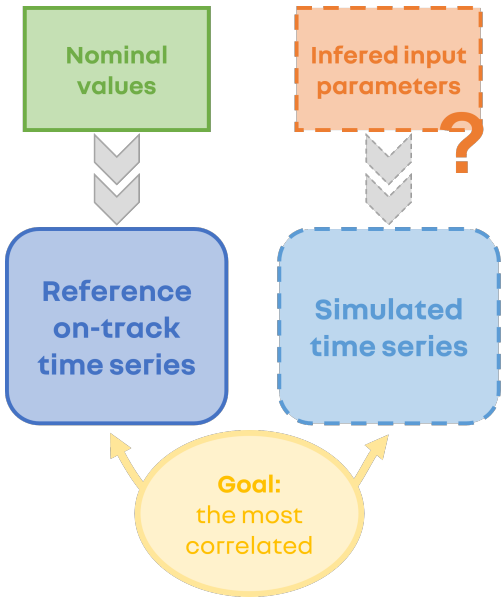


Figure 1.5: Inverse problem diagram

- **Uncertainties in nominal values.** The reference on-track time series are given with their associated input parameters called nominal values. These values are subject to **uncertainties**, particularly due to sensor inaccuracies, manufacturing tolerances, or errors in computational estimations. For those reasons, we can substantially improve the simulated time series by modifying the nominal values.

- **Parameter spaces.** The aim is to explore the parameter spaces created around the nominal values to find the optimal combination of values, i.e., the parameters that simulate the time series most correlated with the reference ones.

To solve the inverse problem, we investigate several options. We will test frequentist and Bayesian approaches.

Bayesian inference

Bayesian statistics fundamentally relies on conditional probabilities, which assess the likelihood of an event A occurring given that another event B has already occurred. It is denoted as $p(A|B)$. According to the definition, the conditional probability $p(A|B)$ can be calculated using the formula $p(A|B) = p(A \cap B)/p(B)$, provided that the probability of event B, $p(B)$, is non-zero. This formula effectively describes how the occurrence of B adjusts the probability of A and encapsulates the essence of Bayesian inference, where beliefs are updated with new evidence. Such methodologies are central to numerous applications in statistics and machine learning, enabling more informed decision-making under uncertainty.

Bayesian inference represents a fundamental approach to statistical inference, where the goal is to make probabilistic statements about unknown parameters or hypotheses based on observed data. This method contrasts with frequentist inference by incorporating prior

beliefs about the parameters before data is observed. The process begins by formulating a hypothesis about a population, then collecting data, and finally using it to update the probability of the hypothesis.

Consider two random variables θ and Y. Bayes' theorem provides a powerful link between these variables, allowing us to update our knowledge about θ after observing Y. The theorem is given by

$$\pi(\theta|Y) = \frac{p(Y|\theta)\pi_0(\theta)}{p(Y)}\,.$$

In this formula, $\pi(\theta|Y)$ is the posterior distribution, representing our updated belief about θ after observing Y. The term $p(Y|\theta)$ is the likelihood function, which measures the plausibility of observing Y given θ. The prior distribution is the $\pi_0(\theta)$, reflecting our beliefs about θ before observing any data. Finally, $p(Y)$ is the marginal likelihood, also known as the evidence, and is calculated by integrating the likelihood over all possible values of θ, weighted by the prior

$$p(Y) = \int p(Y|\theta)\pi_0(\theta)\mathrm{d}\theta\,.$$

This integral helps normalize the posterior to ensure it sums to one and thus forms a valid probability distribution.

Bayesian inference, therefore, revolves around updating our prior beliefs in light of new evidence, a process central to decision-making in conditions of uncertainty. The posterior distribution encapsulates all that is known about the variable θ after considering the new data Y and our prior knowledge, making it a powerful tool for various statistical applications.

Our primary interest lies in determining the posterior distribution $\pi(\theta|Y = y_{\text{obs}})$, which represents the probability distribution of the random variable θ given the data y_{obs} that has been observed. Once the posterior distribution is established, a common task is to estimate a single point value of the parameter θ that best represents the posterior. This estimation is typically achieved through the maximum a posteriori (MAP) criterion, which selects the value of θ that maximizes the posterior distribution. Mathematically, this is expressed as

$$\widehat{\theta}_{\text{MAP}} \in \arg\max_{\theta\in\Theta} \pi(\theta|Y = y_{\text{obs}})\,.$$

This value is particularly significant because it represents the most probable value of θ given the observed data, considering both the likelihood of θ given Y and the prior beliefs about θ.

In particular complex contexts, computing the likelihood function is impractical or intractable due to the model's complexity or the data's nature. In these cases, *likelihood-free methods*

provide a valuable alternative, allowing the estimation of the posterior distribution without explicitly calculating the likelihood. These methods are particularly useful when simulating data from the model, which is feasible, but directly assessing the likelihood is not.

One common approach within likelihood-free methods is to generate samples that approximate the posterior distribution. It is achieved by drawing a large number of samples $\theta_1, \ldots, \theta_n$ from the distribution and using these samples to construct an empirical representation of the posterior given by

$$\pi(\theta|Y = y_{\text{obs}}) \approx \frac{1}{n} \sum_{i=1}^{n} \delta_{\theta_i}(X)\,,$$

where δ_{θ_i} denotes the Dirac measure centered at θ_i. As the sample size n increases, the empirical distribution formed by these Dirac spikes converges to the true posterior distribution.

The rejection method is a fundamental likelihood-free algorithm that enables sampling from complex probability distributions when direct sampling is challenging or the likelihood is intractable. One must specify a loss function ℓ and select a suitable tolerance parameter h to implement the rejection method. The steps involved in the rejection method are as follows.

Algorithm 1 Generic rejection sampler

Initialization:

- Sample independent particles according to the prior $\theta_1, \ldots, \theta_n \sim \pi_0$

Iterations:

1: **for** $k = 0, \ldots, K$ **do**
2: Generate the data associated with particles y_i, $\forall i = 1, \ldots, n$
3: Determine accepted particles $A_k = \{i : \ell(y_i, y_{\text{obs}}) \leq h\} \subseteq \{1, \ldots, n\}$
4: Save accepted values $(\theta_i)_{i \in A_k}$
5: **Update step:** Generate the new particles $\theta_1, \ldots, \theta_n \sim \pi_0$

The tolerance parameter h plays a crucial role in this method. Setting h to a small value leads to a stricter acceptance criterion, potentially resulting in a higher rejection rate but a closer approximation to the true posterior. Conversely, a larger h value generally increases the acceptance rate but may compromise the quality of the approximation.

Choosing the appropriate loss function ℓ is critical, as it needs to be specifically tailored to the data's characteristics and the unique requirements of the problem at hand. The loss function plays a fundamental role in how a model's predictions are evaluated and refined. Depending on the distinct objectives and constraints of different segments of the analysis, various approaches to defining and minimizing the loss may be necessary.

In the context of inverse problems, the loss function might include penalties for complexity or deviations from prior knowledge or expected behavior. This is particularly important

when direct computation of likelihoods is impractical. This critical decision will be thoroughly discussed in Section 1.2.

The resolution of the inverse problem is dealt with in Chapter 4.

1.1.3 Surrogate model

A crucial stage in all likelihood-free algorithms involves generating the output y linked with the candidate inputs θ (see line 2 of the Algorithm 1). The two are linked through the simulator S: for a given set of parameters, time series are generated

$$y = S(\theta)\,,$$

where $\theta \in \mathbb{R}^p$ and $y \in \mathbb{R}^T$. Here, y concatenates several types of time series one after the other, so we have $\sum_{r=1}^{R} T_r = T$ where T_r is the size of the r-th time series. In the Data description context, we have four distinct types of time series: ego speed, ego acceleration, target speed, and distance, so $R = 4$.

In likelihood-free algorithms, using the simulator S would be the optimal solution for generating time series based on input parameters, but it requires excessive computational resources. To overcome this issue, we substitute the simulator with a pre-trained surrogate model, which mimics the simulator and predicts y for a given θ using $\widehat{S}(\theta)$.

Building such a model is one of the fundamental objectives of the machine learning framework: the supervised statistical learning to solve a forward problem. We aim to predict an output variable Y belonging to a space $\mathcal{Y}$, based on an input variable θ from a space Θ. Both Θ and $\mathcal{Y}$ are assumed to be measurable spaces, facilitating the application of probability and measure theory. A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is introduced to quantify the accuracy of predictions. This function calculates the prediction error $\ell(\widehat{y}, y)$, where $\widehat{y}$ is the predicted output and y is the true output.

The space of all potential predictors, denoted by $\mathcal{F}$, consists of all measurable functions f that map from Θ to $\mathcal{Y}$. The choice of $\mathcal{F}$ is crucial as it defines the set of all possible models that can be used to predict y from θ.

Given a joint probability distribution P over the pair (θ, Y), the risk associated with a predictor f is defined as the expected value of the loss function over this distribution

$$\mathcal{R}_P(f) = \mathbb{E}_{(\theta,Y)\sim P}\big[\ell(f(\theta), Y)\big]\,.$$

Mathematically, the risk quantifies the average performance of the predictor f across the entirety of the data distribution, providing a measure of how well f generalizes to new, unseen

data. The objective is to identify the predictor f within $\mathcal{F}$ that minimizes this risk, which is essential in developing effective predictive models.

For a given sample $(\theta_1, y_1), \ldots, (\theta_n, y_n)$ i.i.d. of distribution P on $\Theta \times \mathcal{Y}$, the goal is to construct a predictor $\widehat{f}_n : \Theta \to \mathcal{Y}$ which minimises the empirical risk

$$\widehat{f}_n \in \underset{\widehat{f} \in \mathcal{F}}{\arg\min} \left\{ \mathcal{R}_n(\widehat{f}) = \frac{1}{n} \sum_{i=1}^{n} \ell\big(\widehat{f}(\theta_i), y_i\big) \right\}$$

for a given loss ℓ. The empirical risk is an unbiased estimator of the risk $\mathcal{R}_P(\widehat{f}_n)$.

The surrogate model can be constructed using various methods with varying degrees of sophistication, like random forests, convolutional neural networks, recurrent neural networks,..., etc. We solve this forward problem in Chapter 2.

As mentioned above in Section 1.1.2, the loss function ℓ needs to be carefully chosen. This context can be slightly different, we aim to primarily focus on capturing the accuracy of predictions and the closeness of the estimated outputs to the actual values. By selecting an appropriate loss function, we ensure that the predictor not only minimizes theoretical risk but also aligns closely with practical, real-world needs and the specific characteristics of the data.

1.2 Loss functions study

We aim to define what 'two similar time series' means and provide a quantification method. The first possibility is to consider one or several values like the mean, variance, kurtosis and skewness [Mardia, 1970]. The literature also offers various metrics like the cross-correlation [Bourke, 1996] or the two classical root mean squared error (RMSE) and mean absolute error (MAE), which have been compared in several articles [Chai and Draxler, 2014, Wang and Lu, 2018, Willmott and Matsuura, 2005]. Some metrics are specific to time series, such as dynamic time warping (DTW) [Senin, 2008], soft-DTW [Cuturi and Blondel, 2017], or DILATE [Le Guen and Thome, 2019]. Another trend that emerged a few years ago is integrating the derivative of the time series into the metric calculation, such as derivative-DTW [Górecki and Łuczak, 2013, Łuczak, 2016]. We also want to test three ideas: (1) calculate the mean of the errors and include their variance, (2) consider a scaled version of the RMSE, and (3) mix RMSE and DTW losses. All these functions are described in the next section.

1.2.1 Mathematical description of the loss functions

Remark. In Section 1.2.1, y_t denotes the t-th coordinate of the time series y.

Statistics

For y, a time series of size T, the mean and the variance are given by

$$\mu_y = \frac{1}{T}\sum_{t=1}^{T} y_t\,, \quad \text{and} \quad \sigma_y^2 = \frac{1}{T}\sum_{t=1}^{T}(y_t-\mu_y)^2\,.$$

The Fisher definition of the kurtosis and the Fisher-Pearson coefficients of skewness are given by

$$k_y = \frac{\widehat{m}_4(y)}{\widehat{m}_2(y)^2}\,, \quad \text{and} \quad s_y = \frac{\widehat{m}_3(y)}{\widehat{m}_2(y)^{3/2}}\,,$$

where $\widehat{m}_k(y) = \frac{1}{T}\sum_{t=1}^{T}(y_t-\mu_y)^k$. With z an another time series of size T, we define the first studied loss

$$\ell_{\text{stat}}(y,z) = \sqrt{(\mu_y-\mu_z)^2 + (\sigma_y^2-\sigma_z^2)^2 + (k_y-k_z)^2 + (s_y-s_z)^2}\,.$$

Cross-correlation

The second studied loss is the mean of the cross-correlation defined by

$$\ell_{\text{corr}}(y,z) = \frac{1}{2T-1}\sum_{k=0}^{2T-2}\sum_{i=0}^{T-1} y_i z_{i-k+T-1}\,.$$

Classical losses

We define the third and fourth studied losses, the classical RMSE and MAE

$$\ell_{\text{RMSE}}(y,z) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t-z_t)^2}\,, \quad \text{and} \quad \ell_{\text{MAE}}(y,z) = \frac{1}{T}\sum_{t=1}^{T}|y_t-z_t|\,.$$

Adding the derivative

As we are dealing with physical time series, such as speeds, accelerations, or distances, it is appropriate to include their derivative with respect to time in the loss calculation. Therefore, we integrate the RMSE of their derivative

$$\ell_{\text{deriv}}(y,z) = \ell_{\text{RMSE}}(y,z) + \ell_{\text{RMSE}}(y',z')\,,$$

where y' stands for the discrete derivative $y'_t = y_{t+1} - y_t$.

Scaled version

As presented in Section 1.1.1, the output y is a concatenation of four types of time series

$$y = \left(y^{(1)} \mid y^{(2)} \mid \cdots \mid y^{(R)} \right),$$

where $y^{(r)}$ is of size T_r and $\sum_{r=1}^{R} T_r = T$. Each one has its mean and variance. They need to be scaled to ensure that each time series has the same weight in the loss calculation. That is why we consider a scaled version of the RMSE

$$\ell_{\text{s-RMSE}}(y, z) = \ell_{\text{RMSE}}(\widetilde{y}, \widetilde{z}),$$

where $\widetilde{y}$ denotes the scaled vector of y defined by

$$\widetilde{y}_t = \frac{(y_t - \mu_{y^{(k)}})}{\sigma_{y^{(k)}}} \text{ if } y_t \in y^{(k)}.$$

Dynamic time warping

To define the DTW and the soft-DTW, we first need to introduce the cost matrix with the absolute errors

$$\Delta(y, z) = \left(|y_i - z_j|\right)_{i,j} \in \mathbb{R}^{T \times T}.$$

The DTW is then obtained by solving

$$\ell_{\text{DTW}}(y, z) = \min_{A \in \mathcal{A}_{T,T}} \langle A, \Delta(y, z) \rangle,$$

where $\langle ., . \rangle$ is the inner product and $\mathcal{A}_{T,T} \subset \{0, 1\}^{T \times T}$ is the set of binary alignment matrices, that is paths on a $T \times T$ matrix that connect the upper-left matrix entry to the lower-right one using only $\downarrow, \rightarrow, \searrow$ moves.

The soft-DTW is very similar, but a smoothing parameter $\gamma \geq 0$ is added

$$\ell_{\text{soft-DTW}}(y, z) = \min\nolimits^{\gamma} \{ \langle A, \Delta(y, z) \rangle,\ A \in \mathcal{A}_{T,T} \},$$

where

$$\min\nolimits^{\gamma} \{a_1, \ldots, a_T\} = \begin{cases} \min_{t \leq T} a_t, & \gamma = 0, \\ -\gamma \log \sum_{t=1}^{T} \exp(-a_t/\gamma), & \gamma > 0. \end{cases}$$

For more details, please refer to Cuturi and Blondel [2017].

DILATE

The DILATE loss is defined by a combination of a shape loss and a temporal one with a weight parameter α

$$\ell_{\text{DILATE}}(y, z) = \alpha \ell_{\text{shape}}(y, z) + (1 - \alpha)\ell_{\text{temporal}}(y, z)\,,$$

where the shape loss is a γ-soft-DTW and the temporal part is derived from a time distortion index (TDI) which consists to penalized the DTW, for more details please refer to the Section 3.1 of Le Guen and Thome [2019].

Mixing RMSE and DTW

As it is done with the DILATE loss, we try to mix the RMSE and the DTW with a weight parameter α

$$\ell_\alpha(y, z) = \alpha \ell_{\text{RMSE}}(y, z) + (1 - \alpha)\ell_{\text{DTW}}(y, z)\,.$$

1.2.2 Comparing the results of the loss functions

We first simulate n scenarios denoted $y_1, \ldots, y_n$ to determine our context's most appropriate loss function. For each possible ℓ presented above, we compute the best scenario which verifies

$$\underset{y_i,\ i=1,\ldots,n}{\arg\min}\ \ell(y_i, y_\varphi)\,,$$

where y_φ is the reference on-track time series. Each loss selects a scenario y_i, and all the selected y_i are compared to determine the best one. Some losses have selected identical time series, and the corresponding equivalences are listed in Table 1.2. Additionally, the metrics ℓ_{stat} and ℓ_{corr} are directly disregarded because they don't effectively differentiate y_i's and produce inaccurate results. The set $\mathcal{Y}_\mathcal{L}$ contains a total of eight scenarios selected by the eight losses ℓ_j defined in the table below.

ℓ_1	ℓ_2	ℓ_3	$\times$	$\times$	ℓ_4	ℓ_5	ℓ_6	ℓ_7	ℓ_8
ℓ_{MAE} $\ell_{\alpha=0.9}$ $\ell_{\alpha=0.75}$	ℓ_{RMSE} $\ell_{\text{soft-DTW}}$	ℓ_{deriv} $\ell_{\alpha=0.25}$	ℓ_{stat}	ℓ_{corr}	$\ell_{\text{s-RMSE}}$	ℓ_{DTW}	ℓ_{DILATE}	$\ell_{\alpha=0.5}$	$\ell_{\alpha=0.1}$

Table 1.2: Considered losses and their equivalents

Upon examining the visualization of the selected scenarios $\mathcal{Y}_\mathcal{L}$ in Figure 1.6, it becomes apparent that the DTW and DILATE losses have chosen unsuitable time series for ego speed and ego acceleration. While the selected time series for target speed and distance are

correct, these losses can be disregarded. The differences in the remaining losses are not easily discernible. Therefore, our selection will be based on the errors obtained by each chosen time series summarized in Table 1.3.

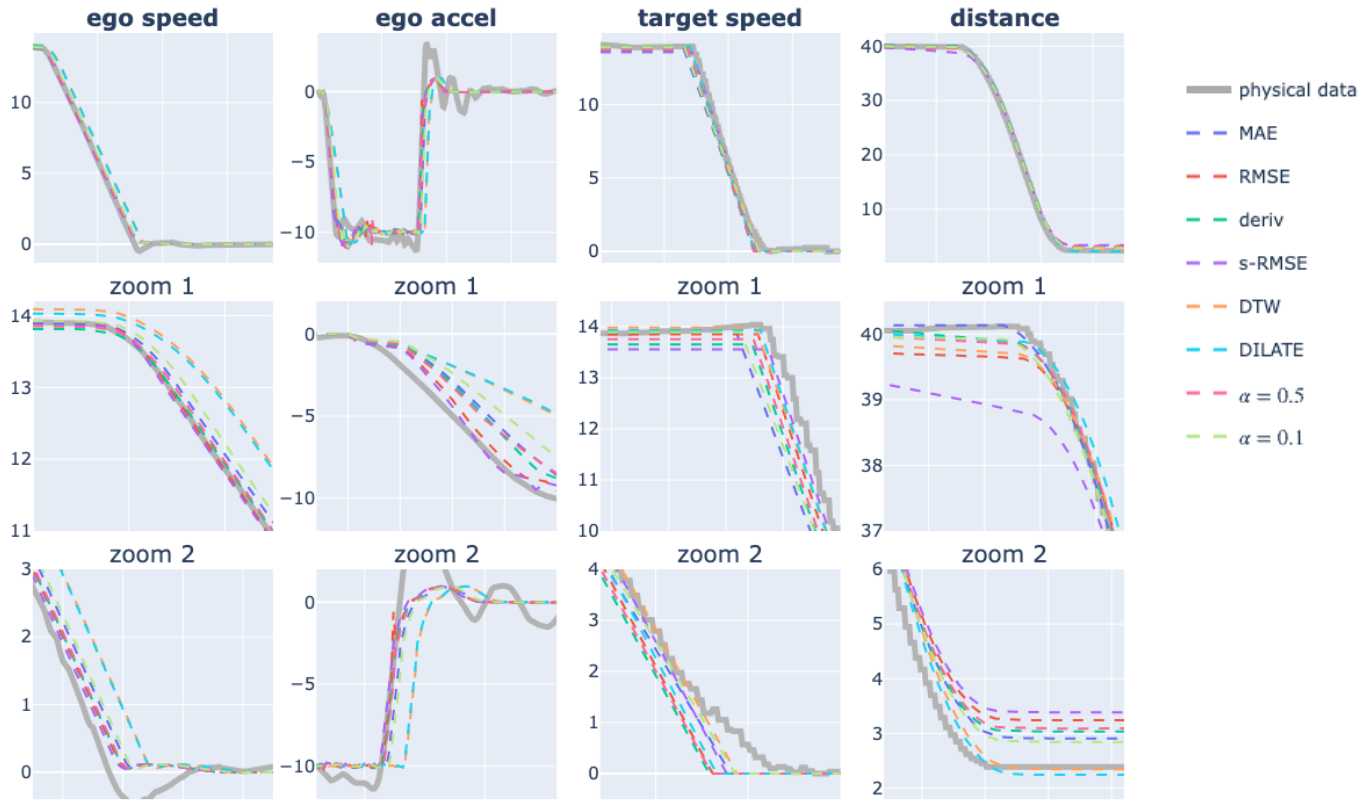


Figure 1.6: Comparison of the time series selected by the different metrics

Referring to the errors in Table 1.3, the time series selected by s-RMSE are the best for ego speed and ego acceleration according to the majority of metrics, except according to DTW and DILATE but DTW and DILATE were previously ruled out as options.

For measuring target speed, s-RMSE is the third best metric available, after DTW and DILATE, providing a unified choice for ego speed, ego acceleration, and target speed.

When considering the distance time series, the s-RMSE selects inadequate values, while the MAE performs better. The distance issue is that the most effective metrics now become the least effective. Choosing a scenario that accurately corresponds to all four time series appears complex due to the intrinsic differences between the simulated and on-track data. To address this limitation, we will use the metric that satisfies the most significant number of time series, namely the s-RMSE.

ego speed									
		error calculated with							
		MAE	RMSE	deriv	s-RMSE	DTW	DILATE	$\alpha = 0.5$	$\alpha = 0.1$
output selected with	MAE	0.1711	0.2447	0.2605	0.0383	0.0264	4.2007	0.1355	0.0482
	RMSE	0.0981	0.1594	0.1755	0.0277	**0.0255**	4.3066	0.0924	0.0389
	deriv	0.1128	0.1699	0.1858	0.0290	0.0280	4.3594	0.0989	0.0422
	s-RMSE	**0.0910**	**0.1402**	**0.1565**	**0.0251**	0.0264	4.3697	**0.0833**	**0.0378**
	DTW	0.4591	0.6703	0.6934	0.0881	0.0311	4.1637	0.3507	0.0950
	DILATE	0.4400	0.6462	0.6691	0.0871	0.0290	**3.9953**	0.3376	0.0907
	$\alpha = 0.5$	0.1256	0.1844	0.2002	0.0307	0.0271	4.2887	0.1057	0.0428
	$\alpha = 0.1$	0.2154	0.3023	0.3186	0.0441	0.0269	4.1465	0.1646	0.0544

ego acceleration									
		error calculated with							
		MAE	RMSE	deriv	s-RMSE	DTW	DILATE	$\alpha = 0.5$	$\alpha = 0.1$
output selected with	MAE	0.6752	1.0346	1.2144	0.2058	0.1845	62.2643	0.6095	0.2695
	RMSE	0.5686	0.8352	1.1405	0.1646	0.1796	62.9846	0.5074	0.2451
	deriv	0.6057	0.8793	1.0561	0.1731	0.1767	60.9140	0.5280	0.2469
	s-RMSE	**0.5422**	**0.7975**	**0.9925**	**0.1569**	0.1811	61.4509	**0.4893**	**0.2428**
	DTW	1.0319	2.1159	2.4472	0.4310	0.1794	60.3540	1.1476	0.3730
	DILATE	1.0266	2.0983	2.4202	0.4276	0.1758	**59.8188**	1.1370	0.3680
	$\alpha = 0.5$	0.6190	0.9065	1.1216	0.1795	**0.1757**	60.7977	0.5411	0.2488
	$\alpha = 0.1$	0.7469	1.2144	1.4244	0.2440	0.1787	60.5676	0.6966	0.2823

target speed									
		error calculated with							
		MAE	RMSE	deriv	s-RMSE	DTW	DILATE	$\alpha = 0.5$	$\alpha = 0.1$
output selected with	MAE	0.3633	0.6018	0.6700	0.0775	0.0342	9.8578	0.3180	0.0910
	RMSE	0.2921	0.4161	0.4838	0.0470	0.0375	11.6069	0.2268	**0.0754**
	deriv	0.4524	0.5938	0.6614	0.0612	0.0752	22.3870	0.3345	0.1271
	s-RMSE	**0.2914**	**0.3440**	**0.4113**	**0.0260**	0.0938	30.8822	**0.2189**	0.1188
	DTW	**0.1778**	**0.2687**	**0.3364**	0.0376	**0.0289**	14.3354	**0.1488**	**0.0528**
	DILATE	0.2309	0.3354	0.4031	0.0396	0.0308	9.9898	0.1831	0.0612
	$\alpha = 0.5$	0.3816	0.5199	0.5878	0.0567	0.0567	17.9132	0.2883	0.1031
	$\alpha = 0.1$	0.3047	0.5092	0.5772	0.0661	**0.0334**	**9.4996**	0.2713	0.0810

distance									
		error calculated with							
		MAE	RMSE	deriv	s-RMSE	DTW	DILATE	$\alpha = 0.5$	$\alpha = 0.1$
output selected with	MAE	**0.3269**	**0.4089**	**0.5261**	**0.0079**	0.1009	39.0608	**0.2549**	0.1317
	RMSE	0.5509	0.6159	0.7329	0.0134	0.1779	108.4473	0.3969	0.2217
	deriv	0.3683	0.4503	0.5670	0.0096	0.1211	54.6183	0.2857	0.1540
	s-RMSE	**0.7529**	**0.8202**	**0.9372**	0.0187	**0.2021**	**142.6890**	**0.5112**	**0.2639**
	DTW	0.3816	0.5270	0.6441	**0.0284**	**0.0412**	13.6892	0.2841	**0.0898**
	DILATE	0.3477	0.4779	0.5948	0.0253	0.0478	**9.1476**	0.2629	0.0908
	$\alpha = 0.5$	0.4147	0.5055	0.6224	0.0100	0.1212	65.3123	0.3133	0.1596
	$\alpha = 0.1$	0.3446	0.4174	0.5341	0.0109	0.0942	36.4045	0.2558	0.1265

Table 1.3: Losses results

The **bolded errors** are the best in each column.

The **bolded errors in teal colour** are the third best in each column.

The **bolded errors in purple colour** are the worst in each column.

1.3 Contributions

Chapter 2. Construction of the surrogate model

Surrogate models are extensively utilized across various domains to enhance efficiency and accuracy, employing methods such as polynomial chaos expansions [Sraj et al., 2016], radial basis functions and Kriging [Beglerovic et al., 2017], bayesian surrogate models [Ford et al., 2011], surrogate response surface models [Mattis and Wohlmuth, 2018], artificial neural networks [Xu et al., 2020]. These models have proven particularly valuable in the automotive industry for applications like car seats [Long et al., 2021], suspension components [Jiang et al., 2021], human-product interaction [Ahmed et al., 2018] or autonomous vehicles validation [Beglerovic et al., 2017].

This chapter focuses on building a surrogate model using supervised machine learning to replicate and replace the Renault numerical simulator. This model aims to predict simulated time series based on specific input parameters, serving as a generative model. It is constructed from a dataset of pre-simulated scenarios and is designed to mirror the simulator's output while ensuring faster computation times.

The surrogate model must meet two critical criteria: it should be computationally efficient to expedite the calibration process and highly accurate to replace the simulator effectively. The goal is to create a reliable estimator of the simulator that can handle multivariate output generation.

The chapter progresses by discussing about using different data formats; testing traditional machine learning techniques like k-nearest neighbors (Section 2.3.1), random forests (Section 2.3.2), and kernel ridge regression (Section 2.3.3); adapting polynomial chaos expansion for functional contexts (Section 2.4); applying dimension reduction through principal components analysis (Section 2.5); developing several neural networks architectures with deep forest (Section 2.6.1), recurrent neural network (Section 2.6.2) and convolutional neural network (Section 2.6.3); and exploring hybrid and aggregated models (Section 2.7).

These discussions aim to refine the surrogate model to make it a viable replacement for the numerical simulator, ensuring both speed and precision in generating realistic time series. We aim to provide a catalog of ad hoc supervised machine learning models tailored to our specific context and perform optimally regarding computational efficiency and effectiveness.

Chapter 3. Theoretical guarantees for functional expert aggregation

Remark. The work presented in this chapter was carried out in collaboration with Hugo Chardon and Étienne Donier-Meroz. It will result in the publication of an article. We contributed equally to the design and writing.

In the previous chapter, surrogate models were constructed for Renault's digital simulator with the objective of forecasting time series data based on non-temporal input parameters. In particular, we looked at expert aggregation, which has shown good performances. The method consists of a linear combination of multiple predictors, called experts, with associated weights. As experts are time series, the relative efficiency of each expert may vary at different time steps. Consequently, the weights assigned to each expert will also depend on time. The standard methodologies for expert aggregation necessitate the discretization of time series and weights. Although discretization is a necessary numerical procedure, it should not be included in the theoretical analysis of prediction algorithms. Furthermore, in contrast to the majority of cases where past values are employed to forecast future outcomes, the objective here is to construct the entire time series from non-temporal input parameters.

This framework, strongly motivated by my thesis, is adaptable to a large number of fields, including physics, economics, biology, and more. It can be applied in all contexts where data are continuous functions. For instance, in the social sciences, it can be used to predict gross domestic product based on demographic, economic, and geographical indicators. In biology, it can be employed to examine the evolution of a species as a function of its reproductive rate, food availability, and other factors.

The new context is defined by considering time series and weights as continuous functions of time and seeking to generate the entire time series. Thus, a new framework with specific input and output spaces, a model, a loss function, and a risk must be defined. The objective is to extend the concept of linear expert aggregation by integrating functional predictors and functional weights. It is imperative that the introduction of this continuous component must be rigorously conducted to demonstrate the convergence of algorithms under the appropriate conditions.

Once the new framework has been precisely defined, we will proceed to examine the theoretical guarantees of deterministic and stochastic projected gradient descent (PGD), as well as stochastic mirror gradient descent (MGD). The convergence rates of these approaches have been the subject of extensive studies in several sources. For deterministic PGD, see Rigollet [2015]. For stochastic PGD, see Lacoste-Julien et al. [2012]. For stochastic MGD, see Bubeck [2015]. The objective here is to verify the continued validity of these guarantees and, if necessary, adapt them to our new framework.

Chapter 4. Solving the inverse problem

The objective is to address the inverse problem of recreating on-track time series produced through simulation. To achieve this, we must identify the input parameters that generate the closest possible time series when provided to the numerical simulator.

Mathematically, the reference on-track time series y_φ and its associated input nominal parameters θ_0 are provided, and we want to find the input parameters θ^* such that $y^* := S(\theta^*)$ is as close as possible to y_φ. For a given distance d and a space to explore Θ, we need to solve

$$\theta^* \in \underset{\theta \in \Theta}{\arg\min}\, d\big(S(\theta), y_\varphi\big)\,.$$

The nominal values θ_0 are subject to uncertainties, mainly due to sensor inaccuracies, manufacturing tolerances, or errors in computational estimations. For those reasons, $\theta^* \neq \theta_0$ is generally used. Therefore, we can substantially improve the simulated time series by modifying the nominal values and inferring θ^*. The exploration space Θ is defined using the nominal values θ_0, which provide prior information to guide the search for the optimal solution.

The simulator S, which maps input parameters to output time series, exhibits inherent error compared to on-track experiments. Consequently, we incorporate additive noise into our statistical model. This chapter will explore various assumptions about this noise and discuss the corresponding results.

Initially, we assume that the noise is Gaussian with different types of covariance matrices. In this setting, we demonstrate that computing the maximum likelihood estimator aims to minimize the mean squared error (MSE). We then evaluate the performance using a frequentist approach by optimizing the MSE and a Bayesian approach using Bayesian synthetic likelihood.

Subsequently, we relax the assumption of Gaussian noise and consider noise with an unknown distribution. In this case, our objective is to test the effectiveness of a frequentist approach by optimizing the objective function $\theta \mapsto d\big(S(\theta), y_\varphi\big)$, and to explore Bayesian methods using sequential Monte Carlo (SMC) samplers.

Ultimately, we demonstrate the effectiveness of Bayesian methods by incorporating prior information throughout the prior distribution. The SMC sampler, developed for this specific problem, can adapt to other contexts, yielding overall good results. This work should be seen as the development of a generic, automatic methodology.

Chapter 2

Construction of the surrogate model

Contents

Remark. Throughout Chapter 2, unless otherwise specified, $u_{i,t}$ denotes the t-th coordinate of the i-th sample u_i taken from the whole data set $u_1, \dots, u_n$. The notation $y^{(r)}$ stands for the r-th type of time series, as a scenario y is composed with R distinct types of time series.

2.1 Introduction

Surrogate models are widely used in all types of domains and contexts. They have already demonstrated their usefulness and efficiency using a wide variety of possible methods: polynomial chaos expansions [Sraj et al., 2016], radial basis functions and Kriging [Beglerovic et al., 2017], bayesian surrogate models [Ford et al., 2011], surrogate response surface models [Mattis and Wohlmuth, 2018], artificial neural networks [Xu et al., 2020].

Surrogate models are also largely used in the automotive field and have demonstrated their accuracy in many applications like car seats [Long et al., 2021], suspension components [Jiang et al., 2021], human-product interaction [Ahmed et al., 2018] or autonomous vehicles validation [Beglerovic et al., 2017].

In this chapter, we construct a surrogate model with supervised machine learning methods that mimics and replaces the Renault simulator by predicting the simulated time series for a given set of parameters. We aim to create a model that accurately replicates the simulator's behavior while maintaining reasonable calculation times.

Contrary to the time series forecasting framework, where the objective is to predict the future from the past, the surrogate model aims to predict a whole time series from a set of parameters. More precisely, it is a generative model. The dataset used to build the model is a set of beforehand simulated scenarios output by the Renault simulator. The various input parameters need to be carefully chosen so that the database correctly represents the desired parameter definition space.

The use of a surrogate model in our context is particular: we want to integrate it into an iterative process to replace the simulator, which is too time-consuming. As such, it must comply with two specific constraints.

- The model must have reasonable calculation times to reduce the overall time taken by the calibration method. Faster calculation times are preferred.
- The model should strive for maximum accuracy to ensure its usefulness. If it deviates too much from the simulator, its integration will be ineffective, as the calibration process will not yield accurate results.

Under the guise of having an efficient calibration methodology, Figure 2.1 demonstrates the benefits of having a more accurate surrogate model, which makes it possible to generate even more realistic time series.

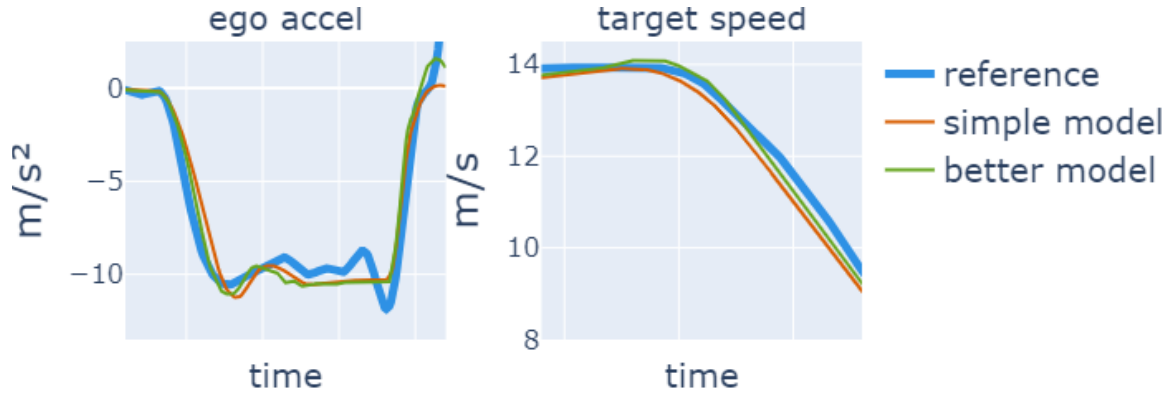


Figure 2.1: Reference on-track test and time series simulated with parameters inferred with a simple model vs. a better model

As a reminder, we want to construct an estimator $\widehat{S}$ of the simulator S which for a given $\theta \in \mathbb{R}^p$, predicts $y \in \mathbb{R}^T$ by $\widehat{y} = \widehat{S}(\theta)$. The uniqueness of this context is the multivariate output, which differs from the traditional context. While most numerical methods adapt well, it is important to define the formulas carefully in the multivariate case.

