Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

Dynamic Safe Active Learning with NARX Gaussian Processes

Veronica Crespi

Course of Study:	Computer Science
Examiner:	Prof. Dr. rer. nat. Marc Toussaint
Supervisor:	Dr. -Ing. Duy Nguyen-Tuong
Commenced:	15. April 2019
Completed:	15. October 2019

Abstract

Black-box modelling using Gaussian Processes has been widely and successfully studied and applied to model complex dynamic systems. So far, however, very little attention has been paid to the processes of obtaining the necessary data to train such systems in an efficient and safe manner. Zimmer et al. [ZMN18] proposed a Safe Active Learning framework for Time-Series Modeling with Gaussian Processes, which can be used to learn a Nonlinear Exogenous (NX) representation of a dynamic system in an efficient manner while considering safety constraints. In this masters' thesis, the problem of efficiently and safely learning a Nonlinear Autoregressive Exogenous (NARX) representation of a dynamic system is addressed. With this purpose, an extension of the framework by Zimmer et al. was designed and implemented. Finally, the developed framework was evaluated in a real-world application. The results show an improvement on the original framework performance, as well as the suitability of the approach for real-world dynamic system modelling.

Acknowledgments

This work would not have been possible without the financial support of the Bosch Center for Artificial Intelligence.

I would like first to thank my advisors Prof. Dr. rer. nat. Marc Toussaint and Dr. -Ing. Duy Nguyen-Tuong for offering me this thesis and for their guidance and support.

I am profoundly grateful to my tutor, Christoph Zimmer, for his mentoring and support throughout the semester.

Special thanks also go to Mark Schillinger for providing me with the needed analytical models to run my experiments and being always available for questions and requests.

I would also like to thank my colleagues from the Bosch Center for Artificial Intelligence for the interesting discussions and feedback about my work.

Contents

List of Figures

List of Algorithms

List of Abbreviations

GP Gaussian Process. 9

ML Machine Learning. 21

NARX Nonlinear Autoregressive Exogenous. 3

NRMSE Normalized Root Mean Square Error. 20

NX Nonlinear Exogenous. 3

RMSE Root Mean Square Error. 9

1 Introduction

Black-box dynamic Modelling plays an important role in addressing the issue of interacting, controlling and optimizing the functioning of physical systems that cannot be accurately and fully represented by a white-box model. Recently, researchers have shown an increased interest in modelling these systems using supervised Machine Learning techniques. In particular, Gaussian Processes have been successfully applied in this area [CDD+17; KMRG04; NSP09; PKPB07; RAO+15; RD16]. However, obtaining an informative data set is of particular concern in this scenario, since annotations and measurements of the physical system are usually expensive. Furthermore, safety considerations need to be taken when running these measurements in order not to damage the system.

Safe Active Learning deals exactly with the above mentioned problem of generating a small, informative and safe data set with which to train the desired Machine Learning algorithm without incurring in high measurement costs or compromising the systems' integrity. A Safe Active Learning exploration is characterized by its iterative choosing the best points to be tested in the system, performing the measurements in the system, adding the obtained information to the training data set and retraining the Machine Learning algorithm with the extended data set. Consequently, the key component of such a framework is the acquisition function used to choose the next exploration iteration. Previous research [SNE+15; ZMN18] have used the entropy criterion subject to some safety constraints to drive the exploration.

Zimmer et al. [ZMN18] addressed the Safe Active Learning with Gaussian Processes problem in the context of Time Series Modeling. In their work, a Safe Active Learning Framework that plans parameterizable trajectories to test on the system at each iteration was developed. Much of their research focused on identifying and evaluating the chosen acquisition function and safety guarantees. Conversely, the study did not focus on the choice of the learnt model structure. As a result, a Nonlinear Exogenous model was used, which is usually suboptimal when modelling stateful dynamic systems. Nonlinear Exogenous models consider the output of the system to be dependent only on the inputs, disregarding the dependency with previous outputs.

The purpose of this masters' thesis is to extend the framework developed by Zimmer et al. to consider Nonlinear Autoregressive Exogenous models. Nonlinear Autoregressive

Exogenous models differ from Nonlinear Exogenous in that they also consider the dependency of the system's output on previous outputs and is, therefore, believed to better explain real dynamic systems. This work also compares the different methods that can be used in order to make this modification in the model's structure possible.

The main contributions of this thesis can be summarized as:

- This work extends the Safe Active Learning with Gaussian Processes for Time Series Modelling framework developed by Zimmer et al. [ZMN18] by incorporating Nonlinear Autoregressive Exogenous Gaussian Processes to learn the model.
- This work assesses the impact of modelling dynamic systems with different model structures (Nonlinear Autoregressive Exogenous and Nonlinear Exogenous ones).
- This work considers the implications of different approximations of multi steps ahead predictions in the context of a Safe Active Learning framework. Approximation by mean value, Moment Matching and Monte Carlo Sampling are considered.
- This work empirically evaluates the proposed framework and compares it with the work by Zimmer et al. [ZMN18].

Outline

The remainder of this thesis is structured as follows:

Chapter 2 – Literature Review: provides an overview of the related work relevant for this thesis.

Chapter 3 – Theoretical Background: addresses the main theoretical topics related to this work, such as supervised Machine Learning and Active Learning foundations.

Chapter 4 – Methods: is concerned with the concrete modifications made to the work by Zimmer et al. [ZMN18] and its implications.

Chapter 5 – Evaluation: Here, the experiments on a real world problem, together with their results are presented.

Chapter 6 – Discussion, Conclusions and Outlook: summarizes this work and provides an overview of its applications and open challenges.

2 Literature Review

2.1 Dynamic Modelling

Many physical systems are so complex or uncertain that they cannot be accurately and fully represented by a mathematical model. However, we still need to be able to define these systems' behaviour in a way that we can create them and interact with them. For this reason, there has been an increasing interest in Intelligent Systems that can help design, optimize and control such physical systems without the need for a mathematical model [SX17].

Many efforts have been made in the robotics community to find model learning methods that serve for control purposes. A detailed survey of the different methods and algorithms used for model learning in this field can be found in the work of Nguyen-Tuong & Peters [NP11]. Some of the most popular algorithms in this area are: Reinforcement Learning [ACQN07; AMS97; NCD+06], Neuronal Networks [ÅT06; CYD+06; GH02; PCK02] and Gaussian Processes [KMRG04; NSP09; PKPB07]. In the case of Reinforcement Learning [SB18], the agent learns the model as it interacts with the environment and can, therefore, decide at each iteration what to explore next. On the contrary, in the case of plain Neural Networks or Gaussian Processes, like the ones used in the above mentioned works, the Machine Learning algorithm is trained only after the complete set of training data is collected. As a result, the training set is redundant and sub-optimal. Variations of these algorithms can be made to guide the data collection and make the training set smaller and less redundant. This approach is called Active Learning and is discussed in the following section. In this thesis, an active learning framework is used in order to perform a more efficient data collection.

Using Gaussian Processes to model Time Series of Dynamic Systems [ROE+13] has also been a topic of interest lately. This allows us to model a wide variety of systems [CDD+17; RAO+15; RD16], where the time and order of the inputs play an important role. In their work, Girard et al. [GRCM03] addressed the problem of multi-steps ahead Time Series forecasting with GPs, which is of high interest for this thesis. Particularly, they study multi-steps predictions in NARX models, where the output of the model is used as part of the input for the next time steps. Because of this model structure, an approximation is needed to be able to propagate the uncertainty in the inputs, which is caused by the feedback loop. In their work, they proposed to use moment matching, which was proven to perform comparably to Monte Carlo approximations and to outperform the naive approach. For this reason, their approach will be considered in this master's thesis when computing multiple steps ahead predictions.

2.2 Active Learning

The goal of Active Learning is to optimize the choice of points for the training set, so that the time and cost of unnecessary measurements performed in the real physical system can be spared. For the purpose of this master's thesis, this section focuses only on Active Learning using GPs, but analogous literature can be found for other Machine Learning techniques [Set09].

Seo et al. [SWGO00] addressed this topic and presented two different metrics to guide the exploration, both based on the optimal experiment design [FH12]. The first metric, proposed by Cohn [Coh94] aimed to minimize the generalization error, while the second one, presented by MacKay [Mac92], aimed to maximize the information gain, which translates into choosing the point with the highest the expected variance. As expected, both metrics performed better than random exploration. In their work, however, they considered the GP parameters to be known and could, therefore, use an a priori design of the experiment (choose the complete training set before performing the measurement in the physical system).

Krause et al. [KG07] considered the problem when the GP parameters are unknown and a sequential approach is needed. In this case, they considered entropy and mutual information as metrics and chose each next point to add to the training set, taking into account the measurements for the previous ones. This approach was then compared to the a priori one and proved to work better in the case of unknown GP parameters.

Although these Active Learning approaches address the problem of efficient exploration, they do not consider the risk of exploring without safety considerations, which is crucial in dynamic modelling. Disregarding safety in this context can lead to a breakdown and in-utilization of the physical system and can even compromise the safety of the people operating it. Therefore, the main focus of this thesis is to find a way of performing an efficient exploration, while guarding the system's safety throughout the process.

2.3 Safe Active Learning

Safe Exploration has been studied in depth in the context of Reinforcement Learning and is still an ongoing field of research. Geibel et al. [GW05], for example, defined the risk as the probability of ending in some pre-defined unsafe states and then tried to minimize this risk. Constraints on the probability of constraints violations were also considered [Gei06]. Learning by demonstration has also been explored by different authors [ACN10; PR11]. Garcia & Fernandez [GF12] used a predefined safe but suboptimal policy as a starting point for the learning process and then improved it using safe exploration using a risk function. Gillulay & Tomlin [GT11] used a reachability analysis to obtain safety guarantees. For a more detailed overview and comparison of these methods, please refer to this survey by Garcia & Fernandez [GF15].

Safe Exploration in the context of Bayesian Optimization and Active Learning with GPs has also been considered. However, it has not yet been so extensively explored.

Srinivas et al. [SKKS09; SKKS12] studied the case of GP optimization for the bandit setting with bounded regrets. Building upon their work, Sui et al. developed SafeOpt[SGBK15] and StageOpt [SZBY18] algorithms for Safe Exploration for Optimization with GPs in static bandit-like environments. In their algorithms, they used the prediction uncertainty to guide exploration and confidence bounds to predict the safety of unexplored areas. The output of the system was modelled with a GP and the safety constraint could only be defined as a lower bound on the system's output, which makes this algorithm unsuitable for more complex problems, like the one addressed by this thesis. In a later work by Berkenkamp et al. and by Kirschner et al., modified versions of the SafeOpt algorithm were successfully applied to optimize the controller's parameters of a quadrotor vehicle [BKS16; BSK16], and the beam intensity of the Swiss Free Electron Laser [KMH+19], avoiding expensive and dangerous system failures.

Berkenkamp et al. [BMSK16] explore and lay the theoretical foundations for the Safe Learning of Regions of Attraction with GPs using a first approximation of the model and a corresponding Lyapunov function.

2.3.1 Safe Active Learning with Gaussian Processes (GPs)

Schreiter et al. [SNE+15] developed a framework for Safe Exploration for Active Learning with GPs. This algorithm consisted of training a GP Classifier to distinguish safe and unsafe areas and using the differential entropy criterion [KG07] to guide the exploration. In their work, they include a theoretical analysis on the safety of the exploration, providing an upper bound to the probability of exploring unsafe areas. Additionally, they tested their framework with the inverse pendulum hold up problem.

Contrary to the above presented research, this approach allowed the system to learn domain-specific safety regions that are more complex than just a threshold on the system's output. However, their approach was still a stationary one, unsuited for the modelling of dynamic systems.

Building upon Schreiter et al.'s work [SNE+15], Zimmer et al. [ZMN18] addressed the Safe Active Learning with GPs problem in the context of Time Series Modeling. In their proposed algorithm, two GPs were trained to learn simultaneously the system model and a system safety metric function (a function that takes positive values inside safe areas and negative ones everywhere else and can, therefore, be used to distinguish between this areas). Both GPs had a NX structure, taking as input a fixed number of previous inputs to the physical system. Using the aforementioned GPs, a Safe Active Learning algorithm was introduced, corresponding to the following constraint optimization problem:

$$x^* = \arg\max_x I(x),\ st\ P(g(x) \geq 0) \geq 1 - \alpha \tag{2.1}$$

Here, $I(x)$ corresponds to the optimality criterion that drove the exploration, while $g(x)$ represents the learnt safety metric function. Then, the constraint of this problem forced the x^* to be safe with a probability higher than $1 - \alpha$, being α the customizable risk probability tolerance.

Finally, Zimmer et al. tried the proposed algorithm in the learning of a surrogate model of a high-pressure fluid system, which will be taken as a comparative baseline for this master's thesis. It is important to mention, though, that Zimmer et al. used an NX analytical model of the real system as ground truth, so the real RMSE and safety accuracy that is obtained when compared with the real physical system is not reported in the paper.

In this thesis, a more realistic approach is taken, using a NARX analytical model of the real system as ground truth. Such a structure for the model is believed to be more accurate as it incorporates the previous outputs of the system in the current input and can, therefore, model the system state, while in the NX case, the system state is discarded, as the outputs are only considered to be a function of the actual and previous inputs. Particularly, in the case of study, the analytical NX model of the system reported higher Normalized Root Mean Square Errors (NRMSEs) (22.2%, 15.6% and 10.2%) than the NARX one (8.3%, 13.4% and 8.4%), when compared with the real physical system for 3 different data sets. Additionally, given the aforementioned arguments in favour of NARX models, a NARX structure will be used for the learnt GPs in this thesis. As a result, better modelling of the system, measured by lower RMSE, is expected.

3 Theoretical Background

3.1 Supervised Machine Learning

In Supervised Machine Learning, a Machine Learning (ML) algorithm is trained on a predefined set of labelled data $D = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{i=n}$ to obtain a representation or model: $f : x \mapsto y$, that explains the data and can predict new unknown points. Consequently, for the algorithm to learn a meaningful model, a good amount of representative labelled data is needed. The main drawback of this approach is that obtaining the necessary labelled data can be very expensive and time consuming depending on the scenario. For this reason, it is important to find a way of obtaining a representative and small training set that allows us to train the model with a low number of representative labelled data. Active Learning proposes a way to tackle this issue.

3.2 Gaussian Processes

3.2.1 Problem Statement

Gaussian processes are stochastic processes used to model systems characterized by Equation (3.1). Where $f(x)$ denotes the scalar output of the system (also called target) and y the observed output, which differs from $f(x)$ by some additive noise (ϵ), which is normally distributed with mean 0 and variance σ.

$$y = f(x) + \epsilon,\ \epsilon \sim \mathcal{N}(0,\, \sigma^2) \tag{3.1}$$

There are two different ways of understanding Gaussian Processes: the weight-space view and the function view. In the weight-space view, the GP is considered simply as a linear regression with Gaussian noise. This can be represented by the following equations:

$$\begin{aligned} f(x) &= w^T \phi(x) \\ y &= f(x) + \epsilon,\ \epsilon \sim \mathcal{N}(0,\, \sigma^2) \end{aligned} \tag{3.2}$$

Where $f(x)$ is constructed as a linear combination of some predefined features $\phi(x)$ of the input x.

In the function-space view, the GP describes a distribution over functions:

$$f(x) \sim GP\Big(m(x),\, k(x,\, x')\Big) \tag{3.3}$$

Defined by its mean and covariance functions:

$$\begin{aligned} m(x) &= \mathbb{E}\Big[f(x)\Big] \\ k(x,\, x') &= \mathbb{E}\Big[\Big(f(x) - m(x)\Big)\Big(f(x') - m(x')\Big)\Big] \end{aligned} \tag{3.4}$$

From here on, only the function-space view will be considered to explain the learning and predicting phases. For an equivalent explanation for the weight-space view, please refer to [Ras03].