This chapter explores, defines, and tests methods for solving a specific time series generation problem. It begins by discussing traditional machine learning methods in Section 2.3, followed by the polynomial chaos expansion (PCE) adapted to the functional context in Section 2.4, and dimension reduction using principal component analysis (PCA) in Section 2.5. We examine and develop some neural network architectures in Section 2.6. Finally, the results of our first article [Carlier et al., 2023] are presented in Section 2.7 with hybrid and aggregated models.

2.2 Different data formats

Before building a model, it is essential to determine the data format. This can significantly impact the quality of the model. The output data is specific as it includes four distinct time series, each with varying total time steps. That is why it is necessary to provide them with special treatment that considers this multi-temporal component.

2.2.1 Vector format

The first step is to create as many columns as necessary to concatenate the time series one after the other. In this way, we associate a vector of parameters with a vector of time series.

Input. We have n samples $\theta = (\theta_1, \dots, \theta_n)$ where $\theta_i \in \mathbb{R}^p,\ \forall i = 1, \dots, n$.

$$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix} = \begin{pmatrix} \theta_{1,1} & \theta_{1,2} & \dots & \theta_{1,p} \\ \theta_{2,1} & \theta_{2,2} & \dots & \theta_{2,p} \\ \vdots & \vdots & & \vdots \\ \theta_{n,1} & \theta_{n,2} & \dots & \theta_{n,p} \end{pmatrix} \in \mathbb{R}^{n \times p}$$

Output. We have n samples $y = (y_1, \dots, y_n)$ where $y_i \in \mathbb{R}^T,\ \forall i = 1, \dots, n$ and $\sum_{r=1}^{R} T_r = T$ with T_r the size of the r-th type of time series.

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \left(\begin{array}{ccc|c|ccc} y_{1,1}^{(1)} & \dots & y_{1,T_1}^{(1)} & \dots & y_{1,1}^{(R)} & \dots & y_{1,T_R}^{(R)} \\ y_{2,1}^{(1)} & \dots & y_{2,T_1}^{(1)} & \dots & y_{2,1}^{(R)} & \dots & y_{2,T_R}^{(R)} \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_{n,1}^{(1)} & \dots & y_{n,T_1}^{(1)} & \dots & y_{n,1}^{(R)} & \dots & y_{n,T_R}^{(R)} \end{array}\right) = \left(\begin{array}{c|c|c} \vdots & & \vdots \\ y^{(1)} & \dots & y^{(R)} \\ \vdots & & \vdots \end{array}\right) \in \mathbb{R}^{n \times T}$$

We consider the current format classic, but it does not consider the time component. By outputting a vector in this way, the model lacks the information that these are distinct time series with temporal dependencies. Therefore, we propose the second format below, incorporating this information into the input parameters.

2.2.2 Matrix format

We will multiply the rows for this new format instead of adding more columns. This way, we will have only four columns in the output, each for a time series. The parameter vector will have a new value for the relevant time step. The format's only constraint is that all time series must have the same number of time steps. In this case, we take $T_r = T,\ \forall r = 1, \ldots, R$.

Input. Including a column of parameters for time steps multiplies the number of lines by T.

$$\Theta = \begin{pmatrix} \Theta_1 \\ \hline \ldots \\ \hline \Theta_n \end{pmatrix} = \left(\begin{array}{cccc|c} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,p} & 1 \\ " & " & \ldots & " & 2 \\ \vdots & \vdots & & \vdots & \vdots \\ " & " & \ldots & " & T \\ \hline & & \ldots & & \\ \hline \theta_{n,1} & \theta_{n,2} & \ldots & \theta_{n,p} & 1 \\ " & " & \ldots & " & 2 \\ \vdots & \vdots & & \vdots & \vdots \\ " & " & \ldots & " & T \end{array}\right) \in \mathbb{R}^{nT \times (p+1)}$$

Output. Each column corresponds to a time series.

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \hline \ldots \\ \hline \mathbf{y}_n \end{pmatrix} = \left(\begin{array}{cccc} y_{1,1}^{(1)} & y_{1,1}^{(2)} & \cdots & y_{1,1}^{(R)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ y_{1,T}^{(1)} & y_{1,T}^{(2)} & \cdots & y_{1,T}^{(R)} \\ \hline & \ldots & \ldots & \\ \hline y_{n,1}^{(1)} & y_{n,1}^{(2)} & \cdots & y_{n,1}^{(R)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ y_{n,T}^{(1)} & y_{n,T}^{(2)} & \cdots & y_{n,T}^{(R)} \end{array}\right) \in \mathbb{R}^{nT \times R}$$

Although this alternative format considers time steps, it significantly increases the number of rows in the data matrix and requires a time series of the same size.

In Section 2.8, we will compare these two formats by examining the four time series presented in Section 1.1.1 and with equal size T.

2.3 Classical machine learning methods

We tested three classic methods: k-nearest neighbors (k-NN), first developed by Fix and Hodges [1951] and later expanded by Cover and Hart [1967], random forest (RF), initially introduced by Breiman [2001], which offers many theoretical guarantees, including robustness [Roy and Larocque, 2012], and kernel ridge regression (KRR), which has been introduced and explained many times in Cristianini and Shawe-Taylor [2000], Saunders et al. [1998], and which shows the advantages of being used in our context [Exterkate et al., 2016].

2.3.1 k-nearest neighbors

The k-nearest neighbors (k-NN) method is one of the most fundamental and straightforward algorithms. For an in-depth reading, please refer to Peterson [2009]. k-NN can be used for classification and regression problems. For the given problem, the nearest k elements will assign either a class or a real value to the new input.

Let consider a regression problem where $(\theta_1, y_1), \ldots, (\theta_n, y_n)$ are pairs taking values in $\mathbb{R}^p \times \mathbb{R}^T$, where y is the real values of θ. For a given norm $\|.\|$ on $\mathbb{R}^p$ and a new point $\theta \in \mathbb{R}^p$, let $(\theta_{(1)}, y_{(1)}), \ldots, (\theta_{(n)}, y_{(n)})$ be a reordering of the training data such that

$$\|\theta_{(1)} - \theta\| \leq \cdots \leq \|\theta_{(n)} - \theta\| .$$

Considering the k-nearest neighbours to predict the new output aims to compute

$$\widehat{y}_{k\text{-NN}} = \widehat{S}_{k\text{-NN}}(\theta) = \frac{1}{k} \sum_{j=1}^{k} y_{(j)} ,$$

where we take the average of k vectors.

There are two parameters to define: k and the norm $\|.\|$. The choice of these two parameters is highly dependent on the data being considered. However, the Euclidean norm is the most commonly used. Implementing this approach becomes increasingly challenging as the parameter space grows due to the curse of dimensionality. Similarly, the more training data there is, the longer it will take the algorithm to search for the closest points.

2.3.2 Random forests

The random forests method takes the idea of bootstrap aggregating, also known as bagging [Breiman, 1996], and improves it. This method aims to construct a better estimator using different predictors, which will reduce the prediction variance [Chesneau, 2020, Genuer and

Poggi, 2017]. Random forests are a method that aggregates predictors. It is particularly effective when used with low-bias predictors. In the random forest case, trees will be used as predictors. The pseudo-code is provided in Algorithm 2 where $\theta_{i,j}$ denotes the j-th coordinate of the i-th sample.

Algorithm 2 Random forest algorithm

1: **for** $a = 1, \ldots, A$ **do**

2: Generate a bootstrap sample: choosing a random subset from the entire dataset $\{(\theta_i, y_i),\ \forall i \in I_a\}$

3: Fitting a *binary tree* to the bootstrap sample

- Random selection of J features $\{\theta_{i,j},\ \forall j \in J_a\}$ where $|J_a| = J$ and $J_a \subset \{1, \ldots, p\}$
- Construct the maximal tree $\mathcal{T}_a$ over this sub-dataset and sub-features, without pruning

4: The final output is obtained by averaging all decision tree outputs

Within the framework of regression, for a random forest $\{\mathcal{T}_a,\ a = 1, \ldots, A\}$, the estimator is given by

$$\widehat{y}_{\mathsf{RF}} = \widehat{S}_{\mathsf{RF}}(\theta) = \frac{1}{A} \sum_a \mathcal{T}_a(\theta)\,.$$

The random forests method is easy to implement and performs very well. There are only two parameters to calibrate: A and J. However, their interpretation is not straightforward. For regression, using $J = \sqrt{p}$ is recommended, where p is the number of covariates of $\theta \in \mathbb{R}^p$. If J is equal to p, this amounts to bagging.

2.3.3 Kernel ridge regression

The kernel ridge regression is quickly explained by Welling [2013]. We first introduce the ridge regression in a multivariate context and then the kernel trick.

Ridge regression

The ridge regression corresponds to a classic linear regression to which we have added a quadratic constraint on the coefficients. The objective is to find the coefficients $\beta \in \mathbb{R}^{T \times p}$ which minimise

$$\min_{\beta \in \mathbb{R}^{T \times p}} \left\{ \sum_{i=1}^{n} \|y_i - \beta \theta_i\|^2 + \lambda \sum_{t=1}^{T} \|\beta_{t,:}\|^2 \right\},$$

where $\beta_{t,:}$ is the t-th line of the matrix β.

This regression is very useful when the outputs are highly correlated. The exact solution to this problem is $\widehat{\beta} := \widehat{\beta}_{\text{ridge}} = \left(\lambda I_p + \theta^\top \theta\right)^{-1} \theta^\top y$.

Kernel trick

The kernel trick is a method for solving a non-linear problem with a linear regressor. The representation space of the input data is transformed into a higher-dimensional space by using a transformation φ. All the feature vectors θ are replaced by $\varphi_\theta := \varphi(\theta)$. The solution is now given by

$$\widehat{\beta} = \left(\lambda I_p + \varphi_\theta^\top \varphi_\theta\right)^{-1} \varphi_\theta^\top y \,.$$

From this point forward, the calculation depends solely on the scalar product between the two transformations φ_θ. However, this calculation is performed in high dimensions, making it difficult, if not impossible, to carry out. Therefore, we must revert to calculating a kernel $K(u, v) = \varphi_u^\top \varphi_v$, which, thanks to Theorem 2.1, is more accessible.

Theorem 2.1 (Mercer's theorem)**.** *Suppose K is a continuous symmetric positive-definite kernel and φ is a positive function. The kernel K is associated with the integral operator T_K defined by $T_K(\varphi)(\theta) = \int_a^b K(\theta, s)\varphi(s)\mathrm{d}s$. There is an orthonormal basis $(e_k)_k$ of $L^2[a, b]$ consisting of eigenfunctions of T_K such that the corresponding sequence of eigenvalues $(\lambda_k)_k$ is nonnegative. Then K has the representation*

$$K(s, t) = \sum_k \lambda_k e_k(s) e_k(t) \,,$$

where the convergence is absolute and uniform.

Remark. All these results hold also for $y_i = S(\theta_i) + \varepsilon$ instead of $y_i = \beta\theta_i + \varepsilon$.

2.4 Polynomial chaos expansion

Polynomial chaos expansion provides a functional approximation of a model using its spectral representation on a base of suitably constructed polynomial functions [Blatman and Sudret, 2011, Crestaux et al., 2009, Glaser et al., 2016]. Marelli and Sudret [2021] explains the construction details, which inspired the following section.

Let θ represent the random variable associated with the input parameters and f_θ its density, and let Y represent the random variable of the output time series. Assume $(\theta_i)_{i=1,\dots,n}$ are independent. We decompose Y into an orthogonal polynomial basis

$$Y = S(\theta) = \sum_a \lambda_a \psi_a(\theta)\,,$$

where λ_a are deterministic coefficients to compute and ψ_a are multivariate polynomials.

To compute the multivariate polynomials ψ_l, we consider $\{\pi_j^{(i)},\ j \in \mathbb{N}\}$ a set of univariate polynomials that are orthonormal to $f_{\theta_i}(\theta)$ the probability density function of θ_i. The polynomials satisfy

$$\langle \pi_j^{(i)}, \pi_k^{(i)} \rangle = \mathbb{E}[\pi_j^{(i)} \pi_k^{(i)}] = \delta_{j,k} \quad \forall j,k \in \mathbb{N} \text{ and } i = 1,\dots,n\,,$$

where $\delta_{j,k} = \mathbb{1}_{j=k}$ and the degree of $\pi_j^{(i)}$ is equal to $j \times \mathbb{1}_{j>0} + \mathbb{1}_{j=0}$.

By considering the following tensor product

$$\psi_a = \pi_{a_1}^{(1)} \otimes \cdots \otimes \pi_{a_n}^{(n)}\,,$$

with $a = (a_1,\dots,a_n)$, we construct a basis of multivariate polynomials $\{\psi_a,\ a \in \mathbb{N}^n\}$. The polynomials then satisfy

$$\langle \psi_a, \psi_b \rangle = \mathbb{E}[\psi_a \psi_b] = \delta_{a,b} \quad \forall a,b \in \mathbb{N}^n\,.$$

Initially, PCE is formulated in the context of Gaussian random variables. Therefore, Hermite polynomials are used as the univariate polynomial basis $\{\pi_j\}_j$. Each type of random variable has a corresponding set of functions.For other potentially unknown laws, performing a nonlinear mapping (probabilistic transformation) to return to classical variables and perform a generalized PCE is possible. Alternatively, we can generate a custom set of orthonormal polynomials.

Although this approach is suitable for our context because it incorporates a time component, we could not generate a time series for a new theta after extensive research. As the Chaospy package does not include any generative methods, we cannot present concrete results.

2.5 Dimensionality reduction with principal component analysis

Dimensionality reduction is crucial in this thesis. While our data does not require many input parameters, other cases may necessitate variable selection or dimension reduction. The output vector size can become huge when utilizing data in vector format. Therefore, one potential solution is to reduce the dimensionality of the time series and summarise them in a few values instead of retaining all time steps. We explored various principal component analysis (PCA) versions to achieve this.

2.5.1 Classical

Initially invented by Pearson [1901] and later independently developed and named by Hotelling [1933], PCA is a technique used to identify a base of representation in which data is decomposed into principal components. The principal components are determined by decomposing the covariance matrix into its eigenvalues. For an in-depth reading, please refer to Jolliffe [2002].

To compute the covariance matrix, we first center the data y by calculating the average across the columns. The centered matrix is given by

$$y_{\mathsf{c}} := \left(y_{i,t} - \overline{y}_t\right)_{\substack{i=1,\dots,n \\ t=1,\dots,T}}, \tag{2.5.1}$$

where $\overline{y}_t = \frac{1}{n}\sum_{i=1}^{n} y_{i,t}$ for $t = 1, \dots, T$. The data can also be normalized by dividing each column by its standard deviation; if so, it is the correlation matrix that we need to compute. The covariance matrix is

$$\rho_y := \frac{1}{T} y_{\mathsf{c}}^\top y_{\mathsf{c}}\,. \tag{2.5.2}$$

We now need to calculate the vector $\lambda = (\lambda_t)_{t=1,\dots,T}$ which contains the eigenvalues of ρ_y and the matrix $v = (v_{s,t})_{s,t=1,\dots,T}$ of the eigenvectors associated to the eigenvalues. The desired number of principal components is selected by ranking λ and v in descending order of eigenvalues and choosing the top $c_p \in [\![1, T]\!]$ components. If $c_p = T$, no change is made, and all the information is kept.

Multiplying y by the v matrix reduces the data size. It is important to note that the PCA can be inverted to return the data to the original space. The following formula is used for this purpose

$$\widehat{y}_{\mathsf{pca}} = y_c v v^\top + \overline{y}\,,$$

where $\overline{y} = (\overline{y}_t)_{t=1,\dots,T}$ is the vector of the column's average.

2.5.2 Functional

The following section introduces functional PCA by first defining functional data and its capabilities. We then explain functional PCA and how it works.

Functional data

Functional data involves interpreting our data as functions or curves rather than as a series of point observations. It is beneficial for time series, as they can then be interpreted as functions of time [Wohlenberg, 2021]. The benefits are varied:

- The function can be evaluated at any time step, even though no measurements have been taken;
- If several functions are dealt, it is possible to set them to the same time step;
- It is possible to study the derivative of the function;
- Finally, it provides a more natural and intuitive view of the data.

Functional PCA method

In a traditional PCA, the eigenvectors associated with the data have to be calculated. In the functional framework, due to the nature of the data, these are no longer vectors but eigenfunctions [Shang, 2014]. The data is expressed in the form

$$y(t) = \big(y_1(t), y_2(t), \dots, y_n(t)\big)\,,$$

and its mean is $\mu(t)$.

Theorem 2.2 (Karhunen-Loève decomposition)**.** *A stochastic process y can be expressed as*

$$y(t) = \mu(t) + \sum_{k\geq 1} \xi_k \varphi_k(t)\,,$$

where ξ_k are the principal components associated to the eigenfunctions φ_k, and are given by

$$\xi_k = \int_{\mathcal{T}} (y(t) - \mu(t))\varphi_k(t)\mathrm{d}t\,,$$

which verify for all $k, l \geq 1$ and $k \neq l$

$$\mathbb{E}[\xi_k] = \mathbb{E}[\xi_k \xi_l] = 0 \quad \textit{and} \quad \mathrm{Var}(\xi_k) = \lambda_k\,.$$

The covariance matrix is calculated differently using the formula

$$\rho_y(s,t) = \mathsf{Cov}(y(s), y(t)) = \sum_{k\geq 1} \lambda_k \varphi_k(s)\varphi_k(t)\,,$$

where λ_k are the eigenvalues and φ_k the corresponding eigenfunctions. With the decomposition of the Theorem 2.2, $y(t)$ is approximated by a finite sum

$$y(t) \approx y_K(t) := \mu(t) + \sum_{k=1}^{K} \xi_k \varphi_k(t)\,.$$

To compute the eigenfunctions $(\varphi_k)_{k\geq 1}$, the following quantities need to be maximised

$$\varphi_1 = \underset{\|\varphi\|_2=1}{\arg\max} \left\{ \mathrm{Var} \int_{\mathcal{T}} (y(t) - \mu(t))\varphi(t)\mathrm{d}t \right\}\,,$$
$$\forall k \geq 2,\ \varphi_k = \underset{\substack{\|\varphi\|_2=1 \\ \langle \varphi, \varphi_j \rangle = 0 \\ j=1,2,\ldots,k-1}}{\arg\max} \left\{ \mathrm{Var} \int_{\mathcal{T}} (y(t) - \mu(t))\varphi(t)\mathrm{d}t \right\}\,,$$

where $\|\varphi\|_2 = (\int_{\mathcal{T}} \varphi^2(t)\mathrm{d}t)^{1/2}$ and $\langle \varphi_i, \varphi_j \rangle = \int_{\mathcal{T}} \varphi_i(t)\varphi_j(t)\mathrm{d}t$.

We use the `FDApy` package, which was implemented by Golovkine [2021].

2.5.3 Sparse

The sparse PCA, initially introduced by Zou et al. [2006], is an extension of the PCA, which introduces sparsity structures to the variables. One disadvantage of classical PCA is that the principal components are typically linear combinations of all variables. Sparse PCA addresses this issue by identifying components that are linear combinations of only a few variables. Specific coefficients in the linear combinations have a value of zero.

Consider the centered matrix (2.5.1) and its covariance matrix (2.5.2). Let k be an integer $1 \leq k \leq T$. The sparse PCA problem aims to maximize the variance along a direction represented by vector $v \in \mathbb{R}^T$ while constraining its cardinality

$$\max_{\substack{\|v\|_2=1 \\ \|v\|_0 \leq k}} v^\top \rho_y v\,,$$

where $\|.\|_0$ is the pseudo-norm that counts the number of non-zero components. If $k = T$, it is equivalent to the classical PCA. Moreover, sparse PCA doesn't guarantee that principal components are orthogonal, but the orthogonality can be achieved by adding more constraints.

2.6 Neural networks

Deep learning is a subset of machine learning methods based on artificial neural networks. It provides a wide range of techniques for solving any problem: supervised or unsupervised, regression or classification, etc. It has shown remarkable success in many domains, including computer vision, natural language processing, and speech recognition, revolutionizing the field of artificial intelligence.

The structure of the human brain inspires neural networks. They consist of interconnected nodes called artificial neurons, which model the neurons in a brain and are organized into layers. All nodes are connected by edges, which model the synapses in a brain. Each neuron receives input, processes it through an activation function, and produces an output that feeds into the next layer. See Figure 2.2 for an example of a neural network containing two layers, each with r and q neurons.

A formal neuron is defined by a model characterized by an internal state calculated from input signals $\theta = (\theta_1, \dots, \theta_p)$ and g an activation function

$$g\left(w_0 + \sum_{k=1}^{p} w_k \theta_k\right) = g(w_0 + w^\top \theta)\,,$$

where w_0 is the neuron bias and $w^\top = (w_1, \dots, w_p)$ is the vector containing the weights associated with each neuron, whose values are estimated during the network learning phase.

There are several activation functions, including the threshold function $g(u) = \mathbb{1}_{u \geq 0}$, sigmoid function $g(u) = (1 + \exp(u))^{-1}$, ReLU function $g(u) = \max(0, u)$, softmax function $g(u) = \exp u / \sum_i \exp(u_i)$, and radial function $g(u) = (2\pi)^{-1/2} \exp(-u^2/2)$.

Trying to build a neural network that outperforms traditional machine learning methods is a natural next step. To achieve this, we explored three specific types. We examine the deep forest algorithm, which combines random forest and neural networks. We then test recurrent neural networks, precisely long short-term memory models designed explicitly for predicting time series. Finally, we test convolutional neural networks, demonstrating natural capabilities in predicting time series by considering time dependency in spatial structure.

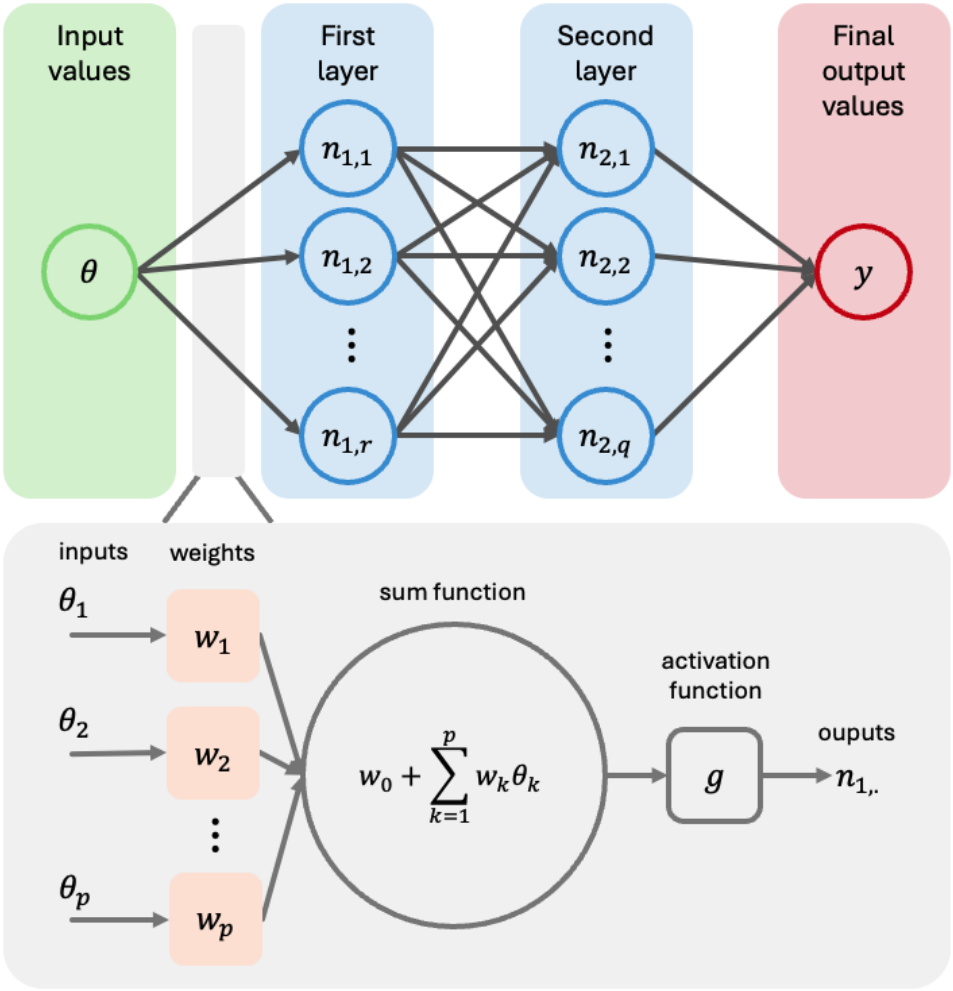


Figure 2.2: Diagram showing a neural network containing two layers one of r neurons and the other of q neurons.

2.6.1 Deep forest

Deep forest (DF) is a deep learning approach introduced by Zhou and Feng [2017, 2019] that integrates the principles of ensemble learning, exemplified by random forests, with the hierarchical representation learning typically associated with neural networks. The architecture of a DF model is structured as a cascade of layers, each comprising multiple decision trees. As training progresses, each successive layer in the cascade aims to capture more abstract data representations. Every decision tree is trained independently within a layer, allowing for a diverse set of features and patterns to be learned. The predictions from these individual trees are then collectively aggregated to form the final model output. This method leverages ensemble methods' robustness and generalization capabilities while also harnessing the power of deep learning's ability to learn complex hierarchical representations, making DF particularly effective for a wide range of tasks.

2.6.2 Recurrent neural network

Recurrent neural networks (RNNs) are a specialized class of artificial neural networks crafted to handle sequential data effectively. They distinguish themselves from traditional feedforward neural networks by featuring connections that form directed cycles, enabling them to maintain a hidden state. This hidden state acts as a memory of previous inputs, thus capturing information from the past to influence current processing. This dynamic ability to model temporal behavior makes RNNs particularly useful for applications such as time series prediction, sequence generation, and natural language processing.

Long short-term memory (LSTM) networks were developed to enhance the capabilities of RNNs, particularly in addressing the vanishing gradient problem that hampers learning in deeper models, as detailed by Hochreiter and Schmidhuber [1997]. LSTMs are a refined version of the basic RNN architecture, with memory cells and sophisticated gating mechanisms. These gates - specifically the input, forget, and output gate - manage the flow of information by deciding what to retain and exclude from the cell state. This selective retention and propagation of information allow LSTMs to capture long-range dependencies in sequential data more effectively than standard RNNs while mitigating the impact of vanishing gradients. This makes LSTMs adept at modeling complex sequences with crucial long-term contexts.

RNN and LSTM models have the particular feature of using past time values as inputs to predict the future. We must adapt their use in our context since the inputs are timeless parameters. As shown in Figure 2.3, we utilize the input parameters to predict the initial 30 time series steps using a traditional machine learning approach. These initial steps are then fed into an LSTM layer to predict the remaining steps of the time series. In addition to the LSTM layer, we incorporate a perceptron layer to retain the input parameter values within the model.

The final architecture of this model, named 30-multi-LSTM, is provided in Figure 2.4. The value 30 and the ML model are chosen to provide the best global performance. The LSTM layer is dependent on the number of time series present. In our case, four types of time series (ego speed and acceleration, target speed, and distance) result in four concatenated LSTM layers.

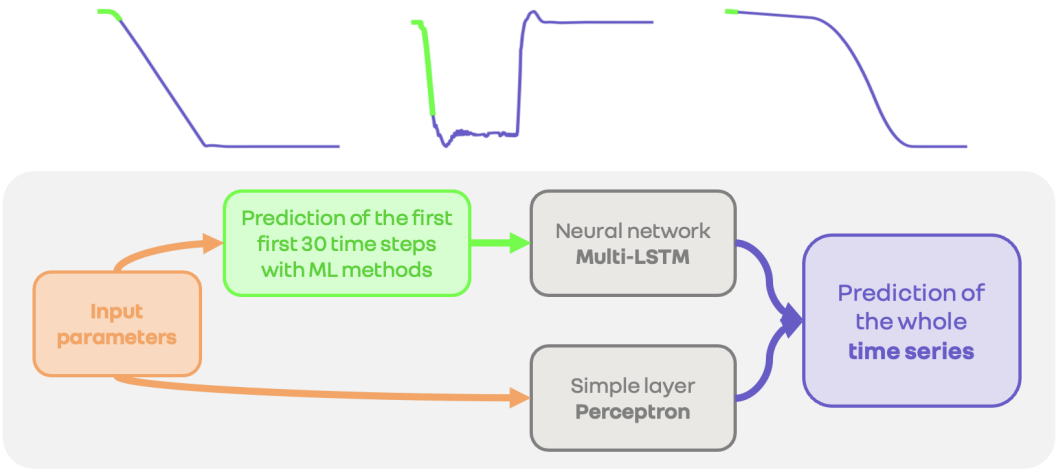


Figure 2.3: Diagram of the 30-multi-LSTM model

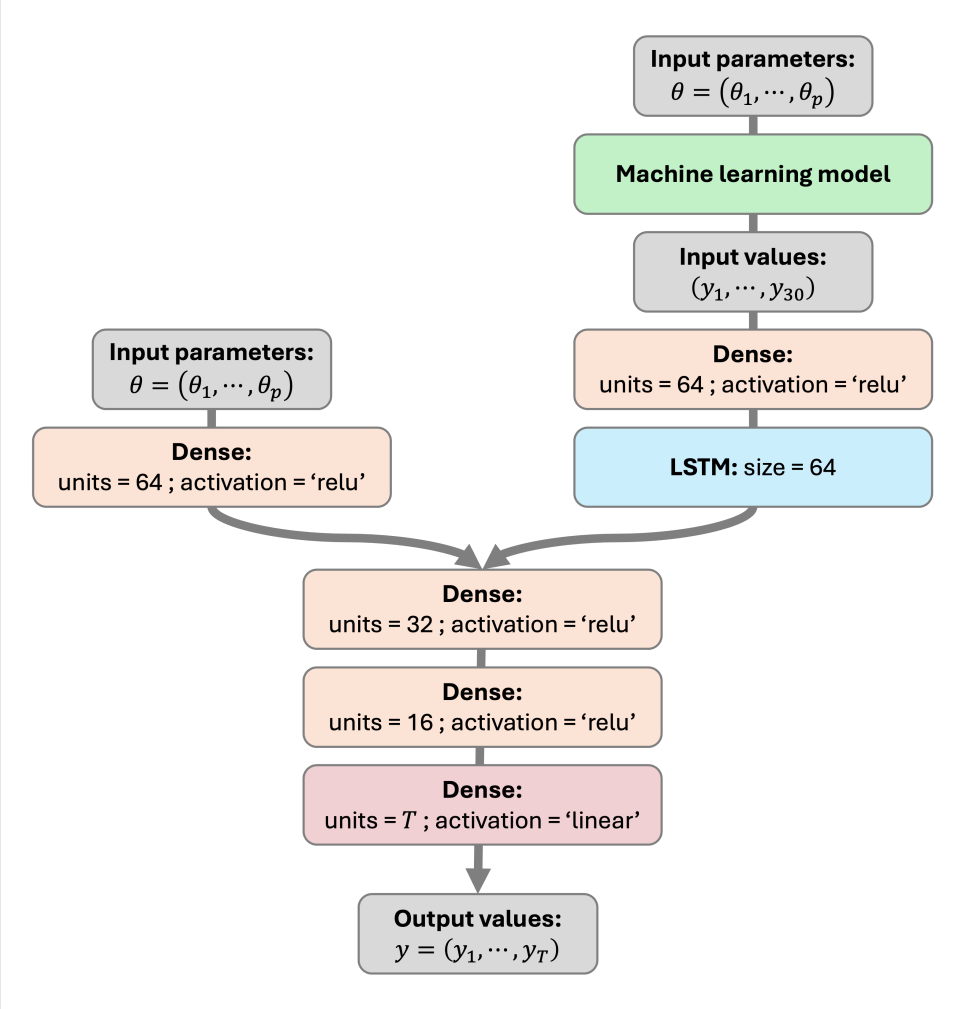


Figure 2.4: Architecture of the 30-multi-LSTM model for y a time series

2.6.3 Convolutional neural network

Convolutional neural networks (CNNs) are a category of deep learning models that are especially effective when working with image data. Unlike traditional models such as recurrent neural networks (RNNs), CNNs can capture various contextual nuances of visual information without depending on sequential data. The architecture of a CNN involves multiple layers that utilize filters to process input images. These filters systematically scan the images to identify and extract key features such as edges and textures. After feature extraction, the information undergoes a pooling process to reduce dimensionality and then is forwarded to fully connected layers. Throughout the training phase, a CNN progressively learns to detect and interpret complex patterns within images automatically. This capability renders CNNs highly suitable for tasks that involve image recognition and object detection, as they can effectively discern and classify different objects within an image based on learned patterns.

The first prominent CNN architecture was developed by Lecun et al. [1998], marking a significant milestone in the evolution of neural networks. Since then, CNNs have undergone extensive improvements and refinements, consistently setting new benchmarks in performance. A notable example of such advancements is the AlexNet model, introduced by Krizhevsky et al. [2012], which significantly pushed the boundaries of what was possible with CNNs. More recently, the application of CNN models has extended beyond traditional image and video tasks to include time series forecasting, as highlighted in studies like Borovykh et al. [2017] and Koprinska et al. [2018]. In these applications, CNNs uniquely treat temporal dependencies as spatial structures. This approach allows them to effectively predict future values in time series data by leveraging their powerful spatial feature extraction capabilities, thus adapting the strengths of CNNs from visual to temporal data analysis.

We develop our model to adapt to our context and optimize the hyperparameters for the best performance. Its architecture is described in Figure 2.5.

A vector of size p input parameters is converted into a vector of length T, where $T \geq p$. Since `Conv1D` layers reduce the data size, `UpSampling1D` layers are interleaved to obtain a larger output vector. The increase in size must be gradual for optimum performance, hence the multiplication of layers. After testing multiple activation functions, combining several through alternating permits obtain better results.