3.2.2 Training

A GP is a non-parametric Supervised Machine Learning Method. This means, that:

- A GP is trained using a training set $D = \Big\{\Big(x_i, y_i\Big)\Big\}_{i=1}^{i=n}$, where each pair (x_i, y_i) is obtained from the model from Equation (3.1).
- There are no predefined parameters to be learnt. Instead, the whole training set is used in the mapping function.

As seen in Equation (3.3) and Equation (3.6), a GP is characterized by 2 functions: $m(x)$ and $k(x,\, x')$. Commonly a constant mean is used (frequently $m(x) = 0$) and the covariance matrix is set to be the Gaussian Squared Exponential (also called Radial Basis) Function, shown in Equation (3.5).

$$k_{RBF}(x_i,\, x_j) = e^{-\frac{1}{2}(x_i - x_j)^T \theta^{-1} (x_i - x_j)} \tag{3.5}$$

Therefore, training such a GP implies computing the mean and covariance matrix of the training set as follows:

$$\begin{aligned} m(x) &= \frac{1}{n}\sum_{i=1}^{n} y_i \\ K_{i,j} = k_{RBF}(x_i,\, x_j) &= e^{-\frac{1}{2}(x_i - x_j)^T \theta^{-1}(x_i - x_j)} \end{aligned} \tag{3.6}$$

As a result, the probability of the observed targets given the inputs can be expressed as in Equation (3.7), with a covariance matrix that also takes the additive noise of the observations into account.

$$P(Y|X) = \mathcal{N}(m(x),\, K + I\sigma^2) \tag{3.7}$$

3.2.3 Predictions

Considering a zero mean GP, its joint probability over the observed noisy outputs and predicted targets is given by:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + I\sigma^2 & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{3.8}$$

The GP prediction of the targets f_* conditioned by the training set and the given inputs X_* of the system can be derived from Equation (3.8) and results in Equation (3.9) (Refer to [Ras03] for the complete derivation).

$$\begin{aligned} f_*|X, y, X_* &\sim \mathcal{N}(\overline{f_*},\, cov(f_*)) \\ \overline{f_*} &= \mathbb{E}\Big[f_*|X, y, X_*\Big] = K(X_*,X)\Big[K(X,X) + I\sigma^2\Big]^{-1} y \\ cov(f_*) &= K(X_*,X_*) - K(X_*,X)\Big[K(X,X) + I\sigma^2\Big]^{-1} K(X,X_*) \end{aligned} \tag{3.9}$$

3.2.4 Hyperparameter tuning

Choosing different parameters for a given covariance function or choosing completely different covariance functions results in different GPs. The covariance function and its parameters are the hyperparameters of the GP.
In the above mentioned case, the lengthscale (θ) is the hyperparameter of the GP with covariance function k_{RBF} (Equation (3.5)). When tuning the hyperparameters, different hyperparameters are considered in order to find the most suitable ones for a given

problem. This can be done using different methods, such as Grid and Random Search, and Bayesian Optimization.

3.3 Active Learning

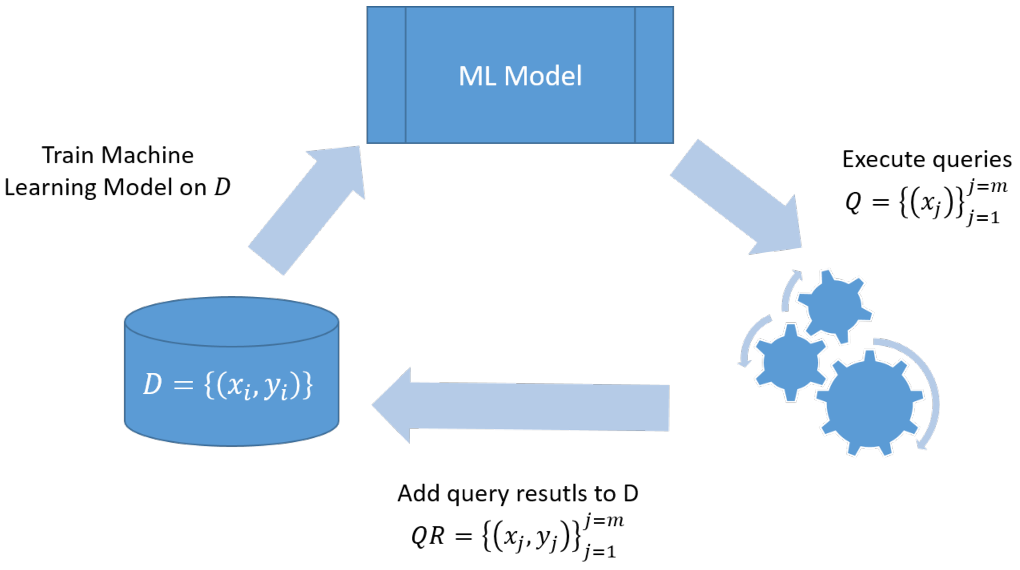

ML Model
Train Machine Learning Model on D
Execute queries $Q = \{(x_j)\}_{j=1}^{j=m}$
$D = \{(x_i, y_i)\}$
Add query resutls to D $QR = \{(x_j, y_j)\}_{j=1}^{j=m}$

Figure 3.1: Active Learning System

Active Learning is a subfield of Supervised Machine Learning, where the ML algorithm decides on which data it will be trained. The typical configuration of an Active Learning system is depicted in Figure 3.1. Here the ML algorithm is first trained with a small initial set of labelled data $D = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{i=n}$. Subsequently, the algorithm defines the queries: $Q = \left\{\left(x_i\right)\right\}_{i=1}^{i=m}$. These queries correspond to the data points that the algorithm considers will help most to improve the actual model. After the query (or set of queries) is executed and a new labelled data point (or set of them) is obtained, the algorithm recomputes the learned representation based on the complete data set $D^* = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{i=n} \cup \left\{\left(x_j, y_j\right)\right\}_{j=1}^{j=m}$ and decides on the next queries to execute. This process goes on until a certain goal is reached (e.g. error or convergence criterion) or for a fixed number of iterations.

The core of every Active Learning system is the criterion used to select the next most informative query to be executed. After this criterion (I) is defined, the maximization problem can be stated as in Equation (3.10). Commonly, the most informative query is chosen as the one corresponding to the point with the highest entropy, which is the point with the most uncertain prediction. Consequently, (I) can be defined as in Equation (3.11) or Equation (3.12).

$$Q = \arg\max_x I(x) \tag{3.10}$$

$$I(\Sigma(x)) = \det(\Sigma(x)) \tag{3.11}$$

$$I(\Sigma(x)) = trace(\Sigma(x)) \tag{3.12}$$

A comparison of the Active Learning approach with randomly sampling the input space can be seen in Figure 3.2. Here both algorithms are initialized with 2 labelled data points, a Gaussian Process is used to make the predictions, and the criterion from Equation (3.11) is used to define the next query in the Active Learning system. In this example, one can clearly see that the Active Learning system obtains far better predictions than random sampling in every iteration.

3.3.1 Safe Active Learning

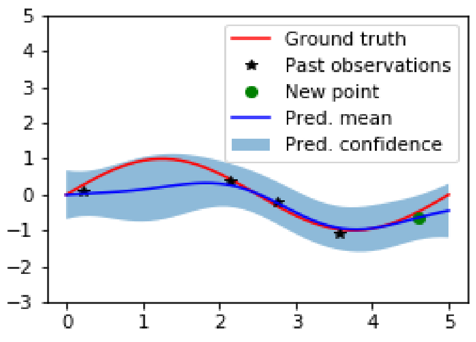
In theory, Active Learning can improve any Machine Learning training to make it more query efficient (i.e. obtain more information with fewer queries), but in practice, this alone is a rather naive approach as we often need care for the safety of the system in which we run the queries.

Safety considerations play a key role when modelling physical systems with a ML algorithm. In this scenario, one needs to make sure that the physical system will not be broken, in order to be able to continue executing queries. To this end, a set of safety constraints are predefined (or learnt) and used to restrict the set of queries that the Active Learning system can select.

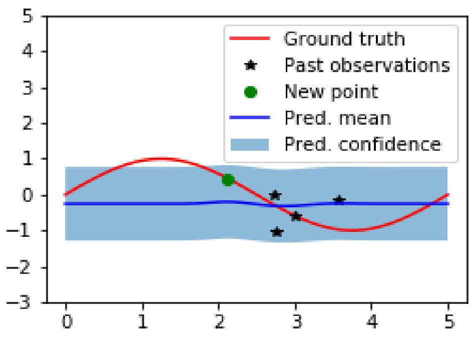
The Safe Active Learning optimization problem results in the constrained version of Equation (3.10), that is:

$$\begin{aligned} &Q = \arg\max_x I(x) \\ &st\; g_i(x) \geq 0 \; for \; i = 1, ..., n \end{aligned} \tag{3.13}$$

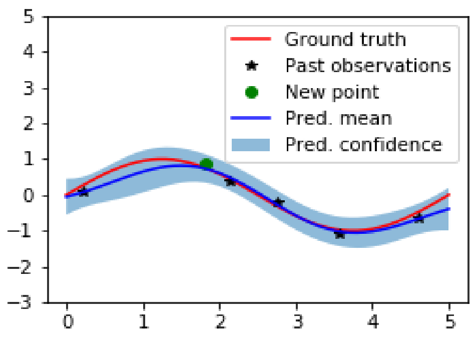
Where the n safety constraints are modeled by some functions g_i, which are greater or equals to zero when x is safe and negative otherwise.



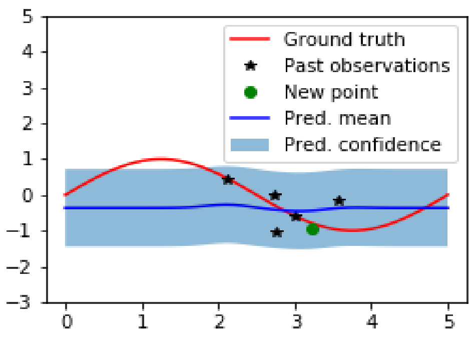
(a) Active Learning, iteration 3



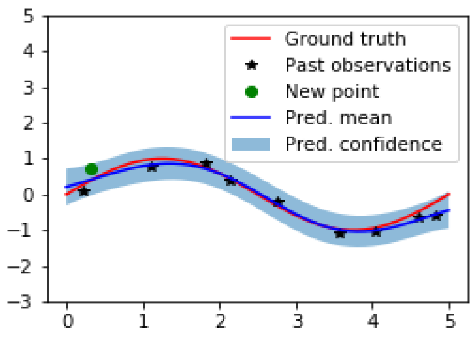
(b) Random sampling, iteration 3



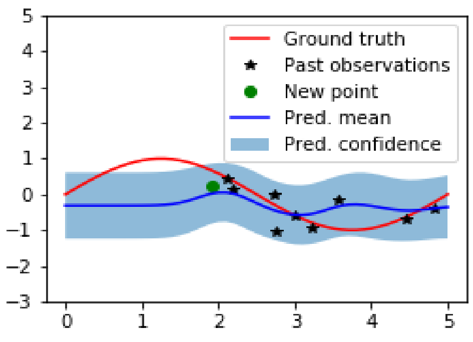
(c) Active Learning, iteration 4



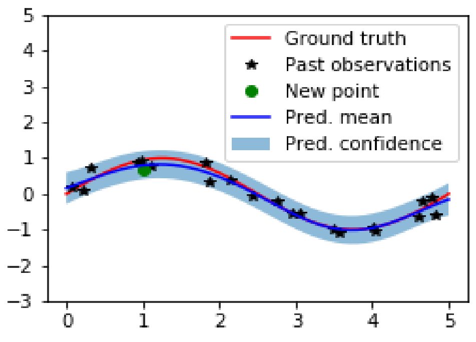
(d) Random sampling, iteration 4



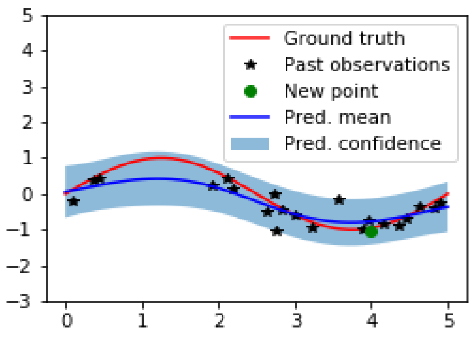
(e) Active Learning, iteration 8



(f) Random sampling, iteration 8



(g) Active Learning, iteration 20



(h) Random sampling, iteration 20

Figure 3.2: Active Learning vs random sampling strategy

3.4 Time Series

A Time Serie refers to a sequence of discrete time data points. For example, measurements of temperature carried out every hour for 2 days, or a person's weight recorded every month for his or her entire life. When analysing and predicting Time Series, different models can be used. The two models relevant to this work are explained below.

3.4.1 Nonlinear Exogenous Model

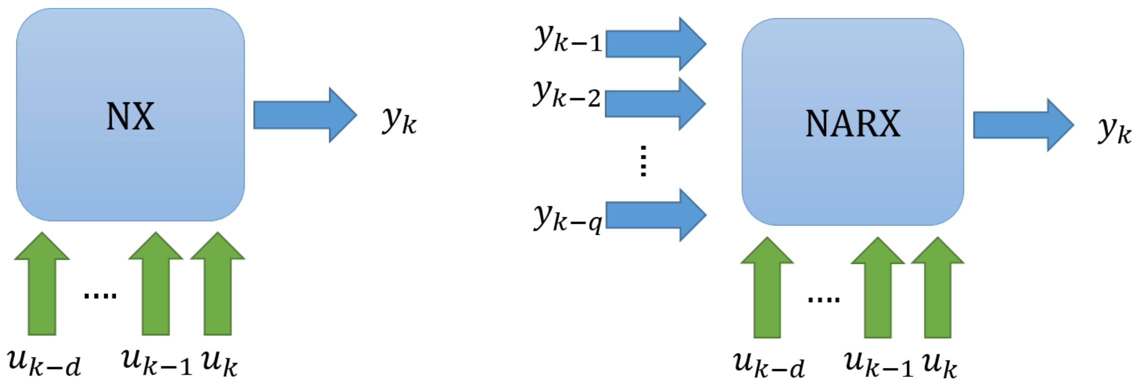
An NX system is one where the output depends on a nonlinear manner on the last d exogenous inputs, as well as on the current one. As illustrated in Figure 3.3a, the input x_k of the system at time step k is composed by the exogenous variables starting form the time step $k-d$ up to the current time step $(u_{k-d}, ..., u_{k-1}, u_k)$, which relate in a nonlinear way to the output y_k.

$$\begin{aligned} u_k &= (u_{k,1}, u_{k,2}, ..., u_{k,l}),\ u_k \in \mathbb{R}^l \\ x_k^{NX} &= (u_k, u_{k-1}, ..., u_{k-d}),\ x_k \in \mathbb{R}^{(d+1)\times l} \end{aligned} \tag{3.14}$$

3.4.2 Nonlinear Autoregressive Exogenous Model

In the case of NARX systems, the output depends non linearly not only on the $d+1$ exogenous variables but also on the last q system outputs, as stated in Equation (3.15). This can be clearly seen in Figure 3.3b, where the output y_k is a function of $u_{k-d}, ..., u_{k-1}, u_k$ and $y_{k-1}, y_{k-2}, ..., y_{k-q}$.

$$x_k^{NARX} = (y_{k-1}, .., y_{k-q}, u_k, u_{k-1}, ..., u_{k-d}),\ x_k \in \mathbb{R}^{q+((d+1)\times l)} \tag{3.15}$$



(a) NX System Structure

(b) NARX System Structure

Figure 3.3: System structures for modeling and predicting Time Series

4 Methods

4.1 Problem Statement

The aim of this work is to design and implement a Safe Active Learning framework for Time Series Modeling using NARX Gaussian Processes. Such a framework should be able to learn a physical model by exploring the most informative trajectories while taking the safety of the system into consideration.

This thesis builds upon the previous work by Zimmer et al. on "Safe Active Learning for Time-Series Modeling with Gaussian Processes" [ZMN18], where only NX GPs were considered, and extends it by considering the case of NARX GPs.

The framework presented in [ZMN18] can be illustrated by Algorithms 4.1 and 4.2.

Algorithm 4.1 shows a high-level representation of the Safe Active Learning framework developed in [ZMN18]. To start this framework the following inputs are required: α: Represents the allowed probability to enter an unsafe area during the Safe Exploration;

Algorithmus 4.1 Safe Active Learning for Time Series modeling

```
 1: function SAFEAL($\alpha$, $iterationsNumber$, $initialSafeTrays$, $initialPoint$)
 2:     $(trainingSetX, trainingSetY, trainingSetSafety) = initialSafeTrays$
 3:     $modelGP$ = TRAINGP($trainingSetX$, $trainingSetY$)
 4:     $safetyGP$ = TRAINGP($trainingSetX$, $trainingSetSafety$)
 5:     $start = initialPoint$
 6:     for $iterationsNumber$ do
 7:         $nextTray$ = FINDTRAY($modelGP$, $safetyGP$, $\alpha$, $start$)
 8:         $outputTray, safetyTray$ = EXECUTEINSYSTEM($nextTray$)
 9:         $modelGP$ = UPDATEGP($nextTray$, $outputTray$)
10:         $safetyGP$ = UPDATEGP($nextTray$, $safetyTray$)
11:         $start = nextTray.last$
12:     end for
13:     return $modelGP, safetyGP$
14: end function
```

$iterationsNumber$: Number of iterations of the framework. This is the number of new trajectories that will be collected during the Safe Exploration; $initialSafeTrays$: Initial set of trajectories that are known to be safe and were already explored; $initialPoint$: Starting point for the Safe Exploration.

Before starting the Safe Exploration, 2 GPs are trained using the initial data, the first one representing the physical model and the second one to model the safety constraints of the system (Lines 2-4). Afterwards, the starting point of the Safe Exploration is set to the given initial point (Line 5). At each iteration: The next trajectory to explore is chosen (Line 7) and tested in the system, obtaining the system output and the safety information for the given trajectory (Line 8). Then, this system output and the safety information are used to update the model and safety GPs (Lines 9-10). Finally, in order to continue the exploration, the starting point of the next iteration is set to be the last point of the actual trajectory (Line 11).

The key component of the Safe Exploration is the one that chooses the next trajectory τ^* to be explored (Line 7).

A trajectory τ of length s is characterized by the s consecutive inputs x_k of the system, as stated in Equation (4.1).

$$\begin{aligned} &u_k = (u_{k,1}, u_{k,2}, ..., u_{k,l}),\ u_k \in \mathbb{R}^l \\ &x_k = (u_k, u_{k-1}, ..., u_{k-d}),\ x_k \in \mathbb{R}^{(d+1)\times l} \\ &\tau = (x_{k+1}, ..., x_{k+s}),\ \tau \in \mathbb{R}^{s\times(d+1)\times l} \end{aligned} \tag{4.1}$$

The function that chooses the next trajectory is defined in Algorithm 4.2 and corresponds to the constraint optimization problem stated in Equation (4.2), with $I(\tau)$ being the optimality criterion that drives the exploration and $g(\tau)$ being the safety metric function.

$$\begin{aligned} &\tau^* = \arg\max_x I(\tau),\ st\ P(g(\tau) \geq 0) \geq 1 - \alpha \\ &I : \mathbb{R}^{s\times(d+1)\times l} \rightarrow \mathbb{R} \\ &\alpha \in \mathbb{R},\ 0 < \alpha < 1 \\ &g : \mathbb{R}^{s\times(d+1)\times l} \rightarrow \mathbb{R} \end{aligned} \tag{4.2}$$

The optimality criterion $I(\tau)$ represents the predictive information gain that will be obtained if the trajectory τ is explored, and is calculated by the function $objective$ (Lines 7-12).

Algorithmus 4.2 Find next trajectory

```
 1: function FINDTRAY($modelGP$, $safetyGP$, $\alpha$, $start$)
    // Find optimum end for the trajectory starting at $start$
 2:     $endTray$ = FINDMIN(OBJECTIVE($modelGP$, $start$), SAFETYCONSTRAINT($safetyGP$, $start$, $\alpha$))
    // Reconstruct trajecory from start and end points
 3:     $nextTray$ = TRAYFROMTO($start$, $endTray$)
 4:     return $nextTray$
 5: end function
 6: function OBJECTIVE($modelGP$, $start$)
    // This function takes the model and the start point of the trajectory and returns the
    objective function for this specific optimization problem
 7:     return function EVALUATEOBJECTIVE( $endTray$)
 8:         $tray$ = TRAYFROMTO($start$, $endTray$)
 9:         $\Sigma$ = $modelGP$.PREDICTIVECOVARIANCE($tray$)
10:         return I($\Sigma$)
11:     end function
12: end function
13: function SAFETYCONSTRAINT($safetyGP$, $start$, $\alpha$)
    // This function takes the model, the start point of the trajectory and the predefined
    $\alpha$, and returns the constraint function for this specific optimization problem
14:     return function EVALUATESAFETYCONSTRAINT( $endTray$)
15:         $tray$ = TRAYFROMTO($start$, $endTray$)
16:         $\mu$ = $safetyGP$.PREDICTIVEMEAN($tray$)
17:         $\Sigma$ = $safetyGP$.PREDICTIVECOVARIANCE($tray$)
18:         $probDistSafety$ = $\mathcal{N}(\mu, \Sigma)$
19:         $cumulativeProbSafe$ = $probDistSafety$.CUMULATIVEPROB($x > 0$)
20:         $constraint$ = $cumulativeProbSafe$ - (1-$\alpha$)
21:         return $constraint$
22:     end function
23: end function
```

The corresponding constraint is computed by $safetyConstraint$ (Lines 13-22). Here, the safety metric $g(\tau)$ returns positive values for trajectories that remain inside the safe area, and negative values otherwise. This function is the one learnt by the safety GP. Consequently, the mean and variance of the safety GP's prediction are used to compute the distribution over $g(\tau)$, and afterwards to compute the probability of the trajectory being safe (Lines 16-19) Summarizing, this constraint optimization problem is solved by finding the most informative trajectory that is safe with a probability higher than $1 - \alpha$. For this specific implementation, the trajectories are defined by their start and endpoint, but other parameterizations are also possible.

Since NARX GP structures are considered in this thesis, the above described framework needs to be modified accordingly. The main difference between using an NX- and a NARX-structure to predict trajectories is that, while in the NX case n-steps ahead predictions can be computed in a straight forward manner, uncertainties in the GP's inputs have to be considered in order to make these predictions using a NARX structure. This problem is explained in detail in the next subsection.

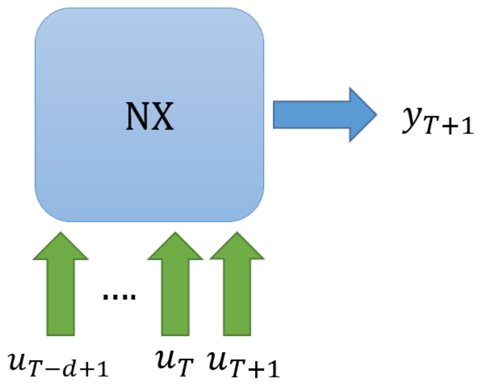
To tackle the problem of predicting a trajectory using a NARX GP some modifications in Algorithm 4.2 need to be introduced. In particular, the trajectory predictions for the $objective$ and $safetyConstraint$ functions in Lines 9 and 16-17 have to be modified.

4.1.1 Trajectory predictions using NARX models

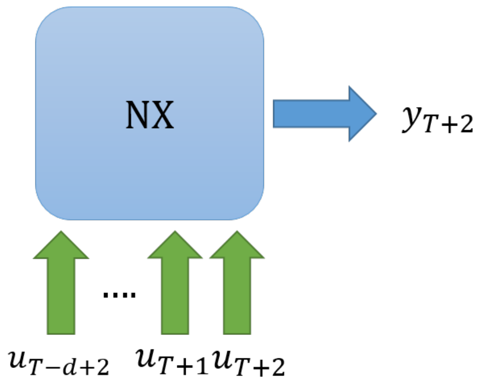
As explained in Section 3.4, the output of a NARX model depends not only on some inputs of the system but also on some previous outputs. This dependency on previous outputs differentiates the NARX structures from the NX ones (see Equation (4.3)) and plays a big role when using GPs with this structure to predict trajectories. Figures 4.2 and 4.1 clearly illustrate this difference.

$$\begin{aligned} &x_k = (y_{k-1}, .., y_{k-q}, u_k, u_{k-1}, ..., u_{k-d}),\ x_k \in \mathbb{R}^{q+((d+1)\times l)} \\ &\tau = (x_{k+1}, ..., x_{k+s}),\ \tau \in \mathbb{R}^{s\times(q+((d+1)\times l))} \end{aligned} \tag{4.3}$$

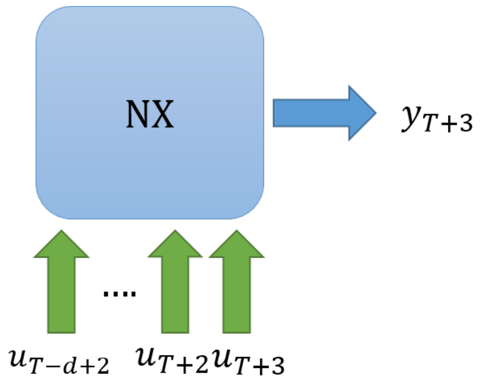
Figure 4.1 shows that in the case of an NX model the output of the system at each time step only depends on some known inputs, which makes it possible to predict every step of the trajectory independently. In contrast, the output of the NARX model also depends on the previous outputs of the model (as shown in Figure 4.2), which causes the step-wise prediction of the trajectory dependent on the previous steps' outputs. Considering that these models are approximated using GPs, the output at each time-step is a Gaussian distribution characterized by its mean and covariance matrix (as stated in



(a) First step (time-step $t = T + 1$)

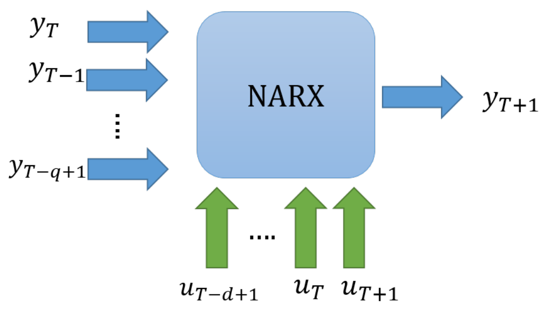


(b) Second step (time-step $t = T + 2$)

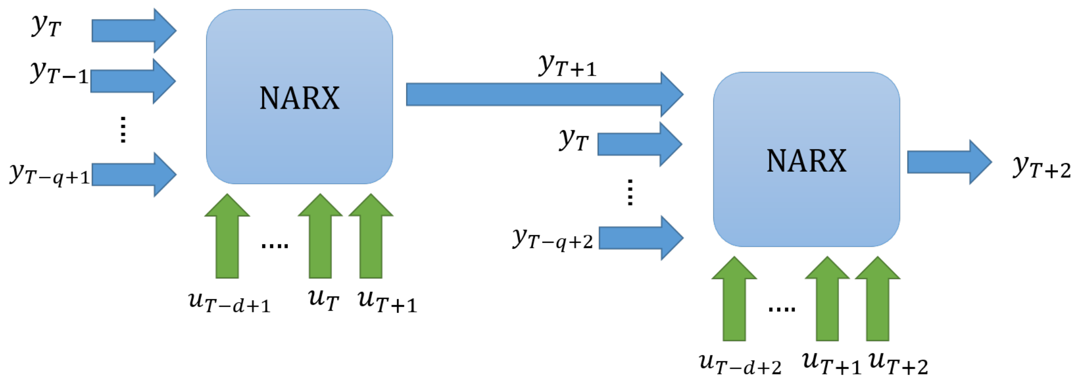


(c) Third step (time-step $t = T + 3$)

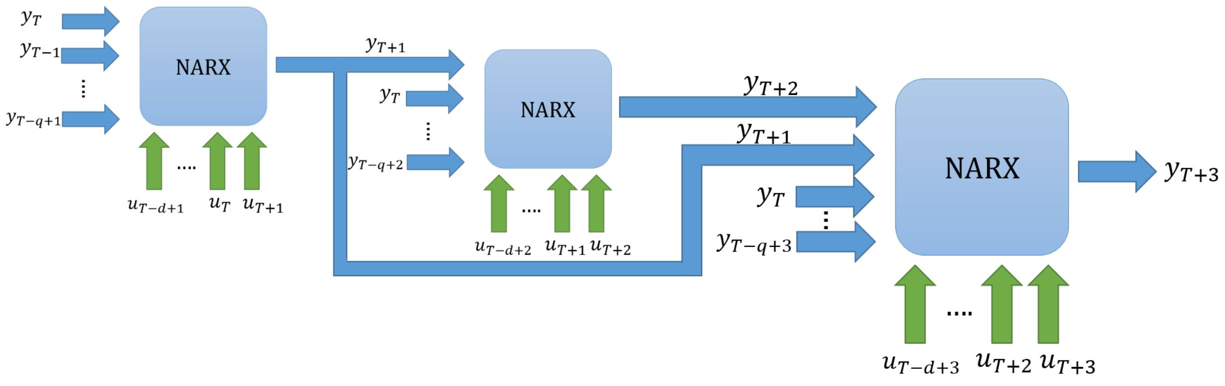
Figure 4.1: Trajectory prediction with NX models - Predictions corresponding to a 3-step trajectory, using a NX model that takes as input the last d inputs. The trajectory to be predicted starts from the starting point y_T, having its first step at time-step $t = T + 1$ and following input history: $u_{T-d+1}, .., u_T$.



(a) First step (time-step $t = T + 1$)



(b) Second step (time-step $t = T + 2$)



(c) Third step (time-step $t = T + 3$)

Figure 4.2: Trajectory prediction with NARX models - Predictions corresponding to a 3-step trajectory, using a NARX model that takes as input the last q outputs and the last d inputs. All the past outputs (outputs up to the time-step $t = T$) are known. The trajectory to be predicted starts from the starting point y_T, having its first step at $t = T + 1$ and the following input history: $u_{T-d+1}, .., u_T$ and output history: $y_{T-q+1}, .., y_T$.

Equation (4.4)). As this output is then used to predict the next steps of the trajectory, the inputs of our models are no longer known values, but probabilities distributions. This setting is known as noisy inputs or uncertainty propagation in GPs.

$$y_t \sim \mathcal{N}(\mu_t, \Sigma_t) \tag{4.4}$$

4.2 Proposed solutions

There are many different approaches to handle uncertainty propagation in GPs that can be applied to the trajectory prediction problem. The most widely used for this purpose are:

- Naive Approximation by mean: Consists of replacing the complete distribution, only by its mean value. This is the easiest approach, but also the one discarding the most information.
- Linearization of the posterior mean [KF09]: Here, the Gaussian Process is approximated by its first term of the Taylor Expansion. This technique still discards a lot of information on the real distribution, as it approximates the GP linearly.
- Moment Matching Approximation [Gir04]: This approach is slightly more complex than the previous approaches, but the results can be derived analytically. It approximates the resulting distribution with a Gaussian distribution. As a result, this approximation is not very accurate if the distribution differs significantly from a Gaussian.
- Approximation by Quadratures [Vin18]: This method is more complex and computationally more expensive than the previous ones, but yields better results for non Gaussian distribution and allows for error analysis of its approximations.
- Monte Carlo Approximation [RC13]: This method, when run with a sufficiently large number of samples, is widely used as ground truth when evaluating distributions approximations.