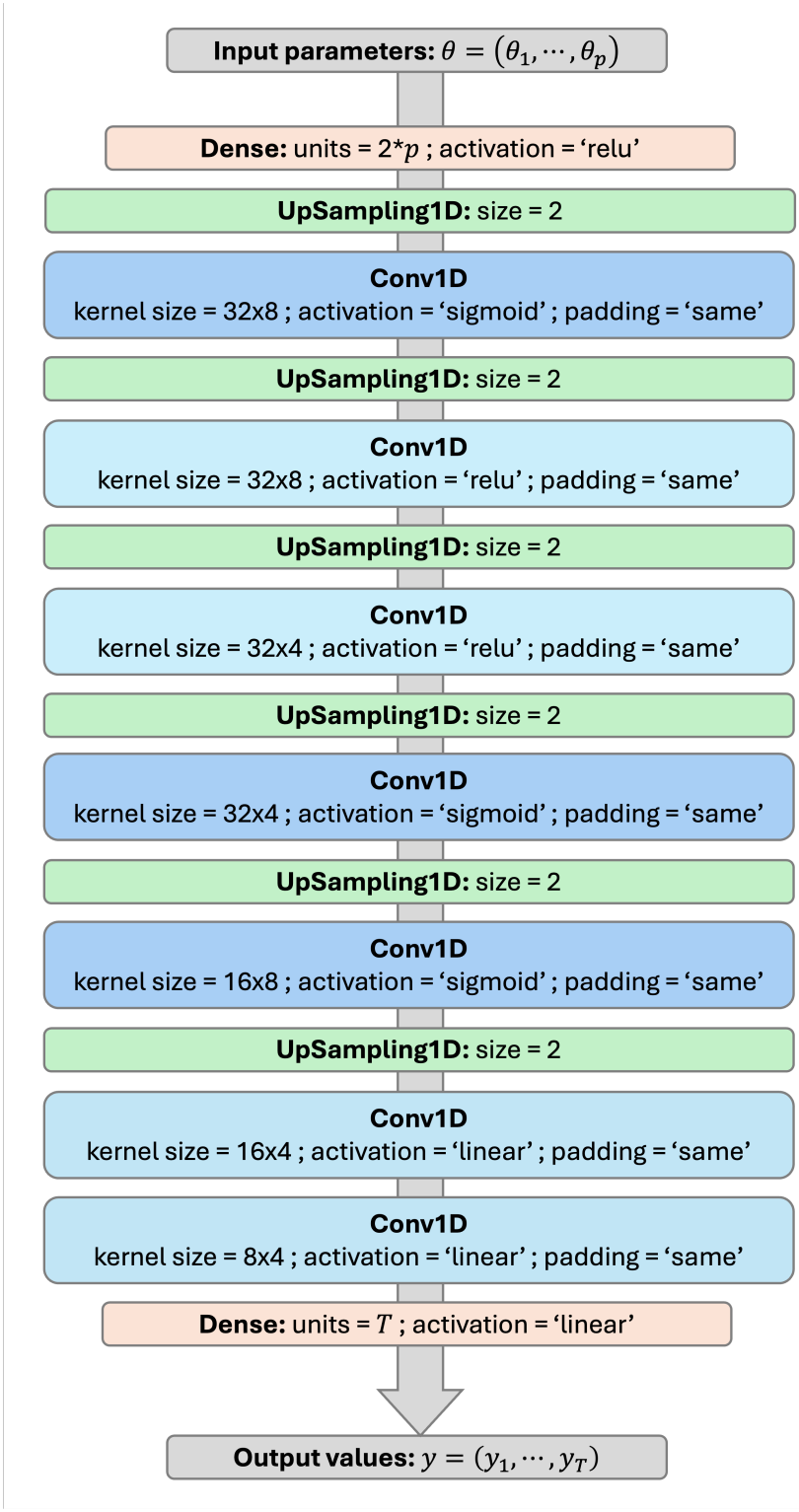


Figure 2.5: Architecture of our CNN model

2.7 Hybrid and aggregated models

This section presents three new models: two with a hybrid approach and one with aggregation. The first two choose the best method at each time step t of the time series, and the third performs a mixture of methods by giving them different weights computed with expert aggregation. These approaches are described in the next section.

2.7.1 Mathematical description

The history of work on expert aggregation dates back to the publications of Blackwell [1956] and Hannan [1957] in game theory. In learning theory, Vovk [1995] first proposed the problem of sequentially predicting arbitrary sequences. Refer to Cesa-Bianchi and Lugosi [2006] work for a complete review.

We have access to M experts noted $(f_m)_{m=1,\dots,M}$ and each one provides a prediction for each time step t noted $f_{m,t}(\theta)$ where θ is a given input parameter. Expert aggregation then predict each time step $\widehat{y}_t$ from past observations $y_1, \dots, y_{t-1}$ and current and past predictions from experts $(f_{m,j})$ for $m = 1, \dots, M$ and $j = 1, \dots, t$. At each time step t, a weight vector $\omega_t = (\omega_{1,t}, \dots, \omega_{M,t})$ is constructed to compute the prediction

$$\widehat{y}_t = \sum_{m=1}^{M} \omega_{m,t} f_{m,t}(\theta) \,,$$

where $\sum_{m=1}^{M} \omega_{m,t} = 1$ and $\omega_{m,t} \geq 0$.

To ensure the quality of the expert prediction by considering past observations and predictions, we utilize the following time loss function

$$L_{m,T}(\theta) := \sum_{t=1}^{T} \ell(f_{m,t}(\theta), y_t) \,,$$

where ℓ is the RMSE defined in paragraph Classical losses of Section 1.2.1.

Only one weight vector value equals 1, and the rest is 0 to construct the two hybrid models. For the aggregated model, at least two values are non-zero. In this case, the weight vector is calculated using the traditional exponentially weighted aggregation (EWA) algorithm [Mourtada, 2016], which is given in Algorithm 3 for a given pair (θ, y).

Traditionally, the EWA algorithm is used to forecast the future of a unique time series sequentially, like the electricity consumption in France, for example [Devaine et al., 2013]. But, in our context, we aim to construct a model that fits a database composed of several pairs $(\theta_i, y_i)_{i=1,\dots,n}$ of inputs and outputs. To construct the weight vector, we need to consider all these values.

Algorithm 3 Exponentially weighted aggregation algorithm

Initialization:

1: Define the parameter $\eta > 0$

2: Set the initial weights $\omega_{m,1} = 1/M,\ \forall m = 1, \ldots, M$

3: Compute the prediction of the first time step $\widehat{y}_1 = \sum_{m=1}^{M} \omega_{m,1} f_{m,1}(\theta)$

Iterations:

4: **for** $t = 2, \ldots, T$ **do**

5: For each $m = 1, \ldots, M$, update the weights

$$\omega_{m,t} = \frac{\exp\left(-\eta L_{m,t-1}(\theta)\right)}{\sum_j \exp\left(-\eta L_{j,t-1}(\theta)\right)}$$

6: Make the prediction by aggregating $\widehat{y}_t = \sum_{m=1}^{M} \omega_{m,t} f_{m,t}(\theta)$

Consequently, Algorithm 3 is used to construct a weight matrix $\Omega^{(i)}$ for each pair (θ_i, y_i) of the training set. This matrix is defined by

$$\Omega^{(i)} = \begin{pmatrix} \omega_{1,1}^{(i)} & \ldots & \omega_{1,T}^{(i)} \\ \vdots & & \vdots \\ \omega_{M,1}^{(i)} & \ldots & \omega_{M,T}^{(i)} \end{pmatrix} \in \mathbb{R}^{M \times T}$$

for $i = 1, \ldots, n$, resulting in n weight matrices. The time series for a new θ will be generated by taking the average of those matrices $\overline{\Omega} := \frac{1}{n} \sum_i \Omega^{(i)}$ and performing the aggregated expert prediction

$$\widehat{y}_{t,i} = \sum_{m=1}^{M} \overline{\Omega}_{m,t} f_{m,t}(\theta_i)\,,$$

where $\overline{\Omega}_{m,t} = \frac{1}{n} \sum_i \omega_{m,t}^{(i)}$ is the m-th line and t-th column value of $\overline{\Omega}$.

2.7.2 The choice of experts

The aim is to construct several efficient machine learning models that provide distinct predictions. Once fitted, a unique model is chosen for each time step to build the hybrid model, and the aggregated model is constructed with an EWA algorithm.

Remark. From now until the end of Section 2.7.2, we provide a construction example of hybrid and aggregated models derived from our first article [Carlier et al., 2023].

The experts are selected among standard learning methods described above: k-nearest neighbors (k-NN, Section 2.3.1), random forests (RF, Section 2.3.2), kernel ridge regression (KRR, Section 2.3.3), deep forest (DF, Section 2.6.1) and convolutional neural networks (CNN, Section 2.6.3). Concerning random forests, we developed a global model that simultaneously predicts the four time series (1-RF) and another obtained by training four distinct random forest models, one per time series (4-RF). We also consider a model that combines random forests with a dimensionality reduction method, like classical PCA (PCA-RF, Section 2.5).

2.7.3 Construction of the hybrid and aggregated models

We first construct the seven models described above with the training set. The models are tested on the validation set by calculating the RMSE for each time series and model. The corresponding results are given in Section 2.7.3.

method	ego speed	ego accel	target speed	distance	mean
k-NN	11.00	64.77	8.02	36.83	30.15
KRR	2.20	7.14	1.63	17.86	7.21
CNN	0.18	3.85	**1.06**	**4.16**	**2.31**
DF	1.32	8.88	5.80	12.19	7.05
1-RF	1.36	9.54	4.86	13.31	7.27
4-RF	**0.12**	**1.47**	2.34	10.84	3.69
PCA-RF	2.33	20.53	5.58	20.96	12.35

Table 2.1: RMSE for each time series on validation set with the seven methods.

We notice essential differences between the time series. Each method performs more or less well with each time series. In this case, 4-RF performs best for the ego series, while CNN performs best for target speed and distance. We then want to detail further the error values: for all methods, the RMSE is calculated for each time step, Figure 2.6 represents it.

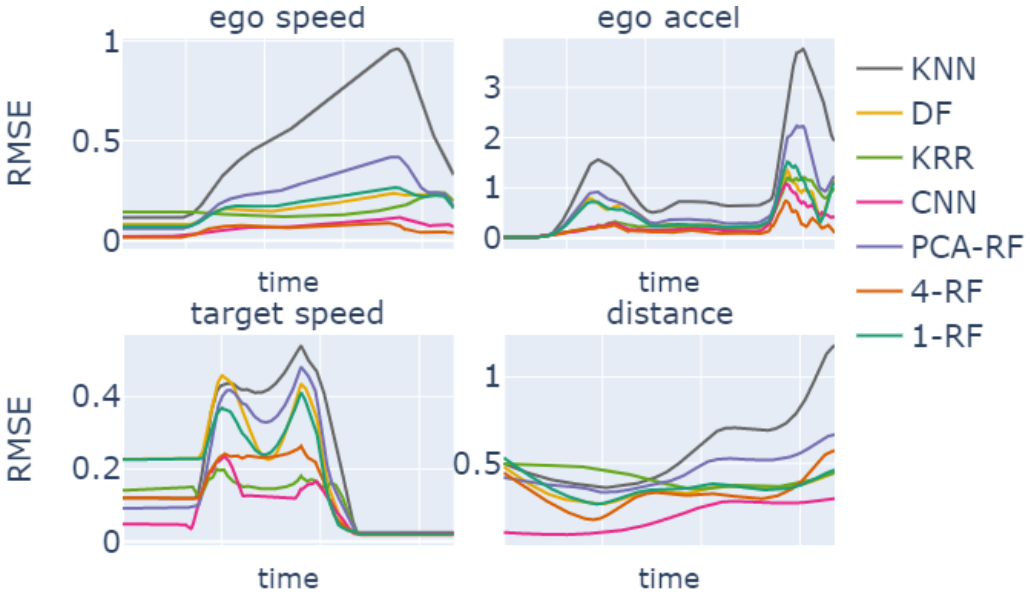


Figure 2.6: RMSE obtained at each time step
with each method on the validation set

With these detailed RMSE, we can distinguish which method performs better at each time step. CNN outperforms other methods for distance, but it is unclear for the rest of the series. These results suggest the construction of hybrid and aggregated models specific to each step.

We build three new models: two with a hybrid approach and one with aggregation. The first two consist of choosing the best method at each step, and the third one performs a mixture of methods by giving them different weights computed with an expert aggregation.

- **Hybrid 1:** for each time step, we select the best method among the seven proposed;
- **Hybrid 2:** we keep the three most used methods in hybrid 1 (CNN, 4-RF, and PCA-RF, see Section 2.7.3) and select the best one for each time step;
- **Aggregated:** it is built with an EWA algorithm described in Section 2.7.1.

CNN	**4-RF**	**PCA-RF**	**KRR**	**1-RF**	**DF**	k**-NN**
1081	997	430	93	53	28	2
40%	37%	16%	3%	2%	1%	$>1\%$

Table 2.2: Number of time steps for which the methods are selected
in the hybrid 1 model

Figure 2.7 shows which method is selected at each time step, and the weights given to each method for each time step are shown in Figure 2.8.

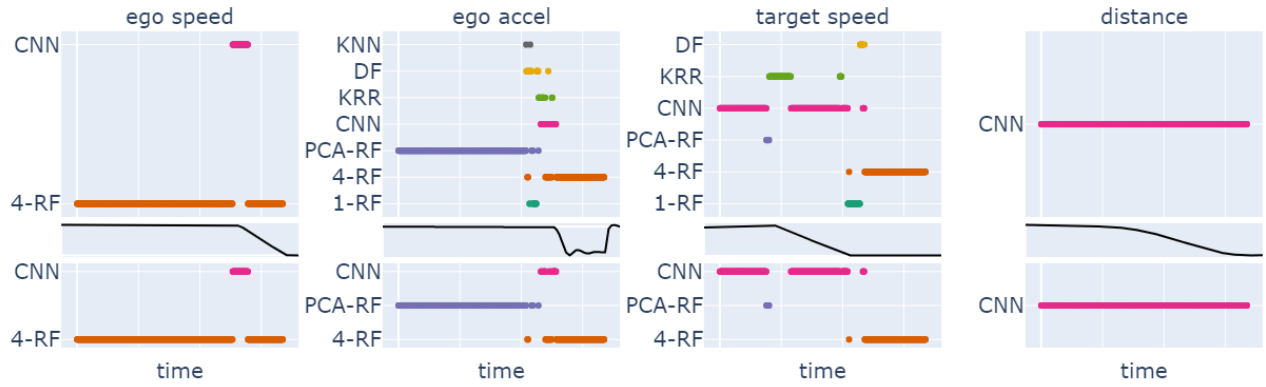


Figure 2.7: Selected method at each time step on the validation set
First line is hybrid 1: choices among the 7 methods;
third line is hybrid 2: choices among the 3 best methods.

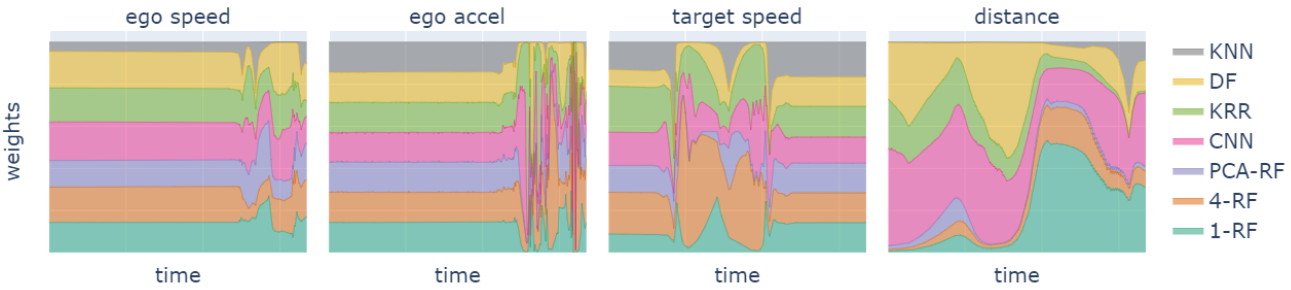


Figure 2.8: Associated weights for each expert at each time step
on the validation set

All numerical results obtained with these three approaches are summarized in the sections below. First, we compute RMSE values on the **validation set**, which has been used to tune hybrid and aggregated approaches: for each time step, the choice of the methods for hybrid models, and the weights for aggregated one. Then, we calculate RMSE values on the **test set** to confirm the results and evaluate the generalization quality.

Validation set

The validation set results are given in Table 2.3. The three new models are better than the CNN and 4-RF models. The aggregated is relatively better. Hybrid 1 and hybrid 2 are similar. To conclude on the best model, let us now see **how these results generalize to the test set**.

method	ego speed	ego accel	target speed	distance	mean
CNN	0.18	3.85	1.06	4.16	2.31
4-RF	0.12	1.47	2.34	10.84	3.69
Hybrid 1	0.11	1.46	0.93	4.16	1.66
Hybrid 2	0.11	1.46	1.04	4.16	1.69
Aggregated	**0.07**	**0.59**	**0.24**	**1.36**	**0.56**

Table 2.3: RMSE for each time series with CNN, 4-RF, hybrid and aggregated models on the validation set

Test set

Table 2.4 summarizes the test set results. The aggregated model is no longer the best, and it is even worse than the CNN model. The distribution of the weights cannot be generalized and is too specific to the validation set. However, the results of the two hybrid models are still better than those of the CNN and 4-RF models.
With the hybrid 1 model, the prediction of target speed deteriorates. From these results, we can see **the advantage of the hybrid 2 model**: for each time series, it improves predictions, although the choice of methods was made on the validation set. Whatever the database used to calibrate, the hybrid 2 model generalizes best.

method	ego speed	ego accel	target speed	distance	mean
CNN	0.23	3.34	1.00	2.52	1.77
4-RF	0.13	1.36	2.36	9.84	3.42
Hybrid 1	0.12	1.35	1.12	2.52	1.28
Hybrid 2	0.12	1.35	**1.00**	2.52	**1.25**
Aggregated	0.50	3.66	1.08	3.38	2.16

Table 2.4: RMSE for each time series with CNN, 4-RF, hybrid and aggregated models on the test set

2.8 Conclusion

Table 2.5 presents a summary of errors obtained using the methods described in Chapter 2. Table 2.6 provides information on the training time and the time required to make 500 predictions for each approach. Both tables should be considered when selecting the most suitable method based on performance requirements and time constraints.

Of the traditional approaches (1-RF, 4-RF, k-NN, KRR), developing a random forest model for each time series (4-RF) is the best-performing approach. It demonstrated better results on the validation and test bases, obtaining 2.243 and 3.047. The prediction time is also reasonable at 8 ms.

We couple the RF approaches with the matrix format (matrix 1-RF and matrix 4-RF) and various PCA (PCA 4-RF, Sparse PCA 4-RF, and fPCA 4-RF). The matrix format, although promising on the training data with errors of 0.310 and 0.201, does not generalize well and does not improve the results on the validation and test data, obtaining 2.688 and 4.012 for matrix 1-RF and 2.692 and 3.999 for matrix 4-RF. Additionally, the training for this task can last from 30 minutes to 2 hours, and the prediction times are well above 144 ms and 484 ms. Adding PCA does not improve results and even degrades computational times due to the calculation of decompositions.

The DF model produces results equivalent to those of classical ML methods and is not particularly efficient. Other neural network approaches (CNN and multi-LSTM) demonstrate excellent results with errors of less than 1 for all data sets. Among these approaches, the CNN model yields the best results. Additionally, it has a reasonable prediction time of 3 ms, although its training time of 6 hours is longer than the other models.

Finally, in conclusion, the hybrid and aggregated models, the aggregated model produced disappointing results with errors of 0.996 and 1.484. However, with the validation set, the hybrid model performed better with an error of 0.422, but its error of 0.607 on the test set was worse than CNN's error of 0.558. As both of these methods add calculation time, it is unwise to use them. However, their inclusion can enhance the outcome as demonstrated in Section 2.7.3. Therefore, their usage depends on the context and is preferred or avoided accordingly.

Numerical results

RMSE $(\times 10^{-1})$	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
train	1.108	0.998	0.310	**0.201**	1.051	1.136	1.050
validation	2.535	2.243	2.688	2.692	2.572	2.598	2.583
test	3.308	3.047	4.012	3.999	3.465	3.481	3.473

RMSE $(\times 10^{-1})$	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
train	2.693	0.917	1.221	0.500	0.463	$\times$	$\times$
validation	2.966	3.234	1.791	0.500	0.507	**0.422**	0.996
test	3.903	4.009	2.791	**0.558**	0.665	0.607	1.484

Table 2.5: Numerical results obtained with all methods presented above
Hybrid and aggregated with KRR, 4-RF, CNN and multi LSTM

Computation times

time	1-RF	4-RF	matrix 1-RF	matrix 4-RF	PCA 4-RF	Sparse PCA 4-RF	fPCA 4-RF
training	2 min	2 min	30 min	2 hours	$<$ 1 min	16 min	$<$ 1 min
prediction	5 ms	8 ms	144 ms	484 ms	44 ms	412 ms	436 ms

time	k-NN	KRR	DF	CNN	multi LSTM	hybrid	aggre-gated
training	1 sec	1 sec	21 min	6 hours	33 min	5 sec (+)	4 min (+)
prediction	1 ms	$<$ 1 ms	26 ms	3 ms	27 ms	40 ms	6 sec

Table 2.6: Computation times obtained with all methods presented above
Hybrid and aggregated with KRR, 4-RF, CNN and multi LSTM
(+): *add expert training times*

Chapter 3

Theoretical guarantees for functional expert aggregation

Remark. The work presented in this chapter was carried out in collaboration with Hugo Chardon and Étienne Donier-Meroz. It will result in the publication of an article. We contributed equally to the design and writing.

Contents

3.1 Introduction

3.1.1 Motivation

In the previous chapter, we demonstrated how to construct a surrogate model for Renault's digital simulator. In particular, we tested an expert aggregation approach, which exhibited good performances, as described in Section 2.7. Expert aggregation involves linearly combining predictors, referred to as experts. In our case, the experts are time series, and as some experts may be better at some times and worse at others, we want to aggregate these with weights that vary over time. Standard methodologies estimate experts' weights by discretizing both time series and weights, like the exponentially weighted aggregation algorithm performing a sequential optimization [Dalalyan and Tsybakov, 2008]. Although discretization is a necessary numerical procedure, it should not be included in the theoretical analysis of algorithms. Consequently, the results obtained are independent of the step size or discretization method used. My thesis presents an additional challenge due to the necessity of predicting the entire time series from a single, non-temporal parameter. This differs from the majority of time series prediction papers, which typically focus on predicting the future from the past.

Our approach is different as we consider time series and experts' weights as continuous functions in time and aim to predict the whole time series based on a non-temporal input parameter. This results in the definition of a new framework with specific input and output spaces, model, loss, and risk designed to generalize linear expert aggregation to functional predictors and functional coefficients.

Although the concrete problem of my thesis has strongly motivated this chapter, it is no less general. The proposed framework can be used in a wide range of contexts across various domains, including physics, economics, finance, social sciences, biology, epidemiology, and so forth, provided that the data being analyzed involves continuous functions [Duffie, 2010, Gabaix, 2009, Keeling and Eames, 2005, Peskin, 2018, Real and Biek, 2007]. This functional aggregation setting provides a framework for forecasting continuous functions in their entirety without relying on temporal dependencies. The simulated Renault data presented in my thesis are a good example. The goal is to predict the trajectory of a physical object through space and time based on its intrinsic properties, such as its mass, and environmental factors, such as friction coefficients. But there are plenty of other examples, like in the social sciences. For instance, one may seek to explain a country's gross domestic product over time based on demographic, economic, and geographic indicators [Tümer and Akkuş, 2018]. Similarly, in biology, one would wish to predict the evolution of a particular species' population based on its reproduction rate, food availability, and other factors [Elith and Leathwick, 2009].

The introduction of this continuous component must be rigorously addressed to demonstrate that algorithms still converge under the right conditions. We now define the formal framework within which we will work.

3.1.2 Context and objectives

Many problems in modern data science and statistics involve data that vary over a continuous domain, as demonstrated in Cuevas [2014]. Such data may encapsulate the dynamics of a continuous process varying over a specific time range or spatial patterns in multidimensional imaging data. For the sake of clarity, we focus in this work on the case of a univariate continuous parameter that one can think of as time.

Usually, a time series consists of a finite set of observations $(Y(t_1), \ldots, Y(t_N))$, representing measurements from a continuous-time process $Y(t)$. Each observation $Y(t_j)$ is the value of the process Y at specific times t_j, for $1 \leq j \leq N$. While the number of measurements defines the apparent dimension of the data, this dimension may not accurately reflect the complexity of the learning task. It is essential to develop statistical guarantees for predictors of functional data that are independent of the measurement methods used. That is why we instead aim to predict functional outputs based on finite-dimensional covariates using a framework of predictor aggregation.

Another distinctive component of our work is that in contrast to the extensive body of literature on time series forecasting [Basu and Michailidis, 2015, Brockwell and Davis, 2002, Hamilton, 1994], we focus on predicting outputs as fully functional objects. This implies that temporal causality should not be a factor in our setting. In the case of a variable $Y(t)$ with t ranging in, e.g., $[0, 1]$, the objective is to return an estimator $\widehat{Y}$ of Y such that the values $Y(s)$ are not used as regressors to predict the values $\widehat{Y}(t)$ for $t \geq s$. The objective is to compute a predictor $\widehat{Y}$, which predicts the value of $Y(t)$ for all t based on a non-temporal input parameter denoted X.

Let us now develop a regression framework that encompasses the above-mentioned modeling examples.

3.1.3 A new regression framework

In the following, we define a framework that fits all the examples described above. This allows us to propose a new type of pseudo-linear expert aggregation.

Pseudo-linear expert aggregation

In our Renault digital simulator context, we consider data (x, y) where $x \in \mathbb{R}^p$ is a non-temporal input parameter and $y \in \mathcal{S}$ is the output as a function of time. Here, we take $\mathcal{S} = C^2([0,1], \mathbb{R})$, which denotes the set of real-valued, twice continuously differentiable functions on the interval $[0,1]$.

We want to aggregate predictors with weights varying over time. For this, let us introduce some notations. Maps from $\mathbb{R}^p$ to $\mathcal{S}$ will be called *functionals*. If f denotes such a functional, we denote by f_x the function

$$\begin{array}{rccc} f_x : & [0,1] & \to & \mathbb{R} \\ & t & \mapsto & f(x)(t)\,. \end{array}$$

Let $M \geq 1$ and $f_1, \ldots, f_M$ be M functionals $\mathbb{R}^p \to \mathcal{S}$, that we call the *experts*. Let $\mathcal{W}$ be a subspace of twice continuously differentiable maps $[0,1] \to \mathbb{R}^M$. Let $\mathcal{F} = \{F_W, W \in \mathcal{W}\}$ denote the set of aggregates, that are the functionals $F_W : \mathbb{R}^p \to \mathcal{S}$ defined as the aggregation of the experts f_i by the weights w_i, for every $x \in \mathbb{R}^p$, by

$$F_{W,x} : t \in [0,1] \mapsto \sum_{j=1}^{M} w_j(t) f_{j,x}(t)\,,$$

Here, $W(t) = (w_1(t), \ldots, w_M(t)) \in \mathcal{W}$.

As detailed later in Section 3.1.4, this notion of aggregation with non-constant weights generalizes that of linear aggregation or aggregation of experts by allowing the weights to be functions instead of real numbers.

Loss selection

As mentioned above, we consider a set $\mathcal{S}$ of twice continuously differentiable functions of time. This set is endowed with the following L^2-norm

$$\forall y \in \mathcal{S} \qquad \|y\|_{L^2}^2 = \int_0^1 y(t)^2 \mathrm{d}t\,.$$

For any $W \in \mathcal{W}$ and observation $z = (x, y)$, the loss $\ell(W, z)$ of the aggregate F_W at the observation z is defined by

$$\ell(W, z) = \left\|F_{W,x} - y\right\|_{L^2}^2 = \left\|\sum_{j=1}^{M} w_j f_{j,x} - y\right\|_{L^2}^2.$$

We aim at minimizing the risk function $\mathcal{R}(W) = \mathbb{E}\|F_{W,X} - Y\|_{L^2}^2$, where the expectation is taken with respect to the joint distribution of the random vector X and random function Y.

3.1.4 Related work

Expert aggregation. The studies of Stoltz [2010] and Devaine et al. [2013] summarize some fundamental results in the field of sequential prediction of individual sequences with expert advice. Both agree that the existing theoretical research on expert aggregation is limited. However, motivated by their performance in numerical applications [Du et al., 2022, Lepikhin et al., 2020, Shazeer et al., 2017], all the studies - to the best of our knowledge - solely focused on the case of sequential forecasting, where the values of $Y(t)$ and $\widehat{Y}(t)$ are discretized. On the other hand, we have not identified articles addressing situations in which time series and experts' weights are considered continuous time functions.

Linear aggregation. Given basis functions $\varphi_1, \ldots, \varphi_d : \mathcal{X} \to \mathbb{R}$, linear aggregation [Juditsky and Nemirovski, 2000, Nemirovski, 2000] intends to find the best predictor of the form $f_\theta(x) = \sum_{j=1}^{d} \theta_j \varphi_j(x) = \langle \theta, x' \rangle$, where $\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$ are the weights and $x' = (\varphi_1(x), \ldots, \varphi_d(x)) \in \mathbb{R}^d$.

As mentioned in Section 3.1.3, the linear aggregation framework does not correspond to our work as the weights θ_j are scalars. In our case, the weights vary and depend on time. This implies that the concept of linearity cannot be applied, which blocks the use of the linear aggregation framework described above.

Finally, we examined *functional regression*, which is also analogous to our approach and may serve as a general framework for our work. The following paragraph presents various learning tasks that transform classical linear regression into functional regression.

From linear regression to functional linear regression. Linear regression is the most widely used statistical tool to model the dependence of a quantitative outcome variable on multivariate features. It fits into the realm of supervised statistical learning, with the square loss $\ell(y, y') = (y - y')^2$, feature space $\mathcal{X} = \mathbb{R}^d$, output space $\mathcal{Y} = \mathbb{R}$ and the class of comparison $\mathcal{F} = \{f_\theta : x \mapsto \langle \theta, x \rangle, \theta \in \mathbb{R}^d\}$ being the set of all linear maps from $\mathbb{R}^d$ to $\mathbb{R}$. As a first evolution, linear regression can easily be generalized to the case of a multivariate outcome $Y \in \mathbb{R}^q$, $q \geq 1$, in which case the loss is the Euclidean distance on $\mathbb{R}^q$ and the set of all possible linear predictors is identified to the space of matrices of size $q \times d$ [Giraud, 2021]. Going further, functional regression is a recent and extensively studied topic [Aneiros et al., 2022, Greven and Scheipl, 2017, Kalogridis and Van Aelst, 2019] that allows optimizing over functional spaces like the space L_2. The functional framework is obtained using the linear regression problem with an infinite-dimensional feature space $\mathcal{X}$ and real outputs $\mathcal{Y} = \mathbb{R}$. The simplest model consists of letting $\mathcal{X}$ be a separable Hilbert space $\mathcal{H}$. The class of comparison is the set of all linear maps from $\mathcal{H}$ to $\mathbb{R}$. They can be written as $x \mapsto \langle \theta, x \rangle$ for some $\theta \in \mathcal{H}$, where $\langle \cdot, \cdot \rangle$ is the inner product on $\mathcal{H}$. The shift to this

infinite-dimensional covariates setting has mainly been studied in the literature on reproducing kernel Hilbert spaces (RKHSs) [Steinwart and Christmann, 2008]. An RKHS is a Hilbert space of functions in which point evaluation is a continuous linear functional.Due to their construction, they are well-suited to the context of functional data. In the literature, we find functional regression [Yuan and Cai, 2010] and functional classification [Berrendero et al., 2018]. Point-evaluation function values are often used to construct a function in the classical RKHS setting. However, as previously mentioned, there is a large amount of data that are not point-evaluation function values. Wang and Xu [2019] have defined functional RKHS to process non-point-evaluation functional data.

Nevertheless, concerning functional regression, as the considered output space is $\mathbb{R}$, this framework is not sufficiently generic to encompass our work. We need to consider a functional output space. This approach must be extended in order to accommodate the additional considerations. Concerning RKHS, a linear function is required, which is not our case, functional weights prevent the linearity property. RKHSs are not adapted to our problem.

The following paper presents a more generic novel approach.

Function-on-function linear regression. The recent work of Benatia et al. [2017] proposes an innovative approach to estimate functional regression models where both the regressor $Z(t)$ and the response $Y(t)$ are functions in possibly infinite-dimensional Hilbert spaces. The authors extend the traditional multivariate regression model by treating the regression coefficient as an unknown operator Π. They introduce Tikhonov regularization to estimate the regression coefficient, applying a penalty on the L_2-norm of the coefficient under the assumption that the relationship between the response and the regressor is not sparse. This method differs from traditional L_1 penalties, such as those used in Lasso. The paper derives the rate of convergence of the mean squared error and establishes the asymptotic distribution of the estimator, providing theoretical solid foundations for the estimation process.

Although this innovative work is quite generic, it still does not provide a framework in which our model can be included. Our coefficients are functional, so our predictors are not linear functions from $\mathbb{R}^p$ to $\mathcal{H}$. Consequently, we are unable to apply the conventional framework of linear regression, whether extended or not. It is, therefore, necessary to describe a pseudo-linear regression with functional coefficients and functional outputs. To the best of our knowledge, this context has not been identified in any published paper.

		Input space	
		Finite	**Infinite**
Output space	**Finite**	linear regression	functional regression
	Infinite	our work (pseudo-linear)	Benatia et al. (linear)

Table 3.1: Summary of the different presented frameworks according to the input and output spaces considered

3.1.5 Organization of the chapter

In Section 3.2, we provide an overall description of the framework and verify that smoothness and strong convexity are true for the chosen loss function. Moving on to Section 3.3, we discuss the application and the convergence rate of deterministic projected gradient descent (PGD). In Section 3.4, the evaluation of a stochastic approach with PGD and mirror gradient descent (MGD) is presented. Finally, Section 3.5 concludes and puts the various results into perspective.

3.2 Framework overall description

Consider an unknown functional denoted as F^*, defined as follows

$$\begin{array}{rccc} F^* : & \mathbb{R}^p & \to & \mathcal{S} \\ & x & \mapsto & F^*(x) =: F^*_x \,, \end{array} \tag{3.2.1}$$

which maps non-temporal input values x to output time series F^*_x. The output time series are considered as functions of time and therefore belong to $C^2([0,1])$. In my thesis context, F^* corresponds to Renault's digital simulator, which, for a given input parameter, generates output time series.

To estimate F^*, we consider estimators constructed from an aggregation of M experts, denoted as $f_1, \ldots, f_M$, which are functions mapping from $\mathbb{R}^p$ to $\mathcal{S}$. The experts can be taken from the predictors proposed and constructed in Chapter 2. The class of functions for comparison is defined by

$$\mathcal{F} = \left\{ \sum_{j=1}^{M} w_j f_j \ , \ W : [0,1] \to \Delta^M \right\},$$

where, for all t, $W(t) = (w_1(t), \dots, w_M(t))^\top$, and the functions w_j correspond to the j-th coordinate of W, representing the time varying weights. Here, $\Delta^M = \{u \in \mathbb{R}_+^M,\ \sum_j u_j = 1\}$ denotes the simplex of $\mathbb{R}^M$. The notation $w_j f_j$ denotes the functional

$$\begin{array}{rccl} w_j f_j : & \mathbb{R}^p & \to & \mathcal{S} \\ & x & \mapsto & w_j f_j(x) =: w_j f_{j,x} \end{array}$$

where $w_j f_{j,x}$ is a function of time.

3.2.1 First definitions and propositions

The variable $W \in \mathcal{W}$ represents a weight vector consisting of M functions $w_j : [0,1] \to \mathbb{R}^+$. For all $t \in [0,1]$ and $1 \leq j \leq M$, $w_j(t)$ denotes the weight assigned to expert j at time t. We aim to embed the space in which W resides into a Hilbert space, enabling the use of convex optimization techniques.

Embedding. Let us consider a function $U : [0,1] \to \mathbb{R}^M$ that is square-integrable. We denote $U \in L^2([0,1], \mathbb{R}^M)$ if each coordinate function is square-integrable, i.e., $u_j \in L^2([0,1], \mathbb{R})$ for all j. For the sake of simplicity, we denote $\mathcal{H} := L^2([0,1], \mathbb{R}^M)$. Then, for any $U, V \in \mathcal{H}$, we define the following inner product

$$\langle U, V \rangle_{\mathcal{H}} := \int_0^1 \langle U(t), V(t) \rangle_2 \, \mathrm{d}t = \int_0^1 \sum_{j=1}^M u_j(t) v_j(t) \, \mathrm{d}t \tag{3.2.2}$$

where $\langle \cdot, \cdot \rangle_2$ denotes the standard Euclidean inner product on $\mathbb{R}^M$. We denote $\| \cdot \|_{\mathcal{H}}$ as the associated norm.

Proposition 3.1. *The space $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is a Hilbert space.*

The proof of this result can be found in Section 3.A.1.

Constraints. Throughout, the weights must always sum up to 1, implying that the weight functions W need to be simplex-valued. Let us defined

$$\mathcal{C} = \{W \in \mathcal{H},\ W([0,1]) \subset \Delta^M\}\ , \tag{3.2.3}$$

where Δ^M represents the simplex in $\mathbb{R}^M$. Hence, $\mathcal{C}$ constitutes the space of paths on Δ^M. It is evident that $\mathcal{C}$ is convex. The subsequent result demonstrates that it is also closed concerning the Hilbertian norm $\| \cdot \|_{\mathcal{H}}$ defined in (3.2.2).

Proposition 3.2. *The set $\mathcal{C}$ defined by* (3.2.3) *is closed in $\mathcal{H}$ for the norm $\|\cdot\|_{\mathcal{H}}$ associated with the inner product* (3.2.2)*.*

The proof of this result can be found in Section 3.A.2.

Loss function and regression problem.

Problem formulation. Let X be a random variable with distribution P. We want to solve the optimization problem

$$\min_{W\in\mathcal{C}}\Big\{\,\mathcal{R}(W) := \mathbb{E}_P[\ell(W,X)]\,\Big\} \tag{3.2.4}$$

where the loss function ℓ is defined by

$$\ell(W,x) = \frac{1}{2}\big\|W^\top F_x - f_x^*\big\|_{L^2}^2 = \frac{1}{2}\int_0^1 \big(W^\top F_x(t) - f_x^*(t)\big)^2\,\mathrm{d}t\,.$$

For all $x\in\mathbb{R}^p$, we used the notation $F_x := F(x) = \big(f_1(x),\dots,f_M(x)\big)^\top$ for the functional vector of experts by reusing the notation introduced in (3.2.1). We denote by W^* its constrained minimizer

$$W^* = \operatorname*{arg\,min}_{W\in\mathcal{C}} \mathcal{R}(W)\,,$$

which is well-defined since the population loss is strictly convex, as demonstrated in the following section.