For the purpose of this thesis, only the naive, Moment Matching and Monte Carlo approximations are considered. Linearization was disregarded, as better results can be obtained using Moment Matching, without much difference in computational time and complexity. Furthermore, as the optimality criterion depends on a Gaussian covariance function, only Gaussian distribution are considered. Therefore, approximations by quadratures were also left out. The chosen approaches have different compromises between accuracy and computation time, which are two key characteristics for the problem at hand, as the trajectory predictions need to be executed multiple times when solving the constraint optimization problem, and also on their accuracy will depend the efficiency and safety of the exploration.

4.2.1 Naive Approximation by Mean

The naive approach consists of simply approximating the given distribution by its mean value. This approach for handling the uncertainty in the inputs is the easiest and fastest one to compute but also the one that discards the most information, as the complete distribution is only represented by one value.

A trajectory prediction of length s using this approach looks as follows:

- $t = T + 1$

  Straightforward prediction with the known inputs: $y_{T+1} \sim \mathcal{N}(\mu_{T+1}, \sigma^2_{T+1})$
- $t = T + 2$

  Input of the GP prediction: $\begin{bmatrix} \mu_{T+1} & y_T & \cdots & y_{T-q+2} & u_{T+2} & \cdots & u_{T-d+2} \end{bmatrix}^T$
  Prediction: $y_{T+2} \sim \mathcal{N}(\mu_{T+2}, \sigma^2_{T+2})$
- $t = T + 3$

  Input of the GP prediction: $\begin{bmatrix} \mu_{T+2} & \mu_{T+1} & y_T & \cdots & y_{T-q+3} & u_{T+3} & \cdots & u_{T-d+3} \end{bmatrix}^T$
  Prediction: $y_{T+3} \sim \mathcal{N}(\mu_{T+3}, \sigma^2_{T+3})$
- $t = T + s$

  Input of the GP prediction:
  $u_{T+s} = \begin{bmatrix} \mu_{T+s-1} & \cdots & \mu_{T+1} & y_T & \cdots & y_{T-q+s} & u_{T+s} & \cdots & u_{T-d+s} \end{bmatrix}^T$
  Prediction: $y_{T+s} \sim \mathcal{N}(\mu_{T+s}, \sigma^2_{T+s})$

When applying this approach in the trajectory prediction for the NARX model, the algorithm needs to be modified as shown in Algorithm 4.3. In particular, predictions need to be made step-wise, and the outputs' means saved to be input in the following time-steps predictions (function $naivePrediction$). Only after all predictions are made the covariance matrix of the trajectory can be computed (Lines 5 and 13).

Algorithmus 4.3 Naive predictions

```
function OBJECTIVE($modelGP$, $start$)
    return function EVALUATEOBJECTIVE( $endTray$)
        $tray$ = TRAYFROMTO($start$, $endTray$)
        $trayOutputs$ = NAIVEPREDICTION($modelGP$, $tray$)
        $\Sigma$ = $modelGP$.PREDICTIVECOVARIANCE($tray$, $trayOutputs$)
        return I($\Sigma$)
    end function
end function
function SAFETYCONSTRAINT($safetyGP$, $start$, $\alpha$, $endTray$)
    return function EVALUATESAFETYCONSTRAINT( $endTray$)
        $tray$ = TRAYFROMTO($start$, $endTray$)
        $trayOutputs$ = NAIVEPREDICTION($safetyGP$, $tray$)
        $\mu$ = $trayOutputs$
        $\Sigma$ = $safetyGP$.PREDICTIVECOVARIANCE($tray$, $trayOutputs$)
        $probDistSafety$ = $\mathcal{N}(\mu, \Sigma)$
        $cumulativeProbSafe$ = $probDistSafety$.CUMULATIVEPROB($x > 0$)
        $constraint$ = $cumulativeProbSafe$ - (1-$\alpha$)
        return $constraint$
    end function
end function
function NAIVEPREDICTION($GP$, $tray$)
    $trayOutputs$ = []
    for $tray$.STEPS( )do
        $\mu$ = $GP$.PREDICTIVEMEAN($tray$, $trayOutputs$)
        $trayOutputs$.ADD($\mu$)
    end for
    return $trayOutputs$
end function
```

4.2.2 Moment Matching Approximation

Using Girard's equations for Moment Matching [Gir04], the prediction of a GP at a noisy (or uncertain) input can be approximated by a normal distribution using the exact mean and covariance of the real distribution.

These Moment Matching equations can be derived given the GP that makes noise-free predictions (Equation (4.5)) and the probability distribution over the input (Equation (4.6)).

The GP with noise-free predictions is defined by:

$$\begin{aligned} \mu_G(\nu) &= \sum_{i=1}^{N} \beta_i C_G(\nu, x_i) \\ \sigma_G^2(\nu) &= C_G(\nu,\nu) - \sum_{i,j=1}^{N} K_{ij}^{-1} C_G(\nu, x_i) C_G(\nu, x_j) \\ \beta &= K^{-1} t \\ t &= \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}^T \\ C_G(x_i, x_j) &= e^{-\frac{1}{2}(x_i - x_j)^T W^{-1} (x_i - x_j)} \end{aligned} \tag{4.5}$$

where x_i is the ith input, t is the $N \times 1$ vector of observed outputs, K is the $N \times N$ covariance matrix, and $C_G(x_i, x_j)$ is the Gaussian covariance function, with given length-scale W^2.

The inputs' probability distribution is defined by its mean ν and covariance Σ_x, as follows:

$$x \sim \mathcal{N}(\nu, \Sigma_x) \tag{4.6}$$

The exact mean $m^{ex_G}(\nu, \Sigma_x)$ and variance $v^{ex_G}(\nu, \Sigma_x)$ of the GP with uncertain input can be computed using Moment Matching and result in Equations 4.7 and 4.8. For a detailed derivation of these equations please refer to Girard's work [Gir04].

$$\begin{aligned} m^{ex_G}(\nu, \Sigma_x) &= \sum_{i=1}^{N} \beta_i C_G(\nu, x_i) C_{corr}(\nu, x_i) \\ C_{corr}(\nu, x_i) &= \frac{1}{|\mathbb{I} + W^{-1}\Sigma_x|^2} e^{\left[\frac{1}{2}(\nu - x_i)^T \Delta^{-1} (\nu - x_i)\right]} \\ \Delta^{-1} &= W^{-1} - (W + \Sigma_x)^{-1} \end{aligned} \tag{4.7}$$

$$\begin{aligned}
v^{exG}(\nu, \Sigma_x) &= \sigma_G^2(\nu) + \sum_{i,j=1}^{N} K_{ij}^{-1} C_G(\nu, x_i) C_G(\nu, x_j)(1 - C_{corr_2}(\nu, \bar{x}_{ij})) \\
&\quad + \sum_{i,j=1}^{N} \beta_i \beta_j C_G(\nu, x_i) C_G(\nu, x_j)(C_{corr_2}(\nu, \bar{x}_{ij}) - C_{corr}(\nu, x_i) C_{corr}(\nu, x_j)) \\
C_{corr_2}(\nu, x) &= \frac{1}{|\left(\frac{W}{2}\right)^{-1} \Sigma_x + \mathbb{I}|^2} e^{\left[\frac{1}{2}(\nu - x)^T \Lambda^{-1} (\nu - x)\right]} \qquad (4.8) \\
\Lambda^{-1} &= 2W^{-1} - \left(\frac{1}{2}W + \Sigma_x\right)^{-1} \\
\bar{x}_{ij} &= \frac{x_i + x_j}{2}
\end{aligned}$$

This procedure can be applied iteratively to predict multiple steps ahead of a NARX GP, as explained by Girard et al. [Gir04; GM05; GRCM03]. At each time-step, the input distribution is computed and then the output is approximated by a normal distribution with the mean and variance calculated using Equations 4.7 and 4.8.

To illustrate this procedure, one can take the general NARX model depicted in Figure 4.2 as an example, where the history of inputs and outputs up to time $t = T$ are known and the trajectory to be predicted starts at time $t = T$ and has length s. At the first step ($t = T + 1$), the original noise-free prediction can be used, as all the inputs are known. For the following steps the uncertainty introduced by the inputs y_{T+h} (with $1 \leq h \leq s$) has to be considered when computing the input distribution as follows:

- $t = T + 1$

  Straightforward prediction: $y_{T+1} \sim \mathcal{N}(\mu_{T+1}, \sigma_{T+1}^2)$

- $t = T + 2$

  The input distribution is characterized by:

$$\nu_{T+2} = \begin{bmatrix} \mu_{T+1} \\ y_T \\ \vdots \\ y_{T-q+2} \\ u_{T+2} \\ \vdots \\ u_{T-d+2} \end{bmatrix} \text{ and } \Sigma_{x_{T+2}} = \begin{bmatrix} \sigma_{T+1}^2 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

- $t = T + 3$

  The input distribution is characterized by:

$$\nu_{T+3} = \begin{bmatrix} m^{ex_G}_{T+2} \\ \mu_{T+1} \\ y_T \\ \vdots \\ y_{T-q+3} \\ u_{T+3} \\ \vdots \\ u_{T-d+3} \end{bmatrix} \text{ and } \Sigma_{x_{T+3}} = \begin{bmatrix} v^{ex_G}_{T+1} & Cov(y_{T+2}, y_{T+1}) & 0 & \cdots & 0 \\ Cov(y_{T+1}, y_{T+2}) & \sigma^2_{T+1} & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix}$$

- $t = T + s$

  The input distribution is characterized by:

$$\nu_{T+s} = \begin{bmatrix} m^{ex_G}_{T+s-1} & m^{ex_G}_{T+s-2} & \cdots & \mu_{T+1} & y_T & \cdots & y_{T-q+s} & u_{T+s} & \cdots & u_{T-d+s} \end{bmatrix}^T$$

$$\text{and } \Sigma_{x_{T+s}} = \begin{bmatrix} v^{ex_G}_{T+s-1} & Cov(y_{T+s-1}, y_{T+s-2}) & \cdots & Cov(y_{T+s-1}, y_{T+1}) \\ Cov(y_{T+s-2}, y_{T+s-1}) & v^{ex_G}_{T+s-2} & \cdots & Cov(y_{T+s-2}, y_{T+1}) \\ \vdots & \vdots & \vdots & \vdots \\ Cov(y_{T+1}, y_{T+s-1}) & Cov(y_{T+1}, y_{T+s-2}) & \cdots & \sigma^2_{T+1} \\ 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

Where the following abbreviations were used:

$$\mu_t = \mu_G(\nu_t)$$

$$\sigma^2_t = \sigma^2_G(\nu_t)$$

$$m^{ex_G}_t = m^{ex_G}(\nu_t, \Sigma_{x_t})$$

$$v^{ex_G}_t = v^{ex_G}(\nu_t, \Sigma_{x_t})$$

Moreover, not only the mean and variances of the previous outputs are needed to compute this predictions, but also the covariances between these outputs. These covariances can be computed using Equation (4.9), that serves to compute the covariance between the input and output of the GP. This equation gives us a vector with the covariance between each of the inputs and the output, so the covariance between the output and each of the previous outputs can be obtained from this vector. For a detailed derivation of Equation (4.9) please refer to [Gir04].

$$\begin{aligned}
Cov[y_{T+h}, x_{T+h}] &= \sum_{i=1}^{N} \beta_i C(\nu_{T+h}, x_i)(\mathbb{I} + W\Sigma_{T+h}^{-1})^{-1} x_i \\
x_{T+h} &= \begin{bmatrix} y_{T+h-1} & \cdots & y_{T-q+h} & u_{T+h} & \cdots & u_{T-d+h} \end{bmatrix}^T \\
Cov[y_{T+h}, x_{T+h}] &= \Big[Cov[y_{T+h}, y_{T+h-1}] \cdots Cov[y_{T+h}, y_{T-q+h}] \\
&\qquad Cov[y_{T+h}, u_{T+h}] \cdots Cov[y_{T+h}, u_{T-d+h}]\Big]
\end{aligned} \tag{4.9}$$

Algorithm 4.4 shows the modified algorithm using Moment Matching for the trajectory predictions. In this case, the predictive mean and covariance are computed by the function $MomentMatchingPrediction$. As mentioned before, the key to this procedure is to keep computing the input distribution and then, apply the Moment Matching formulas to obtain the predictions. At the beginning of the trajectory the inputs are all known, so there is no underlying distribution, but just the known values (Lines 18-20). Then, for the first step, the prediction is exactly the noise-free one from the GP (Line 23). For the following steps, the input mean and covariance needs to be updated with the output distribution of previous time-steps, so after each step mean, variance and covariances are stored (Lines 24-25 and 32-34). The input's mean and covariance are computed taking into account the step's inputs and the previous outputs (Lines 27-28). And finally, the predictions are computed using the Moment Matching equations presented above (Lines 29-31). The mean (m^{ex_G}) is computed with Equation (4.7), the variance (v^{ex_g}) with Equation (4.8) and the covariance between outputs with Equation (4.9).

Algorithmus 4.4 Moment Matching Predictions

```
1: **function** OBJECTIVE($modelGP$, $start$)
2:     **return function** EVALUATEOBJECTIVE( $endTray$)
3:         $tray$ = TRAYFROMTO($start$, $endTray$)
4:         $\mu$, $\Sigma$ = MOMENTMATCHINGPREDICTION($modelGP$, $tray$)
5:         **return** I($\Sigma$)
6:     **end function**
7: **end function**
8: **function** SAFETYCONSTRAINT($safetyGP$, $start$, $\alpha$, $endTray$)
9:     **return function** EVALUATESAFETYCONSTRAINT( $endTray$)
10:        $tray$ = TRAYFROMTO($start$, $endTray$)
11:        $\mu$, $\Sigma$ = MOMENTMATCHINGPREDICTION($safetyGP$, $tray$)
12:        $probDistSafety$ = $\mathcal{N}(\mu, \Sigma)$
13:        $cumulativeProbSafe$ = $probDistSafety$.CUMULATIVEPROB($x > 0$)
14:        $constraint$ = $cumulativeProbSafe$ - (1-$\alpha$)
15:        **return** $constraint$
16:    **end function**
17: **end function**
18: **function** MOMENTMATCHINGPREDICTION($GP$, $tray$)
19:    $outputMeans$ = $tray$.OUTPUTHISTORY
20:    $outputVariances$ = 0
21:    $outputCovariance$ = 0
22:    **for** $step$ in $tray$.STEPS( )**do**
23:        **if** ($step$ is first) **then**
24:            $\mu_G$, $\sigma_G^2$ = $GP$.PREDICT($step$)
25:            $outputMeans$.ADD($\mu_G$)
26:            $outputVariances$.ADD($\sigma_G^2$)
27:        **else**
28:            $u$ = INPUMEANFORSTEP($step$.INPUT , $outputMeans$)
29:            $\Sigma_x$ = INPUTCOVFORSTEP( $outputVariances$, $outputCovariance$)
30:            $m^{ex_G}$ = MOMENTMATCHINGMEAN($GP$, $u$, $\Sigma_x$)
31:            $v^{ex_G}$ = MOMENTMATCHINGVAR($GP$, $u$, $\Sigma_x$)
32:            $Cov$ = MOMENTMATCHINGVAR($GP$, $u$, $\Sigma_x$)
33:            $outputMeans$.ADD($m^{ex_G}$)
34:            $outputVariances$.ADD($v^{ex_G}$)
35:            $outputCovariance$.ADD($Cov$)
36:        **end if**
37:    **end for**
38:    $\Sigma_{tray}$ = PREDCOVARIANCE($outputVariances$, $outputCovariance$)
39:    **return** $outputMeans$, $\Sigma_{tray}$
40: **end function**
```

4.2.3 Monte Carlo Approximation

This method consists of drawing S samples from the output distribution of the GP at the first step of the trajectory and then using these samples to obtain predictions for the following steps. Resulting in S predictions for the output at each time-step, that can be used to reconstruct the underlying probability distribution.

A trajectory prediction of length s using this approach looks as follows:

- $t = T + 1$

  Predict using the known inputs: $y_{T+1} \sim \mathcal{N}(\mu_{T+1}, \sigma^2_{T+1})$
  Draw N samples from the obtained distribution

- $t = T + 2$

  For each drawn sample y^i_{T+1}:
  Use the following input for the GP prediction:
  $\begin{bmatrix} y^i_{T+1} & y_T & \cdots & y_{T-q+2} & u_{T+2} & \cdots & u_{T-d+2} \end{bmatrix}^T$
  Prediction: $y^i_{T+2} \sim \mathcal{N}(\mu^i_{T+2}, {\sigma^2_{T+2}}^i)$
  Draw one random sample from each obtained distribution.

- $t = T + 3$

  For each pair of drawn samples (y^i_{T+1}, y^i_{T+2}):
  Use the following input for the GP prediction:
  $\begin{bmatrix} y^i_{T+2} & y^i_{T+1} & y_T & \cdots & y_{T-q+3} & u_{T+3} & \cdots & u_{T-d+3} \end{bmatrix}^T$
  Prediction: $y^i_{T+3} \sim \mathcal{N}(\mu^i_{T+3}, {\sigma^2_{T+3}}^i)$
  Draw one random sample from each obtained distribution.

- $t = T + s$

  For each tuple of drawn samples $(y^i_{T+1}, y^i_{T+2}, ..., y_{T+s-1})$:
  Use the following input for the GP prediction:
  $u_{T+s} = \begin{bmatrix} \mu_{T+s-1} & \cdots & \mu_{T+1} & y_T & \cdots & y_{T-q+s} & u_{T+s} & \cdots & u_{T-d+s} \end{bmatrix}^T$
  Prediction: $y^i_{T+s} \sim \mathcal{N}(\mu^i_{T+s}, {\sigma^2_{T+s}}^i)$
  Draw one random sample from each obtained distribution.

Finally, with all the drawn samples the mean and covariance of the trajectory is computed.

For this approach the procedure is modified as illustrated by (Algorithm 4.5). A major difference is that a new parameter needs to be defined. This is the number of samples

Algorithmus 4.5 Monte Carlo Predictions

```
function OBJECTIVE($modelGP$, $start$, $N$)
    return function EVALUATEOBJECTIVE( $endTray$)
        $tray$ = TRAYFROMTO($start$, $endTray$)
        $\mu$, $\Sigma$ = MONTECARLOPREDICTION($modelGP$, $tray$, $N$)
        return I($\Sigma$)
    end function
end function
function SAFETYCONSTRAINT($safetyGP$, $start$, $\alpha$, $endTray$, $N$)
    return function EVALUATESAFETYCONSTRAINT( $endTray$)
        $tray$ = TRAYFROMTO($start$, $endTray$)
        $\mu$, $\Sigma$ = MONTECARLOPREDICTION($safetyGP$, $tray$, $N$)
        $probDistSafety = \mathcal{N}(\mu, \Sigma)$
        $cumulativeProbSafe$ = $probDistSafety$.CUMULATIVEPROB($x > 0$)
        $constraint$ = $cumulativeProbSafe$ - (1-$\alpha$)
        return $constraint$
    end function
end function
function MONTECARLOPREDICTION($GP$, $tray$, $N$)
    $samples$ = [][]
    for $step$ in $tray$.STEPS( )do
        if ($step$ is first) then
            $\mu_G$, $\sigma_G^2$ = $GP$.PREDICT($step$)
            samples[$step$] = SAMPLESFROMNORMDIST($\mu_G$, $\sigma_G^2$, $N$)
        else
            for $i$ in range($N$) do
                $\mu_G$, $\sigma_G^2$ = $GP$.PREDICT($step$, $samples$[:][$i$])
                samples[$step$][$i$] = SAMPLESFROMNORMDIST($\mu_G$, $\sigma_G^2$, 1)
            end for
        end if
    end for
    $\mu_{tray}$ = PREDMEAN(samples)
    $\Sigma_{tray}$ = PREDCOVARIANCE(samples)
    return $\mu_{tray}$, $\Sigma_{tray}$
end function
```

used to approximate the real distributions. The larger this number, the more precise the approximation will be, but also the more time consuming the calculation. The function $monteCarloPrediction$ uses samples to approximate the predictions and then takes the best fitting Normal approximation of these predictions (Lines 30-31). To obtain the samples, at the first step of the trajectory N samples are drawn from the GP prediction (Lines 21-22). Then, these samples are used as inputs for the following time-steps' predictions (Line 25). Afterwards, from each of the predictions generated by a sample history, a new sample is drawn from the prediction distribution and added to the corresponding history (Line 26). Finally, the trajectory prediction is approximated using the samples mean and covariance (Lines 30-31).

5 Evaluation

5.1 Application

The particular case of study in this thesis is the safe active learning of the high-pressure injection system addressed in [ZMN18] and depicted in Figure 5.1.

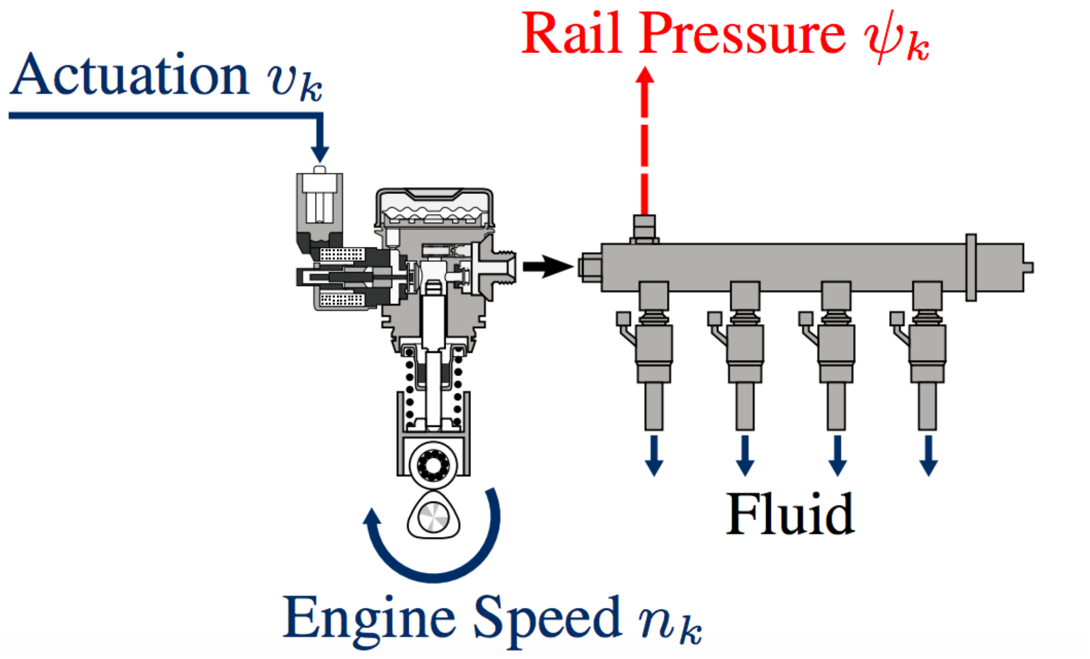
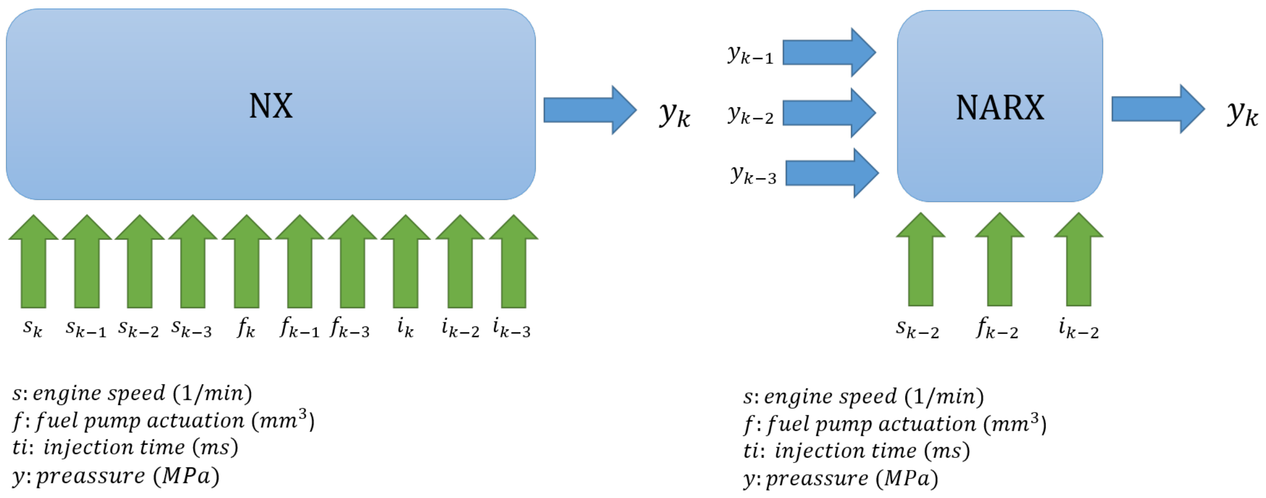


Figure 5.1: High-Pressure Injection System (taken from [ZMN18])

In this system the engine speed v_k and fuel pump actuation n_k are the controllable inputs and the rail pressure ψ_k is the system's output. The sub-fix k represent the time-step number of the input/output.

Due to lack of access to the real physical system for experiments, an analytical model of the system was used as a ground truth. This model is illustrated in Figure 5.2b. Here, the inputs s and f can be controlled, while ti is fixed during the experiments. When tested on the physical system this NARX model reported a NRMSE of 8.3%, 13.4% and 8.4% for 3 different test sets, while the analytical NX model used by Zimmer et al. as a ground truth (Figure 5.2a) reported NRMSEs of 22.2%, 15.6% and 10.2% for the same test sets. As the analytical NARX model is closer to the real physical system, it is considered to be more suitable to be used as a ground truth.


NX
y_k
s_k s_{k-1} s_{k-2} s_{k-3} f_k f_{k-1} f_{k-3} i_k i_{k-2} i_{k-3}
s: engine speed (1/min)
f: fuel pump actuation (mm^3)
ti: injection time (ms)
y: preassure (MPa)
y_{k-1}
y_{k-2}
y_{k-3}
NARX
y_k
s_{k-2} f_{k-2} i_{k-2}
s: engine speed (1/min)
f: fuel pump actuation (mm^3)
ti: injection time (ms)
y: preassure (MPa)

(a) NX model used in [ZMN18] **(b)** NARX model

Figure 5.2: Analytical Models used to model the High-Pressure Injection System - Models that result from the Principal Component Analysis of the system (courtesy of Mark Schillinger)

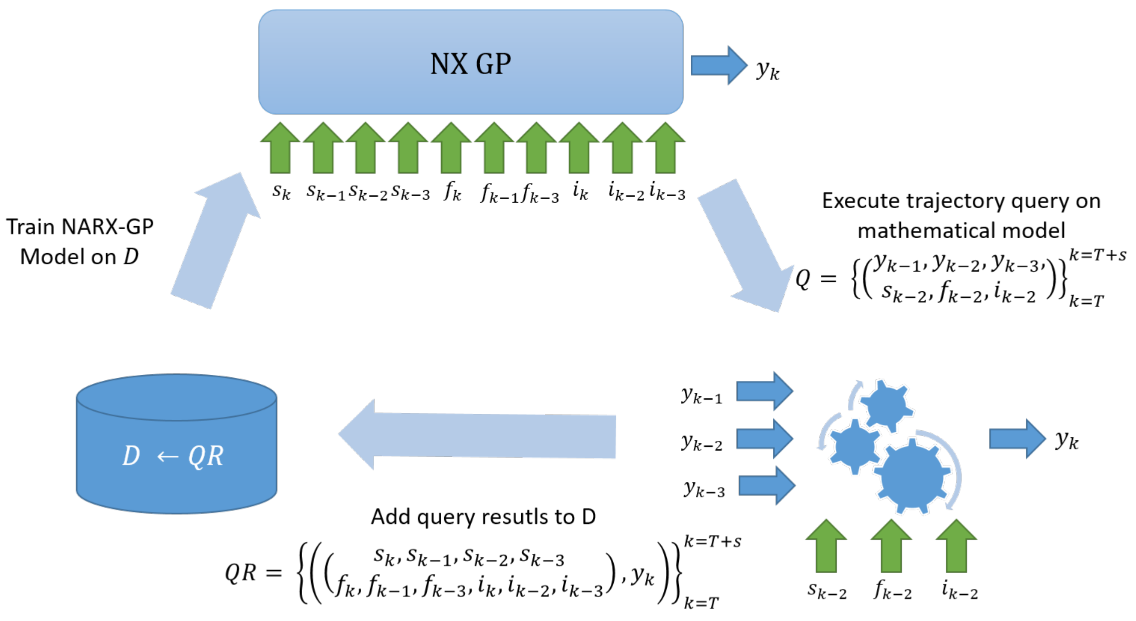
5.2 Experiments

Four different settings were developed and tested with two main goals: first, to compare the performance of NX GPs against NARX ones when learning a physical model with a NARX structure, and second, to compare the chosen different approaches for the NARX scenario (described in Section 4.2).

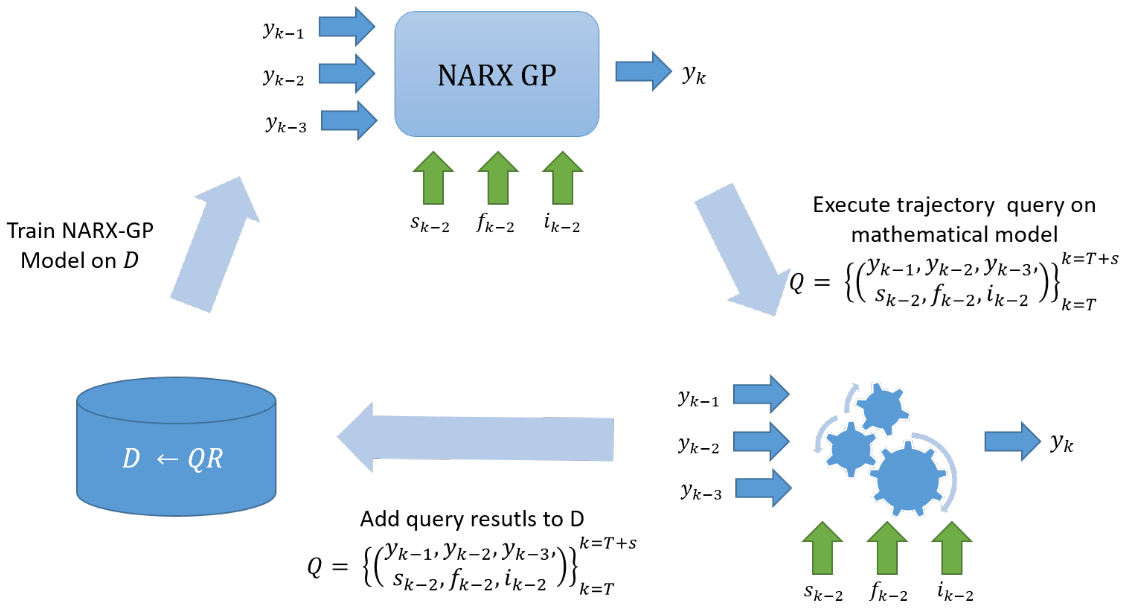
In all experiments the mathematical model of Figure 5.2b was used as a ground truth and 5-steps trajectories were planned at each iteration of the Safe Exploration.