3.2.2 Regularity of the loss function

Let us first compute the gradient and the Hessian of the considered loss function. The following results define them.

Proposition 3.3 (gradient and Hessian of the loss)**.** *For all $W\in\mathcal{H}$ and $x\in\mathbb{R}^p$, the gradient and Hessian of ℓ are given by*

$$\nabla_W \ell(W,x) = (W^\top F_x - f_x^*)F_x\,,$$
$$\nabla_W^2 \ell(W,x) = F_x F_x^\top\,.$$

Proof. Let $W, H \in \mathcal{H}$ and $x \in \mathbb{R}^p$. Then

$$\begin{aligned}\ell(W+H,x) &= \frac{1}{2}\left\|W^\top F_x + H^\top F_x - f_x^*\right\|_{L^2}^2 \\ &= \ell(W,x) + \langle H^\top F_x, W^\top F_x - f_x^*\rangle_{L^2}^2 + \frac{1}{2}\left\|H^\top F_x\right\|_{L^2}^2 \\ &= \ell(W,x) + \int_0^1 H(t)^\top F_x(t)\left(W(t)^\top F_x(t) - f_x^*(t)\right)\mathrm{d}t + \frac{1}{2}\underbrace{\left\|H^\top F_x\right\|_{L^2}^2}_{=H^\top F_x F_x^\top H}.\end{aligned}$$

So, for a given x, the gradient is

$$\nabla_W \ell(W,x) = (W^\top F_x - f_x^*)F_x$$

which is a vector of size M of functions of $\mathcal{S}$.
And for a given x, the Hessian is

$$\nabla_W^2 \ell(W,x) = F_x F_x^\top$$

which is a matrix of size $M \times M$ of functions of $\mathcal{S}$. □

To compute the minimizer of the problem (3.2.4), our objective is to demonstrate that $\mathcal{R}$ possesses favorable regularity properties. Specifically, we aim to show that it is smooth and strongly convex. We first recall the corresponding definitions and check that the loss function (3.2.4) satisfies them.

Definition 3.1 (strong convexity). *A function F from a Hilbert space $\mathcal{H}$ to $\mathbb{R}$ is μ-strongly convex on a convex set C if for all $x, y \in C$,*

$$F(y) \geq F(x) + \left\langle \nabla F(x), y - x\right\rangle_{\mathcal{H}} + \frac{\mu}{2}\|y-x\|_{\mathcal{H}}^2.$$

Definition 3.2 (smoothness). *A function $F : \mathcal{H} \to \mathbb{R}$ is ν-smooth if its gradient ∇F is a ν-Lipschitz map from $\mathcal{H}$ to $\mathcal{H}$, ie for all $x, y \in \mathcal{H}$,*

$$\left\|\nabla F(x) - \nabla F(y)\right\|_{\mathcal{H}} \leq \nu\|x-y\|_{\mathcal{H}}.$$

Proposition 3.4 (regularity). *Under the assumption that the experts and the distribution of X satisfy $\mathbb{E}\left[F_X F_X^\top\right] = I_M$, the risk $\mathcal{R}$ is μ-strongly convex and ν-smooth with $\mu = 2$ and $\nu = M$.*

Proof. As $\mathbb{E}\left[F_X F_X^\top\right] = I_M$, the differentiation theorem under the integral sign gives $\nabla_W^2 \mathcal{R}(W) = \mathbb{E}\left[F_X F_X^\top\right] = I_M$, and it then shows that $\mathcal{R}$ is μ-strongly convex with $\mu = 2$. In

order to prove that $\mathcal{R}$ is ν-smooth, let $W_1, W_2 \in \mathcal{H}$. Then

$$\nabla \ell(W_1, x) - \nabla \ell(W_2, x) = \big((W_1 - W_2)^\top F_x\big) F_x = F_x F_x^\top (W_1 - W_2)\,.$$

By injecting it into the calculation

$$\begin{aligned}
\big\|\nabla \mathcal{R}(W_1) - \nabla \mathcal{R}(W_2)\big\| &= \Big\|\mathbb{E}\big[\nabla \ell(W_1, X) - \nabla \ell(W_2, X)\big]\Big\| \\
&= \Big\|\mathbb{E}\big[F_X F_X^\top (W_1 - W_2)\big]\Big\| \\
&\leq \mathbb{E}\Big[\big\|F_X F_X^\top (W_1 - W_2)\big\|\Big] \\
&\leq \mathbb{E}\Big[\big\|F_X F_X^\top\big\|_{\mathsf{op}}\Big] \, \|W_1 - W_2\| \\
&= \mathbb{E}\big[\|F_X\|_{\mathcal{H}}^2\big] \, \|W_1 - W_2\|\,.
\end{aligned}$$

By assumption

$$\mathbb{E}\big[\|F_X\|_{\mathcal{H}}^2\big] = \mathbb{E}\bigg[\int_0^1 \sum_{j=1}^M f_{j,X}(t)^2 \mathrm{d}t\bigg] = \mathrm{Tr}\left(\mathbb{E}[F_X F_X^\top]\right) = M$$

So $\nabla \mathcal{R}$ is ν-Lipschitz that is $\mathcal{R}$ is ν-smooth with $\nu = M$. □

The condition $\mathbb{E}\big[F_X F_X^\top\big] = I_M$ means that the experts are uncorrelated and normalized to have variance one. Additionally, we can assume that they are bounded in absolute value, allowing us to provide a more precise upper bound.

3.3 Deterministic optimization

In the next section, we address the deterministic case, assuming initially that we have access to an oracle of order 0 and 1, allowing us to evaluate both $\mathcal{R}$ and $\nabla \mathcal{R}$ for all W. This assumption will be relaxed in Section 3.4, where we move on to the more general stochastic case.

Let $\mathcal{H}$ be our Hilbert space, as mentioned earlier, with its induced norm $\|\cdot\|_{\mathcal{H}}$ given by equation (3.2.2). We consider the optimization problem defined by equation (3.2.4), where $\mathcal{R}$ is a convex function defined on $\mathcal{H}$, and $\mathcal{C} \subset \mathcal{H}$ is a closed convex subset of $\mathcal{H}$. The standard orthogonal projection operator onto $\mathcal{C}$ is given by

$$\Pi_{\mathcal{C}}(g) = \operatorname*{arg\,min}_{W \in \mathcal{C}} \|g - W\|_{\mathcal{H}}\,.$$

Using various versions of gradient descent, the convergence rate will depend on the regularity properties of $\mathcal{R}$: Lipschitz continuity, smoothness, convexity, or strong convexity.

We examined the projected gradient descent (PGD) algorithm, where the k-th iteration with step size η is defined as

$$W_{k+1} = \Pi_{\mathcal{C}}\big(W_k - \eta \nabla \mathcal{R}(W_k)\big)\,. \tag{3.3.1}$$

The objective is to establish the algorithm's convergence at an exponential rate within our specific framework. The proof provided in Bubeck [2015] for the case where the objective function is defined in finite-dimensional space can be adapted to our framework with minimal modifications. We rewrite the proof for the sake of clarity and to unify notations. It is necessary to demonstrate that the function $\mathcal{R}$ is both strongly convex and smooth. This enables the application of the result given in Theorem 3.3 of Section 3.A.3. The main result is as follows.

Theorem 3.1. *With $(W_k)_{k\geq 0}$ defined by (3.3.1) and under the assumptions of Proposition 3.4, for all $k \geq 0$,*

$$\|W_k - W^*\|_{\mathcal{H}}^2 \leq \exp\left(-\frac{2k}{M}\right) \|W_0 - W^*\|_{\mathcal{H}}^2\,.$$

This ensures an exponential convergence rate when implementing projected gradient descent with this particular class of functions $\mathcal{F}$.

Proof. By Proposition 3.4, $\mathcal{R}$ is μ-strongly convex and ν-smooth with $\mu = 2$ and $\nu = M$. In addition $\mathcal{C}$ is a close and convex subset of $\mathcal{H}$, so by Theorem 3.3 (given and proved in Section 3.A.3), we conclude that

$$\|W_k - W^*\|_{\mathcal{H}}^2 \leq \left(1 - \frac{2}{M}\right)^k \|W_0 - W^*\|_{\mathcal{H}}^2\,.$$

Bounding from above $\left(1 - \frac{2}{M}\right)^k = \exp\left(k \log\left(1 - \frac{2}{M}\right)\right) \leq \exp\left(-\frac{2k}{M}\right)$ yields the expected exponential rate. □

3.4 Stochastic optimization

In the deterministic case, we assumed that we had access to order 0 and 1 oracles, which means that we could compute $\mathcal{R}(W)$ and $\nabla\mathcal{R}(W) = \mathbb{E}[\nabla \ell(W, X)]$ for all W. As a reminder, here W denotes the expert weights, and X is the non-temporal input values that follows a P distribution on $\mathbb{R}^p$. In most cases, having access to order 0 and 1 oracles is a restrictive assumption. That is why it is more reasonable to suppose that we can only compute order 0 oracle and have only access to $\nabla \ell(W, X)$. Please refer to Section 3 of Rigollet [2015] for a complete reading.

By considering that the values of $\mathcal{R}$ and its gradient $\nabla\mathcal{R}$ are not universally available across all W, the adoption of a stochastic optimization strategy is required. The theoretical guarantees supporting this approach in our specified context are detailed subsequently.

In stochastic optimization, subgradients are essential for addressing non-differentiable convex functions. A subgradient generalizes the concept of a gradient by ensuring that for a convex function f, a vector g at point x satisfies the inequality $f(y) \geq f(x) + g^\top(y - x)$ for all y. This enables the application of gradient-based methods to non-smooth problems.

Let $(X_k)_{k=1,2,\dots}$ be an i.i.d. sample with the same distribution as $X \sim P$, where P is unknown. We now consider having access to g_k, an unbiased estimator of $\nabla\mathcal{R}(W_k)$ which satisfies $\mathbb{E}[g_k] = \nabla\mathcal{R}(W_k)$. In our case, the estimated gradients are defined by

$$g_k = \nabla\ell(W_k, X_k), \quad \forall k \geq 0\,. \tag{3.4.1}$$

This notation will be used in the following formulas.

3.4.1 Projected gradient descent

In a stochastic context, the iteration of the PGD algorithm is given by

$$W_{k+1} = \Pi_{\mathcal{C}}\big(W_k - \eta g_k\big)$$

where g_k is defined by (3.4.1).

We now introduce the following proposition, summarizing the main result of Lacoste-Julien et al. [2012]. It guarantees the linear rate convergence of the PGD in a stochastic context. All the necessary computations to prove it are provided in that paper.

Proposition 3.5. *Let $\Pi_{\mathcal{C}}$ be the orthogonal projection on $\mathcal{C}$ and let us assume that*

(i) $\mathbb{E}[g_k]$ is a subgradient of $\mathcal{R}$ at W_k,
(ii) $\mathbb{E}[\|g_k\|^2] \leq B^2$ for some $B > 0$.

Then, for all $k \geq 1$,

$$\mathbb{E}\|W_k - W^*\|_{\mathcal{H}}^2 \leq \frac{4B^2}{\mu^2(k+1)}\,.$$

The estimated gradient g_k given by (3.4.1) satisfies Assumption (i) of Proposition 3.5. Indeed, we can compute the expectation with respect to P as follows

$$\begin{aligned}\mathbb{E}_P[g_k] &= \mathbb{E}_P[\nabla \ell(W_k, X_k)] \\ &= \nabla \mathbb{E}_P[\ell(W_k, X_k)] \\ &= \nabla \mathcal{R}(W_k)\end{aligned}$$

where X and all the X_k's follow the distribution P. Concerning Assumption (ii), it is often reasonable to assume that the variance of stochastic gradients g_k is bounded. In our case, this assumption remains plausible due to the properties of our objective function, the risk $\mathcal{R}$, which is μ-strong convex and ν-smooth.

The Proposition 3.5 remains valid in our context because it relies solely on the well-definedness of the Euclidean projection $\Pi_{\mathcal{C}}$, given that $\mathcal{C}$ is closed and convex.

3.4.2 Mirror gradient descent

One natural approach is to utilize mirror gradient descent (MGD) in the context of constrained optimization, particularly in the context of simplex. Nevertheless, the transition from constant coefficients to functional coefficients is not as straightforward as it is with PGD. The objective of this section is to provide motivation and demonstrate the potential of this method for addressing our problem. However, the work is still in progress.

MGD is an optimization technique tailored for non-Euclidean geometries like the ℓ_1-ball. It often outperforms traditional methods such as PGD, which operates within the Euclidean (or ℓ_2-) ball. This enhanced efficiency is primarily attributed to its utilization of Bregman divergences, which better capture the structure of specific problems, particularly those involving sparsity. For an extensive examination of MGD, including its algorithmic framework and comparisons with alternative methods, Chapter 4 of Bubeck [2015] is highly recommended. This chapter comprehensively analyzes the method's theoretical aspects and practical applications, highlighting its adaptability and advantages in complex optimization scenarios.

Generalities on mirror gradient descent

Before implementing MGD, it is imperative to establish a mirror map Φ, as defined by Definition 3.3. This mirror map serves as a convex function pivotal in shaping the geometry of the optimization process. Once Φ is defined, the associated Bregman divergence D_Φ can be computed. This divergence, as defined by Definition 3.4, quantifies the distance between points in the space transformed by Φ.

Definition 3.3 (Mirror map)**.** *Let $\mathcal{D} \subset \mathcal{H}$ be a convex open set such that $\mathcal{C}$ is contained in its closure, that is $\mathcal{C} \subset \overline{\mathcal{D}}$ and $\mathcal{C} \cap \mathcal{D} \neq \emptyset$. We say that $\Phi : \mathcal{D} \to \mathbb{R}$ is a mirror map if it satisfies the following properties*

(i) Φ is ρ-strongly convex and differentiable.

(ii) The gradient of Φ takes all possible values.

(iii) The gradient of Φ diverges on the boundary of $\mathcal{D}$, that is

$$\lim_{x \to \partial\mathcal{D}} \|\nabla\Phi(x)\|_{\mathcal{H}} = +\infty\,.$$

Definition 3.4 (Bregman divergence)**.** *For given differentiable, α-strongly convex function $\Phi : \mathcal{D} \to \mathbb{R}$, we define the Bregman divergence associated with Φ to be*

$$D_\Phi(y, x) = \Phi(y) - \Phi(x) - \nabla\Phi(x)^\top (y - x)\,. \tag{3.4.2}$$

Given the mirror map Φ and its corresponding Bregman divergence D_Φ, one iteration k of the MGD algorithm is described in Figure 3.1 and defined by these following four steps.

One iteration k of the mirror gradient descent algorithm

1. Map W_k to the dual space $p_k := \nabla\Phi(W_k)$,
2. Update in the dual space using a gradient step

$$\nabla\Phi(W'_{k+1}) = p_k - \eta g_k \tag{3.4.3}$$

   with g_k defined by (3.4.1),
3. Map back $\nabla\Phi(W'_{k+1})$ to the primal space by applying $(\nabla\Phi)^{-1}$,
4. Project back to the feasible region $\mathcal{C}$ by computing

$$W_{k+1} = \arg\min_{W \in \mathcal{C}} D_\Phi(W, W'_{k+1})\,. \tag{3.4.4}$$

We observe that one can rewrite an MGD iteration as follows

$$\begin{aligned} W_{k+1} &= \arg\min_{W \in \mathcal{C}} \; D_\Phi(W, W'_{k+1}) \\ &= \arg\min_{W \in \mathcal{C}} \; \Phi(W) - \nabla\Phi(W'_{k+1})^\top W \\ &= \arg\min_{W \in \mathcal{C}} \; \Phi(W) - \big(\nabla\Phi(W_k) - \eta g_k\big)^\top W \\ &= \arg\min_{W \in \mathcal{C}} \; \eta g_k^\top W + D_\Phi(W, W_k)\,. \end{aligned}$$

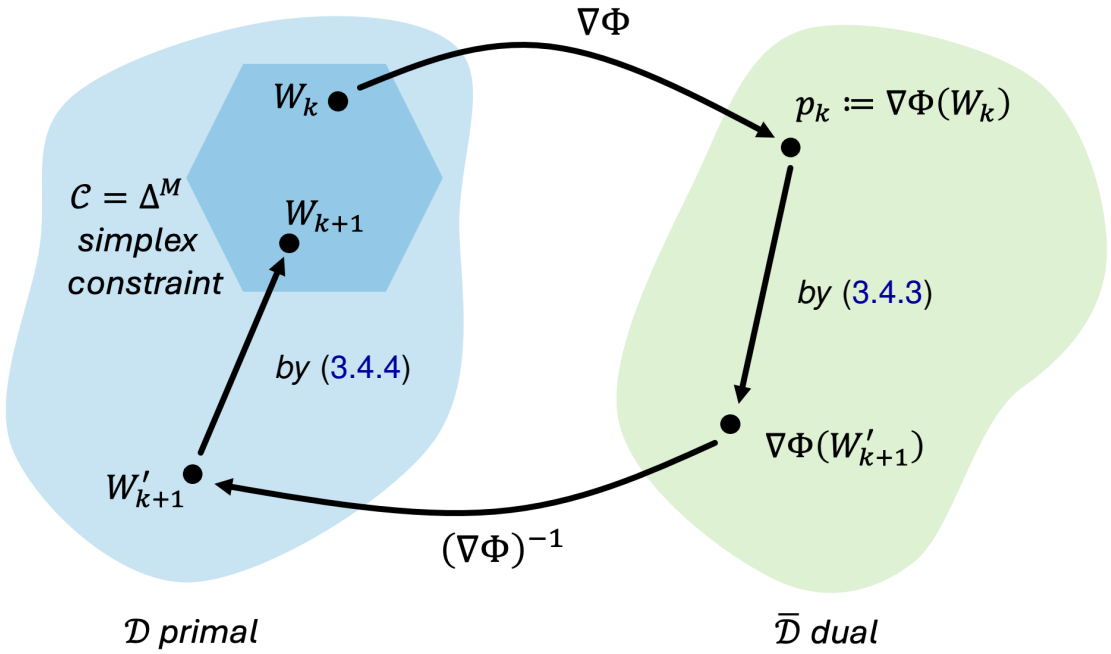


Figure 3.1: Diagram of one mirror gradient descent iteration

Application to our case

Let us consider the negative entropy defined by

$$\Phi(x) = \sum_{j=1}^{M} x_j \log(x_j)$$

which satisfies all the assumptions of Definition 3.3. It then can be used as a mirror map, and we compute the associated Bregman divergence (3.4.2)

$$D_\Phi(x, y) = \sum_{j=1}^{M} x_j \log \frac{x_j}{y_j}$$

which corresponds to the Kullback-Leibler (KL) divergence denoted $\mathrm{KL}(x, y) := D_\Phi(x, y)$.

As Properties (i) and (iii) of Definition 3.3 are satisfied and since we need to project on the simplex, which is a closed set, this guarantees the existence and uniqueness of the minimizer

$$\Pi_{\mathcal{C}}^{\mathrm{KL}}(y) := \arg\min_{x \in \mathcal{C}} \mathrm{KL}(x, y) \tag{3.4.5}$$

which allows the projection (3.4.4) into the constraint space $\mathcal{C}$.

Why consider negative entropy?

> **Proposition 3.6.** *The Kullback-Leibler divergence projection on the simplex amounts to a simple ℓ_1-normalization $y \mapsto y/\|y\|_1$.*

The proof is taken from Section 2.4.4, page 76 of Rigollet [2015].

Proof. The Lagrangian of (3.4.5) is given by

$$\mathcal{L}(x, y) = \sum_{j=1}^{M} x_j \log\left(\frac{x_j}{y_j}\right) + \sum_{j=1}^{M}(y_j - x_j) + \lambda\left(\sum_{j=1}^{M} y_j - 1\right).$$

By solving $\partial\mathcal{L}/\partial y_j = 0$, we obtain $y_j = \eta x_j$, $\forall j = 1, \dots, M$. As $x \in \mathcal{C} \cap \mathcal{D}$, then $\sum_{j=1}^{M} x_j = 1$ and so $\eta = \left(\sum_{j=1}^{M} y_j\right)^{-1}$. By injecting these results in the back projection step (3.4.4), we obtain

$$W_{k+1} = \frac{W'_{k+1}}{\|W'_{k+1}\|_1}.$$

□

This choice significantly simplifies Step 4. of the MGD algorithm by reducing it to a simple ℓ_1-normalization step. This not only streamlines the computational process but also ensures that the solution adheres to the geometric constraints of the simplex, which is critical for many applications in statistical and probabilistic analysis.

Application to the functional setting. The convergence rate for an MGD iteration is outlined under the assumption of convex and Lipschitz properties of the objective function $\mathcal{R}$, as detailed in Theorem 4.2., page 299 of Bubeck [2015]. This result is summarized in the Theorem 3.2. The proof is given in Bubeck [2015].

In our case, the negative entropy is ρ-strongly convex with $\rho = 1$. Please refer to pages 6-7 of Rigollet [2015] for the demonstration of the strong convexity of the negative entropy.

> **Theorem 3.2.** *Let Φ be a mirror map ρ-strongly convex, $R^2 = \sup_{W \in \mathcal{C}} \Phi(W) - \Phi(W_1)$, and $\mathcal{R}$ be convex and ν-Lipschitz. Then, stochastic mirror descent with $\eta = \frac{R}{\nu}\sqrt{\frac{2\rho}{K}}$ satisfies*
>
> $$\mathcal{R}\left(\frac{1}{K}\sum_{k=1}^{K} W_k\right) - \mathcal{R}(W^*) \leq R\nu\sqrt{\frac{2}{\rho K}}.$$

However, we have previously demonstrated that the objective function $\mathcal{R}$ is not only convex but μ-strongly convex. Moreover, the Lipschitz property will have to be changed to a smooth property. It is reasonable to posit that the final rate can be improved by rewriting the end of the demonstration with the integration of the strong convexity assumption and the modification from Lipschitz to smooth property. This work is still in progress, and the final optimum convergence rate has not yet been determined.

Moreover, in our case, the situation is somewhat more complex because, in reality, the weights are not scalar but functions of time. Consequently, choosing the negative entropy might not necessarily be a good choice. Instead of projecting onto a *simple* simplex, we are projecting onto a *functional* simplex, which may require modifying the considered mirror map, if not more. This aspect is currently under study.

3.5 Conclusion

We have illustrated the effectiveness of these optimization methods in a context of functional prediction involving an aggregation of experts with functional weights. Moreover, this mathematical formulation allows us to approach the problem from a different perspective, enabling the exploration of alternative methods that might not have been initially considered.

In conclusion, the deterministic case obviously exhibits a higher convergence rate than the stochastic case. Among the stochastic approaches, PGD currently demonstrates a better rate; however, we are confident that MGD will achieve comparable performance when incorporating the strong convexity hypothesis.

Lastly, our goal is to accurately delineate the MGD in the functional context and ensure its optimal functionality and the consistency of the associated tools. This will enable us to integrate the strong convexity assumption into the stochastic MGD bound. Once all the theoretical parts are dealt with, we aim to conduct experiments to evaluate the numerical effectiveness of these approaches.

	determinist PGD	**stochastic PGD**	**stochastic MGD**
regularity properties	μ-strong convex ν-smooth	μ-strong convex ν-smooth	convex ν-Lipschitz
convergence rate	$O\big(\exp(-K)\big)$	$O(K^{-1})$	$O(K^{-1/2})$

Table 3.2: Final convergence rates obtained with each PGD method

Stochastic MGD still in progress

3.A Appendix: proofs of Chapter 3

3.A.1 Proof of Proposition 3.1

Proof of Proposition 3.1. Let us consider the application

$$(u, v) \in \mathcal{H}^2 \mapsto \langle u, v \rangle = \int_0^1 u(t)^\top v(t) \mathrm{d}t \,.$$

This defines an inner product on $\mathcal{H}$. All properties are inherited from the inner products on $L^2([0,1])$ and on $\mathbb{R}^M$, respectively

$$\langle f, g \rangle_{L^2} = \int_0^1 fg \quad \text{and} \quad \langle u, v \rangle_2 = u^\top v \,.$$

Hence $\| \cdot \|$ as defined in (3.2.2) automatically defines a norm on $\mathcal{H}$. It remains to check that $\mathcal{H}$ is complete. Since $\mathcal{H}$ is isomorphic to $L^2([0,1])^M$, which is a product of Banach spaces, hence a Banach space.

□

3.A.2 Proof of Proposition 3.2

Proof of Proposition 3.2. Let $(f_n)_{n \in \mathbb{N}}$ be a sequence in $\mathcal{C}$, which converges to some $f \in \mathcal{H}$ for the norm $\| \cdot \|_{\mathcal{H}}$, ie

$$\int_0^1 \|f_n(t) - f(t)\|_2^2 \, \mathrm{d}t \underset{n \to +\infty}{\longrightarrow} 0 \,.$$

Then there exists a subsequence $(f_{n_k})_{k \in \mathbb{N}}$ which converges pointwise almost everywhere to f in $\mathbb{R}^M$ with respect to the Euclidean norm $\| \cdot \|_2$, meaning that for almost all $t \in [0,1]$,

$$\sum_{j=1}^M (f_{n_k}^j(t) - f^j(t))^2 \underset{k \to +\infty}{\longrightarrow} 0 \,,$$

Which in turn implies that for all j, $f_{n_k}^j(t) \underset{k \to +\infty}{\longrightarrow} f^j(t)$, so that, since all f_{n_k} are in $\mathcal{C}$,

$$\sum_{j=1}^M f^j(t) = 1 \,.$$

□

3.A.3 Deterministic PGD for smooth and strongly convex objectives

In this section, we assume the objective function F to be ν-smooth and μ-strongly convex and that we have access to an order 1 oracle of F. To deal with the constrained setting, we use the projected gradient descent (PGD) with constant step size η. The iteration is

$$\theta_{k+1} = \Pi_{\mathcal{C}}\big(\theta_k - \eta \nabla F(\theta_k)\big)\,,$$

where $\mathcal{C}$ is the space of constraints included in $\mathcal{H}$.

Actually, PGD is not a proper *descent* algorithm because the value of F does not necessarily decrease after each iteration because of the projection. Therefore we only get the convergence of the iterates $(\theta_k)_{k\geq 0}$. In the proof of convergence, we will need the following key observations.

Lemma 3.1. *For all k,*

$$\langle \nabla F(\theta_k), \theta_{k+1} - \theta^* \rangle \leq \nu \langle \theta_k - \theta_{k+1}, \theta_{k+1} - \theta^* \rangle\,.$$

Proof. Let $\tilde{\theta}_{k+1} = \theta_k - \eta \nabla g(\theta_k)$ so that $\theta_{k+1} = \Pi_{\mathcal{C}}(\tilde{\theta}_{k+1})$. If $\tilde{\theta}_{k+1} \in \mathcal{C}$, the projection changes nothing so $\theta_{k+1} = \tilde{\theta}_{k+1}$ and we have equality. Otherwise, by definition of the projection operator, since θ^*, θ_{k+1} are in $\mathcal{C}$, $\theta_{k+1} - \tilde{\theta}_{k+1}$ has negative orientation with respect to $\theta_{k+1} - \theta^*$, that is

$$\left\langle \theta_{k+1} - \theta_k + \frac{1}{\nu}\nabla F(\theta_k), \theta_{k+1} - \theta^* \right\rangle \leq 0\,,$$

and rearranging the terms yields the expected result. □

In a generic case where F is the objective function, which is ν-smooth and μ-strongly convex, and θ_k is the value obtained at the k-th step of a PGD algorithm, the following result stands.

Theorem 3.3. *Let F be an ν-smooth and μ-strongly convex function, $\mathcal{C}$ a closed convex subset of $\mathcal{H}$, and*

$$\theta^* = \arg\min_{\theta \in \mathcal{C}} F(\theta)\,.$$

Then, for all $k \geq 1$,

$$\|\theta_k - \theta^*\|^2 \leq \left(1 - \frac{\mu}{\nu}\right)^k \|\theta_0 - \theta^*\|^2\,.$$

Proof. By definition of θ^*, it holds $0 \leq F(\theta_{k+1}) - F(\theta^*)$. Writing

$$F(\theta_{k+1}) - F(\theta^*) = F(\theta_{k+1}) - F(\theta_k) + F(\theta_k) - F(\theta^*)$$

and using the fact that F is ν-smooth and μ-strongly convex, we get

$$\begin{aligned}
0 \leq F(\theta_{k+1}) - F(\theta^*) &\leq \langle \nabla F(\theta_k), \theta_{k+1} - \theta_k \rangle + \frac{\nu}{2}\|\theta_k - \theta_{k+1}\|^2 + \langle \nabla F(\theta_k), \theta_k - \theta_* \rangle - \frac{\mu}{2}\|\theta_k - \theta^*\|^2 \\
&= \langle \nabla F(\theta_k), \theta_{k+1} - \theta^* \rangle + \frac{\nu}{2}\|\theta_k - \theta_{k+1}\|^2 - \frac{\mu}{2}\|\theta_k - \theta^*\|^2 \\
&\leq \nu \langle \theta_k - \theta_{k+1}, \theta_{k+1} - \theta^* \rangle + \frac{\nu}{2}\|\theta_k - \theta_{k+1}\|^2 - \frac{\mu}{2}\|\theta_k - \theta^*\|^2 \\
&= \nu \langle \theta_k - \theta_{k+1}, \theta_k - \theta^* \rangle - \frac{\nu}{2}\|\theta_k - \theta_{k+1}\|^2 - \frac{\mu}{2}\|\theta_k - \theta^*\|^2 .
\end{aligned}$$

Therefore

$$2\langle \theta_{k+1} - \theta_k, \theta_k - \theta^* \rangle \leq -\|\theta_k - \theta_{k+1}\|^2 - \frac{\mu}{\nu}\|\theta_k - \theta^*\|^2 .$$

This allows us to write

$$\begin{aligned}
\|\theta_{k+1} - \theta^*\|^2 &= \|\theta_{k+1} - \theta_k\|^2 + \|\theta_k - \theta^*\|^2 + 2\langle \theta_{k+1} - \theta_k, \theta_k - \theta^* \rangle \\
&\leq \|\theta_k - \theta^*\|^2 - \frac{\mu}{\nu}\|\theta_k - \theta^*\|^2 = \left(1 - \frac{\mu}{\nu}\right) \|\theta_k - \theta^*\|^2 .
\end{aligned}$$

The result follows by simple recursion. □

Chapter 4

Solving the inverse problem

Contents

4.1 Introduction

4.1.1 Context and objectives

When a car company builds a new vehicle, it must test the onboard driving aids to ensure that they function properly and comply with all relevant rules and laws. As onboard systems become increasingly numerous and complex, the number of tests required to test them also increases. As on-track testing is costly and time-consuming, companies are building numerical simulators to test their vehicles digitally. However, their reliability must first be shown before simulators can be used for homologation purposes. This is achieved by demonstrating a sufficiently high correlation score between real on-track tests and simulated tests for specific predefined contexts.

A test combines input parameters that define the tested context and corresponding output time series. To maximize the correlation score, it is necessary to identify the input parameters that will simulate the time series most closely resembling an on-track reference time series. This corresponds to the **calibration of the simulator**.

4.1.2 Problem formulation

To calibrate the Renault numerical simulator, we aim to solve the following **inverse problem**: for reference on-track time series denoted y_φ, we want to recover the input parameters θ that simulate the time series $S(\theta)$ closest to the reference ones.

The input parameters θ and the output time series y are linked through the simulator S. We aim to reproduce a real time series with the simulator, which is not perfectly realistic. The simulation has an inherent error when compared to real experiments. To model this error, noise is added to the statistical model

$$y = S(\theta) + \varepsilon\,.$$

Given the reference on-track time series y_φ, we want to find the input parameters θ^* such that $y^* := S(\theta^*)$ is as close as possible to y_φ, that is, we want to solve the optimization problem

$$\theta^* \in \underset{\theta\in\Theta}{\arg\min}\; d\big(S(\theta), y_\varphi\big)\,, \tag{4.1.1}$$

where d is a distance as discussed in Section 1.2 and the parameter space Θ is defined in Section 1.1.2.

We solve this problem using both frequentist and Bayesian approaches. In frequentist methods, we directly solve the optimization problem (4.1.1) and recover a point value estimation

of θ^*. In Bayesian methods, we estimate the posterior distribution given by Bayes' theorem

$$\pi(\theta|y_\varphi) \propto p(y_\varphi|\theta)\pi_0(\theta)\,. \tag{4.1.2}$$

This necessitates the specification of a prior distribution $\pi_0(\theta)$ and the capacity to compute or estimate the likelihood $p(y_\varphi|\theta)$.

In our context, the simulator function S can only be evaluated in given points. Therefore, the objective function (4.1.1) and its gradient, as well as the likelihood (4.1.2) are not accessible. This leads us to use *gradient-free* and *likelihood-free* methods. In the following section, we present various methods that can be employed to solve our problem and discuss the most efficient approach in each context.

4.2 Mathematical description

Recall that we observe time series

$$Y_i = S(\theta) + \varepsilon_i, \quad \forall i = 1, \dots, n \tag{4.2.1}$$

where, for each $i \in \{1, \dots, n\}$, Y_i is a time series of size T. The noise ε_i models the difference between on-track time series and simulated ones. We do not have precise information regarding the nature of this noise, and it can differ from one context to another. Consequently, we propose the following different hypotheses to model it.

4.2.1 Gaussian noise assumption

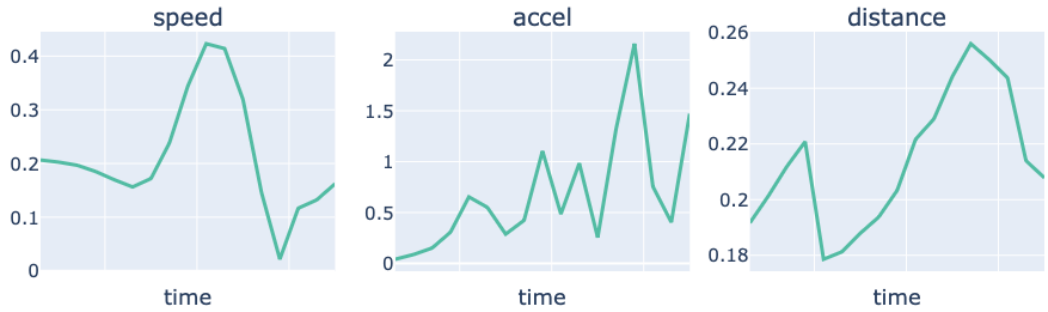


Figure 4.1: Example of simple noise between on-track and simulated time series

In certain scenarios and for specific vehicles, the simulated data are sometimes quite similar to the on-track data, resulting in a relatively straightforward error. For example, Figure 4.1 illustrates the noise between the on-track and simulated time series for a simple scenario.

This type of case allows for the formulation of assumptions regarding the nature of the noise, for example, by assuming it to be Gaussian.

The arguably simplest assumption on the noise is to assume that it follows a Gaussian distribution $\varepsilon_i \sim \mathcal{N}(0, \Sigma)$. In this case, the observation $\overrightarrow{\mathcal{Y}} = (Y_1, \dots, Y_n)$ defined in (4.2.1) is Gaussian and the likelihood of the model is given by

$$\mathcal{L}(\theta, \Sigma, \overrightarrow{\mathcal{Y}}) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{T/2} \det(\Sigma)^{1/2}} \exp\left(\frac{-1}{2} \Big(Y_i - S(\theta)\Big)^\top \Sigma^{-1} \Big(Y_i - S(\theta)\Big) \right) .$$

The maximum likelihood estimators (MLE) in this setting is then defined as

$$\begin{aligned}
(\widehat{\theta}_{\mathsf{MLE}}, \widehat{\Sigma}_{\mathsf{MLE}}) : &= \underset{\theta \in \Theta, \Sigma \succeq 0}{\arg\max} \log \mathcal{L}(\theta, \Sigma, \overrightarrow{\mathcal{Y}}) \\
&= \underset{\theta \in \Theta, \Sigma \succeq 0}{\arg\min} \left\{ n \log \det(\Sigma) + \sum_{i=1}^{n} \Big(Y_i - S(\theta)\Big)^\top \Sigma^{-1} \Big(Y_i - S(\theta)\Big) \right\} \\
&= \underset{\theta \in \Theta, \Sigma \succeq 0}{\arg\min} \left\{ n \log \det(\Sigma) + \sum_{i=1}^{n} \big\| \Sigma^{-1/2} \big(Y_i - S(\theta)\big) \big\|_{2,T}^2 \right\} , \qquad (4.2.2)
\end{aligned}$$

where $\Sigma \succeq 0$ means that Σ is a positive semi-definite matrix and $\|.\|_{2,T}$ stands for the euclidean norm for a vector of size T. This model is already quite flexible and depends, for example, on restrictions that can be made on Σ. Let us discuss some of these classical restrictions.