The chosen experiment settings were the following:

1. NX GPs structure: This experiment is designed to test the performance of the framework developed by Zimmer et al. [ZMN18] when the real system has an underlying NARX structure and is used as benchmark for evaluating the results' of this masters' thesis. Consequently, NX GPs were used to learn the model and


NX GP
y_k
s_k s_{k-1} s_{k-2} s_{k-3} f_k f_{k-1} f_{k-3} i_k i_{k-2} i_{k-3}
Train NARX-GP Model on D
Execute trajectory query on mathematical model
$Q = \left\{\begin{pmatrix} y_{k-1}, y_{k-2}, y_{k-3}, \\ s_{k-2}, f_{k-2}, i_{k-2} \end{pmatrix}\right\}_{k=T}^{k=T+s}$
$D \leftarrow QR$
y_{k-1}
y_{k-2}
y_{k-3}
y_k
Add query resutls to D
$QR = \left\{\left(\begin{pmatrix} s_k, s_{k-1}, s_{k-2}, s_{k-3} \\ f_k, f_{k-1}, f_{k-3}, i_k, i_{k-2}, i_{k-3} \end{pmatrix}, y_k\right)\right\}_{k=T}^{k=T+s}$
s_{k-2} f_{k-2} i_{k-2}

(a) Safe Active Learning using NX GP - The framework starts with some initial training set, that is used to train the NX GPs for modelling the system and the safety constraint. Afterwards, the next trajectory is chosen and executed in the analytical model used as ground truth (analytical NARX model), and then the model's output and safety information are added to the training set. The procedure starts again now, using the new training set.


y_{k-1}
y_{k-2}
y_{k-3}
NARX GP
y_k
s_{k-2} f_{k-2} i_{k-2}
Train NARX-GP Model on D
Execute trajectory query on mathematical model
$Q = \left\{\begin{pmatrix} y_{k-1}, y_{k-2}, y_{k-3}, \\ s_{k-2}, f_{k-2}, i_{k-2} \end{pmatrix}\right\}_{k=T}^{k=T+s}$
$D \leftarrow QR$
y_{k-1}
y_{k-2}
y_{k-3}
y_k
Add query resutls to D
$Q = \left\{\begin{pmatrix} y_{k-1}, y_{k-2}, y_{k-3}, \\ s_{k-2}, f_{k-2}, i_{k-2} \end{pmatrix}\right\}_{k=T}^{k=T+s}$
s_{k-2} f_{k-2} i_{k-2}

(b) Safe Active Learning using NARX GP - The framework starts with some initial training set, that is used to train the NARX GPs for modelling the system and the safety constraint. Afterwards, the next trajectory is chosen and executed in the analytical model used as ground truth (analytical NARX model), and then the model's output and safety information are added to the training set. The procedure starts again now, using the new training set.

Figure 5.3: Experiments settings

the safety metric, using the mathematical model of Figure 5.2b as ground truth, as depicted in Figure 5.3a. The chosen NX structure used here is the same one used by Zimmer et al., which is the best NX structure found for this problem according to its Principal Component Analysis (courtesy of Mark Schillinger) (see Figure 5.2a).

A similar experiment was done by Zimmer et al. with the difference, that in their work they have also used an mathematical model with the same structure as ground truth.

2. NARX GPs structure: To evaluate the methods described in Chapter 4, NARX GPs were used to learn the model and the safety metric, using the mathematical model of Figure 5.2b as ground truth (see Figure 5.3b).

   For this case, the underlying NARX structure is assumed to be known and therefore used the learnt GP also has the structure depicted in Figure 5.2b.

   In this scenario, the following chosen approaches for uncertainty propagation estimation were tested:

   a) Naive trajectory predictions using mean value (described in Section 4.2.1)
   b) Trajectory predictions using Moment Matching (described in Section 4.2.2)
   c) Trajectory predictions using Monte Carlo (described in Section 4.2.3)

For implementation details plese refer to Appendix A.

5.3 Results

5.3.1 NX GP structure (Figure 5.3a)

When training NX GPs to learn the mathematical NARX model, the results from Figure 5.4 were obtained. In this case, one can see that the NX GP cannot adapt well to the ground truth model, even when a high risk of entering the unsafe region is allowed ($\alpha = 0.9$) and many iterations of the Safe Active Learning framework are run. Moreover, if a very low risk of entering the unsafe area is chosen ($\alpha = 0.2$), the algorithm is not able to explore enough trajectories, as to improve the first approximation of the model.

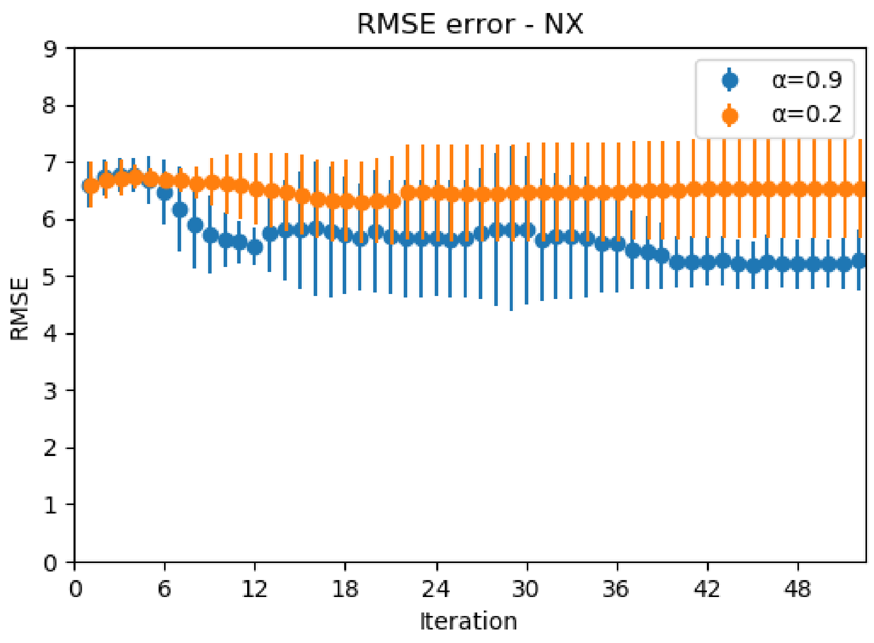


Figure 5.4: Reported RMSE accross 50 iterations of the Safe Active Learning framework for NX GPs [ZMN18] with different risk allowances of entering the unsafe area ($\alpha = 0.9$ and $\alpha = 0.2$).

5.3.2 NARX GP structure (Figure 5.3b)

For the case of the NARX GPs the experiments are separated into three, corresponding to the three different approximations methods used for uncertainty propagation in the trajectories' prediction:

1. Naive trajectory predictions using mean value

   Figure 5.5 reports the obtained RMSE for this method, over 50 iterations of the Safe Active Learning algorithm. Here, it can be observed that varying the allowed risk to enter an unsafe area has an impact on the exploration performance, as higher risk allows wider exploration, whereas lower risk requires a more conservative exploration. However, the allowed risk does not have such significant impact on the performance of the exploration as one would expect. This approach, despite using the simplest and fastest one of the approximations, still performed better than the chosen benchmark. The significant improvement over using an NX GP to learn the model can be seen in Figure 5.6.

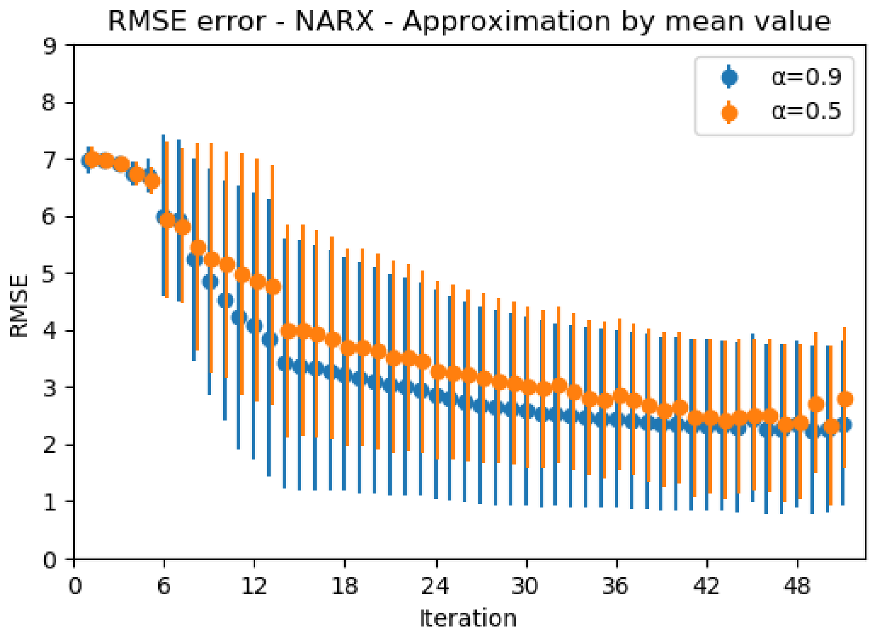


Figure 5.5: Reported RMSE accross 50 iterations of the Safe Active Learning framework using a NARX model with the Naive approximation by mean value with different risk allowances of entering the unsafe area ($\alpha = 0.9$ and $\alpha = 0.5$).

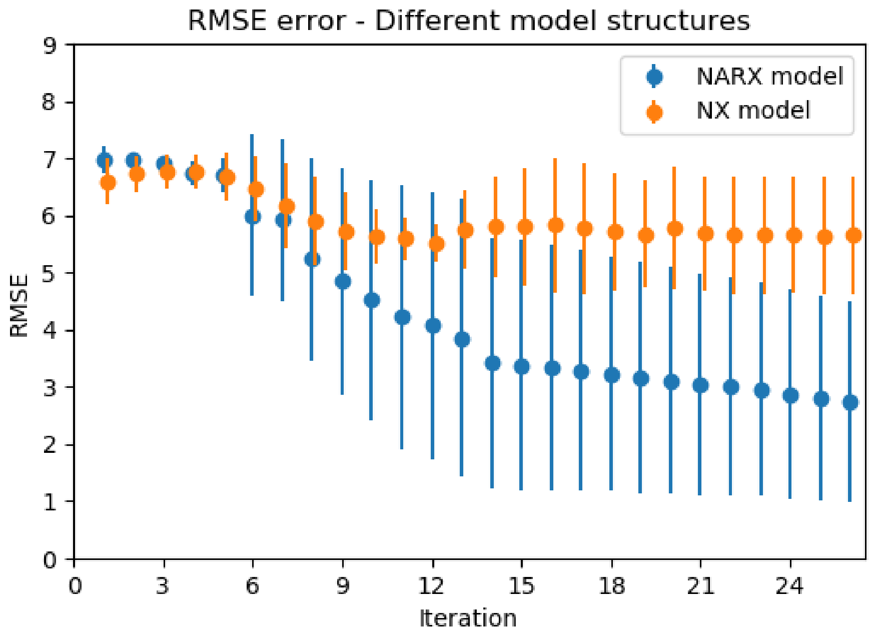


Figure 5.6: Reported RMSE error for the Safe Active Learning framework using NX GPs and NARX ones with naive approximation by mean value, across 50 iterations with allowed risk of entering the unsafe area of $\alpha = 0.9$

2. Trajectory predictions using Moment Matching

   When using Moment Matching approximations, the results of the Safe Active Learning algorithm show a similar performance to the naive approximation by mean value. A comparison between these two approximations can be seen in Figure 5.8. In this scenario, when the exploration risk is modified a similar effect to the one reported for the naive approximation can be observed (Figure 5.7).

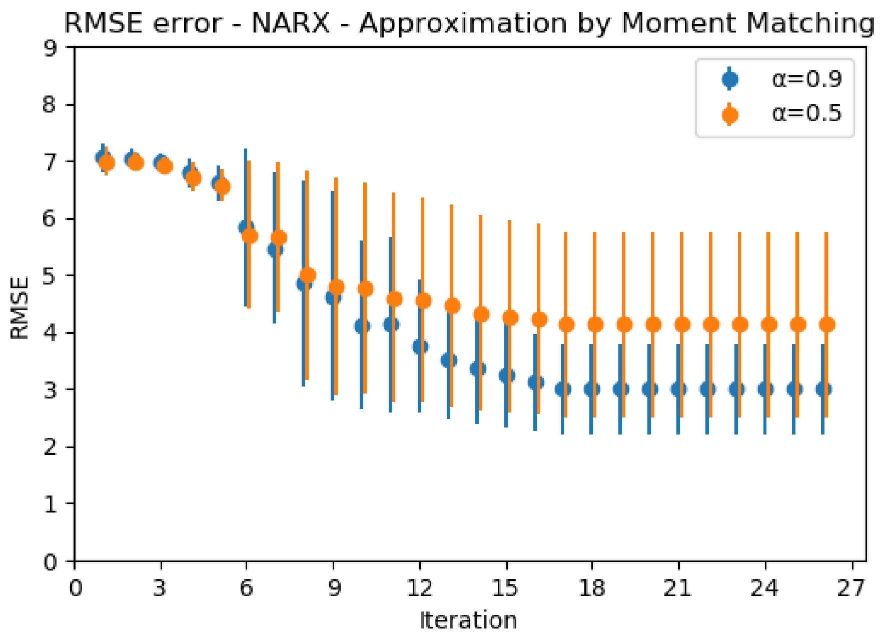


Figure 5.7: Reported RMSE across 25 iterations of the Safe Active Learning framework for NARX GPs using the Moment Matching approximation with different risk allowances of entering the unsafe area ($\alpha = 0.9$ and $\alpha = 0.5$).

3. Trajectory predictions using Monte Carlo

   Monte Carlo approximation are commonly used as ground truth when approximating a distribution. As expected, results show that this is the approximation that produces a better performance (see Figure 5.8). However, it is interesting to see that the above mentioned approximations produce a fair result with significantly less computational time.

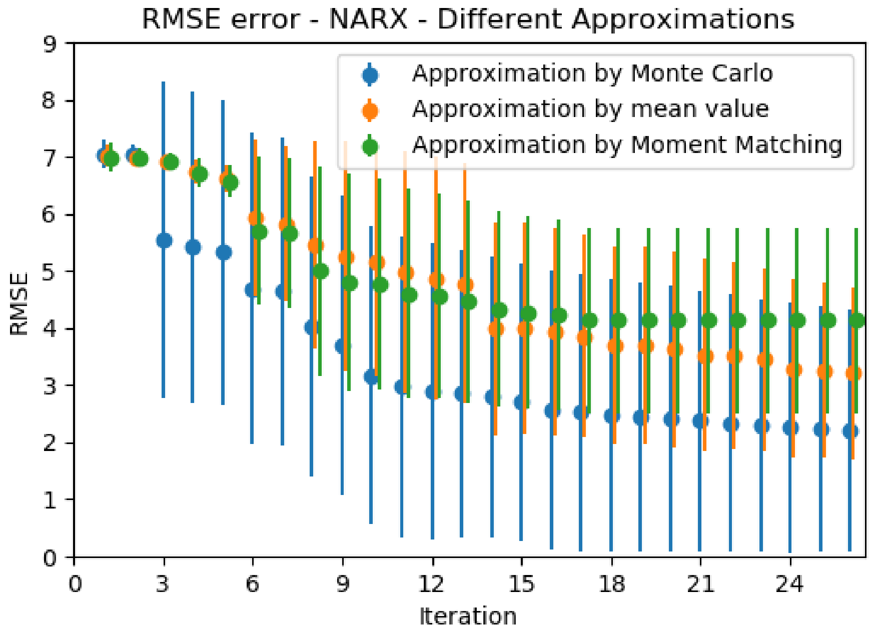
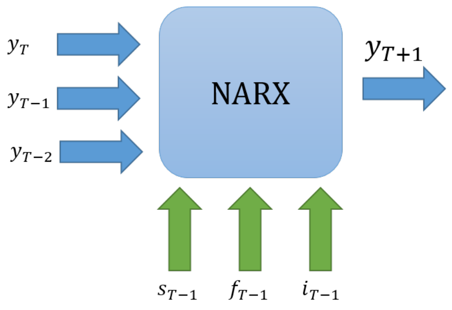


Figure 5.8: Reported RMSE across 25 iterations of the Safe Active Learning framework for NARX GPs using the Naive approximation by mean value, Moment Matching approximation, and Monte Carlo Sampling approximation (using 100 samples) with risk allowances of entering the unsafe area of ($\alpha = 0.5$).

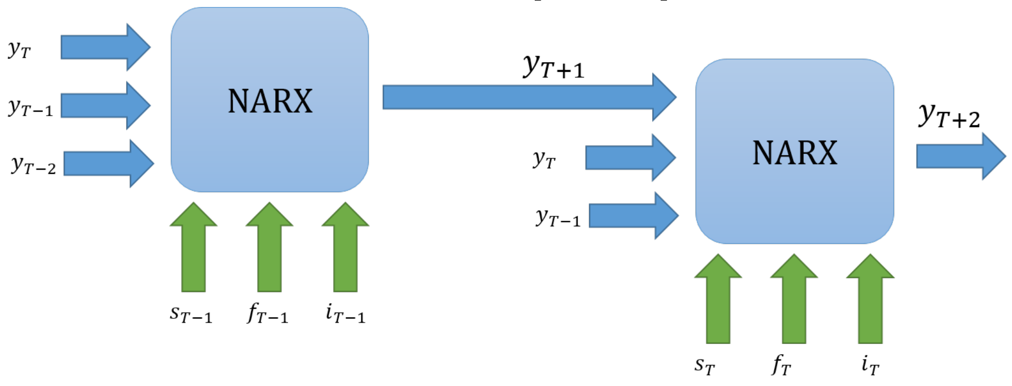
From the above presented results, one can observe that despite the used approximation method for the trajectories' predictions, changing the allowed risk probability does not have such a large impact on the exploration performance as one would expect. To analyse this phenomenon, one can start looking at the characteristics of the optimization problem that guides the exploration (Equation (4.2)). While in the work by Zimmer et al. [ZMN18], all 10 input variables were controllable and optimizable , this is not the case when the learnt model has a NARX structure.

Looking at Figure 5.2b it is clear that for this model only 3 input variables out of 6 are controllable at each step. Furthermore, when considering the case of a 3-step trajectory as the objective of the optimization problem only 3 inputs can be optimized in the whole trajectory. As depicted in Figure 5.9, to plan the complete 3-step trajectory only the inputs $s_{T+1}, f_{T+1}, i_{T+1}$ can be optimized, since the other controllable inputs correspond to past time-steps and are no longer controllable.

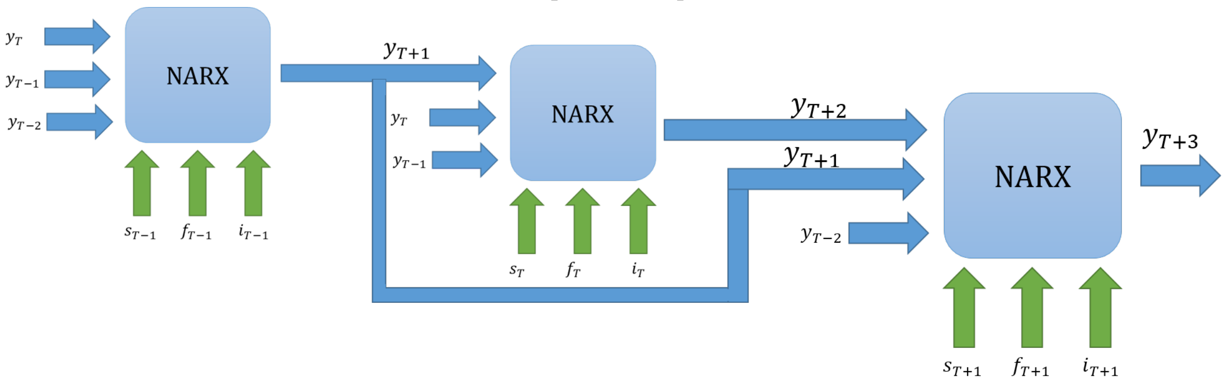
Then, when considering a 5-steps trajectory, as the one used for these experiments, only 9 input variables out of 30 can be optimized for the complete trajectory. In contrast, in the case of a 5-step trajectory with an NX model, all 50 input variables of the 5-step



(a) First step (time-step $t = T + 1$)



(b) Second step (time-step $t = T + 2$)



(c) Third step (time-step $t = T + 3$)

Figure 5.9: Trajectory prediction with NARX model (specific case of Figure 4.2) - Predictions of the High Pressure Injection System corresponding to a 3-step trajectory, using a NARX model. Here, it can be seen that all the controllable inputs tuples (s_k, f_k, i_k) but one correspond to the previous trajectory $(k \leq T)$. As a result, only the inputs $s_{T+1}, f_{T+1}, i_{T+1}$ can be optimized for the 3-step trajectory.

trajectory can be optimized. This difference plays a key role when solving the constraint optimization problem. The search space for the NARX case is smaller than the complete input space of the system, which makes it more complicated to fully explore the safe area until its frontiers with the unsafe area. And it is exactly near the frontiers of the unsafe area, where different allowed risk probability would decide differently on continuing to expand the exploration towards the unsafe area or not. Therefore, higher allowed risk probability does not imply such a big performance boost in the NARX variant of the Safe Active Learning framework, as they do in the NX case.

In addition to the above mention experiments, the following extra experiments were run (details can be found in Appendix B):

1. Test the performance of the Safe Active Learning algorithm compared to random exploration: Results in Appendix B.1 show that the Safe Active Learning algorithm outperforms the random exploration, as expected.
2. Impact of the observation noise introduced by the feedback loop: Results in Appendix B.2 show that the noise introduced by the feedback loop, characteristic of NARX models degrades the performance of the learnt GP.
3. Accuracy of the information metric approximated by different methods: Appendix B.3 shows how accurate the naive approximation by mean value, Moment Matching approximation, and Monte Carlo Sampling approximation (with different number of samples) can approximate the information metric. Monte Carlo Sampling approximation outperformed both Moment Matching and the Naive approximation by mean value only after a couple of iterations. Also, Moment Matching was shown to be a better approximation than the Naive approximation by mean value, as expected from **??**.

In summary, using the Safe Active Learning framework with a NARX GP, independent of the approximation method used to predict trajectories, outperformed the previous work by Zimmer et al. [ZMN18]. Moreover, results show that Safe Active Learning exploration obtained better and faster convergence when compared with safe random exploration. Finally, different methods were used to approximate the multiple steps ahead predictions, all resulting in similar exploration performance.

6 Discussion, Conclusions and Outlook

Discussion

In this masters' thesis, the aim was to assess the problem of Safe Active Learning for Time Series Modelling using Nonlinear Autoregressive Exogenous Gaussian Processes. Consequently, the Safe Active Learning framework proposed by Zimmer et al. [ZMN18] was extended to be able to learn Nonlinear Autoregressive Exogenous models.

The results of this study indicate that the newly designed and implemented framework for Dynamic Safe Active Learning with NARX GPs outperformed the work by Zimmer et al. It is interesting to note that even when using the simplest and fastest approximation (the Naive approximation by mean value) this Safe Active Learning Framework for NARX GP still showed a faster convergence and lower RMSE than the framework developed by Zimmer et al.

Regarding different approximations for multiple steps ahead predictions with NARX GPs, the current study found that the Monte Carlo Sampling approximation (with 100 samples) resulted in a better performance of the framework than the Naive approximation by mean value and Moment Matching. Moreover, the framework performed similarly when using Moment Matching or the Naive approximation by mean value. These results are in agreement with those obtained by the extra experiment in Appendix B.2. This experiment indicates that the better information metric approximations are obtained by Monte Carlo approximations with more than 5 samples, and that the approximation by Moment Matching was slightly more accurate than the Naive approximation by mean value. This implies that a higher accuracy can be obtained by using the most computationally complex approach (Monte Carlo Sampling), resulting in a better performance of the Safe Active Learning algorithm. If the computational complexity is to be regarded, however, a more simple approach (the Naive approximation by mean value or Moment Matching) can be employed and still obtain a better performance than the Safe Active Learning framework with NX GPs.

Contrary to expectations, this study did not find a significant difference when running the developed framework with different allowed risk probabilities. These results are likely to be related to the structure of the NARX trajectory, as discussed in Section 5.3.2.

Extra experiments in Appendix B indicate that the developed Safe Active Learning framework performs better than a safe random exploration and that the observational noise has a significant negative impact when learning a NARX model.

Conclusion

The results of this study indicate that the developed framework can be applied in the real-world to model physical systems in an efficient and safe manner. Moreover, results show that this work outperforms the original framework developed by Zimmer et al., as well as random exploration, when learning dynamic systems with an underlying NARX structure. Furthermore, results regarding the different trajectories prediction approximations indicate that good results can be obtained with the naive approximation by mean value, as well as by the moment matching approximation.

Outlook

- For this study, the real systems' NARX structure was assumed to be known. A further study could assess the problem of learning a NARX model with an unknown structure.
- In this work, different approximations are presented to predict multiple steps ahead using a NARX GP. However, the use of a multi-output GP, to be able to predict n-steps ahead in one shot, was not considered. Further studies regarding the implementation of such a model would be worthwhile.
- Contrary to expectations, this study did not find a significant difference in the exploration performance when varying the allowed risk probability. A possible explanation for this might be that the number of optimizable variables when planning a trajectory was too low. A natural progression of this work is to analyse whether increasing the number of optimizable variables shows a more favorable result. This could be achieved, for example, by increasing the number of steps of the planned trajectory or by modifying the NARX structure to include more variables from system inputs that could be optimized.
- Results show that the observational noise in the feedback loop of the model degrades the performance of the predictions, how to better handle this noise could be usefully explored in further research.

A Implementation details

The complete implementation was made in Python version 3 and the following libraries were used:

- GPyTorch [GPB+18]: used for the training, update and prediction of Gaussian Processes.
- PyTorch [PGC+17]: used for the tensor representation of the GP's inputs and outputs required by GPyTorch.
- SciPy [JOP+01]: its SLSQP minimize module was used with 25 different starts as blackbox optimization method to solve the constrained optimization problem, and its stats module was used for computing cumulative probabilities.
- NumPy [VCV11]: used for the arrays and matrices manipulation.
- Matplotlib [Hun07]: used for results plotting.

All experiments were run using 5 different random seeds, and from there mean and variances of the plots are calculated.

B Extra Experiments

B.1 Safe Active Learning vs Random Safe Exploration

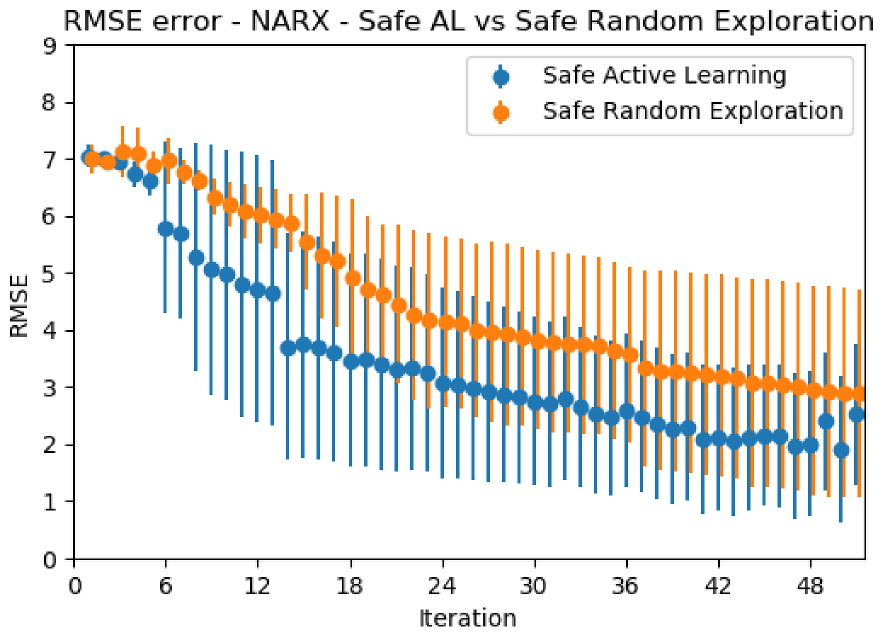
Figure B.1 shows a comparison between the performance of the developed Safe Active Learning framework and a Random Safe Exploration. Random Safe Exploration is performed by choosing random trajectories that fulfill the safety constraint. Results show that the Safe Active Learning framework performs better than the Random Safe Exploration.

B.2 Impact of observational noise in the feedback loop in NARX models

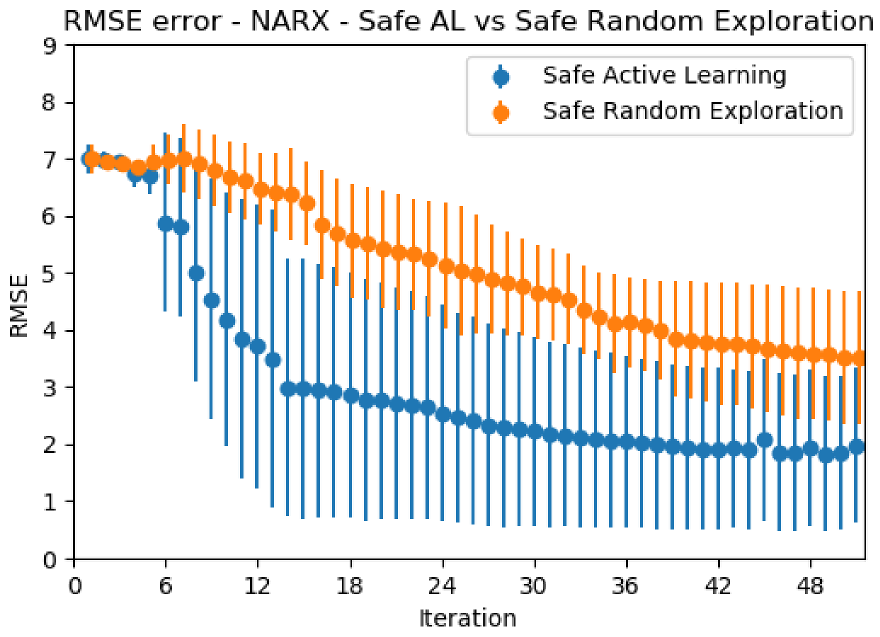
A distinctive characteristic of the NARX models is that previous outputs of the system are input to the model through a feedback loop. Since observational noise in the outputs of physical system is usually considered, in the case of NARX models, these feedback loops introduced noisy inputs to the model. Figure B.2 shows the improvement in the performance of the Safe Active Learning framework when there is no observational noise.

B.3 Information Metric Approximations

To test the accuracy of the three different chosen approximations for multi steps ahead predictions the following test was conducted: Predictions of the information metric for 5-steps trajectories were made using the Naive approximation by mean value, Moment Matching and Monte Carlo Sampling approximations. Results can be seen in Figure B.3. As expected, Moment Matching approximations outperformed the Naive approximation by mean value. Moreover, Monte Carlo approximations outperformed both of the previous mentioned methods with only a couple of samples.