- Assume first that $\Sigma = \sigma^2 I_T$, with $\sigma^2 > 0$ and I_T is the identity matrix of size T. Then the MLE (4.2.2) is
$$\underset{\theta \in \Theta, \sigma^2 > 0}{\arg\min} \left\{ nT \log(\sigma^2) + \frac{1}{\sigma^2} \sum_{i=1}^{n} \|Y_i - S(\theta)\|_{2,T}^2 \right\} .$$
Optimizing with respect to θ shows that the MLE of θ is the ordinary least-squares estimator
$$\widehat{\theta}_{\mathsf{MLE}} = \underset{\theta \in \Theta}{\arg\min} \left\{ \sum_{i=1}^{n} \|Y_i - S(\theta)\|_{2,T}^2 \right\} = \underset{\theta \in \Theta}{\arg\min} \left\{ \sum_{i=1}^{n} \mathsf{MSE}\big(Y_i, S(\theta)\big) \right\} .$$

- We are interested in data where the observed time series y is a concatenation of R time series such as vehicles' speed and acceleration, ... It can thus be written
$$y = \left(y^{(1)} \mid y^{(2)} \mid \dots \mid y^{(R)} \right) ,$$
where each time series $y^{(r)}$ is of size T_r and satisfies $\sum_{r=1}^{R} T_r = T$.
In order to take into account possible heterogeneity between time series of different

natures, we can modify classical mean squared by multiplying each squared error by different weights, changing

$$\|Y_i - S(\theta)\|_{2,T}^2 = \sum_{r=1}^{R} \|Y_i^{(r)} - S(\theta)^{(r)}\|_{2,T_r}^2$$

into

$$\|Y_i - S(\theta)\|_{2,T,w}^2 = \sum_{r=1}^{R} w_r \|Y_i^{(r)} - S(\theta)^{(r)}\|_{2,T_r}^2 .$$

This is done for example when we consider the scaled mean squared error (s-MSE)

$$\arg\min_{\theta\in\Theta} \left\{ \sum_{i=1}^{n} \text{s-MSE}\big(Y_i, S(\theta)\big) \right\} ,$$

where the weights $w_r = 1/\widehat{\sigma}_r^2$ here are obtained by scaling the data before computing the MSE.

- The last case we'll consider is the most general where Σ is only assumed to be positive semi-definite. We will consider this particular case by using a generic method as Bayesian synthetic likelihood (BSL), by introducing prior knowledge on θ and Σ and computing the corresponding posteriors. For more details, see Section 4.3.

4.2.2 Assumption free

In some situations, the Gaussian assumption of the previous section may not be realistic. For example, as shown in Figure 4.2, the noise can be more sophisticated. We can even decompose it into trends and residuals. See Appendix A for an example and more discussions on this point. In the case of this more complex type of noise, it is necessary to consider alternative methodologies that do not rely on the Gaussian assumption. We present such methods for both frequentist and Bayesian approaches.

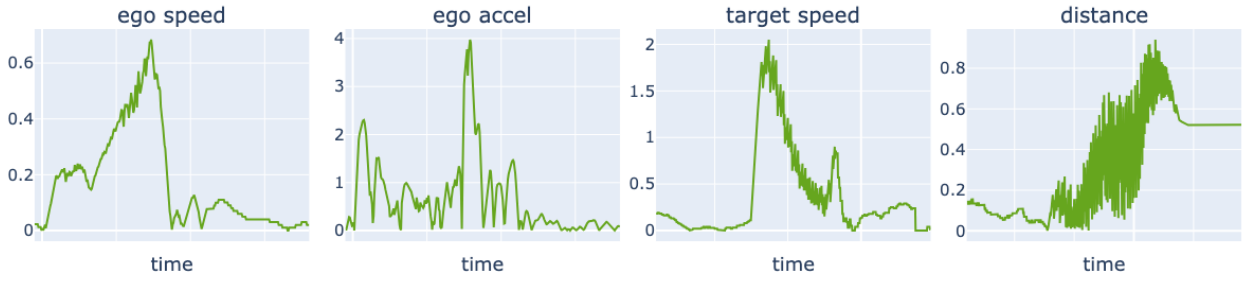


Figure 4.2: Example of sophisticated noise between on-track and simulated time series

Frequentist approach. In general, we optimize the objective function $\theta \mapsto d\big(S(\theta), y_\varphi\big)$, where d is the MSE or the s-MSE. This is equivalent to the MLE in the Gaussian case, as discussed in the previous section. However, as the function S is unknown, we cannot access its gradients. However, as we can evaluate S in $\theta \in \Theta$, we can use *gradient-free optimizers* methods, like the one given in Blanke [2024].

Bayesian approach. Without the Gaussian assumption, we cannot use a BSL method as introduced in Section 4.2.1. There are many alternative Bayesian methods [Tarantola, 2005] for solving this problem in the general context. An example of the needed methodology is described in Giraldi et al. [2017]. The problem is to estimate the posterior distribution $\pi(\theta|y_\varphi)$ while, as previously stated, S is unknown and can only be evaluated at specific values of θ. To estimate the likelihood distribution $p(y_\varphi|\theta)$, we consider *likelihood-free methods*, like sequential Monte Carlo (SMC) samplers, to generate a sample $\theta_1, \ldots, \theta_n$ of the posterior distribution, named *accepted particles*, and then approximate the posterior by

$$\pi(\theta|y_\varphi) \approx \frac{1}{n}\sum_{i=1}^{n} \delta_{\theta_i}(\theta)\,, \tag{4.2.3}$$

where δ denotes the Dirac measure. As the sample size gets larger, the finite sample approximation approximates better. For more details, see Section 4.5.

4.3 Bayesian synthetic likelihood

As mentioned in Section 4.2.1, the generic Gaussian noise assumption naturally leads to the popular BSL method, which allows us to generate posterior samples on the assumption that the likelihood is Gaussian. This method is detailed and compared to approximate Bayesian computation in Price et al. [2018]. The theoretical guarantees of these methods are discussed in Frazier et al. [2022].

Recall that, rather than optimizing the objective function directly, the BSL method aims to determine the posterior distribution defined by

$$\pi(\theta|y_\varphi) \propto p(y_\varphi|\theta)\pi_0(\theta)\,,$$

where $p(y_\varphi|\theta)$ stands for the likelihood function and $\pi_0(\theta)$ is the prior. Here, the unknown likelihood distribution $p(y_\varphi|\theta)$ is supposed to be Gaussian $\mathcal{N}\big(y_\varphi; S(\theta), \Sigma\big)$ and the parameters $S(\theta)$ and Σ need to be estimated.

To proceed, preliminary estimators are used to estimate the parameters $S(\theta)$ and Σ from the

observations $(y_1, \ldots, y_n)$. These estimators are defined by

$$S_n = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad \text{and} \qquad \Sigma_n = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - S_n)(y_i - S_n)^\mathsf{T}\,. \tag{4.3.1}$$

These are then used to estimate $p(y_\varphi|\theta)$ by the following mixture of the Gaussians $\mathcal{N}(y_\varphi; S_n, \Sigma_n)$

$$\pi_{\mathsf{BSL},n}(\theta|y_\varphi) \propto p_n(y_\varphi|\theta)\pi_0(\theta)\,, \tag{4.3.2}$$

where $p_n(y_\varphi|\theta) = \int \mathcal{N}(y_\varphi; S_n, \Sigma_n) \prod_{i=1}^{n} p(y_i|\theta)\,\mathrm{d}y_{1:n}$.

The distribution sampled with BSL given in (4.3.2) is not equal to the *ideal* (but not available) target

$$\pi_{\mathsf{BSL}}(\theta|y_\varphi) \propto \mathcal{N}\big(y_\varphi; S(\theta), \Sigma\big)\pi_0(\theta)\,.$$

However, Drovandi et al. [2015] did show under mild conditions that π_{BSL} is the limit of $\pi_{\mathsf{BSL},n}$ for $n \to \infty$. For more details about these mild conditions, see Section 4.A.1. In Price et al. [2018], they have demonstrated that BSL shows little sensitivity to n, and they conjecture that the estimator bias decreases fairly rapidly so that small to moderate n is sufficient. Therefore, n can be set to maximize the overall computational efficiency.

4.4 Waste-free sequential Monte Carlo sampler

Although Bayesian synthetic likelihood approaches appear to offer a promising solution, our context does not necessitate the utilization of an algorithm of this nature. As a reminder, the model under consideration is given by

$$Y = S(\theta) + \varepsilon\,.$$

Given the simulator's deterministic nature, the incorporation of noise becomes imperative for generating a random component within the model. This is achieved through the synthesis of noise and its subsequent addition to the data. This is done to be able to use probabilistic and Bayesian methods.

Therefore, the value of the parameter ε is known as it is manually synthesized. The initial noise employed is precisely Gaussian. By taking $\varepsilon \sim \mathcal{N}(0, \Sigma)$, now $Y \sim \mathcal{N}\big(S(\theta), \Sigma\big)$ and the problem to be solved can be considered within the framework of a Gaussian model.

As a consequence, for an observation $(y_1, \ldots, y_n)$ such as $y_i \sim \mathcal{N}\big(S(\theta), \Sigma\big)$ for all $i = 1, \ldots, n$,

the log-likelihood of the model, to within one additive constant, is exactly

$$\begin{aligned} l(\theta) &:= \log \mathcal{L}(\theta, \Sigma; y_1, \ldots, y_n) \\ &= -\frac{1}{2} \sum_{i=1}^{n} \big(y_i - S(\theta)\big)^\top \Sigma^{-1} \big(y_i - S(\theta)\big) \,. \end{aligned} \tag{4.4.1}$$

As we can compute exactly the function $\theta \mapsto S(\theta)$ for any given θ, the *waste-free* SMC sampler, proposed by Dau and Chopin [2021], is the chosen solution to estimate the true posterior. This procedure permits to obtain an estimation of the true posterior, without any ABC or BSL approximation, given by

$$\begin{aligned} \pi(\theta | y_\varphi) &\propto \pi_0(\theta) \exp\big(l(\theta)\big) \\ &\propto \pi_0(\theta) \prod_{i=1}^{n} \varphi\Big(y_i; S(\theta), \Sigma\Big) \end{aligned}$$

where $l(\theta)$ is the log-likelihood given by (4.4.1). This approach is reasonable, provided that the function $\theta \mapsto S(\theta)$ can be calculated with absolute precision. In the absence of such a requirement, a BSL approach would be necessary.

What does *waste-free* mean? Traditional SMC samplers often resample a large number of particles, resulting in redundancy and inefficiency. The *waste-free* version aims to alleviate this by resampling only M particles, named *ancestors*, from the total n particles, for $M \ll n$, then propagating each resampled particle $P - 1$ times through the MCMC kernel, forming a new sample of size $n = MP$. In this way, particle redundancy is reduced and sample diversity is increased.

This algorithm is implemented in the `particles` library, developed by Chopin [2024]. The convergence rate of this algorithm and all algorithm details are discussed in the original article of Dau and Chopin. The procedure is given by Algorithm 4.

Let n and M denote two positive real numbers which satisfy $M \ll n$. n denotes the number of particles in the method, and M is the number of ancestors to resample. Let π_0 be a Gaussian prior as defined in Table 4.1 and $l(\theta)$ the log-likelihood defined by (4.4.1). Let $0 = \lambda_{-1} < \ldots < \lambda_K = 1$ denote the tempering exponents. They are real numbers defined following Brent's method, see [Dau and Chopin, 2021, Section 5.1], and are iteratively adapted to ensure that the effective sample size is $n/2$. For any $\lambda > 0$, let $\pi_\lambda(\mathrm{d}\theta) \propto \pi_0(\mathrm{d}\theta) \exp\{\lambda l(\theta)\}$. Let $M^{(k)}$ denote a $\pi_{\lambda_{k-1}}$-invariant Markov kernel. See Section 4.A.2 for Markov kernel and invariant distribution definitions. Here, we take a Gaussian random-walk Metropolis kernel, whose covariance is automatically tuned on the particle samples and is $\pi_{\lambda_{k-1}}$-invariant.

Algorithm 4 Waste-free SMC sampler (tempering version)

Initialization:

- Draw the first particles according to the prior $\theta_1^{(0)}, \dots, \theta_n^{(0)} \sim \pi_0$
- Set the corresponding weights $\omega_i^{(0)} = 1/n \quad \forall i = 1, \dots, n$

Iterations:

1: **for** $k = 0, \dots, K$ **do**

2: Draw M indices with replacement based on particle weights $\omega_1^{(k)}, \dots, \omega_n^{(k)}$

$$A_1^{(k)}, \dots, A_M^{(k)} \subseteq \{1, \dots, n\}$$

3: **for** $a = A_1^{(k)}, \dots, A_M^{(k)}$ **do**

4: Compute the corresponding ancestor $\tilde{\theta}_{m,1}^{(k)} = \theta_a^{(k-1)}$

5: Move the ancestor $P - 1$ times through the $\pi_{\lambda_{k-1}}$-invariant kernel $M^{(k)}$

$$\tilde{\theta}_{m,p}^{(k)} = M^{(k)}\big(\tilde{\theta}_{m,p-1}^{(k)}, \mathrm{d}\theta^{(k)}\big) \quad \forall p = 2, \dots, P$$

6: Gather variables $\big(\tilde{\theta}_{m,p}^{(k)}\big)_{\substack{m=1,\dots,M \\ p=1,\dots,P}}$ so as to form new sample $\theta_1^{(k)}, \dots, \theta_n^{(k)}$

7: Compute the weights and renormalize them

$$\omega_i^{(k)} = \frac{\exp\left\{\big(\lambda^{(k)} - \lambda^{(k-1)}\big) l(\theta_i^{(k)})\right\}}{\sum_{j=1}^n \omega_j^{(k)}} \quad \forall i = 1, \dots, n$$

4.5 Approximate Bayesian computation

As previously stated, the waste-free method heavily relies on the Gaussian assumption. In situations where this assumption is not realistic, this method should not be employed. To proceed in such situations, we use approximate Bayesian computation (ABC) approaches introduced in Sisson et al. [2007], which are more flexible. Indeed, ABC methods are *likelihood-free* schemes because they do not require explicit computation of the likelihood. Instead, they rely on simulating data from the model and comparing this simulated data with observed data. We refer to Sisson et al. [2018] for a more complete and in-depth presentation of those algorithms.

In the following section, we will start with the most commonly used ABC method, the *rejection algorithm*, and proceed to modify it in order to construct an algorithm that is tailored to our specific problem.

4.5.1 Construction of our algorithm

The rejection algorithm described in Algorithm 5 represents the most conventional approach to ABC and is arguably the simplest to implement. It involves successively drawing N parameter samples, referred to as *particles*, according to a predefined prior, then simulating the associated data. Finally, the particles are accepted or rejected based on whether the simulated data are sufficiently close to the observed data. Once all the accepted particles have been obtained, the posterior is estimated using (4.2.3).

Algorithm 5 Rejection algorithm

Initialization:

- Sample independent particles according to the prior $\theta_1^{(0)}, \dots, \theta_n^{(0)} \sim \pi_0$

Iterations:

1: **for** $k = 0, \dots, K$ **do**

2: Generate the time series associated with particles

$$y_i^{(k)} = S\big(\theta_i^{(k)}\big) \quad \forall i = 1, \dots, n$$

3: Determine accepted particles $A_k = \Big\{ i : d\big(y_i^{(k)}, y_\varphi\big) \leq h \Big\} \subseteq \{1, \dots, n\}$

4: **Update step:**

Generate the new particles $\theta_1^{(k+1)}, \dots, \theta_n^{(k+1)} \sim \pi_0$

From this generic rejection sampler, we will develop a new algorithm by adding and combining various components listed below. First, we will assign importance weights to accepted particles to distinguish between different quality levels. Second, rather than drawing new particles according to prior π_0 at each iteration, we make this distribution evolve at each step to make it increasingly specific. The addition of this component transforms our algorithm into a sequential Monte Carlo (SMC) sampler. Finally, the tolerance parameter h that determines whether a particle is accepted or not will also be updated at each step to make particle acceptance increasingly challenging throughout iterations.

1. **Particle weight.** To distinguish accepted particles, we assign them an importance weight depending on their proximity to the target. Each accepted particle θ gets a weight equal to

$$\exp\Big(-\frac{1}{h} d\big(S(\theta), y_\varphi\big) \Big).$$

Hence, particles closer to the target get higher weights as these are more likely to be distance minimizers. The parameter h represents the tolerance threshold that dictates whether a particle is accepted. A smaller value of h results in stricter acceptance criteria, making it more challenging for a particle to be accepted. When h is included in the weight computation, a smaller h leads to a higher weight.

2. **Evolving distributions.** The idea between a SMC sampler is to construct a sequence of slowly changing intermediary distributions $\pi_1, \ldots, \pi_{K-1}$ which bridge between the prior π_0 and the posterior π_K. In this manner, particles drawn according to the prior will traverse the distributions until they reach the final distribution that approximates the posterior. Let

$$\left(\theta_1^{(k)}, \omega_1^{(k)}\right), \ldots, \left(\theta_n^{(k)}, \omega_n^{(k)}\right) \tag{4.5.1}$$

be a weighted sample drawn from the intermediary distribution π_k. To construct the subsequent π_{k+1} distribution, using the k-th sample (4.5.1), we execute a kernel density estimation defined by

$$\pi_{k+1}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \omega_i^{(k)} G_\rho \left(\theta - \theta_i^{(k)}\right)$$

to recover a continuous probability distribution. Here, G_ρ is a Gaussian kernel. We finally employ the *rejection sampling* method [Peng, 2022], a Monte Carlo sample algorithm, to create fresh particles based on this new distribution. It is given by Algorithm 6 with $f = \pi_{k+1}$.

Algorithm 6 Rejection sampling method

Initialization:

- Given the target density f and a sample $\theta_1, \ldots, \theta_n$

Iterations:

1: Simulate $U \sim \text{Unif}([0, 1])$

2: Simulate $\theta \sim \text{Unif}\big(\big[\min(\theta_1, \ldots, \theta_n), \max(\theta_1, \ldots, \theta_n)\big]\big)$

3: **if** $U \leq f(\theta)$ **then**

4: Accept candidate θ

3. **Monotonic decreasing tolerance parameter.** SMC makes probability distributions evolve, and we add an acceptance-rejection component that relies on a tolerance parameter. Then, we can also modify this parameter to make it evolve in time and vary following a monotonically decreasing sequence

$$h_k = k^{-1/\gamma} h_0 \,,$$

where the divider parameter γ and the initial value h_0 are tuning parameters. The tolerance decreases with each iteration k and tends to 0. The divider parameter slows down the convergence rate. In this way, the successive samples improve the approximation of the partial posterior [Sisson et al., 2007].

Our final algorithm is now an ABC SMC sampler. It combines an ABC approach (the particles' acceptance/rejection to override the unknown likelihood constraint) with an SMC sampler (intermediate distributions to transform particles drawn according to prior into particles drawn according to posterior). In more concrete terms, the particles accepted by the ABC approach are used to build the successive distributions of the SMC sampler.

It is described in Figure 4.3. The steps colored in black represent the generic rejection algorithm given by Algorithm 5, while the steps colored in green correspond to the additions described by these three above items.

Lines 1 and 9 correspond to the *monotonic decreasing tolerance parameter* in Item 3. Lines 5 and 8 indicate the incorporation of *particle weight* computation described by Item 1. Line 10 is the step detailed in *evolving distributions* by Item 2.

A stopping rule has been added at line 7 to halt the algorithm's progress when no particle is accepted.

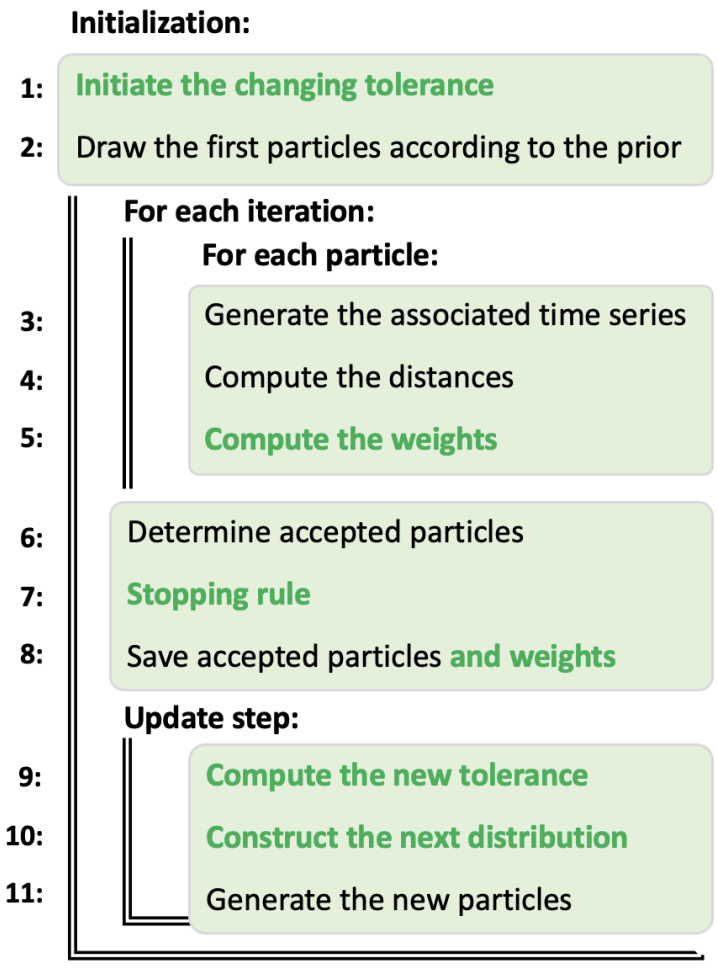


Figure 4.3: Pipeline displaying the order of each stage

4.5.2 Mathematical description

The particles generated with our SMC procedure are sampled from the joint distribution proportional to

$$\begin{aligned} p\big(\theta, y \,|\, d(y, y_\varphi)\big) &= K_h\big(d(y, y_\varphi)\big) I\big(d(y, y_\varphi) \leq h\big) p(\theta, y) \\ &= K_h\big(d(y, y_\varphi)\big) I\big(d(y, y_\varphi) \leq h\big) p(y|\theta) \pi_0(\theta)\,, \end{aligned}$$

where K_h is a standard smoothing kernel function $K_h(u) = \frac{1}{h}\exp(-\frac{u}{h})$, I denotes the indicator function, $I(Z) = 1$ if Z is true and $I(Z) = 0$ otherwise, and $h > 0$ is a scaling parameter.

The ABC approximation of the partial posterior $\pi(\theta|y_\varphi)$ is obtained after integration over y

$$\pi_{\mathsf{ABC}}(\theta|y_\varphi) \propto \int K_h\big(d(y, y_\varphi)\big) p(y|\theta)\pi_0(\theta)\mathrm{d}y\,.$$

As $h \to 0$, $K_h\big(d(y, y_\varphi)\big) \to \delta_{y_\varphi}(y)$ and the approximation gets more accurate

$$\begin{aligned} \lim_{h\to 0} \pi_{\mathsf{ABC}}(\theta|y_\varphi) &= \int \delta_{y_\varphi}(y) p(y|\theta)\pi_0(\theta)\mathrm{d}y \\ &= p(y_\varphi|\theta)\pi_0(\theta) \propto \pi(\theta|y_\varphi)\,. \end{aligned}$$

If h is a small tolerance, $\pi_{\mathsf{ABC}}(\theta|y_\varphi)$ is a good approximation of the true posterior. For a review of SMC methods convergence rate, please refer to Crisan and Doucet [2000].

4.5.3 ABC SMC algorithm

Let π_0 be a Gaussian prior as defined in Table 4.1, d the s-RMSE distance, $h = d\big(S(\theta_0), y_\varphi\big)$ the initial tolerance parameter, $n = 500$ the number of particles to consider, and $\gamma = 6$ the divider parameter. The procedure is given by Algorithm 7.

Parameter	μ	σ
ego vehicle initial speed	50	1.6
target vehicle initial speed	50	1.6
target vehicle braking force	-6	0.8
initial distance between vehicles	40	1.6
front braking efficiency	1	0.4
rear braking efficiency	1	0.4
AEB latency parameter	1	0.08
braking system parameter	1	0.08

Table 4.1: Prior Gaussian $\mathcal{N}(\mu, \sigma^2)$ of input parameters

Algorithm 7 Rejection and decreasing SMC sampler

Initialization:

- Initiate the changing tolerance $h_0 = h$
- Draw the first particles according to the prior $\theta_1^{(0)}, \dots, \theta_n^{(0)} \sim \pi_0$

Iterations:

1: **for** $k = 0, \dots, K$ **do**

2: Generate the associated time series $y_i^{(k)} = S\big(\theta_i^{(k)}\big) \quad \forall i = 1, \dots, n$

3: Determine accepted particles $A_k = \{i : d(y_i^{(k)}, y_\varphi) \leq h_k\} \subseteq \{1, \dots, n\}$

4: **Stopping rule:** Card$(A) = 0$

5: **Update step:**

6: Compute the new tolerance $h_{k+1} = h_0(k+1)^{-1/\gamma}$

7: Compute the weights $\omega_i^{(k)} = \exp\big(-d(y_i^{(k)}, y_\varphi)/h_k\big) \quad \forall i \in A_k$

8: Construct the next distribution with a weighted KDE

$$\pi_{k+1}(\theta) = \frac{1}{|A_k|} \sum_{i \in A_k} \omega_i^{(k)} G_\rho\left(\theta - \theta_i^{(k)}\right)$$

9: Generate the new particles with a *rejection sampling* method

$$\theta_1^{(k+1)}, \dots, \theta_n^{(k+1)} \sim \pi_{k+1}$$

4.6 Constructing and evaluating estimators

In all algorithms, we have to generate a lot of time series to evaluate $S(\theta)$ for many values of θ. Using the actual simulator S turns out to require excessive computational resources in practical situations. Thus, we substitute the simulator with a pre-trained surrogate model, which mimics the simulator and predicts y for a given θ using $\widehat{S}(\theta)$. We use the convolutional neural network model obtained in Chapter 2 for this step.

We now aim to test the approaches described in Section 4.6.1 with the surrogate model $\widehat{S}$ to construct the distinct parameter estimators. We explain in Section 4.6.2 how to assess the quality of all the approaches.

4.6.1 Tested approaches

Classical optimization. As discussed in Section 4.2.2, we use the optimization package of Blanke [2024]. It is used to minimize the objective function $f : \theta \mapsto d\big(\widehat{S}(\theta), y_\varphi\big)$ where d stands for the RMSE and the s-RMSE, $\widehat{S}$ for the surrogate model and y_φ for the reference on-track time series. In our simulations, we used three distinct algorithms to determine the best: Hill Climbing, Powell's Method, and Random Restart Hill Climbing.

Acceptation and rejection. We use the acceptation and rejection algorithm to select parameter values that meet the acceptability condition.

Synthetic likelihood and sequential Monte Carlo. Finally, we test the two solutions described in Section 4.2. Firstly, Algorithm 4, and secondly, Algorithm 7, which combines rejection, SMC sampler and decreasing parameter tolerance.

4.6.2 Quality assessment of results

Frequentist approaches directly output estimators $\widehat{\theta}_{\text{opt}}$ while Bayesian approaches output posterior probability distributions. To convert these into estimators, we compute the weighted mean and the maximum a posteriori (MAP) as follows: let $\theta_1, \dots, \theta_N$ be the final population of accepted particles and $\omega_1, \dots, \omega_N$ their associated weights.

1. **Weighted mean.** The weighted mean is defined as

$$\left(\sum_{n=1}^{N} \omega_n\right)^{-1} \sum_{n=1}^{N} \omega_n \theta_n \,.$$

2. **(Coordinate-wise) Maximum a posteriori.** For each coordinate $j = 1, \dots, p$, we compute the empirical approximation of the posterior as

$$\left(\sum_{n=1}^{N} \omega_n \right)^{-1} \sum_{n=1}^{N} \omega_n \delta_{\theta_{n,j}}(\theta) \,,$$

where $\theta_{n,j}$ stands for the j-th coordinate of the n-th final accepted particles. Then, this empirical distribution is integrated with respect to a smooth kernel to produce a smoother approximation of the posterior. Finally, this approximation is maximized using a standard *Monte Carlo sampling*.

Why coordinate-wise maximization over a smooth approximation of the posterior? Global maximization is a computationally costly process, whereas local maximization is a more cost-effective alternative. Furthermore, local maximization is often a satisfactory solution [Wright, 2015], particularly when the objective function is approximate or when the global optimum is not a significant outcome, which is the case here. Doucet et al. [2009] has shown that for a huge quantity of particles, a smoothing step is necessary. Indeed, smoothing approximation posterior enables the capture of global trends, as opposed to focusing on local fluctuations. It reduces the impact of noise potentially captured by particles by building a more reliable posterior approximation, thereby facilitating a good MAP estimation.

To test all methods, the final estimators $\widehat{\theta}$ is used in both the surrogate model $\widehat{S}$ and the simulator S and the outputs $\widehat{S}(\widehat{\theta})$ and $S(\widehat{\theta})$ are compared with the actual time series y_φ. We show in the results the distances $d(\widehat{S}(\widehat{\theta}), y_\varphi)$ and $d(S(\widehat{\theta}), y_\varphi)$ for different methods and the distance d being the s-RMSE.

We also show the s-RMSE gap

$$\left| \text{s-RMSE}\Big(S(\widehat{\theta}), y_\varphi\Big) - \text{s-RMSE}\Big(\widehat{S}(\widehat{\theta}), y_\varphi\Big) \right| , \tag{4.6.1}$$

and the s-RMSE on the surrogate model

$$\text{s-RMSE}\Big(S(\widehat{\theta}), \widehat{S}(\widehat{\theta})\Big) \,. \tag{4.6.2}$$

These allow to assess the impact of using the surrogate model in the inverse problem, even though the simulator is the optimization target.

4.6.3 Numerical results and comparisons

		$\widehat{\theta}^1_{\text{opt}}$	$\widehat{\theta}^2_{\text{opt}}$	$\widehat{\theta}_{\text{AR}}$	$\widehat{\theta}^1_{\text{RD-SMC}}$	$\widehat{\theta}^2_{\text{RD-SMC}}$	$\widehat{\theta}^1_{\text{BSL}}$	$\widehat{\theta}^2_{\text{BSL}}$	$\widehat{\theta}^3_{\text{BSL}}$
option		RMSE	s-RMSE	mean	mean	MAP	mean	MAP	mean + early stopping
$\widehat{\text{S}}$	**s-RMSE**	0.1763	0.1580	0.1742	0.1650	0.1613	**0.1486**	**0.1486**	0.1609
S	**s-RMSE**	0.1781	0.1664	0.1787	0.1673	**0.1628**	0.1665	0.1672	0.1675
s-RMSE gap		0.0018	0.0084	0.0045	0.0023	**0.0015**	0.0179	0.0186	0.0066
model error		0.0201	0.0347	0.0297	**0.0197**	0.0240	0.0917	0.0946	0.0786
time		12 min	12 min	9 min	2 min	2.5 min	25 min	26 min	11 min

Table 4.2: Final numerical results

The **bold values** represent the smallest values in each line.

For $\widehat{\theta}^3_{\text{BSL}}$ description, please refer to Section 4.6.4 [Early stopping in BSL].

Optimizing RMSE vs. s-RMSE. Whether using the surrogate model $\widehat{S}$ or the simulator S, the errors obtained by optimizing the s-RMSE are smaller, ranging from 0.1763 and 0.1781 to 0.1580 and 0.1664. This is not surprising, it simply means that is better to minimize the s-RMSE to compute estimators when we want to minimize the distance to y_φ w.r.t. to the s-RMSE.

Optimization vs. Bayesian approaches. Start with the error on the surrogate model $\widehat{S}$. The estimator $\widehat{\theta}^2_{\text{opt}}$ has an s-RMSE of 0.1580, which is smaller than the errors of the two RD-SMC estimators. For $\widehat{S}$, optimization is, in this sense, the optimal solution. This is no longer true for the errors on the simulator S. Here, the optimization estimator has an s-RMSE of 0.1664, larger than the error of 0.1628 obtained from the 2$^{\text{nd}}$ RD-SMC estimator.
The analysis of the s-RMSE gap shows that Bayesian methods offer lower deviation than optimization ones. Indeed, the s-RMSE gaps with RD-SMC estimators are lower than 0.0023 while it is equal to 0.0084 for $\widehat{\theta}^2_{\text{opt}}$. This shows that the performance of $\widehat{\theta}^2_{\text{opt}}$ does not generate as well as those of RD-SMC. As the performances of RD-SMC are slightly better on S, with performance closer to those on the simulator, these methods are ultimately more interesting in our application.

Acceptation/rejection vs. sequential Monte Carlo. The SMC method outperforms the basic acceptance/rejection approach in all aspects. SMC leads to smaller errors and a smaller s-RMSE gap. Additionally, the computation times are four times shorter. Even though this finding is not surprising, it highlights the effectiveness and benefits of SMC samplers.

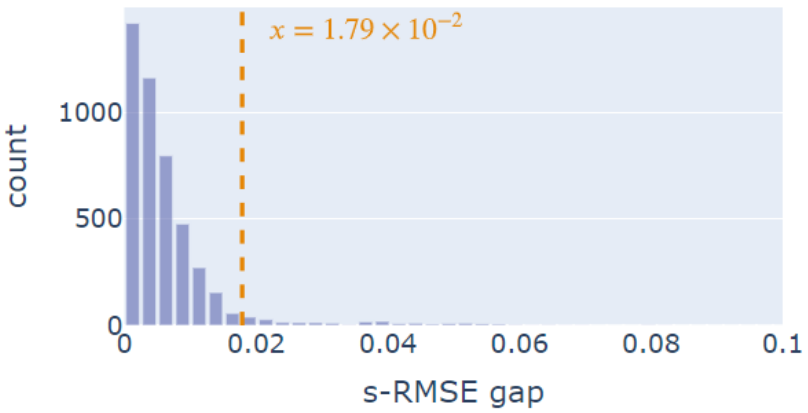
Mean vs. maximum a posteriori. We used two approaches to calculate the final parameter vector: the mean and the maximum a posteriori. For both $\widehat{\theta}^1_{\text{RD-SMC}}$ and $\widehat{\theta}^2_{\text{RD-SMC}}$, we found the MAP advantageous as it reduced both the s-RMSE and its gap, from 0.1650 to 0.1613 with the model, from 0.1673 to 0.1628 with the simulator and from 0.0023 to 0.0015 for the gap. Concerning the two BSL estimators, $\widehat{\theta}^1_{\text{BSL}}$ and $\widehat{\theta}^2_{\text{BSL}}$, computing the MAP does not alter the s-RMSE of 0.1486 obtained with the model. The s-RMSE with the simulator is nearly identical from 0.1665 to 0.1672. We can conclude that the MAP estimation may be beneficial. Nevertheless, the estimation of MAP is a computationally demanding process that is susceptible to error. The weighted average remains a valuable tool for comparing the two estimators and retaining the most appropriate one for each case.

Synthetic likelihood vs. sequential Monte Carlo. The synthetic likelihood estimator $\widehat{\theta}^1_{\text{BSL}}$ outperforms all methods for the inverse problem on the surrogate model with a s-RMSE of 0.1486. However, when the parameters are used in the simulator to evaluate their practical performance, they achieved a s-RMSE on the simulator of 0.1665. These poor generalization properties are reflected in the s-RMSE gap of 0.0179. This method seems very encouraging, even if it takes slightly longer than the alternatives. This suggests improving the surrogate model to obtain better results on the simulator; see the next section for more details.

4.6.4 Why synthetic likelihood is failing ?

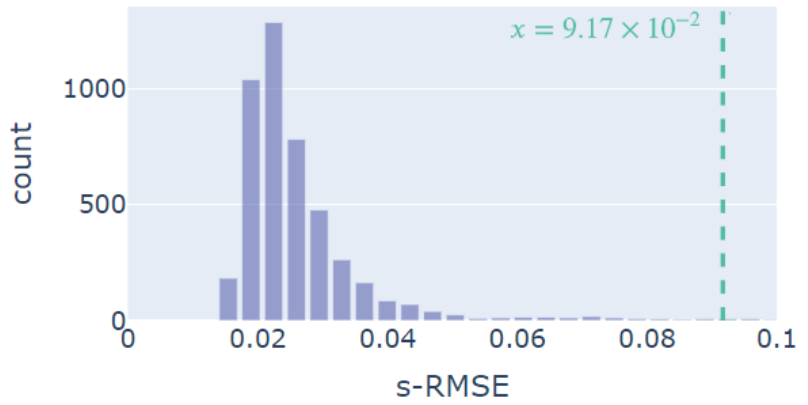
We compute the s-RMSE gap (4.6.1) and the s-RMSE on the surrogate model (4.6.2) using the training data that was simulated upstream to build the surrogate model $\widehat{S}$. The corresponding histogram and statistical description are presented in Figure 4.4 and Figure 4.5. The s-RMSE gap and the s-RMSE on the surrogate model for $\widehat{\theta}^1_{\text{BSL}}$ are respectively

$$\left|\text{s-RMSE}\big(S(\widehat{\theta}^1_{\text{BSL}}), y_\varphi\big) - \text{s-RMSE}\big(\widehat{S}(\widehat{\theta}^1_{\text{BSL}}), y_\varphi\big)\right| = 1.79 \times 10^{-2}\,,$$
$$\text{and} \qquad \text{s-RMSE}\big(S(\widehat{\theta}^1_{\text{BSL}}), \widehat{S}(\widehat{\theta}^1_{\text{BSL}})\big) = 9.17 \times 10^{-2}\,.$$



	$\times 10^{-2}$
mean	0.6368
std.	0.8154
min	0.0001
max	11.3767
50 %	0.4341
95 %	**1.7412**
99 %	4.7443

Figure 4.4: Histogram and statistic description of the s-RMSE gap obtained with the training data of the surrogate model



	$\times 10^{-2}$
mean	2.6693
std.	1.1981
min	1.3727
max	18.8641
50 %	2.3486
95 %	4.4444
99 %	**8.3648**

Figure 4.5: Histogram and statistic description of the s-RMSE obtained with the surrogate model on the training data

The s-RMSE gap of $\widehat{\theta}^1_{\text{BSL}}$ is among the top 5% with the highest gap. Additionally, the quality of the surrogate model with the parameters estimated by the BSL estimator is poor, with the prediction falling in the worst 1%. It shows that the BSL method has identified parameters for which the model performs poorly, resulting in a substantial s-RMSE gap compared to other parameter values. We conclude that the model performance is responsible for this disappointing result of the synthetic likelihood approach.

To robustify the estimator $\widehat{\theta}^1_{\text{BSL}}$, an early stopping step is added to Algorithm 4 to prevent over-optimization of the surrogate model.

Early stopping in BSL. The addition of early stopping results in degraded errors for the second BSL estimator $\widehat{\theta}^2_{\text{BSL}}$ compared to the first one. However, it successfully achieves the objective of reducing the s-RMSE gap: it was previously 0.0179, but it has now decreased to 0.0066, placing it within the average gap. By adding early stopping, the BSL estimator is similar to the first RD-SMC estimator but takes more time.

4.6.5 The two best estimators

So far, two estimators stand out from the crowd $\widehat{\theta}^2_{\text{RD-SMC}}$ and $\widehat{\theta}^1_{\text{BSL}}$. Each option appears to possess distinct advantages and drawbacks.

We compute the s-RMSE obtained by the two estimators for the four distinct time series. As described in Table 4.3, they perform similarly in ego speed and acceleration time series. However, there is a disparity in the target speed and distance time series. It demonstrates the excellent performance of the RD-SMC estimator on the simulator.

	method	ego speed	ego accel	target speed	distance
s-RMSE ($\times 10^{-2}$)	$S(\widehat{\theta}^2_{\text{RD-SMC}})$	2.54	15.66	**3.11**	**1.91**
	$S(\widehat{\theta}^1_{\text{BSL}})$	2.65	15.64	4.43	2.41

Table 4.3: s-RMSE obtained with the three best approaches for the four distinct time series

Furthermore, we consider the four errors from the two estimators described in Table 4.2 and noted $(\text{err}_{i,j})_{j=1,\dots,4}$ for $i = 1, 2$. We conduct a weighted average $\sum_j \alpha_j \text{err}_{i,j}$ that prioritizes the s-RMSE and simulator errors, and we determine a ranking of these estimators. The BSL estimator consistently ranks first due to its exceptional performance with the surrogate model. However, it runs for an extended period and possesses a significant s-RMSE gap. The RD-SMC estimator takes second place as the simulator performs optimally for s-RMSE but inadequately for other errors.

Figure 4.6 shows the simulated time series based on the parameters derived from the two estimators. We only display target speed and distance, as ego speed and ego acceleration do not show any differences. For target speed, the BSL estimator performs better for the first time steps (upper left plot), while the RD-SMC estimator performs better for the last time steps (lower left plot). For distance, the RD-SMC estimator seems to be better throughout the time series (right plots). Achieving a suitable fit for the target speed simultaneously with the distance poses some difficulties.

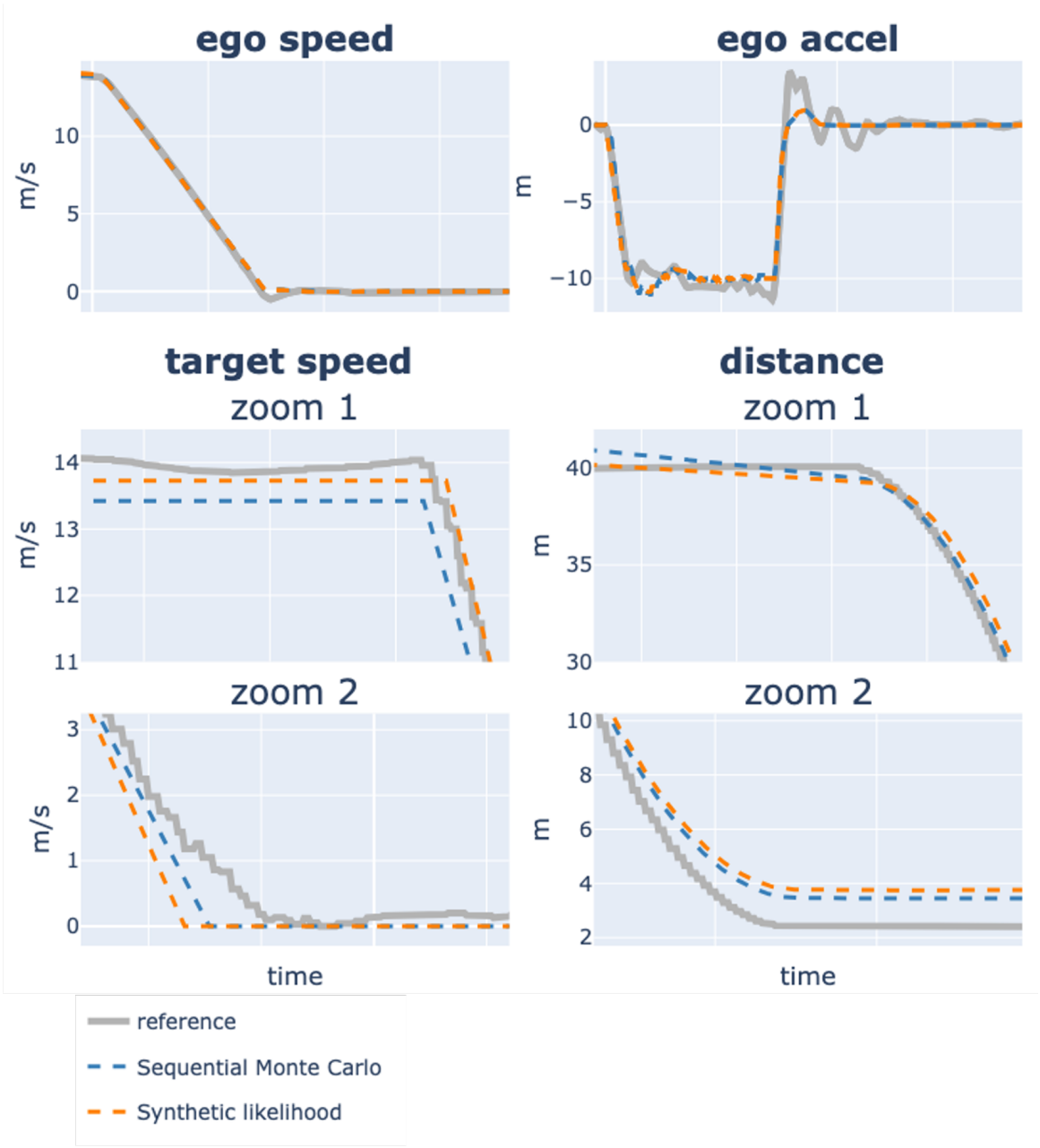


Figure 4.6: Time series simulated with the parameter values obtained using the three best estimators

Although the Renault group aims to determine a single vector of parameters that enables the simulation of the most realistic time series possible, we could construct two separate estimators. Either we work with the intrinsic differences between the simulation and the on-track reality to determine the best parameters with the existing resources, or we remove these constraints and determine two different sets of parameters, making it possible to infer target speed and distance.

4.7 Conclusion

Currently, $\widehat{\theta}^2_{\text{RD-SMC}}$ is the most reliable estimator as it performs best on S, which is our final objective. Additionally, the s-RMSE gap is small, so the performance on the surrogate model gives a good idea of what we will get on S.

However, $\widehat{\theta}^1_{\text{BSL}}$ shows promise as it performs better on the surrogate model. Therefore, it has the potential to perform better on S when the model is improved. But for now, considering that the calculation time for SMC is shorter than for BSL, we are inclined to choose SMC.

Finally, although we develop the methodology with specific data in mind, the overall approach can be extended to other contexts.

Benefits of a Bayesian approach

In this paper, we demonstrate the effectiveness of Bayesian approaches. They make the integration of prior information possible across the prior. For instance, it allows for the definition of probabilistic regions of interest around the nominal values, as described in Section 1.1.2 [• Parameter spaces].

Massive simulation

To assess the performance of optimization, sequential Monte Carlo, and Bayesian synthetic likelihood, we utilize the Renault simulator to simulate $N = 100{,}000$ scenarios randomly denoted $y = (y_1, \dots, y_N)$. We randomly select n scenarios denoted by $\mathcal{S}_n = \{y_i,\ \forall i \in I \subset [\![1, N]\!] \text{ and } |I| = n\}$ for $1 \leq n \leq N$ and we compute

$$\min_{y \in \mathcal{S}_n} \text{s-RMSE}(y, y_\varphi)\,.$$

Repeating these two steps several times, we recover several minimal s-RMSE values, and we compute the 95 % quantile. The obtained result is given in Figure 4.7. The green example point indicates that for 27,000 considered scenarios, we obtain a minimum score of 0.1628 with a probability of 0.95. The points for the three methods are obtained by calculating

$$\text{s-RMSE}\big(S(\widehat{\theta}^1_{\text{opt}})\,, y_\varphi\big), \quad \text{s-RMSE}\big(S(\widehat{\theta}^2_{\text{RD-SMC}}), y_\varphi\big)\,, \quad \text{and} \quad \text{s-RMSE}\big(S(\widehat{\theta}^1_{\text{BSL}}), y_\varphi\big)\,.$$

The associated x-coordinate is given by the number of scenarios used to build model $\widehat{S}$ and the number of scenarios generated in the inverse problem-solving loop, i.e., around 5,500 scenarios.

Firstly, it is evident that the optimization and BSL algorithms perform poorly, achieving no better than a random draw. On the other hand, the SMC algorithm demonstrates superior performance, obtaining a score of 0.1628 with 5,500 scenarios utilized instead of the 27,000 scenarios required to achieve the same score with a random draw. Furthermore, the lowest score obtainable through random draws is 0.1620, indicating that the score achieved through the SMC is highly comparable to the optimal result. The outcome of this study provides clear evidence of the value and efficacy of the approach taken by SMC to automate and enhance the calibration process of Renault's digital simulator.

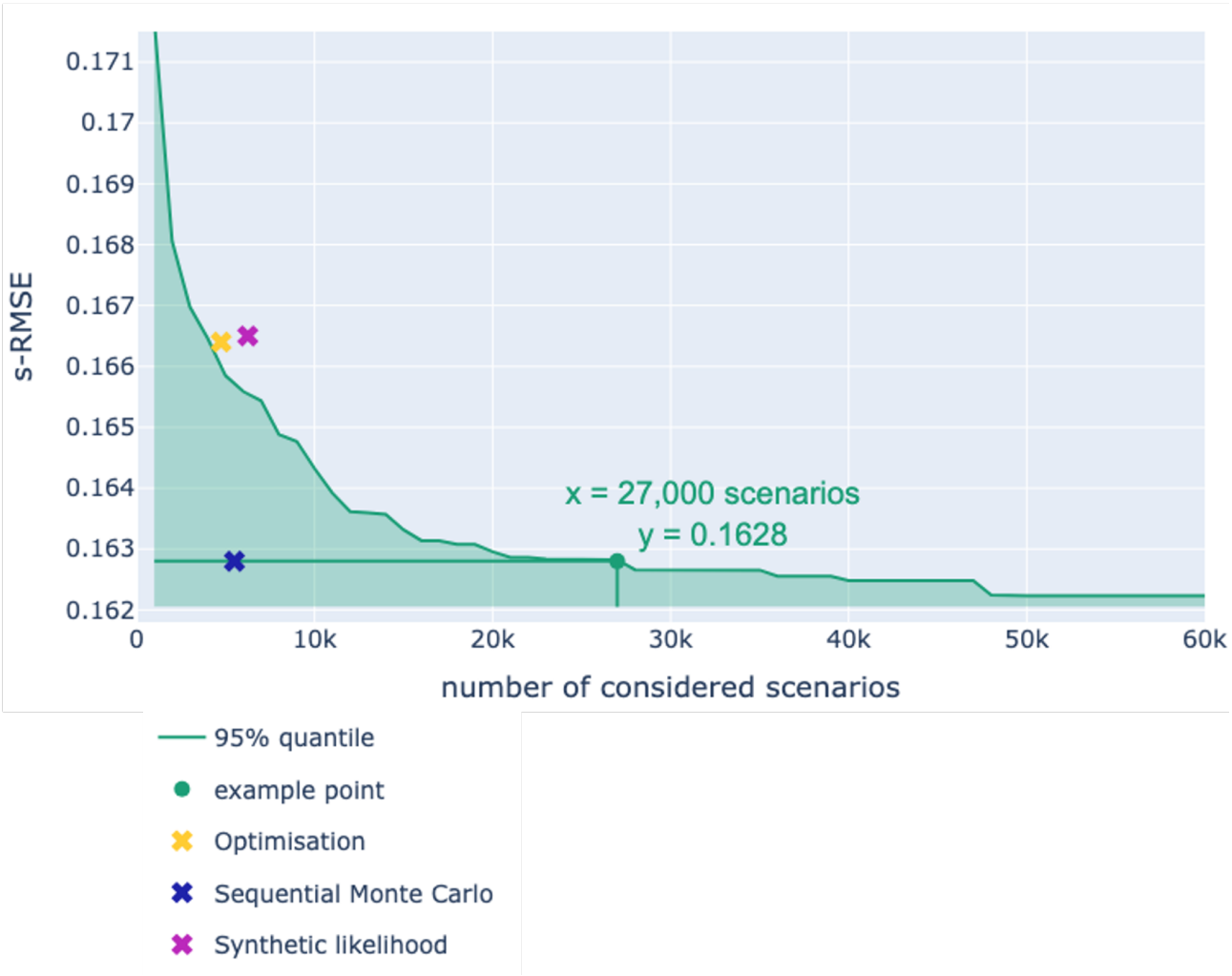


Figure 4.7: Quantile 95% of the minimum s-RMSE score obtained as a function of the number of considered scenarios

4.A Appendix of Chapter 4

4.A.1 Mild conditions for the BSL posterior convergence

Let $p_n(y|\theta)$ be the BSL artificial likelihood and $p(y|\theta)$ be the auxiliary likelihood. In our context, it is derived from the following normal distribution $\mathcal{N}\big(y; S(\theta), \Sigma\big)$.

Result 4.1. *Assume that* $p_n(y|\theta) \underset{n\to\infty}{\to} p(y|\theta)$ *for all* θ *with a positive prior support,* $\inf_n \int_\Theta p_n(y|\theta)p(\theta)\mathrm{d}\theta > 0$ *and* $\sup_{\theta\in\Theta} p(y|\theta) < \infty$*. Then,*

$$\lim_{n\to\infty} \pi_{\mathsf{BSL},n}(\theta|y) = \pi_{\mathsf{BSL}}(\theta|y)\,.$$

The S_n and Σ_n estimators, defined by (4.3.1), of $S(\theta)$ and Σ are unbiased, ensuring that the artificial BSL likelihood converges to the auxiliary likelihood. Furthermore, the two additional conditions regarding the lower and upper bounds are also met, given that we are considering normal distributions. This ensures that the artificial BSL posterior $\pi_{\mathsf{BSL},n}(\theta|y_\varphi)$ will converge to the ideal target posterior $\pi_{\mathsf{BSL}}(\theta|y_\varphi)$ for $n \to \infty$.

4.A.2 Definitions

Definition 4.1 (Markov kernel)**.** *Let* $(X, \mathcal{A})$ *and* $(Y, \mathcal{B})$ *be measurable spaces. A Markov kernel with source* $(X, \mathcal{A})$ *and target* $(Y, \mathcal{B})$ *is a map* $\kappa : \mathcal{B} \times X \to [0, 1]$ *with the following properties*

(i) For every (fixed) $B_0 \in \mathcal{B}$*, the map* $x \mapsto \kappa(B_0, x)$ *is* $\mathcal{A}$*-measurable.*

(ii) For every (fixed) $x_0 \in X$*, the map* $B \mapsto \kappa(B, x_0)$ *is a probability measure on* $(Y, \mathcal{B})$*.*

Definition 4.2 (Invariant distribution)**.** *A distribution* π *is said to be invariant or stationary for a Markov kernel,* M*, if* $\pi M = \pi$*.*

Chapter 5

Introduction générale en français

Contents

5.1 Contexte et objectifs de la thèse

Le véhicule autonome (AD) et les systèmes avancés d'aide à la conduite (ADAS) occupent une place de plus en plus importante à l'échelle mondiale. De nombreuses entreprises automobiles investissent massivement et réalisent des avancées significatives. Les ADAS améliorent la sécurité routière en aidant les conducteurs à l'aide de diverses technologies. Ces systèmes deviennent de plus en plus complexes et fournissent une assistance à de multiples niveaux, comme le soulignent des études récentes [Ziebinski et al., 2017]. Les ADAS sont divisés en deux catégories principales : les systèmes passifs et les systèmes actifs. Les ADAS passifs fournissent aux conducteurs des informations qui leur permettent de prendre des décisions plus sûres et mieux informées. Par exemple, les systèmes d'alerte de collision avant (FCW) et d'aide au changement de voie (LCA) alertent le conducteur des dangers potentiels et lui suggèrent des actions, mais ces systèmes ne contrôlent pas le véhicule. À l'inverse, les ADAS actifs jouent un rôle plus direct dans le fonctionnement du véhicule en prenant de manière autonome des décisions visant à prévenir les accidents ou à améliorer le confort de conduite. Il s'agit par exemple du freinage d'urgence automatique (AEB), qui déclenche de manière autonome le freinage pour éviter les collisions, ou du régulateur de vitesse adaptatif, qui ajuste automatiquement la vitesse du véhicule tout en

maintenant une distance de sécurité avec les autres voitures. Les ADAS actifs améliorent non seulement la sécurité en intervenant dans des situations potentiellement dangereuses, mais aussi le confort de conduite en réduisant la charge de travail du conducteur. Les ADAS passifs et actifs sont essentiels pour développer des technologies automobiles plus sûres et représentent une étape essentielle vers les véhicules entièrement autonomes, qui visent à réduire considérablement les accidents de la route et à améliorer l'expérience de conduite.

Un récent sondage d'opinion aux États-Unis [Diez et al., 2021] indique que l'utilisation des ADAS a augmenté rapidement. Selon les données de l'enquête datant de 2020, 96 % des nouveaux véhicules sont équipés d'au moins une fonction ADAS. De plus, 58 % des personnes interrogées ont indiqué qu'elles étaient intéressées par des ADAS lors de l'achat de leur prochain véhicule, ce qui témoigne d'un intérêt et d'une acceptation croissants de la part des consommateurs. Malgré cet enthousiasme, les utilisateurs sont très sceptiques, 86 % d'entre eux exprimant des doutes quant à leur fiabilité. De même, 80 % des participants ont exprimé le souhait que ces systèmes fonctionnent plus efficacement. Ces doutes et cette méfiance souligne l'importance d'une réglementation stricte et d'une réelle fiabilité de ces technologies, garantissant ainsi leur bon fonctionnement et leur sûreté et permettant le renforcement de la confiance des consommateurs dans ces technologies. L'augmentation de la complexité de ces systèmes et l'intégration d'un nombre croissant de capteurs, comme l'indique Kim et al. [2018], nécessitent des tests développés et complets. À mesure que les ADAS évoluent, les cas d'usage potentiels se multiplient, ce qui complique les réglementations nationales et internationales. Ce paysage réglementaire en expansion nécessite des tests plus rigoureux et plus étendus pour la certification des véhicules, comme le souligne Kalra and Paddock [2016].

Les processus de certification des ADAS et des véhicules autonomes sont notoirement compliqués, longs, incomplets et onéreux en raison de la nécessité de réaliser les essais dans le monde réel, sur piste. L'un des défis majeurs des essais sur piste consiste à reproduire des conditions environnementales spécifiques. Par exemple, certains scénarios météorologiques sont difficiles à simuler avec précision de par, notamment, l'absence de canons à neige ou de souffleries sur les pistes d'essai. De plus, le contrôle de la température extérieure est impossible. En outre, certains scénarios spécifiques présentent des risques trop élevés pour la sécurité de tous et ne sont donc pas réalisables dans le monde réel. Comme par exemple, l'évaluation du comportement d'un véhicule lancé à grande vitesse dans un scénario de sortie de route. Toutefois, comme le souligne Lakomicki [2018], l'industrie automobile est en train de développer des méthodes d'essai plus optimales, en s'éloignant de plus en plus des essais traditionnels sur piste. Les constructeurs comme Renault se tournent vers les simulateurs numériques et physiques en tant qu'outils complémentaires pour affiner le processus d'essai. Ces simulateurs offrent un environnement contrôlé et sûr dans lequel divers scénarios de conduite et conditions

environnementales peuvent être reproduits et testés avec précision. Cette approche permet non seulement de réduire les coûts et la durée des essais, mais aussi d'améliorer la rigueur et la sécurité du processus de certification. En intégrant ces technologies innovantes, l'industrie automobile vise à améliorer la fiabilité et l'efficacité des essais de véhicules, ce qui aboutira à terme à des technologies ADAS plus sûres et plus efficaces.

Renault simulator. Le simulateur numérique de Renault repose sur la suite logicielle SCANeR^{TM} studio, développée par AVSimulation [2023] et qui est spécialement conçue pour les simulations automobiles. Sa fonction première est de soutenir le développement et les essais du véhicule autonome et des ADAS. Ce simulateur fournit un ensemble d'outils essentiels qui facilitent la création d'un environnement virtuel réaliste. Il permet aux utilisateurs de définir des environnements routiers complexes, des dynamiques de véhicules, des schémas de circulation et des conditions météorologiques variables, offrant ainsi un cadre complet pour des essais rigoureux. Pour lancer une simulation, nous devons définir divers **paramètres d'entrée** afin de mettre en place le scénario souhaité. Ces paramètres peuvent inclure, par exemple, la vitesse initiale du véhicule, l'efficacité du freinage et d'autres dynamiques propres au véhicule. Une fois ces paramètres définis, le simulateur génère des **séries temporelles** qui décrivent le comportement dans le temps des véhicules impliqués, comme l'évolution de la vitesse et de l'accélération. Un **scénario** du simulateur est défini par une combinaison de paramètres d'entrée et des séries temporelles qui en résultent. Ces données permettent une analyse précise de la manière dont les différents paramètres influent sur les performances du véhicule dans les conditions souhaitées. Pour plus de détails sur les données simulées, reportez-vous à la Section 5.1.1, qui donne un aperçu plus approfondi de la structure et de l'utilité des données.

Avant que les simulations puissent être officiellement intégrées dans le processus d'homologation des véhicules, il est impératif de vérifier que le simulateur fonctionne correctement, comme le démontrent les travaux de Nabhan [2020]. Il est essentiel d'établir que les résultats des simulations sont conformes avec les données obtenues lors des essais réels sur piste. Bien que les méthodes existantes permettent de réaliser cette validation, elles manquent souvent d'automatisation et nécessitent un temps de mise en œuvre important, ce qui les rend moins efficaces pour une utilisation fréquente ou à grande échelle. Pour résoudre ce problème, nous proposons une méthodologie automatisée pour **calibrer le simulateur**. Cette approche innovante vise à garantir que les données générées par le simulateur sont suffisamment corrélées avec les données réelles sur la piste. Notre objectif est d'établir une procédure qui permette l'évaluation de la qualité de la simulation par une comparaison directe avec les données sur piste, suivie d'ajustements nécessaires pour améliorer la précision. Cette méthodologie permet de simplifier le processus consistant à aligner les séries temporelles simulées sur les séries réelles et améliore ainsi de manière

significative la fiabilité et l'utilité du simulateur. L'automatisation du processus de calibration réduit le temps et la main-d'œuvre nécessaires, augmentant ainsi l'efficacité et permettant des étalonnages plus fréquents si nécessaire. En fin de compte, cela permet d'obtenir des données plus précises et plus fiables, ce qui est crucial pour le développement et la certification des véhicules autonomes et des ADAS.

Pour améliorer la compréhension du sujet, la section suivante présente les données utilisées. Ces données serviront exclusivement pour calibrer notre méthodologie et faciliter la prise de décisions en toute connaissance de cause. Par ailleurs, nous avons pour objectif de développer une méthode de calibration la plus automatique possible, qui lui permettra ainsi de dépasser le cadre actuel et ainsi être généralisée à divers types de données. Cette possibilité d'application plus large souligne la polyvalence et la pertinence de notre approche, suggérant que nos résultats pourraient être utiles pour des défis similaires dans des domaines connexes.

5.1.1 Description des données

Dans cette thèse, nous nous intéressons à un scénario de freinage automatique d'urgence (AEB) impliquant deux véhicules : le véhicule **target**, qui est en tête, et le véhicule **ego**, qui suit. Au départ, les deux véhicules roulent à des vitesses constantes prédéterminées. Le scénario se déroule lorsque le véhicule target freine brusquement, ce qui conduit le véhicule ego à activer son système de freinage d'urgence. Cette réaction est cruciale pour éviter une collision et la dynamique de cette interaction est évaluée pour déterminer l'efficacité du système AEB.

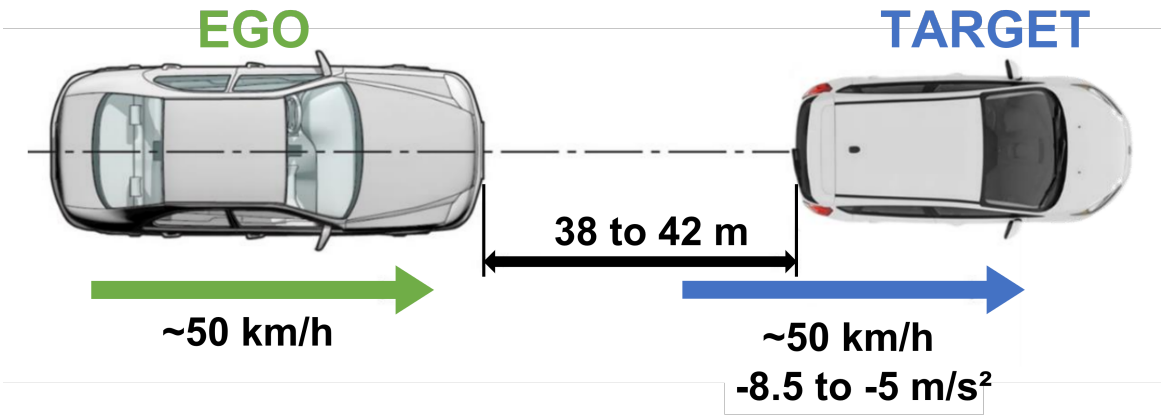


Figure 5.1: Scénario de freinage automatique d'urgence impliquant les véhicules ego et target

Un scénario est défini comme une combinaison de deux composantes principales.

- **Paramètres d'entrée.** Ces variables définissent les conditions nécessaires à la réalisation des simulations. Elles comprennent des paramètres qui fixent les conditions initiales des véhicules, telles que leur vitesse et la force de freinage. D'autres paramètres concernent des paramètres internes telles que le temps de réaction du système AEB et son efficacité, qui influencent directement la réponse du véhicule. Des facteurs environnementaux tels que le coefficient d'adhérence au sol sont également spécifiés, car ils ont un impact significatif sur la dynamique du véhicule dans diverses conditions routières.
- **Séries temporelles en sortie.** Ces données capturent le comportement dynamique des véhicules tout au long de la simulation. Elles comprennent des mesures détaillées telles que la vitesse et l'accélération du véhicule ego, la vitesse du véhicule target et l'évolution de la distance qui les sépare. Ces données sont essentielles pour analyser l'efficacité du système AEB et d'autres dispositifs de sécurité dans des conditions d'urgence simulées, car elles permettent de comprendre comment les véhicules interagissent et réagissent aux données d'entrée définies.

Deux types principaux de scénarios sont utilisés pour tester et développer les systèmes de sécurité des véhicules : les scénarios simulés et les scénarios réels faits sur piste. Chaque type de scénario a un objectif distinct.

- **Scénarios simulés.** Ils sont générés grâce au simulateur numérique, ce qui offre une grande souplesse en matière de conception expérimentale. Les utilisateurs peuvent définir divers paramètres d'entrée pour adapter les scénarios à des besoins d'essai spécifiques. Cela permet également de reproduire des expériences dans des mêmes conditions, ce qui est parfois nécessaire pour les essais et la validation. En outre, les scénarios simulés sont rentables, ils éliminent de nombreuses contraintes logistiques et financières associées aux environnements d'essai physiques.
- **Scénarios réels faits sur piste.** Ces scénarios sont des essais physiques réalisés sur des pistes d'essai spécialisées. En raison des coûts élevés associés à la mise en place et à l'exécution de ces tests, ils sont peu nombreux. Les tests sur piste réelle suivent généralement un contexte bien défini avec des paramètres fixés, appelés paramètres nominaux, afin de garantir que les scénarios reproduisent fidèlement les événements potentiels du monde réel. Les données collectées lors de ces essais sont des séries temporelles variées. Elles sont essentielles pour valider la précision et la fiabilité des simulations.

Paramètres nominaux. Les paramètres nominaux définissent le scénario de référence que l'on souhaite tester et sont utilisés comme valeurs initiales lors des essais sur piste. L'objectif principal est d'approcher ces séries temporelles de référence faites sur piste par le biais de la simulation. Le défi consiste à identifier et à ajuster les paramètres d'entrée du simulateur afin d'obtenir des séries temporelles simulées qui ressemblent le plus possible aux données réelles de référence. Ce processus réalise un affinage des paramètres d'entrée, en partant des valeurs nominales et en les réajustant pour obtenir les simulations les plus réalistes possible.

Type	Paramètre	Valeur nominale	Intervalle
Scénario	ego initial speed	50	[48, 52]
	target initial speed	50	[48, 52]
	target braking force	-6	[-7, -5]
	initial distance between the two	40	[38, 42]
MADA (ego)	front braking efficiency	1	[0.5, 1.5]
	rear braking efficiency	1	[0.5, 1.5]
	AEB latency parameter	1	[0.9, 1.1]
	braking system parameter	1	[0.9, 1.1]

Table 5.1: Définition des paramètres d'entrée

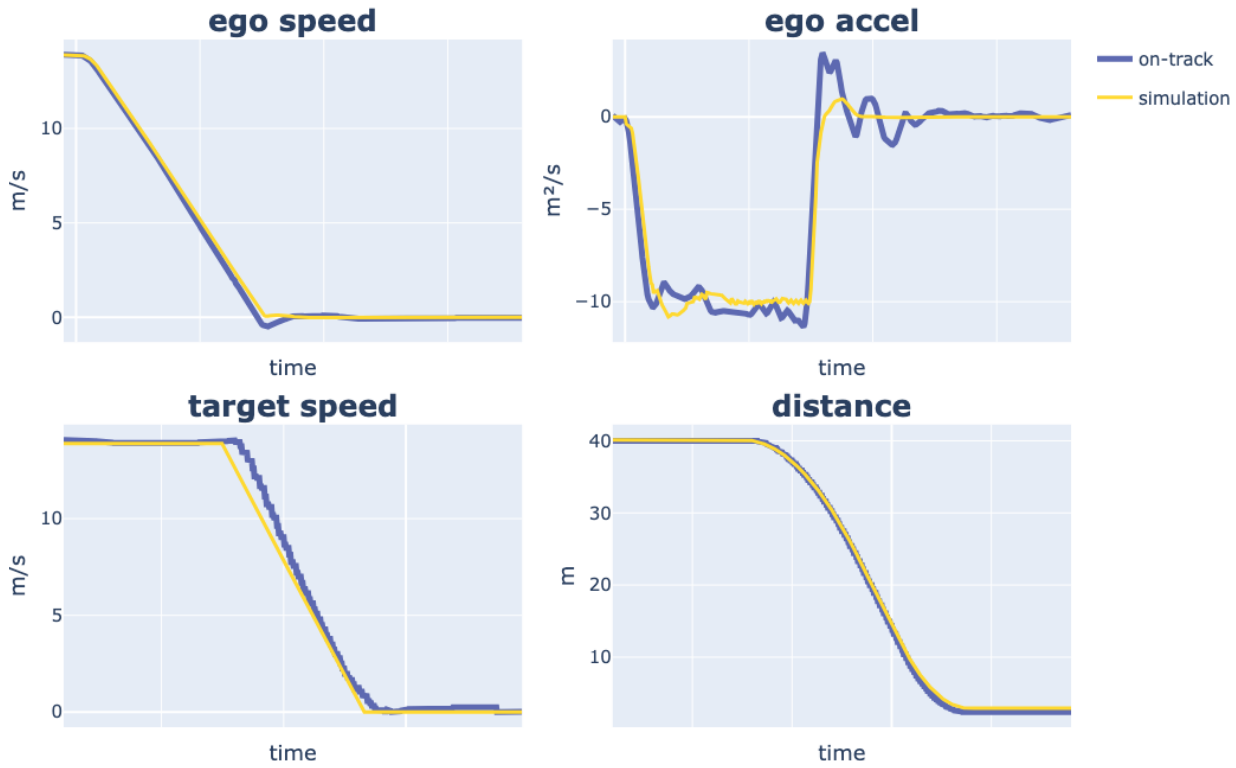


Figure 5.2: Séries temporelles de référence faites sur piste et un exemple de simulation

Les valeurs que l'on utilisera des paramètres nominaux sont données en Table 5.1. Nous explorerons les intervalles définis par les valeurs autorisées par le simulateur pour le scénario spécifique que nous considérerons.

La Figure 5.2 montre quatre séries temporelles de référence faites sur piste et des séries temporelles simulées avec les valeurs nominales associées. Les séries temporelles nominales simulées paraissent similaires aux séries temporelles de référence. Bien qu'elles soient assez proches, notre but est d'affiner les paramètres du simulateur pour obtenir une simulation encore plus proche. La Figure 5.3 démontre l'efficacité de notre approche. Cette figure illustre comment, en ajustant avec précision les paramètres, il est possible d'améliorer les séries temporelles simulées, en les alignant encore plus avec les valeurs obtenues sur piste.

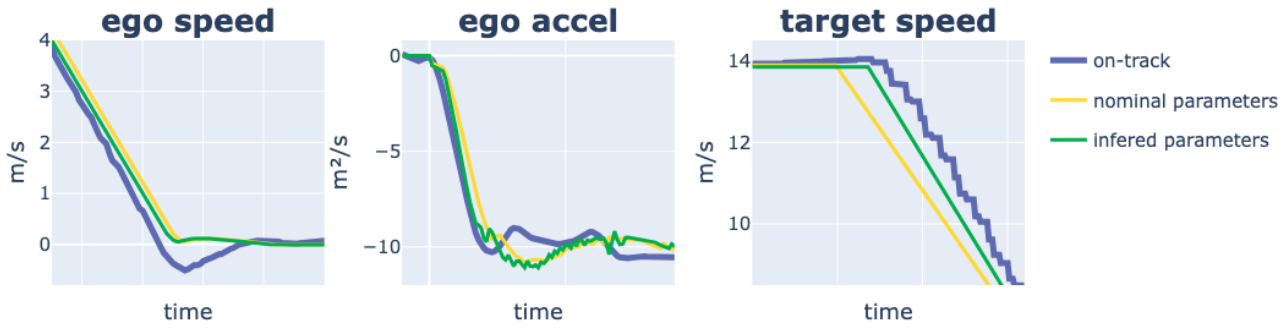


Figure 5.3: Comparaison des séries temporelles simulées avec les paramètres nominaux vs. les paramètres calibrés

Il est à noter que nous considérons ici un scénario impliquant un système AEB. Dans ce genre de cas, en injectant les paramètres nominaux dans le simulateur, les séries temporelles simulées obtiennent des résultats plutôt satisfaisants en reproduisant fidèlement les séries temporelles faites sur piste. Cependant, il est important de garder en tête que ce n'est pas toujours le cas. Dans d'autres contextes, le simulateur peut avoir un comportement sous-optimal en utilisant les paramètres nominaux. Par conséquent, nos efforts pour optimiser les paramètres sont alors nécessaires. En affinant ces paramètres, nous pouvons améliorer de manière significative la précision des séries temporelles simulées, en veillant à ce qu'elles reflètent fidèlement les conditions et les comportements du monde réel. C'est pourquoi ce processus de calibration est essentiel pour améliorer la validité et l'utilité des données simulées dans tous les cas de figure.

5.1.2 Problème inverse

Pour **calibrer le simulateur**, nous avons pour objectif de résoudre le **problème inverse** suivant : nous avons un essai de référence décrit par des séries temporelles sur piste et nous voulons trouver les paramètres d'entrée qui permettront de simuler les séries temporelles les plus proches à celles de référence.

Définition d'un problème inverse

En machine learning, les problèmes sont caractérisés par une relation entre entrée et sortie qui est encapsulée dans un modèle. Pour une entrée spécifique x, la sortie y est calculée en utilisant le modèle f au travers de la formule $y = f(x)$. L'objectif premier du machine learning est de comprendre et caractériser la relation qui existe entre l'entrée x et la sortie y. Il est essentiel de comprendre ce lien car il dicte le comportement du modèle. En fonction de la nature du problème considéré et selon les hypothèses sous-jacentes, cette relation peut être déterministe ou probabiliste. Un modèle déterministe renverra systématiquement la même sortie pour des conditions d'entrée équivalentes, tandis qu'un modèle probabiliste incorporera de l'aléatoire, permettant de générer différentes sorties même pour des entrées similaires.

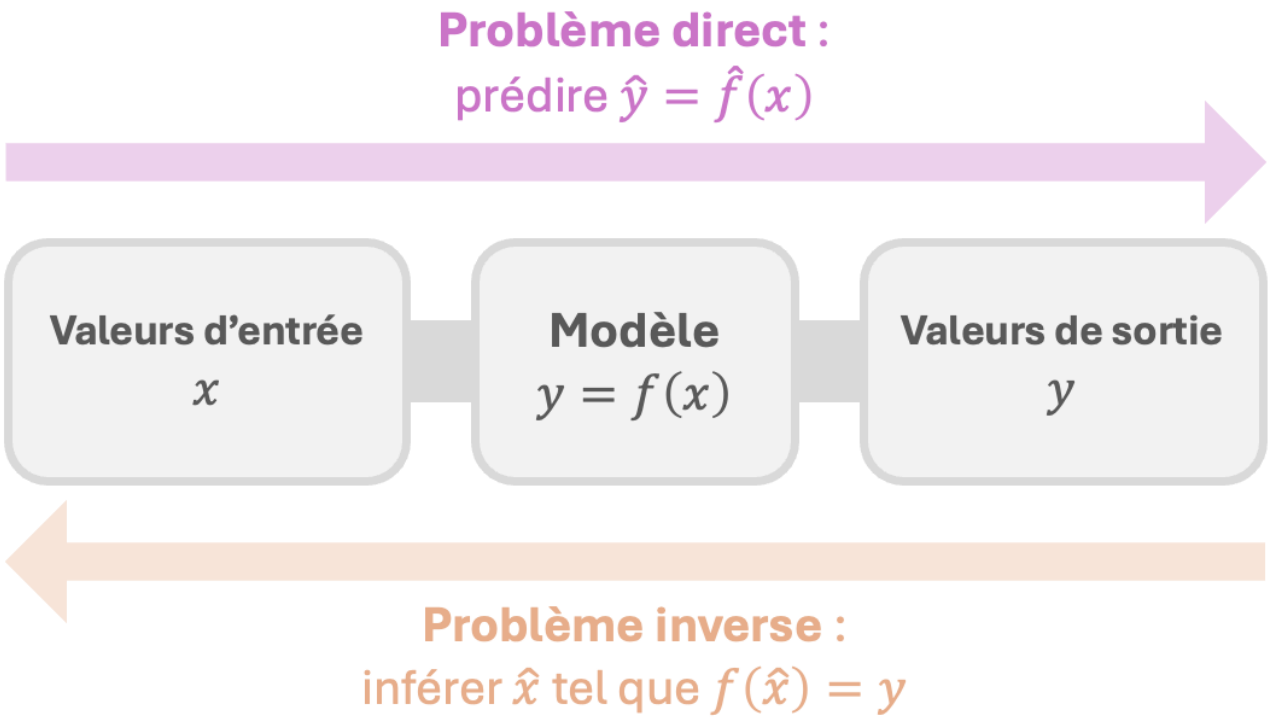


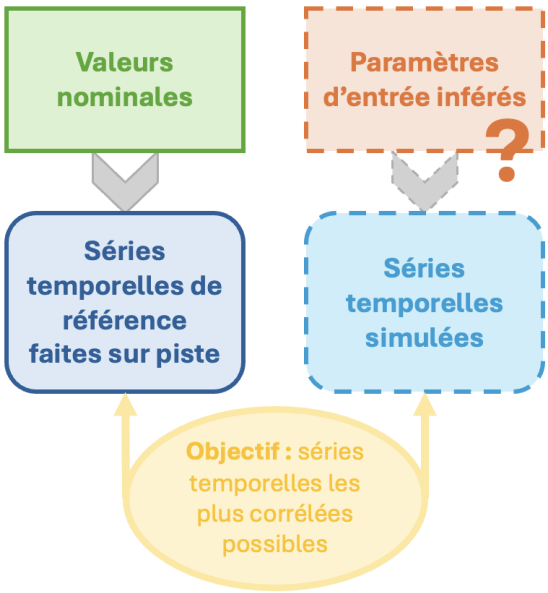
Figure 5.4: Schéma des problèmes direct et inverse

Il existe deux types de problèmes fondamentaux. Il sont représentés en Figure 5.4 et définis ci-après.

- **Problème direct.** Il faut prédire la sortie y en se basant sur une entrée spécifique x. Pour cela, il faut construire un estimateur $\widehat{f}$ qui reproduit le plus fidèlement possible le comportement de f, permettant ainsi des prédictions fidèles de y pour tout x donné. La précision et la performance de $\widehat{f}$ doivent être évaluées et sont cruciales car elles impactent directement la fiabilité des prédictions faites par le modèle.
- **Problème inverse.** Contrairement au problème direct, ici le but est d'estimer l'entrée x pour une sortie observée donnée y. Il faut alors inférer les valeurs de x qui auraient pu conduire au y observé. Il s'agit de réaliser une rétro-analyse du modèle f afin de déterminer quelles entrées produiraient la sortie observée dans le monde réel lorsqu'elles sont utilisées par le modèle. Ce processus est souvent compliqué et requiert des méthodes statistiques sophistiquées afin de traiter la non-unicité et l'instabilité potentielles de la solution, en particulier dans les cas où différentes entrées peuvent produire des sorties équivalentes.

Résoudre le problème inverse

L'objectif de résoudre notre problème inverse est brièvement décrit en Figure 5.5. Nous ajusterons les paramètres d'entrée du simulateur en fonction des erreurs identifiées entre les données simulées et les données réelles faites sur piste.



- **Incertitudes sur les valeurs nominales.** Les séries temporelles de référence faites sur piste sont données avec leurs paramètres d'entrée associés appelés valeurs nominales. Ces valeurs sont sujettes à des **incertitudes**, notamment en raison de l'imprécision des capteurs, des tolérances de réalisation ou d'erreurs dans les estimations numériques. Pour toutes ces raisons, il est possible d'améliorer considérablement les séries temporelles simulées en modifiant légèrement les valeurs nominales.

Figure 5.5: Schéma du problème inverse

• **Espaces des paramètres.** L'objectif est d'explorer les espaces de paramètre crées autour des valeurs nominales afin de trouver la combinaison de valeurs optimale, i.e., les paramètres qui permettront de simuler les séries temporelles les plus corrélées avec celles de référence.

Pour résoudre le problème inverse, nous étudions plusieurs options. Nous testerons des approches fréquentistes et bayésiennes.

Inférence bayésienne

Les statistiques bayésiennes reposent sur les probabilités conditionnelles, qui évaluent la probabilité qu'un événement A se produise étant donné qu'un autre événement B s'est déjà produit. Elle est désignée par $p(A|B)$. La probabilité conditionnelle $p(A|B)$ peut être calculée à l'aide de la formule $p(A|B) = p(A \cap B)/p(B)$, à condition que la probabilité de l'événement B, $p(B)$, soit non nulle. Cette formule décrit efficacement la manière dont l'occurrence de B modifie la probabilité de A et résume l'essence de l'inférence bayésienne, où les croyances sont mises à jour avec de nouvelles réalisations. De telles méthodologies sont au cœur de nombreuses applications en statistique et en apprentissage automatique, permettant une prise de décision plus éclairée en cas d'incertitude.

L'inférence bayésienne représente une approche fondamentale de l'inférence statistique, dont l'objectif est de faire des constatations probabilistes sur des paramètres inconnus ou des hypothèses sur la base de données observées. Cette méthode s'oppose à l'inférence fréquentiste en incorporant des croyances a priori sur les paramètres avant que les données ne soient observées. Le processus commence par la formulation d'une hypothèse sur une population, puis collecte des données et enfin, les utilise pour mettre à jour la probabilité de l'hypothèse.

Considérons deux variables aléatoires θ et Y. Le théorème de Bayes fournit un lien entre ces variables, nous permettant de mettre à jour nos connaissances sur θ après avoir observé Y. Le théorème est donné par

$$\pi(\theta|Y) = \frac{p(Y|\theta)\pi_0(\theta)}{p(Y)}\,.$$

Dans cette formule, $\pi(\theta|Y)$ est la distribution postérieure, représentant notre croyance actualisée sur θ après avoir observé Y. Le terme $p(Y|\theta)$ est la fonction de vraisemblance, qui mesure la plausibilité de l'observation de Y compte tenu de θ. La distribution a priori est $\pi_0(\theta)$, reflétant nos croyances sur θ avant l'observation de toute donnée. Enfin, $p(Y)$ est la vraisemblance marginale, parfois appelée évidence, et est calculée en intégrant la vraisemblance sur toutes les valeurs possibles de θ, pondérée par la distribution a priori

$$p(Y) = \int p(Y|\theta)\pi_0(\theta)\,\mathrm{d}\theta\,.$$

Cette intégrale correspond à la constante de normalisation qui permet de normaliser la valeur postérieure afin qu'elle constitue une distribution de probabilité valide.

L'inférence bayésienne consiste donc à mettre à jour nos croyances antérieures à la lumière de nouvelles données, un processus essentiel à la prise de décision dans des conditions d'incertitude. La distribution postérieure englobe tout ce que l'on sait sur la variable θ après avoir pris en compte les nouvelles données Y et nos connaissances préalables, ce qui en fait un outil puissant pour diverses applications statistiques.

Notre intérêt principal réside dans le calcul de la distribution postérieure $\pi(\theta|Y = y_{\text{obs}})$, qui représente la distribution de probabilité de la variable aléatoire θ compte tenu des données y_{obs} qui ont été observées. Une fois la distribution a posteriori établie, une tâche courante consiste à estimer une valeur ponctuelle du paramètre θ qui représente le mieux la distribution postérieure. Cette estimation est généralement réalisée à l'aide du critère du maximum a posteriori (MAP), qui sélectionne la valeur de θ qui maximise la distribution. Mathématiquement, ce critère s'exprime comme suit

$$\widehat{\theta}_{\text{MAP}} \in \arg\max_{\theta \in \Theta} \pi(\theta|Y = y_{\text{obs}}) \,.$$

Cette valeur est particulièrement importante car elle correspond à la valeur la plus probable de θ compte tenu des données observées, en considérant à la fois la probabilité de θ compte tenu de Y et les croyances préalables sur θ.

Dans des contextes particulièrement complexes, le calcul de la fonction de vraisemblance n'est pas réalisable en raison de la complexité du modèle ou de la nature des données. Dans ces cas, les méthodes *likelihood-free* offrent une solution permettant l'estimation de la distribution postérieure sans calculer explicitement la vraisemblance. Ces méthodes sont particulièrement utiles lorsque l'on est capable de simuler des données à partir du modèle mais que l'évaluation directe de la vraisemblance ne l'est pas.

Une approche courante dans les méthodes likelihood-free consiste à générer des échantillons qui se rapprochent de la distribution postérieure. Pour ce faire, on tire un grand nombre d'échantillons $\theta_1, \ldots, \theta_n$ selon la distribution puis ils sont utilisés pour construire une représentation empirique de la distribution a posteriori donnée par

$$\pi(\theta|Y = y_{\text{obs}}) \approx \frac{1}{n} \sum_{i=1}^{n} \delta_{\theta_i}(X) \,,$$

où δ_{θ_i} désigne la mesure de Dirac centrée sur θ_i. Au fur et à mesure que la taille de l'échantillon n augmente, la distribution empirique formée par ces pics de Dirac converge vers la véritable distribution postérieure.

La méthode de rejet est un algorithme likelihood-free qui permet d'échantillonner des distributions de probabilité complexes lorsque l'échantillonnage direct est difficile ou que la vraisemblance est incalculable. Pour mettre en œuvre la méthode de rejet, il faut spécifier une fonction de perte ℓ et sélectionner un paramètre de tolérance approprié h. Les étapes de la méthode de rejet sont les suivantes.

Algorithm 8 Échantilloneur de rejet générique

Initialization:

- Échantillonner les particules indépendamment selon le prior $\theta_1, \ldots, \theta_n \sim \pi_0$

Iterations:

1: **for** $k = 0, \ldots, K$ **do**
2: Générer les données associées aux particules y_i, $\forall i = 1, \ldots, n$
3: Déterminer les particules acceptées

$$A_k = \{i : \ell(y_i, y_{\text{obs}}) \leq h\} \subseteq \{1, \ldots, n\}$$

4: Sauvegarder les particules acceptées $(\theta_i)_{i \in A_k}$
5: **Mise à jour:** Générer les nouvelles particules $\theta_1, \ldots, \theta_n \sim \pi_0$

Le paramètre de tolérance h joue un rôle important dans cette méthode. En fixant h à une valeur faible, on obtient un critère d'acceptation plus strict, ce qui peut se traduire par un taux de rejet plus élevé, mais une meilleure approximation de la vraie valeur postérieure. Inversement, une valeur h plus grande augmente généralement le taux d'acceptation mais peut compromettre la qualité de l'approximation.

Le choix de la fonction de perte appropriée ℓ est critique également, car elle doit être spécifiquement adaptée aux caractéristiques des données et aux exigences du problème considéré. La fonction de perte détermine la manière dont les prédictions d'un modèle sont évaluées. En fonction des objectifs et des contraintes propres aux différentes problématiques, diverses choix de fonction de perte peuvent s'avérer nécessaires.

Dans le contexte des problèmes inverses, la fonction de perte peut inclure des pénalités pour la complexité ou les erreurs par rapport aux connaissances a priori ou au comportement attendu. Ceci est particulièrement important lorsque le calcul direct des vraisemblances n'est pas réalisable.

La résolution du problème inverse est traitée dans le Chapitre 4.

5.1.3 Modèle de substitution

Une étape cruciale de tous les algorithmes likelihood-free consiste à générer la sortie y liée aux entrées candidates θ (voir la ligne 2 de l'Algorithme 8). Les deux sont liés par le simulateur S : pour un ensemble donné de paramètres, des séries temporelles sont générées

$$y = S(\theta)\,,$$

où $\theta \in \mathbb{R}^p$ et $y \in \mathbb{R}^T$. Ici, y concatène plusieurs types de séries temporelles les unes après les autres, nous avons donc $\sum_{r=1}^{R} T_r = T$ où T_r est la taille de la r-ième série temporelle. Dans le contexte donné par la Description des données, nous avons quatre types distincts de séries temporelles : la vitesse d'ego, l'accélération d'ego, la vitesse de target et la distance entre les deux, donc $R = 4$.

Dans les algorithmes likelihood-free, l'utilisation du simulateur S serait la solution optimale pour générer des séries temporelles basées sur les paramètres d'entrée, mais elle nécessite des ressources informatiques excessives. Pour résoudre ce problème, nous remplaçons le simulateur par un modèle de substitution pré-entraîné, qui imite le simulateur et prédit y pour un θ donné en utilisant $\widehat{S}(\theta)$.

La construction d'un tel modèle est l'un des objectifs fondamentaux du machine learning : résoudre un problème direct avec de l'apprentissage statistique supervisé. Notre objectif est de prédire une variable de sortie Y appartenant à un espace $\mathcal{Y}$, sur la base d'une variable d'entrée θ provenant d'un espace Θ. On suppose que Θ et $\mathcal{Y}$ sont des espaces mesurables, ce qui facilite l'application de la théorie des probabilités et des mesures. Une fonction de perte $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ est introduite pour quantifier la précision des prédictions. Cette fonction calcule l'erreur de prédiction $\ell(\widehat{y}, y)$, où $\widehat{y}$ est la sortie prédite et y est la valeur réelle.

L'espace de tous les prédicteurs potentiels, désigné par $\mathcal{F}$, est constitué de toutes les fonctions mesurables f qui vont de Θ dans $\mathcal{Y}$. Le choix de $\mathcal{F}$ est important car il définit l'ensemble de tous les fonctions possibles qui peuvent être utilisés pour prédire y à partir de θ.

Étant donné une distribution de probabilité jointe P sur le couple (θ, Y), le risque associé à un prédicteur f est défini comme l'espérance de la fonction de perte sur cette distribution

$$\mathcal{R}_P(f) = \mathbb{E}_{(\theta,Y)\sim P}\big[\ell(f(\theta), Y)\big]\,.$$

Mathématiquement, le risque quantifie la performance moyenne du prédicteur f sur l'ensemble de la distribution des données, ce qui permet de mesurer la capacité de généralisation de f à de nouvelles données inédites. L'objectif est d'identifier le prédicteur f dans $\mathcal{F}$ qui minimise ce risque, ce qui est essentiel pour développer des modèles de prédiction efficaces.

Pour un échantillon donné $(\theta_1, y_1), \ldots, (\theta_n, y_n)$ i.i.d. de la distribution P sur $\Theta \times \mathcal{Y}$, l'objectif est de construire un prédicteur $\widehat{f}_n : \Theta \to \mathcal{Y}$ qui minimise le risque empirique

$$\widehat{f}_n \in \underset{\widehat{f} \in \mathcal{F}}{\arg\min} \left\{ \mathcal{R}_n(\widehat{f}) = \frac{1}{n} \sum_{i=1}^{n} \ell\big(\widehat{f}(\theta_i), y_i\big) \right\}$$

pour une fonction de perte donnée ℓ. Le risque empirique est un estimateur sans biais du risque $\mathcal{R}_P(\widehat{f}_n)$.

Le modèle de substitution peut être construit à l'aide de diverses méthodes plus ou moins sophistiquées, comme les forêts aléatoires, les réseaux neuronaux convolutifs, les réseaux neuronaux récurrents,..., etc. Nous résolvons ce problème dans le Chapitre 2.

Comme mentionné précédemment dans la Section 5.1.2, la fonction de perte ℓ doit être choisie avec soin. Ce choix peut être légèrement différent en fonction du contexte. Dans le cas du modèle de substitution, nous visons principalement à capturer la précision des prédictions et la proximité des sorties estimées par rapport aux valeurs réelles. En sélectionnant une fonction de perte appropriée, nous nous assurons que le prédicteur ne minimise pas seulement le risque théorique, mais qu'il s'aligne aussi étroitement sur les besoins pratiques du monde réel et sur les caractéristiques spécifiques des données.

5.2 Contributions

Chapitre 2. Construction du modèle de substitution

Les modèles de substitution sont grandement utilisés dans une large variété de domaines pour améliorer l'efficacité et la précision des outils. Des méthodes telles que les expansions polynomiales du chaos [Sraj et al., 2016], les fonctions de base radiales et le Krigeage [Beglerovic et al., 2017], les modèles de substitution bayésiens [Ford et al., 2011], les modèles de substitution à surface de réponse [Mattis and Wohlmuth, 2018], les réseaux de neurones artificiels [Xu et al., 2020] peuvent être utilisées. Les modèles de substitution se sont également révélés particulièrement utiles dans l'industrie automobile pour des applications telles que les sièges de voiture [Long et al., 2021], les composants de suspension [Jiang et al., 2021], l'interaction *human-product* [Ahmed et al., 2018] ou la validation des véhicules autonomes [Beglerovic et al., 2017].

Le Chapitre 2 se concentre sur la construction d'un modèle de substitution utilisant l'apprentissage automatique supervisé pour reproduire et remplacer le simulateur numérique de Renault. Ce modèle vise à prédire les séries temporelles simulées sur la base de paramètres d'entrée spécifiques, servant de modèle génératif. Il est construit à partir d'un ensemble de données simulées en amont et est conçu pour reproduire les résultats du simulateur tout en garantissant des temps de calcul plus rapides.

Le modèle de substitution doit répondre à deux caractéristiques essentiels : il doit être efficace sur le plan computationnel pour accélérer le processus de calibration et très précis pour remplacer efficacement le simulateur. L'objectif est de créer un estimateur du simulateur fiable qui puisse gérer la génération de sorties multivariées.

Le chapitre aborde l'utilisation de différents formats de données, en testant les techniques traditionnelles d'apprentissage automatique comme les k-*nearest neighbors* (Section 2.3.1), les forêts aléatoires (Section 2.3.2), et la régression ridge à noyau (Section 2.3.3), en adaptant l'expansion du chaos polynomial aux contextes fonctionnels (Section 2.4) ; l'application de la réduction des dimensions par l'analyse des composantes principales (Section 2.5) ; le développement de plusieurs architectures de réseaux de neurones avec la *deep forest* (Section 2.6.1), le réseau neuronal récurrent (Section 2.6.2) et le réseau de neurones convolutif (Section 2.6.3) ; et l'exploration de modèles hybrides et agrégés (Section 2.7).

Ces discussions visent à perfectionner le modèle de substitution pour en faire un substitut viable au simulateur numérique, en garantissant à la fois la rapidité et la précision dans la génération de séries temporelles. Nous visons à fournir un catalogue de modèles d'apprentissage automatique supervisé ad hoc adaptés à notre contexte spécifique et fonctionnant de manière optimale en termes d'efficacité et de performance de calcul.

Chapitre 3. Garanties théoriques pour l'agrégation d'experts fonctionnels

Remarque. Le travail de ce chapitre a été réalisé en collaboration avec Hugo Chardon et Étienne Donier-Meroz. Il ménera à la publication d'un article. Nous avons contribué de manière égale à la conception et à la rédaction.

Dans le chapitre précédent, nous avons construit des modèles de substitution pour le simulateur numérique de Renault, avec pour objectif de prédire des séries temporelles à partir de paramètres d'entrée non-temporels. Nous nous sommes notamment intéressés à l'agrégation d'experts, qui démontrent de bonnes performances. Elle consiste à combiner linéairement plusieurs prédicteurs, appelés experts. Comme les experts sont des séries temporelles, certains peuvent être meilleurs que d'autres à certains pas de temps, les poids dépendront alors également du temps. Les méthodologies standards d'agrégation discrétisent les séries temporelles et les poids. Bien que la discrétisation soit numériquement nécessaire, elle ne devrait pas être inclue dans l'analyse théorique des algorithmes. Par ailleurs, le contexte de ma thèse est assez spécifique. En effet, contrairement à la grande majorité des cas où les valeurs du passé sont utilisées pour prédire le futur, ici nous souhaitons générer l'entièreté de la série temporelle à partir de paramètres d'entrée non-temporels.

Cette démarche, bien que fortement motivée par ma thèse, est adaptable à de nombreux domaines comme la physique, l'économie, la biologie, ..., tous les contextes où les données sont des fonctions continues. Par exemple, en sciences sociales, lorsque l'on souhaite prédire le produit intérieur brut en fonction d'indicateurs démographiques, économiques et géographiques. En biologie, lorsque l'on s'intéresse à l'évolution d'une espèce en fonction de son taux de reproduction, de la disponibilité de la nourriture et d'autres facteurs.

Considérer les séries temporelles et les poids comme des fonctions continues du temps et chercher à générer l'entièreté de la série temporelle définissent un problème particulier. Il est alors nécessaire de définir un nouveau cadre avec des espaces d'entrée et de sortie, un modèle, une fonction de perte et un risque spécifiques. L'objectif est de généraliser l'agrégation linéaire d'experts à des prédicteurs et poids fonctionnels. L'introduction de cette composante continue doit être rigoureusement réalisée afin de démontrer la convergence des algorithmes, sous couvert des bonnes conditions.

Une fois ce nouveau cadre rigoureusement défini, nous nous intéresserons aux garanties théoriques de la descente de gradient projeté (PGD) déterministe et stochastique, ainsi que la descente de gradient miroir (MGD) stochastique. Les vitesses de convergence de ces approches sont étudiées et décrites : dans Rigollet [2015] pour le PGD déterministe, dans Lacoste-Julien et al. [2012] pour le PGD stochastique et dans Bubeck [2015] pour le MGD stochastique. L'objectif ici est de vérifier que ces garanties restent valables et de les adapter à notre nouveau cadre si nécessaire.

Chapitre 4. Résolution du problème inverse

L'objectif de ce chapitre est de résoudre le problème inverse en recréant les séries temporelles faites sur piste avec le simulateur. Pour cela, il nous faut identifier les paramètres d'entrée qui, une fois injectés dans le simulateur, permettront de générer les séries temporelles les plus proches possibles de la réalité.

Mathématiquement, soit y_φ les séries temporelles de référence faites sur piste et θ_0 leurs paramètres d'entrée nominaux associés, nous voulons retrouver les paramètres d'entrée θ^* tels que $y^* := S(\theta^*)$ est le proche possible de y_φ. Pour une distance d donnée et un ensemble Θ à explorer, on doit résoudre $\theta^* \in \arg\min_{\theta\in\Theta} d\big(S(\theta), y_\varphi\big)$.

Les valeurs nominales θ_0 sont sujettes à des incertitudes, principalement à cause d'imprécisions des capteurs, de tolérances de réalisation ou d'erreur computationnelle. C'est pour ces raisons que souvent $\theta^* \neq \theta_0$. Par ailleurs, il est alors possible de grandement améliorer les séries temporelles simulées en raffinant les valeurs nominales et en inférant θ^*. L'espace à explorer Θ est défini selon les valeurs nominales θ_0, ce qui permet de fournir des informations a priori et ainsi guider les recherches pour trouver la solution optimale.

Le simulateur S, qui lie les paramètres d'entrée aux séries temporelles en sortie, présente une erreur inhérente par rapport aux essais faits sur piste. Par conséquent, il est nécessaire d'intégrer un bruit additif dans notre modèle statistique afin de modéliser cette différence. Ce chapitre explore différentes hypothèses concernant la nature de ce bruit et discute des résultats correspondants.

Dans un premier temps, nous supposons que le bruit est gaussien avec différents types de matrices de covariance. Dans ce cadre, nous démontrons que le calcul de l'estimateur du maximum de vraisemblance équivaut à minimiser l'erreur quadratique moyenne (MSE). Nous évaluons ensuite les performances de cette approche à l'aide d'une méthode fréquentiste en optimisant la MSE puis d'une approche bayésienne en utilisant la vraisemblance synthétique bayésienne (BSL).

Par la suite, nous relâchons l'hypothèse d'un bruit gaussien et considérons un bruit dont la distribution est inconnue. Dans ce cas, notre objectif est de tester l'efficacité d'une approche fréquentiste en optimisant la fonction objective $\theta \mapsto d\big(S(\theta), y_\varphi\big)$, puis d'explorer les méthodes bayésiennes en utilisant des échantillonneurs de Monte Carlo séquentiels (SMC).

Enfin, nous démontrons l'efficacité des méthodes bayésiennes qui permettent d'incorporer des informations a priori à l'aide du prior. De plus, l'échantillonneur SMC développé pour ce problème spécifique, peut également être utilisé dans d'autres contextes et donner de bons résultats. Il faut voir ce travail comme le développement d'une méthodologie générique et automatique.

Appendices

Appendix A

Noise decomposition

In general, a time series Yt is decomposed into three components: (1) the trend T_t, which captures the long-term orientation; (2) the seasonality S_t, which comprises what is repeated; and (3) the error or residual ε_t, which has a zero mean and must have a low variance compared to the other two. This decomposition can be additive $Y_t = T_t + S_t + \varepsilon_t$, or multiplicative $Y_t = T_t S_t(1+\varepsilon_t)$. Furthermore, it is possible to define additive and multiplicative compositions $Y_t = (T_t + S_t)\varepsilon_t$.

The objective is to identify the residuals ε_t by estimating the trend T_t and seasonality S_t. Thus, we recover noise consistent with our data by randomly generating white noise and adding T_t and S_t. The time series to be decomposed is illustrated in Figure A.1. It corresponds to the difference in absolute value between the reference on-track time series y_φ and the time series simulated with the nominal parameters $S(\theta_0)$.

Upon initial examination, the data appears to lack any discernible seasonality. To recover a stationary time series, we need to detrend it to identify the residuals. Therefore, we will assume an additive composition.

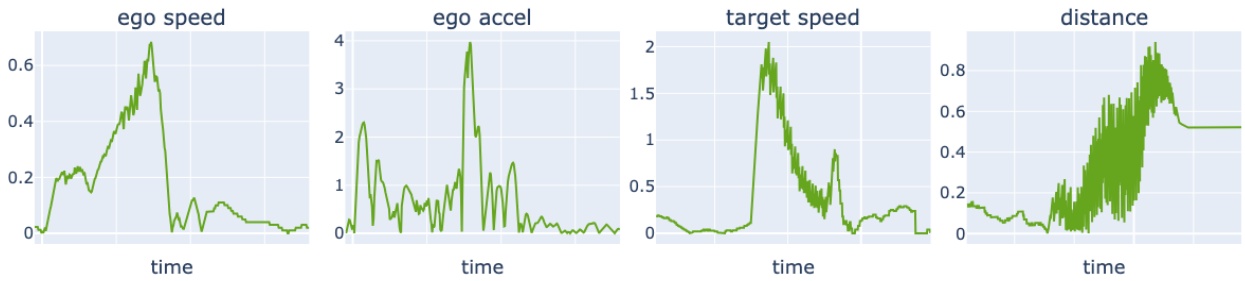


Figure A.1: Initial noise

The concepts of autocorrelation (ACF) and partial autocorrelation (PACF) represent two mathematical tools that permit the assessment of a time series' stationarity. The autocorrelation calculation measures the temporal dependence by computing the correlation between a time series and a time-lagged version of itself.

Let us first define the covariance and the correlation between two variables X_1 and X_2

$$\begin{aligned}\text{cov}(X_1, X_2) &= \mathbb{E}[X_1 X_2] - \mathbb{E}[X_1]\mathbb{E}[X_2] \\ \text{corr}(X_1, X_2) &= \frac{\text{cov}(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}}\end{aligned}$$

where σ_{X_1} and σ_{X_2} are the standard deviations.

Let Y be our time series of interest. Its autocovariance of order h is defined by $\gamma_h(Y) := \text{cov}(Y_t, Y_{t+h})$ and its autocorrelation of order h is given by

$$\rho_h(Y) := \frac{\gamma_h(Y)}{\gamma_0(Y)}$$

where h is the considered time lag. If μ is the mean of Y and σ^2 its variance, we have

$$\rho_h(Y) = \frac{1}{\sigma^2}\mathbb{E}\left[(Y_t - \mu)\,(Y_{t+h} - \mu)\right].$$

The **autocorrelation** is a value that ranges from -1 to 1, with 1 indicating a perfect correlation and -1 indicating an ideal anti-correlation. The **partial autocorrelation** of order h for Y is defined by

$$\begin{aligned}r_1(Y) &= \rho_1(Y) \\ r_h(Y) &= r_{Y_2,\dots,Y_h}(Y_1, Y_{h+1})\end{aligned}$$

where

$$r_{Y_2,\dots,Y_h}(Y_1, Y_{h+1}) = \text{corr}\big(Y_1 - P_{M(Y_2,\dots,Y_h)}(Y_1),\ Y_{h+1} - P_{M(Y_2,\dots,Y_h)}(Y_{h+1})\big)$$

and $P_{M(Y_2,\dots,Y_h)}(Y) = \beta_2 Y_2 + \cdots + \beta_h Y_h$ defined the linear projection of Y on the space generated by the random variables and denoted $M(Y_2, \dots, Y_h)$. For more details, please refer to Goude [2024a].

In Figure A.2 is given the ACF and PACF of our initial time series represented in Figure A.1.

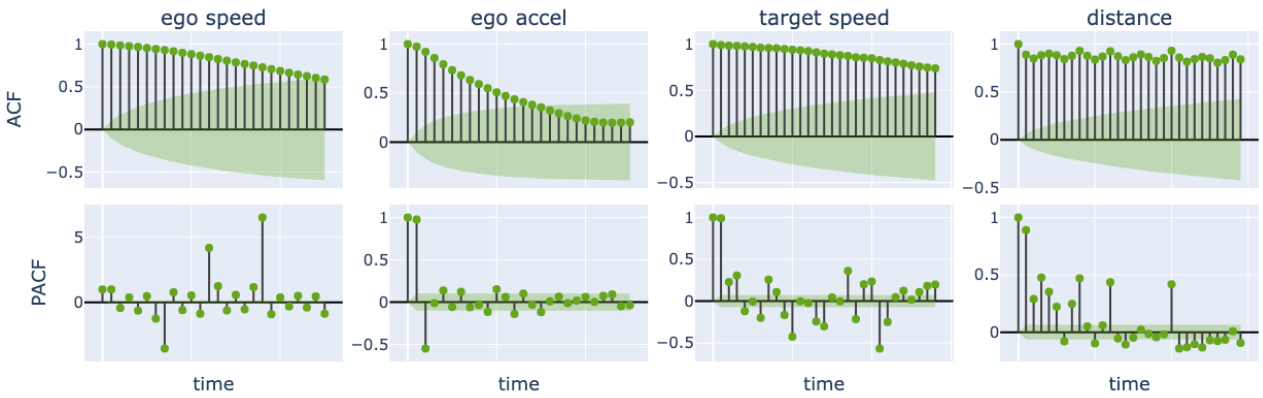


Figure A.2: Autocorrelation and partial autocorrelation of the initial noise

A.1 Decomposition examples

There are many methodologies for estimating the trend in a time series, including moving averages, linear regression, polynomial regression, non-parametric estimation (utilising local polynomials), and others. An example is given in Figure A.3. For our time series, we focus on local polynomials decomposition. We need to determine the best hyperparameters.

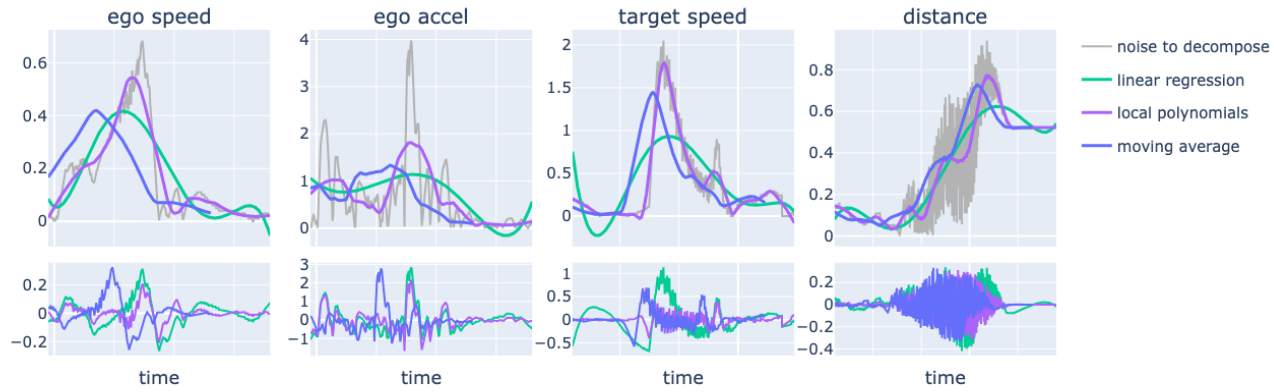


Figure A.3: Noise decomposition example

A.2 Local polynomials decomposition

For a given $h > 0$ and a kernel K, let us define the following function

$$W_{h,t}(x) = \frac{K\left(\frac{x-t}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-i}{h}\right)}.$$

As described in Goude [2024b], the local polynomial estimator of degree q of the time series Y, associated with the window h and the kernel K, is

$$\widehat{f}_h(x) = \arg\min_P \sum_{t=1}^{n} W_{h,t}(x)\|Y_t - P(x_t - x)\|^2$$

where $P(u) = \sum_{j=0}^{q} a_j u^j$ is a polynomial of degrees q.

Numerically, we used a Savitzky-Golar filter, which is made to smooth the data by using convolution and fitting successive sub-sets of adjacent data points with a low-degree polynomial by the method of linear least squares. For more details, please refer to Schafer [2011].

The goal is to determine the best polynomial degree q and the best window parameter h. We first take $q = 3$ and try distinct parameters h. All the results are presented in Figure A.4. The best smoothing for each type of time series is made with $h = 30$ as shown in Figure A.5.

The plots at the bottom of these two figures illustrate the discrepancy between the noise to be decomposed and the proposed smoothed curve.

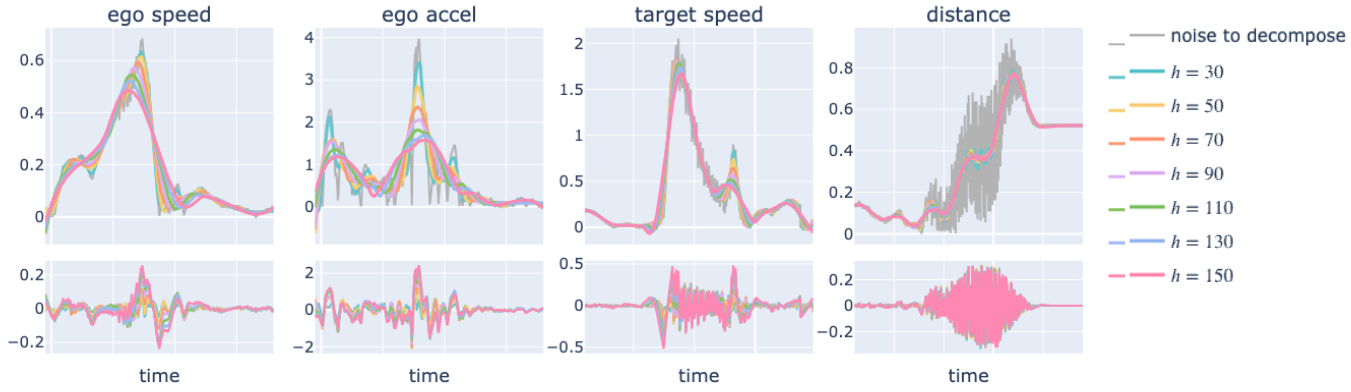


Figure A.4: Noise decomposition with local polynomials

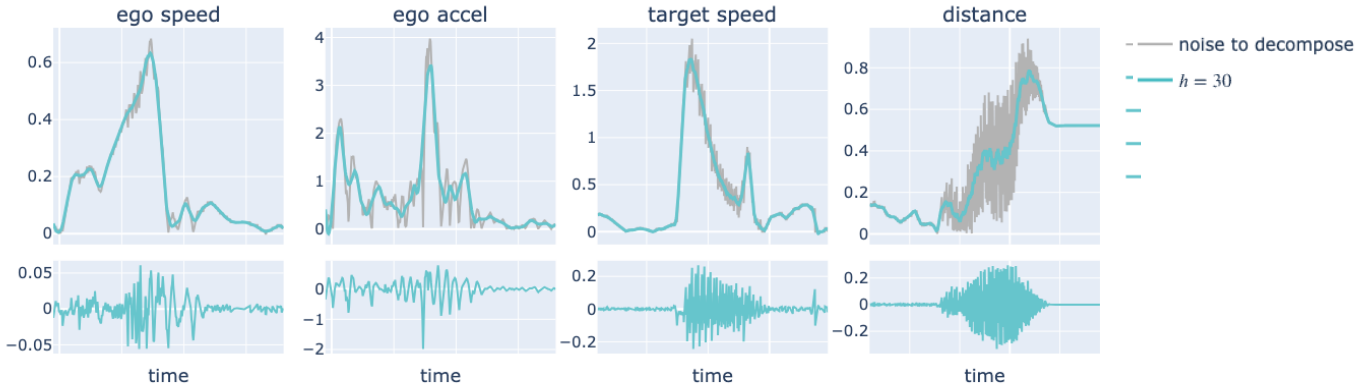


Figure A.5: Best local polynomials decomposition

A.3 Final result

By subtracting the previously estimated trend with $h = 30$, we obtain the final time series presented in Figure A.6. We can again study their autocorrelation and partial autocorrelation, which are presented in Figure A.7.

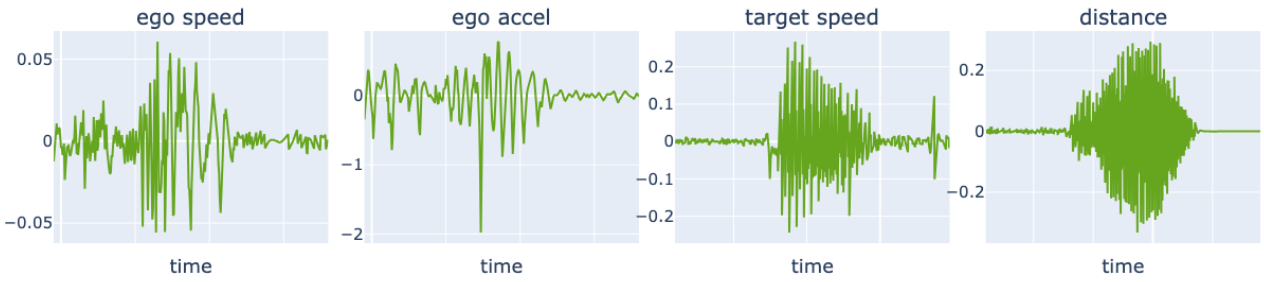


Figure A.6: Final noise

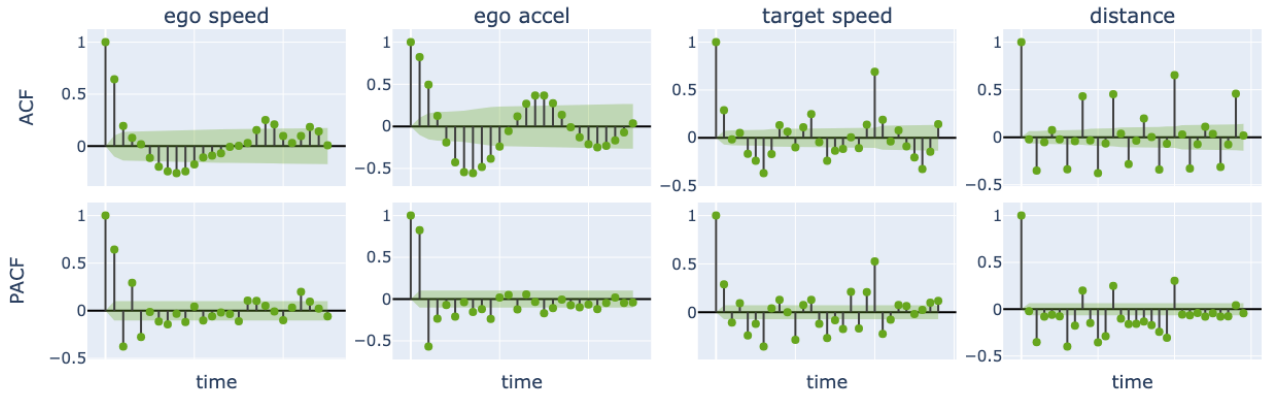


Figure A.7: Autocorrelation and partial autocorrelation of the final noise

The ACF indicates the presence of seasonality in ego speed and ego accel. However, upon examination of the time series' shape in Figure A.6, it is evident that this is merely noise rather than a genuine reflection of seasonality. An augmented Dickey-Fuller test [Hays, 2021] provides evidence that our detrend time series is stationary. The p-values presented in Table A.1 demonstrate that the ego speed and distance series were not stationary before detrending and are now stationary.

p-value	ego speed	ego accel	target speed	distance
initial noise	0.34	0.03	0.05	0.86
final noise	2×10^{-5}	2×10^{-11}	7×10^{-6}	1×10^{-21}

Table A.1: Dickey–Fuller test

Appendix B

Sensitivity analysis

It is common practice to attempt to establish the extent to which specific parameters influence the inputs and outputs of a model to inform the prediction process. Various methodologies have been employed, including feature importance and permutation importance coefficients in random forest models, Sobol indices, and Shapley-Shubik power indexes. For an in-depth reading about feature importance, please refer to Breiman [2001], and about Sobol and Shapley-Shubik indices, please refer to Broto [2020].

The efficacy of these various methodologies is evaluated on the data presented in Section 1.1.1. We have eight input parameters and four distinct types of time series as output for 2414 time steps.

B.1 Random forest importance

We have built a random forest model for each type of time series, and we compute the corresponding feature importance in Figure B.1 and the corresponding permutation importance in Figure B.2.

B.1.1 Feature importance

Feature importance is a fundamental sensitivity analysis tool. It involves calculating a score for each input feature to establish its relative importance in the decision-making process. The higher the score for a given feature, the greater its impact on the model's ability to predict a specific variable.

The front braking efficiency greatly influences the prediction of ego speed and acceleration. In second place comes ego initial speed. Regarding target speed and distance, target initial acceleration is the leading variable. Then comes the distance between ego and target and the ego initial speed.

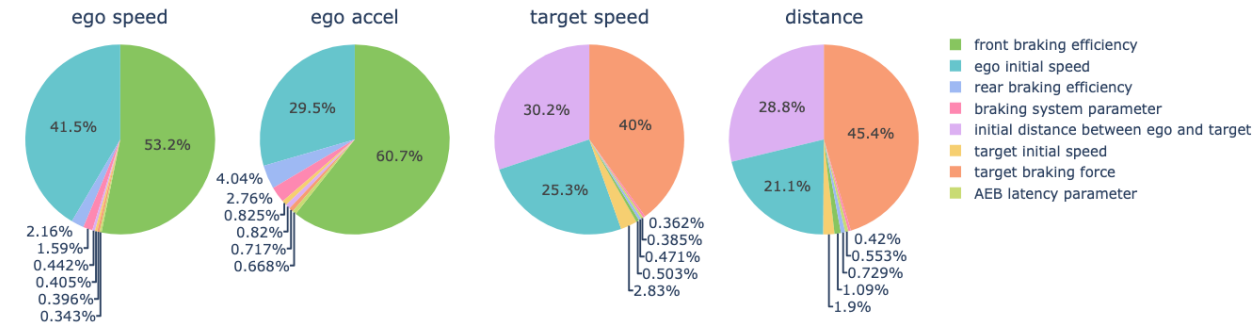


Figure B.1: Random forest feature importance

B.1.2 Permutation feature importance

As discussed in Altmann et al. [2010], feature importance has a critical bias. To correct this, they introduce the importance of permutation. Please refer to Molnar [2022, Section 8.5.] for a complete explanation.

The importance of the permutation feature quantifies the increase in the model's prediction error resulting from the random permutation of feature values. This process disrupts the relationship between the feature and the true outcome.

We assess a feature's significance by determining how much the model's prediction error grows when the feature's values are shuffled. If the error increases upon shuffling, the feature is necessary as the model depends on it to make predictions. Conversely, if the error remains the same when the feature's values are shuffled, the feature is deemed unimportant, indicating that the model does not rely on it for predictions.

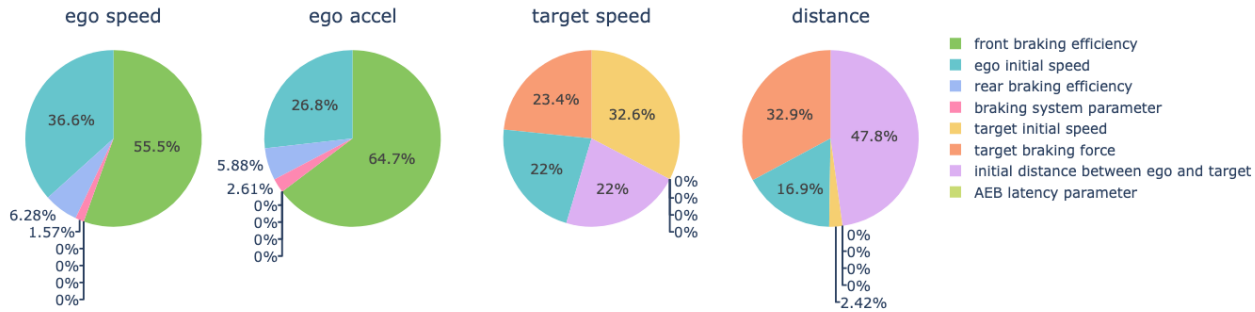


Figure B.2: Random forest permutation importance

The permutation importance metric yielded comparable results to those obtained through the classic feature importance approach, except for target initial speed, which emerged as one of the top four features influencing target speed.

B.2 Sobol' indices

Sobol indices represent a widely utilized approach to sensitivity analysis. They assess the extent to which the variance in a model's output can be attributed to the various input parameters. By quantifying each input variable's individual and joint influence on the output, Sobol indices provide an overview of the relative importance and interaction of parameters within a model.

There are three orders of Sobol indices. Their signification is described below.

- **First order:** indicates the proportion of output variance the input explains.
- **Second order:** evaluates interaction effects between two inputs.
- **Total order:** a high total order index indicates that the variable has a high influence on the output and interacts strongly with the other variables.

The first and total order results are expressed in percentages and presented in Figure B.3. The second order is given in Figure B.4.

Figure B.3: Sobol first and total order indices

First-order Sobol indices indicate that for ego speed series, front braking efficiency and ego initial speed collectively account for approximately 43% of the output variance. Similarly, for ego acceleration, front braking efficiency accounts for 47% of the variance, while ego initial speed accounts for 22%. Finally, initial distance, target braking force, and ego initial speed collectively account for 41%, 24%, and 18%, respectively, of the variance in distance.

The total order Sobol indices corroborate these findings, demonstrating that the abovementioned parameters remain paramount.

Second-order Sobol indices show no interaction between input variables.

Figure B.4: Sobol second order indices

B.3 Shapley-Shubik power index

The Shapley–Shubik power index was first introduced by Shapley and Shubik [1954]. It provides a rigorous metric for assessing the relative importance of individual characteristics in prediction models. Initially designed to assign equitable payoffs among players forming coalitions, these indices have been adapted to analyze the impact of each variable on a model's predictions. This approach enables the precise quantification of the contribution of each characteristic to the model's performance, facilitating the understanding and interpretation of the results obtained.

We use the same random forest models as before to compute the Shapley-Shubix indexes, and the results are presented in Figure B.5.

The Shapley-Shubik index results diverge from those of feature importance. The most influential parameter regarding ego speed and acceleration is ego initial speed. In the case of target speed, it is target initial speed, while in the context of distance, it is the distance between ego and target.

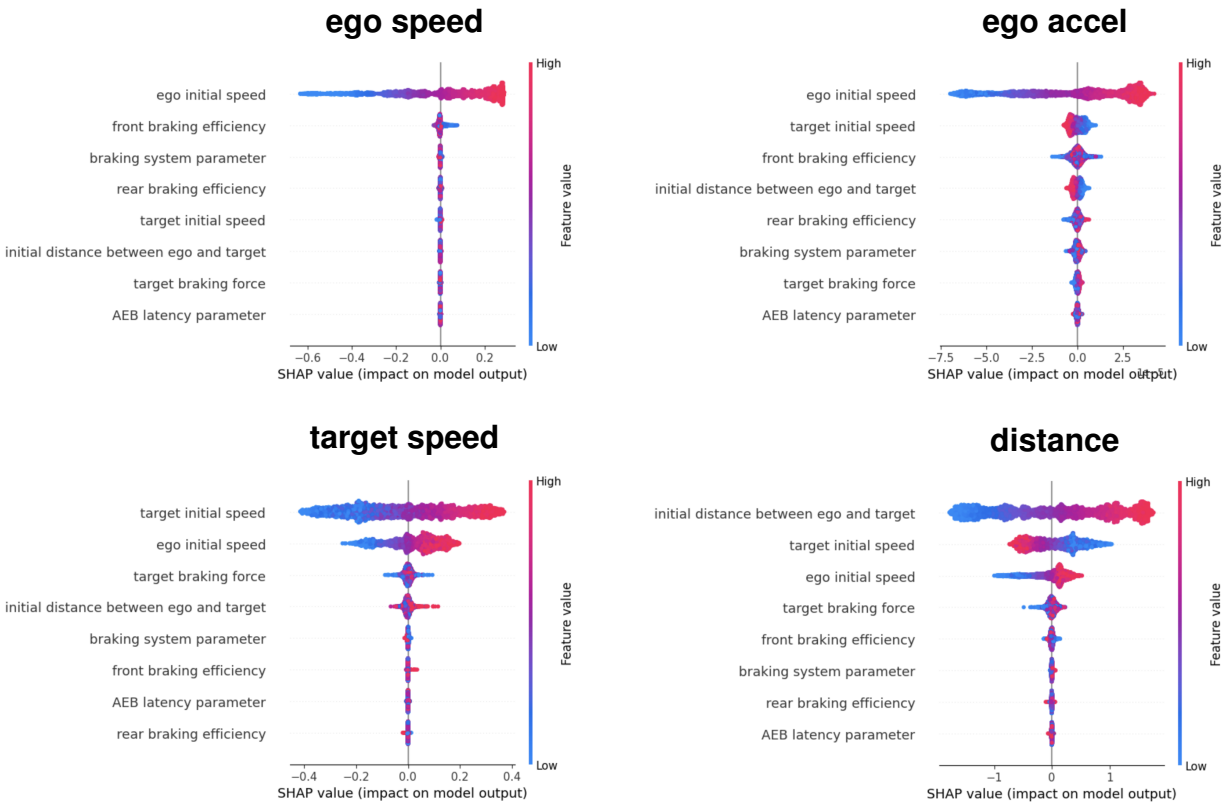


Figure B.5: Shapley-Shubik power index

B.4 General conclusion of the sensitivity analysis

The outcomes of the distinct methodologies are contingent upon the models constructed upstream by random forest. Moreover, the most pivotal input variable selected is not always the same, contingent upon the approach utilized. For ego speed and ego acceleration, both feature importance and permutation importance concur with the Sobol indices. Conversely, only permutation importance aligns with the Sobol indices and the Shapley-Shubik index for distance. For target speed, only permutation importance and the Shapley-Shubik index agree.

parameter	**feature importance**	**permutation importance**	**Sobol**	**Shapley-Shubik**
ego speed	front braking efficiency	front braking efficiency	front braking efficiency	ego initial speed
ego accel	front braking efficiency	front braking efficiency	front braking efficiency	ego initial speed
target speed	target braking force	target initial speed	ego initial speed	target initial speed
distance	target braking force	initial distance between ego and target	initial distance between ego and target	initial distance between ego and target

Table B.1: Most influential parameter selected by each method for each time series

Bibliography

Salman Ahmed, Mihir Sunil Gawand, Lukman Irshad, and H Onan Demirel. Exploring the design space using a surrogate model approach with digital human modeling simulations. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1B, 2018. 28, 33, 118

André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010. 131

Germán Aneiros, Silvia Novo, and Philippe Vieu. Variable selection in functional regression models: a review. *Journal of Multivariate Analysis*, 188:104871, 2022. 62

AVSimulation. https://www.avsimulation.com/scaner-studio/, 2023. 12, 106

Sumanta Basu and George Michailidis. Regularized estimation in sparse high-dimensional time series models. *The Annals of Statistics*, 43(4):1535 – 1567, 2015. 60

Halil Beglerovic, Michael Stolz, and Martin Horn. Testing of autonomous vehicles using surrogate models and stochastic optimization. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017. 28, 33, 118

David Benatia, Marine Carrasco, and Jean-Pierre Florens. Functional linear regression with functional response. *Journal of Econometrics*, 201(2):269–291, 2017. 63

José R Berrendero, Antonio Cuevas, and José L Torrecilla. On the use of reproducing kernel hilbert spaces in functional classification. *Journal of the American Statistical Association*, 113(523):1210–1218, 2018. 63

David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 1956. 50

Simon Blanke. Gradient-Free-Optimizers: Simple and reliable optimization with local, global, population-based and sequential techniques in numerical search spaces. https://github.com/SimonBlanke, 2024. 85, 94

Géraud Blatman and Bruno Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 230(6):2345–2367, 2011. 40

Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017. 48

Paul Bourke. Cross correlation. *Cross Correlation", Auto Correlation—2D Pattern Identification*, 1996. 22

Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996. 37

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001. 37, 130

Peter J. Brockwell and Richard A. Davis. *Introduction to time series and forecasting*. Springer, 2002. 60

Baptiste Broto. *Sensitivity analysis with dependent random variables: Estimation of the Shapley effects for unknown input distribution and linear Gaussian models*. Thesis, Université Paris-Saclay, 2020. 130

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015. 29, 69, 71, 74, 119

Clara Carlier, Arnaud Franju, Matthieu Lerasle, and Mathias Obrebski. Construction of a surrogate model: Multivariate time series prediction with a hybrid model. *Driving Simulation Conference 2023 Europe VR*, 8:143–149, 2023. 34, 51

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006. 50

Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geosci. Model Dev.*, 7: 1247–1250, 2014. 22

Christophe Chesneau. Introduction aux arbres de décision (de type CART). cel-02281064, December 2020. 37

Nicolas Chopin. Sequential Monte Carlo in python. https://particles-sequential-monte-carlo-in-python.readthedocs.io/en/latest/, 2024. 87

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on information theory*, 13(1):21–27, 1967. 37

Thierry Crestaux, Olivier Le Maıtre, and Jean-Marc Martinez. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering & System Safety*, 94(7):1161–1172, 2009. Special Issue on Sensitivity Analysis. 40

Dan Crisan and Arnaud Doucet. Convergence of sequential monte carlo methods. *Signal Processing Group, Department of Engineering, University of Cambridge, Technical Report CUEDIF-INFENGrrR38*, 1:525, 2000. 92

Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines. *CambridgeUniversityPress, Cambridge, UK*, 2000. 37

Antonio Cuevas. A partial overview of the theory of statistics with functional data. *Journal of Statistical Planning and Inference*, 147:1–23, 2014. 60

Marco Cuturi and Mathieu Blondel. Soft-DTW: a differentiable loss function for time-series. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 894–903. PMLR, 2017. 22, 24

Arnak Dalalyan and Alexandre B. Tsybakov. Aggregation by exponential weighting, sharp pac-bayesian bounds and sparsity. *Machine Learning*, 72(1):39–61, 2008. 59

Hai-Dang Dau and Nicolas Chopin. Waste-Free Sequential Monte Carlo. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):114–148, 11 2021. 87

Marie Devaine, Pierre Gaillard, Yannig Goude, and Gilles Stoltz. Forecasting electricity consumption by aggregating specialized experts: A review of the sequential aggregation of specialized experts, with an application to slovakian and french country-wide one-day-ahead (half-) hourly predictions. *Machine Learning*, 90:231–260, 2013. 50, 62

Jérémy Diez, Elsa Lanaud, and Matthias de Saint Denis. Acceptabilité du véhicule automatisé : Revue bibliographique des travaux 2020-2021, 2021. Direction Générale des Infrastructures, des Transports et de la Mer (DGITM), Service de l'administration générale et de la stratégie (SAGS) and Etudes et prospectives (EP). 105

Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009. 95

Christopher C Drovandi, Anthony N Pettitt, and Anthony Lee. Bayesian indirect inference using a parametric auxiliary model. *Statistical Science*, 30(1), 2015. 86

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of

language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022. 62

Darrell Duffie. *Dynamic asset pricing theory*. Princeton University Press, 2010. 59

Jane Elith and John R Leathwick. Species distribution models: ecological explanation and prediction across space and time. *Annual review of ecology, evolution, and systematics*, 40:677–697, 2009. 59

Peter Exterkate, Patrick JF Groenen, Christiaan Heij, and Dick van Dijk. Nonlinear forecasting with many predictors using kernel ridge regression. *International Journal of Forecasting*, 32(3):736–753, 2016. 37

Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis. nonparametric discrimination; consistency properties. *USAF School of Aviation Medicine*, 1951. 37

Eric B Ford, Althea V Moorhead, and Dimitri Veras. A bayesian surrogate model for rapid time series analysis and application to exoplanet observations. *Bayesian Analysis*, 6(3): 475–499, 2011. 28, 33, 118

David T Frazier, David J Nott, Christopher C Drovandi, and Robert Kohn. Bayesian inference using synthetic likelihood: Asymptotics and adjustments. *Journal of the American Statistical Association*, pages 1–12, 2022. 85

Xavier Gabaix. Power laws in economics and finance. *Annu. Rev. Econ.*, 1(1):255–294, 2009. 59

Robin Genuer and Jean-Michel Poggi. Arbres CART et Forêts aléatoires, Importance et sélection de variables. hal-01387654v2, January 2017. 37

Loïc Giraldi, Olivier P Le Maître, Kyle T Mandli, Clint N Dawson, Ibrahim Hoteit, and Omar M Knio. Bayesian inference of earthquake parameters from buoy data using a polynomial chaos-based surrogate. *Comput Geosci 21*, pages 683–699, 2017. 85

Christophe Giraud. *Introduction to high-dimensional statistics*. Chapman and Hall/CRC, 2021. 62

Philipp Glaser, Michael Schick, Kosmas Petridis, and Vincent Heuveline. Comparison between a polynomial chaos surrogate model and markov chain monte carlo for inverse uncertainty quantification based on an electric drive test bench. *ECCOMAS Congress*, pages 8809–8826, 2016. 40

Steven Golovkine. `FDApy`: a python package for functional data. *arXiv preprint arXiv:2101.11003*, 2021. 43

Tomasz Górecki and Maciej Łuczak. Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, 26:310–331, 2013. 22

Yannig Goude. Modélisation de séries stationnaires. MAP-STA2 : Séries chronologiques, 2024a. URL https://www.imo.universite-paris-saclay.fr/~yannig.goude/Materials/time_series/cours4_proc_stationnaires.pdf. 125

Yannig Goude. Introduction aux séries temporelles, tendance et composante saisonnière. MAP-STA2 : Séries chronologiques, 2024b. URL https://www.imo.universite-paris-saclay.fr/~yannig.goude/Materials/time_series/cours2_tendance_composante_saisonniere.pdf. 127

Sonja Greven and Fabian Scheipl. A general framework for functional regression modelling. *Statistical Modelling*, 17(1-2):1–35, 2017. 62

James D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994. 60

James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957. 50

Jude C Hays. Time series analysis, 2021. 128

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 46

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. 41

Rongchao Jiang, Zhenchao Jin, Dawei Liu, and Dengfeng Wang. Multi-objective lightweight optimization of parameterized suspension components based on nsga-ii algorithm coupling with surrogate model. *Machines*, 9(6), 2021. 28, 33, 118

Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002. 41

Anatoli Juditsky and Arkadii Nemirovski. Functional aggregation for nonparametric regression. *The Annals of Statistics*, 28(3):681–712, 2000. 62

Ioannis Kalogridis and Stefan Van Aelst. Robust functional regression based on principal components. *Journal of Multivariate Analysis*, 173:393–415, 2019. 62

Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016. 11, 105

Matt J Keeling and Ken TD Eames. Networks and epidemic models. *Journal of the royal society interface*, 2(4):295–307, 2005. 59

Jonghyuk Kim, Hyunwoo Hwangbo, and Soyean Kim. An empirical study on real-time data analytics for connected cars: Sensor-based applications for smart cars. *International Journal of Distributed Sensor Networks*, 14(1):1550147718755290, 2018. 11, 105

Irena Koprinska, Dengsong Wu, and Zheng Wang. Convolutional neural networks for energy time series forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018. 48

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 48

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *Preprint arXiv:1212.2002*, 2012. 29, 70, 119

Paula Lakomicki. *Démarche incrémentale pour qualifier la fiabilité du système de perception et de décision du véhicule autonome*. Thesis, Université de Technologie de Troyes, 2018. 11, 105

Vincent Le Guen and Nicolas Thome. Shape and time distortion loss for training deep time series forecasting models. *Advances in Neural Information Processing Systems*, 32: 4191–4203, 2019. 22, 25

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 48

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. 62

Jiangqi Long, Yaoqing Liao, and Ping Yu. Multi-response weighted adaptive sampling approach based on hybrid surrogate model. *IEEE Access*, 9:45441–45453, 2021. 28, 33, 118

Kanti V Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530, 1970. 22

Stefano Marelli and Bruno Sudret. Uqlab user manual - polynomial chaos expansions. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2021. Report # UQLab-V1.4-104. 40

Steven A Mattis and Barbara Wohlmuth. Goal-oriented adaptive surrogate construction for stochastic inversion. *Computer Methods in Applied Mechanics and Engineering*, 339: 36–60, 2018. 28, 33, 118

Christoph Molnar. Interpretable machine learning. A Guide for Making Black Box Models Explainable, 2022. URL https://christophm.github.io/interpretable-ml-book. 131

Jaouad Mourtada. Prédiction séquentielle par agrégation d'experts. *Master dissertation*, 2016. 50

Marc Nabhan. *Models and algorithms for the exploration of the space of scenarios: toward the validation of the autonomous vehicle*. Thesis, Université Paris-Saclay, 2020. 12, 106

Arkadi Nemirovski. Topics in non-parametric statistics. *Lectures on Probability Theory and Statistics: Ecole d'Ete de Probabilites de Saint-Flour XXVIII-1998*, 28:85–277, 2000. 62

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572, 1901. 41

Roger D. Peng. Advanced Statistical Computing. https://bookdown.org/rdpeng/advstatcomp/rejection-sampling.html, 2022. 90

Michael E Peskin. *An introduction to quantum field theory*. CRC press, 2018. 59

Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009. 37

Leah F Price, Christopher C Drovandi, Anthony Lee, and David J Nott. Bayesian synthetic likelihood. *Journal of Computational and Graphical Statistics*, 27(1):1–11, 2018. 85, 86

Leslie A Real and Roman Biek. Spatial dynamics and genetics of infectious diseases on heterogeneous landscapes. *Journal of the Royal Society Interface*, 4(16):935–948, 2007. 59

Philippe Rigollet. 18.657: Mathematics of Machine Learning, October 2015. 29, 69, 74, 119

Marie-Hélène Roy and Denis Larocque. Robustness of random forests for regression. *Journal of Nonparametric Statistics*, 24(4):993–1006, 2012. 37

Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. *Proceedings 15th International Conference on Machine Learning*, 1998. 37

Ronald W Schafer. What is a savitzky-golay filter?[lecture notes]. *IEEE Signal processing magazine*, 28(4):111–117, 2011. 127

Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008. 22

Han Lin Shang. A survey of functional principal component analysis. *AStA Advances in Statistical Analysis*, pages 121–142, 2014. 42

Lloyd S Shapley and Martin Shubik. A method for evaluating the distribution of power in a committee system. *American political science review*, 48(3):787–792, 1954. 133

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 62

Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 2007. 88, 90

Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC Press, 2018. 88

Ihab Sraj, Kyle T Mandli, Omar M Knio, Clint N Dawson, and Ibrahim Hoteit. Quantifying uncertainties in fault slip distribution during the tōhoku tsunami using polynomial chaos. *Ocean Dynamics*, 67, 2016. 28, 33, 118

Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008. 63

Gilles Stoltz. Agrégation séquentielle de prédicteurs: méthodologie générale et applications à la prévision de la qualité de l'air et à celle de la consommation électrique. *Journal de la Société française de Statistique*, 151(2):66–106, 2010. 62

Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005. 85

Abdullah Erdal Tümer and Aytekin Akkuş. Forecasting gross domestic product per capita using artificial neural networks with non-economical parameters. *Physica A: Statistical Mechanics and its Applications*, 512:468–473, 2018. 59

Vladimir G Vovk. A game of prediction with expert advice. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 51–60, 1995. 50

Rui Wang and Yuesheng Xu. Functional reproducing kernel hilbert spaces for non-point-evaluation functional data. *Applied and Computational Harmonic Analysis*, 46(3):569–623, 2019. 63

Weijie Wang and Yanmin Lu. Analysis of the mean absolute error (mae) and the root mean square error (rmse) in assessing rounding model. *IOP Conference Series: Materials Science and Engineering*, 324:012049, 2018. 22

Max Welling. Kernel ridge regression. *Max Welling's class notes in machine learning*, pages 1–3, 2013. 38

Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82, 2005. 22

Johannes Wohlenberg. Functional principal component analysis and functional data. *Toward Datascience*, June 2021. 42

Stephen J Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015. 95

Zhihao Xu, Chunxue Yu, Hao Sun, and Zhifeng Yang. The response of sediment phosphorus retention and release to reservoir operations: Numerical simulation and surrogate model development. *Journal of Cleaner Production*, 271:122688, 2020. 28, 33, 118

Ming Yuan and T Tony Cai. A reproducing kernel hilbert space approach to functional linear regression. *The Annals of Statistics*, 38(6):3412–3444, 2010. 63

Zhi-Hua Zhou and Ji Feng. Deep Forest: towards an alternative to deep neural networks. In *IJCAI*, pages 3553–3559, 2017. 45

Zhi-Hua Zhou and Ji Feng. Deep forest. *National science review*, 6(1):74–86, 2019. 45

Adam Ziebinski, Rafal Cupek, Damian Grzechca, and Lukas Chruszczyk. Review of advanced driver assistance systems (adas). *AIP Conference Proceedings*, 1906(1), 2017. 10, 104

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. 43

Maciej Łuczak. Hierarchical clustering of time series data with parametric derivative dynamic time warping. *Expert Systems with Applications*, 62:116–130, 2016. ISSN 0957-4174. 22

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre: Calibration de modèles de systèmes complexes pour la construction du jumeau numérique du véhicule autonome

Mots clés: Inférence bayésienne, apprentissage statistique, modèle complexe

Résumé:
La validation des véhicules autonomes et des aides à la conduite ne peut plus reposer exclusivement sur des essais réels, en raison de la diversité des cas d'usage, des systèmes impliqués et des niveaux de fiabilité requis. La simulation numérique émerge comme une solution prometteuse pour compléter ces tests. Toutefois, pour que la simulation soit utilisable dans un cadre de certification, il est crucial de démontrer sa fiabilité et d'établir une corrélation suffisante avec les résultats obtenus sur piste. Cette thèse se concentre sur les problèmes dits inverses : à partir d'observations réelles du système, il faut déduire les paramètres d'entrée qui ont généré ces observations. L'objectif est d'identifier les paramètres qui permettront aux simulations de reproduire au mieux les observations réelles lors des essais.

Cette étape de calibration de modèle, ou inférence des paramètres, requiert généralement de nombreuses simulations, ce qui impose des coûts computationnels importants. Pour surmonter cette contrainte, cette thèse propose l'élaboration d'un modèle de substitution qui imite le comportement de la simulation originale tout en réduisant significativement les coûts computationnels. Ce modèle revient à résoudre ce que l'on appelle le problème direct, qui consiste à prédire les résultats de sortie pour des entrées spécifiques.

L'objectif global de cette thèse est de développer diverses méthodologies statistiques pour adresser ces deux problèmes, direct et inverse. Il est particulièrement crucial de concevoir des approches qui s'ajustent automatiquement à chaque contexte, afin de faciliter la généralisation de ces méthodes à un large éventail de situations.

Title: Calibration of complex system models for the construction of the digital twin of the autonomous vehicle

Keywords: Bayesian inference, statistical learning, complex model

Abstract:
The validation of autonomous vehicles and driving aids can no longer rely exclusively on real-life tests due to the diversity of use cases, systems involved, and reliability levels required. Digital simulation is emerging as a promising solution to complement these tests. However, for simulation to be usable in a certification framework, it is crucial to demonstrate its reliability and establish sufficient correlation with on-track results. This thesis addresses the issue of inverse problems, which involve deducing input parameters from real system observations. The objective is to identify parameters enabling simulations to reproduce real observations during testing.

The calibration step, or parameter inference, generally requires numerous simulations, which imposes significant computational costs. This thesis proposes developing a surrogate model that mimics the behavior of the original simulation while significantly reducing computational costs. This model represents a solution to the so-called direct problem, which involves predicting output results for specific inputs.

The overarching objective of this thesis is to develop a range of statistical methodologies to address these two problems, direct and inverse. It is particularly important to design approaches that can be automatically adapted to different contexts to facilitate the generalization of these methods to a wide range of situations.

Institut Polytechnique de Paris
91120 Palaiseau, France