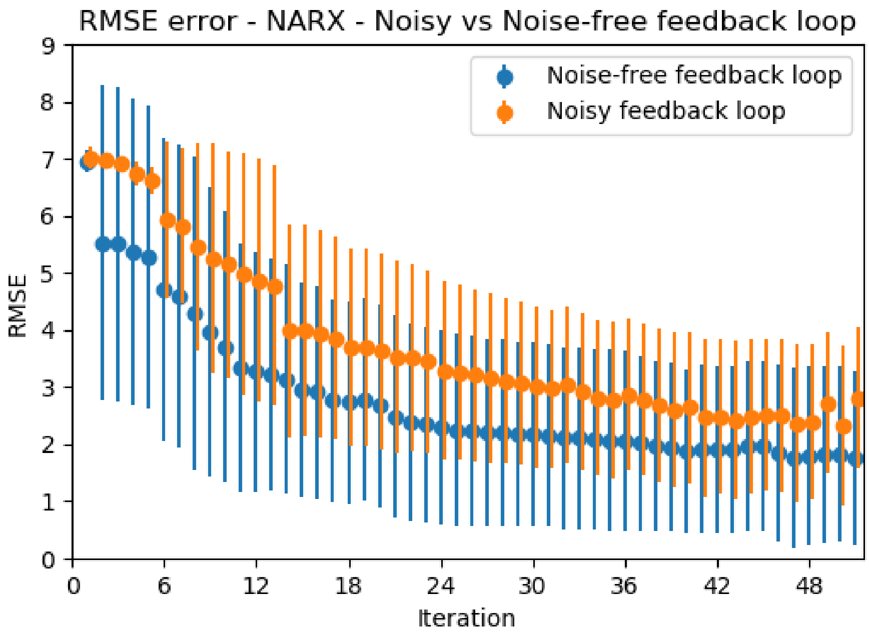


(a) Allowed risk of entering the unsafe area $\alpha = 0.5$

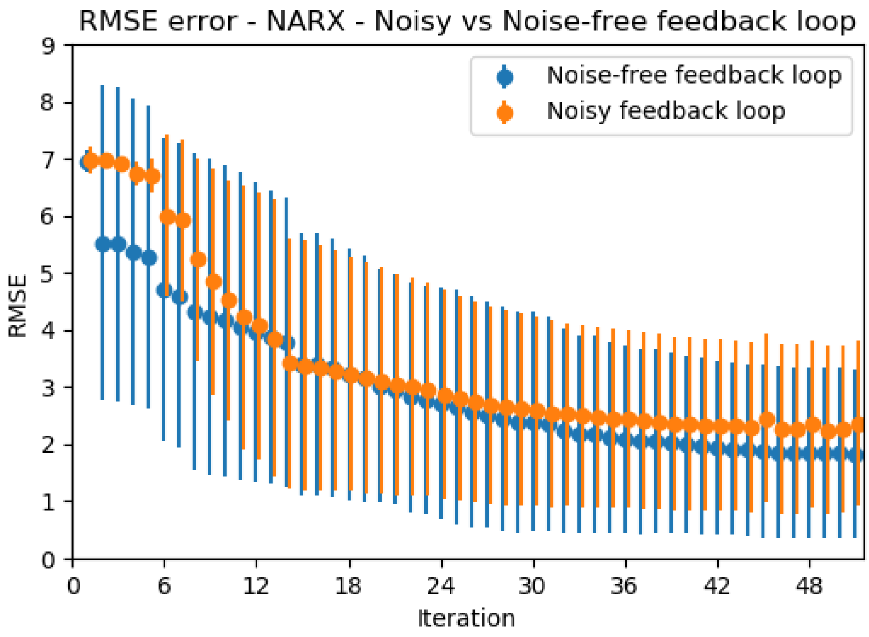


(b) Allowed risk of entering the unsafe area $\alpha = 0.9$

Figure B.1: RMSE for Safe Exploration using a NARX model - Reported RMSE for the developed Safe Active Learning framework and a Random Safe Exploration. For the Random Safe Exploration: Trajectories are chosen randomly and tested with the safety constraint, until one fulfills it, then that is the chosen trajectory of the iteration.



(a) Allowed risk of entering the unsafe area $\alpha = 0.5$



(b) Allowed risk of entering the unsafe area $\alpha = 0.9$

Figure B.2: RMSE for Safe Active Learning exploration using a NARX model with and without observational noise

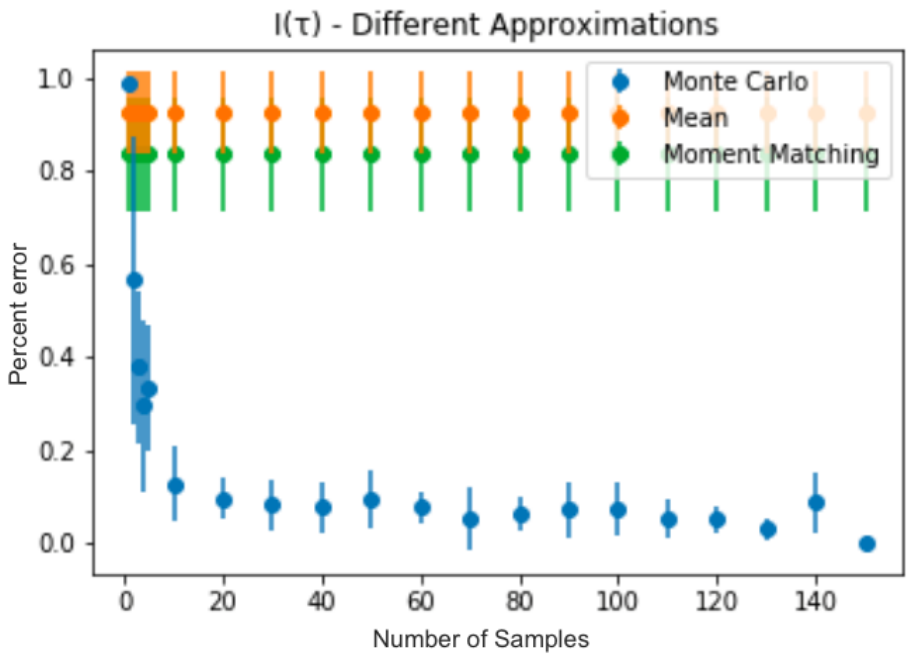


Figure B.3: Reported percent error for different approximations of $I(\tau)$ - Reported error for the information metric of 10 different 5-step trajectories using the Naive approximation by mean value, Moment Matching approximation and Monte Carlo Sampling approximation with different number of samples. These trajectories were predicted using the same NARX GP. As the metric seems to converge for the Monte Carlo approximation using 150 samples, this was used as the real value of the metric to compute the percent error (y-axis). The x-axis corresponds to the number of samples used and is only relevant for the Monte Carlo approximations.

Bibliography

[ACN10] P. Abbeel, A. Coates, A. Y. Ng. “Autonomous helicopter aerobatics through apprenticeship learning.” In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639 (cit. on p. 19).

[ACQN07] P. Abbeel, A. Coates, M. Quigley, A. Y. Ng. “An application of reinforcement learning to aerobatic helicopter flight.” In: *Advances in neural information processing systems*. 2007, pp. 1–8 (cit. on p. 17).

[AMS97] C. G. Atkeson, A. W. Moore, S. Schaal. “Locally weighted learning for control.” In: *Lazy learning*. Springer, 1997, pp. 75–113 (cit. on p. 17).

[ÅT06] B. M. Åkesson, H. T. Toivonen. “A neural network model predictive controller.” In: *Journal of Process Control* 16.9 (2006), pp. 937–946 (cit. on p. 17).

[BKS16] F. Berkenkamp, A. Krause, A. P. Schoellig. “Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics.” In: *arXiv preprint arXiv:1602.04450* (2016) (cit. on p. 19).

[BMSK16] F. Berkenkamp, R. Moriconi, A. P. Schoellig, A. Krause. “Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes.” In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 4661–4666 (cit. on p. 19).

[BSK16] F. Berkenkamp, A. P. Schoellig, A. Krause. “Safe controller optimization for quadrotors with Gaussian processes.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 491–496 (cit. on p. 19).

[CDD+17] L.-F. Cheng, G. Darnell, B. Dumitrascu, C. Chivers, M. E. Draugelis, K. Li, B. E. Engelhardt. “Sparse multi-output Gaussian processes for medical time series prediction.” In: *arXiv preprint arXiv:1703.09112* (2017) (cit. on pp. 15, 17).

[Coh94] D. A. Cohn. “Neural network exploration using optimal experiment design.” In: *Advances in neural information processing systems*. 1994, pp. 679–686 (cit. on p. 18).

[CYD+06] H. Cao, Y. Yin, D. Du, L. Lin, W. Gu, Z. Yang. “Neural-network inverse dynamic online learning control on physical exoskeleton.” In: *International Conference on Neural Information Processing*. Springer. 2006, pp. 702–710 (cit. on p. 17).

[FH12] V. V. Fedorov, P. Hackl. *Model-oriented design of experiments*. Vol. 125. Springer Science & Business Media, 2012 (cit. on p. 18).

[Gei06] P. Geibel. “Reinforcement learning for MDPs with constraints.” In: *European Conference on Machine Learning*. Springer. 2006, pp. 646–653 (cit. on p. 19).

[GF12] J. Garcia, F. Fernández. “Safe exploration of state and action spaces in reinforcement learning.” In: *Journal of Artificial Intelligence Research* 45 (2012), pp. 515–564 (cit. on p. 19).

[GF15] J. Garcıa, F. Fernández. “A comprehensive survey on safe reinforcement learning.” In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480 (cit. on p. 19).

[GH02] D. Gu, H. Hu. “Neural predictive control for a car-like mobile robot.” In: *Robotics and Autonomous Systems* 39.2 (2002), pp. 73–86 (cit. on p. 17).

[Gir04] A. Girard. “Approximate methods for propagation of uncertainty with Gaussian process models.” PhD thesis. Citeseer, 2004 (cit. on pp. 35, 38, 39, 41).

[GM05] A. Girard, R. Murray-Smith. “Gaussian processes: Prediction at a noisy input and application to iterative multiple-step ahead forecasting of time-series.” In: *Switching and learning in feedback systems*. Springer, 2005, pp. 158–184 (cit. on p. 39).

[GPB+18] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, A. G. Wilson. “GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration.” In: *Advances in Neural Information Processing Systems*. 2018 (cit. on p. 59).

[GRCM03] A. Girard, C. E. Rasmussen, J. Q. Candela, R. Murray-Smith. “Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting.” In: *Advances in neural information processing systems*. 2003, pp. 545–552 (cit. on pp. 17, 39).

[GT11] J. H. Gillulay, C. J. Tomlin. “Guaranteed safe online learning of a bounded system.” In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 2979–2984 (cit. on p. 19).

[GW05] P. Geibel, F. Wysotzki. “Risk-sensitive reinforcement learning applied to control under constraints.” In: *Journal of Artificial Intelligence Research* 24 (2005), pp. 81–108 (cit. on p. 19).

[Hun07] J. D. Hunter. “Matplotlib: A 2D graphics environment.” In: *Computing in science & engineering* 9.3 (2007), p. 90. URL: https://matplotlib.org/ (cit. on p. 59).

[JOP+01] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/ (cit. on p. 59).

[KF09] J. Ko, D. Fox. “GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models.” In: *Autonomous Robots* 27.1 (2009), pp. 75–90 (cit. on p. 35).

[KG07] A. Krause, C. Guestrin. “Nonmyopic active learning of gaussian processes: an exploration-exploitation approach.” In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 449–456 (cit. on pp. 18, 19).

[KMH+19] J. Kirschner, M. Mutn, N. Hiller, R. Ischebeck, A. Krause. “Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces.” In: *arXiv preprint arXiv:1902.03229* (2019) (cit. on p. 19).

[KMRG04] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, A. Girard. “Gaussian process model based predictive control.” In: *Proceedings of the 2004 American control conference*. Vol. 3. IEEE. 2004, pp. 2214–2219 (cit. on pp. 15, 17).

[Mac92] D. J. MacKay. “Information-based objective functions for active data selection.” In: *Neural computation* 4.4 (1992), pp. 590–604 (cit. on p. 18).

[NCD+06] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang. “Autonomous inverted helicopter flight via reinforcement learning.” In: *Experimental robotics IX*. Springer, 2006, pp. 363–372 (cit. on p. 17).

[NP11] D. Nguyen-Tuong, J. Peters. “Model learning for robot control: a survey.” In: *Cognitive processing* 12.4 (2011), pp. 319–340 (cit. on p. 17).

[NSP09] D. Nguyen-Tuong, M. Seeger, J. Peters. “Model learning with local gaussian process regression.” In: *Advanced Robotics* 23.15 (2009), pp. 2015–2034 (cit. on pp. 15, 17).

[PCK02] H. D. Patino, R. Carelli, B. R. Kuchen. “Neural networks for advanced control of robot manipulators.” In: *IEEE Transactions on Neural networks* 13.2 (2002), pp. 343–354 (cit. on p. 17).

[PGC+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer. “Automatic differentiation in PyTorch.” In: (2017). URL: https://pytorch.org/ (cit. on p. 59).

[PKPB07] C. Plagemann, K. Kersting, P. Pfaff, W. Burgard. “Heteroscedastic gaussian process regression for modeling range sensors in mobile robotics.” In: *Snowbird learning workshop*. 2007 (cit. on pp. 15, 17).

[PR11] F. J. G. Polo, F. F. Rebollo. “Safe reinforcement learning in high-risk tasks through policy improvement.” In: *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE. 2011, pp. 76–83 (cit. on p. 19).

[RAO+15] V. Rajpaul, S. Aigrain, M. A. Osborne, S. Reece, S. Roberts. “A Gaussian process framework for modelling stellar activity signals in radial velocity data.” In: *Monthly Notices of the Royal Astronomical Society* 452.3 (2015), pp. 2269–2291 (cit. on pp. 15, 17).

[Ras03] C. E. Rasmussen. “Gaussian processes in machine learning.” In: *Summer School on Machine Learning*. Springer. 2003, pp. 63–71 (cit. on pp. 22, 23).

[RC13] C. Robert, G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013 (cit. on p. 35).

[RD16] N. S. Raghavendra, P. C. Deka. “Multistep ahead groundwater level time-series forecasting using Gaussian process regression and ANFIS.” In: *Advanced Computing and Systems for Security*. Springer, 2016, pp. 289–302 (cit. on pp. 15, 17).

[ROE+13] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, S. Aigrain. “Gaussian processes for time-series modelling.” In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1984 (2013), p. 20110550 (cit. on p. 17).

[SB18] R. S. Sutton, A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 17).

[Set09] B. Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009 (cit. on p. 18).

[SGBK15] Y. Sui, A. Gotovos, J. Burdick, A. Krause. “Safe exploration for optimization with Gaussian processes.” In: *International Conference on Machine Learning*. 2015, pp. 997–1005 (cit. on p. 19).

[SKKS09] N. Srinivas, A. Krause, S. M. Kakade, M. Seeger. “Gaussian process optimization in the bandit setting: No regret and experimental design.” In: *arXiv preprint arXiv:0912.3995* (2009) (cit. on p. 19).

[SKKS12] N. Srinivas, A. Krause, S. M. Kakade, M. W. Seeger. “Information-theoretic regret bounds for gaussian process optimization in the bandit setting.” In: *IEEE Transactions on Information Theory* 58.5 (2012), pp. 3250–3265 (cit. on p. 19).

[SNE+15] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, M. Toussaint. “Safe exploration for active learning with Gaussian processes.” In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2015, pp. 133–149 (cit. on pp. 15, 19, 20).

[SWGO00] S. Seo, M. Wallat, T. Graepel, K. Obermayer. “Gaussian process regression: Active data selection and test point rejection.” In: *Mustererkennung 2000*. Springer, 2000, pp. 27–34 (cit. on p. 18).

[SX17] Y. C. Shin, C. Xu. *Intelligent systems: modeling, optimization, and control*. CRC press, 2017 (cit. on p. 17).

[SZBY18] Y. Sui, V. Zhuang, J. W. Burdick, Y. Yue. “Stagewise safe bayesian optimization with gaussian processes.” In: *arXiv preprint arXiv:1806.07555* (2018) (cit. on p. 19).

[VCV11] S. Van Der Walt, S. C. Colbert, G. Varoquaux. “The NumPy array: a structure for efficient numerical computation.” In: *Computing in Science & Engineering* 13.2 (2011), p. 22. URL: https://numpy.org/ (cit. on p. 59).

[Vin18] J. Vinogradska. “Gaussian Processes in Reinforcement Learning: Stability Analysis and Efficient Value Propagation.” PhD thesis. Technische Universität, 2018 (cit. on p. 35).

[ZMN18] C. Zimmer, M. Meister, D. Nguyen-Tuong. “Safe Active Learning for Time-Series Modeling with Gaussian Processes.” In: *Advances in Neural Information Processing Systems*. 2018, pp. 2730–2739 (cit. on pp. 3, 15, 16, 20, 29, 47, 48, 51, 54, 56, 57).

All links were last followed on October 1st, 2019.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature