SIMULATION OF APPROXIMATED GAUSSIAN PROCESS AUTOREGRESSIVE MODELS

Tadej Krivec

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Juš Kocijan, Jožef Stefan Institute, Ljubljana, Slovenia

Evaluation Board:
Prof. Dr. Đani Juričić, Chair, Jožef Stefan Institute, Ljubljana, Slovenia
Prof. Dr. Igor Škrjanc, Member, Faculty of Electrical Engineering, Ljubljana, Slovenia
Dr. Gregor Gregorčič, Member, AVL Company, Graz, Austria

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Tadej Krivec

SIMULATION OF APPROXIMATED GAUSSIAN PROCESS AUTOREGRESSIVE MODELS

Doctoral Dissertation

SIMULACIJA APROKSIMIRANIH AVTOREGRESIJSKIH MODELOV NA PODLAGI GAUSSOVSKIH PROCESOV

Doktorska disertacija

Supervisor: Prof. Dr. Juš Kocijan

Ljubljana, Slovenia, February 2023

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. Juš Kocijan for the continuous support throughout my Ph.D study and related research, for his patience and for the motivation. His guidance helped me in all the time of research and writing of this thesis. I will be forever grateful to him for introducing me to the interesting world of Gaussian processes.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Dr. Đani Juričić, Prof. Dr. Igor Škrjanc, and Dr. Gregor Gregorčič, for taking the time from their busy schedule to review my thesis. I would also like to thank the Slovenian Research Agency which financially supported my Ph.D. grant.

My sincere thanks also go to Associate Prof. Dr. Gregor Verbič at the Centre for Future Energy Networks, School of Electrical and Information Engineering, University of Sydney, who provided me with an opportunity to join his laboratory where the study in Chapter 6 was conducted. I would also like to thank the American Slovenian Education Foundation for making the research visit possible.

I thank my fellow labmates for the stimulating discussions during many coffee breaks. Last but not least, I would like to thank my family, friends, and my better half for their support throughout writing this thesis.

Abstract

This thesis presents the simulation of approximated Gaussian process autoregressive models. Gaussian process models are a Bayesian nonparametric regression method, the main advantage of which is the quantification of uncertainty in closed-form. However, the closed-form solution for the marginal likelihood results in a cubic computational complexity with respect to the data set size. An additional downside is the analytically intractable propagation of the uncertain inputs through the nonlinear covariance function, which implies that the training of the dynamical Gaussian process models cannot be obtained in closed-form.

Autoregressive models reduce the training to that of the static case. In Gaussian processes, this allows for an analytical expression of the marginal likelihood. Consequently, they allow for an effortless extension of the existing approximations of Gaussian processes, reducing the computational complexity and permitting non-Gaussian likelihoods. However, the poor computational complexity in the numerical estimation of the simulation persists.

In this thesis, we present an approximation to the simulation of Gaussian process models. We propose a unified view of the simulation for the pseudo-input-based Gaussian processes, invariant to the specific approximation up to taking a static sample from the pseudo-input posterior and the choice of the covariance function. We propose an algorithm where a single parameter controls the trade-off between the computational complexity and the accuracy. Our vectorized implementation allows for the acceleration of the unified simulation algorithm on general-purpose graphics processing units, wrapping the contemporary software frameworks specific to Gaussian processes.

Practical justification of the autoregressive Gaussian process regression is demonstrated on two case studies where the quantification of uncertainty is important. The first case study considers modeling the atmospheric variables near a nuclear power plant where our forecasts are used to predict the condition of the atmosphere during the passage of the radioactive pollution cloud. In the second case study, modeling of the electrical load and photovoltaic generation in the greater area of Sydney is presented, where the forecasts of the respective outputs are used to estimate the dynamic export limits of households to preserve the stability and robustness of the electrical grid.

Povzetek

V doktorski disertaciji predstavimo aproksimacijo avtoregresijskih simulacijskih modelov na podlagi gaussovskih procesov. Modeli gaussovskih processov so bayesovska neparametrična regresijska metoda, katere glavna prednost leži v analitični kvantifikaciji negotovosti. Njihovo slabost predstavlja učenje modelov na podlagi maksimizacije marginalne pogojne porazdelitve, ki rezultira v kubični računski kompleksnosti glede na količino podatkov. Dodaten problem predstavlja analitična nezmožnost propagiranja negotovih vhodov skozi nelinearno kovariančno funkcijo, kar pomeni, da učenje v dinamičnih modelih gaussovskih procesov ni analitično mogoče.

Avtoregresijski modeli poenostavijo učenje dinamičnih modelov na učenje statičnih sistemov. V kombinaciji z gaussovskimi procesi avtoregresivni modeli omogočajo preprosto razširitev obstoječih približkov modelov gaussovskih procesov, ki znižujejo kubično računsko zahtevnost pri učenju in omogočajo uporabo negaussovske pogojne porazdelitve opazovanih podatkov. Še vedno pa ostane problem zahtevne računske kompleksnosti pri numerični simulaciji omenjenih približkov.

V doktorski disertaciji predstavimo alternativno aproksimacijo simulacijskih modelov na podlagi gaussovskih procesov. Predlagan je enotni pogled algoritma za simulacijo približkov gaussovskih procesov na podlagi psevdo vhodov, ki je neodvisen od specifičnega približka po realizaciji statičnega vzorca s posteriorne porazdelitve psevdo vhodov in izbire kovariančne funkcije. Aproksimacija je predstavljena, kjer nastavitev enega parametra omogoča kompromis med računsko zahtevnostjo in natančnostjo algoritma. Naša vektorizirana izvedba omogoča pospeševanje enotnega simulacijskega algoritma na splošno namenskih grafičnih procesnih enotah, pri čemer so uporabljena sodobna programska orodja, značilna za gaussovske procese.

Pristop avtoregresijskih modelov gaussovskih procesov je prikazan na dveh študijah, kjer je kvantifikacija negotovosti pomembna. Prva študija obravnava modeliranje atmosferskih spremenljivk v bližini jedrske elektrarne, kjer se naše napovedi uporabljajo za napovedovanje stanja atmosfere med prehodom oblaka radioaktivnega onesnaženja. V drugi študiji predstavimo modeliranje električne obremenitve in fotovoltaične proizvodnje na širšem območju Sydneyja, kjer se napovedi ustreznih izhodov uporabljajo za oceno dinamičnih omejitev izvoza električne energije malih gospodinjstev za zagotavljanje stabilnosti in robustnosti električnega omrežja.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ADAM	...	adaptive moment estimation
ARD	...	automatic relevance determination
ASHRAE	...	American Society of Heating, Refrigerating and Air-conditioning Engineers
CIMC	...	conditionally independent Monte Carlo
CIN	...	conditionally independent naïve
CPU	...	central processing unit
DER	...	distributed energy resources
DOF	...	degrees of freedom
DSO	...	distribution network operators
ELBO	...	evidence lower bound
FCMC	...	fully correlated Monte Carlo
FCN	...	fully correlated naïve
FIC	...	fully independent conditional
FIR	...	finite impulse response
FITC	...	fully independent training conditional
GMM	...	Gaussian mixture model
GP	...	Gaussian process
GP-NARX	...	Gaussian process nonlinear autoregressive model with exogenous inputs
GP-NOE	...	Gaussian process nonlinear output-error model
GP-SSM	...	Gaussian process state-space model
GPU	...	graphics processing unit
GPGPU	...	general-purpose graphics processing unit
HGP-NARX	...	heteroskedastic Gaussian process nonlinear autoregressive model with exogenous inputs
IID	...	independently and identically distributed
IIR	...	infinite impulse response
KL	...	Kullback-Leibler
LBFGS	...	limited-memory Broyden-Fletcher-Goldfarb-Shanno
MA	...	moving average
MAP	...	maximum a posteriori
MC	...	Monte Carlo
MCMC	...	Markov Chain Monte Carlo

MLE	...	maximum likelihood estimation
MLL	...	marginal log-likelihood
MSLL	...	mean standardized log loss
NARX	...	nonlinear autoregressive model with exogenous inputs
NGD	...	natural gradient descent
NP	...	noise propagation
NPP	...	nuclear power plant
NWP	...	numerical weather prediction
OPF	...	optimal power flow
PGM	...	probabilistic graphical model
PIMC	...	pseudo independent Monte Carlo
PV	...	photovoltaic
RBF	...	radial basis function
RMSE	...	root mean squared error
SMSE	...	standardized mean squared error
SVGP	...	scalable variational Gaussian process
TCMC	...	thresholded correlated Monte Carlo
VC	...	Vapnik-Chervonenkis
VFE	...	variational free energy
WRF	...	Weather Research and Forecasting
W_2	...	2nd Wasserstein distance

Chapter 1

Introduction

1.1 Overview

In this thesis, we will be concerned with the simulation of models built from data. As the world is inherently stochastic, a sound approach is to follow the laws of probability to quantify the uncertainty present in the modeling process. In addition, the processes are rarely static and change over time. The field that is concerned with building mathematical models from data of systems changing over time, i.e. dynamical systems, is called system identification [1].

1.1.1 System identification

System identification is a multistage approach to model dynamical systems from data. It includes the following steps:

- Experiment design to gather informative data;
- Selection of the model structure;
- Choice of the criterion to fit;
- Model estimation;
- Model validation.

It is an iterative procedure that is repeated until the model requirements are satisfied. The focus of system identification are dynamical systems, which are mathematical objects in which a function defines the relationship between time and dependent variables [1]. Dynamical systems can either be expressed by a differential equation when the independent variable, i.e. the time, is continuous, or by a difference equation when the independent variable is discrete. Generally, the modeling approach can be roughly divided to:

- White-box modeling;
- Grey-box modeling;
- Black-box modeling.

White box modeling is based on first principles, e.g. we derive the mathematical model from the laws of physics. Unfortunately, the underlying laws of physics are not always well understood. Grey-box modeling, also known as semi-physical modeling, partially constructs the model from physics, and partially from experimental data [2]. Contrary

to the previous two approaches, black-box modeling assumes no prior knowledge of the physical laws or constraints governing the true system [3]. The model is constructed purely from observed data.

When the model of the dynamical system is linear and time-invariant, the solutions of the desired mathematical objects can be obtained analytically. For that reason, the analysis of linear dynamical systems is very important. Unfortunately, the problems found in practice are generally not linear. A potential solution exists in the form of linearization around a fixed set point, but this may be an inadequate approximation in practice [1]. With the ever-growing amount of information in the world, and the low cost of sensory equipment, a sound approach is to directly learn the nonlinear models from data, i.e. with a grey-box or a black-box approach. The availability of large collections of data justifies the use of nonlinear models, which would otherwise be hard to identify due to the high flexibility, especially in nonlinear dynamical systems.

However, sensors are not perfect and collect data that are corrupted with noise. Additionally, only a finite set of data are observed in terms of the number of data points, and also in terms of the features, where some might be latent [4]. The function that governs the relationship between the past and the future states can also be inherently stochastic. In that case, the future states cannot be estimated with absolute certainty even in the presence of perfect data, i.e. unlimited amounts of data that are not corrupted by noise [4].

These reasons introduce a certain amount of unpredictability into the modeling process. It is important to take this into account, especially when designing models for critical applications, e.g. applications in medicine, autonomous driving, and safety-critical systems. Therefore, statistical theory represents the core of estimating models from data.

1.1.2 Statistical modeling and inference

In grey-box and black-box modeling, the goal is to estimate the relationship between the system inputs and the system outputs, where some of the parameters are free, i.e. they cannot be defined from first principles or cannot be measured in practice. When the free parameters of the models are estimated from data to approximate the mapping from the input to the output space, the approach is called regression [5].

In a most general sense, a regression model can be described in terms of probability with a joint representation of the unknown variables and the observed data, i.e.

$$p\left(\mathcal{D}, w\right) = p\left(\mathcal{D}|w\right) p\left(w\right), \tag{1.1}$$

where $\mathcal{D}$ represents the observed data and w are the free parameters of the model. The dependencies are defined in terms of probability distributions and follow the rules of probability [6].

After the formulation of the joint model, one has to determine the free parameters that describe the observed data well, and also generalize well to previously unseen data. A possible way to determine the unknown parameters is through maximum likelihood estimation (MLE)

$$\hat{w} = \arg\max_{w} p\left(\mathcal{D}|w\right), \tag{1.2}$$

where the prediction at the previously unseen data is specified by $\mathcal{D}_* \sim p\left(\mathcal{D}|\hat{w}\right)$ [6]. The downside of MLE is that the model can overly fit the random patterns in the data. Consequently, the model does not generalize well to previously unseen data.

The solution to better generalization can be in the form of regularization, which penalizes overly flexible solutions [5], [7], [8]. A form of regularization is a Bayesian approach, which treats the free parameters of the model as random variables. A prior distribution is assumed on the unknown parameters, and the posterior distribution of the parameters given the observed data is inferred through the Bayes theorem

$$\underbrace{p\left(w|\mathcal{D},\theta\right)}_{posterior} = \frac{\overbrace{p\left(\mathcal{D}|w\right)}^{likelihood}\overbrace{p\left(w|\theta\right)}^{prior}}{\underbrace{p\left(\mathcal{D}|\theta\right)}_{evidence}} = \frac{p\left(\mathcal{D}|w\right)p\left(w|\theta\right)}{\int p\left(\mathcal{D}|w\right)p\left(w|\theta\right)dw}, \tag{1.3}$$

where θ is introduced to represent the hyperparameters, i.e. the parameters that parametrize the prior distribution over the free parameters. The prediction at the previously unseen data is obtained by averaging the likelihood over the posterior distribution of the parameters given the observed data

$$\mathcal{D}_* \sim \int p\left(\mathcal{D}_*|w\right)p\left(w|\mathcal{D},\theta\right)dw. \tag{1.4}$$

A simple example of the Bayesian approach is presented in Figure 1.1, where the goal is to infer the probability of observing heads in a coin flip. Figure 1.1a and Figure 1.1c depict our prior belief in the fairness of the coin. Figure 1.1b and Figure 1.1d represent the posterior probability in observing heads for a different number of coin tosses, where the same prior was assumed. We can see that with a large number of repetitions, the probability of observing heads is converging to the true value, otherwise the posterior is influenced by the prior. Figure 1.1a and Figure 1.1b show the results in an unbiased coin toss, whereas Figure 1.1c and Figure 1.1d show the results in a biased coin toss.

The Bayesian approach also allows for a neat separation between the aleatoric and the epistemic uncertainty, where the former represents the statistical uncertainty, i.e. the noise, whereas the latter represents the systematic uncertainty, i.e. the information that one could know theoretically but does not know in practice [9].

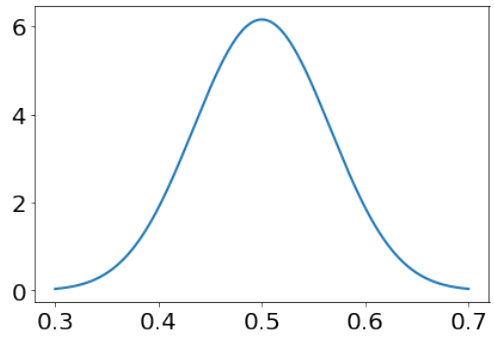
The most challenging part of the Bayesian approach is to infer the posterior distribution over the observed data. The posterior distribution often cannot be obtained in closed-form and is numerically demanding due to the high-dimensional integrals involved in Equation (1.3) and Equation (1.4). The gold standard numerical approximation to the posterior distribution is Markov Chain Monte Carlo (MCMC). MCMC are a class of algorithms that construct a Markov chain that has the stationary distribution proportional to the distribution of interest. Samples of the desired distribution are then obtained sequentially by recording states from the Markov chain [6].

In the Bayesian approach, one still has to determine the parameters that parametrize the prior distribution, i.e. the hyperparameters θ. With a relatively large collection of data, one can simply estimate the hyperparameters from the data, i.e. to optimally determine them from the evidence

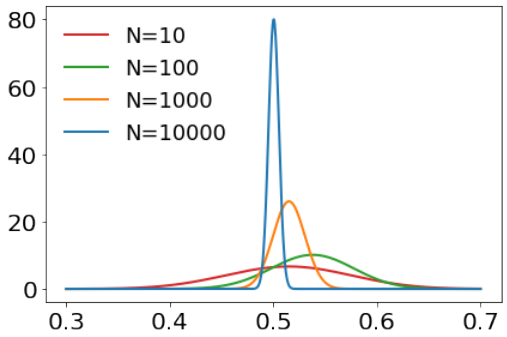
$$\hat{\theta} = \arg\max_{\theta} p\left(\mathcal{D}|\theta\right). \tag{1.5}$$

This approach is called empirical Bayes or Type II. MLE. If the data are scarce, jointly inferring the posterior distribution over the hyperparameters and the model parameters is preferred [6].

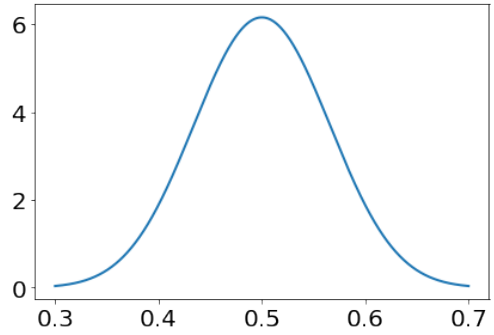
With the ever-growing increase in computational power, the list of statistical models that can model nonlinear input-output relationships is growing rapidly. One of the most popular approaches is with deep neural networks [10], which can scale to large amounts of data and are very flexible (possibly to their detriment). They require precise regularization



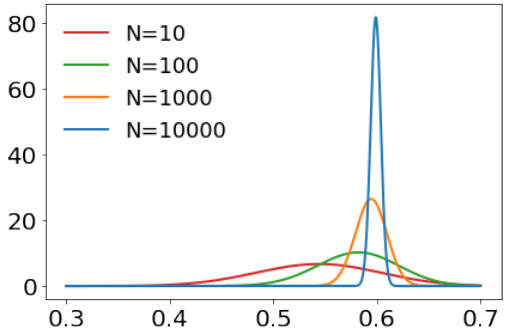
(a) Prior distribution of observing heads in a coin toss.



(b) Posterior distribution of observing heads in an unbiased coin toss.



(c) Prior distribution of observing heads in a coin toss.



(d) Posterior distribution of observing heads in a biased coin toss.

Figure 1.1: Bayesian inference to determine if the coin is biased. The left column depicts the prior distribution of observing heads in a coin toss. The right column depicts the posterior distribution for 10, 100, 1000, and 1000 realizations of a coin toss. Figures in the first row show the results for an unbiased coin ($p = 0.5$), and the figures in the second row show the results for a biased coin ($p = 0.6$).

and choice of architecture which comes at the cost of exploring this space of metaparameters through cross-validation [11]. The search can be computationally demanding, time-consuming, and intractable. Another popular method, especially in modeling of dynamical systems, are fuzzy models [12]–[14].

Quantifying (aleatoric) uncertainty is a challenge in nonlinear parametric models, partially because they are computationally very demanding, and partially because the data are represented with a finite number of basis functions [15]. For that reason, they are unsuitable for safety-critical applications, where point estimates of the parameters are not sufficient [16].

1.1.3 Gaussian process regression

Gaussian process (GP) models provide a well-calibrated predictive distribution at a model output, which can be used to quantify the model fidelity systematically [17]. This makes them suitable for problems where the model uncertainty is desired, e.g. safety-critical applications [18], robust control [19], and fault detection [20]. GPs model the data with an infinite number of basis functions [21]–[24] and belong to a class of rich probabilis-

tic models. The nonparametric structure of GPs reduces the number of metaparameters and automatically scales the model complexity with data size. The model automatically penalizes overly complex models through the evidence [25].

The prior knowledge of the system can be incorporated through the design of the covariance function, which makes GPs a great choice for grey-box modeling [26]. Generally, they can also be combined with deep neural networks, but then inherit the pathological problems of deep learning [27]. A direct approach to automatically learn the intermediate features is to stack multiple layers of GPs similar to deep neural networks, resulting in deep GPs [28]–[30].

Although the marginal and the conditional distribution can be calculated in closed-form, the algorithm results in a cubic computational complexity and quadratic complexity in memory [31]. In the last few decades, a lot of excellent work has been accomplished to reduce the computational complexity without sacrificing much of the elegant properties of GPs.

A direct approach to reduce the computational complexity is to directly speed up the algorithm. The idea is to solve the system of linear equations, i.e. $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$, through fast matrix-vector multiplication, which iteratively solves the linear system using conjugate gradients [31]–[33].

Another approach is with sparse approximations methods that suggest learning $m < n$ pseudo-points which summarize the data [34]–[36]. Some of the methods based on this approach make further assumptions about the prior that break some of the elegant core ideas behind GPs. The most popular method among the family of this type of approximations is the fully independent training conditional (FITC), where the pseudo-points and hyperparameters are jointly determined through gradient-based optimization [37].

A more recent extension of the sparse approximations directly approximates the posterior with variational inference without making further assumptions about the prior. Pseudo-points and hyperparameters are obtained by maximizing the lower bound of the true marginal log-likelihood (MLL), equivalent to minimizing the Kullback-Leibler (KL) divergence between the true and the approximated posterior. As a consequence, the distance between the true and the approximated posterior is rigorously defined.

The parameters of the free variational distribution can be retained as a model parameter, forming a scalable variational GP (SVGP) [38]–[40]. SVGP enables stochastic optimization and gives an unbiased estimation of the lower bound of the true MLL from random subsets of training data. A collapsed bound can be derived, where the optimal parameters of the free variational distribution are determined analytically, resulting in a variational free energy (VFE) approximation [41]. However, this introduces back the full correlation between the data points and the bound cannot be optimized stochastically.

There has also been an approach to join the FITC and the variational approximations under a single framework [42], and an approach to join the fast matrix-vector multiplication algorithms with pseudo-point approximations [43]. Lastly, a computationally appealing approximation to GPs is with a limited number of basis functions [44], which boils the computational problem down to Bayesian linear regression. Unfortunately, approximations based on a finite representation of basis function expansion are too restrictive in their assumptions and exhibit undesirable pathologies at test time, e.g. variance starvation [45]–[47].

Although the approximations were originally developed in the context of static systems, many ideas naturally apply also to dynamical models.

1.1.4 Dynamical Gaussian process models

A straightforward dynamical model that naturally extends from the static case is a direct functional relationship between time and output observations where the mapping is modeled by a GP. The drawback of this type of static regression is that it cannot learn nonlinear dynamics [48].

Another way to represent the nonlinear dynamics is with an infinite impulse response (IIR) model, where the dynamics are represented with a functional relationship between the current output observation and all past exogenous variables. This is practically hard to realize. Therefore, the approach is approximated by a finite impulse response (FIR) model, where the current output observation depends on a finite set of lagged exogenous variables [49].

Augmenting the input with a set of lagged output observations results in a nonlinear autoregressive model with exogenous inputs (NARX). When a GP is used to model the functional relationship, this is called a GP-NARX model. NARX model is an IIR model and can be a more powerful tool than the FIR model [49]. The problem of the FIR model is that the order, i.e. the number of the lagged exogenous inputs, has to be chosen relatively large [50]. FIR models are often overparametrized which can be a problem when combined with GPs. In GP models, the curse of dimensionality prevents the method to be applicable to problems with large input dimensionality. Therefore, NARX models are more practical in combination with GPs since the orders of the models that yield the same accuracy are much smaller when compared to the FIR approach.

The NARX model is also elegant since it reduces the model training to that of the static case. For that reason, the NARX model is easier to scale to large collections of data and to experiment over than more specialized models. NARX approach can even outperform more advanced models that are unable to handle large amounts of data, or models that define an exact generative model but have to resort to approximated inference [48].

However, there is a downside. The limitation of the NARX model is the introduction of noise in the inputs. An approach that can filter the states recurrently is the state-space model [51]–[53] (GP-SSM) or nonlinear output-error model [54] (GP-NOE). When the dynamics are linear, the problem can be efficiently solved using Kalman filtering [55]. For nonlinear dynamics, however, one has to resort to approximations or numerical inference. This results in specialized solutions that are hard to reuse, and can also fail miserably where the approximate inference is not adequate for the problem at hand.

A big challenge in dynamical GP models arises with the propagation of uncertainty through the nonlinear model which cannot be calculated in closed-form. In training, the propagation of uncertainty occurs when the inputs are noisy [56]–[58] or in probabilistic latent models [59]–[61] where the latent space is uncertain.

The propagation of uncertainty also occurs in multi-step-ahead prediction. In one-step-ahead prediction, we are interested in inferring the state of the system for a single step into the future. In this case, the inputs required to predict the state are observed and deterministic. Therefore, the posterior can be obtained in closed-form [23]. However, many applications involve predicting the future state of the system for multiple steps into the future, e.g. the state of the market in a few days, the weather forecast for the next week, or the long-term response of the system based on the known inputs. In this case, the inputs required to estimate future states are unknown and have to be determined iteratively. In GP-NARX models, the Gaussian distribution gets sequentially propagated through the nonlinear model. For that reason, future states cannot be obtained analytically [23].

A special case of the multiple step-ahead prediction is the simulation, where the prediction horizon is infinite [23]. One would wish to obtain the simulation over a certain prediction horizon for two reasons:

1. To forecast the state of the system for multiple steps into the future;
2. To validate the model.

The first point represents the main purpose of building the models from data. It is simply an estimation ot the system output over some prediction horizon. Additionally, simulation represents the validation of the model performance in an operating regime that the model was not trained for. If the simulation describes the observations well, then the assumption of the recurrent structure is justified in practice.

GP-NARX models are appealing for their simple training procedure, but the simulation has to be obtained iteratively. The prediction from the previous time step is used in the inputs for the next time step considered, continuing indefinitely or until the end of the prediction horizon. This makes the inputs uncertain and we have to, therefore, resort to approximations [62]–[66]. Approximated simulation procedures in GP-NARX models are divided to:

- Naïve simulation;
- Approximations of statistical moments;
- Numerical approximations.

Naïve simulation only propagates the mean of the prediction and, therefore, underestimates the forecasted uncertainty [23]. Approximations based on calculating statistical moments are further divided to approximations with a Taylor expansion and exact matching of statistical moments. The former is unfortunately a rough approximation, whereas the latter only exists for a few covariance functions, e.g. the squared exponential function and a linear function [23]. The obvious downside of the methods based on the approximation of statistical moments is that they ignore higher statistical moments.

With the growing computational capabilities, numerical approximations based on Monte Carlo (MC) sampling are gaining more attention. Simulation based on MC sampling does not constrain the model to a particular choice of the covariance function, and can identify higher statistical moments. The samples of the simulated trajectory can also be required by design in hierarchical pipelines where the forecasts of the GP model are used as an input to another model downstream. In that case, an end-to-end marginalization is not analytically tractable when the downstream models are nonlinear [67] and the MC approximation represents the only approach that does not constrain the downstream model to a specific family of functions.

Unfortunately, the existing work on the scaling of GP models does not translate well to sampling from the posterior trajectories in dynamical models. The MC approach scales cubically with respect to the number of predicted steps into the future [52]. MC sampling of GP-NARX trajectories extends the Gaussian distribution to all function evaluations by definition, which implies the storage of consecutive MC samples in memory. This heavily increases the computational complexity of the simulation in GP-NARX models with long prediction horizons [68].

To reduce the computational complexity in simulation, a conditional independence between the consecutive latent values given the latent values that belong to the training data is often assumed. This is a rough simplification which is mathematically imprecise, albeit fast [69]. The recurrent structure of the simulation also implies a dependency between the output samples and the inputs. For that reason, the aforementioned assumption results in a biased estimate of the simulated trajectory.

1.2 Hypotheses

The hypotheses of the thesis are the following:

- H_1: MC simulation of GP-NARX models, which treats the consecutive latent values in simulation independently given the latent values that belong to the training data, produces overly noisy simulation samples and is only a rough approximation to the true fully correlated MC simulation;
- H_2: The demanding computational requirements of the fully correlated GP-NARX simulation can be reduced with sparse approximated GP-NARX models;
- H_3: Pseudo-points affect the simulation. They introduce the opportunity for a new approximation in the simulation of GP-NARX models;
- H_4: An approximation to the fully correlated MC simulation algorithm can be derived for sparse approximated GP-NARX models that can reduce the computational requirements when compared to the ground truth and improve the estimation of the latent response of the MC simulation that treats the consecutive latent values independently.

1.3 Thesis Outline

In Chapter 2, we present an introduction to GP regression. We consider an autoregressive approach to model dynamical systems and describe the simulation of autoregressive GP models. We start with a simple Bayesian linear regression and generalize the approach to nonlinear systems. We consider Bayesian regression in the limit when the number of basis functions goes to infinity and demonstrate how this corresponds to a GP. GPs are then introduced through the function space view and extended with the autoregressive model. An approach to learn the hyperparameters from data is described. To build the intuition behind the simulation of GP-NARX models, we start with a static example, presenting a batch and a sequential algorithm for sampling from an infinitely-dimensional latent posterior. We then define the ground truth numerical simulation of autoregressive GP models through extending the static example to dynamical systems. Existing MC approximations to the simulation of GP-NARX models are defined in relation to the ground truth. Lastly, we demonstrate the autoregressive GP modeling approach on an illustrative example.

In Chapter 3, we introduce sparse approximations of GP models, which reduce the demanding computational complexity through the concept of pseudo-points. The MLLs, predictive posteriors, and pseudo-point posteriors are defined for the sparse models that are frequently used in the GP and engineering literature. An empirical comparison of the prediction of different sparse approximated GP models is considered on a static illustrative example. Lastly, the prediction of different sparse approximated GP-NARX models for modeling dynamical systems is compared on 10 chaotic time-series.

In Chapter 4, a unified algorithm for the simulation of autoregressive GP models is presented which is invariant to the GP approximation up to taking a sample from the pseudo-point posterior. A new approximation for the simulation of autoregressive GP models based on pseudo-inputs is considered which does not increase the dimensionality of the latent space. An approach to simulation is described which introduces a user-defined parameter that can trade-off between the computational demands and the accuracy of the estimated latent response. The approximations of the simulation are validated on sequential sampling of a static function, as well as for the simulation of an illustrative dynamical

system. Hardware acceleration of the simulation on general-purpose graphics processing unit is considered. Lastly, the simulation is validated on two large data set benchmarks, which was previously not possible on a personal computer due to the computational requirements.

In Chapter 5, we present a case study where the forecasts of local weather variables around a nuclear power plant are considered. The forecasts serve as an input to the model of particle dispersion which aims to describe the condition of the atmosphere during the passage of the radioactive pollution cloud. The proposed simulation approximation is validated on a large test data set which is reasonable in the atmospheric sciences so that the data contain multiple seasons with different weather patterns. We compare the proposed approach to the numerical weather prediction model that was previously used for this purpose.

Chapter 5 presents a hybrid model, where the covariance function was selected with a black-box approach. In Chapter 6, a case study considering the model of the electrical load and the photovoltaic (PV) generation in the greater Sydney area is introduced, where the covariance function is determined from the structures present in the data, i.e. we follow a grey-box approach. The heteroskedastic likelihood in the PV generation model, combined with the unconstrained exploration of the covariance function design for both desired quantities, demonstrates how the numerical approximation to the simulation is essential when the GP-NARX models become more complex. The model is validated on different days of the week and different seasons to the persistent and moving average (MA) baselines. A comparison of the electrical load and PV generation forecast is presented where the observations are available in real-time or with a 24-hour delay, to justify the need of smart meters in New South Wales.

Chapter 7 presents the concluding remarks. The hypotheses, results, and future work are considered in retrospect.

1.4 Contributions

The contributions of the thesis are the following:

- Summary of the approximations to the simulation of autoregressive GP models in relation to the ground truth;
- Empirical comparison of different sparse GP approximations for autoregressive prediction. The work was previously published in [69];
- A unified algorithm of the simulation of autoregressive GP models, invariant to the pseudo-point approximation of GPs up to taking a sample from the pseudo-point posterior. The algorithm allows a compact software implementation for different simulation approximations and multiple pseudo-point approximations to GP models;
- Pseudo-independent MC simulation. The approximation does not increase the latent dimensionality with increased prediction horizon. However, the latent uncertainty is still reduced in the posterior and the estimated latent response is closer to the ground truth than the latent response estimated from the simulation that treats the consecutive latent samples independently;
- Thresholded-correlated MC simulation. A user-defined parameter trades-off between the accuracy of the estimated latent response and the computational demands. Pseudo-independent MC simulation and the fully correlated MC simulation are considered as a special case of the approximation;

- Vectorized implementation of the simulation which computes individual MC samples in parallel. Consequently, hardware acceleration on general-purpose graphics processing units can be utilized. A part of the work was previously published in [69];
- Validation of the proposed simulation algorithms on two large benchmarks for modeling dynamical systems. Correlated MC simulation could previously not be obtained due to the computational demands. A part of the work was previously published in [69];
- The improvement of the numerical weather prediction forecast for the local weather variables in the vicinity of the nuclear power plant Krško. Validation of the correlated MC simulation was previously not possible due to the large data set. A part of the work was previously published in [67];
- Composite GP models of electrical load and PV generation, combining a timeseries and NARX models to describe the structural patterns in the observed data;
- Design of a specialized covariance function for modeling the PV generation. Simulation of the composite GP model with heteroskedastic likelihood;
- Open-source toolbox for approximated autoregressive GP models. The toolbox wraps contemporary software frameworks and allows hardware acceleration using general-purpose graphics processing units.

Chapter 2

Autoregressive Gaussian Process Regression

This chapter starts with an introduction to the Bayesian modeling approach and quantification of uncertainty. The idea is demonstrated through linear regression and then gradually extended for nonlinear systems through basis expansion, and lastly to the nonparametric model that is the Gaussian process (GP). We will demonstrate how it is possible to extend the nonparametric modeling of static problems for nonlinear dynamical systems.

We then continue with the analytically intractable simulation in autoregressive GP models. The approximations for the numerical simulation of the autoregressive GP models are presented as a special case of the fully correlated approach that is considered the ground truth. The computational complexity and memory requirements of the aforementioned approximations are defined.

At the end of the chapter, we demonstrate the use of the autoregressive GP modeling approach on an illustrative dynamical system.

2.1 Bayesian Regression

In this thesis, we will consider a data-driven approach to modeling. An important first step is to gather informative data [1]. After the data are obtained, the fundamental problem we are trying to solve is the estimation of the functional relationship between the independent and dependent variables, i.e. between the input and the output data. This approach is called regression [5].

In practice, one can hardly ever determine the mapping with certainty. The uncertainty can arise from multiple sources. For example, the nature of the process we are trying to model can be inherently stochastic. The observed data are also often corrupted by noise. Lastly, the data sample collected is only a finite sample from the population in the sense that:

- The number of data points we can collect is finite;
- Not all predictors that govern the true process are measured, i.e. they are latent.

These are just some of the potential sources of uncertainty. More generally, the uncertainty can be split into two categories:

- Aleatoric uncertainty;
- Epistemic uncertainty.

Aleatoric uncertainty represents the uncertainty due to the randomness in the change of measurements each time the data is collected under the same experiment, i.e. the measurement noise. Epistemic uncertainty represents the systematic uncertainty due to things one could in principle know but does not in practice, i.e. the observed data is finite [9].

Separating and quantifying these two sources of uncertainty is important, especially in safety-critical applications, e.g. medical applications [70]. One can also use the information of the epistemic uncertainty as a piece of important information where the model can be improved. For example, the epistemic uncertainty can identify where predictions over the input domain are uncertain and can help guide the collection of new experimental data [71]–[73]. The uncertainty can also be used in surrogate models to design optimization algorithms that trade-off between the exploration and the exploitation of the input domain to find a minimum of a function [74]–[76].

To successfully separate the aleatoric and the epistemic uncertainty and, therefore, model the uncertainty rigorously, one has to follow the rules of probability [77].

2.1.1 Bayesian approach

The Bayesian approach is the use of probability theory to build models from data [6]. It is a systematic approach, with three key steps:

1. Setting up a joint probability model for all observable and unobservable quantities;
2. Calculating the conditional probability of the unobserved quantities of interest, given the observed data;
3. Evaluating the model and iterating from 1-3.

In the first step, we define a joint probability model that could potentially generate the observed data. The knowledge about the underlying process is ideally incorporated into the model definition. Importantly, the samples from the assumed joint distribution should reflect our prior knowledge and be somewhat consistent with the observed data.

The second step involves the Bayes theorem to calculate the posterior distribution of the quantity of interest given the observed data. The theorem was discovered by the Reverend Thomas Bayes published as "An Essay towards solving a Problem in the Doctrine of Chances", and then independently discovered by Pierre Simon Laplace [78], who gave it its modern mathematical form and scientific application [79]. It is defined by

$$\overbrace{p\left(w|y,\theta,\mathcal{M}\right)}^{\text{posterior distribution}} = \frac{\int \overbrace{p\left(y|w,f,\mathcal{M}\right)}^{\text{likelihood}} \overbrace{p\left(w,f|\theta,\mathcal{M}\right)}^{\text{prior distribution}} df}{\underbrace{p\left(y|\theta,\mathcal{M}\right)}_{\text{marginal likelihood}}}, \tag{2.1}$$

where y represents the observed data, f the latent quantity that is not of interest, and w the latent quantity of interest. Variable θ in Equation (2.1) represents a fixed parameter of the prior. The conditioning on the model $\mathcal{M}$ takes into account all design choices on a meta level. The second step also involves the marginalization of the unobserved quantities

that are not of interest. This can be seen in Equation (2.1), where the unobserved quantity, i.e. f, is integrated out.

The Bayes theorem in Equation (2.1) consists of 4 important parts. *Prior distribution* defines the space of potential hypotheses for the problem at hand, possibly an infinite amount of them. *Likelihood* then weights the prior hypothesis based on observed data, giving more weight to hypotheses that are consistent with the observed data. *Posterior distribution* then simply represents a weighted average of our prior beliefs given the observed data. *Marginal likelihood* is constant with respect to latent variables and serves as a normalizing constant. The marginal likelihood, also known as the *evidence* or *marginal distribution*, can be interpreted as an expectation over the likelihood distribution with respect to the prior distribution [6].

The marginal likelihood is particularly interesting in model selection and serves as an automatic Occam's razor, preferring simple, but not overly simple models [25]. An illustration is presented in Figure 2.1. A k-th realization of the data is denoted by $\tilde{y}^{(k)}$. If the space of the hypothesis is simple, then the marginal distribution is relatively large in this domain, since simple models concentrate their probability density function, or probability mass function, around a limited number of data sets. An example of a similar space is presented by the blue curve in Figure 2.1. On the contrary, a complex space of hypothesis generates a large range of possible data sets and spreads the probability density function over the inputs domain. This space is presented by the green curve in Figure 2.1.

We can see that if the observed data are simple, e.g. the red line denoted by $\tilde{y}^{(1)}$ in Figure 2.1, then the marginal likelihood given that data is greater than the marginal likelihood of more complex models, i.e.

$$p(\tilde{y}^{(1)}|\theta_1, \mathcal{M}_1) > p(\tilde{y}^{(1)}|\theta_2, \mathcal{M}_2) > p(\tilde{y}^{(1)}|\theta_3, \mathcal{M}_3). \tag{2.2}$$

Similarly for data that are more complex, e.g. the red line denoted by $\tilde{y}^{(2)}$ in Figure 2.1, the marginal likelihoods are ordered as

$$p(\tilde{y}^{(2)}|\theta_2, \mathcal{M}_2) > p(\tilde{y}^{(2)}|\theta_3, \mathcal{M}_3) > p(\tilde{y}^{(2)}|\theta_1, \mathcal{M}_1). \tag{2.3}$$

If the generating process is complex, simple hypotheses are not adequate since they by definition cannot generate complex data sets, and their probability of correctly explaining the data is small.

A good approach to start modeling the data is with simple models and gradually building more complexity into them. Simple models are easier to interpret and can serve as a denominator for relative metrics when considering advanced methods. In the next section, we will start off with a simple model, i.e. the Bayesian linear regression [5]. It will help us build intuition and notation for more complex extensions of the linear framework that will be introduced later in the thesis.

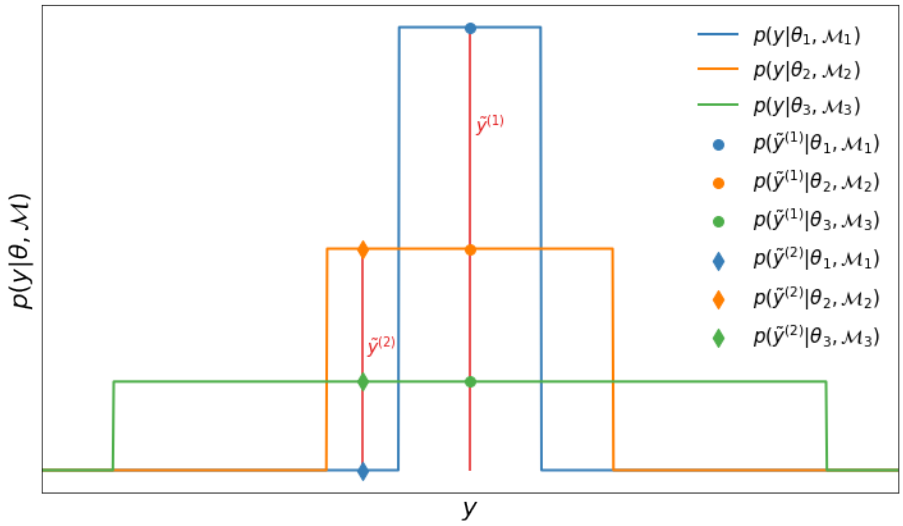


Figure 2.1: An illustration of the automatic Occam's razor in the Bayesian approach. The complexity of the model is increased from $\mathcal{M}_1$ to $\mathcal{M}_3$. The area under a single curve integrates to 1 since this is a distribution over all possible data sets. Two realizations of the data are depicted with a red line and denoted by $\tilde{y}^{(1)}$ and $\tilde{y}^{(2)}$. The figure is reproduced from [25], ch. 28.

2.1.2 Bayesian linear regression

Let us say that we observed some data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, where the rows of $\mathbf{X}$ represent the observed inputs and the elements of $\mathbf{y}$ represent the observed outputs. We assume a linear model

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}, \tag{2.4}$$

where $\mathbf{y} \in \mathbb{R}^{n\times 1}, \mathbf{X} \in \mathbb{R}^{n\times p}$, and $\mathbf{w} \in \mathbb{R}^{p\times 1}$. All vectors are assumed to be column vectors. Vector of weights $\mathbf{w}$ represents the free parameters of the model. We assume that the noise $\boldsymbol{\epsilon}$ is independently and identically distributed (IID) and follows a multivariate Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}\sigma_n^2)$, where $\mathbf{I} \in \mathbb{R}^{n\times n}$. For simplicity let us assume that the noise variance σ_n^2 is known. Let us start off with a standard approach to Bayesian linear regression, where we seek a posterior distribution of the weights given the observed data

$$p\left(\mathbf{w}|\mathbf{y}, \mathbf{X}, \sigma_n^2\right) = \frac{p\left(\mathbf{y}|\mathbf{w}, \mathbf{X}, \sigma_n^2\right) p\left(\mathbf{w}\right)}{p\left(\mathbf{y}|\mathbf{X}, \sigma_n^2\right)}. \tag{2.5}$$

Let us introduce a vector of latent function values

$$\mathbf{f} = \mathbf{X}\mathbf{w}, \tag{2.6}$$

where each value f_i corresponds to a pair of observations $(\mathbf{x}_i, y_i)$. Vector $\mathbf{x}_i^T \in \mathbb{R}^{1\times p}$ represents the i-th row of the design matrix $\mathbf{X}$ such that

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}. \tag{2.7}$$

Since all vectors are assumed to be column vectors, the i-th row of $\mathbf{X}$ is $\mathbf{x}_i^T$, the vector transpose of $\mathbf{x}_i$ [5]. The latent function values $\mathbf{f}$ are corrupted with noise, i.e. $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{2.8}$$

The joint posterior of the vector of latent function values and the weights is defined by

$$p\left(\mathbf{f}, \mathbf{w} | \mathbf{y}, \mathbf{X}, \sigma_n^2\right) = \frac{p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right) p\left(\mathbf{f}|\mathbf{w}, \mathbf{X}\right) p\left(\mathbf{w}\right)}{p\left(\mathbf{y}|\mathbf{X}, \sigma_n^2\right)}. \tag{2.9}$$

Note that the latent values given the weights are obtained by a simple linear transformation defined by Equation (2.6). In probabilistic terms, this linear transformation can be described by a Dirac distribution

$$p\left(\mathbf{f}|\mathbf{w}, \mathbf{X}\right) = \delta\left(\mathbf{f} - \mathbf{X}\mathbf{w}\right). \tag{2.10}$$

The definition of the Dirac distribution can be found in the Appendix by Equation (A.6). If the prior distribution over the weights is specified by

$$p\left(\mathbf{w}\right) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \mathbf{I}\right) \tag{2.11}$$

then

$$p\left(\mathbf{f}|\mathbf{X}\right) = \int p\left(\mathbf{f}|\mathbf{w}, \mathbf{X}\right) p\left(\mathbf{w}\right) d\mathbf{f} = \int \delta\left(\mathbf{f} - \mathbf{X}\mathbf{w}\right) p\left(\mathbf{w}\right) d\mathbf{f} = \mathcal{N}\left(\mathbf{f}|\mathbf{0}, \mathbf{X}\mathbf{X}^T\right), \tag{2.12}$$

where the integral is defined in the Appendix by Equation (A.7). The posterior of the latent function values given the observed data is then defined by

$$p\left(\mathbf{f}|\mathbf{y}, \mathbf{X}, \sigma_n^2\right) = \frac{p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right) p\left(\mathbf{f}|\mathbf{X}\right)}{p\left(\mathbf{y}|\mathbf{X}, \sigma_n^2\right)}, \tag{2.13}$$

where $p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \mathbf{I}\sigma_n^2\right)$. The marginal likelihood is defined by

$$p\left(\mathbf{y}|\mathbf{X}, \sigma_n^2\right) = \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \mathbf{X}\mathbf{X}^T + \mathbf{I}\sigma_n^2\right). \tag{2.14}$$

In general, we are interested in the prediction at the unobserved inputs. The posterior predictive distribution at the unobserved inputs $\mathbf{X}_*$ is defined by

$$p\left(\mathbf{f}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_*, \sigma_n^2\right) = \frac{\int p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right) p\left(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*\right) d\mathbf{f}}{p\left(\mathbf{y}|\mathbf{X}, \sigma_n^2\right)} \tag{2.15}$$

and similarly as in Equation (2.13), a closed-form, i.e. an analytic, solution exists. A simpler way to obtain the prediction is to consider a joint Gaussian model

$$p\left(\mathbf{y}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*, \sigma_n^2\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2 & \mathbf{K}_{f,*} \\ \mathbf{K}_{*,f} & \mathbf{K}_{*,*} \end{bmatrix}\right), \tag{2.16}$$

where $\mathbf{K}_{f,f} = \mathbf{X}\mathbf{X}^T$, $\mathbf{K}_{*,f} = \mathbf{X}_*\mathbf{X}^T$, and $\mathbf{K}_{*,*} = \mathbf{X}_*\mathbf{X}_*^T$. The predictive posterior at the unobserved inputs is then specified by

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{y},&\mathbf{X},\mathbf{X}_*,\sigma_n^2) = \\ &= \mathcal{N}\left(\mathbf{f}_*|\mathbf{K}_{*,f}\left(\mathbf{K}_{f,f}+\mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}, \mathbf{K}_{*,*}-\mathbf{K}_{*,f}\left(\mathbf{K}_{f,f}+\mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f,*}\right), \end{aligned} \tag{2.17}$$

where the solution is simply obtained by conditioning on a joint Gaussian distribution defined by Equation (2.16). The identity of conditioning on a joint Gaussian distribution can be found in the Appendix with Equation (A.4).

A block diagram of the regression model is presented in Figure 2.2a. The quantities indexed from 1 to n represent the data that belong to the training data set, i.e. we have observed the output y_i at the corresponding input x_i. Block diagrams are a great way to represent the model in terms of the input/output mapping. Unfortunately, they cannot depict a specific probabilistic relationship between the variables. It is also difficult to determine which variables are observed, and which are latent.

Probabilistic graphical models (PGMs) are a more general visualization tool in Bayesian modeling literature [7]. A PGM for a Bayesian linear regression is shown in Figure 2.2b. In PGMs, the probabilistic relationships between the variables are rigorously defined. However, PGMs might be more complex to read and are more abstract. Throughout the thesis, we will use both. Block diagrams for their simple and compact representation of the input/output mapping, and PGMs for the precise definition of the probabilistic relationships between the variables.

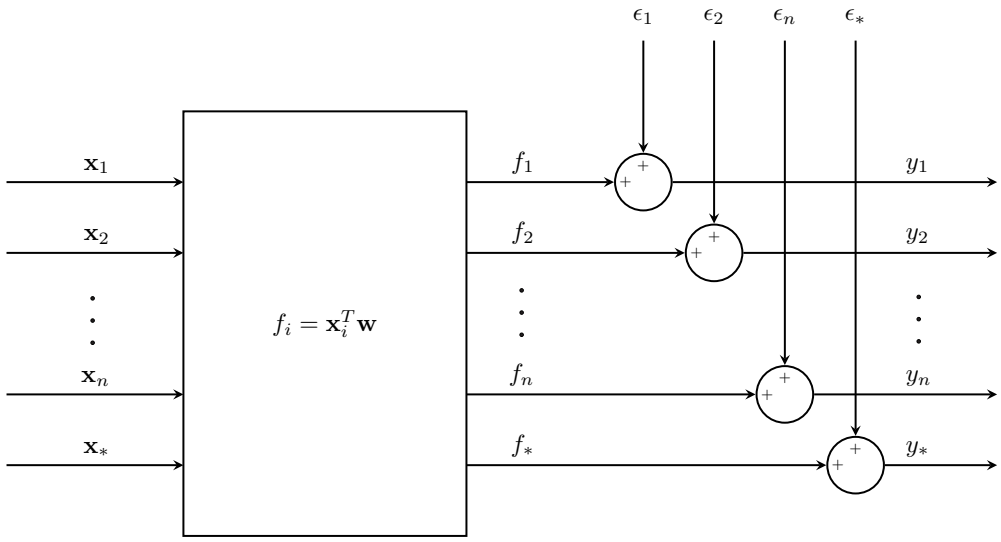
Figure 2.3 shows an example of Bayesian linear regression, where 15 data points were generated by

$$\mathbf{y} = \mathbf{x}w + \mathcal{N}\left(\boldsymbol{\epsilon}|\mathbf{0}, 0.3^2\mathbf{I}\right). \tag{2.18}$$

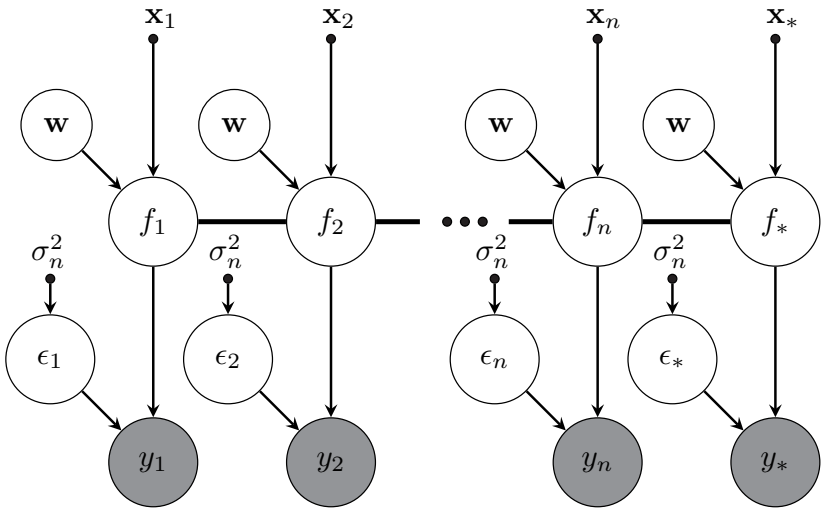
The weight parameter was defined as $w = 0.5$. The observed data are shown with orange crosses. The inputs $\mathbf{x}$ were placed uniformly on the interval $[-3, 3]$. If we look closely at the prior distribution depicted in Figure 2.3, we can see that there is no uncertainty at the input $x = 0$, i.e. the assumed intercept is 0.

The prior distribution over the latent function values, induced by the prior distribution over the weights $p(w) \sim \mathcal{N}(w|0, 1)$, is transformed to the posterior distribution of the latent function values given the noisy observations $\mathbf{y}$. The posterior uncertainty interval is reduced compared to the prior. This is expected, since many of the latent function values, assumed by the prior distribution, are not consistent with the data.

One can quickly see that Bayesian linear regression works well only when the underlying process that generated the data is linear. Unfortunately, many problems are nonlinear and linear assumptions are too restrictive. In the next section, we will present an extension of the linear framework that still preserves the closed-form solutions.

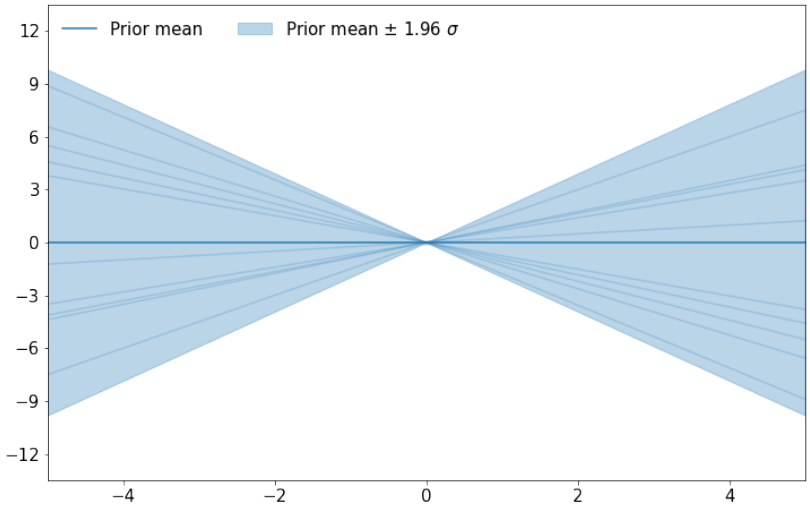


(a) Block diagram of a linear regression model.

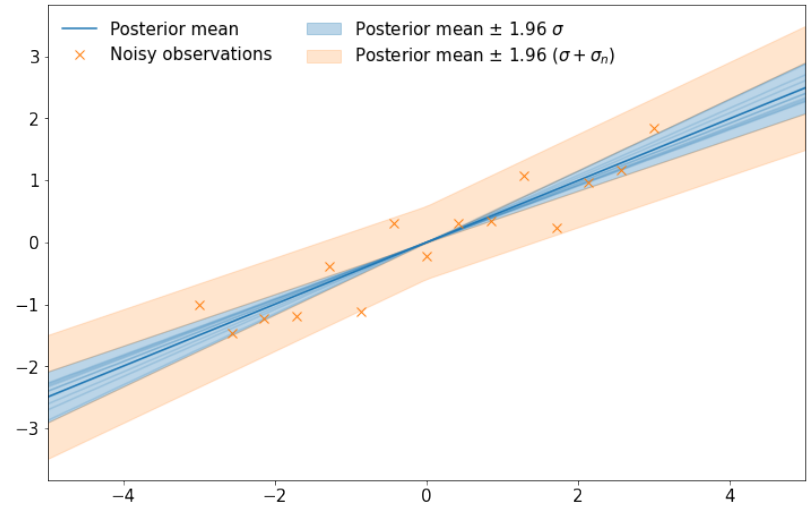


(b) Probabilistic graphical model (PGM) of a linear regression model.

Figure 2.2: Dual graphical representation of a Bayesian linear regression. For linear regression, the function evaluated at the input $\mathbf{x}_i$ is defined by $f(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{w}$. The output y_i is simply the latent value f_i corrupted with noise ϵ_i. The shaded circles in the PGM represent the observed random variables, unshaded circles represent the latent random variables, and solid black circles represent deterministic variables. In linear regression, all latent variables f are fully connected. This is represented by a thick black line. Note that this is not a Markov structure but denotes a full joint Gaussian distribution, i.e. a latent value is connected to all other latent values.



(a) Prior in Bayesian linear regression.



(b) Posterior in Bayesian linear regression.

Figure 2.3: Figure 2.3a represents the prior over the latent function values $p(\mathbf{f}_*|\mathbf{X}_*)$ and some corresponding draws of the latent function. Figure 2.3b represents the posterior over the latent function values $p\left(\mathbf{f}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_*, \sigma_n^2\right)$ and some corresponding draws of the latent function. Solid blue region represents the 95% uncertainty interval, i.e. ± 1.96 standard deviation, where σ^2 represents the latent variance. Orange solid region represents the 95% uncertainty interval that includes the likelihood noise variance σ_n^2.

2.1.3 Bayesian nonlinear regression

An elegant way to preserve the closed-form solution of the linear framework is to expand the input with nonlinear transformations of the input matrix. Nonlinear functions that expand the input space are called basis functions. The input matrix gets transformed into a higher dimensional space where the problem can be modeled by a linear function. The noisy output observations and the weights remain linearly dependent and the math stays the same as in the case of linear regression. We again assume a linear model of the form

$$\mathbf{y} = \Phi(\mathbf{X})\,\mathbf{w} + \boldsymbol{\epsilon}, \tag{2.19}$$

where $\mathbf{f} = \Phi(\mathbf{X})\,\mathbf{w}$. $\Phi = \{\phi_1(\cdot), \phi_2(\cdot), \ldots, \phi_k(\cdot)\}$ represents a set of k nonlinear transformations, i.e. basis functions, and $\Phi(\mathbf{X})$ is the matrix of the expanded input matrix $\mathbf{X}$ with a set of basis functions Φ such that

$$\Phi(\mathbf{X}) = \begin{bmatrix} \phi_1\left(\mathbf{x}_1^T\right) & \phi_2\left(\mathbf{x}_1^T\right) & \ldots & \phi_k\left(\mathbf{x}_1^T\right) \\ \phi_1\left(\mathbf{x}_2^T\right) & \phi_2\left(\mathbf{x}_2^T\right) & \ldots & \phi_k\left(\mathbf{x}_2^T\right) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1\left(\mathbf{x}_n^T\right) & \phi_2\left(\mathbf{x}_n^T\right) & \ldots & \phi_k\left(\mathbf{x}_n^T\right) \end{bmatrix}. \tag{2.20}$$

The mathematical derivation stays the same as in the linear case, only the matrix $\mathbf{X}$ is replaced with its expanded counterpart $\Phi(\mathbf{X})$. The posterior of the latent function at the unobserved inputs $\mathbf{X}_*$ is defined by

$$p\left(\mathbf{f}_*|\mathbf{y}, \Phi(\mathbf{X}), \Phi(\mathbf{X}_*), \sigma_n^2\right) = \frac{\int p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right) p\left(\mathbf{f}, \mathbf{f}_*|\Phi(\mathbf{X}), \Phi(\mathbf{X}_*)\right) d\mathbf{f}}{p\left(\mathbf{y}|\Phi(\mathbf{X}), \sigma_n^2\right)}. \tag{2.21}$$

Explicitly the posterior is specified by

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{y}, &\Phi(\mathbf{X}), \Phi(\mathbf{X}_*), \sigma_n^2) = \\ &= \mathcal{N}\left(\mathbf{f}_*|\mathbf{K}_{*,f}\left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}, \mathbf{K}_{*,*} - \mathbf{K}_{*,f}\left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f,*}\right), \end{aligned} \tag{2.22}$$

where $\mathbf{K}_{f,f} = \Phi(\mathbf{X})\,\Phi(\mathbf{X})^T$, $\mathbf{K}_{*,f} = \Phi(\mathbf{X}_*)\,\Phi(\mathbf{X})^T$, and $\mathbf{K}_{*,*} = \Phi(\mathbf{X}_*)\,\Phi(\mathbf{X}_*)^T$. A block diagram of the Bayesian nonlinear regression is shown in Figure 2.4. The only difference with Figure 2.2a are the nonlinear transformations $\phi(\cdot)$ of the input vectors $\mathbf{x}_i$. Which basis functions should be used depends on the true underlying problem we are trying to model.

The following example is adapted from the book of David J.C. MacKay, ch. 45 [25]. For simplicity, let us assume the input to be a one-dimensional vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.23}$$

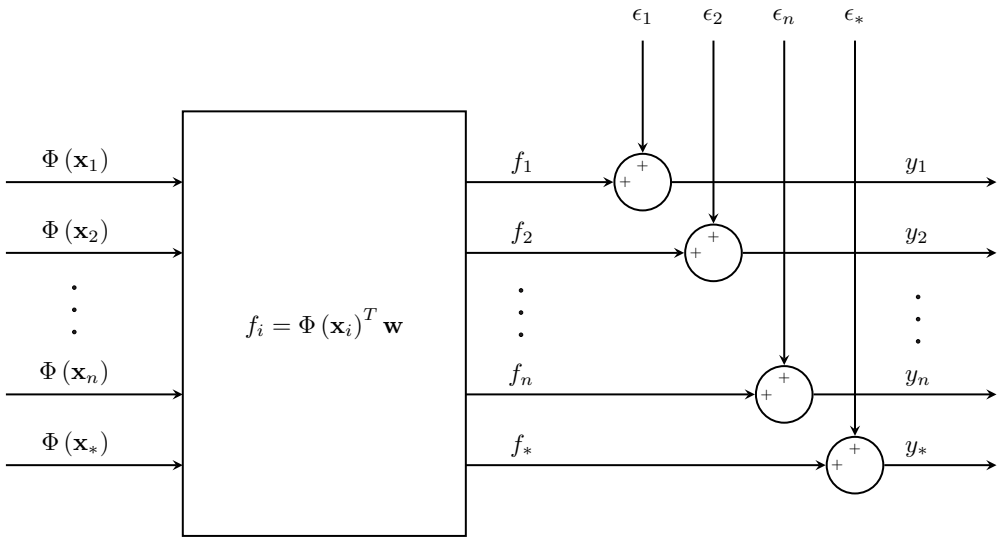


Figure 2.4: Block diagram of a Bayesian nonlinear regression model, where the function evaluated at the input $\mathbf{x}_i$ is defined by $f(\mathbf{x}_i) = \Phi(\mathbf{x}_i)^T \mathbf{w}$. $\Phi(\mathbf{x})$ represents the inputs which are expanded with basis functions. The output y_i denotes the latent value f_i corrupted by ϵ_i.

We consider a set of k squared exponential basis functions

$$\phi = \{\phi_1(x), \phi_2(\cdot), \ldots, \phi_k(\cdot)\}, \tag{2.24}$$

where

$$\phi_i(x) = \sigma_f \exp\left[-\frac{(x - h_i)^2}{2l^2}\right]. \tag{2.25}$$

The functions are centered uniformly on the interval $[h_{min}, h_{max}]$ such that

$$h_i = h_{min} + \Delta h \times i, \tag{2.26}$$

where $\Delta h = (h_{max} - h_{min})/(k-1)$, $\sigma_f^2 = \Delta h$ and $i \in \{0, \ldots, k-1\}$. We generate the data from a nonlinear problem defined by

$$\mathbf{y} = \sin(3\mathbf{x}) + 0.2\cos(10\mathbf{x}) + \mathcal{N}\left(\boldsymbol{\epsilon}|\mathbf{0}, 0.15^2\mathbf{I}\right) \tag{2.27}$$

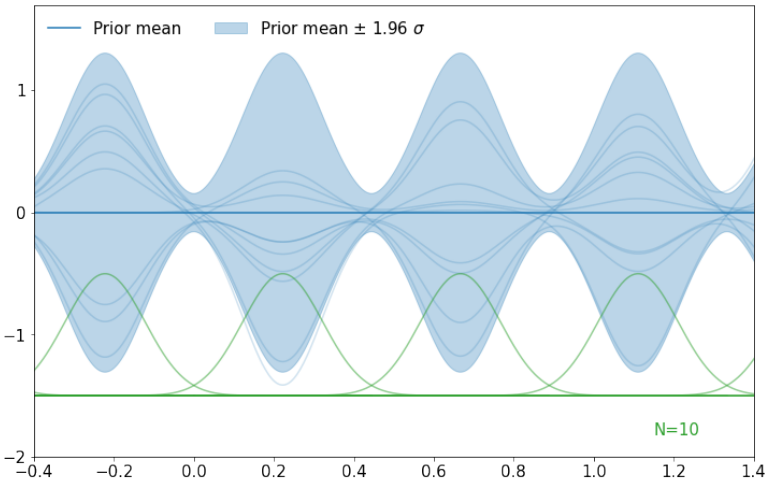
and uniformly place k=10 basis functions on the interval $[h_{max} - h_{min}] = [-2, 2]$. The basis function expansion of the input vector is then defined by the matrix

$$\boldsymbol{\Phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \ldots & \phi_k(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \ldots & \phi_k(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \ldots & \phi_k(x_n) \end{bmatrix}. \tag{2.28}$$

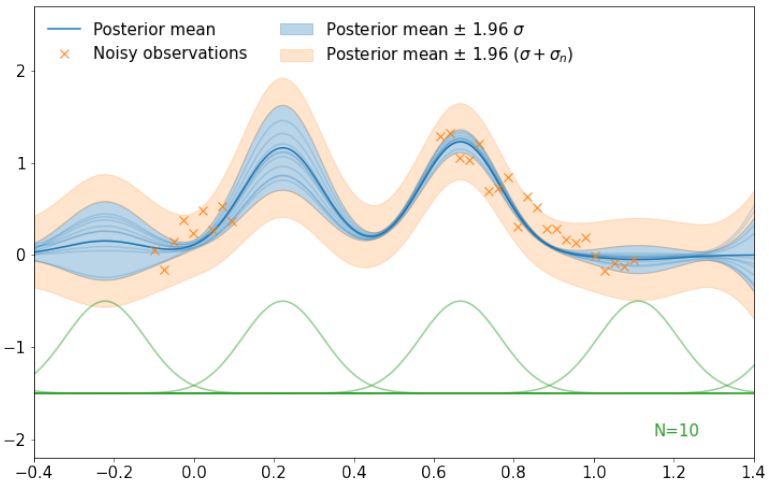
A fixed lengthscale parameter is assumed, i.e. $l = 0.1$. The nonlinear approach to the problem defined by Equation (2.27) is presented in Figure 2.5. Figure 2.5a shows the prior over latent function values and Figure 2.5b shows the corresponding posterior over the latent function values. Uniformly spaced squared exponential functions are represented in

green color. We can see that the nonlinear posterior distribution over the latent function values does not describe the observed data well.

Figure 2.6 shows the same example, only the number of squared exponential functions is increased to $k = 20$ over the same input interval. This results in increased smoothness of the resulting predictive distribution and in a better fit. The example with a 100 squared exponential functions is shown in Figure 2.7. But what if the number of basis functions is increased to infinity?

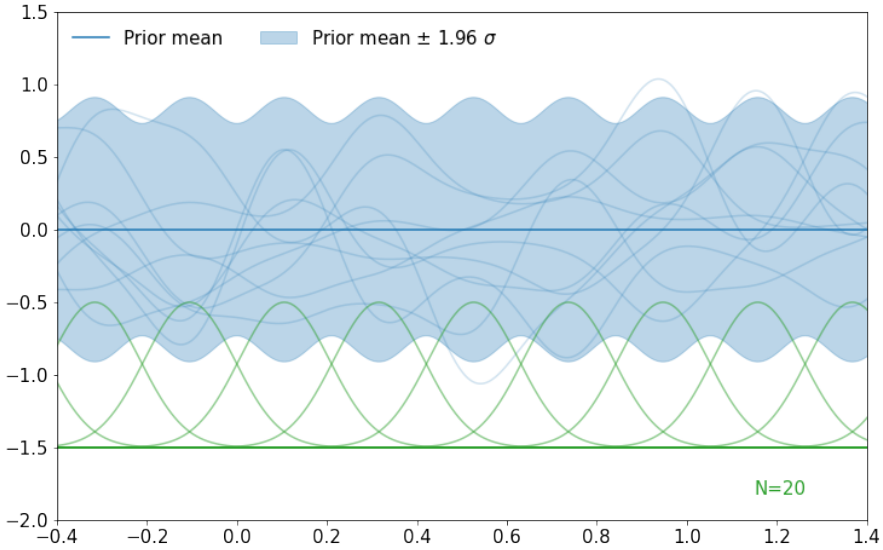


(a) Prior induced with 10 basis functions.

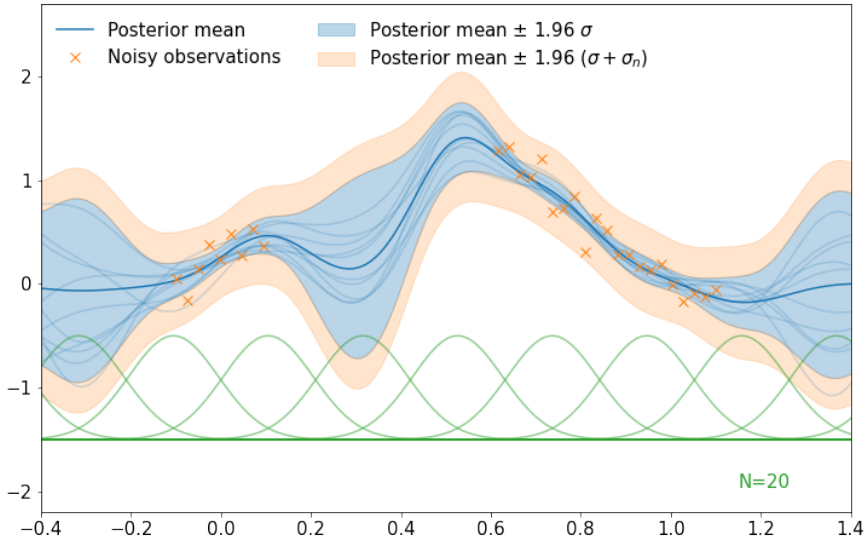


(b) Posterior induced with 10 basis functions.

Figure 2.5: Figure 2.5a shows the prior over the latent function values using 10 squared exponential basis functions that are uniformly spaced on the interval $(h_{max}, h_{min}] = (-2, 2]$. Basis functions are plotted in green. Figure 2.5b shows the corresponding posterior distribution over latent function values given the observed data.

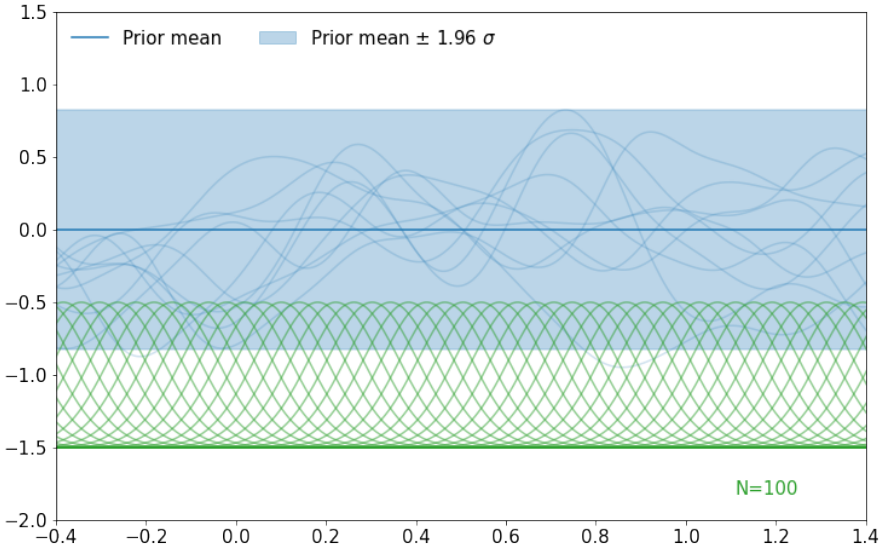


(a) Prior induced with 20 basis functions.

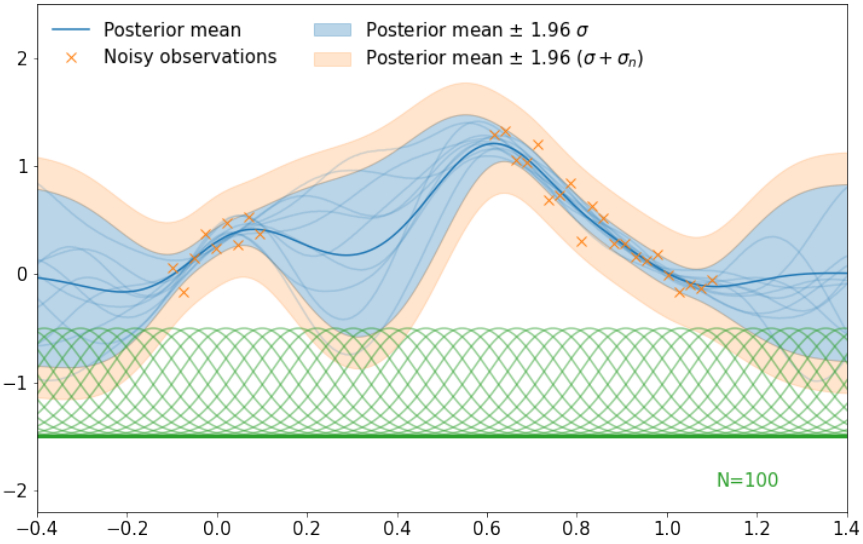


(b) Posterior induced with 20 basis functions.

Figure 2.6: Figure 2.6a shows the prior over the latent function values using 20 squared exponential basis functions that are uniformly spaced on the interval $(h_{max}, h_{min}] = (-2, 2]$. Basis functions are plotted in green. Figure 2.6b shows the corresponding posterior distribution over latent function values given the observed data.



(a) Prior induced with 100 basis functions.



(b) Posterior induced with 100 basis functions.

Figure 2.7: Figure 2.7a shows the prior over the latent function values using 100 squared exponential basis functions that are uniformly spaced on the interval $(h_{max}, h_{min}] = (-2, 2]$. Basis functions are plotted in green. Figure 2.7b shows the corresponding posterior distribution over latent function values given the observed data.

2.1.4 From Bayesian nonlinear regression to Gaussian process regression

We recall that $\mathbf{K}_{f,f} = \mathbf{\Phi}(\mathbf{x})\,\mathbf{\Phi}(\mathbf{x})^T$. Explicitly, the elements of the matrix $\mathbf{K}_{f,f}$ can be defined by

$$\left[\mathbf{K}_{f,f}\right]_{ij} = \sum_{p=1}^{k} \phi_p(x_i)\,\phi_p(x_j)\,, \tag{2.29a}$$

$$\left[\mathbf{K}_{f,f}\right]_{ij} = \sum_{p=1}^{k} \Delta h \,\exp\left[-\frac{(x_i - h_p)^2}{2l^2}\right] \exp\left[-\frac{(x_j - h_p)^2}{2l^2}\right]. \tag{2.29b}$$

If we now consider the uniformly spaced basis function in their limit when $\Delta h \to 0$ and $[h_{max}, h_{min}] \to [-\infty, \infty]$, the sum turns into an integral [25]

$$[\mathbf{K}]_{ij} = \int_{-\infty}^{\infty} \exp\left[-\frac{(x_i - h)^2}{2l^2}\right] \exp\left[-\frac{(x_j - h)^2}{2l^2}\right] dh, \tag{2.30a}$$

$$[\mathbf{K}]_{ij} = \sqrt{\pi l^2}\,\exp\left[-\frac{(x_i - x_j)^2}{4l^2}\right] = k(x_i, x_j)\,. \tag{2.30b}$$

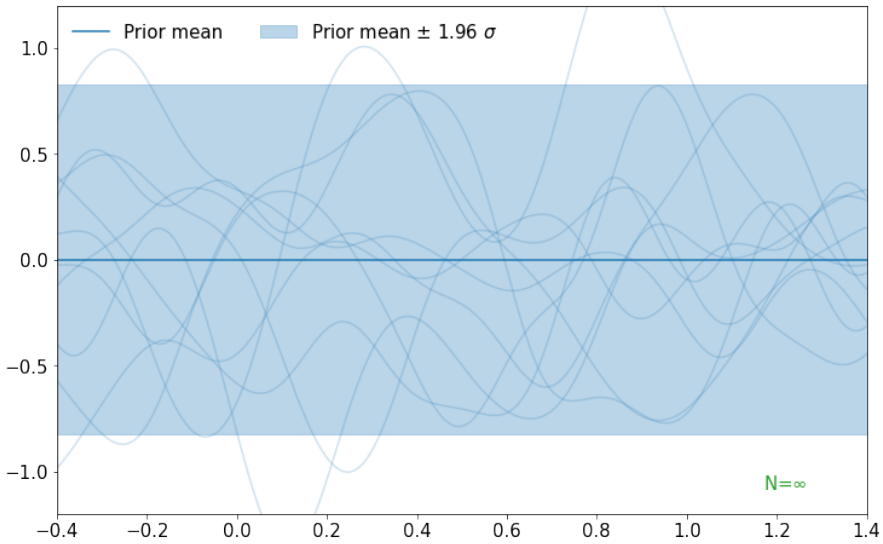
Instead of computing an inner product over an infinite-dimensional feature space, we instead equivalently evaluate a function [80]. Regression where the elements of the covariance matrix $\mathbf{K}_{f,f}$ are evaluated by a function, rather than by an inner product over an infinite-dimensional feature space, is called Gaussian process (GP) regression.

The example above demonstrates the nonparametric nature of GPs. GPs can be viewed as a weighted sum of an infinite number of basis functions. Therefore, the number of model parameters can be interpreted as infinite. The assumptions about the smoothness are, therefore, incorporated through the design of the function $k(\cdot,\cdot)$.

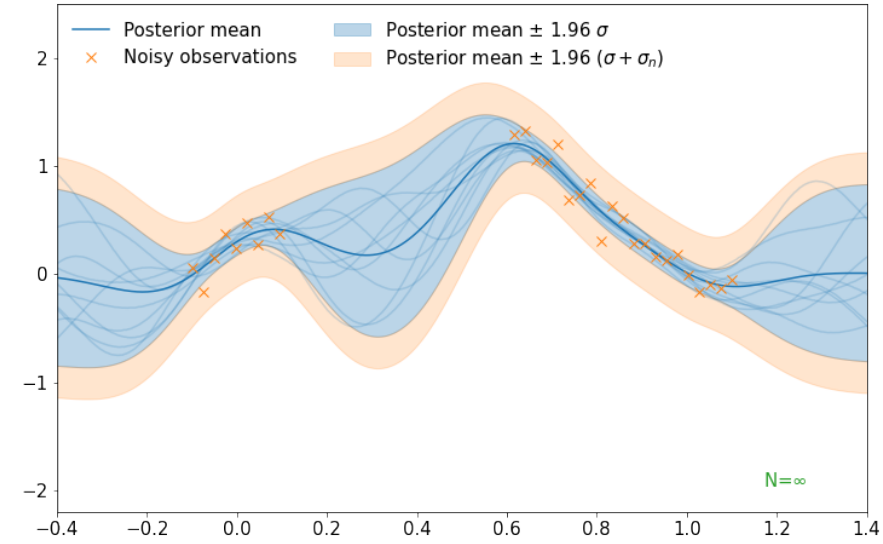
Figure 2.8 shows a GP regression example. The parameters σ_f^2 and l of the selected covariance function, i.e. the radial basis function (RBF) defined by Equation (C.3), were fixed to correspond to the squared exponential function expansion. We can see that the posterior over the latent function values generated by a GP is almost identical to that of the posterior obtained by a 100 basis functions presented in Figure 2.7.

We can generalize this concept and instead use a different function $k(\cdot,\cdot)$ in Equation (2.30b). However, the function should generate a positive definite matrix. A function with such property will be from here on equivalently referred to as the covariance function or the kernel. A number of common functions exist that satisfy this condition [81]. The covariance functions that will be used throughout this thesis are defined in Appendix C.

This introduction to GPs was from the point of basis function expansion. In the next section, we will define GPs through the "function view" representation [22].



(a) Prior induced with an infinite number of basis functions (GP).



(b) Posterior induced with infinite number of basis functions (GP).

Figure 2.8: Figure 2.8a shows the prior over the latent function values using an infinite number of squared exponential functions that are uniformly spaced on the interval $(h_{max}, h_{min}] = (-2, 2]$. Figure 2.8b shows the corresponding posterior distribution over latent function values given the observed data.

2.2 Gaussian Process Regression

Let us again consider a model of the form

$$\mathbf{y} = f(\mathbf{X}) + \boldsymbol{\epsilon} \tag{2.31}$$

and the set of observed data by $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. The noise $\boldsymbol{\epsilon}$ is assumed to be IID and follows a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}\sigma_n^2\right)$. Function f is in general a nonlinear mapping, which we model by a GP. The GP model represented in this section will be in future sections also referred to as the "vanilla GP", which emphasizes a full GP model structure that is not approximated [52].

2.2.1 Prior over functions

GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [22]. It is completely specified by its mean $m(\cdot)$ and covariance function $k(\cdot,\cdot)$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{2.32a}$$
$$k\left(\mathbf{x}, \mathbf{x}'\right) = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}')^T)]. \tag{2.32b}$$

A GP will be denoted by

$$f(\cdot) \sim \mathcal{GP}\left(f(\cdot)|m(\cdot), k(\cdot,\cdot)\right), \tag{2.33}$$

where the $m(\cdot)$ and $k(\cdot,\cdot)$ are parametrized by hyperparameters $\boldsymbol{\theta}$.

2.2.2 Posterior over functions

The prior over functions is transformed to the posterior using the Bayes theorem. The posterior is defined by

$$\mathcal{GP}\left(f(\cdot)|\mathbf{y}, \sigma_n^2, m(\cdot), k(\cdot,\cdot)\right) = \frac{p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right)\mathcal{GP}\left(f(\cdot)|m(\cdot), k(\cdot,\cdot)\right)}{p\left(\mathbf{y}|\sigma_n^2, m(\cdot), k(\cdot,\cdot)\right)}. \tag{2.34}$$

Hereafter, the conditioning on the choice of $m(\cdot)$ and $k(\cdot,\cdot)$ will be replaced with the parameters $\boldsymbol{\theta}$ that parametrize the aforementioned functions for notational simplicity, i.e.

$$\mathcal{GP}\left(f(\cdot)|\mathbf{y}, \boldsymbol{\theta}, \sigma_n^2\right) = \frac{p\left(\mathbf{y}|\mathbf{f}, \sigma_n^2\right)\mathcal{GP}\left(f(\cdot)|\boldsymbol{\theta}\right)}{p\left(\mathbf{y}|\boldsymbol{\theta}, \sigma_n^2\right)}. \tag{2.35}$$

We will usually also omit the conditioning on $\boldsymbol{\theta}$ and σ_n^2 since they will normally be deterministic variables learned from data. We will only introduce them back in the notation when they will be the parameters of interest or become a random variable. Additionally the mean function will be without the loss of generality considered as $m(\cdot) = \mathbf{0}$. The posterior GP is specified by

$$\begin{aligned}
\mathbb{E}\left[p(f(\cdot)|\mathbf{y})\right] &= k(\cdot, \mathbf{X})\left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}, \\
\mathbb{V}\left[p(f(\cdot)|\mathbf{y})\right] &= k(\cdot,\cdot) - k(\cdot, \mathbf{X})\left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1}k(\mathbf{X}, \cdot),
\end{aligned} \tag{2.36}$$

where $\left[\mathbf{K}_{f,f}\right]_{ij} = k\left(\mathbf{x}_i, \mathbf{x}_j\right)$. The posterior over the latent function values can also be factored to

$$\mathcal{GP}\left(f\left(\cdot\right), \mathbf{f}, \mathbf{f}_*|\mathbf{y}\right) = \mathcal{GP}\left(f\left(\cdot\right)|\mathbf{f}, \mathbf{f}_*, \mathbf{y}\right) p\left(\mathbf{f}, \mathbf{f}_*|\mathbf{y}\right), \tag{2.37}$$

where $\mathbf{f}$ represents the vector of latent function values at the observed inputs $\mathbf{X}$ and $\mathbf{f}_*$ the vector of latent function values at the unobserved inputs $\mathbf{X}_*$. Expression $f(\cdot)$ will from here on represent all other latent function values not explicitly defined in the respective joint distribution based on the context, e.g. in Equation (2.37), $f(\cdot)$ represents all latent values not in $\{\mathbf{f}, \mathbf{f}_*\}$. A block diagram and a probabilistic graphical model of a GP regression model are shown in Figure 2.9. Practically, we always work in finite sets of random variables because we can always analytically marginalize out all latent quantities that are not of interest, i.e.

$$p\left(\mathbf{f}, \mathbf{f}_*|\mathbf{y}\right) = \int \mathcal{GP}\left(f\left(\cdot\right)|\mathbf{f}, \mathbf{f}_*, \mathbf{y}\right) p\left(\mathbf{f}, \mathbf{f}_*|\mathbf{y}\right) d\mathbf{f}_{\setminus\{\mathbf{f}, \mathbf{f}_*\}}, \tag{2.38}$$

where $\mathbf{f}_{\setminus\{\mathbf{f}, \mathbf{f}_*\}}$ represents the set of all latent values that are not in $\{\mathbf{f}, \mathbf{f}_*\}$. The marginalization property of a Gaussian distribution allows us to simply "ignore" all other latent variables. Marginalization over a joint Gaussian distribution is defined in the Appendix by Equation (A.3). Marginalization over an infinite-dimensional distribution presented here should serve merely as a sketch. We reference the reader to [82] for a more technically involved analysis. The finite posterior over the vectors of the observed and the unobserved latent values is defined by

$$p\left(\mathbf{f}, \mathbf{f}_*|\mathbf{y}\right) = \frac{p\left(\mathbf{y}|\mathbf{f}\right) p\left(\mathbf{f}, \mathbf{f}_*\right)}{p\left(\mathbf{y}\right)}, \tag{2.39}$$

where the likelihood is $p\left(\mathbf{y}|\mathbf{f}\right) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \mathbf{I}\sigma_n^2\right)$. The marginal likelihood is explicitly defined by

$$p\left(\mathbf{y}\right) = \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right). \tag{2.40}$$

The joint prior is specified by

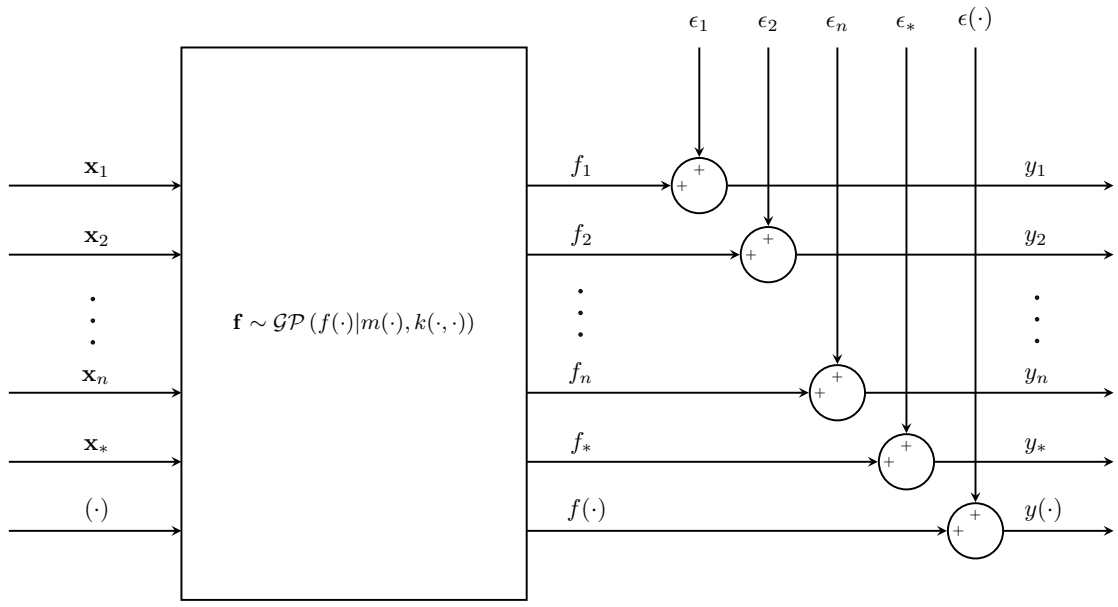
$$p\left(\mathbf{f}, \mathbf{f}_*\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f} \\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{f,f} & \mathbf{K}_{f,*} \\ \mathbf{K}_{*,f} & \mathbf{K}_{*,*}\end{bmatrix}\right), \tag{2.41}$$

where $\mathbf{K}_{f,f}, \mathbf{K}_{*,*}$, and $\mathbf{K}_{f,*}$ represent the covariance matrix between the observed inputs, covariance matrix between unobserved inputs, and their cross-covariance matrix, respectively. Marginalizing out the latent values $\mathbf{f}$ from the joint posterior in Equation (2.39) yields the predictive posterior $p\left(\mathbf{f}_*|\mathbf{y}\right)$ at the unobserved inputs $\mathbf{X}_*$. Again the predictive distribution can be more elegantly expressed as a conditional, derived from the joint distribution, i.e.

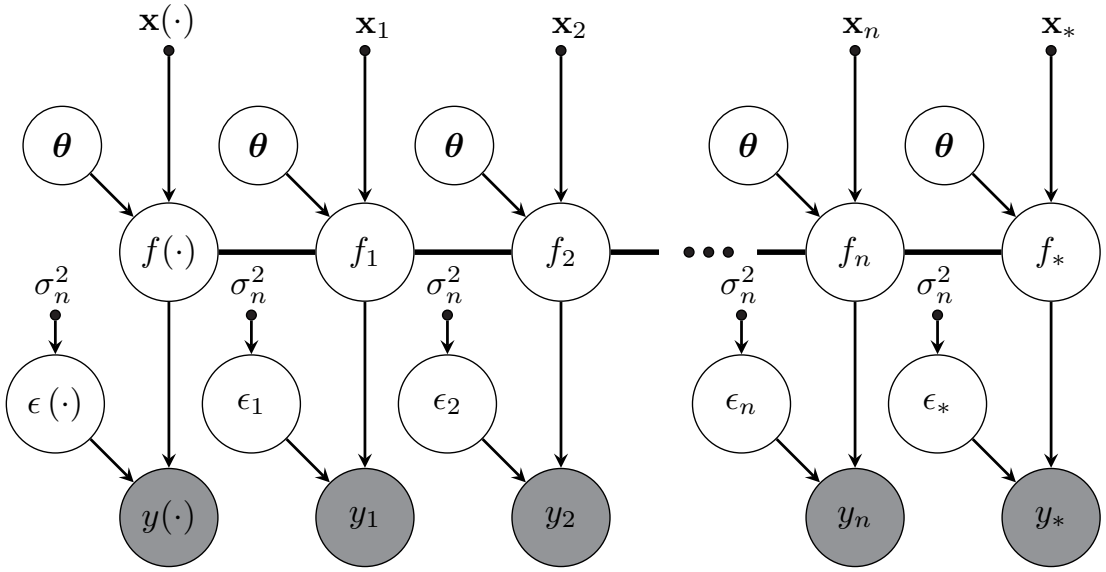
$$p\left(\mathbf{y}, \mathbf{f}_*\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y} \\ \mathbf{f}_*\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2 & \mathbf{K}_{f,*} \\ \mathbf{K}_{*,f} & \mathbf{K}_{*,*}\end{bmatrix}\right), \tag{2.42}$$

where the identity considering the conditioning over a joint Gaussian distribution can be found in the Appendix with Equation (A.4). The posterior at the unobserved inputs $\mathbf{X}_*$ is defined by

$$\begin{aligned}\mathbb{E}\left[p(\mathbf{f}_*|\mathbf{y})\right] &= \mathbf{K}_{*,f}\left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}, \\ \mathbb{V}\left[p(\mathbf{f}_*|\mathbf{y})\right] &= \mathbf{K}_{*,*} - \mathbf{K}_{*,f}\left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f,*}.\end{aligned} \tag{2.43}$$



(a) Block diagram of a GP regression model.



(b) PGM of a GP regression model.

Figure 2.9: Dual graphical representation of a GP regression. In both graphical representations, the function $f(\cdot)$ is modeled by a GP, i.e. $f(\cdot) \sim \mathcal{GP}\left(f(\cdot)|m(\cdot), k(\cdot,\cdot)\right)$. In PGM, the parameters of the mean and the covariance function, as well as the noise variance, will be hereafter omitted for brevity. Again it is important to note that the latent variables are fully connected, which is represented by a thick black line.

Figure 2.8 shows an example of a GP regression. Figure 2.8a shows the prior over latent function values, and Figure 2.8b shows the posterior given the observed data. So far, nothing was said about how the parameters of the mean, the covariance function, and the noise variance were determined. In the next section, we will show how these parameters can be obtained systematically.

2.2.3 Kernel learning

Here, we introduce back the conditioning on the parameters $\sigma_n^2, \boldsymbol{\theta}$ and treat them as random variables. Although technically the hyperparameters are only the parameters of the prior distribution, i.e. $\boldsymbol{\theta}$, we will from here on refer to hyperparameters as the set $\{\boldsymbol{\theta}, \sigma_n^2\}$. In a full probabilistic approach we would consider placing a prior distribution over the hyperparameters and infer their posterior given the observed data

$$p\left(\boldsymbol{\theta}, \sigma_n^2 | \mathbf{y}\right) = \frac{p\left(\mathbf{y}|\boldsymbol{\theta}, \sigma_n^2\right) p\left(\boldsymbol{\theta}, \sigma_n^2\right)}{\int p\left(\mathbf{y}|\boldsymbol{\theta}, \sigma_n^2\right) p\left(\boldsymbol{\theta}, \sigma_n^2\right) d\boldsymbol{\theta}\sigma_n^2}, \tag{2.44}$$

but unfortunately this cannot be evaluated in closed-form.

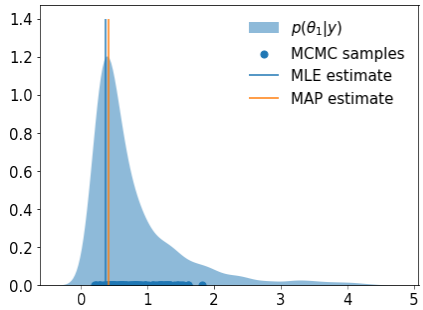
One could design a sampling scheme such as the Markov Chain Monte Carlo (MCMC) [6], [83], [84]. In the case of MCMC inference, the predictive posterior is also numerically approximated since the integration with respect to the hyperparameters cannot be obtained in closed-form. The predictive posterior is, therefore, approximated by an equally weighted sum over r samples, i.e.

$$p\left(\mathbf{f}_*|\mathbf{y}\right) = \int\int p\left(\mathbf{f}_*|\mathbf{y}, \boldsymbol{\theta}, \sigma_n^2\right) p\left(\boldsymbol{\theta}, \sigma_n^2|\mathbf{y}\right) d\boldsymbol{\theta} d\sigma_n^2 \approx \sum_{k=1}^{r} p\left(\mathbf{f}_*|\mathbf{y}, \tilde{\boldsymbol{\theta}}^{(k)}, \left(\tilde{\sigma}_n^2\right)^{(k)}\right), \tag{2.45}$$

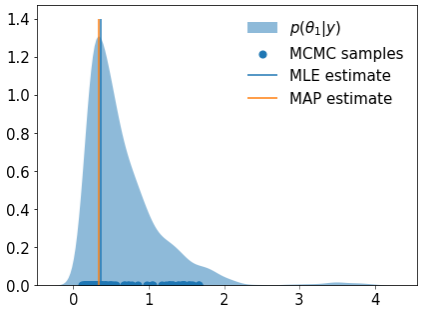
where $\tilde{\boldsymbol{\theta}}^{(k)}, \left(\tilde{\sigma}_n^2\right)^{(k)}$ represent the k-th realization from the posterior $p\left(\boldsymbol{\theta}, \sigma_n^2|\mathbf{y}\right)$ obtained by MCMC.

MCMC samples from the target distribution are correlated and can, therefore, explore the high dimensional space of hyperparameters more efficiently and more precisely than the independent MC approach. Independent MC provides a rough estimate, since the convergence of the algorithm is proportional to $r^{-\frac{1}{2}}$, where r represents the number of samples. However, it can be practical for the initialization of the MCMC algorithm, or in situations where the complexity of MCMC is prohibiting.

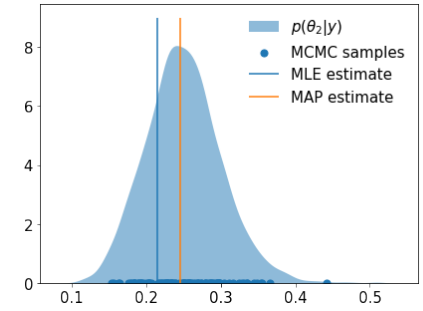
Figure 2.10 shows the results of hyperparameters inference (kernel learning) for the function defined by Equation (2.27). For MCMC, two chains were generated with Hamiltonian MC [84] up to 2500 samples, where the first 1000 samples were discarded as a part of the burn-in period. Blue solid circles show the first 100 samples (that were not discarded). The Blue vertical line represents the MLE solution. Unfortunately, MCMC sampling for GP regression is computationally very demanding as the sampler can converge very slowly due to the posterior distribution over the latent function values being highly correlated [85].



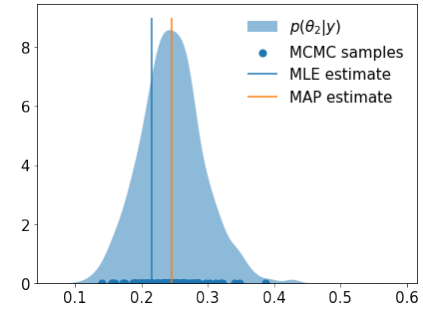
(a) Posterior distribution of the length-scale parameter for the 1st chain.



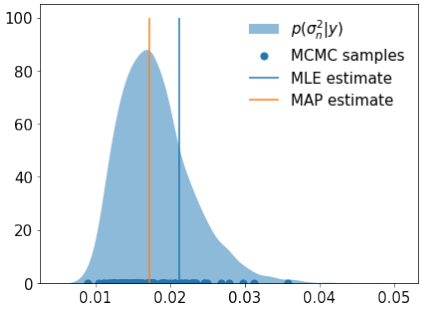
(b) Posterior distribution of the length-scale parameter for the 2nd chain.



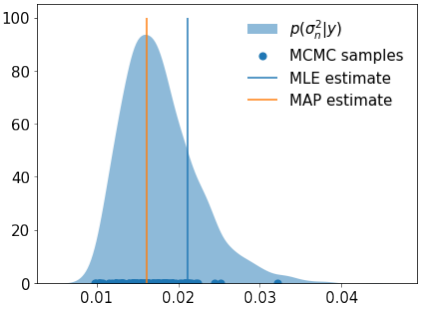
(c) Posterior distribution of the kernel variance parameter for the 1st chain.



(d) Posterior distribution of the kernel variance parameter for the 2nd chain.



(e) Posterior distribution of the noise variance parameter for the 1st chain.



(f) Posterior distribution of the noise variance parameter for the 2nd chain.

Figure 2.10: MCMC inference of the posterior hyperparameters given the observed data. Blue solid circles represent some samples of the posterior distribution of the hyperparameters. The MLE solution of the hyperparameters is represented with a blue vertical line. The maximum a posteriori (MAP) estimate is represented with an orange vertical line. The two columns show individual results for two independent MCMC chains.

When the data sets are relatively large, an alternative approach to MCMC is to maximize the logarithm of the marginal likelihood with respect to the hyperparameters, i.e.

$$\hat{\boldsymbol{\theta}}, \hat{\sigma}_n^2 = \underset{\boldsymbol{\theta}, \sigma_n^2}{\arg\max} \left[\log p\left(\mathbf{y}|\boldsymbol{\theta}, \sigma_n^2\right) \right], \tag{2.46}$$

where

$$\log p\left(\mathbf{y}|\boldsymbol{\theta}, \sigma_n^2\right) = \underbrace{-\frac{1}{2}\mathbf{y}^T \left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)^{-1} \mathbf{y}}_{\text{data fit term}} - \underbrace{\frac{1}{2}\log\left|\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right|}_{\text{regularization term}} - \underbrace{\frac{n}{2}\log 2\pi}_{\text{constant}}. \tag{2.47}$$

We will refer to this term as the marginal log-likelihood (MLL).

The elegant framework comes at the cost of the computational complexity of $\mathcal{O}(n^3)$ due to the inversion of the covariance matrix of the observed data which makes GP regression unpractical for data sets larger than a few thousand data points. [22]. The memory requirements are $\mathcal{O}(n^2)$ because of the storage of the covariance matrix.

In theory, GPs could be approximated with a limited set of basis functions [44], which translates the problem back to Bayesian linear regression and thus reduces the significant computational demand. In the case of stationary kernels, one can use the Fourier-feature-based approximation [86].

Unfortunately, approximations based on the finite representation of basis function expansion are too restrictive in their assumption and exhibit undesirable pathologies at test time [45]–[47].

With the advancement of contemporary software frameworks, the gradient of the MLL can be obtained by the means of automatic differentiation [87], [88]. Automatic differentiation exploits the fact that every algorithm is constructed by a sequence of elementary arithmetic operations and elementary functions. Elementary operations and functions are represented as a tuple that consists of:

1. A forward step;

2. A backward step.

The forward step simply applies the operation over the inputs to produce an output. The backward step calculates the derivative of the forward operation with respect to the free parameters and updates the free parameters based on the selected optimizer. To update the whole sequence of the forward operations, the chain rule is applied repeatedly.

In practice the MLL defined by Equation (2.47) is computed using the Cholesky decomposition of the covariance matrix $\mathbf{L}\mathbf{L}^T = \left(\mathbf{K}_{f,f} + \mathbf{I}\sigma_n^2\right)$, which is defined in the Appendix by Equation (A.10). Obtaining the derivatives over the Cholesky decomposition can be found in [89], and are implemented in contemporary software frameworks, e.g. Tensorflow [90] and, consequently, in a GP specific toolbox GPflow [91].

A common way to optimize the MLL is with a second-order optimization method, such as the Newton's method [92]. Second order optimization uses the Hessian to calculate the direction of the step that minimizes the local quadratic approximation of the cost function. Due to the high computational expense of calculating and inverting the Hessian, an approximation is usually used.

If not stated otherwise, the reader should assume that in the examples used in this thesis, limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) optimizer is used to maximize the MLL, which computes a low-rank approximation of the Hessian with finite differences [93].

So far we have considered how the hyperparameters are learned in GP regression. We also defined how the posterior moments of the prediction are specified in closed-form. However, in certain scenarios, we wish to obtain a sample from the posterior GP. Two examples where the posterior samples are required are:

1. Numerical marginalization with respect to GP prediction;
2. Simulation of dynamical GP models.

In the next section, we will consider how a sample from the posterior is obtained.

2.2.4 Sampling from the posterior GP

Sampling from the posterior GP practically refers to taking a sample from the posterior over functions given the observed data, i.e.

$$\text{draw } \tilde{f}(\cdot) \sim \mathcal{GP}(f(\cdot)|\mathbf{y}), \tag{2.48}$$

where $\tilde{f}(\cdot)$ denotes a realization of the posterior function. Since a function can be seen as an infinite-dimensional vector, taking a sample from an infinite object seems confusing. In practice, we cannot sample from an infinite-dimensional object. A practical trick lies in the marginalization of the latent values that we are not interested in. Since the joint distribution is Gaussian, the marginalization is trivial. For example, the GP posterior can be factored as

$$\mathcal{GP}(f(\cdot), \mathbf{f}_*|\mathbf{y}) = \underbrace{\mathcal{GP}(f(\cdot)|\mathbf{f}_*, \mathbf{y})}_{\text{infinite part}} \underbrace{p(\mathbf{f}_*|\mathbf{y})}_{\text{finite part}}. \tag{2.49}$$

In practice, a sample is drawn from the finite part. The sample from the posterior can be computed in two ways:

1. Batch algorithm;
2. Sequential algorithm.

In static systems, a more convenient batch draw is preferred. However, one has to know the inputs at which the posterior is evaluated beforehand. In the simulation of dynamical GP models, this is not possible since the inputs are evaluated iteratively and the posterior is sampled sequentially. In the next two sections, we will demonstrate how a batch sample and a sequential sample can be obtained from the posterior GP.

Batch sample

Let us explicitly define the test latent function values as

$$\mathbf{f}_* = \{f_1, f_2, f_3, \ldots, f_n\}. \tag{2.50}$$

A batch sample from the finite part, i.e.

$$\text{draw } \tilde{f}_1, \tilde{f}_2, \tilde{f}_3, \ldots, \tilde{f}_n \sim p\left(\mathbf{f}_* | \mathbf{y}\right), \tag{2.51}$$

is simply a draw from a multivariate Gaussian which is defined in the Appendix with Equation (A.8). For generating visually appealing plots, the infinite part can be approximated by

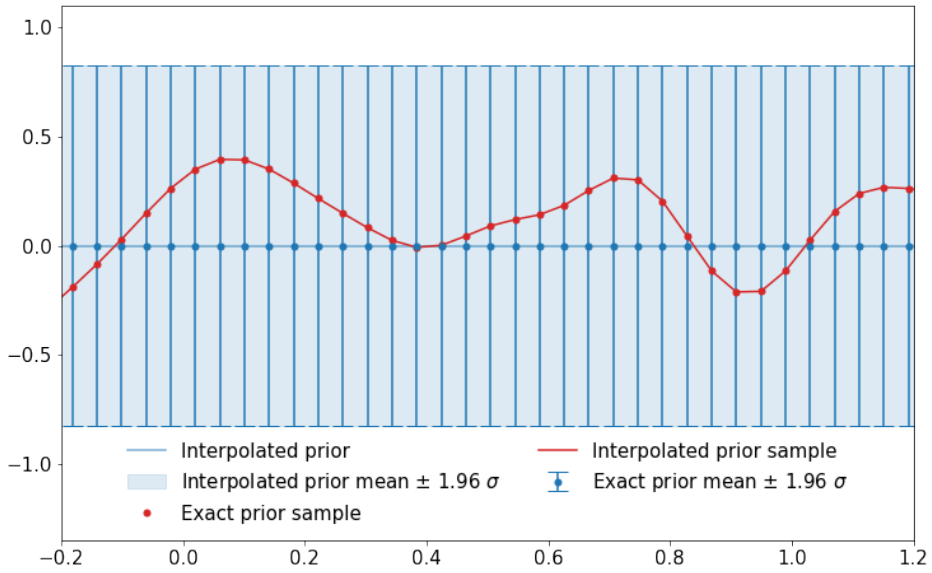
$$\tilde{f}\left(\cdot\right) \sim \mathcal{GP}\left(f\left(\cdot\right) | \tilde{\mathbf{f}}_*, \mathbf{y}\right) \approx g\left(\cdot, \tilde{\mathbf{f}}_*\right), \tag{2.52}$$

where $\tilde{f}\left(\cdot\right)$ represents an approximated function realization at input locations that are not in $\mathbf{X}_*$ and g represents a piecewise linear function that interpolates between the latent realizations of the posterior $\tilde{\mathbf{f}}_*$.

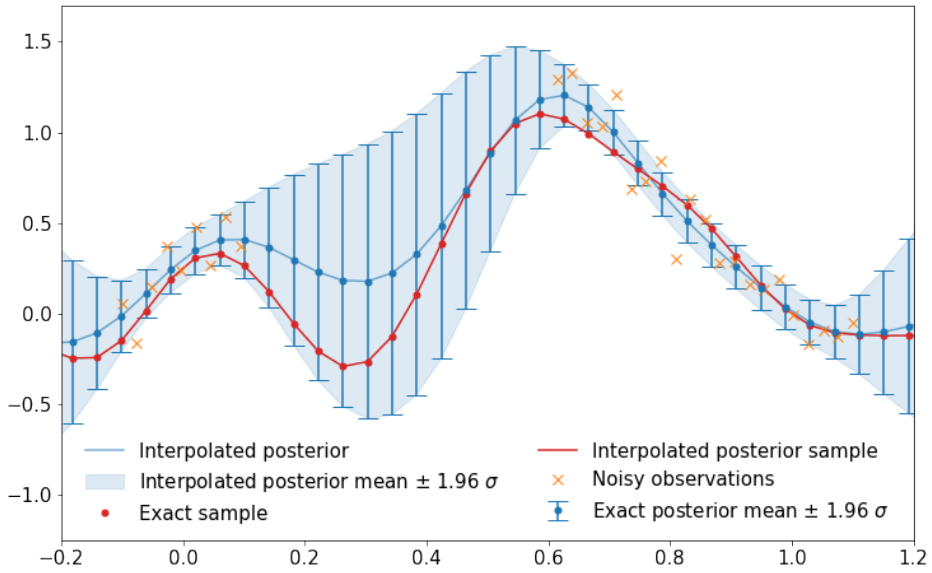
Figure 2.11 represents this approximation colored in red. If we look at the posterior shown in Figure 2.11b, exact samples $\tilde{\mathbf{f}}_*$ are presented with red solid circles. The approximated latent function values $\tilde{f}\left(\cdot\right)$ are shown with red lines that interpolate between the exact latent samples $\tilde{\mathbf{f}}_*$. The same procedure is applied to the prior in Figure 2.11a, where the latent function values are drawn from the prior instead.

The blue bars in Figure 2.11 are also exact and represent the prior or the posterior marginals of the latent function values $\mathbf{f}_*$. The blue green region represents the linearly interpolated part of the marginals, where the mean and the 95% confidence interval are again approximated with a piecewise linear function.

It is important to understand that until an arbitrary latent function f', at an arbitrary input x', is explicitly added to the joint distribution (to the set $\mathbf{f}_*$), one cannot in practice obtain an exact sample at x'. This is a consequence of the nonparametric nature of GPs. Note that this is not conceptually a problem in linear models, where by definition linear interpolation is exact, and so are the samples of the latent function values. For that reason, GPs using a linear covariance function do not justify the cubic computational complexity that originates from the large joint probabilistic model, since the same result can be obtained more efficiently through the Bayesian linear regression framework.



(a) Prior over the latent function.



(b) Posterior over the latent function.

Figure 2.11: A batch sample from the posterior GP. Figure 2.11a represents the prior and Figure 2.11b represents the posterior over the latent function $f(\cdot)$. The means and the confidence intervals shown with blue bars are evaluated at uniformly spaced inputs $\mathbf{X}_*$. They are represented by a finite distribution, i.e. $p(\mathbf{f}_*)$ for the prior or $p(\mathbf{f}_*|\mathbf{y})$ for the posterior, and are exact. The blue solid region represents the linearly interpolated parts. Red circles represent an exact sample $\tilde{\mathbf{f}}_*$ from the prior or the posterior, respectively. Red connecting lines show an interpolated approximation of the sample at all other latent function values.

Sequential sample

An equivalent approach to a batch sample is to sample the finite part of the latent function $p(\mathbf{f}_*|\mathbf{y})$ sequentially. We can obtain a discrete sample $\tilde{\mathbf{f}}_*$ over the inputs $\mathbf{X}_*$ by

$$\begin{aligned} &\text{draw } \tilde{f}_1 \sim p(f_1|\mathbf{y}) \\ &\text{draw } \tilde{f}_2 \sim p(f_2|\tilde{f}_1, \mathbf{y}) \\ &\text{draw } \tilde{f}_3 \sim p(f_3|\tilde{f}_2, \tilde{f}_1, \mathbf{y}) \\ &\qquad \vdots \\ &\text{draw } \tilde{f}_n \sim p(f_n|\tilde{f}_{n-1}, \ldots, \tilde{f}_2, \tilde{f}_1, \mathbf{y}) \end{aligned} \tag{2.53}$$

For plotting, the infinite part is again approximated by

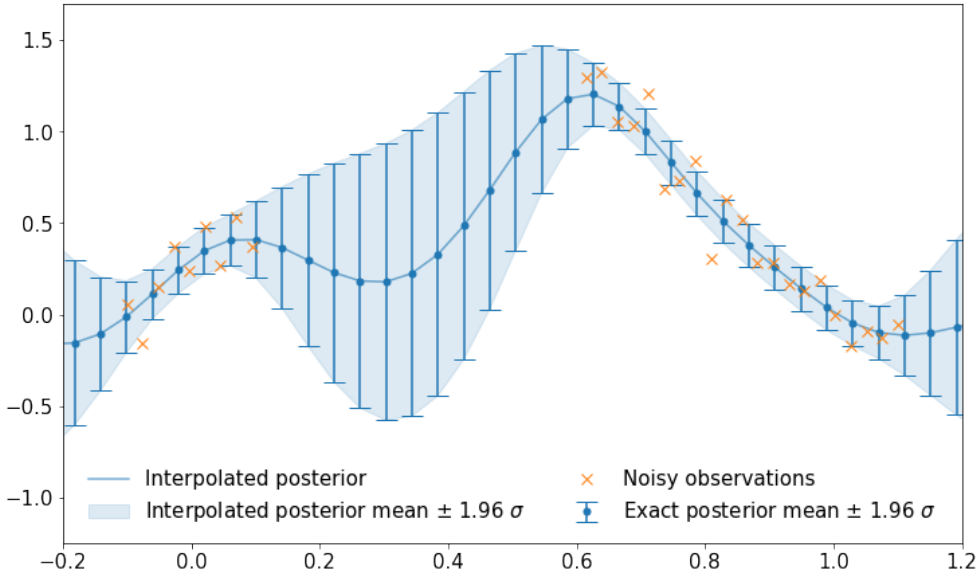
$$\tilde{f}(\cdot) \sim \mathcal{GP}\left(f(\cdot)|\tilde{\mathbf{f}}_*, \mathbf{y}\right) \approx g\left(\cdot, \tilde{\mathbf{f}}_*\right). \tag{2.54}$$

In static problems, the sample draw presented in Equation (2.53) is invariant to permutation of the latent samples. A sequential sample is non-Markovian since a single latent sample depends on all realizations of the latent values up to the respective latent value considered. In practice, the sequential sample would not be applied in static problems as the batch algorithm is more convenient. However, it provides an intuitive view of how a sample from a GP can be obtained if the nature of the problem is dynamic, and the future inputs are not known beforehand, but rather determined iteratively.

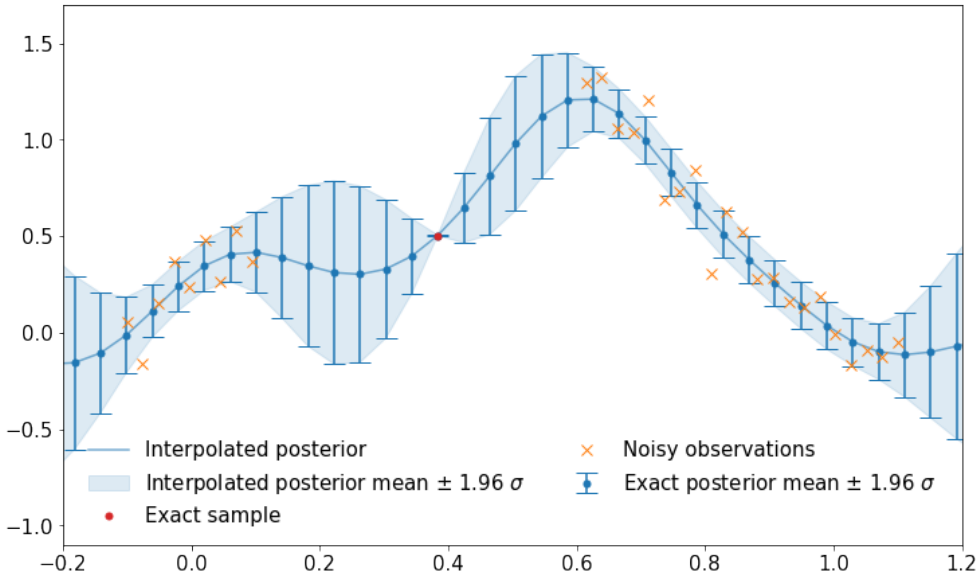
A sequential sample for 3 latent values is demonstrated in Figure 2.12. Samples $\tilde{f}_1, \tilde{f}_2, \tilde{f}_3$ are shown with red solid circles. The marginals for other latent function values not in $\{f_1, f_2, f_3\}$, i.e. of $p\left(\mathbf{f}_{*\setminus\{f_1,f_2,f_3\}}|\tilde{f}_1, \tilde{f}_2, \tilde{f}_3, \mathbf{y}\right)$, are shown with blue bars and are exact. Blue solid region is approximated and represents the linear interpolated part for arbitrary latent function values $f(\cdot)$.

The rest of the thesis will use the approach described in this section to visualize the posterior function mean, variance, or sample, where we will equivalently use the batch or the sequential representation for static or dynamic GPs respectively. This generates visually appealing graphs, but the reader should remember that an exact representation of the latent function value at an arbitrary input can only be obtained if that latent function value is explicitly added to the joint distribution.

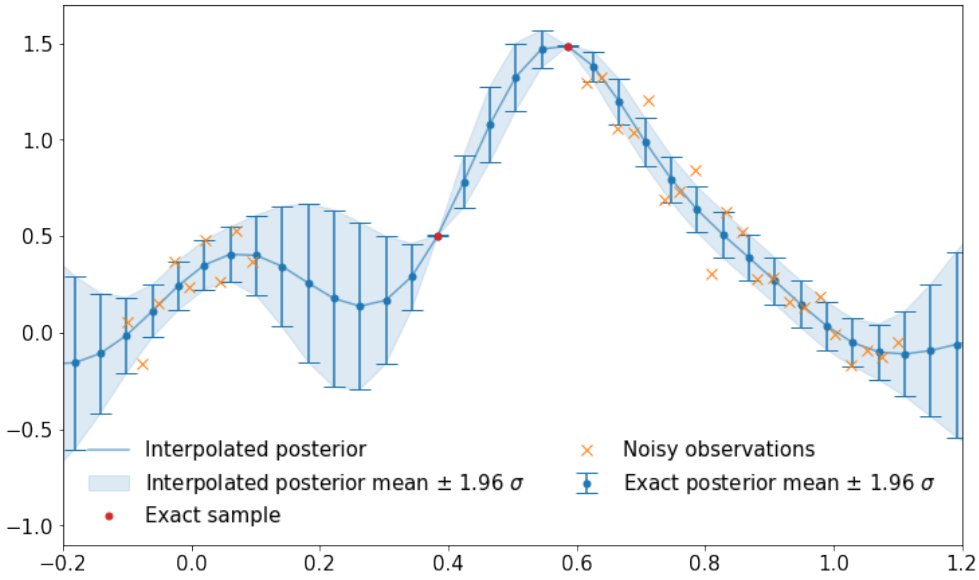
So far we have only dealt with static problems. In the next section, we will introduce a dynamical GP model that can model time-ordered data. The simulation of the dynamical model considered will justify the introduction of the sequential sampling of the posterior that is otherwise not practical in static GPs.



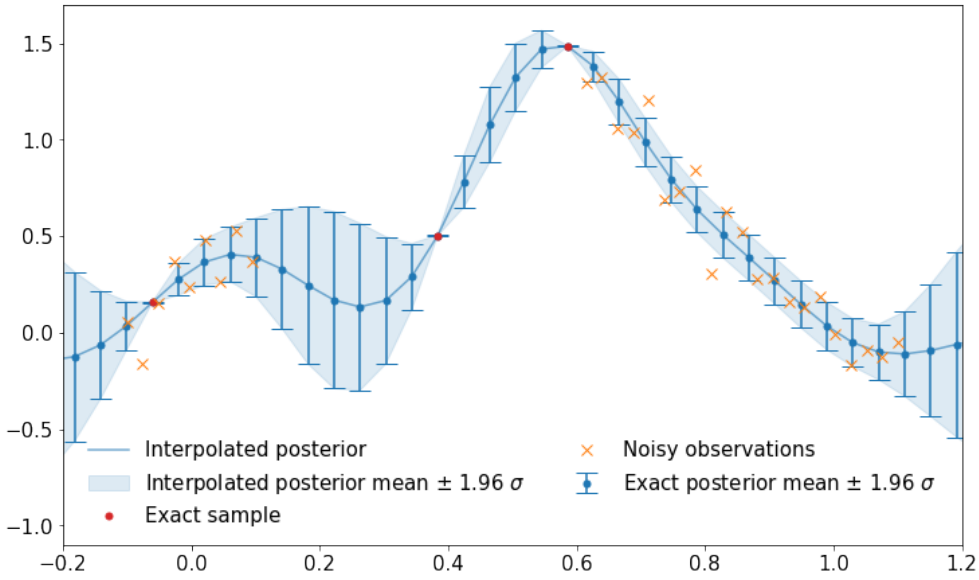
(a) Posterior prediction given the observed data, i.e. $p(f(\cdot)|\mathbf{y})$. Exact marginals are shown with blue bars. The interpolated part is shown with solid blue color.



(b) Posterior prediction given the observed data and 1 latent realization. Exact marginals are shown with blue bars, i.e. $p\left(\mathbf{f}_{*\setminus f_1}|\tilde{f}_1,\mathbf{y}\right)$, and an exact realization of the latent function value $\tilde{f}_1$ is represented by a red circle. The interpolated part is shown with solid blue color.



(c) Posterior prediction given the observed data and 2 latent realizations. Exact marginals are shown with blue bars, i.e. $p\left(\mathbf{f}_{*\setminus\{f_1,f_2\}}|\tilde{f}_1,\tilde{f}_2,\mathbf{y}\right)$, and exact realizations of the latent function values $\{\tilde{f}_1,\tilde{f}_2\}$ are represented by red circles. The interpolated part is shown with solid blue color.



(d) Posterior prediction given the observed data and 3 latent realizations. Exact marginals are shown with blue bars, i.e. $p\left(\mathbf{f}_{*\setminus\{f_1,f_2,f_3\}}|\tilde{f}_1,\tilde{f}_2,\tilde{f}_3,\mathbf{y}\right)$, and exact realizations of the latent function values $\{\tilde{f}_1,\tilde{f}_2,\tilde{f}_3\}$ are represented by red circles. The interpolated part is shown with solid blue.

Figure 2.12: Sequential sample from a GP posterior for 3 latent realizations.

2.3 Autoregressive Gaussian Process Regression

In this section, we will introduce dynamical notation for indexing the latent values and the corresponding observed data. Note that the dynamic notation of indexing the latent values by time steps can be for static problems equivalently replaced by a more common notation in the GP literature, i.e.

$$\begin{aligned} \mathbf{f}_{1:t} &\mapsto \mathbf{f}, \\ \mathbf{f}_{t+1:t+n} &\mapsto \mathbf{f}_*, \\ f(\cdot) &\mapsto f(\cdot). \end{aligned} \tag{2.55}$$

Unfortunately, we need the tedious notation by the time index to represent the simulation of GP-NARX models, which requires ordered latent values to fully and precisely describe the algorithms in mathematical terms. We also want to emphasize how the notation in dynamical GPs corresponds to a more common notation in the system identification literature. The control inputs in this thesis will be denoted by $\mathbf{x}$, but they can be equivalently replaced by a more common notation, i.e.

$$\mathbf{x} \mapsto \mathbf{u}. \tag{2.56}$$

However, in the literature considering approximations of GPs, the letter $\mathbf{u}$ is reserved for pseudo-points, which will be introduced in later chapters, and we would like to avoid that confusion. Additionally, we want to emphasize that the estimated value that is in the system identification field usually denoted by $\hat{\mathbf{y}}_{t+1}$, is in GP regression replaced by the posterior mean at $t+1$, i.e

$$\mathbb{E}[p(y_{t+1}|\mathbf{y}_{1:t})] \mapsto \hat{y}_{t+1}. \tag{2.57}$$

In the following section, we will introduce a dynamical GP model that models the input/output dependency by a nonlinear autoregressive model with exogenous inputs (NARX) model [23].

2.3.1 Nonlinear autoregressive model with exogenous inputs

Let us assume that the true dynamics are governed by

$$f_t = f\left(f_{t-n_a}, \ldots, f_{t-1}, x_{t-n_b}, \ldots, x_{t-1}\right), \tag{2.58}$$

where f is a nonlinear function, f_t an evaluation of the nonlinear function at the respective time step $t+1$, and $n_a, n_b > 0$ denote the number of lags for the delayed outputs and delayed exogenous variables, respectively. In matrix form, this can be represented by

$$\mathbf{f}_{1:t} = f(\mathbf{Z}_{1:t}), \tag{2.59}$$

where $\mathbf{f}_{1:t}$ denotes the vector of latent function values. The input matrix $\mathbf{Z}_{1:t} \in \mathcal{R}^{t\times(n_a+n_b)}$ is represented with a NARX model, where t denotes the number of observations. The i-th row of the matrix $\mathbf{Z}_{1:t}$ is defined by

$$\mathbf{z}_i^T = [f_{i-n_a}, \ldots, f_{i-1}, x_{i-n_b}, \ldots, x_{i-1}]. \tag{2.60}$$

Instead of the true output vector we observe a noisy one, i.e. $\mathbf{y}_{1:t} = \mathbf{f}_{1:t} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}\sigma_n^2)$. The dynamical system is then modeled by

$$\mathbf{y}_{1:t} = f(\mathbf{Z}_{1:t}) + \boldsymbol{\epsilon}, \tag{2.61}$$

where the i-th row of the input matrix $\mathbf{Z}_{1:t}$ is defined by

$$\mathbf{z}_i^T = [y_{i-n_a}, \ldots, y_{i-1}, x_{i-n_b}, \ldots, x_{i-1}]. \tag{2.62}$$

The goal is to infer the latent function f from Equation (2.61). Similarly, as in the static case, the latent function is modeled with a GP.

2.3.2 Kernel learning and prediction

We again introduce the posterior over functions

$$\mathcal{GP}\left(f\left(\cdot\right)|\mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}\right)\mathcal{GP}\left(f\left(\cdot\right)\right)}{p\left(\mathbf{y}_{1:t}\right)}. \tag{2.63}$$

The cost of representing a dynamical model in this form is that now not only the output is corrupted by noise, but so are the inputs. Throughout this thesis, we will refer to this model as the vanilla GP-NARX model. The hyperparameters of the GP-NARX model can be obtained through the maximization of the MLL defined by

$$\log p\left(\mathbf{y}_{1:t}\right) = -\frac{1}{2}\mathbf{y}_{1:t}^{T}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}_{1:t} - \frac{1}{2}\log\left|\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right| - \frac{n}{2}\log 2\pi. \tag{2.64}$$

The prediction at $t+1$ is obtained by marginalizing out the latent values that do not belong to time step $t+1$ out of the posterior, i.e.

$$p\left(f_{t+1}|\mathbf{y}_{1:t}\right) = \int \mathcal{GP}\left(f\left(\cdot\right)|\mathbf{y}_{1:t}\right)d\mathbf{f}_{\setminus\{f_{t+1}\}} = \frac{\int p\left(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}\right)p\left(\mathbf{f}_{1:t},f_{t+1}\right)d\mathbf{f}_{1:t}}{p\left(\mathbf{y}_{1:t}\right)}. \tag{2.65}$$

The mean and the variance of the predictive posterior are specified by

$$\mathbb{E}[f_{t+1}|\mathbf{y}_{1:t}] = \mathbf{K}_{f_{t+1},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}_{1:t}, \tag{2.66a}$$

$$\mathbb{V}[f_{t+1}|\mathbf{y}_{1:t}] = \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f_{1:t},f_{t+1}}. \tag{2.66b}$$

Equivalent, but numerically more stable algorithm, is defined in Appendix E.1.1.

PGM of the GP-NARX model is presented in Figure 2.13. Again, it is important to note that all latent values $\mathbf{f}$ in the PGM are fully connected which is shown with a thick black line. The block diagram for NARX model construction is shown in Figure 2.14, which illustrates how the dynamical input at the time step t is obtained, i.e. by concatenating the lagged exogenous variables and the lagged observations. The block diagrams for training and prediction in a GP-NARX model are shown in Figure 2.15 and Figure 2.16. Although arbitrary latent function values $f\left(\cdot\right)$ are in practice marginalized out, we keep them in the block diagrams to emphasize the omnipresence of the fully connected Gaussian field.

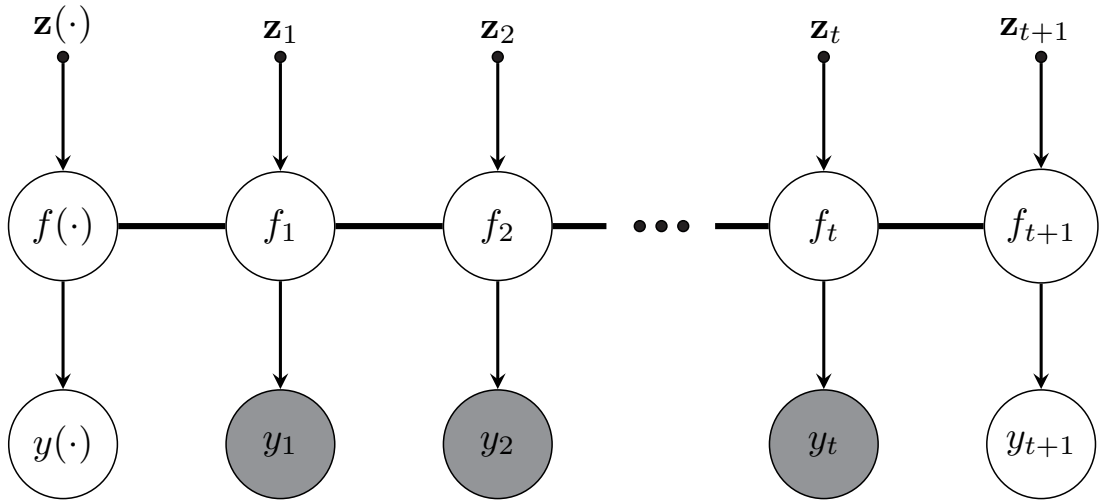


Figure 2.13: PGM of a GP-NARX regression model. The thick black line denotes fully correlated latent values.

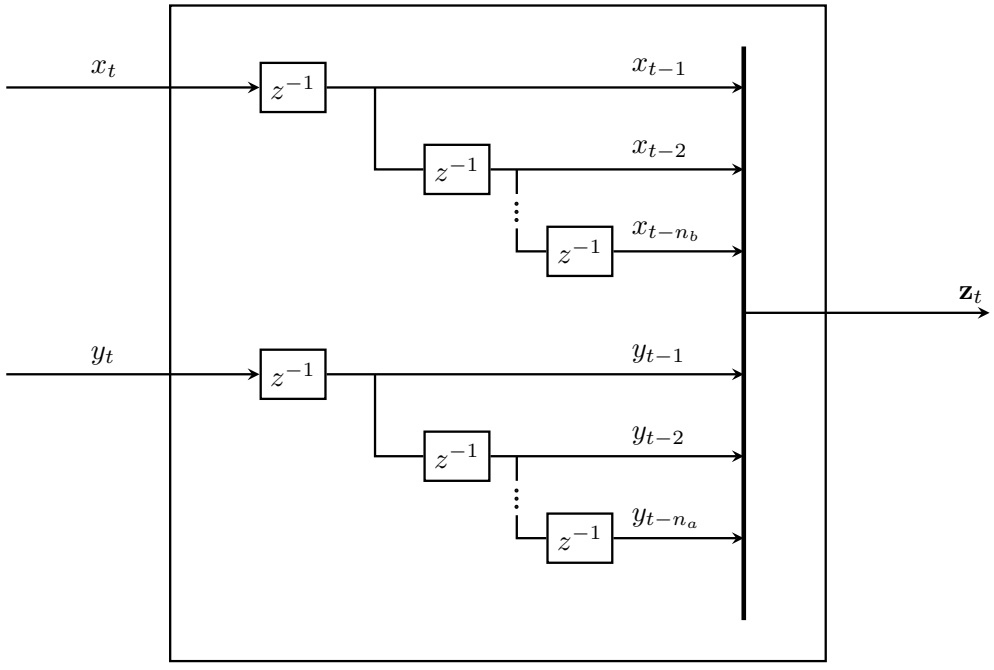


Figure 2.14: Block diagram of NARX model construction. z^{-1} represents the lag operator. The dynamical input at the time step t is obtained by concatenating the lagged exogenous variables and the lagged observations.

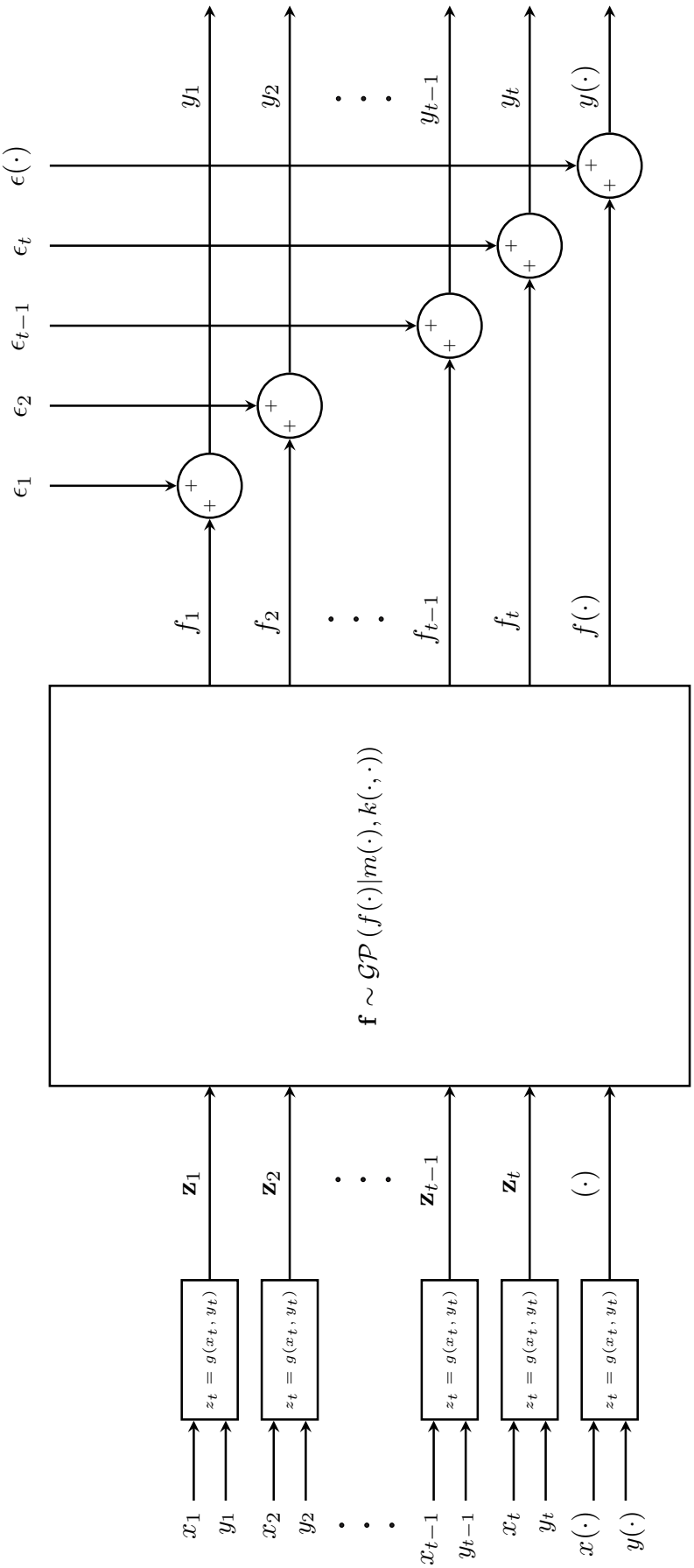


Figure 2.15: Block diagram of a GP-NARX model training. In training, the GP-NARX model is identical to that of the static case, the only difference being the construction of the inputs $\mathbf{z}$. The transformation $g\left(x_t, y_t\right)$ represents the NARX model constructor, shown in Figure 2.14.

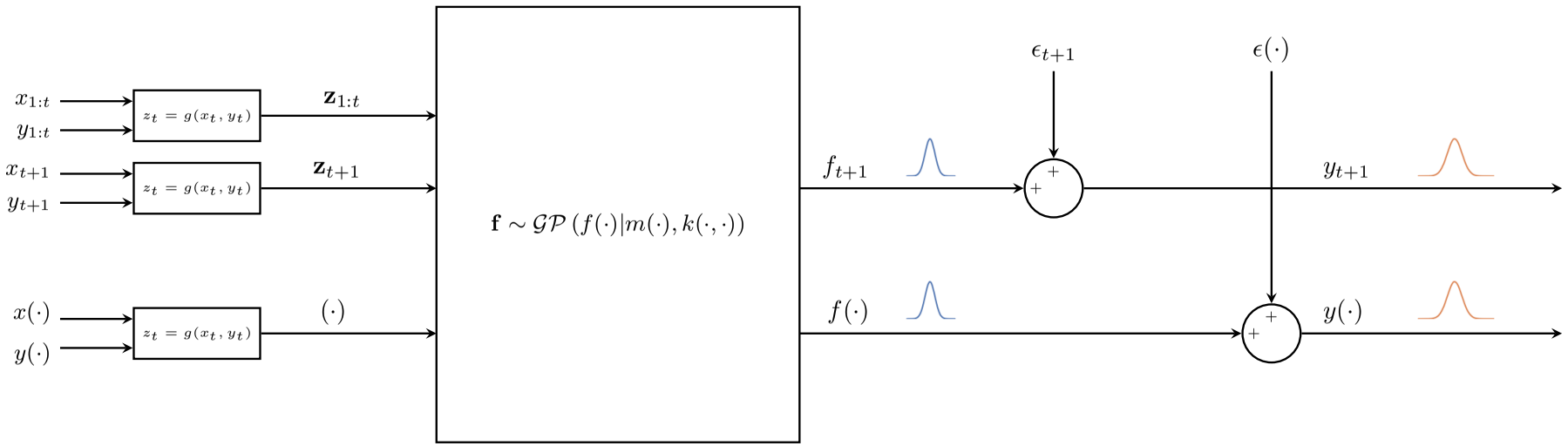


Figure 2.16: Block diagram of a GP-NARX model prediction. In prediction, the GP-NARX model is identical to that of the static case, where the only difference is the construction of the inputs $\mathbf{z}$. The transformation $g\left(x_t, y_t\right)$ represents the NARX model constructor, shown in Figure 2.14.

2.3.3 Simulation

Marginalizing the GP posterior over all the latent quantities that are not of interest, we arrive at the posterior prediction at time steps from $t+1$ up to $t+n$

$$\begin{aligned} p\left(f_{t+1:t+n}|\mathbf{y}_{1:t}\right) &= \int \mathcal{GP}\left(f\left(\cdot\right)|\mathbf{y}_{1:t}\right) d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}} \\ &= \frac{\int p\left(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}\right) p\left(\mathbf{f}_{1:t}, \mathbf{f}_{t+1:t+n}\right) d\mathbf{f}_{1:t}}{p\left(\mathbf{y}_{1:t}\right)}. \end{aligned} \tag{2.67}$$

Note that the estimation of the latent function values, up to the time step $t+n$, can only be obtained in closed-form if the inputs at that time step are known. Unfortunately, that is not the case in the simulation. The inputs have to be determined iteratively, using the past predicted latent values to replace the output data in the dynamical input $\mathbf{z}$. Consequently, the posterior sample from a dynamical GP model has to be determined sequentially. The problem in simulation is best illustrated by an example. Let us consider a model of the form

$$f_{t+1} = f(\mathbf{z}_{t+1}), \tag{2.68}$$

where the input is represented by a NARX model with parameters $n_a = 1, n_b = 2$, i.e.

$$\mathbf{z}_{t+1}^T = [y_t, x_{t-1}, x_t] \tag{2.69}$$

and the function is modeled by a GP. We observe the noise corrupted values $\mathbf{y}_{1:t}$ and the corresponding exogenous variables $\mathbf{x}_{1:t}$ up to the time step t. At the time step $t+1$, the latent function f_{t+1} simply follows the predictive distribution $p\left(f_{t+1}|\mathbf{y}_{1:t}\right)$, defined by Equation (2.66). At the time step $t+2$, the input becomes a random variable, i.e.

$$p(\mathbf{z}_{t+2}^T|f_{t+1}) = \delta\left(\mathbf{z}_{t+2}^T - [f_{t+1}, x_t, x_{t+1}]\right), \tag{2.70a}$$

$$f_{t+1} \sim p\left(f_{t+1}|\mathbf{y}_{1:t}\right). \tag{2.70b}$$

The joint predictive distribution from the time step $t+1$ up to $t+2$ is then defined by

$$p\left(\mathbf{f}_{t+1:t+2}|\mathbf{y}_{1:t}\right) = \int p\left(f_{t+2}|f_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{t+2}\right) p(\mathbf{z}_{t+2}|f_{t+1}) p\left(f_{t+1}|\mathbf{y}_{1:t}\right) d\mathbf{z}_{t+2}. \tag{2.71}$$

Similarly, at the time step $t+3$, the input is defined by

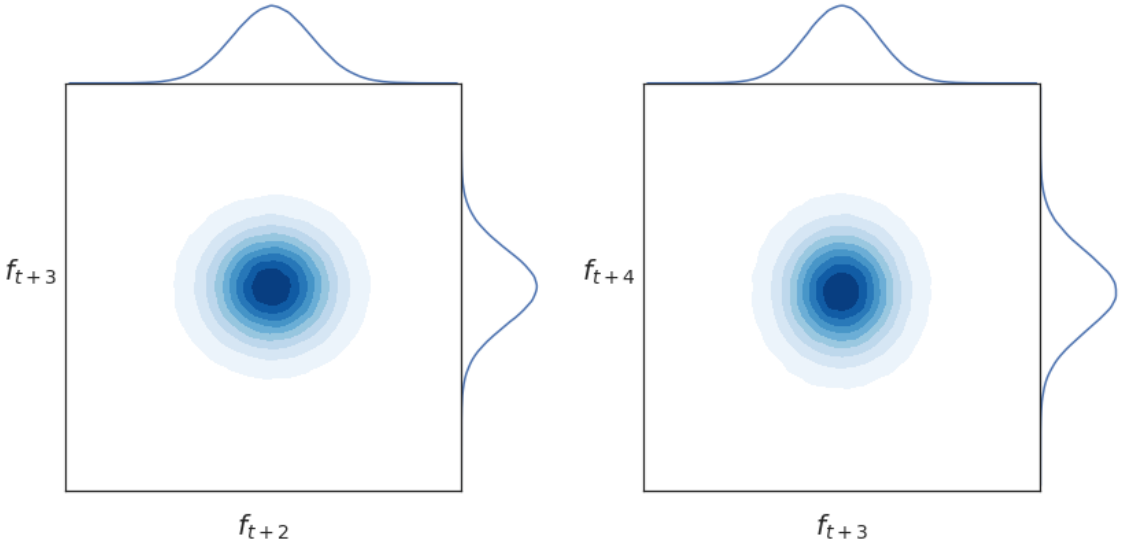
$$p(\mathbf{z}_{t+3}^T|f_{t+2}) = \delta\left(\mathbf{z}_{t+3}^T - [f_{t+2}, x_{t+1}, x_{t+2}]\right), \tag{2.72a}$$

$$f_{t+2} \sim p\left(f_{t+2}|f_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{t+2}\right). \tag{2.72b}$$

The joint predictive distribution from the time step $t+1$ up to $t+3$ is then

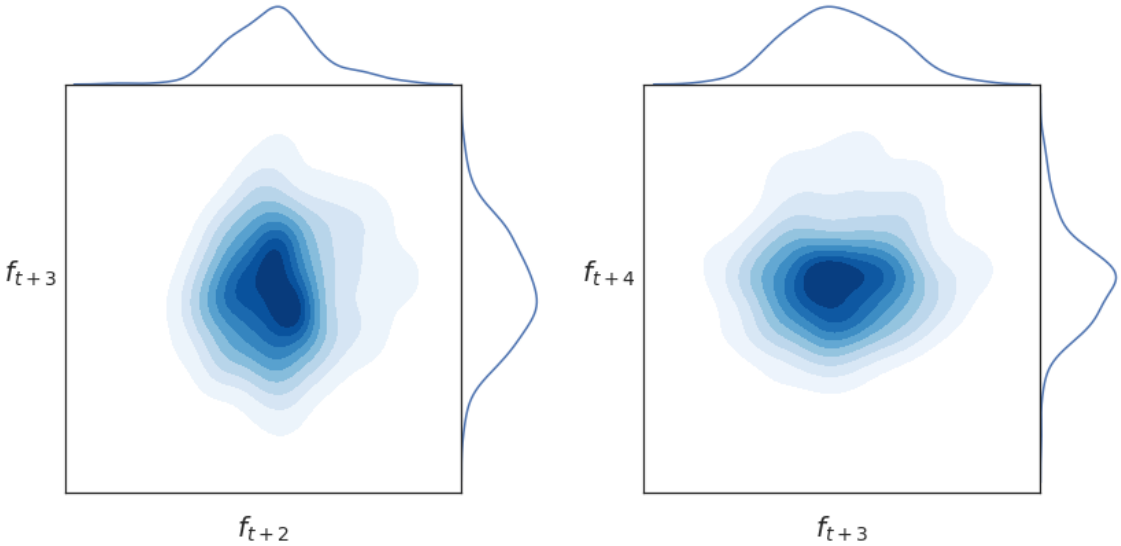
$$\begin{aligned} p\left(\mathbf{f}_{t+1:t+3}|\mathbf{y}_{1:t}\right) = \int\int & p\left(f_{t+3}|f_{t+2}, f_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{t+3}\right) p(\mathbf{z}_{t+3}|f_{t+2}) \cdot \\ & \cdot p\left(f_{t+2}|f_{t+1}, \mathbf{y}_{1:t}, \mathbf{z}_{t+2}\right) p(\mathbf{z}_{t+2}|f_{t+1}) p\left(f_{t+1}|\mathbf{y}_{1:t}\right) d\mathbf{z}_{t+2} d\mathbf{z}_{t+3}. \end{aligned} \tag{2.73}$$

The joint distributions that involve marginalization over the uncertain inputs cannot be evaluated in closed-form, since the Gaussian distribution is propagated through the non-linear covariance function. Therefore, the resulting distribution over the latent function values is not Gaussian anymore. The distinction of the joint posterior over the latent function values between the prediction and the simulation is presented in Figure 2.17 and Figure 2.18, where the joint distribution up to $t+4$ was considered. We can see that in the simulation, the joint distribution between the latent function values is not Gaussian



(a) $p\left(f_{t+3}, f_{t+2}|\mathbf{y}_{1:t}\right)$ (b) $p\left(f_{t+4}, f_{t+3}|\mathbf{y}_{1:t}\right)$

Figure 2.17: Posterior distribution of the consecutive latent function values in prediction, i.e. the inputs are known and are deterministic. The posterior can be obtained in closed-form and is Gaussian.



(a) $p\left(f_{t+3}, f_{t+2}|\mathbf{y}_{1:t}\right)$ (b) $p\left(f_{t+4}, f_{t+3}|\mathbf{y}_{1:t}\right)$

Figure 2.18: Posterior distribution of the consecutive latent function values in simulation, i.e. the inputs are obtained sequentially and are a random variable. The posterior cannot be obtained in closed-form and is not Gaussian.

anymore. Although the joint distribution is not Gaussian, the individual conditionals in Equation (2.73) are. We can approximate the marginalization over the uncertain inputs with MC integration. The joint distribution over the latent function values, generalized for arbitrary parameter value n_a, is defined by

$$p\left(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}\right) = \int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{y}_{1:t})\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\mathbf{f}_{t+1:t+i-1},\mathbf{y}_{1:t},\mathbf{z}_{t+i})\cdot \right. \\ \left. \cdot p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}}, \tag{2.74}$$

where

$$p(\mathbf{z}_{t+i}^T|\mathbf{f}_{t+i-n_a:t+i-1}) = \delta\left(\mathbf{z}_{t+i}^T - [\mathbf{f}_{t+i-n_a:t+i-1}^T, \mathbf{x}_{t+i-n_b:t+i-1}^T]\right). \tag{2.75}$$

A k-th trajectory sample is obtained sequentially by

$$\begin{aligned}
&\text{draw } \tilde{\mathbf{f}}_{t+2-n_a:t+1}^{(k)} \sim p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{y}_{1:t}),\\
&\text{draw } \tilde{\mathbf{z}}_{t+2}^{(k)} \sim p(\mathbf{z}_{t+2}|\tilde{\mathbf{f}}_{t+2-n_a:t+1}^{(k)}),\\
&\text{draw } \tilde{\mathbf{f}}_{t+2}^{(k)} \sim p(\mathbf{f}_{t+2}|\tilde{f}_{t+1}^{(k)},\mathbf{y}_{1:t},\tilde{\mathbf{z}}_{t+2}^{(k)}),\\
&\qquad\qquad\vdots\\
&\text{draw } \tilde{\mathbf{z}}_{t+n}^{(k)} \sim p(\mathbf{z}_{t+n}|\tilde{\mathbf{f}}_{t+n-n_a:t+n-1}^{(k)}),\\
&\text{draw } \tilde{\mathbf{f}}_{t+n}^{(k)} \sim p(\mathbf{f}_{t+n}|\tilde{\mathbf{f}}_{t+1:t+n-1}^{(k)},\mathbf{y}_{1:t},\tilde{\mathbf{z}}_{t+n}^{(k)}).
\end{aligned} \tag{2.76}$$

The noisy values can then be drawn independently

$$\text{draw } \tilde{\mathbf{y}}_{t+1:t+n}^{(k)} \sim p\left(\mathbf{y}_{t+1:t+n}|\tilde{\mathbf{f}}_{t+1:t+n}^{(k)}\right). \tag{2.77}$$

The equation is very similar to the sequential sampling in the static example, defined by Equation (2.53), only that now the order of the latent samples is important. It is determined by the sequential estimation of the inputs which, consequently, breaks the elegant Gaussian property of the latent function values. Posterior marginals that can be seen as a Gaussian mixture model (GMM) at an arbitrary time step $t+i$, where $i > 1$, can be defined by

$$p(\mathbf{f}_{t+i}|\mathbf{y}_{1:t}) \approx \frac{1}{r}\sum_{k=1}^{r} p(\mathbf{f}_{t+i}|\tilde{\mathbf{f}}_{t+1:t+i-1}^{(k)},\mathbf{y}_{1:t},\tilde{\mathbf{z}}_{t+i}^{(k)}), \tag{2.78}$$

where r denotes the number of MC samples.

PGM of a GP-NARX model simulation is presented in Figure 2.19. A block diagram of the simulation is presented in Figure 2.20. In the simulation, a dependency between the inputs and the delayed latent function values is introduced, which can be seen in the block diagram by connections between the latent values and future inputs.

A diagram of the simulation procedure is presented in Figure 2.21. Horizontally, the diagram represents a k-th MC sample of the trajectory, where it is demonstrated how the posterior latent distributions, at the time step considered, depend on all latent realizations up to that time step. Vertically, the diagram shows how the conditional distributions for individual samples define a GMM that approximates the posterior latent distribution at the respective time step.

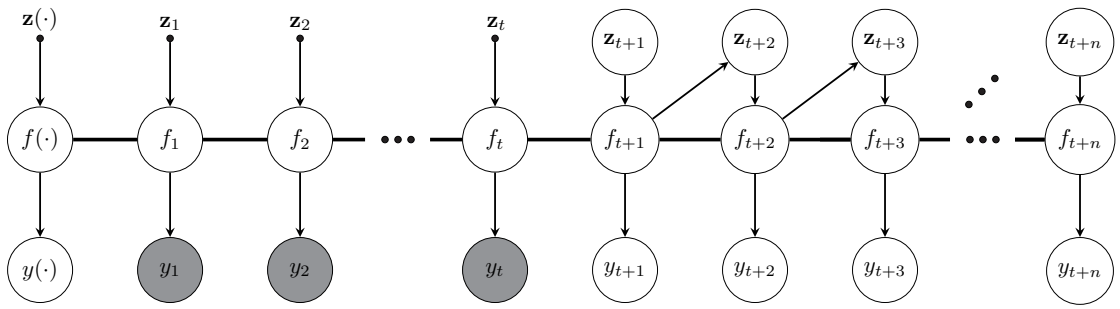


Figure 2.19: PGM of a GP-NARX model simulation. The PGM is presented for a NARX model with parameter $n_a = 1$. The thick black represents fully correlated latent values.

Noise propagation

If the assumed GP model does not separate the epistemic and the aleatoric uncertainty well, i.e. the noise σ_n^2 not only contains the random fluctuations, but also other possible unknowns that could in practice be reduced, one can also propagate the noise corrupted values through the inputs, i.e.

$$p(\mathbf{z}_{t+i}|\mathbf{y}_{t+i-n_a:t+i-1}) = \delta\left(\mathbf{z}_{t+i}^T - [\mathbf{y}_{t+i-n_a:t+i-1}^T, \mathbf{x}_{t+i-n_b:t+i-1}^T]\right). \tag{2.79}$$

Note that in practice this approach might overestimate the simulated variance since the whole σ_n^2 is propagated, even though some of it might just be random noise. And since the latent function is generally nonlinear, the posterior moments of the estimated distribution change as well. On the other hand, if only the latent values are propagated, the simulated variance might be underestimated, if the learned σ_n^2 contains other uncertainties than just random noise. In practice, the decision to propagate the noise or not is a design choice and should be best selected through cross-validation.

Comparing the predictive variances of the two approaches additionally gives valuable insight into the modeled dynamics. E.g., if the noise propagation approach describes the observations better in probabilistic terms, one can conclude that there is still uncertainty in the process that could in practice be reduced. New data could be acquired to model the latent process better and thus reduce the likelihood variance that should ideally only contain random fluctuations, i.e. noise.

The computational complexity of the simulation is unfortunately cubic with respect to the number of steps into the future. In the next section, we will provide an overview of approximations to the numerical simulation of GP-NARX models.

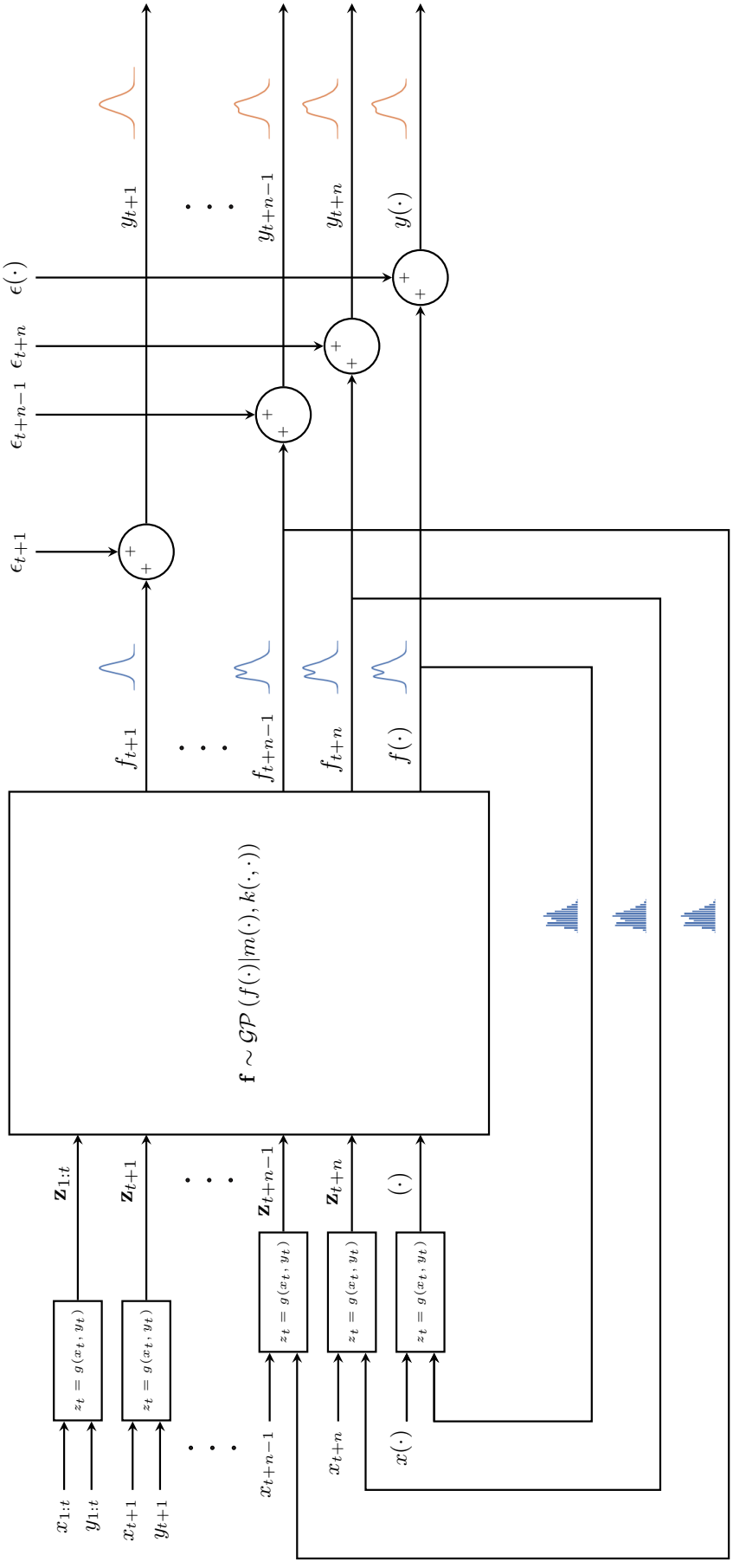


Figure 2.20: Block diagram of a GP-NARX model simulation. In simulation, the GP-NARX model introduces a dependency between the inputs $\mathbf{z}$ and the delayed latent function values $\mathbf{f}$. The transformation $g(x_t, y_t)$ represents the NARX model constructor, shown in Figure 2.14.

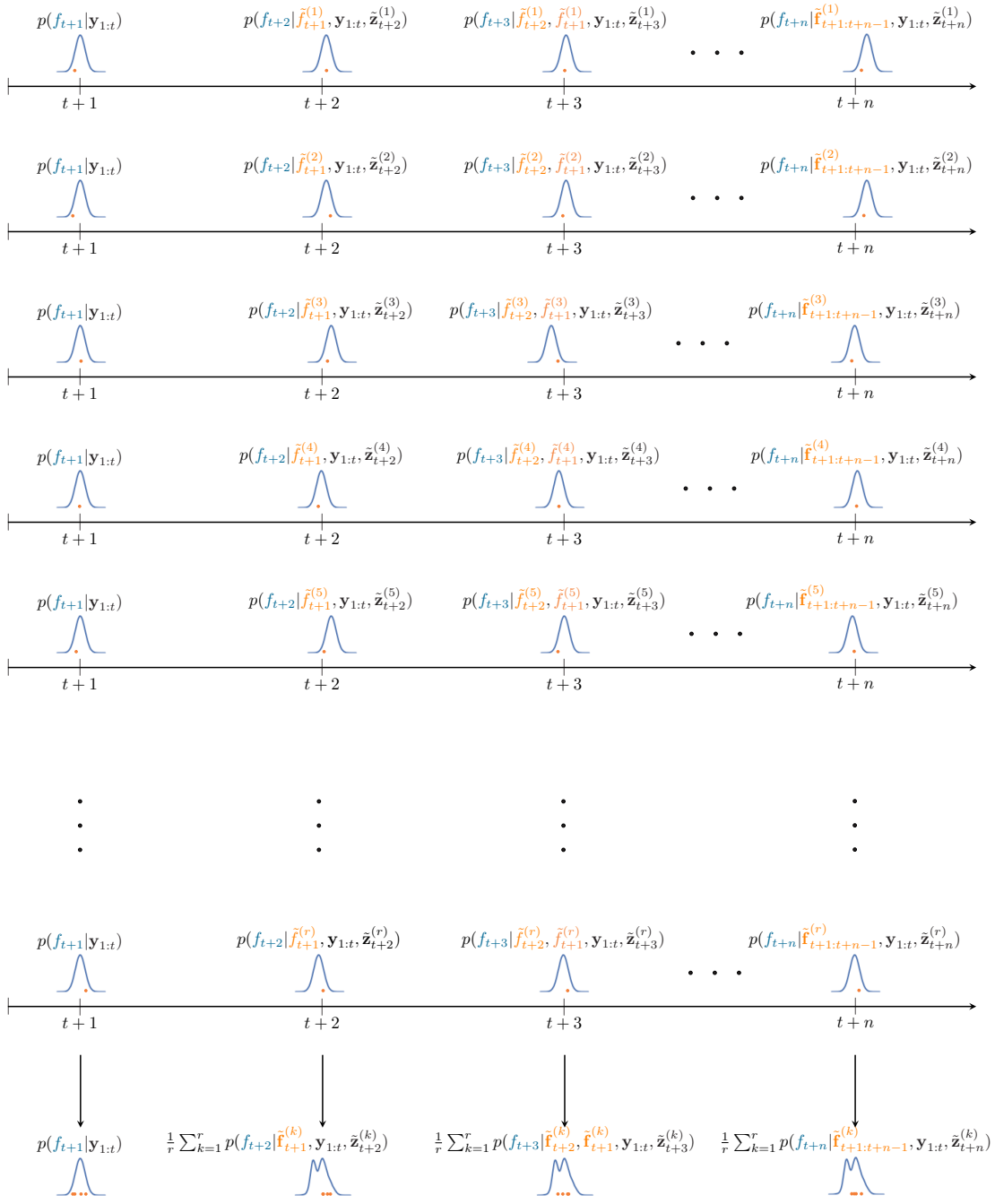


Figure 2.21: Diagram of MC approximation of the GP-NARX model simulation. Orange-filled circles represent the realizations at the time step considered, and the blue distributions show the individual conditional components. At the bottom of the diagram, a GMM representation for the latent function values at the time step considered is shown.

2.3.4 Simulation approximations

In the existing literature considering the simulation of autoregressive GP models, the simulation is often further approximated due to the high computational complexity of the algorithm. Here we will summarize the numerical approximations in relation to the ground truth. The ground truth will refer to the numerical approximation of the simulation defined by Equation (2.74) and Equation (2.75). We will call this approach the fully correlated MC (FCMC) simulation.

Conditionally independent MC (CIMC) simulation assumes that the latent function values up to the time step $t+n$ are conditionally independent given the inputs and the noisy observations, i.e.

$$\begin{aligned} p\left(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}\right) &= \int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{y}_{1:t})\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\cancel{\mathbf{f}_{t+1:t+i-1}},\mathbf{y}_{1:t},\mathbf{z}_{t+i})\cdot\right. \\ &\qquad \left.\cdot p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}} \\ &\approx \int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{y}_{1:t})\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\mathbf{y}_{1:t},\mathbf{z}_{t+i})\cdot\right. \\ &\qquad \left.\cdot p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}} \end{aligned} \tag{2.80}$$

The dependence on the past latent values is omitted. The uncertain inputs are defined by Equation (2.75).

Fully correlated naïve (FCN) simulation keeps the fully correlated latent distribution, defined by Equation (2.74), but considers only the mean of the prediction in the definition of the inputs. The inputs are, therefore, deterministic and numerical integration with respect to the latent values can be obtained in close-form, although sequentially. The inputs are defined by

$$\begin{aligned} p(\mathbf{z}_{t+i}^T|\mathbf{f}_{t+i-n_a:t+i-1}) &\approx \delta\left(\mathbf{z}_{t+i}^T - [\boldsymbol{\mu}_{t+i-n_a:t+i-1}^T, \mathbf{x}_{t+i-n_b:t+i-1}^T]\right), \\ \boldsymbol{\mu}_{t+i-n_a:t+i-1} &= \mathbb{E}\left[p(\mathbf{f}_{t+i-n_a:t+i-1}|\mathbf{y}_{1:t})\right]. \end{aligned} \tag{2.81}$$

Conditionally independent naïve (CIN) simulation further approximates the FCN simulation by considering the latent distribution defined by Equation (2.80) and the inputs defined by Equation (2.81).

Table 2.1 represents the computational complexities and the memory requirements of different approximations to the simulation of GP-NARX models. Parameter r represents the number of MC samples, t the number of observed time steps, and n the number of predicted steps into the future. We can see why the methods using the independence assumption are appealing in practice, since the computational complexity and the memory requirements are heavily reduced in approximations that either assume deterministic inputs, or do not consider all latent values jointly correlated. However, such assumptions at test time break the mathematical definition of GPs, and can result in severe underestimation of the latent uncertainty. On the other side, the computational demands and memory requirements of the FCMC simulation are too restrictive to be useful in practice.

So far the simulation of GP-NARX models was presented in a rather abstract way. In the next section, we will provide an illustrative example of modeling a dynamical system with a GP-NARX model. We will explicitly show how the simulation can be obtained in practice.

Table 2.1: Computational complexity and memory requirements of approximations for the simulation of GP-NARX models. The number of MC samples are denoted by r, the number of observed time steps with t, and the number of predicted steps into the future by n.

Approximation algorithm	Computational complexity	Memory requirements
FCMC simulation	$\mathcal{O}\left(r \cdot n \cdot (n+t)^2\right)$	$\mathcal{O}\left(r \cdot (n+t)^2\right)$
CIMC simulation	$\mathcal{O}\left(r \cdot n \cdot t^2\right)$	$\mathcal{O}\left(t^2\right)$
FCN simulation	$\mathcal{O}\left(n \cdot (n+t)^2\right)$	$\mathcal{O}\left((n+t)^2\right)$
CIN simulation	$\mathcal{O}\left(n \cdot t^2\right)$	$\mathcal{O}\left(t^2\right)$

2.4 Illustrative Example

Let us consider an illustrative problem [68], where the true process is governed by the difference equation

$$\begin{aligned} f_{t+1}^1 &= f_t^1 \exp\left(1 - 0.4 f_t^1 - \frac{\left(2 + 1.2 x_t^1\right) f_t^2}{1 + \left(f_t^1\right)^2}\right), \\ f_{t+1}^2 &= f_t^2 \exp\left(1 - 0.5 x_t^1 - \frac{\left(1.5 - x_t^2\right) f_t^2}{f_t^1}\right), \\ x_t^1 &= \cos\left(2\pi t\right), \\ x_t^2 &= \sin\left(2\pi t\right). \end{aligned} \tag{2.82}$$

We observe noisy values

$$\begin{aligned} y_t^1 &\sim \mathcal{N}\left(y_t^1 | f_t^1, \sigma_n^2\right), \\ y_t^2 &\sim \mathcal{N}\left(y_t^2 | f_t^2, \sigma_n^2\right), \end{aligned} \tag{2.83}$$

where $\sigma_n^2 = 0.05$. The process is modeled by

$$\mathbf{Y}_{1:t} = f(\mathbf{Z}_{1:t}) + \boldsymbol{\epsilon}, \tag{2.84}$$

where $f \sim \mathcal{GP}\left(f(\cdot) | m(\cdot), k(\cdot,\cdot)\right)$. The row of the input matrix $\mathbf{Z}_{1:t}$ is defined by a NARX model, where $n_a = 1$ and $n_b = 2$, i.e.

$$\mathbf{z}_t^T = \begin{bmatrix} y_{t-1}^1 & y_{t-1}^2 & x_{t-1}^1 & x_{t-2}^1 & x_{t-1}^2 & x_{t-2}^2 \end{bmatrix}, \tag{2.85}$$

and the corresponding row of the output matrix $\mathbf{Y}_{1:t} = \begin{bmatrix} \mathbf{y}_{1:t}^1 & \mathbf{y}_{1:t}^2 \end{bmatrix}$ is defined by

$$\mathbf{y}_t^T = \begin{bmatrix} y_t^1 & y_t^2 \end{bmatrix}. \tag{2.86}$$

We assume that the joint posterior of the two outputs is conditionally independent given the observed data, i.e.

$$p\left(\mathbf{f}_{1:t}^1, \mathbf{f}_{1:t}^2 | \mathbf{Y}_{1:t}, \mathbf{Z}_{1:t}\right) = p\left(\mathbf{f}_{1:t}^1 | \mathbf{y}_{1:t}^1, \mathbf{Z}_{1:t}\right) p\left(\mathbf{f}_{1:t}^2 | \mathbf{y}_{1:t}^2, \mathbf{Z}_{1:t}\right). \tag{2.87}$$

Then their joint prior has the form

$$p\left(\begin{bmatrix}\mathbf{f}^1_{1:t}\\ \mathbf{f}^2_{1:t}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}^1_{1:t}\\ \mathbf{f}^2_{1:t}\end{bmatrix} \middle| \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}^1_{f_{1:t},f_{1:t}} & \mathbf{0}\\ \mathbf{0} & \mathbf{K}^2_{f_{1:t},f_{1:t}}\end{bmatrix}\right), \tag{2.88}$$

where the blocks $\mathbf{K}^i_{f_{1:t},f_{1:t}}$ are parametrized by a separate covariance function $k^i(\cdot,\cdot)$. The noise variance is similarly modeled separately for each output dimension, i.e. $\left(\sigma_n^2\right)^i$. For this example, the covariance function was chosen as a combination of a linear and a Matérn($\frac{5}{2}$) kernel such that

$$k^i(\cdot,\cdot) = k^i_{\text{Linear}}(\cdot,\cdot) + k^i_{\text{Matérn}(\frac{5}{2})}(\cdot,\cdot). \tag{2.89}$$

The covariance functions are defined in Appendix C.

Note that the cross terms in the distribution defined by Equation (2.88) could also be populated [94]–[96], instead of being set to $\mathbf{0}$. This increases the joint distribution by 2 times and the computational complexity by 8 times due to the cubic dependency with respect to the number of data points. In practice, this can be best utilized if the data are partially observed, e.g. we are trying to predict one output having observed the other.

In this example, we will assume that the whole time-series is observed and that the dependency between the outputs can be described well by having both lagged outputs in the NARX model. We would expect this to be a good enough approximation when the discrete system is sampled with a relatively high frequency.

We are interested in predicting the process output up to some arbitrary time step into the future $t+n$. We follow Equation (2.74). The input at $t+1$ is defined by

$$\mathbf{z}^T_{t+1} = \begin{bmatrix} y^1_t & y^2_t & x^1_t & x^1_{t-1} & x^2_t & x^2_{t-1}\end{bmatrix}. \tag{2.90}$$

In the first step, the latent values are drawn from the posterior distribution of the latent function values given the data we have observed up to the time step t, i.e.

$$\text{draw } \tilde{f}^1_{t+1} \sim \mathcal{N}(f^1_{t+1}|\mu^1_{t+1}, \Sigma^1_{t+1}), \tag{2.91a}$$
$$\text{draw } \tilde{f}^2_{t+1} \sim \mathcal{N}(f^2_{t+1}|\mu^2_{t+1}, \Sigma^2_{t+1}), \tag{2.91b}$$

where

$$\mu_{t+1,i} = \mathbf{K}^i_{f_{t+1},f_{1:t}}\left(\mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}_{1:t,i}, \tag{2.92a}$$
$$\Sigma_{t+1,i} = \mathbf{K}^i_{f_{t+1},f_{t+1}} - \mathbf{K}^i_{f_{t+1},f_{1:t}}\left(\mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}^i_{f_{1:t},f_{t+1}}. \tag{2.92b}$$

$\mathbf{K}^i_{f_{1:t},f_{1:t}}$ represents the covariance matrix for $\mathbf{f}_{1:t,i}$, and $\mathbf{K}^i_{f_{t+1},f_{1:t}}$ represents the cross-covariance between $\mathbf{f}_{1:t,i}$ and $f_{t+1,i}$. The input at $t+2$ is defined by

$$\tilde{\mathbf{z}}^T_{t+2} = \begin{bmatrix}\tilde{f}^1_{t+1} & \tilde{f}^2_{t+1} & x^1_{t+1} & x^1_t & x^2_{t+1} & x^2_t\end{bmatrix} \tag{2.93}$$

and the next latent realization is obtained, i.e.

$$\text{draw } \tilde{f}^1_{t+2} \sim \mathcal{N}(f^1_{t+2}|\mu^1_{t+2}, \Sigma^1_{t+2}), \tag{2.94a}$$
$$\text{draw } \tilde{f}^2_{t+2} \sim \mathcal{N}(f^2_{t+2}|\mu^2_{t+2}, \Sigma^2_{t+2}). \tag{2.94b}$$

The latent posterior at $t+2$ is specified by

$$\mu_{t+2,i} = \tilde{\mathbf{K}}^i_{f_{t+2},f_{1:t+1}} \begin{bmatrix} \mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2 & \mathbf{K}^i_{f_{1:t},f_{t+1}} \\ \mathbf{K}^i_{f_{t+1},f_{1:t}} & \mathbf{K}^i_{f_{t+1},f_{t+1}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y}_{1:t,i} \\ \tilde{f}_{t+1,i} \end{bmatrix}, \tag{2.95a}$$

$$\Sigma_{t+2,i} = \tilde{\mathbf{K}}^i_{f_{t+2},f_{t+2}} - \tilde{\mathbf{K}}^i_{f_{t+2},f_{1:t+1}} \begin{bmatrix} \mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2 & \mathbf{K}^i_{f_{1:t},f_{t+1}} \\ \mathbf{K}^i_{f_{t+1},f_{1:t}} & \mathbf{K}^i_{f_{t+1},f_{t+1}} \end{bmatrix}^{-1} \tilde{\mathbf{K}}^i_{f_{1:t+1},f_{t+2}}. \tag{2.95b}$$

The input for the time step $t+3$ is defined by

$$\tilde{\mathbf{z}}^T_{t+3} = \begin{bmatrix} \tilde{f}^1_{t+2} & \tilde{f}^2_{t+2} & x^1_{t+2} & x^1_{t+1} & x^2_{t+2} & x^2_{t+1} \end{bmatrix} \tag{2.96}$$

and the next latent realization is obtained by

$$\text{draw } \tilde{f}^1_{t+3} \sim \mathcal{N}(f^1_{t+3} | \mu^1_{t+3}, \Sigma^1_{t+3}), \tag{2.97a}$$

$$\text{draw } \tilde{f}^2_{t+3} \sim \mathcal{N}(f^2_{t+3} | \mu^2_{t+3}, \Sigma^2_{t+3}), \tag{2.97b}$$

where

$$\mu_{t+3,i} = \tilde{\mathbf{K}}^i_{f_{t+3},f_{1:t+2}} \underbrace{\begin{bmatrix} \mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2 & \mathbf{K}^i_{f_{1:t},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{1:t},f_{t+2}} \\ \mathbf{K}^i_{f_{t+1},f_{1:t}} & \mathbf{K}^i_{f_{t+1},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{t+1},f_{t+2}} \\ \tilde{\mathbf{K}}^i_{f_{t+2},f_{1:t}} & \tilde{\mathbf{K}}^i_{f_{t+2},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{t+2},f_{t+2}} \end{bmatrix}^{-1}}_{\left[\tilde{\mathbf{K}}^i_{f_{1:t+2},f_{1:t+2}}\right]^{-1}} \underbrace{\begin{bmatrix} \mathbf{y}_{1:t,i} \\ \tilde{f}_{t+1,i} \\ \tilde{f}_{t+2,i} \end{bmatrix}}_{\eta_{1:t+2,i}}, \tag{2.98a}$$

$$\Sigma_{t+3,i} = \tilde{\mathbf{K}}^i_{f_{t+3},f_{t+3}} - \tilde{\mathbf{K}}^i_{f_{t+3},f_{1:t+2}} \underbrace{\begin{bmatrix} \mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2 & \mathbf{K}^i_{f_{1:t},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{1:t},f_{t+2}} \\ \mathbf{K}^i_{f_{t+1},f_{1:t}} & \mathbf{K}^i_{f_{t+1},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{t+1},f_{t+2}} \\ \tilde{\mathbf{K}}^i_{f_{t+2},f_{1:t}} & \tilde{\mathbf{K}}^i_{f_{t+2},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{t+2},f_{t+2}} \end{bmatrix}^{-1}}_{\left[\tilde{\mathbf{K}}^i_{f_{1:t+2},f_{1:t+2}}\right]^{-1}} \tilde{\mathbf{K}}^i_{f_{1:t+2},f_{t+3}}. \tag{2.98b}$$

This procedure can be repeated up to an arbitrary time step into the future.

Figure 2.22 and Figure 2.23 show the FCMC simulation on the illustrative example where the data were observed up to the time step 500 and then the trajectory was simulated for another 100 steps. If we look at Equation (2.98), we can quickly realize that the dimension of the matrix we are inverting is increased with each additional time step considered.

To reduce the computational complexity one can consider only propagating the mean of the posterior of the latent function values. FCN simulation would replace the latent realizations with their corresponding posterior means. In the example presented above, the input at time step t+3 would then be defined by

$$\mathbf{z}^T_{t+3} = \begin{bmatrix} \mu^1_{t+2} & \mu^2_{t+2} & x^1_{t+2} & x^1_{t+1} & x^2_{t+2} & x^2_{t+1} \end{bmatrix} \tag{2.99}$$

and

$$\eta_{1:t+2,i} = \begin{bmatrix} \mathbf{y}_{1:t,i} \\ \mu_{t+1,i} \\ \mu_{t+2,i} \end{bmatrix}. \tag{2.100}$$

CIN simulation would consider an additional assumption and set the covariance matrix elements to

$$\begin{aligned}\tilde{\mathbf{K}}^i_{f_{1:t+3},f_{1:t+3}} &= \begin{bmatrix}\tilde{\mathbf{K}}^i_{f_{1:t+2},f_{1:t+2}} & \tilde{\mathbf{K}}^i_{f_{1:t+2},f_{t+3}} \\ \tilde{\mathbf{K}}^i_{f_{t+3},f_{1:t+2}} & \tilde{\mathbf{K}}^i_{f_{t+3},f_{t+3}}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{K}^i_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2 & \mathbf{K}^i_{f_{1:t},f_{t+1}} & \tilde{\mathbf{K}}^i_{f_{1:t},f_{t+2}} & \tilde{\mathbf{K}}^i_{f_{1:t},f_{t+3}} \\ \mathbf{K}^i_{f_{t+1},f_{1:t}} & \mathbf{K}^i_{f_{t+1},f_{t+1}} & \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{K}}^i_{f_{t+2},f_{1:t}} & \mathbf{0} & \tilde{\mathbf{K}}^i_{f_{t+2},f_{t+2}} & \mathbf{0} \\ \tilde{\mathbf{K}}^i_{f_{t+3},f_{1:t}} & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{K}}^i_{f_{t+3},f_{t+3}}\end{bmatrix}\end{aligned} \tag{2.101}$$

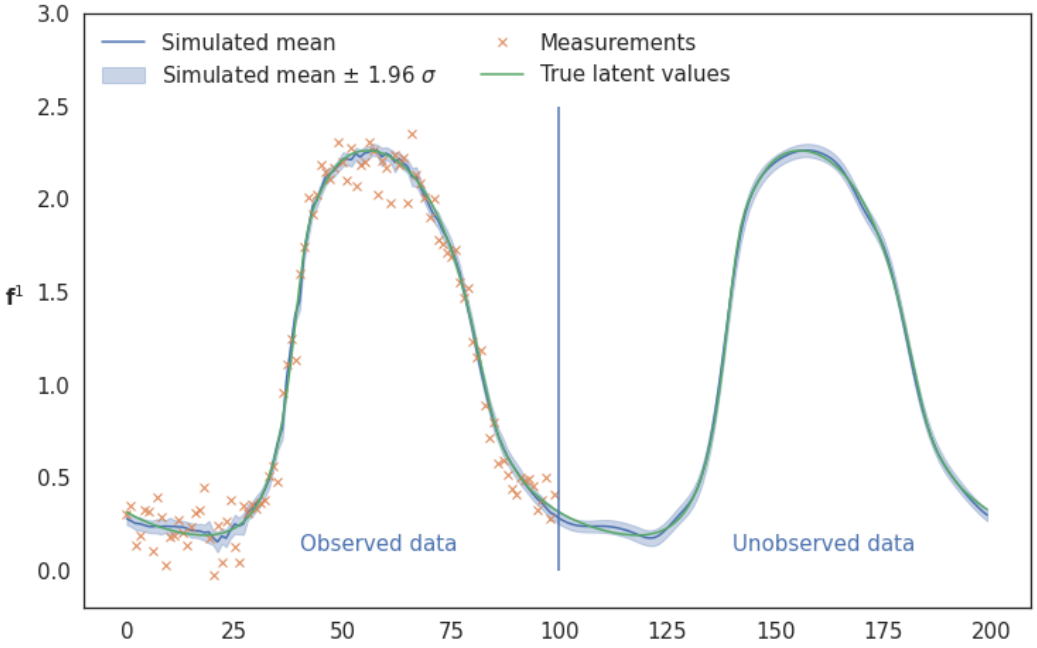
CIMC simulation would do the same approximation to the covariance matrix but introduce back the propagation of the latent uncertainty, i.e. propagate the latent realizations instead of the posterior mean, i.e.

$$\tilde{\mathbf{z}}^T_{t+3} = \begin{bmatrix}\tilde{f}^1_{t+2} & \tilde{f}^2_{t+2} & x^1_{t+2} & x^1_{t+1} & x^2_{t+2} & x^2_{t+1}\end{bmatrix}. \tag{2.102}$$

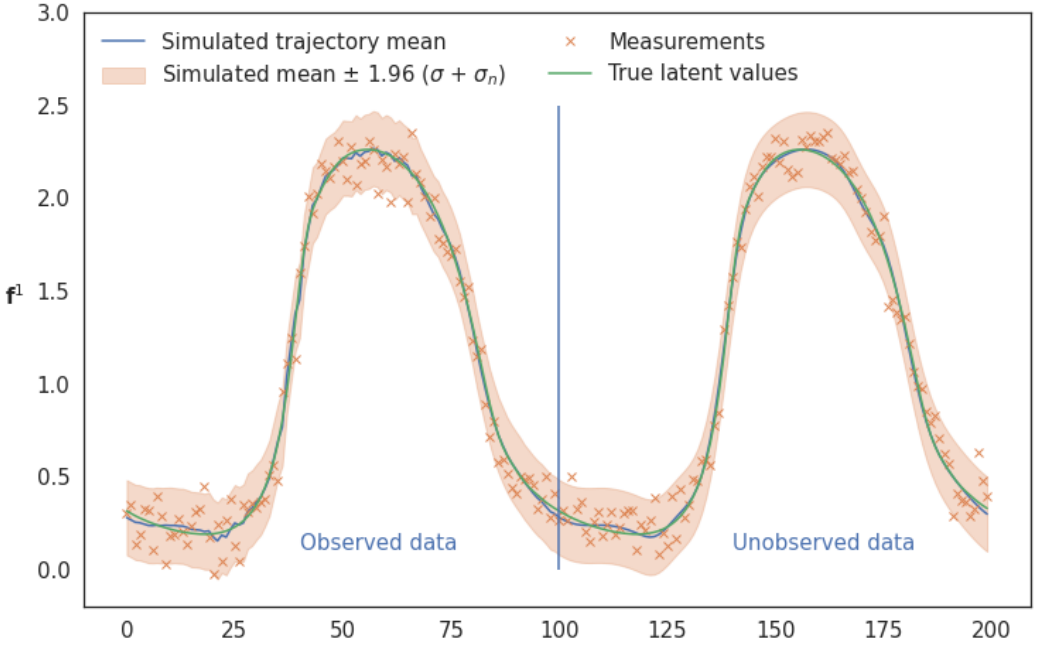
The quantified uncertainty for different approximations is presented in Figure 2.24, where we can see that in some regions, the approximations underestimate the variance, whereas in other regions, the variance is overestimated when compared to the FCMC simulation. The FCN simulation is omitted from the graph because it severely underestimated the variance.

Although the CIN simulation and CIMC simulation performed similarly in this illustrative example, this is generally not the case. The CIN simulation usually underestimates the latent variance [23]. Since the computational complexity of the two algorithms is not significantly different, as shown in Table 2.1, the CIMC approximation is generally preferred over the CIN simulation.

The example only considered 500 observed data points. The simulation was calculated for 100 steps into the future. Practically, GP-NARX regression can only be applied to data sets of a few 1000 observed data points, and even less if simulation is considered, due to the cubic computational complexity with respect to the observed data and predicted steps into the future. In the next chapter, we will consider approximations of the vanilla GP-NARX model, which can reduce the computational complexity in training and in simulation.

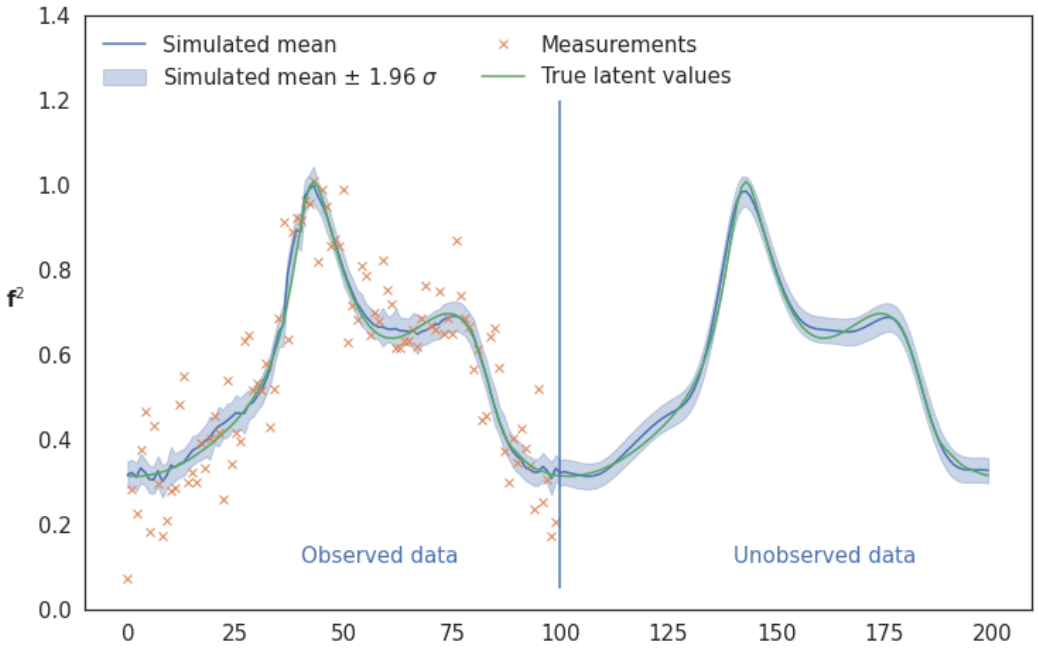


(a) Latent simulation response, i.e. $\mathbf{f}^1_{t+1:t+100}$.

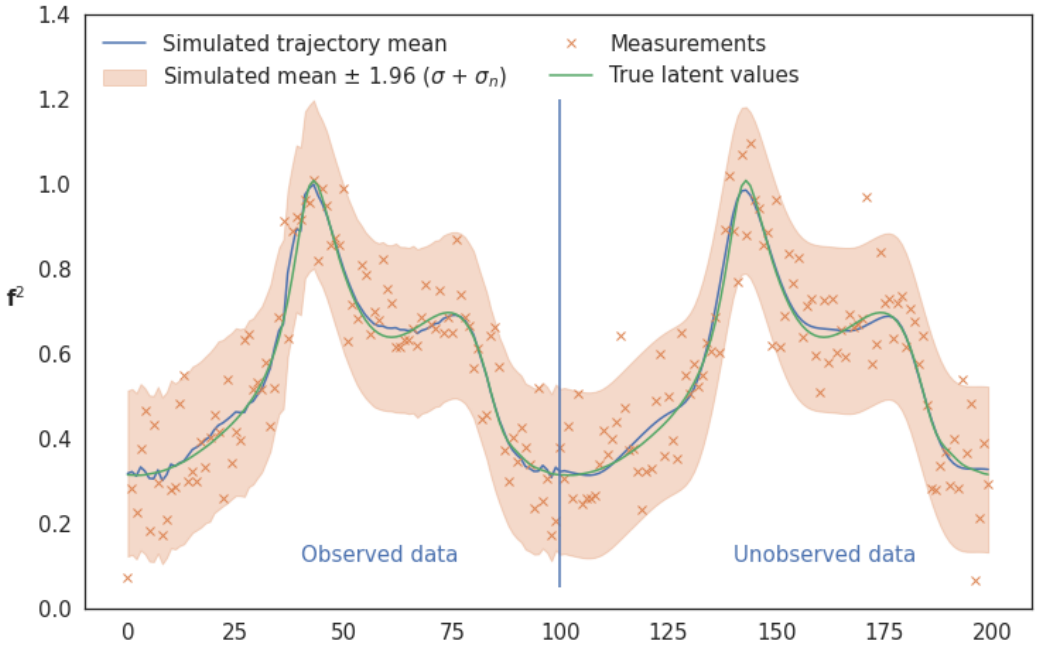


(b) Noisy simulated response for, i.e. $\mathbf{y}^1_{t+1:t+100}$.

Figure 2.22: The data are observed up to the time step $t = 500$. Only the last 100 training data are plotted. The output is then simulated 100 steps into the future. Figure 2.22a shows the first two moments of the latent simulated trajectory, and Figure 2.22b shows the first two moments of the noisy trajectory.

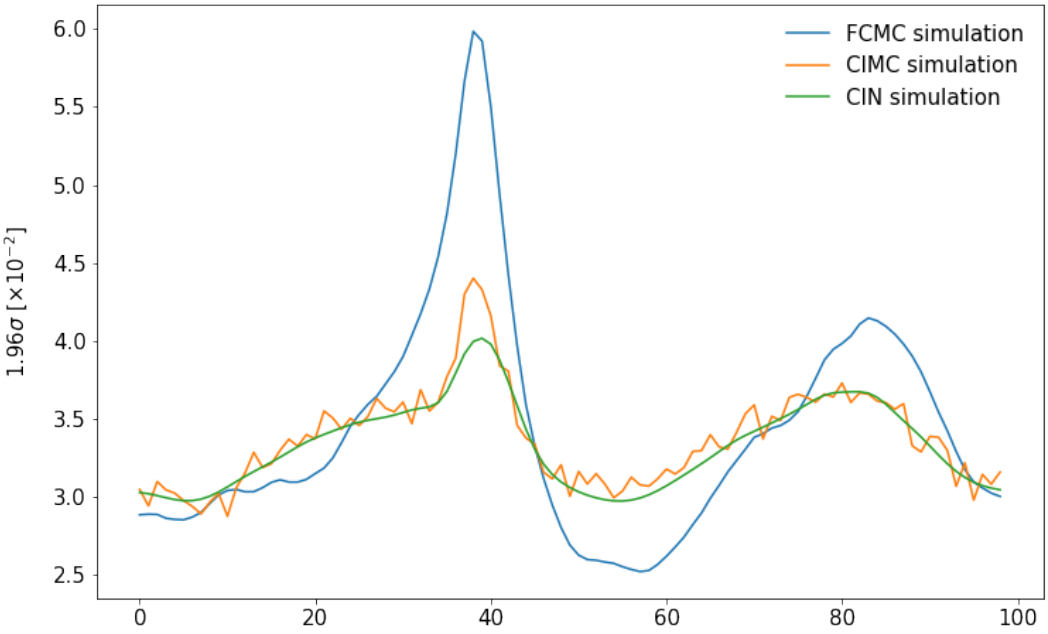


(a) Latent simulation response for, i.e. $\mathbf{f}^2_{t+1:t+100}$.

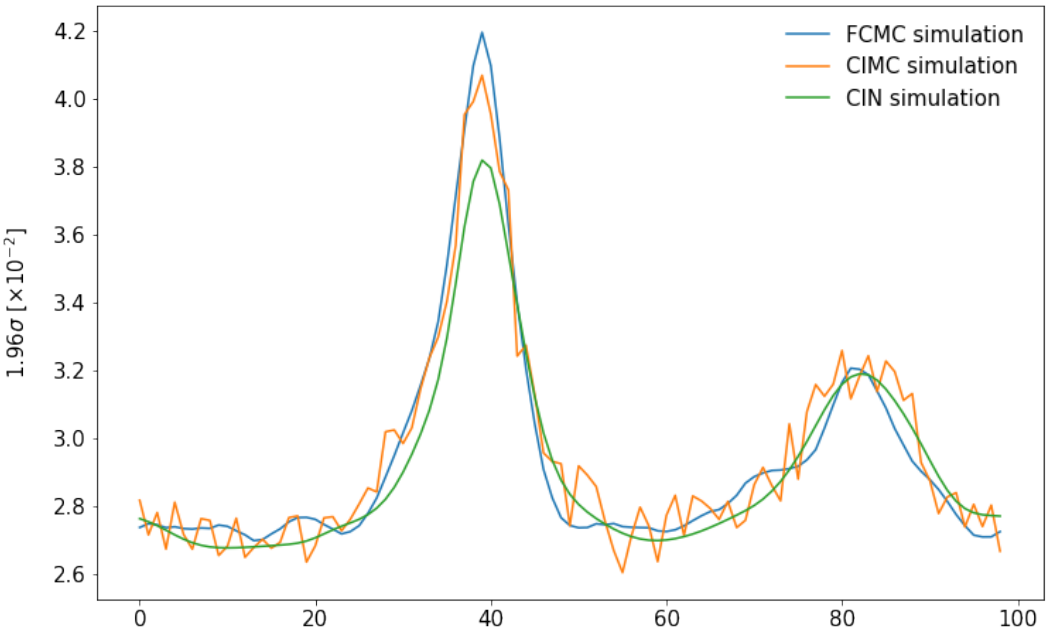


(b) Noisy simulated response for, i.e. $\mathbf{f}^2_{t+1:t+100}$.

Figure 2.23: The data are observed up to the time step $t = 500$. Only the last 100 training data are plotted. The output is then simulated 100 steps into the future. Figure 2.23a shows the first two moments of the latent simulated trajectory, and Figure 2.23b shows the first two moments of the noisy trajectory.



(a) Latent 1.96 standard deviations for $\mathbf{f}^1_{t+1:t+100}$.



(b) Latent 1.96 standard deviations for $\mathbf{f}^2_{t+1:t+100}$.

Figure 2.24: Comparison of the latent 1.96 standard deviations of the resulting trajectory in simulation of an illustrative dynamical example.

Chapter 3

Pseudo-Point Autoregressive Gaussian Process Regression

GP models represent an elegant method that describes both the aleatoric and the epistemic uncertainty in closed-form. The downside of the framework is its cubical computational complexity and quadratic memory requirements. Unfortunately, the computational complexity and memory requirements extend also to the simulation. This chapter introduces sparse approximations of GP-NARX models which significantly reduce the computational complexity of modeling with GPs for large data.

3.1 Sparse Autoregressive Gaussian Process Regression

In this section, we will introduce approximated GP models based on the idea of pseudo-points. We seek to find representative m pseudo-inputs that can describe the data well [34]. This can be seen as a dimensionality reduction, but not in the sense of the feature reduction, i.e. the column space of the input matrix, but in the sense of data points reduction, i.e. the row space of the input matrix. We consider m pseudo-inputs

$$\mathbf{z}'_{1:m} = [z'_1, \ldots, z'_m] \tag{3.1}$$

and the corresponding vector of latent function values

$$\mathbf{u} = [u_1, \ldots, u_m], \tag{3.2}$$

which come from the same joint distribution as $\mathbf{f}_{1:t}$, f_{t+1}, and $f(\cdot)$. This representation is not different from the vanilla GP model, where $\mathbf{u}$ are implicitly present but marginalized out of the joint distribution. For the model to be useful, the latent values $\mathbf{f}_{1:t}$, f_{t+1}, and $f(\cdot)$ are assumed to be conditionally independent given $\mathbf{u}$ such that the joint prior between the latent function values is defined by

$$p(\mathbf{f}_{1:t}, f_{t+1}, f(\cdot)) \approx \int p(\mathbf{f}_{1:t}|\mathbf{u}) p(f_{t+1}, f(\cdot)|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}. \tag{3.3}$$

The finite joint distribution is then represented by marginalizing all latent values that are not of interest, i.e.

$$\begin{aligned} p(\mathbf{f}_{1:t}, f_{t+1}) &= \int p(\mathbf{f}_{1:t}, f_{t+1}, f(\cdot)) d\mathbf{f}_{\setminus\{\mathbf{f}_{1:t}, f_{t+1}\}} \\ &= \int p(\mathbf{f}_{1:t}|\mathbf{u}) p(f_{t+1}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}, \end{aligned} \tag{3.4}$$

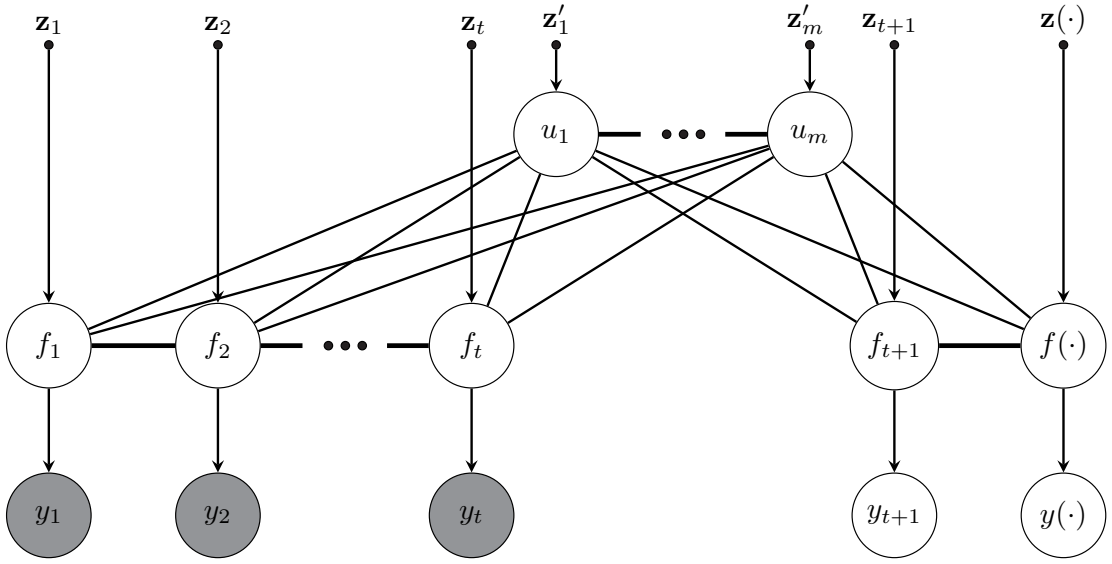


Figure 3.1: PGM of a sparse GP regression model, where the latent values $f_{t+1}, f(\cdot)$, and the latent values $\mathbf{f}_{1:t}$ are conditionally independent given the pseudo-points $\mathbf{u}$. The inputs $\mathbf{z}'_{1:m}$ corresponding to pseudo-points $\mathbf{u}$ are free model parameters. This model does not provide any computational advantages compared to the vanilla GP without further assumptions.

which can again be done analytically by simply ignoring those latent function values, since the joint distribution is Gaussian. The conditional distributions given pseudo-points fully specify the joint Gaussian distribution. They are defined by

$$p(\mathbf{f}_{1:t}|\mathbf{u}) = \mathcal{N}(\mathbf{f}_{1:t}|\mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{u}, \mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}) \tag{3.5a}$$

$$p(f_{t+1}|\mathbf{u}) = \mathcal{N}(f_{t+1}|\mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{u}, \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{Q}_{f_{t+1},f_{t+1}}), \tag{3.5b}$$

where $\mathbf{Q}_{ab} = \mathbf{K}_{a,u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,b}$. Figure 3.1 depicts a PGM of the sparse GP model considered up to this part. Unfortunately, this approximation still does not reduce the computational complexity and we have to consider further assumptions.

3.1.1 Fully independent training conditional approximation

The fully independent training conditional (FITC) approximation considers an exact test conditional $p(f_{t+1}, f(\cdot)|\mathbf{u})$, but an approximated training conditional where

$$p(\mathbf{f}_{1:t}|\mathbf{u}) \approx q(\mathbf{f}_{1:t}|\mathbf{u}) = \prod_{i=1}^{t} p(f_i|\mathbf{u}). \tag{3.6}$$

The latent values that belong to the training data are considered conditionally independent given the pseudo-points $\mathbf{u}$ [37]. Fully independent conditional approximation (FIC) extends this approximation to the test conditional. The conditionals of the FITC approximation are given by

$$q(\mathbf{f}_{1:t}|\mathbf{u}) = \mathcal{N}(\mathbf{f}_{1:t}|\mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{u}, \operatorname{diag}\left[\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}\right]), \tag{3.7a}$$

$$p(f_{t+1}|\mathbf{u}) = \mathcal{N}(f_{t+1}|\mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{u}, \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{Q}_{f_{t+1},f_{t+1}}). \tag{3.7b}$$

Figure 3.2 depicts a PGM of the FITC approximation. Compared to the sparse PGM presented in Figure 3.1, the connections between the training latent values $\mathbf{f}_{1:t}$ are omitted.

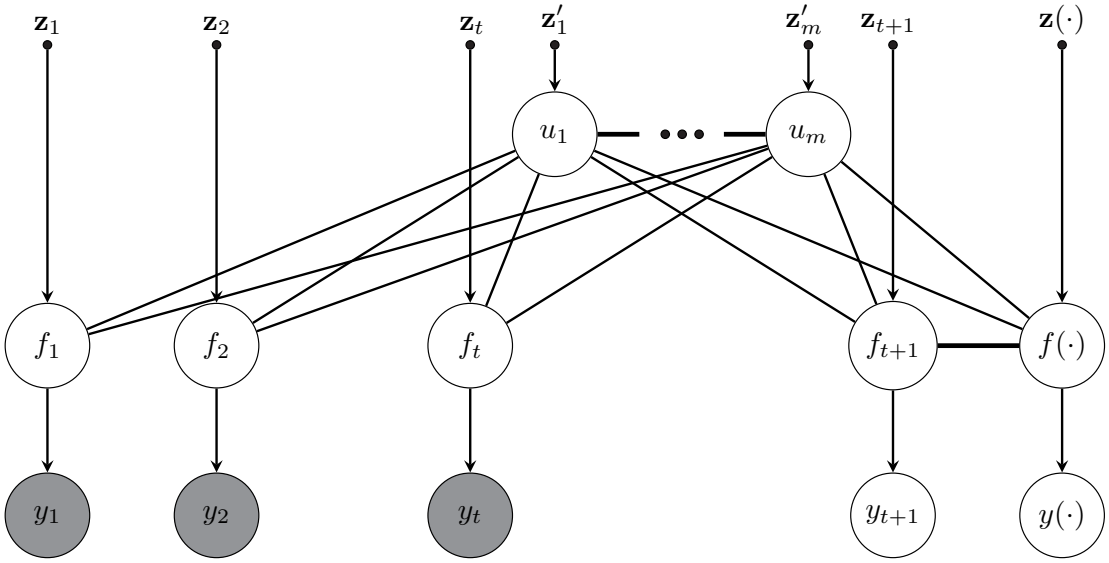


Figure 3.2: PGM of the FITC approximation of the sparse model, where the conditional of the latent function values $f_{t+1}, f(\cdot)$ is considered exact. The conditional of the latent values $\mathbf{f}_{1:t}$ is approximated so that individual latent values are independent given the pseudo-inputs $\mathbf{u}$. The latent function values $\mathbf{f}_{1:t}$ are not directly connected, which can be seen in the PGM.

Marginal log-likelihood

The marginal distribution of the latent function values results in a covariance matrix $\mathbf{K}_{f_{1:t},f_{1:t}}$ with exact values on the diagonal, while the cross terms are approximated, i.e.

$$q(\mathbf{f}_{1:t}) = \mathcal{N}(\mathbf{f}_{1:t}|\mathbf{0}, \mathbf{Q}_{f_{1:t},f_{1:t}} - \text{diag}\left[\mathbf{Q}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\right]), \tag{3.8}$$

where $q(\mathbf{f}_{1:t}) = \int \left(\prod_{i=1}^{t} p(f_i|\mathbf{u})\right) p(\mathbf{u}) d\mathbf{u}$. The noisy marginal follows by

$$q(\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{y}_{1:t}|\mathbf{0}, \mathbf{Q}_{f_{1:t},f_{1:t}} - \text{diag}\left[\mathbf{Q}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\right] + \mathbf{I}\sigma_n^2), \tag{3.9}$$

since $p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) = \mathcal{N}(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}, \mathbf{I}\sigma_n^2)$. The explicit definition of the MLL is then defined by

$$\begin{aligned} \log q\left(\mathbf{y}_{1:t}\right) = &-\frac{1}{2}\mathbf{y}_{1:t}^T \left(\mathbf{Q}_{f_{1:t},f_{1:t}} - \text{diag}\left[\mathbf{Q}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\right] + \mathbf{I}\sigma_n^2\right)^{-1} \mathbf{y}_{1:t} \\ &- \frac{1}{2}\log\left|\mathbf{Q}_{f_{1:t},f_{1:t}} - \text{diag}\left[\mathbf{Q}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\right] + \mathbf{I}\sigma_n^2\right| - \frac{n}{2}\log 2\pi. \end{aligned} \tag{3.10}$$

From here on we will refer to this approximation as the GP-NARX (FITC) model. FITC approximation reduces the computational complexity in training to $\mathcal{O}(tm^2)$, where t and m represent the number of observed steps and the number of inducing points, respectively. The memory requirements are reduced to $\mathcal{O}(m^2)$ since the biggest matrix we have to store in memory is the full covariance matrix over the pseudo-points $\mathbf{u}$, i.e. $\mathbf{K}_{u,u}$. Note that quadratic complexity is introduced over the number of inducing points which is a user-defined parameter.

Predictive distribution

The posterior of the latent function values given the observed data $q(f_{t+1}|\mathbf{y}_{1:t})$ is fully specified by

$$\mathbb{E}[f_{t+1}|\mathbf{y}_{1:t}] = \mathbf{K}_{f_{t+1},u}\boldsymbol{\Sigma}\mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}^{-1}\mathbf{y}_{1:t}, \tag{3.11a}$$

$$\mathbb{V}[f_{t+1}|\mathbf{y}_{1:t}] = \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{Q}_{f_{t+1},f_{t+1}} + \mathbf{K}_{f_{t+1},u}\boldsymbol{\Sigma}\mathbf{K}_{u,f_{t+1}}, \tag{3.11b}$$

where

$$\begin{aligned}\boldsymbol{\Sigma} &= \left(\mathbf{K}_{u,u} + \mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\right)^{-1}, \\ \boldsymbol{\Lambda} &= \operatorname{diag}\left[\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right].\end{aligned} \tag{3.12}$$

Equivalent, but numerically more stable definition, is defined in Appendix E.1.2. For completeness, and to reference in later sections, the posterior of the latent function values $\mathbf{u}$, given the observed data $q(\mathbf{u}|\mathbf{y}_{1:t})$, is specified by

$$\mathbb{E}[\mathbf{u}|\mathbf{y}_{1:t}] = \mathbf{K}_{u,u}\boldsymbol{\Sigma}\mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}^{-1}\mathbf{y}_{1:t}, \tag{3.13a}$$

$$\mathbb{V}[\mathbf{u}|\mathbf{y}_{1:t}] = \mathbf{K}_{u,u}\boldsymbol{\Sigma}\mathbf{K}_{u,u}. \tag{3.13b}$$

Numerically stable implementation of the pseudo-point posterior can be found in Appendix E.2.2. The predictive distribution for a single point can be obtained in $\mathcal{O}(m^2)$.

Unfortunately, the assumption of the conditionally independent latent values $\mathbf{f}_{1:t}$ given the pseudo-points $\mathbf{u}$ within the FITC framework is too restrictive to even view the model as "an approximation" to the GP model. It can be considered as a completely new model since the difference between the FITC approximation and the vanilla GP model is never minimized.

This causes many pathological problems when considering the FITC approximation. Some of them are even interpreted as an advantage of the method (and they can be in special cases where the noise is heteroskedastic [34], [35], [37]). In the next section, we will introduce the variational approximations which approximate the vanilla GP model more systematically. Variational approximations rigorously define and minimize the distance between the approximated GP posterior and the true GP posterior.

3.1.2 Variational approximations

Let us consider the joint model $p(\mathbf{y}_{1:t}, \mathbf{f}_{1:t}, \mathbf{u})$ and its exact MLL

$$\log p(\mathbf{y}_{1:t}) = \log \int\int p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})p(\mathbf{y}_{1:t})d\mathbf{f}_{1:t}d\mathbf{u}. \tag{3.14}$$

We can obtain a lower bound on $\log p(\mathbf{y}_{1:t})$ as

$$\log p(\mathbf{y}_{1:t}) \geq \log p(\mathbf{y}_{1:t}) - \mathrm{KL}[q(\mathbf{f}_{1:t}, \mathbf{u})||p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})], \tag{3.15}$$

where

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \boldsymbol{\Phi}) \tag{3.16}$$

is a free variational distribution with parameters $\mathbf{m}$ and $\boldsymbol{\Phi}$. The derivation of Equation (3.15) is defined in the Appendix by Equation (D.8). We can see in Equation (3.15) that the bound is tight when the Kullback–Leibler (KL) divergence between the approximated posterior distribution $q(\mathbf{f}_{1:t}, \mathbf{u})$ and the posterior distribution $p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})$ is small.

Minimization of the aforementioned KL divergence is equivalent to maximization of the evidence lower bound (ELBO) defined by

$$F(\mathbf{m}, \mathbf{\Phi}, \mathbf{z}'_{1:m}) = \mathbb{E}_{q(\mathbf{f}_{1:t})}[\log p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})]. \tag{3.17}$$

The derivation without constraining the variational distribution to a Gaussian form can be found in the Appendix by Equation (D.12). Only when the collapsed bound is considered in the Appendix D.4, we can see that the Gaussian form is in fact optimal. Lower bounding the MLL pushes the free variational distribution towards the exact posterior over the pseudo-points, i.e. $q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})$ since

$$\begin{aligned} \mathrm{KL}[q(\mathbf{f}_{1:t}, \mathbf{u})||p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})] &= \mathrm{KL}[\cancel{p(\mathbf{f}_{1:t}|\mathbf{u})}q(\mathbf{u})||\cancel{p(\mathbf{f}_{1:t}|\mathbf{u})}p(\mathbf{u}|\mathbf{y}_{1:t})] \\ &= \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u}|\mathbf{y}_{1:t})]. \end{aligned} \tag{3.18}$$

Figure 3.3 depicts a PGM of a sparse variational approximation to the posterior. Compared to the sparse PGM presented in Figure 3.1, the probabilistic variables $\mathbf{y}_{1:t}$ are omitted. The dependency on $\mathbf{y}_{1:t}$ gets introduced back through $q(\mathbf{u})$ which approximates the posterior over the pseudo-points $p(\mathbf{u}|\mathbf{y}_{1:t})$. If the pseudo-points are placed at the training inputs, i.e. $\mathbf{z}'_{1:m} = \mathbf{Z}_{1:t}$, then the variational approximation is equivalent to the vanilla GP. Two variational approximations emerged that can be separated by the way the free variational distribution $q(\mathbf{u})$ is determined, i.e.

- Numerically optimized;
- Analytically derived.

In the next sections, we will describe these two approaches.

3.1.2.1 Scalable variational Gaussian process

The most scalable sparse method is the scalable variational Gaussian process (SVGP) [38], [39], which retains the variational distribution as a free parameter. The parameters of the free variational distribution are then determined numerically by optimizing the ELBO.

Marginal log-likelihood
The likelihood in Equation (3.17) factorizes, i.e.

$$p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) = \prod_{i=1}^{t} p(y_i|f_i). \tag{3.19}$$

The ELBO can then be rewritten to

$$F(\mathbf{m}, \mathbf{\Phi}, \mathbf{z}'_{1:m}) = \sum_{n=1}^{t} \mathbb{E}_{q(f_n)} \left[\log p(y_n|f_n)\right] - \mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right]. \tag{3.20}$$

The KL divergence is explicitly defined by

$$\mathrm{KL}\left[q(\mathbf{u})||p(\mathbf{u})\right] = \frac{1}{2}\left(\mathrm{tr}\left[\mathbf{K}_{u,u}^{-1}\mathbf{\Phi}\right] + \mathbf{m}^T\mathbf{K}_{u,u}^{-1}\mathbf{m} + \log|\mathbf{K}_{u,u}| - \log|\mathbf{\Phi}| - m\right), \tag{3.21}$$

where m represents the number of pseudo-inputs. The learning of variational parameters can be stabilized by reducing correlations in the variational distribution [97], [98]. An equivalent, but whitened, representation of the bound is defined by

$$F(\mathbf{m}, \mathbf{\Phi}, \mathbf{z}'_{1:m}) = \sum_{n=1}^{t} \mathbb{E}_{\hat{q}(f_n)} \left[\log p(y_n|f_n)\right] - \mathrm{KL}\left[\hat{q}(\mathbf{u})||\hat{p}(\mathbf{u})\right], \tag{3.22}$$

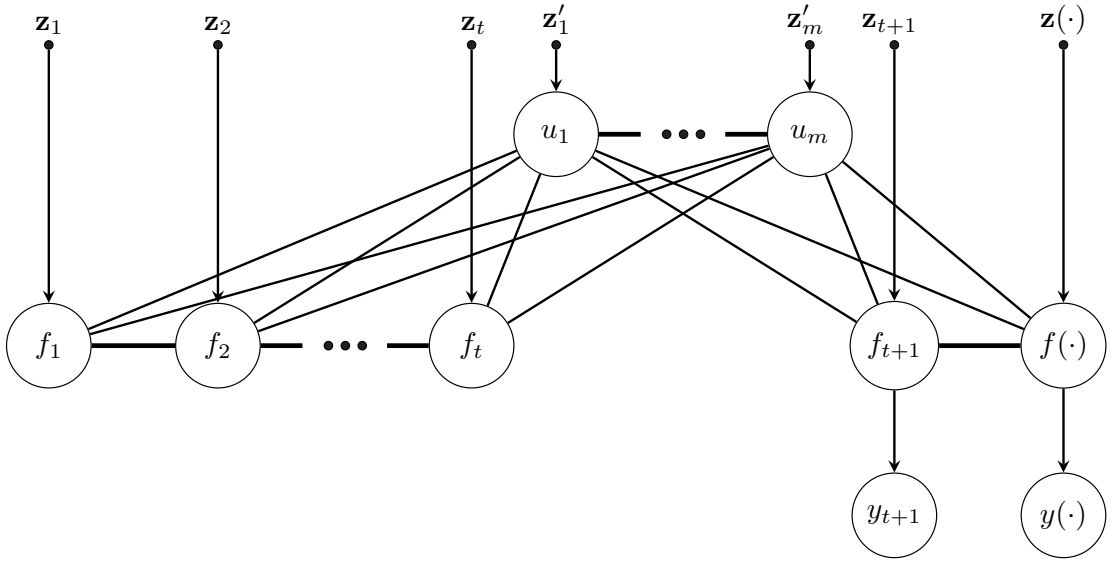


Figure 3.3: PGM of the variational approximations of the sparse model, where the posterior is determined by a rigorous minimization of the KL divergence between $q(\mathbf{f}_{1:t}, \mathbf{u})$ and $p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})$. In the VFE approximation, the distribution is optimally determined in closed-form, whereas in the SVGP approximation, the free parameters are retained to further reduce the computational complexity. Additionally, the SVGP model can be trained in a stochastic manner.

where

$$\begin{aligned} \hat{p}(\mathbf{u}) &= \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ \hat{q}(\mathbf{u}) &= \mathcal{N}(\mathbf{L}_u^{-1}\mathbf{m}, \mathbf{L}_u^{-1}\mathbf{L}_\phi\mathbf{L}_\phi^T\mathbf{L}_u^{-T}) = \mathcal{N}(\hat{\mathbf{m}}, \hat{\mathbf{L}}_\phi\hat{\mathbf{L}}_\phi^T) \end{aligned} \tag{3.23}$$

and $\hat{\mathbf{L}}_\phi = \mathbf{L}_u^{-1}\mathbf{L}_\phi$. The KL divergence is then explicitly defined by

$$\mathrm{KL}\left[\hat{q}(\mathbf{u})||\hat{p}(\mathbf{u})\right] = \frac{1}{2}\left(\mathrm{tr}\left[\hat{\mathbf{L}}_\phi\hat{\mathbf{L}}_\phi^T\right] + \hat{\mathbf{m}}^T\hat{\mathbf{m}} - 2\sum_{i=1}^{t}\log\left[\hat{\mathbf{L}}_\phi\right]_{ii} - m\right). \tag{3.24}$$

Unbiased estimation of the ELBO can be obtained stochastically in batches of training data where the model parameters and pseudo-inputs are jointly learned. The expectations in the MLL defined by Equation (3.22) can be computed by the Gauss-Hermite quadrature [99], since the integrals are only one-dimensional.

Unfortunately, the LBFGS optimization algorithm cannot be used, as the approximation of the Hessian matrix is not robust enough to noise introduced by the stochastic nature of the optimization. However, a successful approach to optimizing the ELBO was found to be an alternating scheme, where a step of the adaptive moment estimation ADAM optimizer [100] is performed on the hyperparameters, followed by a step of the natural gradient descent (NGD) [38], [101] on the variational parameters $\mathbf{m}, \mathbf{\Phi}$.

The computational complexity of the ELBO is $\mathcal{O}(m^3)$, where m represents the number of pseudo-inputs and is a user-defined parameter. The memory requirements are $\mathcal{O}(m^2)$. From here on we will refer to this approximation as the GP-NARX (SVGP) model.

Predictive distribution
The posterior of the latent functions given the observed data is approximated by

$$p\left(\mathbf{f}_{1:t}, f_{t+1}, f\left(\cdot\right), |\mathbf{y}_{1:t}\right) = \int p\left(f\left(\cdot\right)|f_{t+1}, \mathbf{u}\right) p\left(f_{t+1}|\mathbf{u}\right) p\left(\mathbf{f}_{1:t}|\mathbf{u}\right) q\left(\mathbf{u}\right) d\mathbf{u}. \tag{3.25}$$

Marginalizing out the latent values that are not of interest we obtain the predictive distribution

$$\begin{aligned} q\left(f_{t+1}\right) &= \int p\left(\mathbf{f}_{1:t}, f_{t+1}, f\left(\cdot\right)|\mathbf{y}_{1:t}\right) d\mathbf{f}_{\setminus\{f_{t+1}\}} \\ &= \int p\left(f_{t+1}|\mathbf{u}\right) q\left(\mathbf{u}\right) d\mathbf{u}, \end{aligned} \tag{3.26}$$

where $q\left(f_{t+1}\right) \approx p\left(f_{t+1}|\mathbf{y}_{1:t}\right)$. The explicit definition of the approximated predictive distribution $q\left(f_{t+1}\right)$ is specified by

$$\mathbb{E}[f_{t+1}] = \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{m}, \tag{3.27a}$$

$$\mathbb{V}[f_{t+1}] = \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{Q}_{f_{t+1},f_{t+1}} + \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{\Phi}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{t+1}}. \tag{3.27b}$$

Equivalent, but numerically more stable definition, is defined in Appendix E.1.4. The posterior of the pseudo-points $\mathbf{u}$ given the observed data is approximated by $q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})$ and specified by

$$\mathbb{E}[\mathbf{u}] = \mathbf{m}, \tag{3.28a}$$

$$\mathbb{V}[\mathbf{u}] = \mathbf{\Phi}. \tag{3.28b}$$

We emphasize again that $\mathbf{m}$ and $\mathbf{\Phi}$ are learned from data. Numerically stable implementation of the pseudo-point posterior can be found in Appendix E.2.4. The predictive distribution for a single point can be obtained in $\mathcal{O}(m^2)$.

3.1.2.2 Variational free energy

The less scalable variational method is the variational free energy (VFE). However, the benefit of this approach is that the variational distribution is optimally determined by maximizing the ELBO in closed-form.

Marginal log-likelihood
The collapsed ELBO in VFE approximation is defined by

$$F(\mathbf{z}'_{1:m}) = \log\left[\mathcal{N}(\mathbf{y}_{1:t}|\mathbf{0}, \mathbf{Q}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2)\right] - \frac{1}{2\sigma_n^2}\mathrm{tr}(\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}). \tag{3.29}$$

The derivation of the collapsed ELBO can be found in the Appendix D.4. The collapsed bound introduces again the full correlation between the marginals and, therefore, cannot be optimized in a stochastic manner. LBFGS is the usual choice of the optimizer for this bound, which can be computed in $\mathcal{O}(tm^2)$. The memory requirements are $\mathcal{O}(m^2)$. From here on we will refer to this approximation as the GP-NARX (VFE) model.

Predictive distribution
Prediction is defined exactly as in the SVGP model since the models are theoretically the same. The explicit specification of the approximated predictive distribution $q(f_{t+1})$ is again defined by

$$\mathbb{E}[f_{t+1}] = \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{m}, \tag{3.30a}$$
$$\mathbb{V}[f_{t+1}] = \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{Q}_{f_{t+1},f_{t+1}} + \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{\Phi}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{t+1}}, \tag{3.30b}$$

but now the parameters $\mathbf{m}$ and $\mathbf{\Phi}$ are not free.

Although Equation (3.27) and Equation (3.30) are the same, the algorithms to implement the prediction are not. This can be seen in the Appendix in Section E.1.3 and Section E.1.4 where numerically stable implementations of the posteriors are defined. This is because in the case of the SVGP approximation, the variational parameters $\mathbf{m}$ and $\mathbf{\Theta}$ are matrices learned from data, whereas in VFE approximation, they are an algebraic expression. This expression can then be manipulated to result in a more numerically stable implementation.

The posterior of the pseudo-points $\mathbf{u}$ given the observed data is approximated by $q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})$ and specified by

$$\mathbb{E}[\mathbf{u}] = \mathbf{m} = \sigma_n^{-2}\mathbf{K}_{u,u}\mathbf{\Sigma}^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t}, \tag{3.31a}$$
$$\mathbb{V}[\mathbf{u}] = \mathbf{\Phi} = \mathbf{K}_{u,u}\mathbf{\Sigma}^{-1}\mathbf{K}_{u,u}, \tag{3.31b}$$

where

$$\mathbf{\Sigma} = \mathbf{K}_{u,u} + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}. \tag{3.32}$$

Numerically stable implementation of the pseudo-point posterior can be found in Appendix E.2.3. Again, the predictive distribution for a single point can be obtained in $\mathcal{O}(m^2)$.

In the existing literature, the variational approximations are usually preferred over the FITC approximation [35]. The differences between the aforementioned approximations were studied only for static problems. In the next section, we empirically evaluate the performance of the approximations for prediction on 10 dynamical problems. The results are compared to the vanilla GP-NARX model.

3.2 Comparison of Sparse Approximations

Sparse approximations approximate the vanilla GP regression with reduced computational demands. The variational approximations directly approximate the posterior, whereas the FITC approximation, and similar approximations [34], assume additional simplifications of the true sparse model. For that reason, the approximations exhibit different pathologies.

3.2.1 Prediction of a static model

Figure 3.4 and Figure 3.5 show the comparison of different sparse approximations on a static example defined by Equation (2.27). The hyperparameters were learned by MLL maximization. In this thesis, the maximization of the MLL will be equivalently represented by a minimization of the negative MLL for convenience, since minimization is more commonly used in the field of optimization.

Figure 3.4a shows the ground truth, i.e. the posterior fit of the vanilla GP, which is described in Chapter 2. The blue bars represent the predictive distribution of the latent function values at the training data points, i.e. $p(\mathbf{f}|\mathbf{y})$. The confidence interval of 2 standard deviations is presented. The solid blue part of the graph represents the predictive distribution of the test latent function values $\mathbf{f}_*$.

Figure 3.4b, Figure 3.5a, and Figure 3.5b represent the static fits of sparse approximations, where the blue bars show the posterior over the inducing points, i.e. $q(\mathbf{u}|\mathbf{y})$ or $q(\mathbf{u})$. Regression in sparse GPs can be seen as learning a set of pseudo-inputs corresponding to the posterior latent function values that can replace the training latent posteriors, where the number of pseudo-inputs is smaller than the number of training data. The number of pseudo-inputs depends on the complexity of the underlying process and the computational limitations.

We can see that the heteroskedasticity of the fit in the FITC approximation is emphasized, which is due to the conditional independence assumption between the latent function values that belong to the training data. FITC can also be seen as a GP regression where instead of approximating the prior, the likelihood is modified to be heteroscedastic. FITC approximation can be beneficial in scenarios where the likelihood of the observed data is in fact heteroskedastic [37]. However, this is mostly considered a pathological problem of the approximation [35].

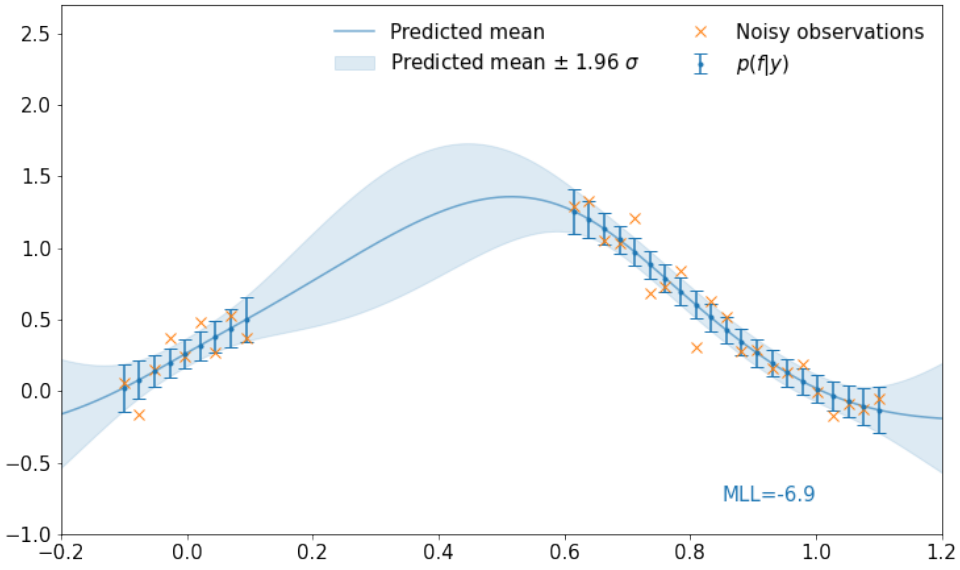
The 2nd Wasserstein distance (W_2), defined by Equation (B.5), between the vanilla GP and a respective approximation is displayed in the bottom left corner. Wasserstein distance defines a distance between probability distributions. Intuitively, it can be seen as the minimum cost of turning one distribution into another. The 2nd Wasserstein distance can be for Gaussian distributions evaluated in closed-form. In this demonstration, fully correlated distributions are compared, i.e. the predictive covariance matrix at test inputs $\mathbf{x}_* \in \mathbb{R}^{p\times 1}$ is $\mathbf{K} \in \mathbb{R}^{p\times p}$.

For the regression of this static function, the W_2 distance clearly shows that the FITC approximation does not compare well to the vanilla GP. Figure 3.5a and Figure 3.5b represent the variational approximation fits, which approximate the vanilla GP much better. This is due to the rigorous minimization of the KL divergence between the true posterior $q(\mathbf{f}_{1:t}, \mathbf{u})$ and the variational posterior $p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})$.

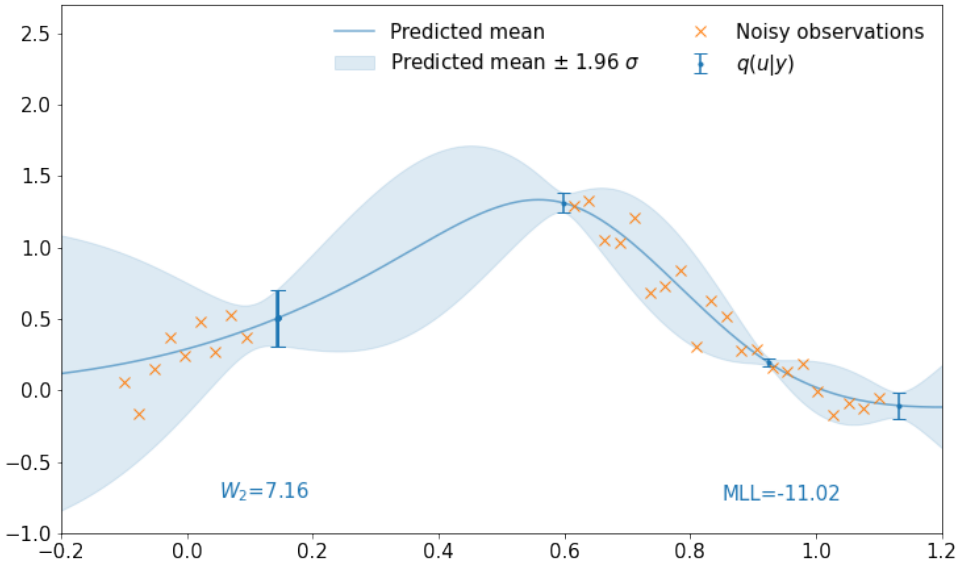
VFE and SVGP approximations are theoretically the same models. The difference is only in how the free variational distribution is determined and how the model is trained. However, different training procedures might result in different hyperparameters. This can be seen in Figure 3.6 which shows the results of MLL optimization in relation to the training steps for different approximations. We can observe that the variational approximations converge to similar MLL values. But they are not the same. We can also see that we need much more optimization steps to train the SVGP model. This is due to the stochastic optimization which is why the MLL in Figure 3.6b is noisy. However, the optimizer in learning the hyperparameters of the SVGP model is of the first order compared to learning hyperparameters in the VFE approximation, where the optimizer is of the second order.

Both variational approximations converge to similar values as vanilla GP. FITC approximation instead converges at a lower value and implies a better fit, which might be deceptive if we use the MLL for model selection. If the underlying process does not exhibit heteroskedastic behavior, the chances are that the FITC approximation will overly emphasize the random heteroskedasticity of the noise in the data. Consequently, we would wrongly select the best model according to the MLL [35]. However, this pathology will tend to be less highlighted in large data sets.

In the next section, we will empirically test the approximations for modeling dynamical systems. We will consider the prediction of different GP-NARX models on chaotic time series.

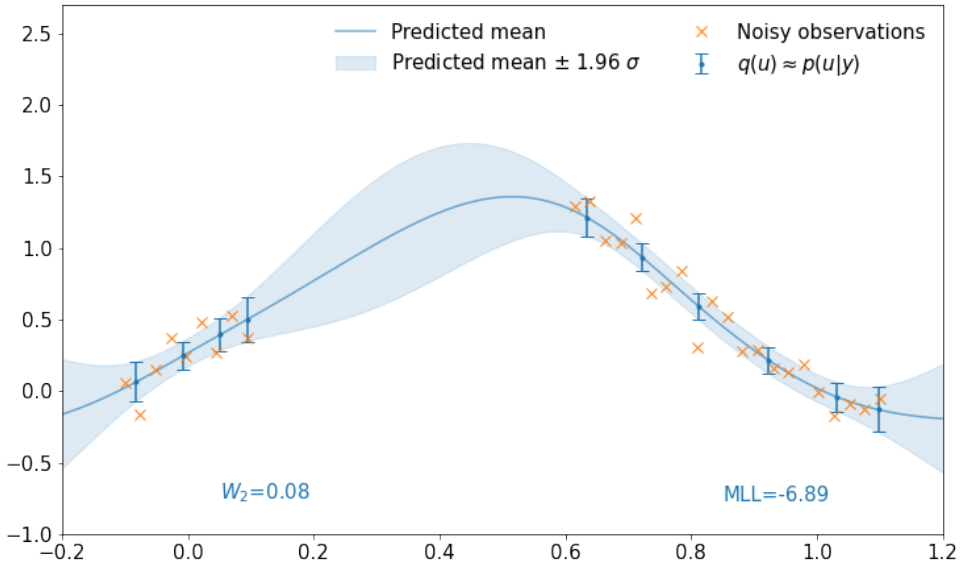


(a) Static fit of a vanilla GP.

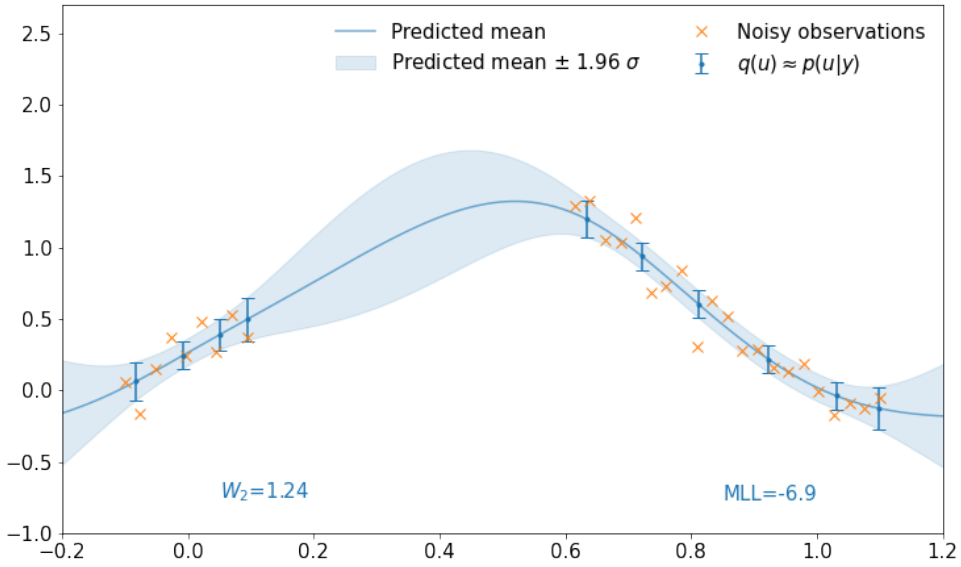


(b) Static fit of a FITC approximation.

Figure 3.4: Static predictions of a function for the vanilla GP and the FITC approximation. The training data can be distilled into a few informative data points. The locations of these data points, i.e. the pseudo-inputs, are determined with optimization. The blue bars represent the respective latent posterior at the training inputs or the pseudo-points for the vanilla GP or the FITC approximation, respectively. MLL depicts the negative marginal log-likelihood after optimization and W_2 depicts the 2nd Wasserstein distance between the fully correlated posterior of the vanilla GP and the FITC approximation.

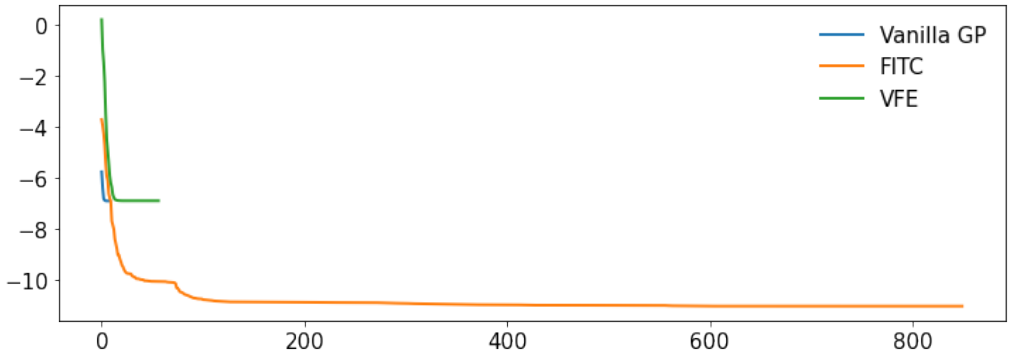


(a) Static fit of a VFE approximation.

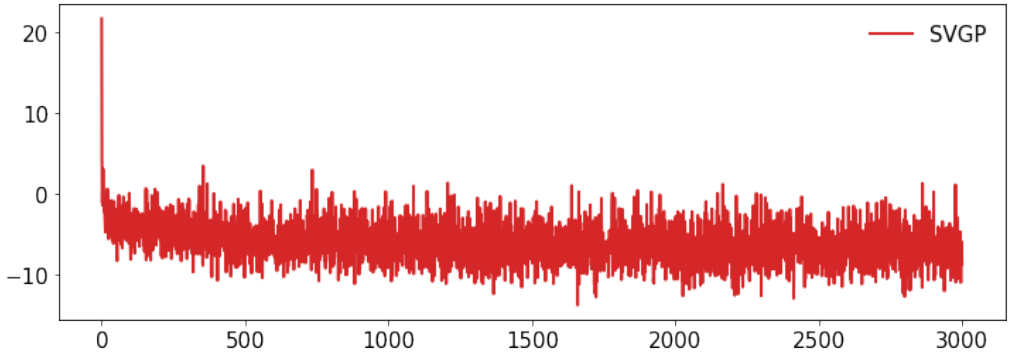


(b) Static fit of a SVGP approximation.

Figure 3.5: Static predictions of a function for the variational GP approximations. The training data can be distilled into a few informative data points. The locations of these data points, i.e. the pseudo-inputs, are determined with optimization. The blue bars represent the respective posterior at the location of the pseudo-points. MLL depicts the negative marginal log-likelihood after optimization and W_2 depicts the 2nd Wasserstein distance between the fully correlated posterior of the vanilla GP and the VFE or SVGP approximation, respectively.



(a) MLL for the vanilla GP, FITC approximation, and VFE approximation.



(b) MLL for the SVGP approximation.

Figure 3.6: Negative MLLs in relation to the training steps for the vanilla GP and its sparse approximations. Vanilla GP, FITC approximation, and VFE approximation were optimized using LBFGS optimizer, whereas SVGP approximation was optimized using ADAM optimizer for the hyperparameters and NDG optimizer for the variational parameters. SVGP was trained for a 100 epochs.

Table 3.1: Comparison of sparse approximations to the vanilla GP-NARX model for prediction in chaotic time-series. The table shows the predictive RMSE and the corresponding 95% confidence interval over random restarts and folds in the cross-validation. The best results of the approximations are emphasized in bold. Compared to the vanilla GP-NARX model, the variational approximations outperform the GP-NARX (FITC) model. [69].

	GP-NARX (FITC)	GP-NARX (VFE)	GP-NARX (SVGP)	vanilla GP-NARX
Hénon map	0.272 ± 0.037	0.205 ± 0.025	**0.195 ± 0.026**	0.198 ± 0.026
Ikeda map	0.383 ± 0.043	0.357 ± 0.036	**0.355 ± 0.036**	0.354 ± 0.037
Logistic map	0.824 ± 0.097	0.748 ± 0.077	**0.744 ± 0.077**	0.746 ± 0.077
Lorenz attractor	0.264 ± 0.040	**0.203 ± 0.039**	0.206 ± 0.039	0.205 ± 0.041
Mackey-Glass	0.392 ± 0.060	**0.363 ± 0.054**	0.365 ± 0.057	0.365 ± 0.056
Quadratic map	0.730 ± 0.075	**0.674 ± 0.066**	0.676 ± 0.065	0.675 ± 0.066
Rössler attractor	0.254 ± 0.056	**0.168 ± 0.086**	0.175 ± 0.128	0.171 ± 0.102
Gauss map	0.267 ± 0.048	**0.225 ± 0.039**	**0.225 ± 0.039**	0.226 ± 0.039
Tent map	0.719 ± 0.099	0.661 ± 0.078	**0.661 ± 0.073**	0.661 ± 0.074
Driven pendulum	0.438 ± 0.056	**0.405 ± 0.048**	**0.405 ± 0.048**	0.407 ± 0.047

3.2.2 Prediction of a dynamical model

In this section, we will compare the prediction of the sparse approximation models for modeling chaotic time series. Chaotic time series belong to a class of nonlinear dynamical systems, i.e. the systems that evolve over time. Chaotic (in this sense) means that the smallest change in the initial conditions produces a very different outcome, even when the governing equations are known exactly. When the governing equations are nonlinear, this is known as deterministic chaos [102].

We empirically validated the sparse approximation methods on 10 chaotic time series. The description of the modeled chaotic time-series can be found in [102]. We standardized the data sets and injected a relatively high Gaussian noise into the time series with a zero mean and a standard deviation of 0.5. 300 pseudo-inputs were used, which were initialized at random.

The whole training data set was used for the batch optimization in GP-NARX (SVGP) model. The order of the NARX model was selected as $n_a = 5$ for all data sets. We used an RBF covariance function with automatic relevance determination (ARD) defined in Appendix C.3. The experiments were run with 10 random restarts. In each random restart, we validated the model with a 5-fold cross-validation. The mean and the variance of the root mean squared error (RMSE), defined by Equation (B.2) were computed over all random restarts and different folds in the cross-validation.

The results of the prediction experiments on the chaotic time series are presented in Table 3.1. The table shows the means and the respective 95% confidence interval of the RMSE over the random restarts and folds in the cross-validation. We can see that, in the case of relatively noisy measurements, the variational methods perform better in regard to the RMSE. The results for the vanilla GP-NARX model are shown for comparison, where we can observe that the variational GP-NARX models perform similarly to the vanilla GP-NARX in terms of the RMSE. However, this is not the case for the GP-NARX (FITC) approximation.

In the next chapter, we will describe an approach that algorithmically unifies the simulation of the vanilla GP-NARX model with its sparse approximations into a single framework. We will analyze the computational complexity of the proposed algorithms and introduce new approximations for the simulation of GP-NARX models.

Chapter 4

Unified Simulation of Autoregressive Gaussian Process Regression

Different approximations with the addition of the vanilla GP-NARX model all define separate training, prediction, and simulation algorithms. This is incredibly tedious for the software development of complex simulation algorithms.

In this section, we will introduce a unified view of the simulation of GP-NARX models which joins the simulation algorithms under a single framework. We will consider existing approximations to the FCMC simulation. A new approximation will be introduced that can trade-off between the computational complexity and the accuracy of the latent simulation by tuning a single user-defined parameter.

4.1 Unified GP-NARX Model Prediction

Similarly, as in the previous sections, we will first start simple to build the intuition, i.e. we will consider prediction. Compared to the previous definition of prediction, we will consider the latent values from $t+1$ to $t+n$ and assume that the inputs are known.

4.1.1 Vanilla GP-NARX model prediction

Let us again consider a standard approach to prediction in the vanilla GP-NARX model which can be defined by

$$\begin{aligned} p(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}) &= \int p(\mathbf{f}_{t+1:t+n}|\mathbf{f}_{1:t},\mathbf{y}_{1:t})p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t})d\mathbf{f}_{1:t} \\ &= \int p(\mathbf{f}_{t+1:t+n}|\mathbf{f}_{1:t})p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t})d\mathbf{f}_{1:t}, \end{aligned} \tag{4.1}$$

where the latent values $f(\cdot)$ are assumed to be already marginalized out. However, we retain the posterior over the latent values that belong to the training data, i.e. $p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t})$, and do not analytically marginalize them out of the joint distribution. For reference, we explicitly define the mean and the variance of the latent values that belong to the training data by

$$\mathbb{E}[\mathbf{f}_{1:t}|\mathbf{y}_{1:t}] = \mathbf{K}_{f_{1:t},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}_{1:t}, \tag{4.2a}$$

$$\mathbb{V}[\mathbf{f}_{1:t}|\mathbf{y}_{1:t}] = \mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f_{1:t},f_{1:t}}. \tag{4.2b}$$

4.1.2 Sparse approximated GP-NARX model prediction

For the sparse approximations, we start from Equation (4.1) and introduce the pseudo-points $\mathbf{u}$ that are in the vanilla GP implicitly marginalized out (they are contained in $f(\cdot)$). The prediction is defined by

$$\begin{aligned} p(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}) &= \int p(\mathbf{f}_{t+1:t+n}|\mathbf{f}_{1:t}, \mathbf{y}_{1:t}) p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t}) d\mathbf{f}_{\setminus\{\mathbf{f}_{t+1:t+n}\}} \\ &= \int\int p(\mathbf{f}_{t+1:t+n}|\mathbf{f}_{1:t}, \mathbf{y}_{1:t}, \mathbf{u}) p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t}, \mathbf{u}) p(\mathbf{u}|\mathbf{y}_{1:t}) d\mathbf{u} d\mathbf{f}_{\setminus\{\mathbf{f}_{t+1:t+n}\}}. \end{aligned} \tag{4.3}$$

If we consider the conditional independence of the training and test data given the pseudo-points, then

$$p(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}) \approx \int\int p(\mathbf{f}_{t+1:t+n}|\mathbf{u}) p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t}, \mathbf{u}) p(\mathbf{u}|\mathbf{y}_{1:t}) d\mathbf{u} d\mathbf{f}_{\setminus\{\mathbf{f}_{t+1:t+n}\}}. \tag{4.4}$$

Marginalizing $p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t}, \mathbf{u})$ out of Equation (4.4), and considering $q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})$ we get

$$\begin{aligned} q(\mathbf{f}_{t+1:t+n}) &\approx \int\int p(\mathbf{f}_{t+1:t+n}|\mathbf{u}) p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t}, \mathbf{u}) q(\mathbf{u}) d\mathbf{u} d\mathbf{f}_{\setminus\{\mathbf{f}_{t+1:t+n}\}} \\ &\approx \int p(\mathbf{f}_{t+1:t+n}|\mathbf{u}) q(\mathbf{u}) d\mathbf{u}. \end{aligned} \tag{4.5}$$

Again, we retain $q(\mathbf{u})$ and do not analytically marginalize over the respective distribution, although a closed-form solution exists. In the next section, we will demonstrate how a unified sample of the posterior can be taken from a vanilla GP-NARX model or its sparse approximations.

4.1.3 Unified sampling in GP-NARX model prediction

From the equations presented in Section (4.1.1) and Section (4.1.2), we can realize that a unifying predictive posterior $p(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t})$, or its approximation $q(\mathbf{f}_{t+1:t+n})$, can be defined by

$$q(\mathbf{f}_{t+1:t+n}) \approx p(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}) = \int p(\mathbf{f}_{t+1:t+n}|\mathbf{u}) q(\mathbf{u}) d\mathbf{u}, \tag{4.6}$$

where

$$\begin{cases} \underbrace{q(\mathbf{u}) = p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t})}_{\text{vanilla GP-NARX}}, \text{ defined by Equation (4.2)}, \\ \underbrace{q(\mathbf{u}) = q(\mathbf{u}|\mathbf{y}_{1:t})}_{\text{GP-NARX (FITC)}}, \text{ defined by Equation (3.13)}, \\ \underbrace{q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})}_{\text{GP-NARX (SVGP)}}, \text{ defined by Equation (3.28)}, \\ \underbrace{q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})}_{\text{GP-NARX (VFE)}} \text{ defined by Equation (3.31)}. \end{cases} \tag{4.7}$$

The main idea behind the unified sampling is that the marginalization over $q(\mathbf{u})$ in Equation (4.6) is not obtained in closed-form, but explicitly retained in the equation. The

distribution over the latent function values is the same as depicted by Equation 4.8, where $\stackrel{d}{=}$ denotes "equal in distribution", i.e.

$$\underbrace{q\left(\mathbf{f}_{t+1:t+n}\right)}_{\text{specific}} \stackrel{d}{=} \int \underbrace{p\left(\mathbf{f}_{t+1:t+n}|\mathbf{u}\right)}_{\text{the same}} \underbrace{q\left(\mathbf{u}\right)}_{\text{specific}} d\mathbf{u}. \tag{4.8}$$

The closed-form solution for $q\left(\mathbf{f}_{t+1:t+n}\right)$ exists but then the algebraic expressions for the prediction become specific for each GP-NARX model and, consequently, the simulation algorithms become specific to each model. But if the posterior over the latent function values $q\left(\mathbf{u}\right)$ is retained, only the respective posterior distribution is specific, whereas the conditional is the same. This is important in simulation, where realizations from the static part, i.e. $q\left(\mathbf{u}\right)$, can easily be obtained. However, the tedious dynamical part is the same for all sparse approximations.

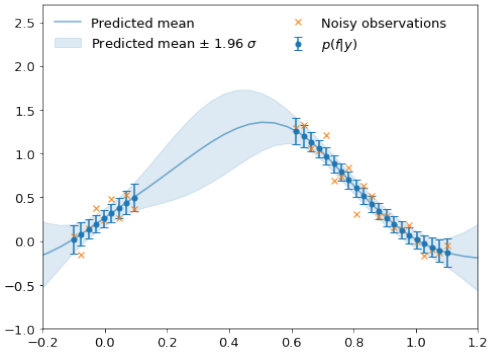
A unified draw from the static predictive distribution can be obtained by firstly taking a batch sample from $q(\mathbf{u})$, followed by a batch sample from the latent function values $\mathbf{f}_{t+1:t+n}$ conditioned on the observed pseudo-points $\tilde{\mathbf{u}}$, i.e.

$$\text{draw } \tilde{\mathbf{u}} \sim q(\mathbf{u}), \tag{4.9a}$$

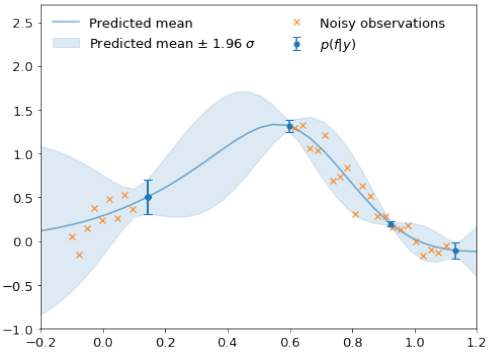
$$\text{draw } \tilde{\mathbf{f}}_{t+1:t+n} \sim p(\mathbf{f}_{t+1:t+n}|\tilde{\mathbf{u}}). \tag{4.9b}$$

Figure 4.1 and Figure 4.2 demonstrate the unified sampling procedure on a static example. The first rows of the aforementioned graphs represent the static prediction of a GP model. The second rows represent the posterior of the latent function after a batch sample is observed from the posterior over the pseudo-points. Lastly, the third rows show a batch sample of the latent function conditioned on the realizations from the pseudo-input posterior. In the case of the vanilla GP model, the pseudo-point posterior is replaced with the respective posterior over the training latent function values.

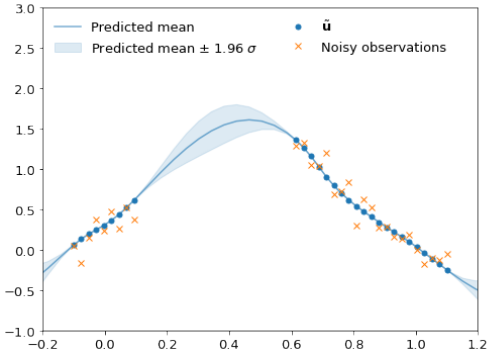
Although this does not simplify the sampling in the prediction of the GP-NARX model, it significantly reduces the algorithmic development in the simulation of the GP-NARX model. In the next section, we will show how a unified simulation sample can be obtained that does not assume the inputs known, but rather estimates them sequentially.



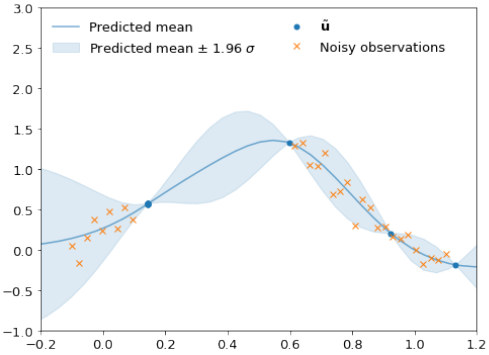
(a) Prediction of the latent function for the vanilla GP model.



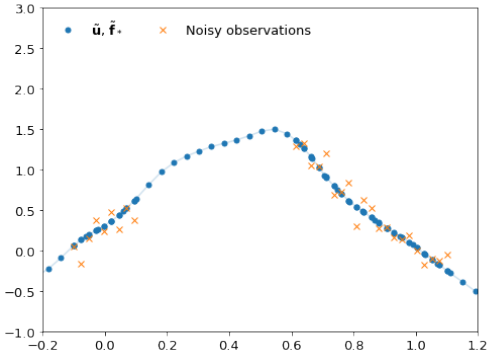
(b) Prediction of the latent function for the FITC approximation.



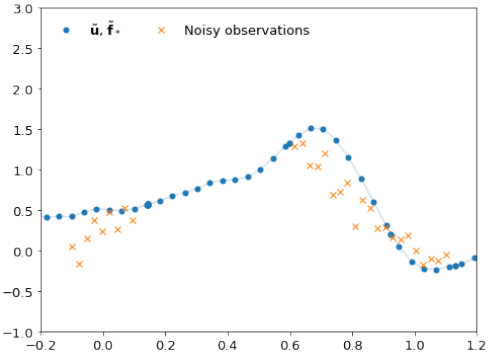
(c) Prediction of the latent function after obtaining $\tilde{\mathbf{u}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP model.



(d) Prediction of the latent function after obtaining $\tilde{\mathbf{u}} \sim q(\mathbf{u}|\mathbf{y})$ for the FITC approximation.

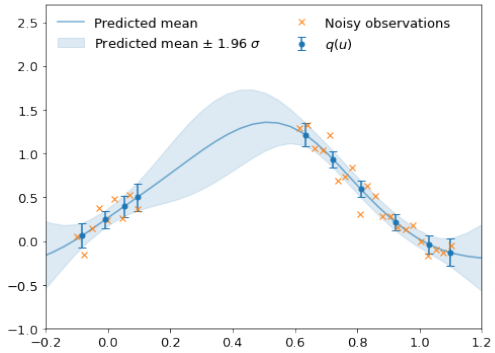


(e) Sample of the latent function, i.e. $\tilde{\mathbf{f}}_* \sim p(\mathbf{f}_*|\tilde{\mathbf{f}})$ for the vanilla GP model.

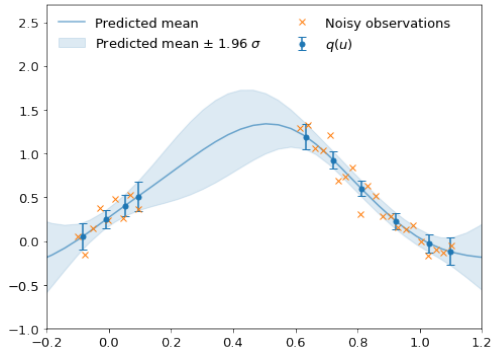


(f) Sample of the latent function, i.e. $\tilde{\mathbf{f}}_* \sim p(\mathbf{f}_*|\tilde{\mathbf{u}})$ for the FITC approximation.

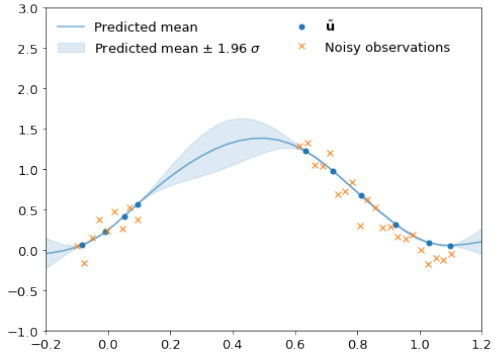
Figure 4.1: Demonstration of a unified sample of the posterior in the vanilla GP model and the FITC approximation. Figure 4.1a and Figure 4.1b represent the respective static prediction. Figure 4.1c and Figure 4.1d show the prediction conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$. Figure 4.1e and Figure 4.1f show the respective realizations of the latent function at the test locations.



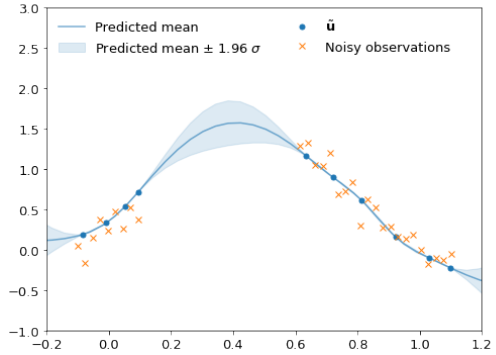
(a) Prediction of the latent function for the VFE approximation.



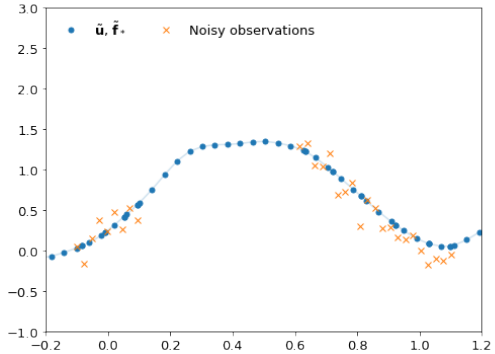
(b) Prediction of the latent function for the SVGP approximation.



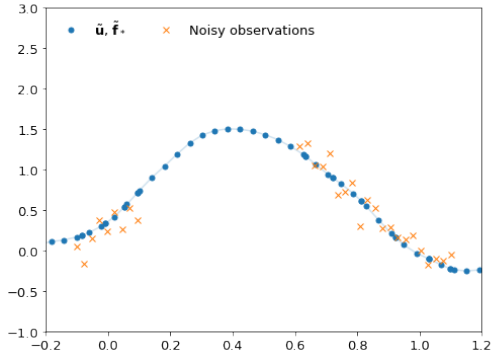
(c) Prediction of the latent function after obtaining $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.



(d) Prediction of the latent function after obtaining $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the SVGP approximation.



(e) Sample of the latent function, i.e. $\tilde{\mathbf{f}}_* \sim p(\mathbf{f}_*|\tilde{\mathbf{u}})$ for the VFE approximation.



(f) Sample of the latent function, i.e. $\tilde{\mathbf{f}}_* \sim p(\mathbf{f}_*|\tilde{\mathbf{u}})$ for the SVGP approximation.

Figure 4.2: Demonstration of a unified sample of the posterior in the VFE and SVGP approximation. Figure 4.2a and Figure 4.2b represent the respective static prediction. Figure 4.2c and Figure 4.2d show the latent prediction conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$. Figure 4.2e and Figure 4.2f show the respective realizations of the latent function at the test locations.

4.2 Unified Algorithm for the Simulation of GP-NARX Models

The joint distribution over the latent function values, generalized for arbitrary parameter value n_a, is defined by

$$q\left(\mathbf{f}_{t+1:t+n}\right) = \int\int \underbrace{p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u})q\left(\mathbf{u}\right)}_{\text{static}} \cdot \\ \cdot \underbrace{\left(\prod_{i=2}^{n} \int p(\mathbf{f}_{t+i}|\mathbf{f}_{t+1:t+i-1}, \mathbf{u}, \mathbf{z}_{t+i}) p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1}) d\mathbf{z}_{t+i}\right)}_{\text{dynamic}} d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}} d\mathbf{u}, \tag{4.10}$$

where

$$p(\mathbf{z}_{t+i}^T|\mathbf{f}_{t+i-n_a:t+i-1}) = \delta\left(\mathbf{z}_{t+i}^T - [\mathbf{f}_{t+i-n_a:t+i-1}^T, \mathbf{x}_{t+i-n_b:t+i-1}^T]\right). \tag{4.11}$$

Similarly, as in the simulation defined by Equation (2.76), a k-th sequential sample of the simulation naturally unrolls through the chained Gaussian distributions, i.e.

$$\text{draw } \tilde{\mathbf{u}}^{(k)} \sim q(\mathbf{u}), \tag{4.12a}$$
$$\text{draw } \tilde{\mathbf{f}}_{t+2-n_a:t+1}^{(k)} \sim p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u}^{(k)}), \tag{4.12b}$$
$$\text{draw } \tilde{\mathbf{z}}_{t+2}^{(k)} \sim p(\mathbf{z}_{t+2}|\tilde{\mathbf{f}}_{t+2-n_a:t+1}^{(k)}), \tag{4.12c}$$
$$\text{draw } \tilde{\mathbf{f}}_{t+2}^{(k)} \sim p(\mathbf{f}_{t+2}|\tilde{f}_{t+1}^{(k)}, \mathbf{u}^{(k)}, \tilde{\mathbf{z}}_{t+2}^{(k)}), \tag{4.12d}$$
$$\vdots \tag{4.12e}$$
$$\text{draw } \tilde{\mathbf{z}}_{t+n}^{(k)} \sim p(\mathbf{z}_{t+n}|\tilde{\mathbf{f}}_{t+n-n_a:t+n-1}^{(k)}), \tag{4.12f}$$
$$\text{draw } \tilde{\mathbf{f}}_{t+n}^{(k)} \sim p(\mathbf{f}_{t+n}|\tilde{\mathbf{f}}_{t+1:t+n-1}^{(k)}, \mathbf{u}^{(k)}, \tilde{\mathbf{z}}_{t+n}^{(k)}). \tag{4.12g}$$

The noisy values can be again drawn independently

$$\text{draw } \tilde{\mathbf{y}}_{t+1:t+n}^{(k)} \sim p\left(\mathbf{y}_{t+1:t+n}|\tilde{\mathbf{f}}_{t+1:t+n}^{(k)}\right). \tag{4.13}$$

Posterior marginals that can be seen as a GMM at an arbitrary time step $t+i$, where $i > 1$, can be defined by

$$q(\mathbf{f}_{t+i}) \approx \frac{1}{r}\sum_{k=1}^{r} p(\mathbf{f}_{t+i}|\tilde{\mathbf{f}}_{t+1:t+i-1}^{(k)}, \mathbf{u}^{(k)}, \tilde{\mathbf{z}}_{t+i}^{(k)}), \tag{4.14}$$

where r denotes the number of MC samples.

Figure 4.3 demonstrates how a MC sample is obtained. At each step, the latent distribution can be represented as a GMM. Single components, i.e. the conditional Gaussian distributions of a GMM, are shown with blue distributions. Orange-colored circles represent realizations from the respective distribution. The unified simulation can be seen as a two-part procedure:

1. Batch sample the latent function values that have known inputs before the simulation begins. Specific to each sparse approximation but static;

2. Iteratively sample the quantities conditioned on the static latent function values. The same for all sparse approximations but dynamic.

Figure 4.4 shows the PGM of a unified simulation. Note that the distribution over $\mathbf{u}$ is specific to the sparse approximation. The unified representation of sparse GP-NARX models introduces the possibility of new approximations to the simulation algorithm. We will consider the approximations in the next section.

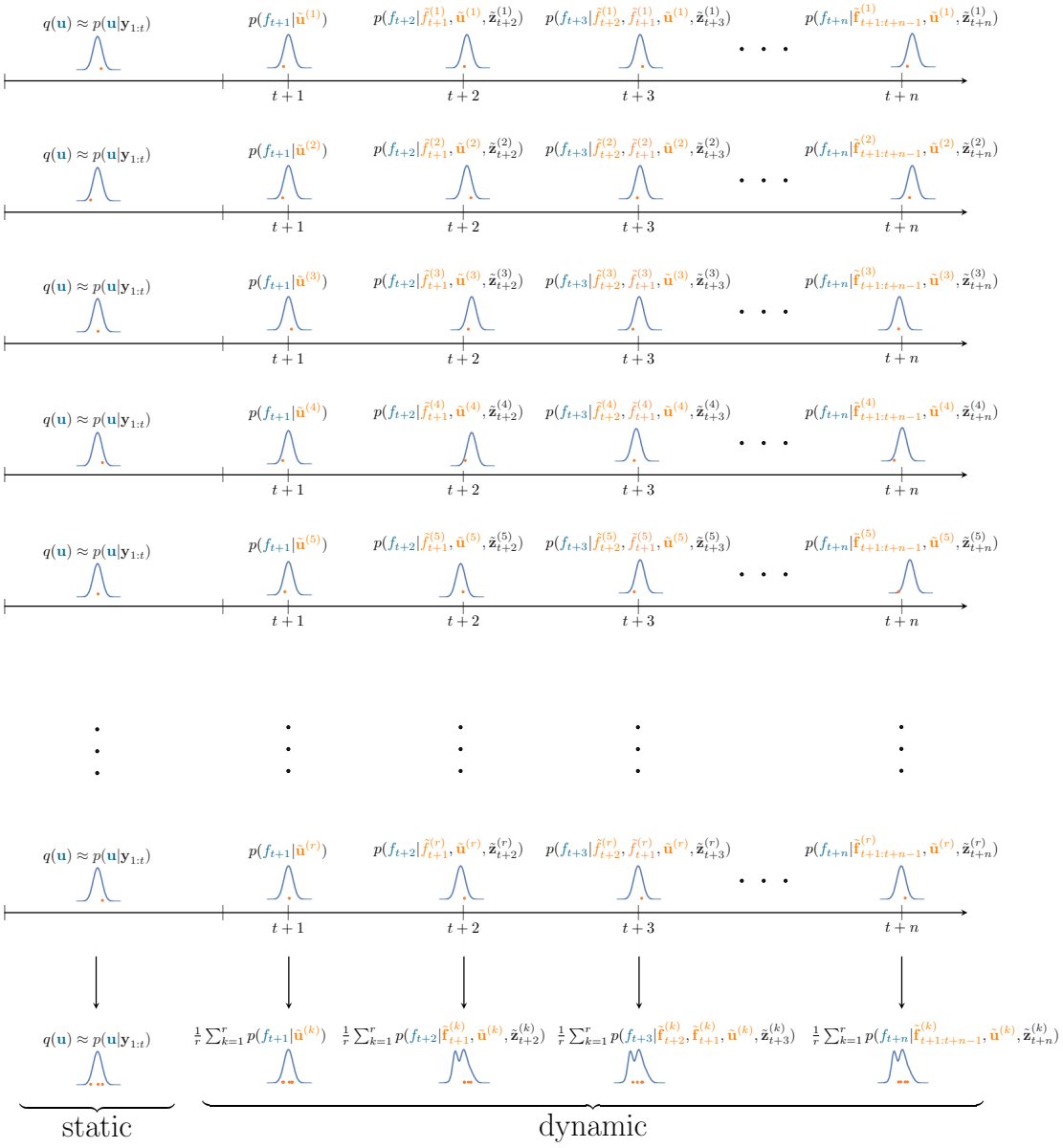


Figure 4.3: Diagram of a unified algorithm for the simulation of GP-NARX models. The figure demonstrates how a MC sample of the simulation is obtained iteratively. The batch sample of the latent function values that have known inputs before the simulation begins is the static part of the algorithm. The iterative sample of the quantities conditioned on the static latent function values is the dynamical part of the algorithm. Orange-colored circles represent realizations at the time step considered. Blue distributions denote the conditionals at the respective time step.

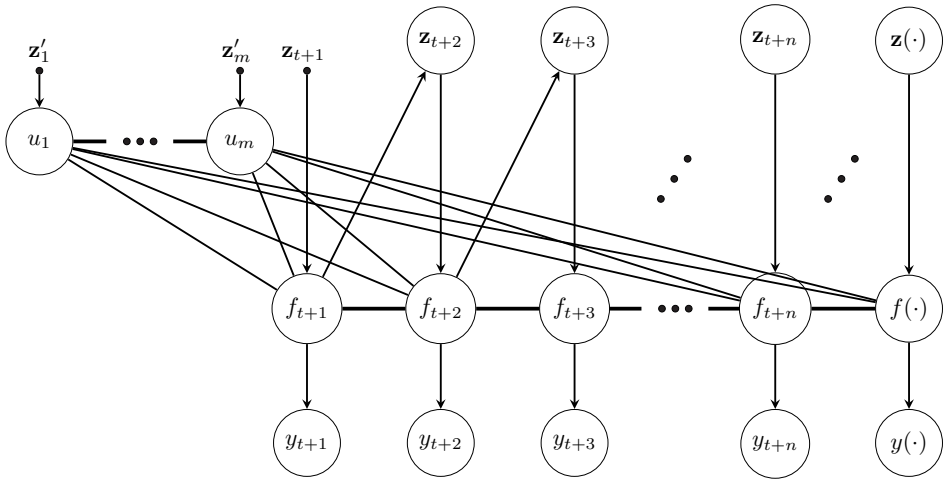


Figure 4.4: PGM of a unified model for the simulation of GP-NARX models. The inputs in simulation are uncertain, i.e. they are random variables, and depend on historical latent function values. For that reason, the simulation cannot be obtained in closed-form and has to be determined numerically.

Approximations to the unified GP-NARX model simulation

In this section, we will summarize the approximations of the ground truth numerical simulation in the context of the unified simulation. From here on we will not consider the naive approximation anymore. Instead, we will introduce another MC approximation that is based on the unified representation of the simulation of GP-NARX models.

We again start with a FCMC simulation, where the latent distribution at an arbitrary time step $t+i$ is defined by

$$\begin{aligned} q\left(\mathbf{f}_{t+1:t+n}\right) = &\int\int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u})q\left(\mathbf{u}\right)\cdot \\ &\cdot\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\mathbf{f}_{t+1:t+i-1},\mathbf{u},\mathbf{z}_{t+i})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}}d\mathbf{u}, \end{aligned} \tag{4.15}$$

which is exact for $q\left(\mathbf{u}\right) = p\left(\mathbf{u}|\mathbf{y}_{1:t}\right)$. With the introduction of pseudo-points $\mathbf{u}$ in the sparse models, we can define a new approximation. This approximation will be from here on referred to as the pseudo independent Monte Carlo (PIMC) simulation. In PIMC simulation, the distribution over the latent function values at the time step $t+i$ is approximated by

$$\begin{aligned} q\left(\mathbf{f}_{t+1:t+n}\right) = &\int\int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u})q\left(\mathbf{u}\right)\cdot \\ &\cdot\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\cancel{\mathbf{f}_{t+1:t+i-1}},\mathbf{u},\mathbf{z}_{t+i})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}}d\mathbf{u} \\ = &\int\int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u})q\left(\mathbf{u}\right)\cdot \\ &\cdot\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\mathbf{u},\mathbf{z}_{t+i})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}}d\mathbf{u}, \end{aligned} \tag{4.16}$$

where the latent values are considered conditionally independent given the realizations of the pseudo-points $\tilde{\mathbf{u}}$.

CIMC simulation does not retain $q(\mathbf{u})$, but instead analytically marginalizes over the aforementioned distribution, pushing the integral in the product term, i.e.

$$
\begin{aligned}
q\left(\mathbf{f}_{t+1:t+n}\right) &= \int\int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u})q\left(\mathbf{u}\right)\cdot \\
&\cdot\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\cancel{\mathbf{f}_{t+1:t+i-1}},\mathbf{u},\mathbf{z}_{t+i})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}}d\mathbf{u} \\
&= \int\int p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{u})q\left(\mathbf{u}\right)\cdot \\
&\cdot\left(\prod_{i=2}^{n}\int p(\mathbf{f}_{t+i}|\mathbf{u},\mathbf{z}_{t+i})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}}d\mathbf{u} \\
&= \int q(\mathbf{f}_{t+2-n_a:t+1})\cdot \\
&\cdot\left(\prod_{i=2}^{n}\int\int p(\mathbf{f}_{t+i}|\mathbf{u},\mathbf{z}_{t+i})q(\mathbf{u})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}d\mathbf{u}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}} \\
&= \int q(\mathbf{f}_{t+2-n_a:t+1})\left(\prod_{i=2}^{n}\int q(\mathbf{f}_{t+i}|\mathbf{z}_{t+i})p(\mathbf{z}_{t+i}|\mathbf{f}_{t+i-n_a:t+i-1})d\mathbf{z}_{t+i}\right)d\mathbf{f}_{\setminus\{f_{t+1:t+n}\}},
\end{aligned}
\tag{4.17}
$$

where $q(\mathbf{u})$ is again specific to each approximation and defined by Equation (4.7).

The main difference between the PIMC and CIMC is in the marginalization over $q(\mathbf{u})$. This can be best seen on a static example in Figure 4.1 and Figure 4.2. The prediction in the first rows of the aforementioned figures corresponds to that of the CIMC simulation, i.e. the test points represented with blue are conditionally independent given the observed data, and $q(\mathbf{u})$ is marginalized out in closed-form after pushing the integral inside the product. The prediction in the second rows of the aforementioned figures corresponds to that of the PIMC simulation, i.e. the test points represented with blue are conditionally independent given the realizations from $q(\mathbf{u})$, and the integral is more correctly considered out of the product term. A realization from $q(\mathbf{u})$ reduces the residual variance in the respective MC sample. Therefore, the consecutive latent values in the PIMC simulation are more correlated than in the CIMC case.

The PIMC simulation has identical computational complexity as the CIMC simulation if we assume that the running time of a batch sample over the pseudo-points is negligible compared to the running time of the dynamical part of the algorithm. In the CIMC and PIMC simulation, the computation can be cached efficiently, i.e. only one Cholesky factor is needed and can be precomputed. However, we can expect that the PIMC simulation will better approximate the FCMC simulation than the CIMC simulation.

But generally, some uncertainty will still be unaccounted for. In the next section, we will present an approximated algorithm for the simulation of GP-NARX models based on the unified FCMC simulation which can in the limit recover the ground truth.

4.3 Thresholded Algorithm for the Simulation of GP-NARX Models

In this section, we will present a simulation algorithm that introduces a trade-off parameter between the computational complexity and the accuracy of the estimated latent distribution in the simulation. The algorithm allows the error minimization of the estimated response while maximizing the computational resources.

Noisy versus latent observations

The true numerical simulation algorithm keeps all past latent states in memory. This is the consequence of the definition of the GP. We want to emphasize that the consecutive realizations are latent. The latent realizations are much more informative than the noisy realizations since they represent an uncorrupted observation of the process.

The difference between the informative nature of the noisy and the latent realizations is presented by the polynomial regression in Figure 4.5. Figure 4.5a shows a linear fit with noisy observations and Figure 4.5b shows a linear fit with latent observations. We can see that for the linear function, only 2 true latent samples are enough to exactly specify the function at an arbitrary input. However, we need much more noisy observations to model the latent function, and some error is still present.

Similarly Figure 4.5c, Figure 4.5d, Figure 4.5e, and Figure 4.5f present the Bayesian regression fits for higher-order polynomials. The observed latent samples that fully specify the latent function can be seen as degrees of freedom (DOF). DOF refers to the maximum number of logically independent values, i.e. the values that have the freedom to vary. In linear regression, these are the free parameters of the model, e.g. weights in polynomial regression. In nonlinear regression, the analysis gets more complicated. A possible definition of the flexibility of the model is the Vapnik–Chervonenkis (VC) dimension [103], which is defined as the cardinality of the largest set of points that the classification algorithm can shatter.

GP regression is equivalent to Bayesian regression with an infinite number of weights, which implies infinite VC dimension (model complexity grows with the number of training points). However, the (useful) covariance functions learned with the maximization of the MLL induce nonlinear functions that can generalize well and are, consequently, relatively smooth. Smoothness can be seen as a form of regularization where the effective DOF are reduced, i.e. the free parameters become correlated [5]. Determining the right amount of smoothness/regularization/model complexity is a crucial concept in data-driven modeling and is directly addressing the bias-variance trade-off problem.

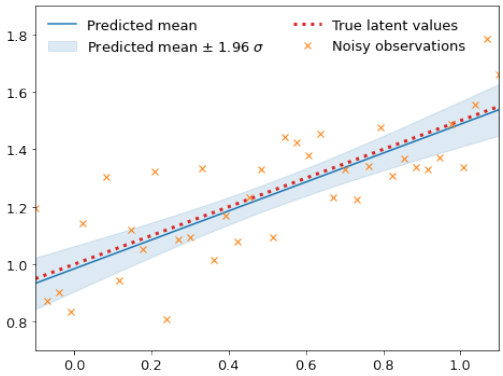
The flexibility of the induced functions by a GP can be seen empirically by how much the latent uncertainty is reduced after observing latent realizations. If the uncertainty is heavily reduced, the possible set of latent functions that can be realized from the posterior gets smaller, and the inducing functions are smoother. If the uncertainty is not heavily reduced, the latent points are not well correlated, i.e. are rough.

For example, the RBF covariance function induces nonlinear functions that are infinitely differentiable. This implies very smooth functions and, consequently, the induced functions are strongly regularized and the latent values highly correlated. For that reason, the RBF covariance function can be a great choice for processes that are governed by relatively smooth mappings. Figure 4.6 depicts the residual variance in the latent function $p(\mathbf{f}_*|\tilde{\mathbf{u}})$ after a sample from the pseudo-point posterior is obtained, i.e. draw $\tilde{\mathbf{u}} \sim q(\mathbf{u})$. We can see that the residual variance of the latent function is small, which implies that given $\tilde{\mathbf{u}}$, not many realizations of the latent functions persist.

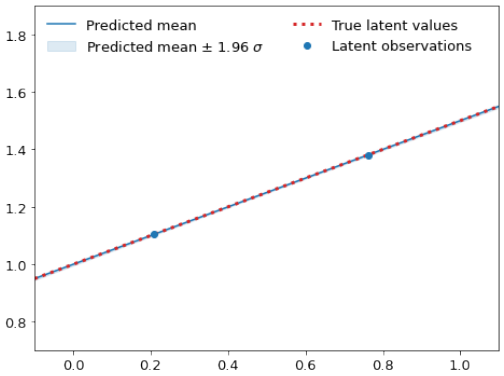
Matérn covariance functions represent a family of functions where a single parameter defines the degree of the differentiability [22]. For example Matérn(ν) covariance functions, defined by Equation (C.5), are $\lceil \nu \rceil$ - 1 differentiable. The induced functions are, therefore, rougher than in the case of the RBF covariance function (although the RBF kernel is a special case of the Matérn kernel where $\nu = \infty$).

Figure 4.7 represents a fit of a static function conditioned on a realization from the pseudo-point posterior for the Matérn($\frac{5}{2}$) covariance function, and Figure 4.8 for the Matérn($\frac{1}{2}$) covariance function. For the Matérn($\frac{5}{2}$) kernel, a reasonable amount of uncertainty is reduced in the posterior. However, for the Matérn($\frac{1}{2}$) covariance function, latent realizations are not very informative since by definition, the neighboring points are not well correlated and the induced functions are very rough. Consequently, the latent uncertainty is not heavily reduced.

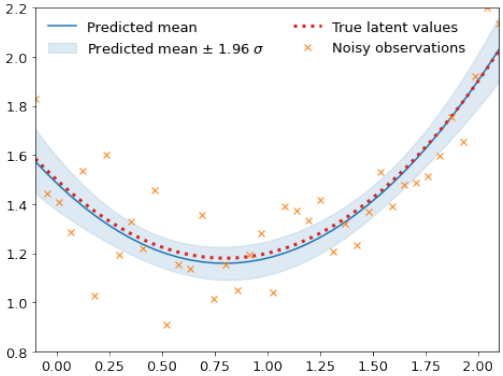
In the next section, we will introduce an approximation to the numerical simulation that is based on the concept above, i.e. useful covariance functions seldom induce very rough functions. Although all the latent function values are retained in the FCMC simulation, the remaining variance in the latent posterior gets heavily reduced after a certain amount of latent realizations. An algorithm can be derived where the uninformative latent realizations are discarded online.



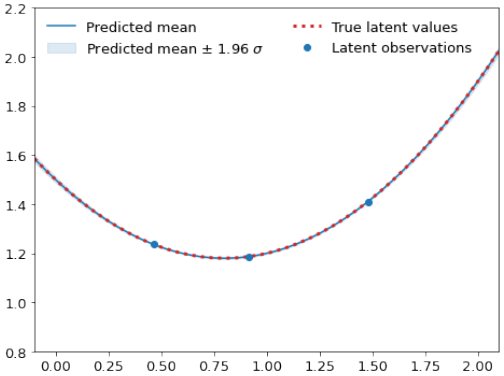
(a) Latent function fit from noisy observations, where $\mathbf{f} = 0.5\mathbf{x} + 1$.



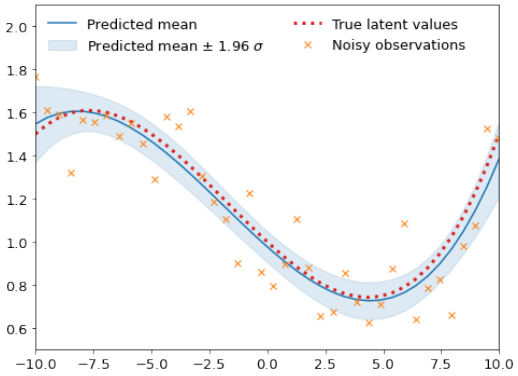
(b) Latent function fit from latent observations, where $\mathbf{f} = 0.5\mathbf{x} + 1$.



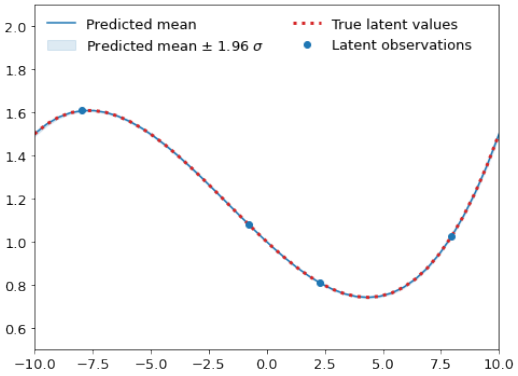
(c) Latent function fit from noisy observations, where $\mathbf{f} = 0.5(\mathbf{x} - 1)^2 + 0.2\mathbf{x} + 1$.



(d) Latent function fit from latent observations, where $\mathbf{f} = 0.5(\mathbf{x} - 1)^2 + 0.2\mathbf{x} + 1$.

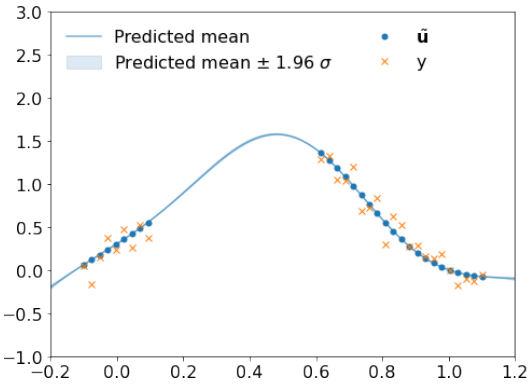


(e) Latent function fit from noisy observations, where $\mathbf{f} = 0.1\mathbf{x}^3 + 0.005\mathbf{x}^2 - 0.1\mathbf{x} + 1$.

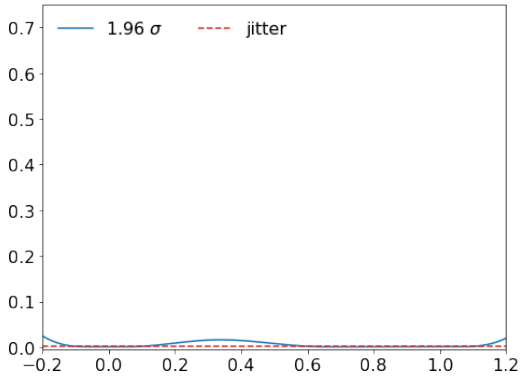


(f) Latent function fit from latent observations, where $\mathbf{f} = 0.1\mathbf{x}^3 + 0.005\mathbf{x}^2 - 0.1\mathbf{x} + 1$.

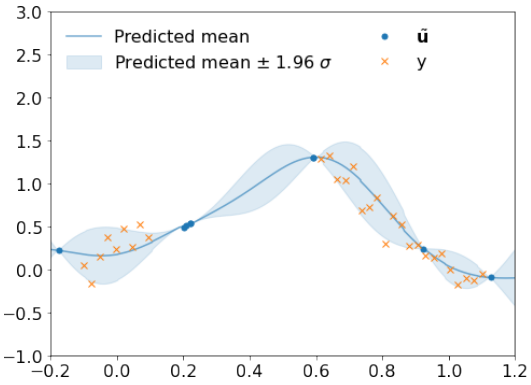
Figure 4.5: Noisy versus latent observations for polynomial regression. The latent values $\mathbf{f}$ were corrupted with Gaussian noise, i.e. $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, 0.15^2\mathbf{I})$.



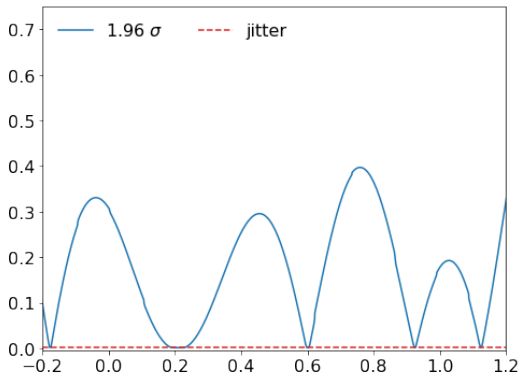
(a) Prediction of the latent function conditioned on $\tilde{\mathbf{f}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP.



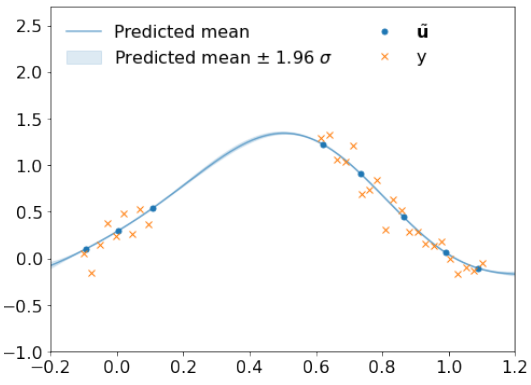
(b) Residual variance of the latent function conditioned on $\tilde{\mathbf{f}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP.



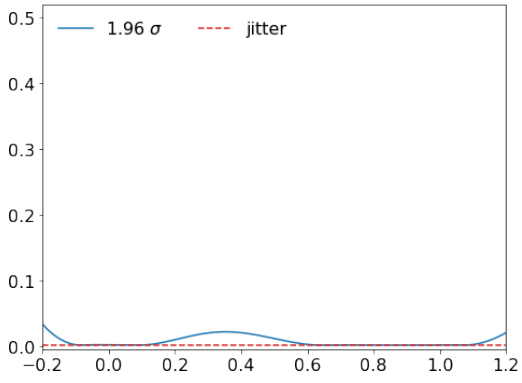
(c) Prediction of the latent function conditioned on $\tilde{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{y})$ for the FITC approximation.



(d) Residual variance of the latent function conditioned on $\tilde{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{y})$ for the FITC approximation.

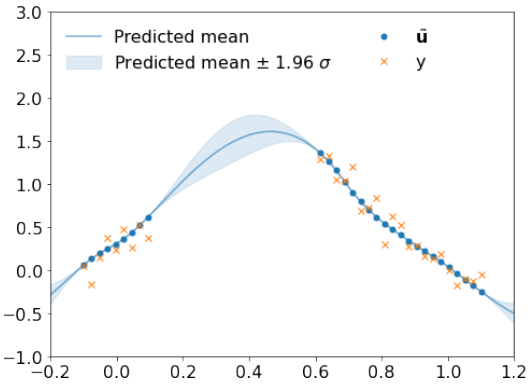


(e) Prediction of the latent function conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.

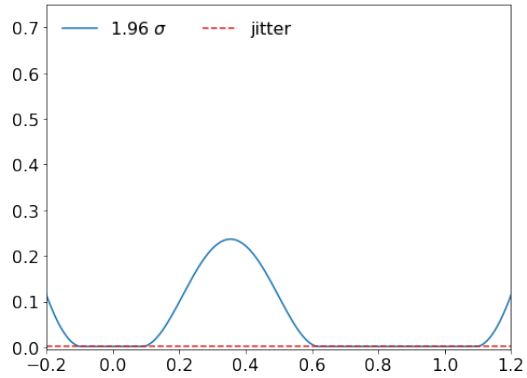


(f) Residual variance of the latent function conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.

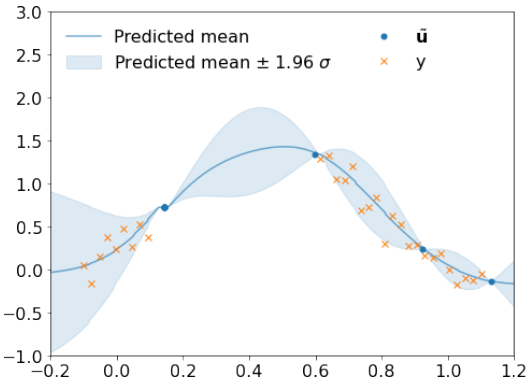
Figure 4.6: Demonstration of the residual variance after a sample is observed from the pseudo-point posterior. The covariance function used was RBF which is defined in Appendix C. Jitter represents the artificial noise added to the matrix diagonals for numerical stability.



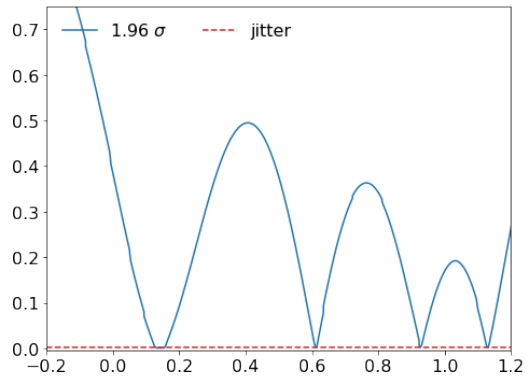
(a) Prediction of the latent function after drawing $\tilde{\mathbf{f}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP.



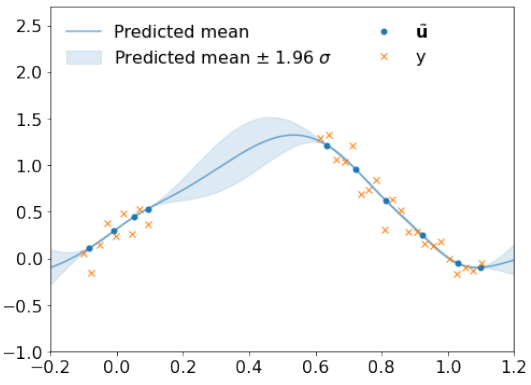
(b) Residual variance of the latent function conditioned on $\tilde{\mathbf{f}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP.



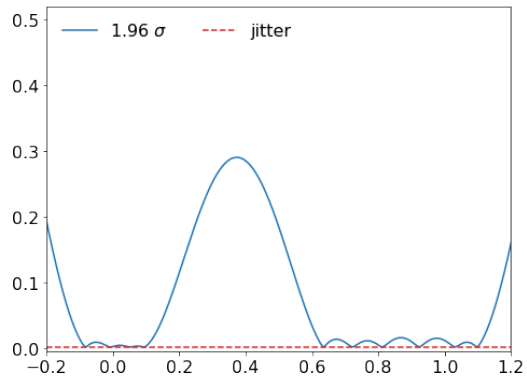
(c) Prediction of the latent function conditioned on $\tilde{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{y})$ for the FITC approximation.



(d) Residual variance of the latent function conditioned on $\tilde{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{y})$ for the FITC approximation.

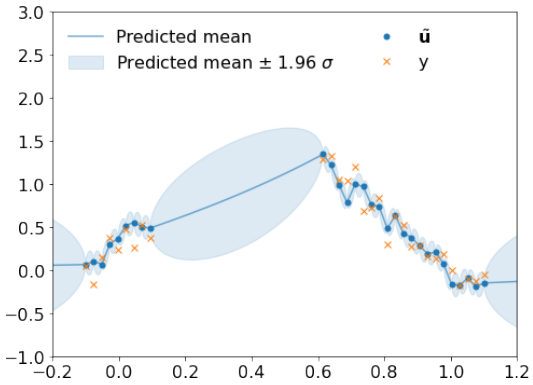


(e) Prediction of the latent function conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.

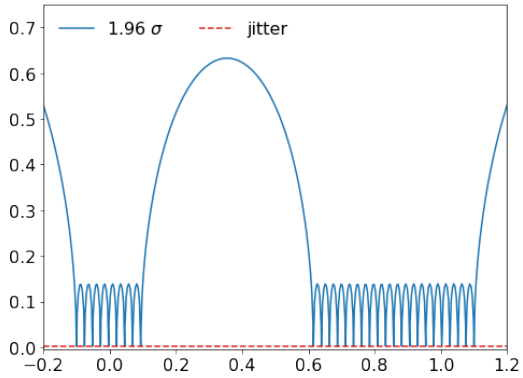


(f) Residual variance of the latent function conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.

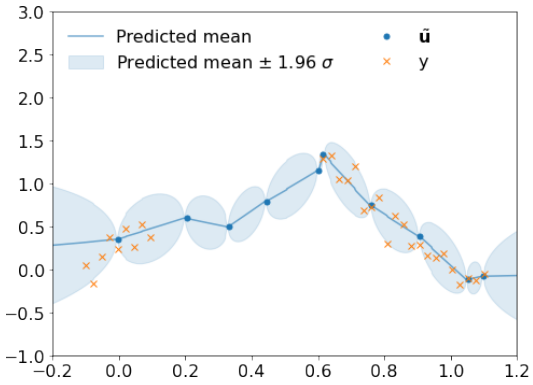
Figure 4.7: Demonstration of the residual variance after a sample is observed from the pseudo-point posterior. The covariance function used was Matérn($\frac{5}{2}$) which is defined in Appendix C. Jitter represents the artificial noise added to the matrix diagonals for numerical stability.



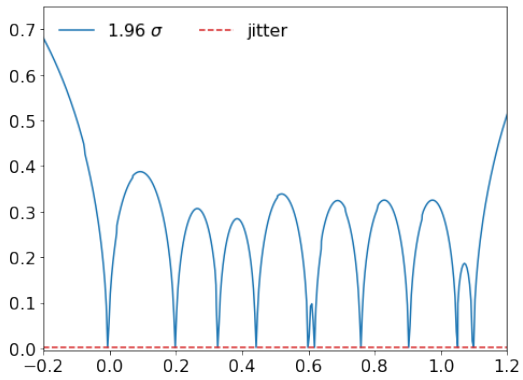
(a) Prediction of the latent function conditioned on $\tilde{\mathbf{f}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP.



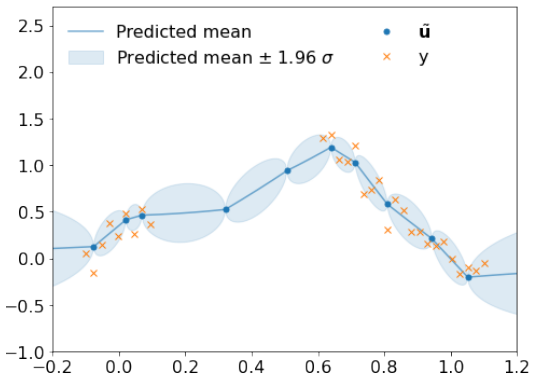
(b) Residual variance of the latent function conditioned on $\tilde{\mathbf{f}} \sim p(\mathbf{f}|\mathbf{y})$ for the vanilla GP.



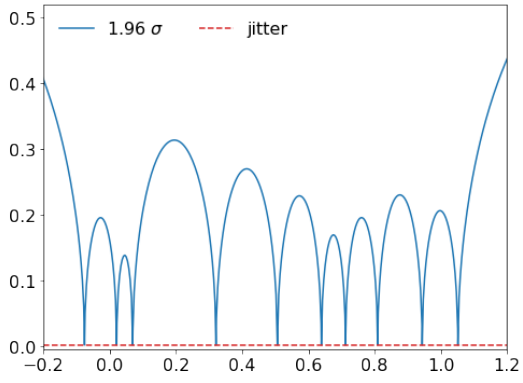
(c) Prediction of the latent function conditioned on $\tilde{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{y})$ for the FITC approximation.



(d) Residual variance of the latent function conditioned on $\tilde{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{y})$ for the FITC approximation.



(e) Prediction of the latent function conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.



(f) Residual variance of the latent function conditioned on $\tilde{\mathbf{u}} \sim q(\mathbf{u})$ for the VFE approximation.

Figure 4.8: Demonstration of the residual variance after a sample is observed from the pseudo-point posterior. The covariance function used was Matérn($\frac{1}{2}$) which is defined in Appendix C. Jitter represents the artificial noise added to the matrix diagonals for numerical stability.

Thresholded simulation algorithm

A way to determine if the latent realization is informative is with the entropy. The entropy specifies the level of uncertainty to the random variable's possible outcomes. For the time step $t+i$, the Shannon entropy [104], [105] of the latent value given the realizations up to the time step i, for a single MC sample, is explicitly defined by

$$H = \frac{1}{2}\log\ \left[2\pi \mathrm{e}\mathbb{V}\left[p(\mathbf{f}_{t+i}|\tilde{\mathbf{f}}^{(k)}_{t+1:t+i-1}, \tilde{\mathbf{u}}^{k}, \tilde{\mathbf{z}}^{(k)}_{t+i})\right]\right], \tag{4.18}$$

where log denotes the natural logarithm. The variance of the predictive distribution given the realization up to the time step i completely specifies the level of uncertainty. Therefore, we can use the variance as a measure of information. The idea of thresholded simulation is to define a **threshold parameter** $\boldsymbol{T}_{\mathbb{V}}$. If the predictive variance at the time step considered exceeds the defined threshold, the latent realizations are considered informative and are kept in memory. Otherwise, the latent realization is discarded.

An independent draw from the thresholded simulation is defined by the following algorithm. Firstly, a sample from the posterior over the pseudo-points is taken and added to the set of observed latent values $\mathcal{F}$, i.e.

$$\begin{aligned} &\text{draw } \tilde{\mathbf{u}} \sim q(\mathbf{u}), \\ &\quad \mathcal{F} = \{\tilde{\mathbf{u}}\}. \end{aligned} \tag{4.19}$$

Secondly, a posterior latent realization is taken up to the time step $t+1$, i.e.

$$\begin{aligned} &\text{draw } \tilde{\mathbf{f}}_{t+2-n_a:t+1} \sim p(\mathbf{f}_{t+2-n_a:t+1}|\mathcal{F}), \\ &\quad \mathcal{F} = \begin{cases} \mathcal{F}, \text{ if } \mathbb{V}\left[p(\mathbf{f}_{t+2-n_a:t+1}|\mathcal{F})\right] \leq T_{\mathbb{V}} \\ \{\tilde{\mathbf{f}}_{t+1}, \mathcal{F}\}, \text{ if } \mathbb{V}\left[p(\mathbf{f}_{t+2-n_a:t+1}|\mathcal{F})\right] > T_{\mathbb{V}} \end{cases}, \end{aligned} \tag{4.20}$$

where the latent observations are added to the set $\mathcal{F}$ only if they are considered informative, i.e. the variance exceeds the user-defined threshold. The procedure is repeated for the time step $t+2$, i.e.

$$\begin{aligned} &\text{draw } \tilde{\mathbf{z}}_{t+2} \sim p(\mathbf{z}_{t+2}|\tilde{\mathbf{f}}_{t+2-n_a:t+1}), \\ &\text{draw } \tilde{\mathbf{f}}_{t+2} \sim p(\mathbf{f}_{t+2}|\mathcal{F}, \tilde{\mathbf{z}}_{t+2}), \\ &\quad \mathcal{F} = \begin{cases} \mathcal{F}, \text{ if } \mathbb{V}\left[p(\mathbf{f}_{t+2}|\mathcal{F}, \tilde{\mathbf{z}}_{t+2})\right] \leq T_{\mathbb{V}} \\ \{\tilde{\mathbf{f}}_{t+2}, \mathcal{F}\}, \text{ if } \mathbb{V}\left[p(\mathbf{f}_{t+2}|\mathcal{F}, \tilde{\mathbf{z}}_{t+2})\right] > T_{\mathbb{V}} \end{cases}. \end{aligned} \tag{4.21}$$

For the time step $t+3$

$$\begin{aligned} &\text{draw } \tilde{\mathbf{z}}_{t+3} \sim p(\mathbf{z}_{t+3}|\tilde{\mathbf{f}}_{t+3-n_a:t+2}), \\ &\text{draw } \tilde{\mathbf{f}}_{t+3} \sim p(\mathbf{f}_{t+3}|\mathcal{F}, \tilde{\mathbf{z}}_{t+3}), \\ &\quad \mathcal{F} = \begin{cases} \mathcal{F}, \text{ if } \mathbb{V}\left[p(\mathbf{f}_{t+3}|\mathcal{F}, \tilde{\mathbf{z}}_{t+3})\right] \leq T_{\mathbb{V}} \\ \{\tilde{\mathbf{f}}_{t+3}, \mathcal{F}\}, \text{ if } \mathbb{V}\left[p(\mathbf{f}_{t+3}|\mathcal{F}, \tilde{\mathbf{z}}_{t+3})\right] > T_{\mathbb{V}} \end{cases}. \end{aligned} \tag{4.22}$$

The latent sample at the time step $t+n$ is obtained by

$$\begin{aligned} &\text{draw } \tilde{\mathbf{z}}_{t+n} \sim p(\mathbf{z}_{t+n}|\tilde{\mathbf{f}}_{t+i-n_a:t+n-1}), \\ &\text{draw } \tilde{\mathbf{f}}_{t+n} \sim p(\mathbf{f}_{t+n}|\mathcal{F}, \tilde{\mathbf{z}}_{t+n}). \end{aligned} \tag{4.23}$$

Table 4.1: Computational complexity and memory requirements of the numerical approximations for the unified simulation of the GP-NARX models. The number of samples is denoted by r, the predictive horizon with n, the number of retained samples in TCMC simulation with p, and the number of pseudo-points by m. In the vanilla GP-NARX case, m is replaced with the number of training data by t.

Simulation Algorithm	Computational Complexity	Memory Requirements
FCMC simulation	$\mathcal{O}\left(r \cdot n \cdot (n+m)^2\right)$	$\mathcal{O}\left(r \cdot (n+m)^2\right)$
TCMC simulation	$\mathcal{O}\left(r \cdot n \cdot (p+m)^2\right)$	$\mathcal{O}\left(r \cdot (p+m)^2\right)$
PIMC simulation	$\mathcal{O}\left(r \cdot n \cdot m^2\right)$	$\mathcal{O}\left(r \cdot m^2\right)$
CIMC simulation	$\mathcal{O}\left(r \cdot n \cdot m^2\right)$	$\mathcal{O}\left(r \cdot m^2\right)$
FCN simulation	$\mathcal{O}\left(n \cdot (n+m)^2\right)$	$\mathcal{O}\left((n+m)^2\right)$
CIN simulation	$\mathcal{O}\left(n \cdot m^2\right)$	$\mathcal{O}\left(m^2\right)$

A MC approximation of the posterior latent distribution that can be seen as a GMM at an arbitrary time step $t+i$ is defined by

$$p(\mathbf{f}_{t+i}|\mathbf{y}_{1:t}) \approx \frac{1}{r}\sum_{k=1}^{r} p(\mathbf{f}_{t+i}|\mathcal{F}^{(k)}, \tilde{\mathbf{z}}_{t+i}^{(k)}), \tag{4.24}$$

where k denotes a single sequential draw, r denotes the number of samples, and $\mathcal{F}^{(k)}$ represents the retained latent samples. From here on we will refer to this approximation as the thresholded correlated MC simulation (TCMC). A flow chart of TCMC simulation is shown in Figure 4.9. If the algorithm above only considered the latent values, i.e. $\mathcal{F}$, the simulation chart more precisely considers input/output pairs represented by $\mathcal{D} = \{\mathbf{Z}, \boldsymbol{\eta}\}$, where $\boldsymbol{\eta}$ represents the vector of retained latent realizations and $\mathbf{Z}$ represents the matrix of the corresponding inputs.

A PIMC simulation can be seen as a special case of the TCMC, where the threshold $T_{\mathbb{V}}$ is sufficiently large so that no latent observations are retained after drawing $\tilde{\mathbf{u}} \sim q(\mathbf{u})$. Also, the FCMC simulation can be seen as a special case of the TCMC simulation, where the threshold $T_{\mathbb{V}} = 0$ and all latent observations are retained.

This kind of representation of the simulation allows us to trade-off between the computational complexity and the degree to which the simulation is approximated. Table 4.1 shows the computational complexities and memory requirements of the unified simulation of the GP-NARX models. The complexity and memory requirements of the TCMC simulation depend on parameter p which denotes the number of retained latent observations. The number of retained latent observations in practice depends on several factors:

- Threshold parameter $T_{\mathbb{V}}$;
- Choice of the covariance function;
- Specifics of the problem, e.g. learned hyperparameters.

In the next section, we will empirically test the simulation approximations. We will compare the approximations to the FCMC simulation and analyze the running times. Several covariance functions will be considered.

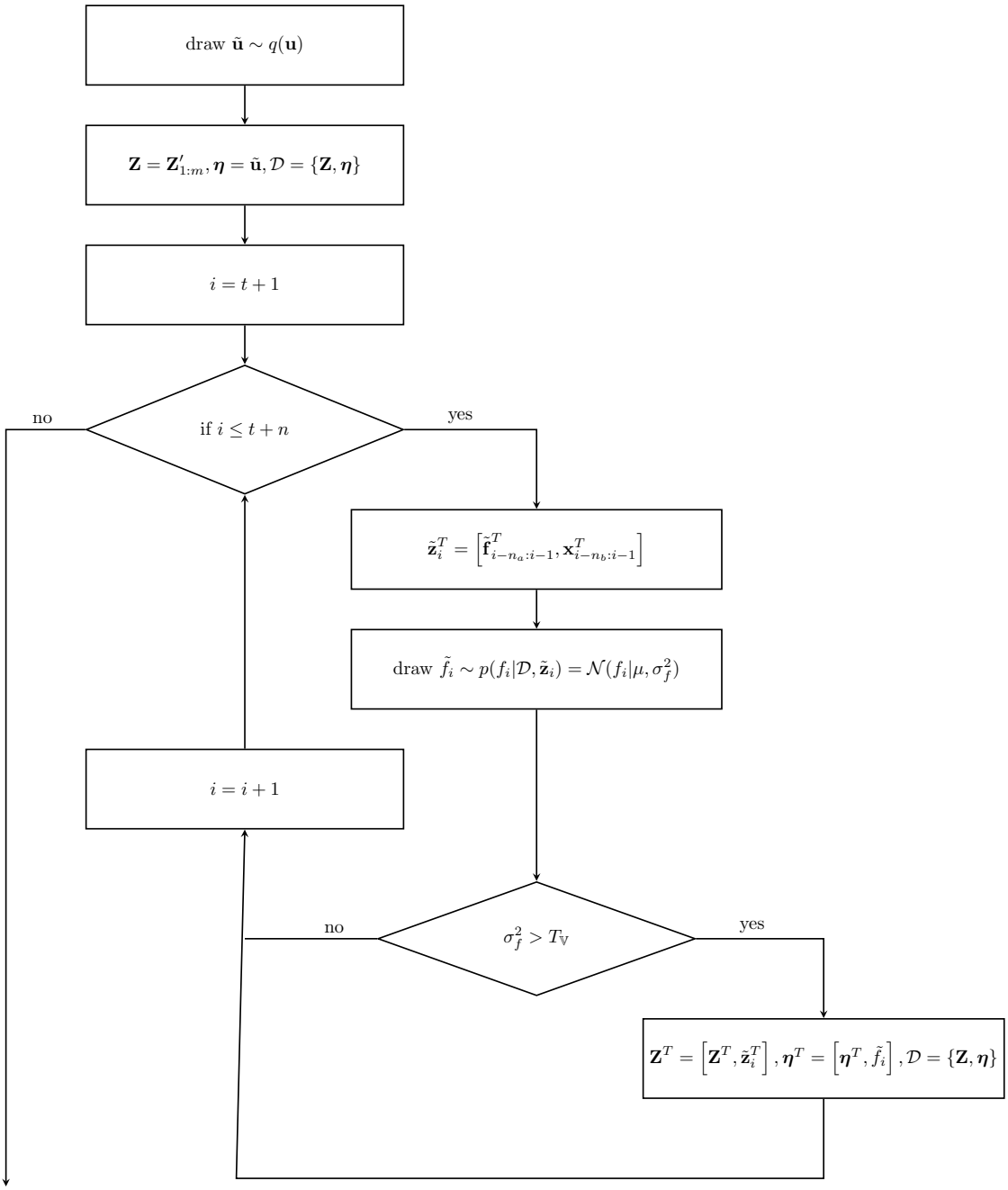


Figure 4.9: Flow chart of the TCMC simulation algorithm. The variance of the latent posterior is used to determine if the latent observations are informative. Uninformative latent observations are discarded.

4.4 Empirical Validation

In this section, we will present an empirical evaluation of the approximated simulation algorithms. We will consider two cases:

- Sequential sampling of a static function;
- Simulation of a GP-NARX model.

Firstly, we will compare the approximated simulated latent distributions between the FCMC estimation of the sequential sampling of a static function and the simulation of GP-NARX models. Secondly, running times will be considered for various covariance functions, different sparse approximations, and the number of pseudo-inducing points. Lastly, we will consider the hardware acceleration of the approximated algorithms with general-purpose graphics processing units GPGPUs.

The practical implementation of the TCMC algorithm has an important distinction from the previously described approach. In the practical TCMC algorithm, the latent samples are retained if the variance of the latent posterior of the independent MC sample at the time step considered exceeds the threshold $T_{\mathbb{V}}$ for any of the r independent MC samples. This would in practice retain more latent observations than in an individual MC run. However, adding a few latent observations that would otherwise be discarded guarantees that the dimensionality of the retained latent observations and the Cholesky factors is preserved for all independent MC samples. This allows us to carefully vectorize the algorithm and significantly reduce the running time.

The algorithms were implemented in contemporary software frameworks, namely, Tensorflow [90] and its GP-specific extension GPflow [91]. The vectorization over the r MC samples, as well as the vectorization over the algebraic expressions in independent MC samples, makes the algorithms appropriate for hardware acceleration, such as running them on GPGPU. The system specifications used in the experiments are defined in Appendix G.

To compare the sequential estimation of the static posterior, or the simulation of a GP-NARX model, we will consider a 2-Wasserstein distance defined by Equation (B.5) between two multivariate Gaussian distributions. In a static problem (or in the prediction of a dynamical system), the ground truth can be evaluated in closed-form, i.e. we compute a fully correlated multivariate distribution at test inputs, i.e.

$$\begin{aligned} p(\mathbf{f}_{t+1:t+n}|\mathbf{y}_{1:t}) &= \mathcal{N}(\mathbf{f}_{t+1:t+n}|\boldsymbol{\mu}_{t+1:t+n}, \boldsymbol{\Sigma}_{t+1:t+n}), \\ \boldsymbol{\mu}_{t+1:t+n} &\in \mathbb{R}^{1\times n}, \\ \boldsymbol{\Sigma}_{t+1:t+n} &\in \mathbb{R}^{n\times n}, \end{aligned} \tag{4.25}$$

where n denotes the number of test points, and $\mathbf{f}_{t+1:t+n}$ a dynamic notation for the latent function values. The latent function values can be for static problems equivalently denoted by $\mathbf{f}_*$. In the simulation of the GP-NARX models, the latent distribution cannot be evaluated in closed-form. The ground is, therefore, estimated from r independent FCMC samples of the trajectory, denoted by $\tilde{\mathbf{F}} \in \mathbb{R}^{n\times r}$ where the estimated mean $\hat{\boldsymbol{\mu}}_{t+1:t+n} \in \mathbb{R}^{1\times n}$ and the variance $\hat{\boldsymbol{\Sigma}}_{t+1:t+n} \in \mathbb{R}^{n\times n}$ are defined by

$$\begin{aligned} [\hat{\boldsymbol{\mu}}_{t+1:t+100}]_i = \bar{x}_i &= \frac{1}{r}\sum_{k=1}^{r} \tilde{\mathbf{F}}_{ik}, \\ \left[\hat{\boldsymbol{\Sigma}}_{t+1:t+100}\right]_{ij} &= \frac{1}{r-1}\sum_{k=1}^{r} (\tilde{\mathbf{F}}_{ik} - \bar{x}_i)(\tilde{\mathbf{F}}_{jk} - \bar{x}_j). \end{aligned} \tag{4.26}$$

The mean and the covariance matrix of the simulation approximations are estimated in the same way.

Sequential sampling of a static function

Firstly, we will consider sequential sampling on a static problem defined by

$$\mathbf{y} = \sin(3\mathbf{x}) + 0.2\cos(10\mathbf{x}) + \mathcal{N}\left(\boldsymbol{\epsilon}|\mathbf{0}, 0.15^2\mathbf{I}\right). \tag{4.27}$$

The sequential sample in a static model is algorithmically similar to the simulation of the GP-NARX models. However, the latent distribution is Gaussian and can be evaluated in closed-form. Therefore, we have an analytical solution as the ground truth to compare to.

The problem defined by Equation (4.27) was modeled by a VFE approximation and a FITC approximation with 30 pseudo-inputs. The latent distribution was estimated from r=500 MC samples. The models were trained by MLL maximization using the LBFGS optimizer. The threshold in thresholded MC sampling was selected as $T_{\mathbb{V}} = 10^{-6}$. This is equivalent to the jitter added to the covariance matrices for numerical stability.

Note that the latent distribution is compared within the model approximation, e.g. we compare a closed-form solution of the latent distribution for the VFE approximation to its respective latent distribution from estimated samples. The empirical evaluation comparing the prediction of different sparse approximations to the vanilla GP model was presented in Section 3.2.2.

Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13 show the 2-Wasserstein distance between the closed-form solution of the latent distribution and the estimated distribution from samples for various covariance functions. The sparse approximated model considered was VFE. The aforementioned figures also depict the running times of different numerical approximations of the sequential sampling with respect to the increasing number of input locations.

The running times of the FCMC estimation cubically expand and cannot even be evaluated for a large number of test inputs in practice. However, the TCMC estimation can be computed only at a fraction of the time of the FCMC estimation. The sequential sampling for 10000 test inputs for the TCMC estimation is only 5% of the running time of the FCMC estimation at 600 test inputs for the RBF covariance function. The percentage is increased when rougher functions are considered, e.g. for the Matérn($\frac{5}{2}$) covariance functions the running time is 10% of the FCMC estimation at 600 test inputs.

However, if we look closely at the 2-Wasserstein distance, which could only be computed up to the 600 test inputs for the ground truth, we can see that the TCMC estimation error is similar as in the FCMC simulation. This is due to the very low threshold value $T_{\mathbb{V}} = 10^{-6}$. Setting the threshold as low as the numerical jitter allows the TCMC estimation to recover the ground truth at only the fraction of the running time for the covariance functions that induce relatively smooth functions.

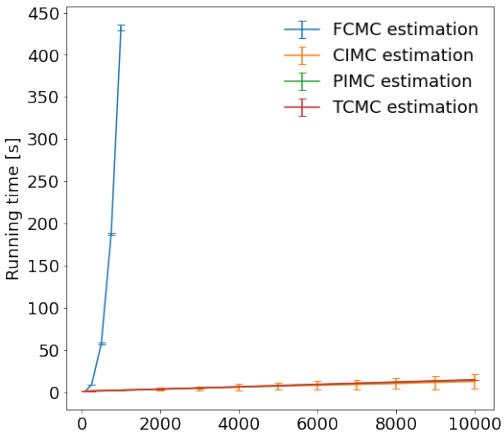
PIMC and CIMC estimations are even less computationally demanding but introduce a certain error which can be seen with the 2-Wasserstein distance presented in the aforementioned figures. This is especially evident with the CIMC estimation where the 2-Wasserstein distance is relatively high when compared to the FCMC estimation.

If the functions induced by the covariance functions are rougher, e.g. the Matérn($\frac{3}{2}$) and Matérn($\frac{1}{2}$) covariance functions, presented in Figure 4.12, Figure 4.13, then the computational benefits of the thresholded sampler are reduced. For the Matérn($\frac{1}{2}$) covariance function, the computational benefits are completely diminished. In the case of the Matérn($\frac{1}{2}$) covariance function, we usually have to resort to CIMC and PIMC estimation. However, we should expect a significant error as shown with the 2-Wasserstein distance in Figure 4.13.

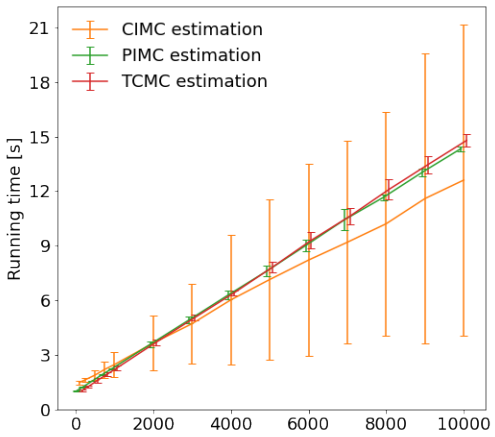
The results of the FITC sparse approximation for the RBF and the Matérn($\frac{5}{2}$) covariance functions are presented in Figure 5.2 and Figure 4.15. The conditional independence assumption in the FITC model decorrelates the latent samples to some degree. VFE approximation keeps the samples fully correlated by definition. For that reason, the running times of the TCMC estimation are increased when compared to the variational approximations. More latent samples are retained, since locally, the latent observations are not so strongly correlated. This can be seen in Figure 5.2 and Figure 4.15, where we can observe that the running times of the TCMC estimation are roughly 200% longer than with the VFE approximation.

Figure 4.16 shows the retained latent points with respect to the increasing number of test location for different covariance functions. We can observe that at some point, the number of retained points is saturated and there is no practical benefit in retaining more latent points. For smoother covariance functions, the point of saturation comes sooner when compared to the rougher kernels. Almost all latent points are retained for the Matérn($\frac{1}{2}$) covariance function, where the data points are extremely uncorrelated between neighboring inputs. The Matérn($\frac{1}{2}$) curve can, therefore, also be seen as a proxy for the retained latent points in the case of the FCMC estimation, e.g. the ground truth retained latent points with the FCMC estimation of the Matérn($\frac{3}{2}$) covariance function would very closely follow the curve from the TCMC estimation of the Matérn($\frac{1}{2}$) kernel.

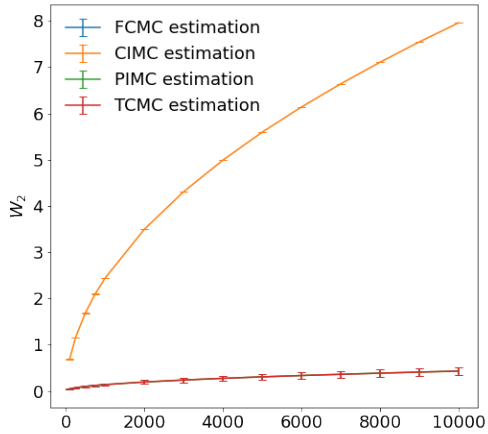
Although the sequential estimation in static problems is similar to the simulation of GP-NARX models, the problem is not the same. However, it is convenient to have a closed-form solution to compare to. In static problems, there is no input/output dependency between the consequent latent samples. For that reason, a single Cholesky factor can be computed for multiple independent MC samples, which significantly reduces the running times and memory requirements of the algorithms. In the next section, we will consider the approximations on an illustrative dynamical system.



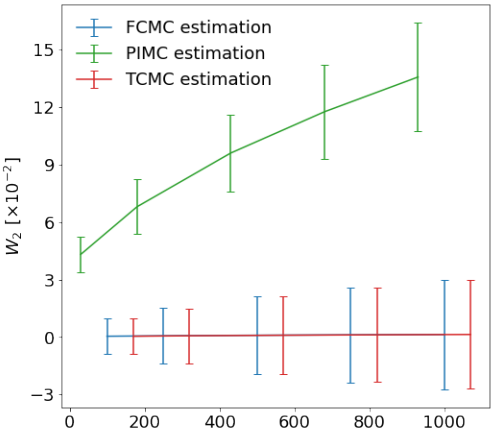
(a) Running times for the VFE approximation and the RBF covariance function.



(b) Closer view of running times for the VFE approximation and the RBF covariance function.

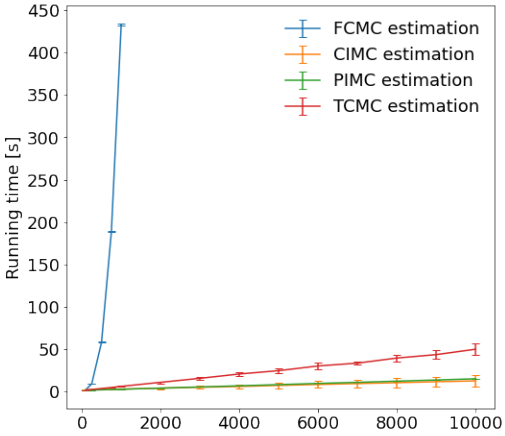


(c) 2-Wasserstein distance for the VFE approximation and the RBF covariance function.

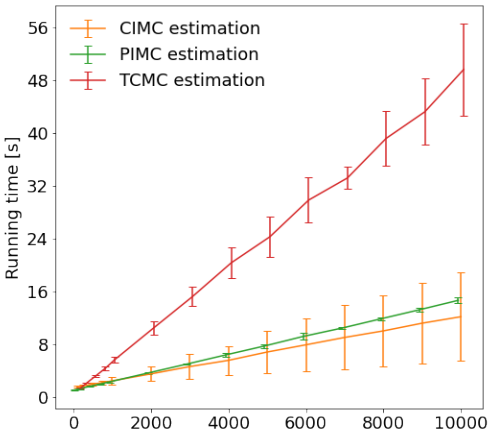


(d) Closer view of the 2-Wasserstein distance for the VFE approximation and the RBF covariance function.

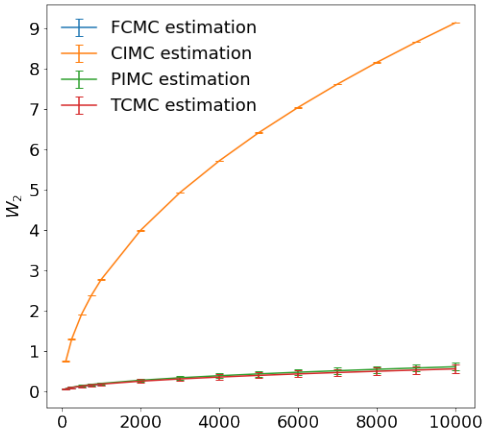
Figure 4.10: The 2-Wasserstein distance and the running times for the VFE approximation in the sequential estimation of the latent distribution using the RBF covariance function.



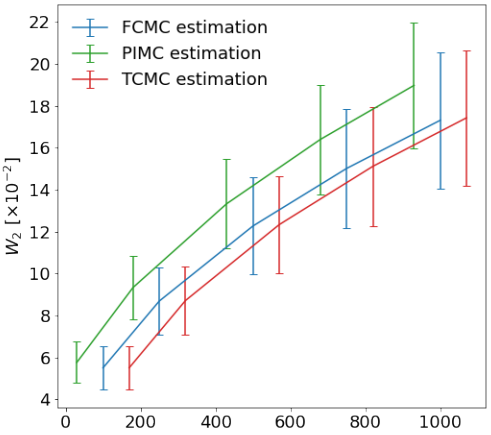
(a) Running times for the VFE approximation and the Matérn($\frac{5}{2}$) covariance function.



(b) Closer view of running times for the VFE approximation and the Matérn($\frac{5}{2}$) covariance function.

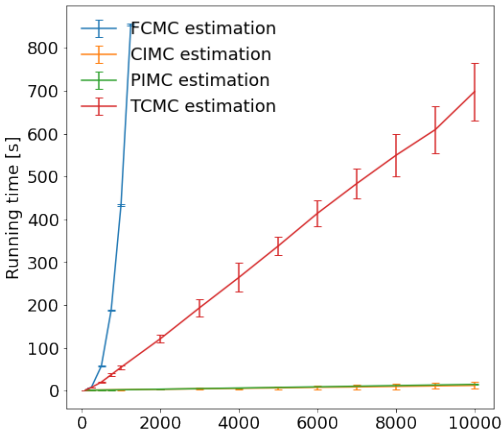


(c) 2-Wasserstein distance for the VFE approximation and the Matérn($\frac{5}{2}$) covariance function.

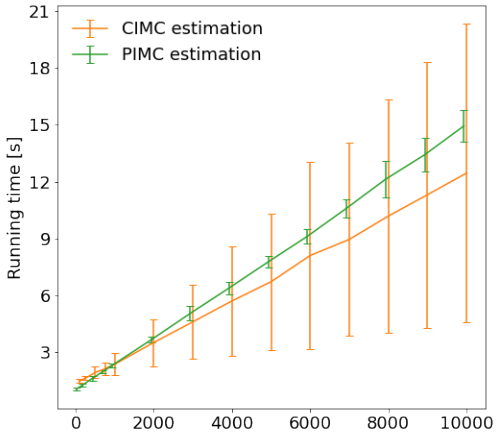


(d) Closer view of the 2-Wasserstein distance for the VFE approximation and the Matérn($\frac{5}{2}$) covariance function.

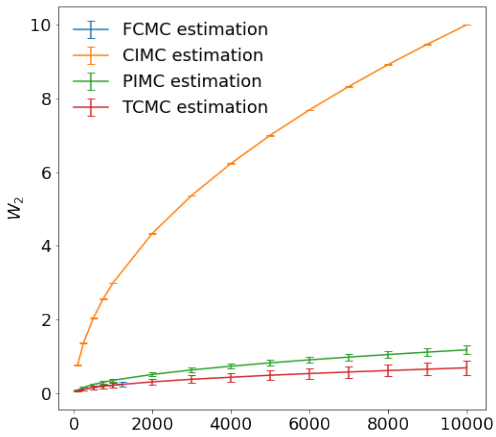
Figure 4.11: The 2-Wasserstein distance and the running times for the VFE approximation in the sequential estimation of the latent distribution using the Matérn($\frac{5}{2}$) covariance function.



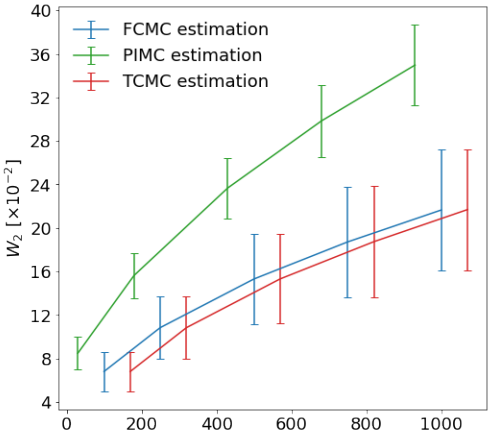
(a) Running times for the VFE approximation and the Matérn($\frac{3}{2}$) covariance function.



(b) Closer view of running times for the VFE approximation and the Matérn($\frac{3}{2}$) covariance function.

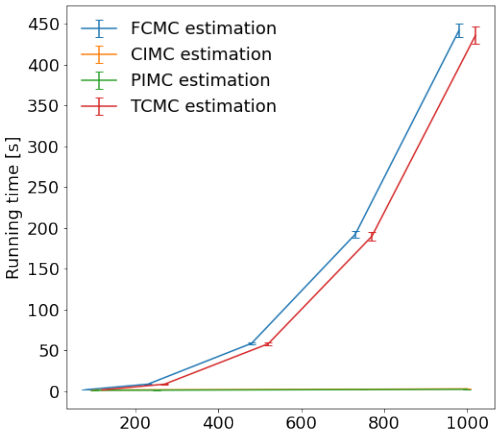


(c) 2-Wasserstein distance for the VFE approximation and the Matérn($\frac{3}{2}$) covariance function.

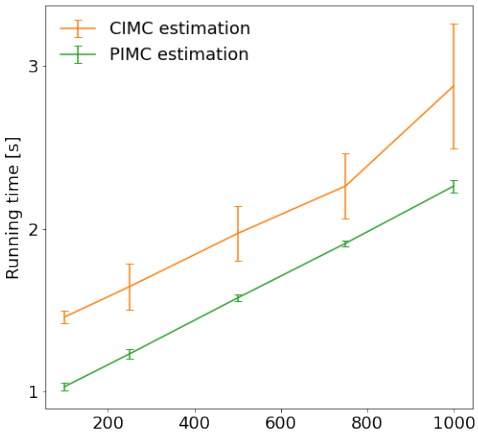


(d) Closer view of the 2-Wasserstein distance for the VFE approximation and the Matérn($\frac{3}{2}$) covariance function.

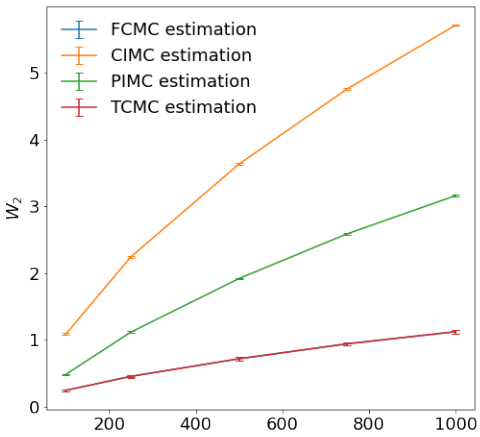
Figure 4.12: The 2-Wasserstein distance and the running times for the VFE approximation in the sequential estimation of the latent distribution using the Matérn($\frac{3}{2}$) covariance function.



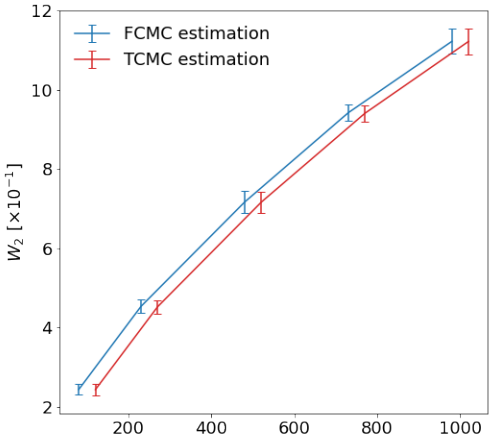
(a) Running times for the VFE approximation and the Matérn($\frac{1}{2}$) covariance function.



(b) Closer view of running times for the VFE approximation and the Matérn($\frac{1}{2}$) covariance function.

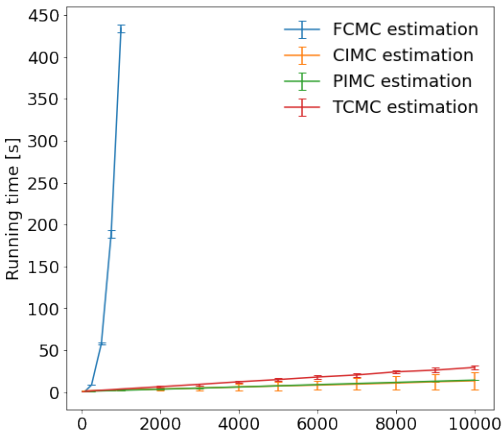


(c) 2-Wasserstein distance for the VFE approximation and the Matérn($\frac{1}{2}$) covariance function.

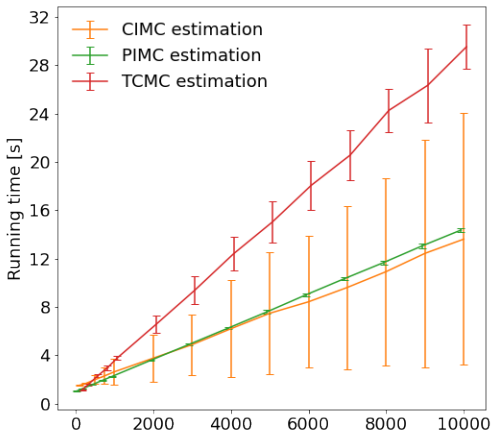


(d) Closer view of the 2-Wasserstein distance for the VFE approximation and the Matérn($\frac{1}{2}$) covariance function.

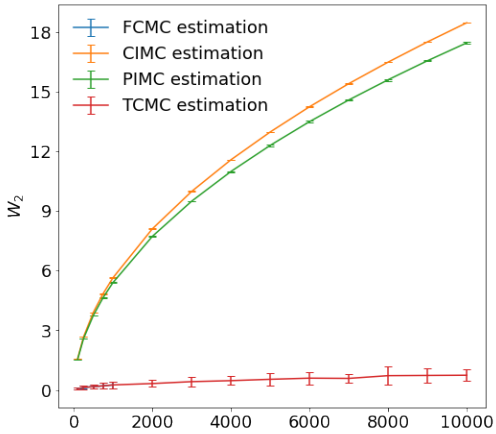
Figure 4.13: The 2-Wasserstein distance and the running times for the VFE approximation in the sequential estimation of the latent distribution using the Matérn($\frac{1}{2}$) covariance function.



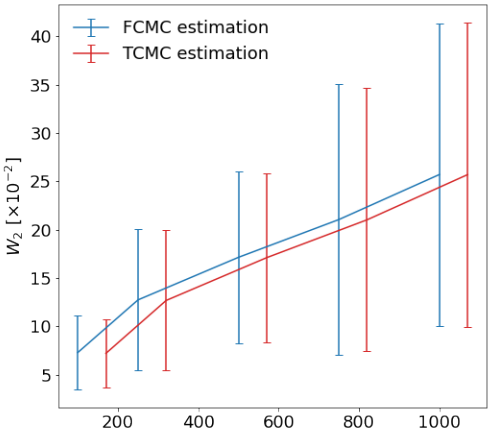
(a) Running times for the FITC approximation and the RBF covariance function.



(b) Closer view of running times for the FITC approximation and the RBF covariance function.

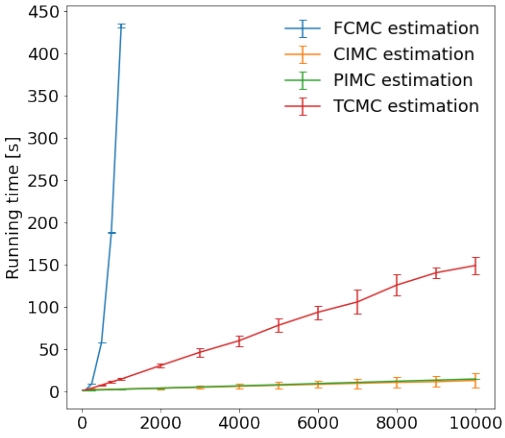


(c) 2-Wasserstein distance for the FITC approximation and the RBF covariance function.

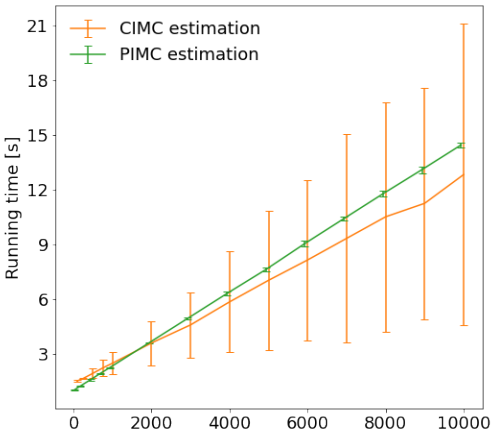


(d) Closer view of the 2-Wasserstein distance for the FITC approximation and the RBF covariance function.

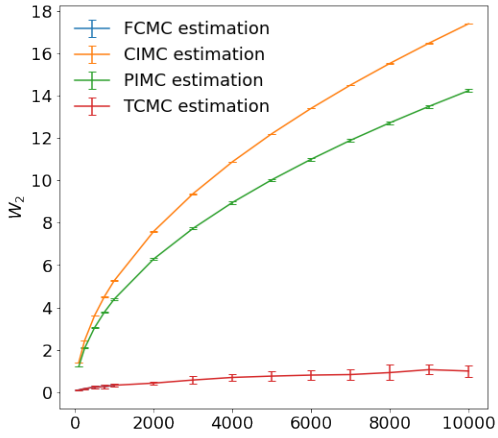
Figure 4.14: The 2-Wasserstein distance and the running times for the FITC approximation in the sequential estimation of the latent distribution using the RBF covariance function.



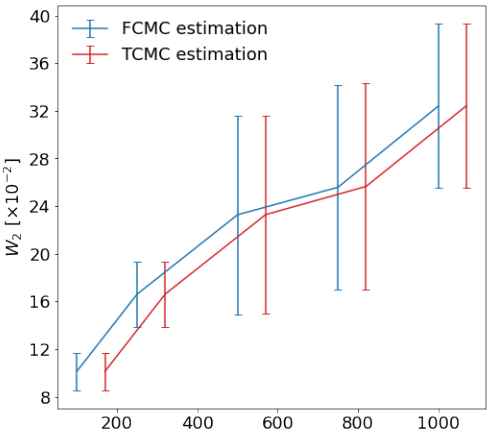
(a) Running times for the FITC approximation and the Matérn($\frac{5}{2}$) covariance function.



(b) Closer view of running times for the FITC approximation and the Matérn($\frac{5}{2}$) covariance function.

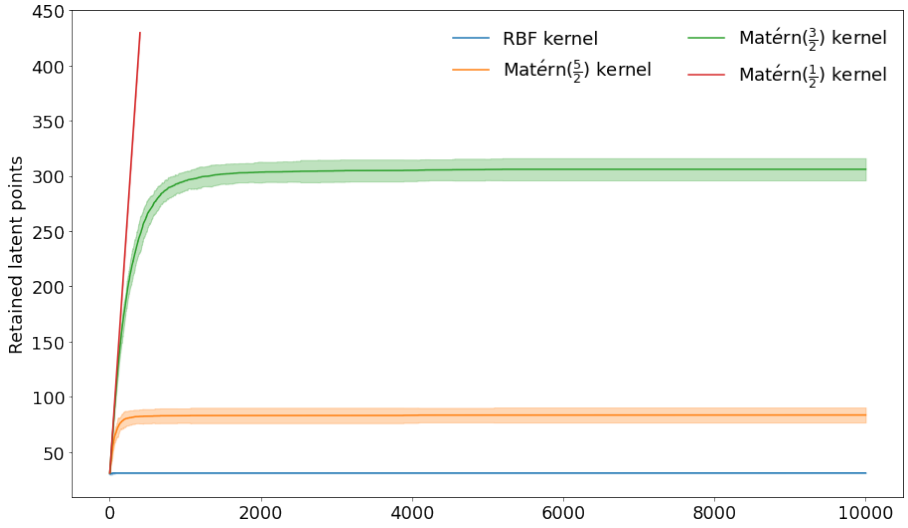


(c) 2-Wasserstein distance for the FITC approximation and the Matérn($\frac{5}{2}$) covariance function.

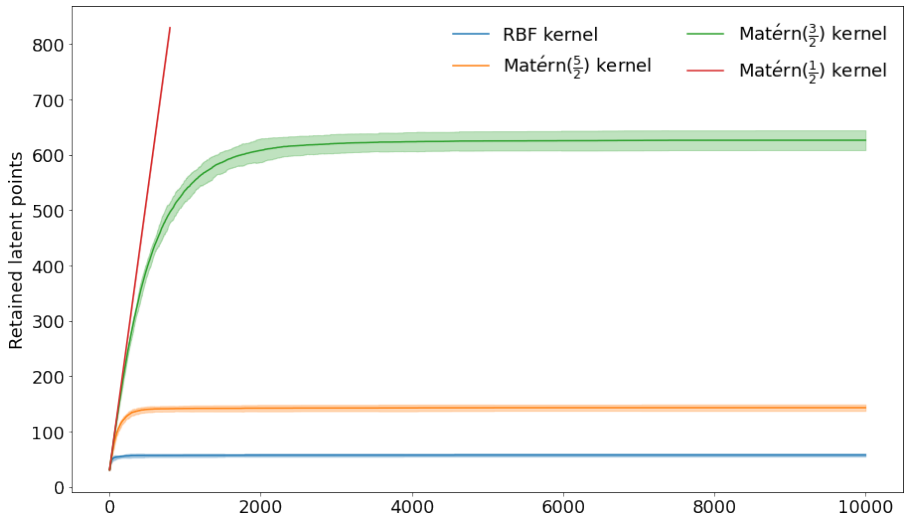


(d) Closer view of the 2-Wasserstein distance for the FITC approximation and the Matérn($\frac{5}{2}$) covariance function.

Figure 4.15: The 2-Wasserstein distance and the running times for the FITC approximation in the sequential estimation of the latent distribution using the Matérn($\frac{5}{2}$) covariance function.



(a) Retained latent points versus the number of input locations for the VFE approximation.



(b) Retained latent points versus the number of input locations for the FITC approximation.

Figure 4.16: Retained latent points versus the number of input locations for different kernels. The results are presented for 10 random restarts, where the solid regions represent the 2 standard deviation intervals.

Simulation of a GP-NARX model

The dynamic problem that is considered in this section is the illustrative example defined by Equation (2.82). For the training data, the system was initialized by $[f_0^1, f_0^2] = [0.300, 0.800]$ and simulated for 900 samples. For the test data, the system was initialized by $[f_0^1, f_0^2] = [0.380, 0.326]$ and simulated for the number of samples considered. The data were corrupted by Gaussian noise with $\sigma_n^2 = 10^{-2}$. The parameters of the NARX models were selected by $n_a = n_b = 4$. The system was modeled by a GP-NARX (VFE) model with 50 pseudo-inputs. The latent distribution was estimated using r=500 MC samples. Hyperparameters were obtained by MLL maximization using the LBFGS optimizer. The threshold in the TCMC simulation was selected as $T_{\mathbb{V}} = 10^{-6}$. The algorithms for the respective simulation approximation are presented in Appendix F.

We want to emphasize that the latent distribution is compared within the model approximation, e.g. we compare the latent distribution in the simulation of the GP-NARX (VFE) model to the respective latent distribution from approximations to the simulation.

Figure 4.17, Figure 4.18, Figure 4.19, and Figure 4.20 show the running times of the simulation up to 10000 steps into the future. We can see that the running times of the FCMC simulation expand quickly as in the static case. For long prediction horizons, the FCMC cannot be realized in practice.

The approximations are computed in reasonable time, which is evident from Figure 4.17, Figure 4.18, and Figure 4.19 for the covariance functions that induce relatively smooth functions. However, the running times in simulation are significantly larger compared to the static estimation since a separate Cholesky factor has to be computed for each independent MC sample.

For the Matérn($\frac{1}{2}$) covariance function in Figure 4.20, the running times of the TCMC simulation are not reduced when compared to the FCMC simulation. In this case, the TCMC simulation is also not practically realizable for large prediction horizons since almost all latent samples are retained and the simulation does not differ much from the FCMC simulation. One can use an approximation that introduces additional assumptions, such as the PIMC and CIMC simulation, but can expect some error in the estimation of the latent response, which can be seen with the 2-Wasserstein distance in Figure 4.20.

However, the CIMC approximation that was generally used in the literature provides zero benefits when compared to the proposed PIMC simulation. The estimated response using a PIMC simulation better approximates the ground truth and can be obtained in identical computational time as the CIMC simulation. Therefore, the PIMC simulation should be preferred to the CIMC simulation in cases when the TCMC simulation cannot be obtained.

For RBF, Matérn($\frac{5}{2}$), and Matérn($\frac{3}{2}$) covariance functions, the TCMC simulation significantly reduced the running time of the simulation. The 2-Wasserstein distance is practically the same as in the FCMC simulation. The threshold $T_{\mathbb{V}} = 10^{-6}$ selected in this section is equal to the jitter that is added to the covariance matrices for numerical stability. It is the amount of variance we are prepared to sacrifice for a better conditioned numerical problem. With the threshold equal to the numerical jitter, the TCMC simulation can be considered exact and is equivalent to the FCMC simulation. However, it can be computed in a reasonable time and can be scaled to larger predictive horizons.

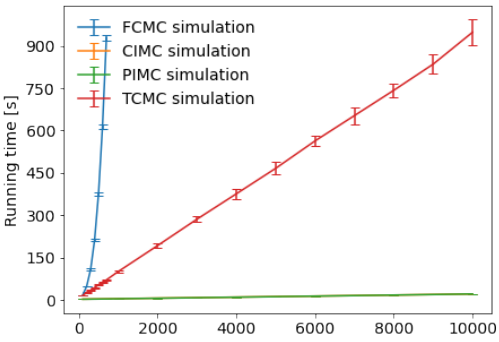
Similarly as in the static case, Figure 4.22 shows the retained latent points with respect to the increasing number of predicted steps into the future for different covariance function. We can again observe that at some point, the number of retained points is saturated and there is no practical benefit in retaining more latent points. Again, the curve for the Matérn($\frac{1}{2}$) covariance function can serve as the proxy for the retained latent points in the FCMC simulation.

Since the Cholesky factors are computed individually for each MC sample, and the algorithms are carefully vectorized, the simulation algorithms are more appropriate for hardware acceleration based on parallel execution of the matrix operations than their static counterparts. Figure 4.21 shows the comparison of running the simulation algorithms on the CPU and the GPGPU. We reduced the number of MC samples to $r = 100$ because we were constrained by our computational resources, i.e. the memory of our GPU. The figure shows the running times with respect to the increasing number of pseudo-inputs.

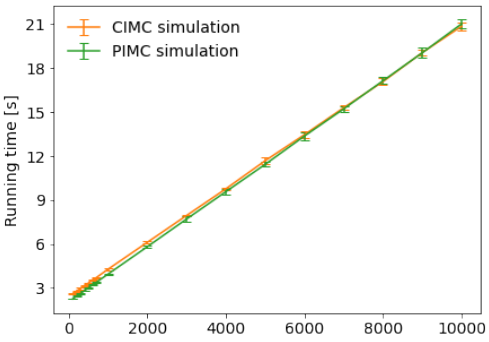
We can observe that when running the simulation using 250 pseudo-inputs for training, the GPGPU computation is slower than the computation on the CPU. When the number of pseudo-inputs is increased to 500 and 750, the GPGPU-accelerated algorithms tend to perform faster. A possible explanation is that due to the data transfer from the CPU to the GPU (and back), some latency is introduced. The computational benefits of the GPGPU acceleration become evident when the matrices are larger, in our case this is when the number of pseudo-inputs is larger. In this case, the benefits of the parallel execution of matrix products outweigh the overhead of the additional latency introduced by the communication between the CPU and the GPU.

In this section, we showed how the TCMC simulation can estimate the exact latent response. The simulation is obtained in only a fraction of the running time needed to estimate the FCMC simulation. In the case of the Matérn($\frac{1}{2}$) covariance function, the computational gains are completely diminished. The reduction of the running times in simulation is most evident when the covariance functions used induce relatively smooth functions. In practice, this does not pose a significant limitation since smooth functions tend to generalize better and are more commonly used.

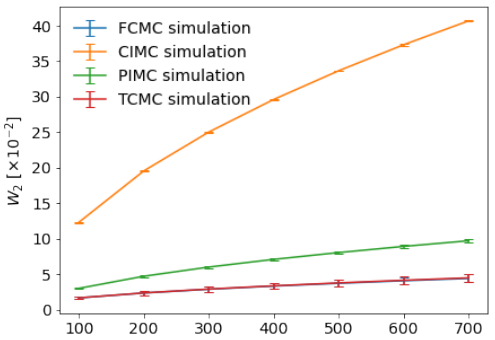
In the next section, we will empirically justify the selection of smoother covariance functions on two benchmarks from dynamic systems literature. The TCMC simulation will be considered for the simulation in large validation data sets. Previously, the correlated samples of the simulated response could not be obtained in practice due to the computational complexity of the correlated algorithm.



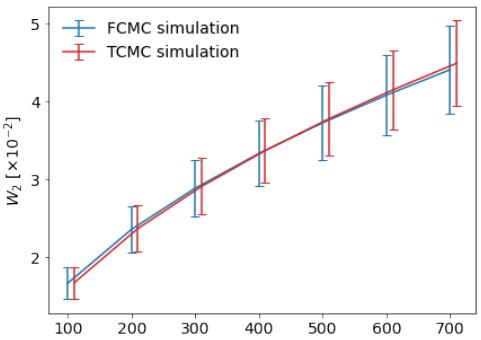
(a) Running time comparison of the simulation.



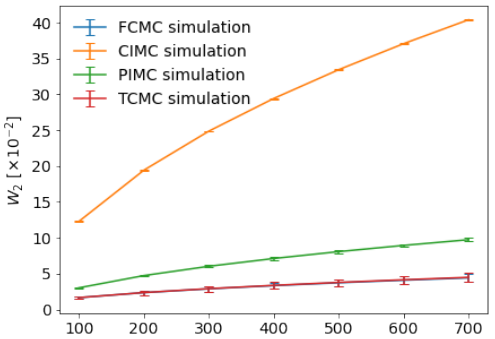
(b) Closer view of the running time comparison of the simulation.



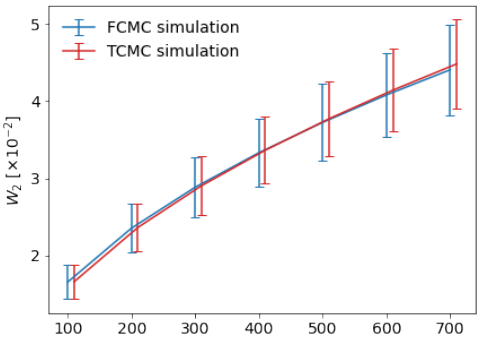
(c) 2-Wasserstein distance for Predator.



(d) Closer view of the 2-Wasserstein distance for Predator.

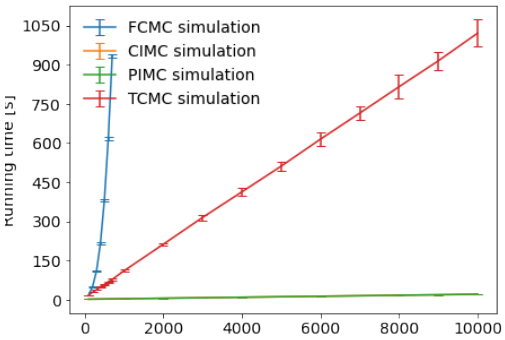


(e) 2-Wasserstein distance for Prey.

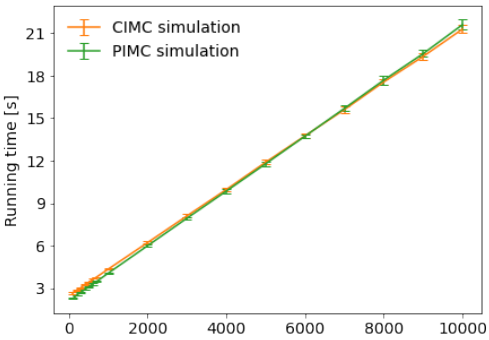


(f) Closer view of the 2-Wasserstein distance for Prey.

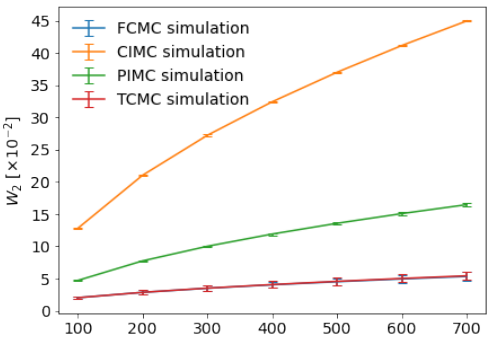
Figure 4.17: The 2-Wasserstein distance and the running times for the simulation of the GP-NARX (VFE) model using the RBF covariance function.



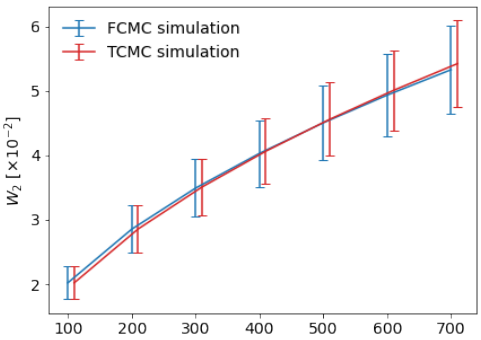
(a) Running time comparison of the simulation.



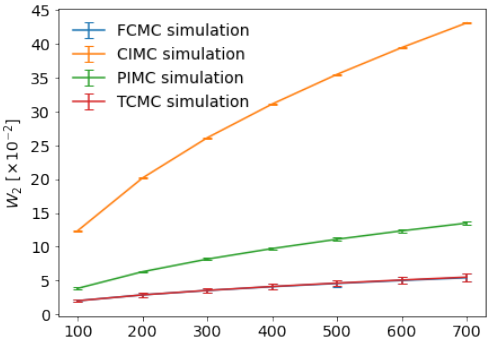
(b) Closer view of the running time comparison of the simulation.



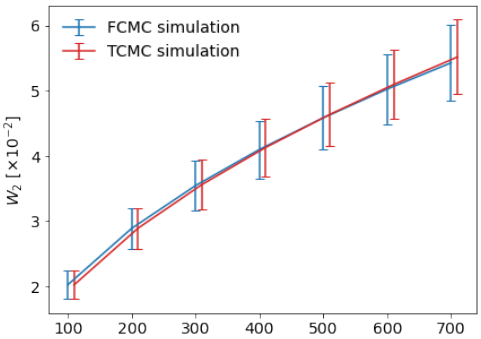
(c) 2-Wasserstein distance for Predator.



(d) Closer view of the 2-Wasserstein distance for Predator.

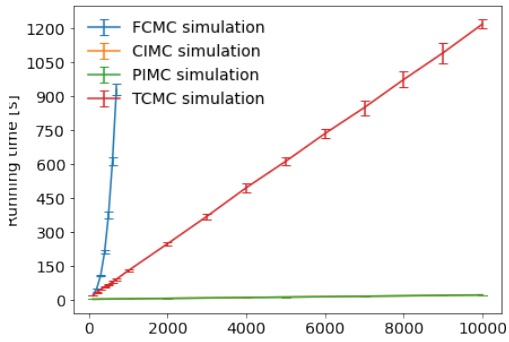


(e) 2-Wasserstein distance for Prey.

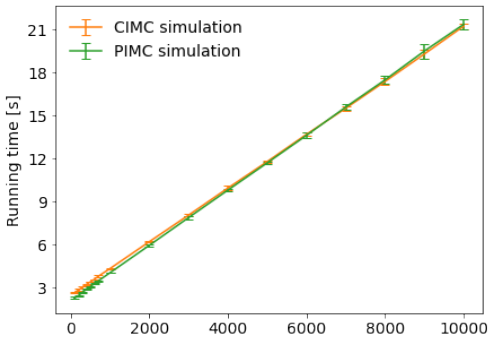


(f) Closer view of the 2-Wasserstein distance for Prey.

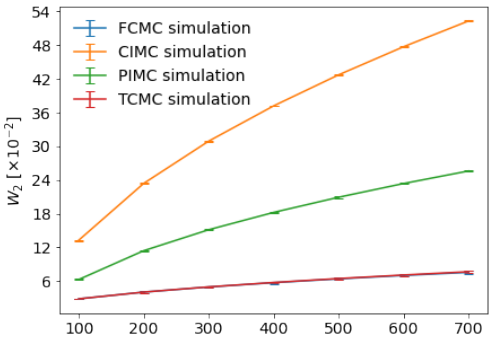
Figure 4.18: The 2-Wasserstein distance and the running times for the simulation of the GP-NARX (VFE) model using the Matérn($\frac{5}{2}$) covariance function.



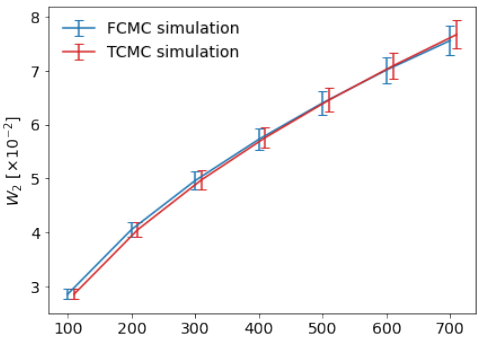
(a) Running time comparison of the simulation.



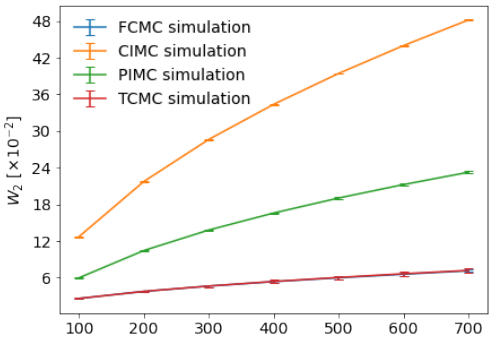
(b) Closer view of the running time comparison of the simulation.



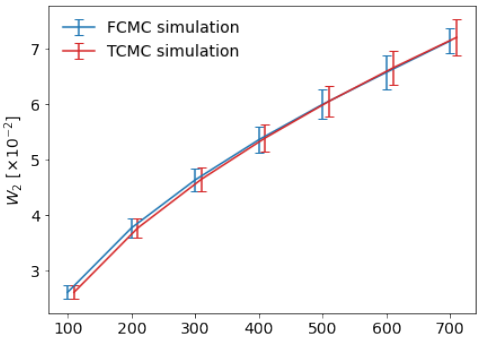
(c) 2-Wasserstein distance for Predator.



(d) Closer view of the 2-Wasserstein distance for Predator.

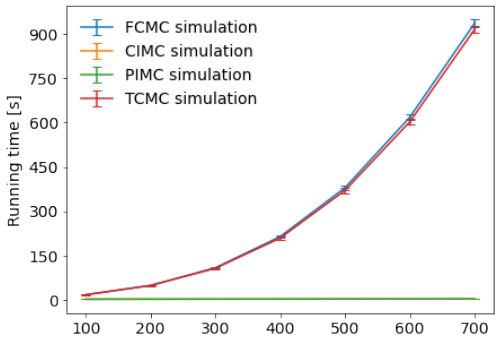


(e) 2-Wasserstein distance for Prey.

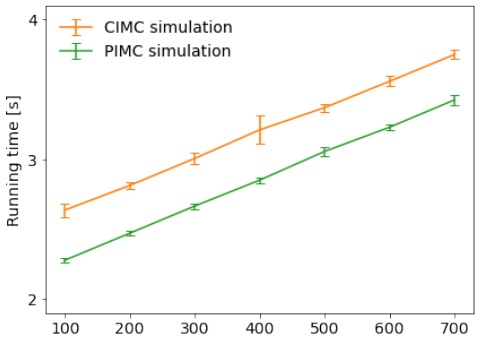


(f) Closer view of the 2-Wasserstein distance for Prey.

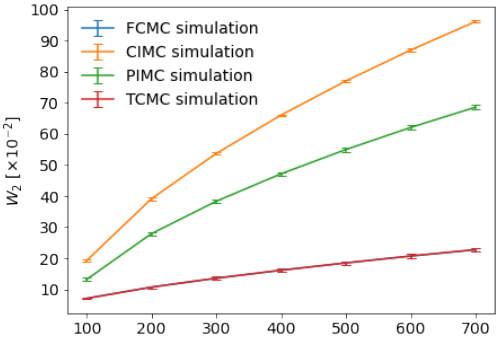
Figure 4.19: The 2-Wasserstein distance and the running times for the simulation of the GP-NARX (VFE) model using the Matérn($\frac{3}{2}$) covariance function.



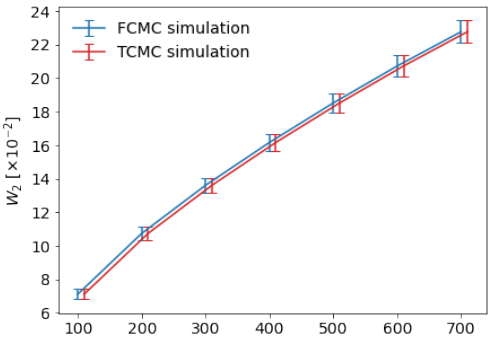
(a) Running time comparison of the simulation.



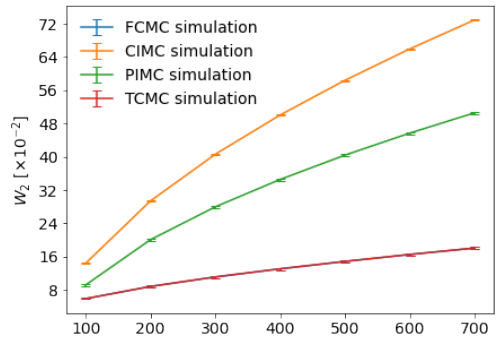
(b) Closer view of the running time comparison of the simulation.



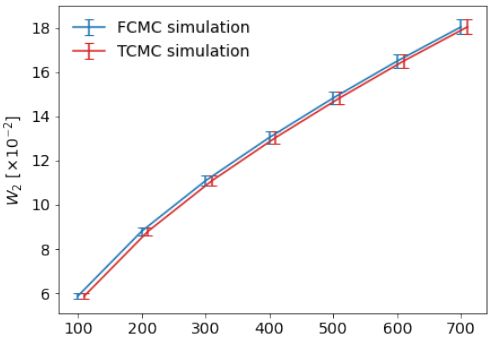
(c) 2-Wasserstein distance for Predator.



(d) Closer view of the 2-Wasserstein distance for Predator.

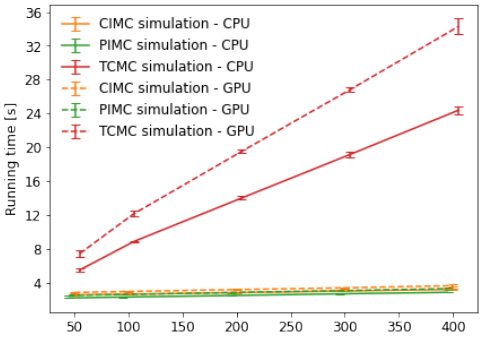


(e) 2-Wasserstein distance for Prey.

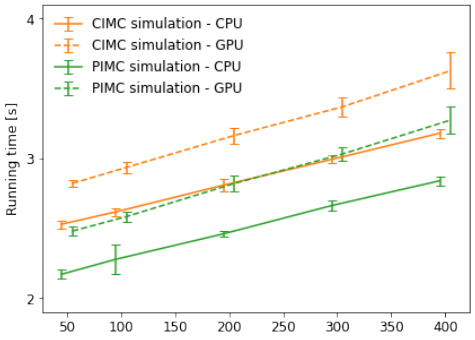


(f) Closer view of the 2-Wasserstein distance for Prey.

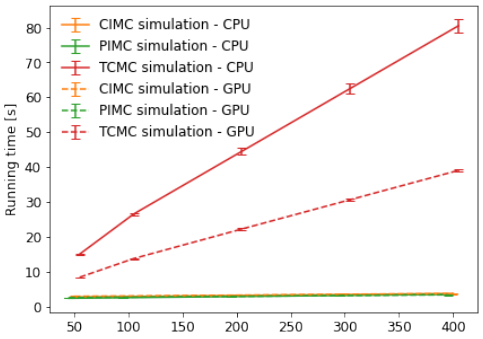
Figure 4.20: The 2-Wasserstein distance and the running times for the simulation of the GP-NARX (VFE) model using the Matérn($\frac{1}{2}$) covariance function.



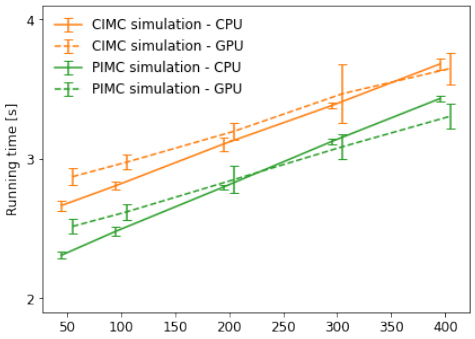
(a) Comparison of running times between CPU and GPGPU for 250 pseudo-inputs.



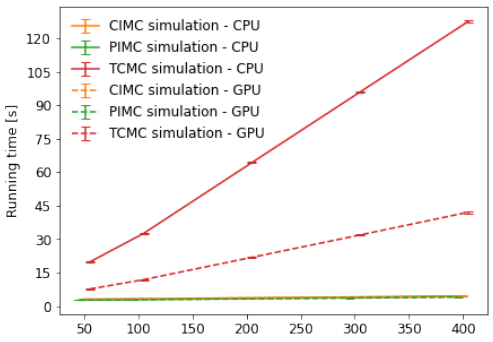
(b) Closer view of the comparison of running times between CPU and GPGPU for 250 pseudo-inputs.



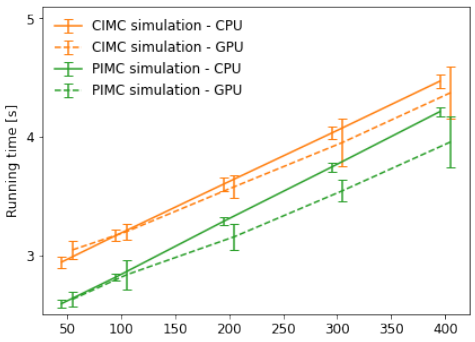
(c) Comparison of running times between CPU and GPGPU for 500 pseudo-inputs.



(d) Closer view of the comparison of running times between CPU and GPGPU for 500 pseudo-inputs.



(e) Comparison of running times between CPU and GPGPU for 750 pseudo-inputs.



(f) Closer view of the comparison of running times between CPU and GPGPU for 750 pseudo-inputs.

Figure 4.21: Comparison of the running times between the CPU and the GPGPU for the simulation of the GP-NARX (VFE) model using the RBF covariance function.

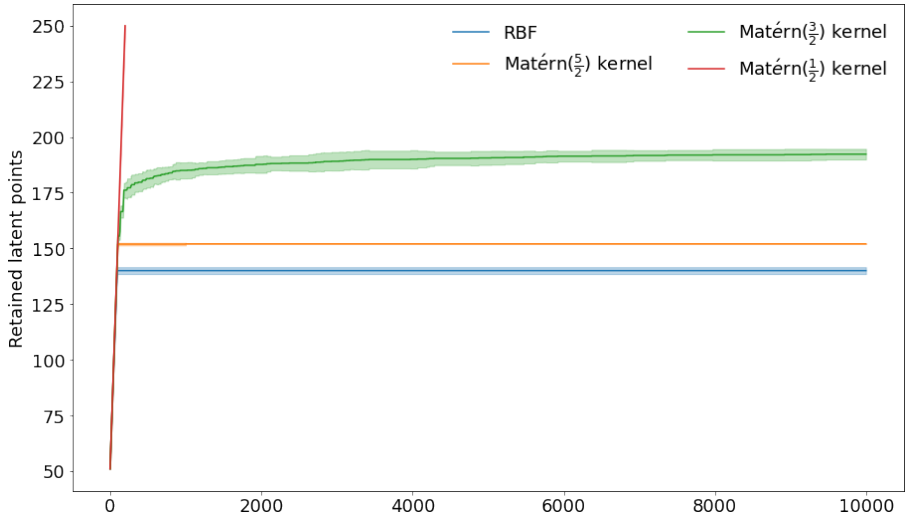


Figure 4.22: Retained latent points versus the number of predicted steps into the future for different kernels in GP-NARX (VFE) model. The results are presented for 10 random restarts, where the solid regions represent the 2 standard deviation intervals.

4.5 Nonlinear Benchmarks for Dynamic System Identification

As demonstrated in the previous section, the TCMC simulation recovers the ground truth in significantly less computational time when the functions were relatively smooth. However, the computational benefits diminish for covariance functions that induce rougher functions. This is generally not a problem in practice for the following reasons:

1. For small data sets, smooth functions are usually preferred because they generalize better;
2. For large data sets, the choice of the covariance function is less important;
3. For large data sets, the uncertainty of the latent function posterior is relatively small.

The first point states that well-regularized models usually perform better. This is introduced to the model with some sort of a smoothness assumption. For GP-specific models, this is defined on a meta-level, i.e. with the properties of the functions induced from the respective covariance functions. An example of a smoothness assumption is the level of differentiability, i.e. how many times is the function differentiable.

The second point is concerned with the choice of the covariance function. In large data sets, the data penalize overly flexible models and the selected kernel is less important. We expect that the model performance when using various covariance functions, but appropriately flexible to reduce the bias, will not be significantly different. In the simulation of GP-NARX models we can take this into account if the goal of the model is to simulate the data over long prediction horizons. We could prefer a smoother covariance function if the performance of the model does not change significantly.

The third point states that the uncertainty of the expected latent function posterior is relatively small in large data sets. The latent uncertainty represents the epistemic uncertainty, which is by definition the uncertainty due to the lack of observed data. When more data are observed, the latent uncertainty is reduced and fewer latent samples are required to sufficiently specify the latent function.

The dynamical system used in the previous section was relatively simple and can be considered an illustrative example. We did not deal with model selection, e.g. determining the best covariance function, but rather just benchmarked the performance of the simulation algorithms. In this section, the modeling of two dynamical benchmarks will be considered. They both provide a relatively large training data set, where the vanilla GP-NARX model cannot be used in practice [69]. Even with sparse models approximating the model training, the test data sets are too large to practically realize the FCMC simulation. We will demonstrate the application of the TCMC simulation on the two benchmarks.

4.5.1 Silverbox

Silverbox benchmark represents the second-order linear time-invariant system with the third-degree polynomial static nonlinearity around it in feedback. It is a real-world system, where the data were collected from an electric circuit. The system theoretically obeys the equation

$$m\frac{d^2y(t)}{dt^2} - d\frac{dy(t)}{dt} + ay(t) + by(t)^3 = u(t). \tag{4.28}$$

We selected the training data from samples 40,586 to 127,410, and the test data as the first 40,495 samples [106]. The system was excited with a white Gaussian noise sequence filtered by a ninth-order discrete-time Butterworth filter with a cut-off frequency of 200 Hz for the test data. The amplitude was varied linearly over the interval from zero to its maximal value. Training data were obtained with the excitation of 10 successive realizations of a random odd multi-sine signal. More detailed information about the benchmark can be found in [107]. We additionally standardized the data and added a Gaussian noise to the observations with the standard deviation of $\sigma_n = 5 \times 10^{-2}$.

The meta-parameters of the NARX model were selected as $n_b = n_a = 10$. We used 300 pseudo-inputs initialized with the k-means clustering algorithm as suggested in [35]. The hyperparameters and the pseudo-input locations were determined by MLL maximization using the LBFGS optimizer.

Table 4.2 shows the prediction results of the 10-fold cross-validation on the training data. We can see that in the case of the Silverbox data, the combination of a linear and a Matérn($\frac{5}{2}$) covariance function performs best, followed by the combination of a linear and a Matérn($\frac{3}{2}$) covariance function. The performance measures show that in the case of the prediction, the model describes the data well. In terms of the mean values of the prediction, this can be seen with the RMSE and the standardized mean squared error (SMSE) metrics, and in terms of probability distributions, this can be seen with the mean standardized log loss (MSLL) metric.

Table 4.3 shows the results of the prediction and the simulation on the test data for the best two covariance functions selected by cross-validation. The simulation results are presented for two options:

1. Simulation without noise propagation (NP);
2. Simulation with NP;

The first option is the TCMC simulation described by Equation 4.24. The threshold was selected as $T_{\mathbb{V}} = 10^{-3}$. The second option propagates the likelihood noise in the inputs as described in Equation (2.79), but otherwise uses the same TCMC simulation. We see in Table 4.3 that the data is well modeled even in the case of simulation, where the simulation without NP performs better. The best covariance function selected by prediction in cross-validation also performs best in simulation. The response on a part of the test data is shown in Figure 4.23.

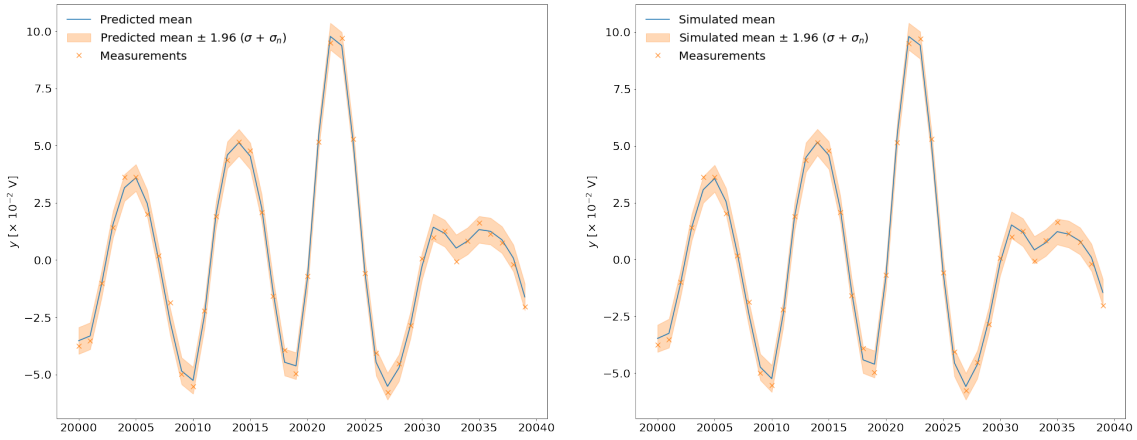
We want to emphasize that the FCMC simulation cannot be obtained for trajectories such as the one presented in the Silverbox data set, where the number of test samples is above 40000. However, with the proposed TCMC simulation we can obtain the correlated latent response. The simulation response was estimated using $r = 500$ independent MC samples of the trajectory.

Table 4.2: Means and the respective 95% confidence intervals of the RMSE, SMSE, and MSLL for the prediction on the Silverbox data. The presented performance metrics are for 10-fold cross-validation over the training data. The best values are emphasized in bold. A linear covariance function (L.) was added to the respective covariance functions.

	L. + RBF	L. + Matérn($\frac{5}{2}$)	L. + Matérn($\frac{3}{2}$)	L. + Matérn($\frac{1}{2}$)
RMSE [$\times 10^{-3}$]	3.284 ± 0.514	**2.913 ± 0.047**	2.915 ± 0.045	2.936 ± 0.052
SMSE [$\times 10^{-3}$]	3.640 ± 1.028	**2.851 ± 0.129**	2.856 ± 0.126	2.896 ± 0.139
MSLL	-2.813 ± 0.151	**-2.930 ± 0.023**	-2.929 ± 0.022	-2.922 ± 0.023

Table 4.3: Results of the prediction and the simulation on the Silverbox test data for the best 2 covariance functions selected by cross-validation. NP denotes noise propagation.

		Prediction	Simulation	Simulation with NP
L. + Matérn($\frac{5}{2}$)	RMSE [$\times 10^{-3}$]	**2.982**	3.014	**3.010**
	SMSE [$\times 10^{-3}$]	**3.105**	3.173	**3.164**
	MSLL	**-2.894**	**-2.912**	-2.860
L. + Matérn($\frac{3}{2}$)	RMSE [$\times 10^{-3}$]	3.069	3.507	3.504
	SMSE [$\times 10^{-3}$]	3.290	4.296	4.289
	MSLL	2.868	-2.768	-2.762



(a) Prediction on the Silverbox test data.

(b) Simulation on the Silverbox test data.

Figure 4.23: Graphical comparison of the prediction and the simulation on the Silverbox test data. Only a subset of data is shown. The responses are shown for the combination of a linear and a Matérn($\frac{5}{2}$) covariance function.

4.5.2 Bouc-Wen

Bouc-Wen benchmark represents the vibrations of a single degree of freedom Bouc-Wen system, representing a hysteretic process. Hysteresis is a dynamical nonlinearity, where the input-output loop persists when the input frequency approaches zero [108]. The system is mathematically defined by

$$m_L \frac{d^2y(t)}{dt^2} + r(y(t), \frac{dy(t)}{dt}) + z(y(t), \frac{dy(t)}{dt}) = u(t), \tag{4.29a}$$

$$r(y(t), \frac{dy(t)}{dt}) = k_L y(t) + c_L \frac{dy(t)}{dt}, \tag{4.29b}$$

$$z(y(t), \frac{dy(t)}{dt}) = \alpha \frac{dy(t)}{dt} - \beta(\gamma|\frac{dy(t)}{dt}||z|^{\upsilon-1}z + \delta\frac{dy(t)}{dt}|z|^{\upsilon}). \tag{4.29c}$$

Training data consist of a multi-sine excitation signal, where 5 periods of the input and output signal were recorded with 40,960 samples. Test data consists of a period excited with a multi-sine signal, with a length of 8,192. For a detailed explanation of model parameters and how the data were generated see [109]. We additionally standardized the data and added Gaussian noise to the observations with the standard deviation of $\sigma_n = 5 \times 10^{-2}$.

The meta-parameters of the NARX model were selected as $n_b = n_a = 10$. As in the Silverbox case, we used 300 pseudo-inputs initialized with the k-means clustering algorithm. The hyperparameters were determined by MLL maximization using the LBFGS optimizer.

Table 4.4 shows the results of the 10-fold cross-validation on the training data. We can see that in the case of the Bouc-Wen data set, a combination of a linear and a RBF covariance function performs best, followed by the combination of a linear and a Matérn($\frac{5}{2}$) covariance function. The performance measures show that in the case of the prediction, the model describes the data well.

Table 4.5 shows the prediction and the simulation results on the test data for the best two covariance functions selected by cross-validation. The results show that the data are well modeled in a probabilistic sense, as well as in the mean sense. The RBF covariance function selected by cross-validation also performs best for prediction on the test data set. However, in simulation, the Matérn($\frac{5}{2}$) covariance function performs significantly better. The response on a part of the test data is shown in Figure 4.24. The simulation response was estimated using $r = 500$ independent MC samples of the trajectory and $T_{\mathbb{V}} = 10^{-3}$.

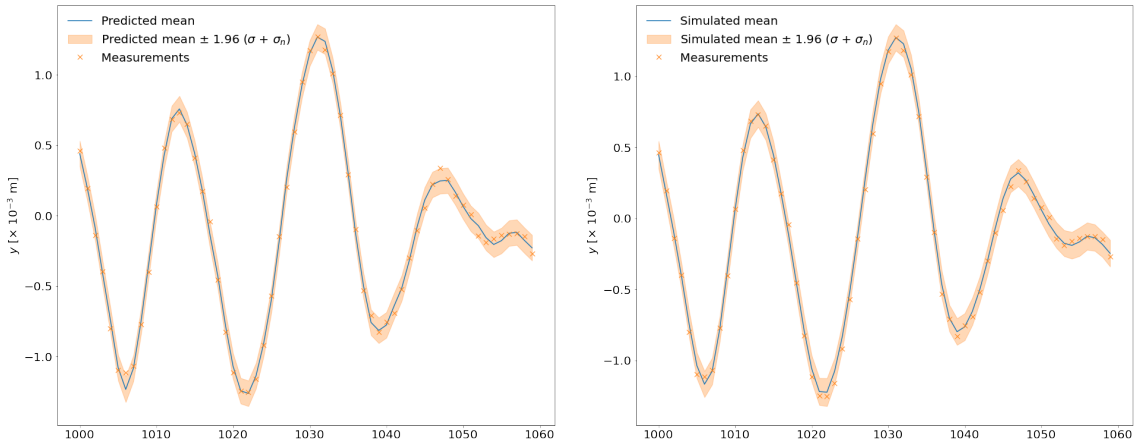
In this chapter, we presented a numerical simulation algorithm that unifies the simulation of the vanilla GP and its sparse approximations. We have proposed an approximation to the simulation that can approximate the ground truth up to a user-defined threshold. If the threshold is defined as the numerical jitter, the TCMC simulation recovers the ground truth while significantly reducing the time of the computation. This is especially evident in large data sets where the posterior uncertainty is small and when smooth covariance functions are used. In the next two chapters, we will present two case studies where the quantification of uncertainty is important and the nature of the problem requires scalable and flexible simulation algorithms.

Table 4.4: Means and the respective 95% confidence intervals of the RMSE, SMSE, and MSLL for the prediction on the Bouc-Wen data. The presented performance metrics are for 10-fold cross-validation over the training data. The best values are emphasized in bold. A linear covariance function (L.) was added to the respective covariance functions.

	L. + RBF	L. + Matérn($\frac{5}{2}$)	L. + Matérn($\frac{3}{2}$)	L. + Matérn($\frac{1}{2}$)
RMSE [$\times 10^{-5}$]	**4.492 ± 0.068**	4.498 ± 0.069	4.506 ± 0.064	4.548 ± 0.069
SMSE [$\times 10^{-3}$]	**4.550 ± 0.418**	4.561 ± 0.424	4.578 ± 0.410	4.665 ± 0.415
MSLL	**-2.696 ± 0.046**	-2.695 ± 0.046	-2.693 ± 0.045	-2.683 ± 0.044

Table 4.5: Results of the prediction and the simulation on the Bouc-Wen test data for the best 2 covariance functions selected by cross-validation. The best values are emphasized in bold. NP denotes noise propagation.

		Prediction	Simulation	Simulation with NP
L. + RBF	RMSE [$\times 10^{-5}$]	4.461	5.880	6.090
	SMSE [$\times 10^{-3}$]	4.462	7.753	8.318
	MSLL	-2.705	-2.367	-2.071
L. + Matérn($\frac{5}{2}$)	RMSE [$\times 10^{-5}$]	**4.458**	**5.863**	6.104
	SMSE [$\times 10^{-3}$]	**4.458**	**7.710**	8.356
	MSLL	**-2.706**	**-2.384**	-2.067



(a) Prediction on the Bouc-Wen test data.

(b) Simulation on the Bouc-Wen test data.

Figure 4.24: Graphical comparison of the prediction and the simulation on the Bouc-Wen test data. Only a subset of data is shown. The responses are shown for the combination of a linear and a Matérn($\frac{5}{2}$) covariance function.

Chapter 5

Modeling the Local Weather Dynamics

In this chapter, we will present a multi-input multi-output GP-NARX model to describe the local weather dynamics in the vicinity of the nuclear power plant (NPP) Krško. The size of the data and the observation of multiple outputs present a significant challenge to GP-NARX models, where an implementation of the vanilla GP-NARX or the FCMC correlated simulation of the sparse approximated GP-NARX models is not possible in practice. For this reason, the case study presents an interesting problem to validate the TCMC approximation for the simulation of GP-NARX models.

5.1 Introduction

Accidents at a nuclear power plant, which could cause the release of radionuclides into the atmosphere, are dangers that are handled with all seriousness by individual nuclear power plants. We must be prepared for action in advance in case of such events by researching the possible developments of the dispersion of pollution in the air.

Appropriate online operating regime models for existing nuclear plants prepare real-time forecasts about the possible development of an event for a day or two ahead [110]. Such a predictive model must spatially and temporally show correctly where the radionuclide cloud would disperse to in the next few hours for the available weather forecast. It should also predict the radionuclide concentrations within a three-dimensional space above the studied area.

The input data for the air-pollution dispersion models are the weather variables which describe the condition of the atmosphere during the passage of the radioactive pollution cloud. At present, we can use the numerical weather prediction (NWP) model to collect the forecasts of the weather variables, but these data are not accurate enough for complex terrain.

This motivates the investigation at hand, which is to make a few-hour forecast of the weather variables describing the conditions in the atmosphere surrounding a nuclear facility. The solution is the integration of the NWP model with the multi-input multi-output GP-NARX model.

5.2 Case Study

The example in our investigation deals with the dispersion of radioactive pollutants hypothetically emitted from the Krško NPP, located in the East of Slovenia. The modeling problem is of pronounced interest because of the surrounding terrain which is considered to be complex. It is surrounded by hills, valleys, a river, and has different land use, e.g. urban, fields, forests, water bodies, etc. The investigated area covers 25 square kilometers around the Krško NPP.

Data used in the investigation are composed of the measurements of the historic weather variables and the NWP model forecasts. The measurements of the weather variables are collected by automatic measurement stations at five different locations. One measurement station is located at Krško NPP and the other five distributed around in the settlements of Brežice, Cerklje, Krško, Libna, and at Cerklje airport. Listed automatic measuring stations take real-time measurements and are averaged over 30-minute intervals. Characteristics of the measurement equipment can be found in [110].

The available measured weather variables are: wind speed and direction, temperature, humidity, air pressure, global solar radiation. An additional signal, diffuse solar radiation, is derived from the NWP model variables using a soft sensor [111].

The same weather variables are also provided as the NWP forecasts for this region. The data set used in our investigation consists of data from the years 2015 through 2017. The data from the year 2015 and the first half of 2016 are used for training and the data from the year 2017 as the test data. Some data gaps in the training data are omitted, which does not influence the results considerably. This resulted in the training set size of $n = 8955$ samples, and the test set size of $n = 17519$ samples.

The training and test splits are reasonable for atmospheric sciences. It is especially important that the size of the test data set contains a large amount of data from different seasons with different weather patterns. The available measured weather variables and the NWP model forecasts are defined in Table 5.1, where the exogenous inputs are represented by $\mathbf{x}$, $\mathbf{y}$ denotes the outputs of interest, and $\mathbf{y}_a$ denotes all the augmented outputs with local measurements. The model is trained, predicted, and simulated on the augmented outputs. However, in the validation of the model, we only consider the weather variables of interest.

5.3 System for Modeling the Particle Dispersion

In critical scenarios, as in the case of advising the prevention actions of a potential nuclear disaster, estimation of the predictive uncertainty of the model is critical. Additionally, we want to use all available information, e.g. the measured weather variables at near geographical locations.

The weather variables describing the atmosphere dynamics (preprocessed and extended with additional variables) serve as an input to the Lagrangian particle dispersion model, which forecasts the air pollution dispersion into the near future. The high-level description for modeling the particle dispersion is presented in Figure 5.1. In this thesis we will focus on the GP-NARX model used for modeling the atmospheric variables. The details and description for other parts of the system can be found in [67].

Table 5.1: The data used for modeling the local weather dynamics. Exogenous inputs are represented by $\mathbf{x}$, $\mathbf{y}$ denotes the outputs of interest, and $\mathbf{y}_a$ denotes all the augmented outputs with local measurements. The Weather Research and Forecasting (WRF) model is the NWP model considered in this case study. The model is trained, predicted, and simulated on the augmented outputs. However, in the validation of the model, we only consider the weather variables of interest.

Abbreviation	Definition	$\mathbf{x}$	$\mathbf{y}$	$\mathbf{y}_a$
WRF_AH	WRF model absolute humidity prediction	×		
WRF_WX	WRF model wind component x prediction	×		
WRF_WY	WRF model wind component y prediction	×		
WRF_CC	WRF model cloud cover prediction	×		
WRF_AT	WRF model air temperature prediction	×		
WRF_GSR	WRF model global solar radiation prediction	×		
WRF_DSR	WRF model diffuse solar radiation prediction	×		
WRF_AP	WRF model air pressure prediction	×		
S_WX	Stolp wind component x measurement at 10m		×	×
S_WY	Stolp wind component y measurement at 10m		×	×
S_AT2	Stolp air temperature measurement at 2m		×	×
S_AT10	Stolp air temperature measurement at 10m		×	×
S_AT40	Stolp air temperature measurement at 40m		×	×
S_AT70	Stolp air temperature measurement at 70m		×	×
S_AH2	Stolp absolute humidity measurement at 2m		×	×
S_AP	Stolp air pressure measurement		×	×
B_AT	Brezice air temperature measurement			×
B_AH	Brezice absolute humidity measurement			×
C_AT	Cerklje air temperature measurement			×
C_AH	Cerklje absolute humidity measurement			×
C_AP	Cerklje air pressure measurement			×
CA_AT	Cerklje airport air temperature measurement			×
CA_AH	Cerklje airport absolute humidity measurement			×
CA_AP	Cerklje airport air pressure measurement			×
K_AT	Krsko air temperature measurement			×
K_AH	Krsko absolute humidity measurement			×
L_AT	Libna air temperature measurement			×
L_AH	Libna absolute humidity measurement			×
S_GSR	Stolp global solar radiation measurement			×

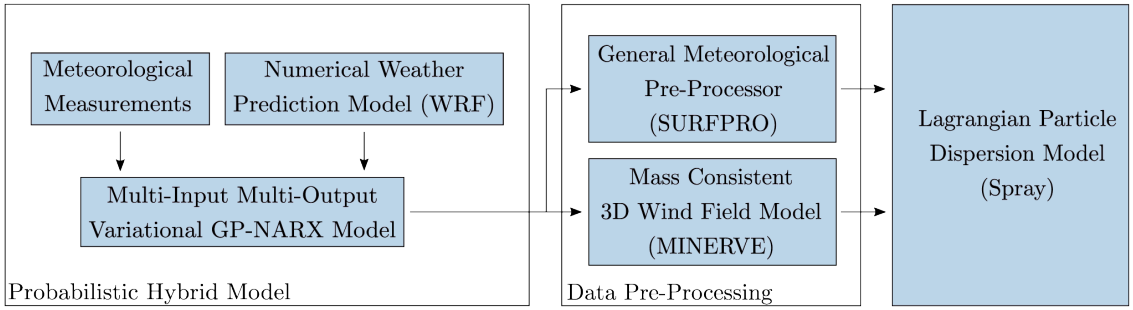


Figure 5.1: The model of the air pollution dispersion at a chosen geographical location. The existing approach uses the forecasts from the NWP model directly as the inputs to the data preprocessing software. The outputs from the data preprocessing software are later used in a Lagrangian particle dispersion model. In our proposed approach, the NWP forecasts are enhanced with a data-driven model resulting in a hybrid model. In such case, the data preprocessing software uses the forecasts from the hybrid model which are then provided to the Lagrangian particle dispersion model.

5.4 Probabilistic Hybrid Model

A probabilistic hybrid model is described in this section, which combines all the available data, i.e. the forecasts of the weather variables from the physical NWP model and the historic measured weather variables. This single integrated model aims to improve the weather variable forecast for the complex terrain of interest. Because of the demanding computational requirements, a variational approximation of the GP-NARX model was used. Namely we used a GP-NARX (VFE) model because it can be trained using the LBFGS optimizer.

The parameters of a NARX model presented by Equation (2.62) were identified by $n_a = 1$ and $n_b = 2$. The model capability was not improved significantly with an increasing number of lags. A second-order system was found as an optimal trade-off between the modeling capability and the computational complexity.

We assume that the outputs of the model are conditionally independent given the observed data, i.e

$$p\left(\mathbf{f}^1_{1:t}, \mathbf{f}^2_{1:t}, \ldots, \mathbf{f}^p_{1:t} | \mathbf{Y}_{1:t}, \mathbf{Z}_{1:t}\right) = p\left(\mathbf{f}^1_{1:t} | \mathbf{y}^1_{1:t}, \mathbf{Z}_{1:t}\right) p\left(\mathbf{f}^2_{1:t} | \mathbf{y}^2_{1:t}, \mathbf{Z}_{1:t}\right) \ldots p\left(\mathbf{f}^p_{1:t} | \mathbf{y}^p_{1:t}, \mathbf{Z}_{1:t}\right), \tag{5.1}$$

where the i-th row of the matrix $\mathbf{Z}_{1:t}$ is defined by $\mathbf{z}_i^T = \begin{bmatrix} \mathbf{y}_{i-1}^T & \mathbf{y}_{i-2}^T & \mathbf{x}_{i-1}^T \end{bmatrix}$. The blocks $\mathbf{K}^i_{f_{1:t}, f_{1:t}}$ are parametrized by the same covariance function $\mathrm{k}(\cdot, \cdot)$, i.e. the hyperparameters are shared. The noise variance is modeled separately for each output dimension, i.e. $\left(\sigma_n^2\right)^i$. Hyperparameters were obtained with the MLL maximization using the LFBGS optimizer. The number of pseudo-inputs was selected as 100, where the initial locations were initialized with the k-means clustering algorithm as suggested in [35].

The covariance function was identified by 10-fold cross-validation over the training data. Table 5.2 shows the proposed covariance functions used in this empirical process where the results are averaged over 10-fold cross-validation. Table 5.2 shows that the best covariance function according to the performance measures for the half-hour prediction was the Matérn($\frac{5}{2}$) kernel. Again, the smooth functions induced by the Matérn($\frac{5}{2}$) covariance function generalize better than functions induced by the rougher Matérn kernels.

Table 5.2: Means and the respective 95% confidence intervals of the RMSE, SMSE, and MSLL for the prediction of the atmospheric variables. The performance metrics are presented for 10-fold cross-validation over the training data. The best values are emphasized in bold.

		RBF	Matérn($\frac{5}{2}$)	Matérn($\frac{3}{2}$)	Matérn($\frac{1}{2}$)
S_WX	RMSE [$\times 10^{-1}$]	**6.018 ± 1.819**	6.041 ± 1.826	6.262 ± 1.855	7.837 ± 2.469
	SMSE [$\times 10^{-1}$]	**4.135 ± 2.005**	4.164 ± 2.002	4.448 ± 1.877	6.837 ± 1.718
	MSLL	**-0.428 ± 0.275**	-0.425 ± 0.274	-0.388 ± 0.242	-0.164 ± 0.133
S_WY	RMSE [$\times 10^{-1}$]	**6.470 ±1.263**	6.490 ± 1.254	6.757 ± 1.225	8.911 ± 1.433
	SMSE [$\times 10^{-1}$]	**1.665 ± 0.977**	1.674 ± 0.969	1.806 ± 0.961	3.166 ± 1.829
	MSLL	**-0.907 ± 0.284**	-0.904 ± 0.282	-0.865 ± 0.264	-0.583 ± 0.293
S_AT2	RMSE [$\times 10^{-1}$]	5.435 ± 1.492	**5.412 ± 1.475**	5.453 ± 1.440	7.044 ± 5.918
	SMSE [$\times 10^{-2}$]	1.412 ± 1.281	**1.400 ± 1.268**	1.417 ± 1.261	2.578 ± 5.088
	MSLL	-2.161 ± 0.470	**-2.165 ± 0.463**	-2.155 ± 0.434	-1.886 ± 0.696
S_AT10	RMSE [$\times 10^{-1}$]	4.724 ± 1.165	**4.694 ± 1.134**	4.695 ± 1.099	6.297 ± 6.387
	SMSE [$\times 10^{-2}$]	1.158 ± 1.023	1.141 ± 0.997	**1.136 ± 0.972**	2.421 ± 6.087
	MSLL	-2.264 ± 0.452	**-2.272 ± 0.439**	-2.267 ± 0.412	-1.954 ± 0.799
S_AT40	RMSE [$\times 10^{-1}$]	4.920 ± 1.253	**4.909 ± 1.237**	4.955 ± 1.243	6.642 ± 6.039
	SMSE [$\times 10^{-2}$]	1.303 ± 1.022	**1.297 ± 1.014**	1.322 ± 1.027	2.762 ± 6.181
	MSLL	-2.193 ± 0.389	**-2.197 ± 0.385**	-2.184 ± 0.376	-1.876 ± 0.771
S_AT70	RMSE [$\times 10^{-1}$]	4.892 ± 1.208	**4.881 ± 1.193**	4.930 ± 1.211	6.918 ± 6.957
	SMSE [$\times 10^{-2}$]	1.389 ± 1.233	**1.382 ± 1.215**	1.411 ± 1.243	3.391 ± 2.760
	MSLL	-2.174 ± 0.444	**-2.177 ± 0.438**	-2.165 ± 0.432	-1.802 ± 0.992
S_AH2	RMSE [$\times 10^{-1}$]	**2.187 ± 1.784**	2.187 ± 1.787	2.229 ± 1.904	3.589 ± 7.437
	SMSE [$\times 10^{-2}$]	2.656 ± 3.297	**2.655 ± 3.301**	2.722 ± 3.309	7.001 ± 20.72
	MSLL	-1.731 ± 0.725	**-1.737 ± 0.709**	-1.730 ± 0.671	-0.700 ± 5.170
S_AP	RMSE [$\times 10^{-1}$]	**2.218 ± 1.200**	2.227 ± 1.199	2.330 ± 1.223	4.826 ± 5.088
	SMSE [$\times 10^{-3}$]	**1.274 ± 1.082**	1.290 ± 1.118	1.420 ± 1.225	7.535 ± 18.56
	MSLL	**-3.293 ± 0.516**	-3.283 ± 0.517	-3.176 ± 0.541	-2.442 ± 0.760

5.5 Short-Term Weather Variables Forecast

In this section, we will consider one step-ahead prediction and multi-step ahead prediction of up to 12 hours into the future. Figure 5.2 shows the scatter plot between the measurements and the half-hour prediction of the weather variables.

The largest variability can be seen in the wind component predictions which proved to be a significant modeling challenge. The Cartesian wind components show a slight bias and the modeled noise does not seem to be independent and identically distributed as assumed with modeling with GPs. This is not an unexpected behavior since the wind predictions are complex and heavily depend on the surrounding terrain as well as other predictors that may not be accessible. Other weather variables on Figure 5.2 show excellent performance results and the noise is modeled appropriately.

Table 5.3 presents the results of the validation on the test set for three performance metrics and a comparison to the NWP forecast that was previously used in practice. We see that the GP-NARX (VFE) model performs better than the NWP forecast with respect to the RMSE. The downside of the RMSE metrics is that it can only take the predicted mean into consideration. Therefore, the MSLL is also included which is suitable for validation in

Table 5.3: RMSE, SMSE, and MSLL metrics for the half-hour prediction of the weather variables of interest. The GP-NARX model prediction is compared to the NWP forecast. Best results are shown in bold.

		S_WX	S_WY	S_AT2	S_AT10
GP-NARX	RMSE	$\mathbf{6.230 \times 10^{-1}}$	$\mathbf{6.489 \times 10^{-1}}$	$\mathbf{4.892 \times 10^{-1}}$	$\mathbf{3.920 \times 10^{-1}}$
	SMSE	$\mathbf{3.534 \times 10^{-1}}$	$\mathbf{1.519 \times 10^{-1}}$	$\mathbf{2.547 \times 10^{-3}}$	$\mathbf{1.714 \times 10^{-3}}$
	MSLL	-0.520	-0.943	-2.981	-3.155
NWP	RMSE	15.25×10^{-1}	15.87×10^{-1}	25.53×10^{-1}	26.69×10^{-1}
	SMSE	21.19×10^{-1}	9.094×10^{-1}	69.34×10^{-3}	79.44×10^{-3}

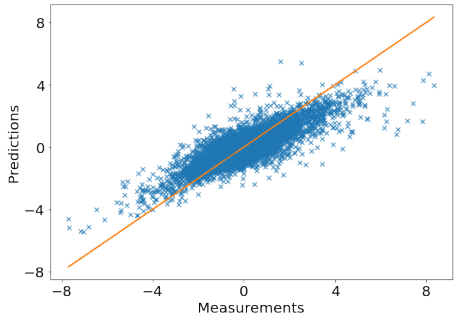
		S_AT40	S_AT70	S_AH2	S_AP
GP-NARX	RMSE	$\mathbf{4.692 \times 10^{-1}}$	$\mathbf{4.504 \times 10^{-1}}$	$\mathbf{2.591 \times 10^{-1}}$	$\mathbf{2.029 \times 10^{-1}}$
	SMSE	$\mathbf{2.475 \times 10^{-3}}$	$\mathbf{2.342 \times 10^{-3}}$	$\mathbf{4.643 \times 10^{-3}}$	$\mathbf{8.126 \times 10^{-4}}$
	MSLL	-3.004	-3.025	-2.693	-3.527
NWP	RMSE	30.31×10^{-1}	30.56×10^{-1}	14.71×10^{-1}	20.45×10^{-1}
	SMSE	103.2×10^{-3}	107.8×10^{-3}	149.6×10^{-3}	825.3×10^{-4}

the form of random variables. We can see that half-hour prediction performs well also in terms of predicted probabilities. The predictive response of the weather variables for the period between the first and the seventh of March 2017 is presented in Figure 5.4.

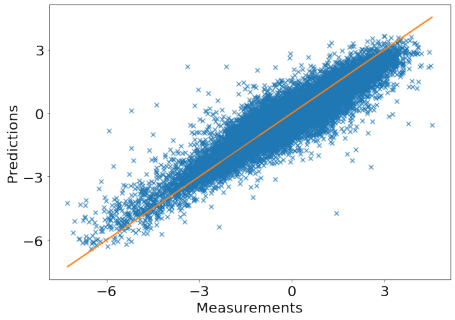
However, to advise the prevention actions in the case of a nuclear accident, we are interested in longer prediction intervals. Multiple step-ahead prediction of the weather variables was obtained with the TCMC simulation with $T_{\mathbb{V}} = 10^{-3}$ where a 100 independent MC samples of future trajectories were computed.

Figure 5.3 shows the performance metrics in relationship with increasing the prediction horizon into the future. A step corresponds to a half-hour increase in time. The prediction error metrics for the forecasts up to 12 hours into the future were considered, where the exogenous inputs in each step are defined by the most recent forecast from the NWP model.

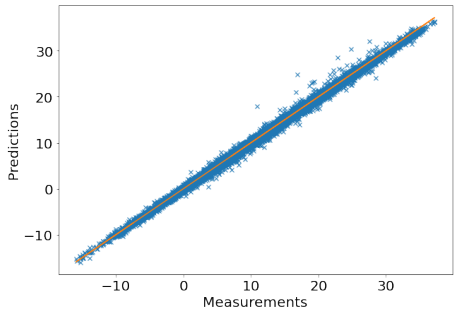
The model performance is worsened with the increased prediction horizon. Temperature, humidity, and air pressure are still relatively well modeled and the negative MSLL still shows that the variables are well described in terms of probabilities. In the case of the wind forecasts, the model performance degrades faster with increased prediction horizon.



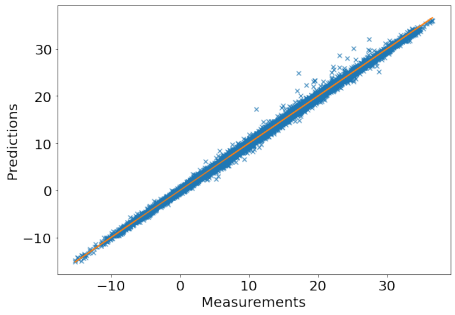
(a) S_WX $[m/s]$



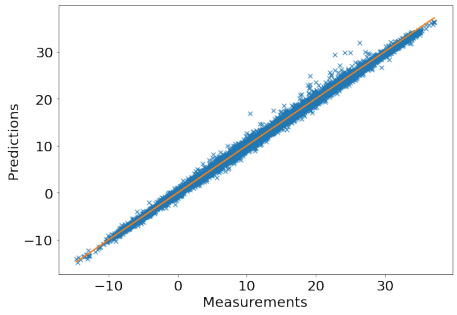
(b) S_WY $[m/s]$



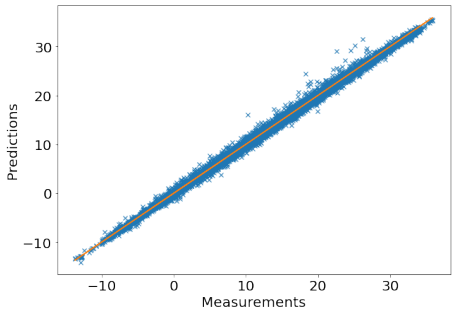
(c) S_AT2 $[^{\circ}C]$



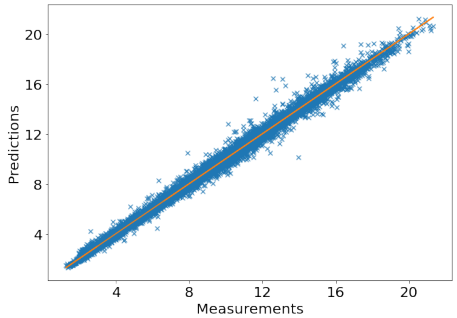
(d) S_AT10 $[^{\circ}C]$



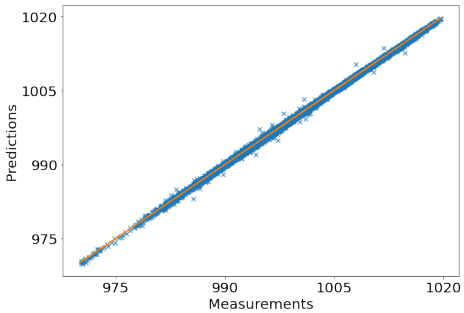
(e) S_AT40 $[^{\circ}C]$



(f) S_AT70 $[^{\circ}C]$

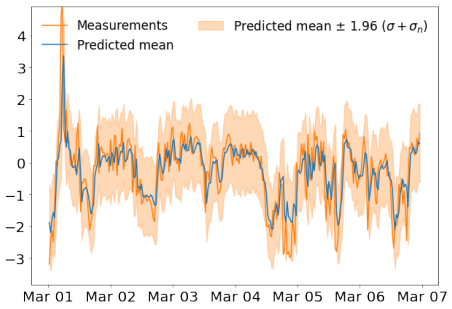


(g) S_AH2 $[g/m^3]$

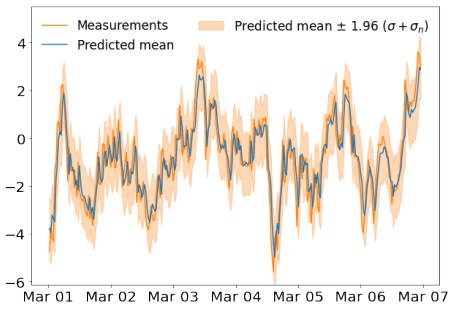


(h) S_AP $[mbar]$

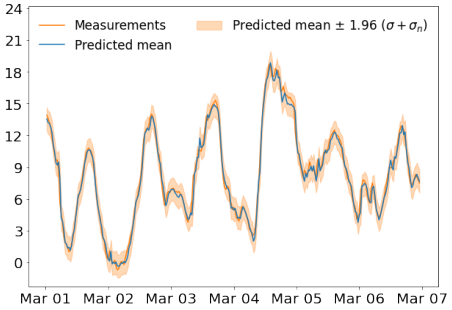
Figure 5.2: Scatter plot showing the dependency between the measured weather variables of interest and their corresponding half-hour prediction presented in blue. The orange line shows the observed weather variables.



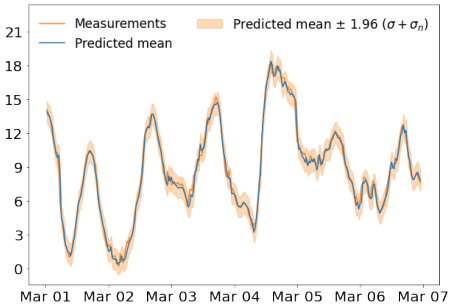
(a) `S_WX` $[m/s]$



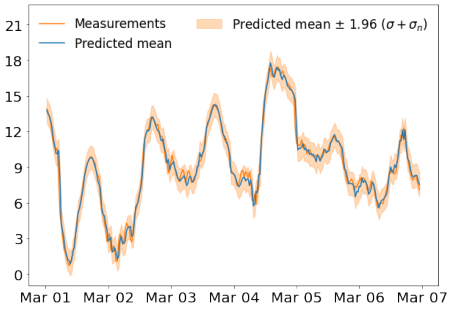
(b) `S_WY` $[m/s]$



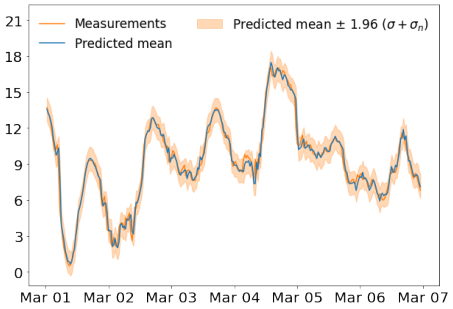
(c) `S_AT2` $[^\circ C]$



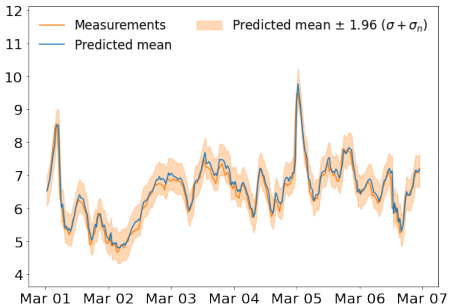
(d) `S_AT10` $[^\circ C]$



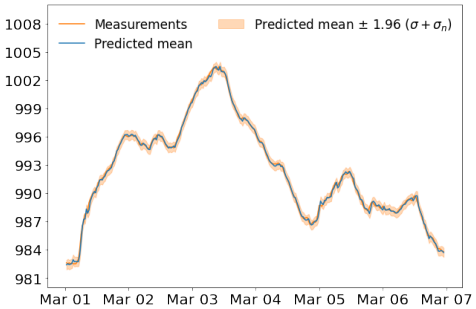
(e) `S_AT40` $[^\circ C]$



(f) `S_AT70` $[^\circ C]$

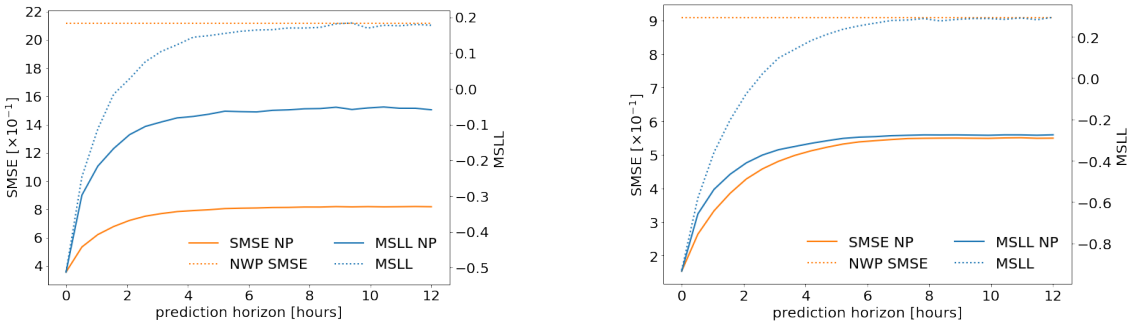


(g) `S_AH2` $[g/m^3]$

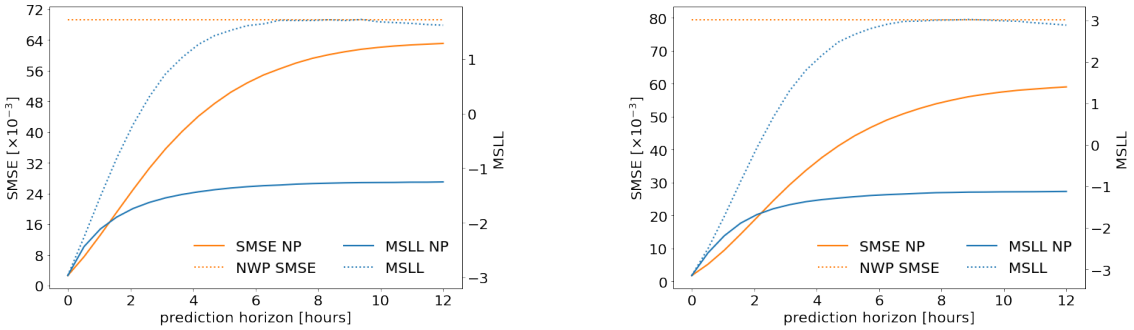


(h) `S_AP` $[mbar]$

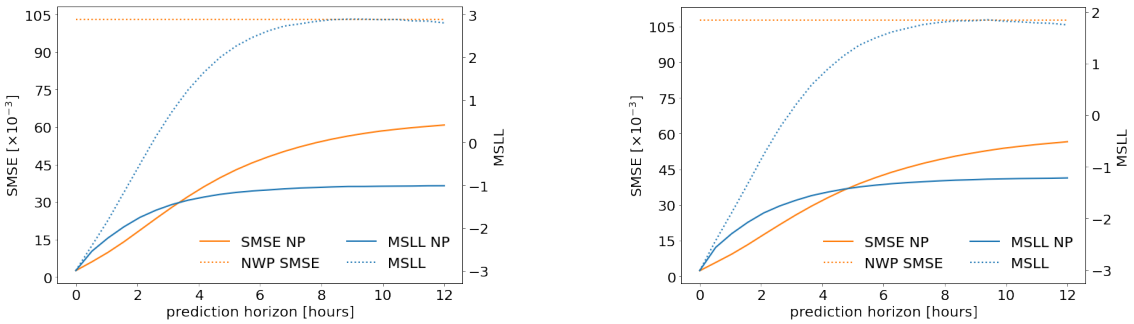
Figure 5.3: Half-hour prediction and the corresponding 95% confidence interval for the weather variables of interest near NPP Krško. The results are shown for the period between the first and the seventh of March 2017.



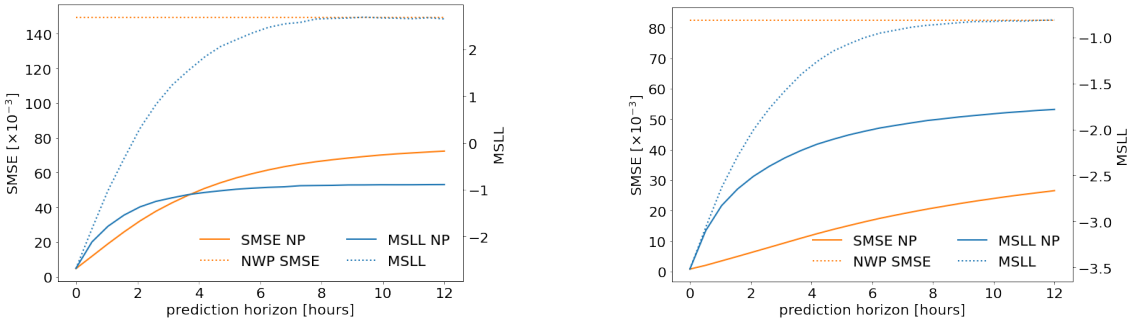
(a) S_WX $[m/s]$ (b) S_WY $[m/s]$



(c) S_AT2 $[^{\circ}C]$ (d) S_AT10 $[^{\circ}C]$



(e) S_AT40 $[^{\circ}C]$ (f) S_AT70 $[^{\circ}C]$



(g) S_AH2 $[g/m^3]$ (h) S_AP $[mbar]$

Figure 5.4: RMSE and MSLL for the 12-hour prediction of weather variables near NPP Krško. Each step corresponds to a half-hour interval. NP denotes noise propagation.

Table 5.4: RMSE, SMSE, and MSLL metrics for the simulation of the weather variables of interest. Simulation without NP is denoted by †, and simulation with NP is denoted by ‡. The response was estimated using 500 MC samples. The GP-NARX model simulation is compared to the NWP forecast that can also be seen as a simulation model. The best results are shown in bold.

		S_WX	S_WY	S_AT2	S_AT10
GP-NARX†	RMSE	$\mathbf{9.445 \times 10^{-1}}$	$\mathbf{12.29 \times 10^{-1}}$	$\mathbf{24.24 \times 10^{-1}}$	$\mathbf{23.04 \times 10^{-1}}$
	SMSE	$\mathbf{8.121 \times 10^{-1}}$	$\mathbf{5.451 \times 10^{-1}}$	$\mathbf{62.57 \times 10^{-3}}$	$\mathbf{59.24 \times 10^{-3}}$
	MSLL	0.141	0.254	1.179	2.100
GP-NARX‡	RMSE	9.459×10^{-1}	$\mathbf{12.29 \times 10^{-1}}$	24.52×10^{-1}	23.38×10^{-1}
	SMSE	8.146×10^{-1}	5.452×10^{-1}	64.02×10^{-3}	60.97×10^{-3}
	MSLL	**-0.068**	**-0.285**	**-1.274**	**-1.147**
NWP	RMSE	15.25×10^{-1}	15.87×10^{-1}	25.53×10^{-1}	26.69×10^{-1}
	SMSE	21.19×10^{-1}	9.094×10^{-1}	69.34×10^{-3}	79.44×10^{-3}

		S_AT40	S_AT70	S_AH2	S_AP
GP-NARX†	RMSE	$\mathbf{23.69 \times 10^{-1}}$	$\mathbf{22.93 \times 10^{-1}}$	11.03×10^{-1}	$\mathbf{13.61 \times 10^{-1}}$
	SMSE	$\mathbf{63.09 \times 10^{-3}}$	$\mathbf{60.73 \times 10^{-3}}$	84.23×10^{-3}	$\mathbf{365.5 \times 10^{-4}}$
	MSLL	1.846	0.930	2.064	-1.267
GP-NARX‡	RMSE	24.17×10^{-1}	23.40×10^{-1}	$\mathbf{11.00 \times 10^{-1}}$	13.75×10^{-1}
	SMSE	65.68×10^{-3}	63.21×10^{-3}	$\mathbf{83.70 \times 10^{-3}}$	373.0×10^{-4}
	MSLL	**-1.019**	**-1.189**	**-0.735**	**-1.653**
NWP	RMSE	30.31×10^{-1}	30.56×10^{-1}	14.71×10^{-1}	20.45×10^{-1}
	SMSE	103.2×10^{-3}	107.8×10^{-3}	149.6×10^{-3}	825.3×10^{-4}

5.6 Long-Term Weather Variables Forecast

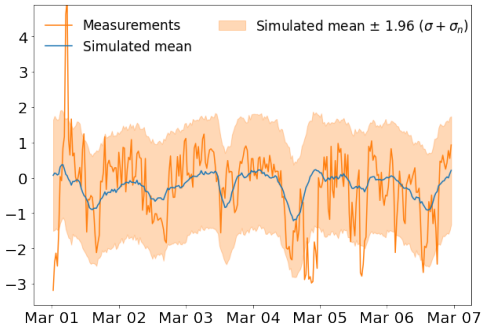
The limit case where the prediction horizon is infinite was also considered. In the simulation, the known parameters and measurements are used to initialize the model. In our case, the model was initialized with the start of the test data. Then the simulation is obtained until the end of the test data.

We want to emphasize that the measured weather variables are unknown in the future and are not used when validating the test data set for multi-step-ahead prediction or simulation. The unknown inputs at an arbitrary time step into the future are replaced with the samples from the previous step. The exogenous inputs in each step are defined by the most recent forecasts from the NWP model, meaning that we assume that the NWP model short-term forecasts are known. In practice, this forecast would have to be replaced with a long-term NWP model forecast if available. The simulation presented in this section is therefore limited with the accuracy of such forecast.

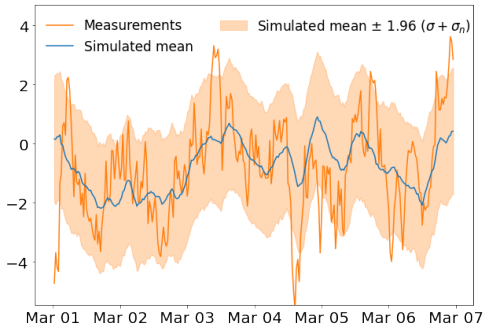
Table 5.4 shows the validation metrics for the simulation on the test data set. We observe that the simulation responses of the weather variables for a GP-NARX (VFE) model are better modeled than the forecasts from the NWP model, which can also be seen as a simulation model. Temperature, humidity, and air pressure are described well even in the extreme case of simulation. Their respective RMSE is small and their corresponding MSLL is negative.

As expected, the worst performance is obtained for wind simulations, although the simulated wind components are still better described in the RMSE sense than the NWP forecasts. Figure 5.5 shows the simulated response of the weather variables and the associated uncertainty for simulating the GP-NARX (VFE) model. As previously shown with the validation metrics, the temperature, humidity, and air pressure are described well with the model. Even the wind simulations capture the dynamics to some degree, although the variance is higher but well-calibrated. The predicted uncertainty interval of 95% captures most of the predicted variability of the measured signals.

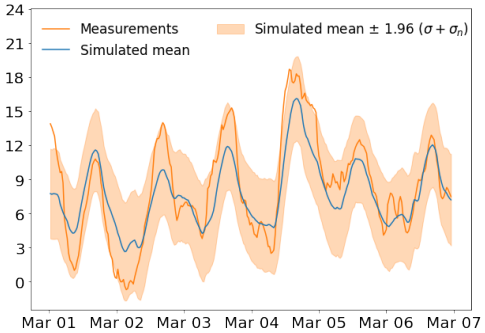
The covariance function in this chapter was selected in a black-box fashion. Although some knowledge is known prior to observing the data, e.g. periodicity, we did not consider adding a periodic component in the design of the covariance function. This is due to the hybrid structure, where the model's forecast based on first principles is already considered in the inputs and contains most of that information. In the next chapter, we consider a case study where the prior knowledge is incorporated into the design of the covariance function and the likelihood model, which consequently requires flexible simulation algorithms.



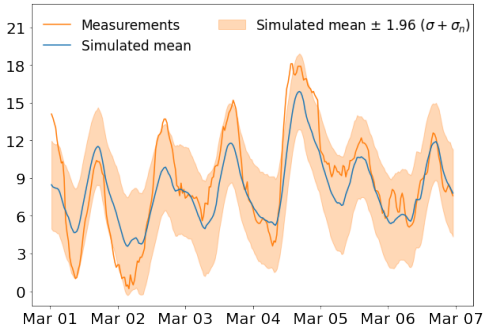
(a) `S_WX` $[m/s]$



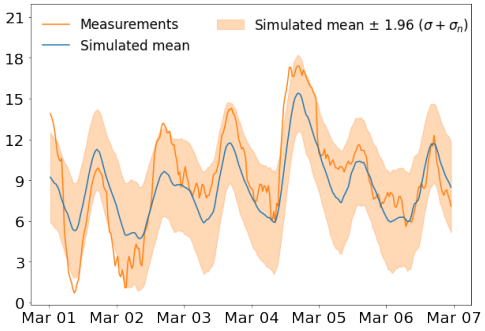
(b) `S_WY` $[m/s]$



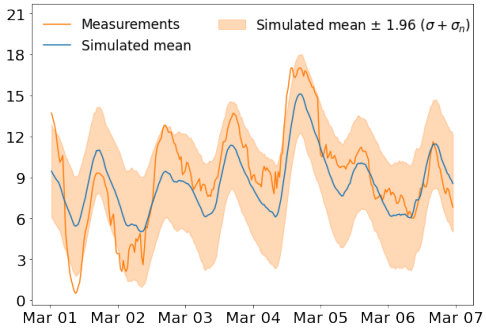
(c) `S_AT2` $[^{\circ}C]$



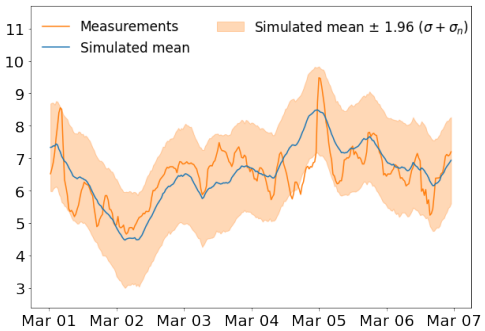
(d) `S_AT10` $[^{\circ}C]$



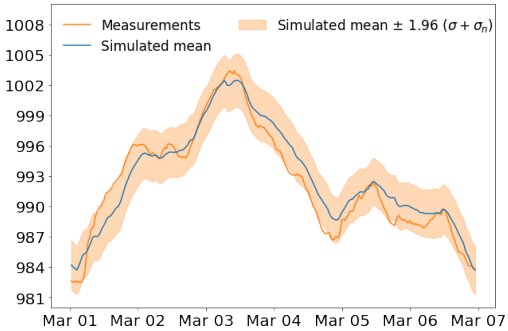
(e) `S_AT40` $[^{\circ}C]$



(f) `S_AT70` $[^{\circ}C]$



(g) `S_AH2` $[g/m^3]$



(h) `S_AP` $[mbar]$

Figure 5.5: Simulated response and the corresponding 95% confidence interval for the weather variables of interest near NPP Krško. The simulation was initialized on the first of January 2017 and the weather variables continuously simulated where the most recent NWP forecasts were used as exogenous inputs at each iteration. The results are shown for the period between the first and the seventh of March 2017.

Chapter 6

Load and Photovoltaic Generation Forecasting

In the previous chapter, we presented a case study where a GP model considered could deal with large amounts of data in training, as well as at test time. A rigorous exploration of the structure in the data through kernel design was not performed, mostly because this structure is already modeled by the physical NWP forecast that is considered in the inputs. In this chapter, we present a case study of load and photovoltaic generation forecasting in the greater area of Sydney. Since such physical forecasts for the outputs considered do not exist in practice, we take a more grey-box approach to model from data.

Autoregressive models allow us to prototype the models quickly in the presence of more complex patterns in the data, e.g. periodicity, heteroskedastic noise, and localized change in process behavior, that are considered with the grey-box approach. The case study justifies the need for a flexible MC approach to simulation, unconstrained from the choice of the covariance function, the assumption of homoskedastic Gaussian noise, or the choice of the GP model approximation.

6.1 Introduction

In power engineering, the optimal power flow (OPF) is an optimization problem that was first introduced in [112]. The objective of OPF is to find a steady state operating point that minimizes the cost of electric power generation while satisfying operating constraints and meeting demand [113]. The constraints are specified by Kirchhoff's Law, line flow constraints, and constraints on voltage and power generation. The constraints are nonlinear so the OPF problem is nonconvex and there is generally no guarantee to find a global solution. However, the optimization problem is well studied in practice.

With the increasing presence of distributed energy resources (DER), the management of electricity generation and consumption is becoming more complex. For example, rooftop solar photovoltaic (PV) systems and home battery storage systems can reduce cost for consumers, but they can also create network problems. Additionally, the load patterns are changing with the introduction of electric vehicles and flexible loads. The distribution network operators (DSO) are not well equipped for the rapid change in consumer behavior [19].

In New South Wales, the DSO companies do not have access to real-time smart meter data. Rather, they can buy them from retailers for the previous day. This makes the management of electricity generation complicated. For that reason, the DSO companies conservatively limit the consumer PV generation with blanket rooftop solar export limits, which results in suboptimal solutions.

In [114]–[116], they proposed dynamic operating limits to limit the PV generation of consumers but assumed the input data given. This is not true in practice and we have to use the forecasts of the respective predictors. If forecasted, the input data are not considered random, although we can never predict the future with absolute certainty. In [19], they proposed a chance-constrained OPF problem that handles the input data probabilistically. The optimization problem requires a probabilistic forecast of the load and PV generation up to 24 hours into the future. Conservative uncertainty intervals of the forecast will result in suboptimal solutions, whereas overconfident uncertainty intervals of the forecasts can result in ill-defined dynamic operating limits which can consequently cause the violation of network limits. For that reason, a model for the load and PV generation that can quantify uncertainty is of great interest.

In [19], they proposed quantile regression based on the generative adversarial network neural network architecture that can simulate statistically representative samples of the load and PV generation in the future. However, this approach does not systematically model the uncertainty in a fully Bayesian approach. GP processes do model the uncertainty systematically and are for that reason a state-of-the-art nonparametric approach to quantifying uncertainty. GPs also allow us to incorporate prior knowledge in the modeling process, which is beneficial for problems such as load and PV generation forecasting since they exhibit structural patterns that can be somewhat determined even before the data are observed.

Previously a GP approach was already applied to load forecasting [117]. However, the model regresses from the time domain to the output domain which was previously shown that it can not handle nonlinear dynamics [48]. The presented model was also validated only for the mean of the forecast and not probabilistically. An approach to load forecasting was also presented in [118] where only the prediction was considered. GPs were also applied for PV generation forecasting, but only for the forecasting interval of 30 minutes or less [119], [120].

The goal of this case study is to forecast the load and PV generation up to 24 hours into the future. The analysis of the delay for the available observations is considered to justify the installation of smart meters and the need for real-time observations from the perspective of the DSO.

Table 6.1: The data used for modeling the load and PV generation in the Greater Sydney Area. Exogenous inputs are represented by $\mathbf{x}_{\text{Load}}$ for modeling the load and $\mathbf{x}_{\text{PV}}$ for modeling the PV generation. The time index is represented by t. Change points are denoted by $\mathbf{c}$ and are derived from the sunset and sundown times. The outputs of interest are denoted by $\mathbf{y}$.

Definition	t	$\mathbf{x}^{\text{Load}}$	$\mathbf{x}^{\text{PV}}$	$\mathbf{c}$	$\mathbf{y}$
Time index [Each step corresponding to a 30min interval]	×				
Relative humidity [%] at the Sydney station		×	×		
Air Temperature [°C] at the Sydney station		×	×		
Station level pressure [hPa] at the Sydney station		×	×		
Derived cloud cover from the New South Wales stations			×		
Calculated global solar radiation for Sydney			×		
Change points derived from sunrise and sunset time for Sydney				×	
Total load					×
Total PV generation					×

6.2 Case Study

The case study considers the load and the PV generation in the Greater Sydney Area. The data were aggregated over all households and averaged over a 30-minute time interval. The cloud cover observations at the automatic ceilometer stations were aggregated over all levels and all weather stations in New South Wales since ceilometer observations specific to Sydney were not available. Observations of weather data were used from the Sydney station for the relative humidity, temperature, and pressure.

The global solar radiation for Sydney was calculated from first principles using the American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE) model [121]. Additionally, we augmented the data with information about the sunset and sundown times for New South Wales. Since the ceilometer observations specific to Sydney were not available, a more general predictor for the cloud cover was derived from the weather stations available in New South Wales. A weighted sum over the aggregated ceilometer data was calculated by

$$\begin{aligned} \text{Daily cloud cover} &= \sum_{h=1}^{48} w(h) \times \text{Aggregated ceilometer observation}(h), \\ w(h) &= \frac{\text{Global solar radiation}(h)}{\sum_{i=1}^{48} \text{Global solar radiation}(i)}, \end{aligned} \tag{6.1}$$

where the aggregated ceilometer observations are weighted by the daily global solar radiation. The hours in the day when global radiation is at its peak are most heavily influenced by the cloud cover when PV generation is considered. Figure 6.2 shows the PV generation example profile and the derived cloud cover predictor for two weeks in different seasons. We can see that the derived predictor is inversely correlated with the PV generation, which is what we would expect. The less cloud cover, the more PV generation is expected.

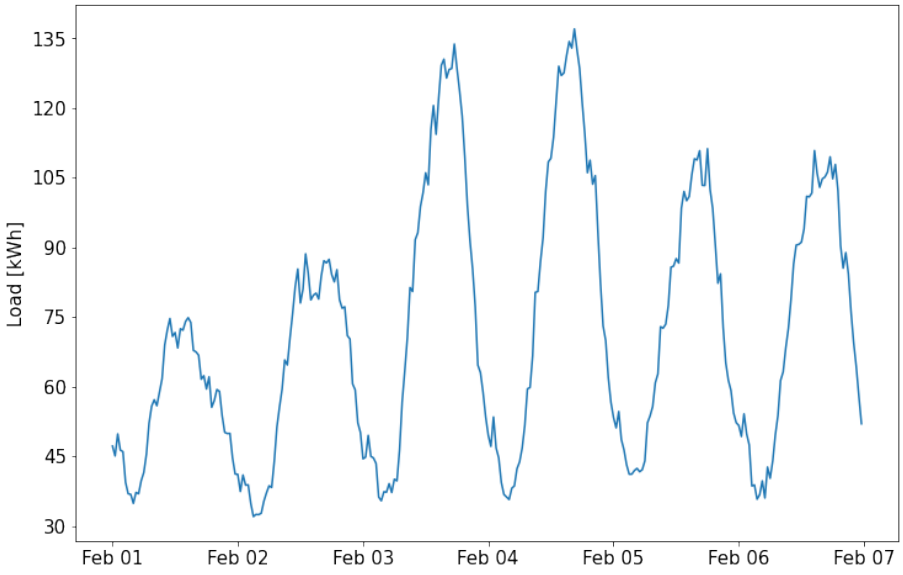
A description of the data used is presented in Table 6.1, where $\mathbf{x}_W$ denotes the weather components, $\mathbf{x}_{\text{Load}}$ and $\mathbf{x}_{\text{PV}}$ the exogenous data used in the respective load or PV generation model, and $\mathbf{c}$ denotes the change points that are used in the PV generation forecast to separate the daily and nightly behaviour.

The data were available only for the year 2019. To validate the model in different seasons and different days of the week, we selected the validation data to be the 6th, 7th, 13th, 14th, 27th, and 28th day of the month. Similarly, the test data was selected to be the 4th, 5th, 11th, 12th, 25th, and 26th day of the month. The training data was selected as the whole yearly data, excluding the days for the validation and the test data set. We assume that the weather data between the consecutive days is not heavily correlated, which is generally not true. Consequently, the estimation of the performance metrics can be slightly biased. However, many of the training/validation/test split strategies exhibit specific downsides (extrapolation, data not representing all seasons, days of the week, etc.) when only one year of data are available. If more data are available, the strategy presented in this case study should be reconsidered.

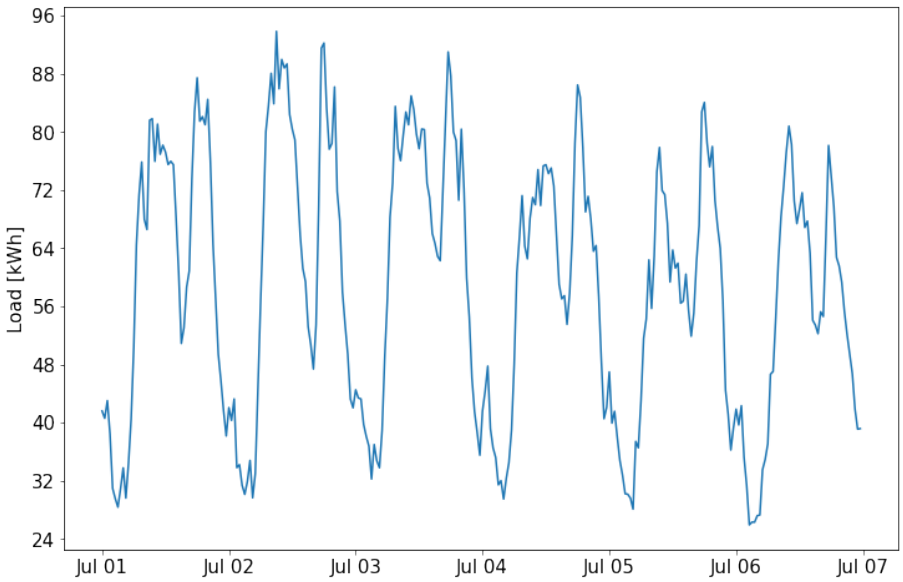
6.3 Probabilistic Model of Load and PV Generation

In this section, we will consider a composite model for load and PV generation forecasting. The models consist of a GP-NARX model combined with a timeseries model (a mapping from the time domain to the output domain) to incorporate the periodic structure of the data. Figure 6.1 and Figure 6.2 show the aggregated load and PV generation for different weeks in Australian summer and winter. We can see that both load and PV generation profiles exhibit daily (and seasonal) periodic behavior. Although the assumption of constant noise might be a good enough approximation for the load profiles, the PV generation profiles exhibit a heteroskedastic behavior where the random fluctuations seem to be larger with increased PV generation. Another structural property in PV generation is the bimodal process behaviour, i.e. the process can be split to the night and day part.

In the next two sections we will demonstrate how to model the structure present in the load and PV generation data. The properties of the problem will be modeled through the design of the covariance function and the likelihood model. The multi-step-ahead forecasts will be estimated using the TCMC simulation.

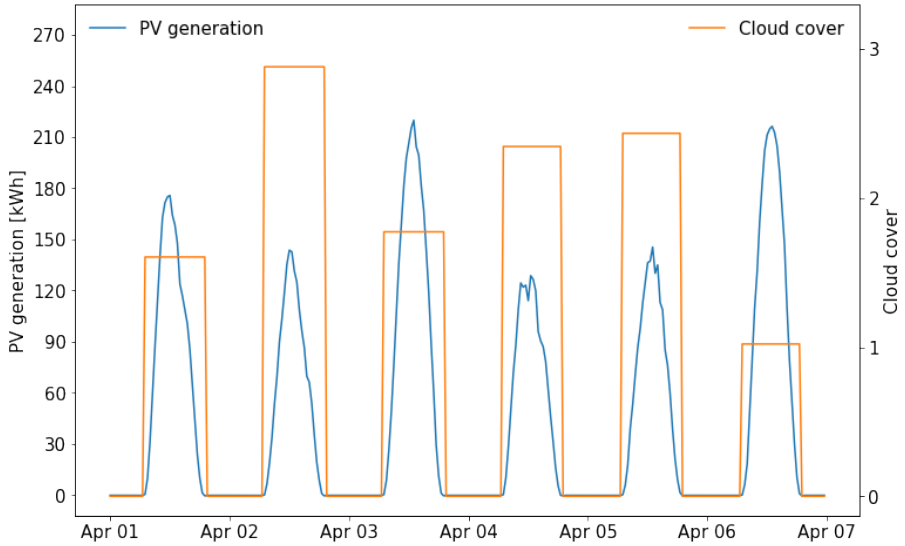


(a) Example of a load profile for the week period in February, 2019.

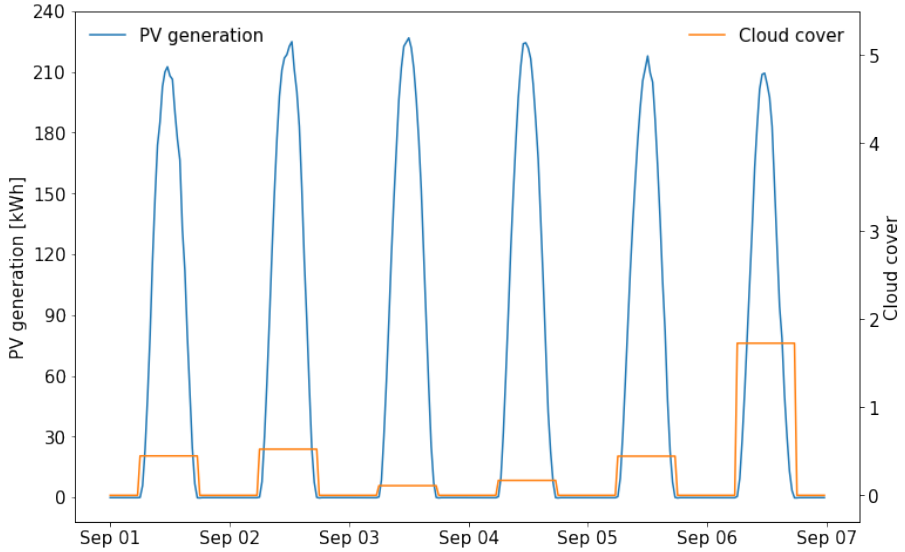


(b) Example of a load profile for the week period in July, 2019.

Figure 6.1: Example of two load profiles for the Greater Sydney Area in different seasons. The data are displayed for weeks in February and July, 2019.



(a) Example of PV generation for the week in April, 2019.



(b) Example of PV generation for the week in September, 2019.

Figure 6.2: Example of two PV generation profiles for the Greater Sydney Area in different seasons with the derived cloud cover predictor. The data are displayed for the weeks in April and September, 2019.

6.3.1 Load model

As we have seen in the Figure 6.1, the load exhibits periodic behavior. This can be modeled by a periodic kernel [122], defined in the Appendix C.5. Let γ define the period of latent function values modeling the electrical load. Given the 30 minute observations, we considered the periods $\gamma_1 = 48$ and $\gamma_2 = 7 \times 48$.

With the amount of data and the Gaussian likelihood assumption, the choice for the autoregressive part of the composite model is the GP-NARX (VFE). The model addresses the large data set with pseudo-point approximation, where the distance between the approximated posterior and the true posterior is rigorously minimized. The Gaussian likelihood assumption allows for a closed-form maximization of the ELBO with respect to the pseudo-point distribution. The ELBO is minimized with the LBFGS optimizer using 700 pseudo-inputs initialized by subsampling the original data set.

Table 6.2 shows the results of the covariance function selection on the validation data set. Vectors $\mathbf{z}$ denote the inputs defined by a NARX model, i.e.

$$\mathbf{z}_i = [y^{\text{Load}}_{i-n_a}, \ldots, y^{\text{Load}}_{i-1}, \left(\mathbf{x}^{\text{Load}}_{i-n_b}\right)^T, \ldots, \left(\mathbf{x}^{\text{Load}}_{i-1}\right)^T]^T, \tag{6.2}$$

where y^{Load} denotes the electrical load observations. Vectors $\mathbf{z}^\dagger$ denote the NARX inputs augmented with the 1-day, 2-day, and 3-day lag of the output, i.e.

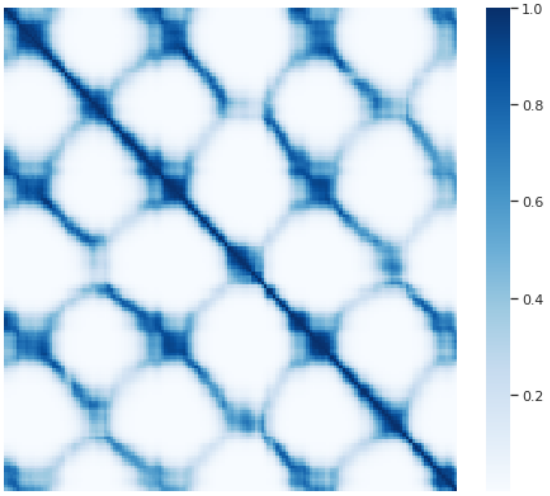
$$\mathbf{z}^\dagger_i = [y^{\text{Load}}_{i-144}, y^{\text{Load}}_{i-96}, y^{\text{Load}}_{i-48}, \mathbf{z}_i^T]^T. \tag{6.3}$$

The lag parameters of the NARX model were selected as $(n_a, n_b) = (2, 1)$. The best kernel was found to be a multiplication of the Matérn($\frac{5}{2}$) kernel (applied on the time index only) with an RBF kernel (applied on the NARX inputs $\mathbf{z}^\dagger_i$) with both the daily and the weekly periodic component. Since we only had the data for one year, we did not consider adding a yearly periodicity.

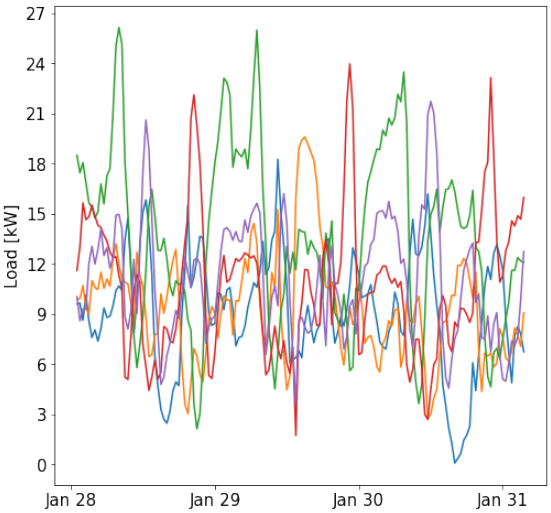
The prior structure of the process can be seen on Figure 6.3. The figure shows the covariance matrix before training on the observed data, after training on the observed data, and the samples of the corresponding latent functions. We can see that after maximization of the ELBO, the structure in the data is heavily emphasized. Learning the hyperparameters from data is important in GPs even if the collection of data is large. In Bayesian regression, the likelihood would eventually outweigh the prior given large amounts of data. However, considering the high-dimensional Gaussian distribution over the observed data, the convergence to a good solution, with respect to increasing the number of observed data, can be rather slow. MLL (ELBO) maximization provides a more data-efficient approach than just relying on the fact that the likelihood will eventually outweigh any reasonably defined prior.

Table 6.2: Results of the validation for modeling the load in the greater area of Sydney. The best values are emphasized in bold.

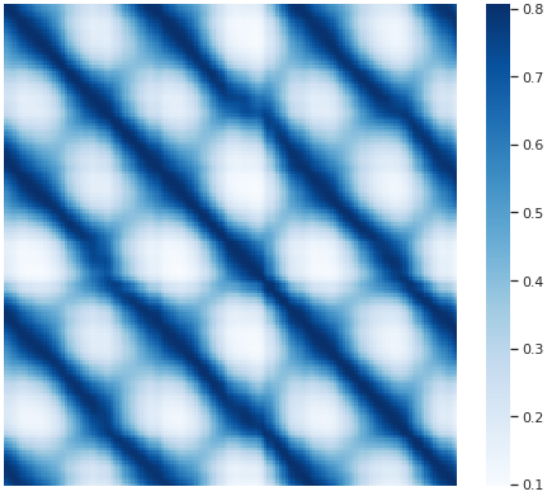
$k(\cdot,\cdot)$	RMSE	SMSE	MSLL
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i,\mathbf{z}_j)$	552.67	0.024	-1.872
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i,\mathbf{z}_j)$	551.44	0.024	-1.874
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i,\mathbf{z}_j)$	598.33	0.028	-1.793
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) + k_{\text{RBF}}(\mathbf{z}_i,\mathbf{z}_j)$	598.84	0.028	-1.792
$k_{\text{PER, RBF}}(t_i,t_j;\gamma_1) + k_{\text{RBF}}(\mathbf{z}_i,\mathbf{z}_j)$	599.40	0.028	-1.791
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_2) \times k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i,\mathbf{z}_j)$	549.34	0.023	-1.878
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_2) + k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i,\mathbf{z}_j)$	598.33	0.028	-1.793
$k_{\text{PER, RBF}}(t_i,t_j;\gamma_2) \times k_{\text{PER, RBF}}(t_i,t_j;\gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i,\mathbf{z}_j)$	548.60	0.023	-1.879
$k_{\text{PER, RBF}}(t_i,t_j;\gamma_2) + k_{\text{PER, RBF}}(t_i,t_j;\gamma_1) + k_{\text{RBF}}(\mathbf{z}_i,\mathbf{z}_j)$	599.40	0.028	-1.791
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	554.17	0.024	-1.870
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	552.60	0.024	-1.873
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	598.57	0.028	-1.793
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) + k_{\text{RBF}}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	599.79	0.028	-1.791
$k_{\text{PER, RBF}}(t_i,t_j;\gamma_1) + k_{\text{RBF}}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	599.03	0.028	-1.792
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_2) \times k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	551.14	0.023	-1.875
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_2) + k_{\text{PER, Matérn}(\frac{5}{2})}(t_i,t_j;\gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	598.50	0.028	-1.794
$k_{\text{PER, RBF}}(t_i,t_j;\gamma_2) \times k_{\text{PER, RBF}}(t_i,t_j;\gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	**547.09**	**0.023**	**-1.883**
$k_{\text{PER, RBF}}(t_i,t_j;\gamma_2) + k_{\text{PER, RBF}}(t_i,t_j;\gamma_1) + k_{\text{RBF}}(\mathbf{z}_i^\dagger,\mathbf{z}_j^\dagger)$	598.94	0.028	-1.793



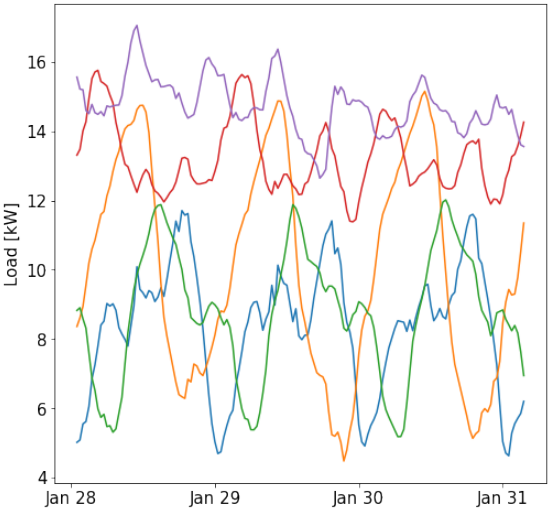
(a) Covariance matrix of the prior for the load model before training.



(b) 5 function samples from the prior for the load model before training.



(c) Covariance matrix of the prior for the load model after training.



(d) 5 function samples from the prior for the load model after training.

Figure 6.3: The covariance matrix of the prior for the load model and the corresponding function samples. The data are displayed for 3 consecutive days.

6.3.2 PV generation model

As observed in Figure 6.2, we will address the three structural properties of PV generation profiles:

- Periodicity;
- Switched behavior between the night and the day;
- Heteroskedastic noise.

Although the first two can still be addressed through covariance function design, the last one cannot. The elegant closed-form solution of the GP framework is broken in this case. For that reason, the GP-NARX (SVGP) model will be used for the autoregressive part of modeling the PV generation, which can handle likelihoods that break the analytic solutions considered with the VFE approximation.

Modeling the periodical behavior was already considered with the load model. Here, we again consider the daily periodicity where $\gamma_1 = 48$ for the half-hour time interval. We do not consider a weekly periodicity. The switched behavior between the day and night is modeled by a change point kernel [122], which is defined in Appendix C.6. The kernel can switch between two kernels similar to fuzzy logic. The membership is modeled by a sigmoid function, or a combination of multiple ones when generalizing for many change points. The change points are derived from the sunrise and sunset times for Sydney. Although the data could also be separated by day and night before the modeling process, the change point kernel provides a convenient way to model the transition between the two modes of the PV generation profiles.

The heteroskedastic likelihood is modeled by a separate GP [123]. The PV generation model is specified by

$$\begin{aligned} \mathbf{f}_{1:t} &\sim \mathcal{GP}\left(\mathbf{f}_{1:t}|\mathbf{0}, \mathbf{K}_{f_{1:t},f_{1:t}}\right), \\ \mathbf{g}_{1:t} &\sim \mathcal{GP}\left(\mathbf{g}_{1:t}|\mathbf{0}, \mathbf{K}_{g_{1:t},g_{1:t}}\right), \end{aligned} \tag{6.4}$$

where $\mathbf{K}_{f_{1:t},1:t}$ is parametrized by a change point kernel $k_{\text{CP},\, k_{\text{Constant}},\, k_{\text{Process}}}$ and $\mathbf{K}_{g_{1:t},g_{1:t}}$ is parametrized by $k_{\text{Likelihood}}$. The likelihood is then defined by

$$\begin{aligned} p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}, \mathbf{g}_{1:t}) &= \prod_{i=1}^{t} p(y_i|f_i, g_i), \\ p(y_i|f_i, g_i) &\sim \mathcal{N}\left(y_i|f_i, h(g_i)^2\right), \end{aligned} \tag{6.5}$$

where $h(\cdot)$ is a transformation to positive-only values since this parameter represents the standard deviation of the Gaussian likelihood. The transformation can be for example an exponential or a softplus function. This model will be from here on referred to as the Heteroskedastic GP-NARX (HGP-NARX). The ELBO of the HGP-NARX model is defined by

$$F(\mathbf{m}, \mathbf{\Phi}, \mathbf{z}'_{1:m}) = \sum_{i=1}^{t} \mathbb{E}_{q(f_i,g_i)}\left[\log p(y_i|f_i, g_i)\right] - \text{KL}\left[q(\mathbf{u}_f)||p(\mathbf{u}_f)\right] - \text{KL}\left[q(\mathbf{u}_g)||p(\mathbf{u}_g)\right], \tag{6.6}$$

where $\mathbf{u}_f$ represents the pseudo-points for the $\mathbf{f}$ process, and $\mathbf{u}_g$ the pseudo-points for the $\mathbf{g}$ process. The KL divergence can be computed in closed-form, but the expectation over the logarithm cannot. The integral is computed with 2D Gauss–Hermite quadrature [99].

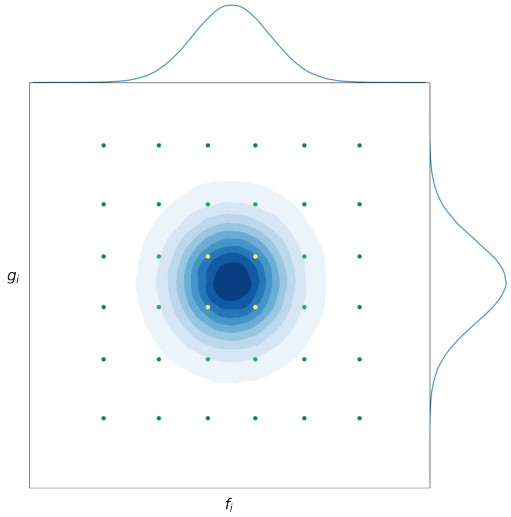


Figure 6.4: 2D Gauss–Hermite quadrature points used to compute the intractable Gaussian expectation over a function. In the case of the HGP-NARX model, the function is defined by $\alpha(y_i, f_i, g_i) = \log p(y_i|f_i, g_i)$.

Gauss–Hermite quadrature approximates the expectation of a function α with respect to a Gaussian distribution.

$$\begin{aligned}
\mathbb{E}_{q(f_i,g_i)}\left[\alpha(f_i, g_i)\right] &\approx \frac{1}{\pi}\sum_{j=1}^{n} w_j \beta(f_{ij}, g_{ij}), \\
\beta(f_{ij}, g_{ij}) &= \alpha\left(f_{ij}\sqrt{2}\sigma_{f_i} + \mu_{f_i}, g_{ij}\sqrt{2}\sigma_{g_i} + \mu_{g_i}\right), \\
q(f_i) &\sim \mathcal{N}(f_i|\mu_{f_i}, \sigma^2_{f_i}), \\
q(g_i) &\sim \mathcal{N}(g_i|\mu_{g_i}, \sigma^2_{g_i}),
\end{aligned} \tag{6.7}$$

where n represents the number of quadrature points, and w_j denotes the corresponding weight that belongs to the tuple (f_{ij}, g_{ij}) which is the input at which the function α is evaluated. The weights and the locations of the function evaluations are given in [99]. In the case of the heteroskedastic model defined above, the transformation $\alpha(y_i, f_i, g_i) = \log p(y_i|f_i, g_i)$. Figure 6.4 shows the tuples (f_{ij}, g_{ij}) with respect to which the weighted sum of $\log p(y_i|f_i, g_i)$ is calculated to approximate the corresponding expectation. The prediction of the latent function values f_{t+1} and g_{t+1} is simple and was defined previously by Equation (3.27). If the latent prediction is specified by

$$\begin{aligned}
q(f_{t+1}) &= \mathcal{N}(f_{t+1}|\mu_{f_{t+1}}, \sigma^2_{f_{t+1}}), \\
q(g_{t+1}) &= \mathcal{N}(g_{t+1}|\mu_{g_{t+1}}, \sigma^2_{g_{t+1}}),
\end{aligned} \tag{6.8}$$

then the prediction of the noise corrupted function value y_{t+1} is specified by

$$q(y_{t+1}) = \mathcal{N}(y_{t+1}|\mu_{f_{t+1}}, \mathbb{E}_{q(g_{t+1})}\left[h(g_{t+1})^2\right] + \sigma^2_{f_{t+1}}). \tag{6.9}$$

The term $\mathbb{E}_{q(g_{t+1})}\left[h(g_{t+1})^2\right] = \mathbb{E}_{q(g_{t+1})}\left[\alpha(g_{t+1})\right]$ can be computed with the 1D Gauss–Hermite quadrature [99]. The derivation of the posterior moments in the noise corrupted prediction is presented in Appendix E.1.5. The latent simulation response is obtained by the TCMC approximation, extended for the heteroskedastic model, defined by the Algorithm F.7.

For training, we used 1400 pseudo-inputs, initialized by subsampling the original data set. We used an alternating scheme for the optimization, where a step of ADAM optimizer [100] is performed on the hyperparameters, followed by a step of the NGD [38], [101] on the variational parameters. 20 inputs were used for the 1D Gauss–Hermite quadrature, and 400 for the 2D Gauss–Hermite quadrature.

Table 6.3 shows the results of the cross-validation on the PV generation validation data set. As in the load model, vector $\mathbf{z}$ denotes the inputs defined by a NARX model, i.e.

$$\mathbf{z}_i = [y^{\mathrm{PV}}_{i-n_a}, \ldots, y^{\mathrm{PV}}_{i-1}, \left(\mathbf{x}^{\mathrm{PV}}_{i-n_b}\right)^T, \ldots, \left(\mathbf{x}^{\mathrm{PV}}_{i-1}\right)^T]^T, \tag{6.10}$$

where y^{PV} denotes the PV generation observations. The lag parameters of the NARX model were selected as $(n_a, n_b) = (2, 1)$. The best kernel for modeling the process in the change point kernel was found to be a multiplication of the Matérn($\frac{5}{2}$) kernel applied on the time index with a RBF kernel applied on the NARX inputs. The best scale transformation function $h(\cdot)$ was found to be the exponential function. The likelihood kernel for the process $\mathbf{g}$ was also tested and is shown in Table 6.4. The best kernel for modeling the likelihood was found to be a multiplication of the Matérn($\frac{5}{2}$) kernel applied on the time index with a RBF kernel applied on the global solar irradiation input vector.

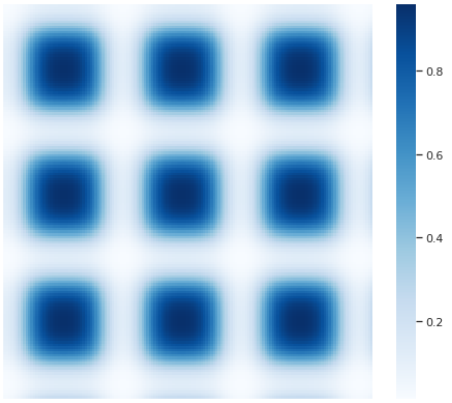
Figure 6.5 shows the separate blocks in designing the change point covariance function. The covariance matrices presented are for 3 consecutive days before training the hyperparameters on the observed data. Figure 6.5a and Figure 6.5b show the matrices corresponding to functions $\alpha(\cdot, \cdot)$ and $\beta(\cdot, \cdot)$, which represent the weights for combining the process kernel and the constant kernel in the change point covariance matrix. Figure 6.5c and Figure 6.5d represent the different kernels used for the day and night part. The final change point kernel is shown in Figure 6.5e, which is calculated as a weighted sum of the day and the night kernel. Figure 6.6 shows the same figures, but after training the hyperparameters on the observed data. This design of an application-specific kernel is a great demonstration of how prior knowledge can be built into the modeling process and why the simulation, unconstrained to the choice of the covariance function, is desired in practice.

Table 6.3: Results of the validation for selecting the process and likelihood covariance function for the PV generation model in the greater area of Sydney. The kernel is defined as a change point kernel switching between the constant kernel k_{Constant} and the process kernel. The change points are derived from the sunrise and sundown times in Sydney. The likelihood kernel used in this experiment was $k_{\text{Likelihood}} = k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1)$ with the corresponding scale transformation $h(\cdot)$. The softplus function is denoted by sfp. and the exponential function is denoted by exp. The selected kernel is emphasized in bold.

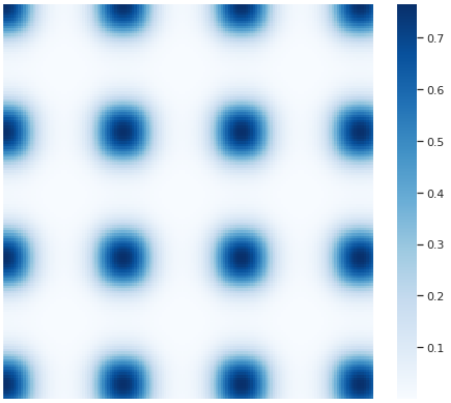
$h(\cdot)$	$k_{\text{Process}}(\cdot,\cdot)$	RMSE	SMSE	MSLL
exp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i, \mathbf{z}_j)$	560.81	0.0020	-5.015
exp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	**566.73**	**0.0020**	**-5.054**
exp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i, \mathbf{z}_j)$	610.74	0.0024	-4.913
exp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) + k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	619.01	0.0024	-4.934
exp.	$k_{\text{PER, RBF}}(t_i, t_j; \gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	570.52	0.0021	-5.060
exp.	$k_{\text{PER, RBF}}(t_i, t_j; \gamma_1) + k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	618.97	0.0024	-4.907
sfp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i, \mathbf{z}_j)$	558.57	0.0020	-4.811
sfp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	587.74	0.0022	-4.768
sfp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i, \mathbf{z}_j)$	613.92	0.0024	-4.907
sfp.	$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) + k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	616.03	0.0024	-4.935
sfp.	$k_{\text{PER, RBF}}(t_i, t_j; \gamma_1) + k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	618.88	0.0024	-4.928

Table 6.4: Results of the validation for selecting the likelihood covariance function for the PV generation model in the greater area of Sydney. The likelihood kernel used in this experiment was $k_{\text{Likelihood}} = k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_{\text{GSR},i}, \mathbf{z}_{\text{GSR},j})$. The function $h(\cdot)$ was the exponential function. Vector $\mathbf{z}_{\text{GSR}}$ denotes the global solar radiation. The selected kernel is emphasized in bold.

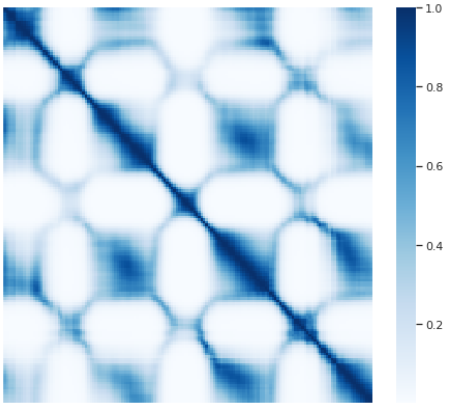
$k_{\text{Process}}(\cdot,\cdot)$	RMSE	SMSE	MSLL
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i, \mathbf{z}_j)$	561.49	0.0020	-5.054
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	**572.63**	**0.0021**	**-5.094**
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) + k_{\text{Matérn}(\frac{5}{2})}(\mathbf{z}_i, \mathbf{z}_j)$	615.46	0.0024	-4.941
$k_{\text{PER, Matérn}(\frac{5}{2})}(t_i, t_j; \gamma_1) + k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	622.33	0.0025	-4.959
$k_{\text{PER, RBF}}(t_i, t_j; \gamma_1) \times k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	574.44	0.0021	-5.083
$k_{\text{PER, RBF}}(t_i, t_j; \gamma_1) + k_{\text{RBF}}(\mathbf{z}_i, \mathbf{z}_j)$	626.96	0.0025	-4.968



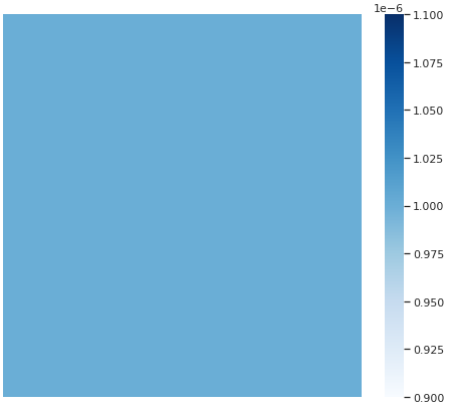
(a) $[\mathbf{A}]_{ij} = \alpha(t_i, t_j)$ for the day period before training.



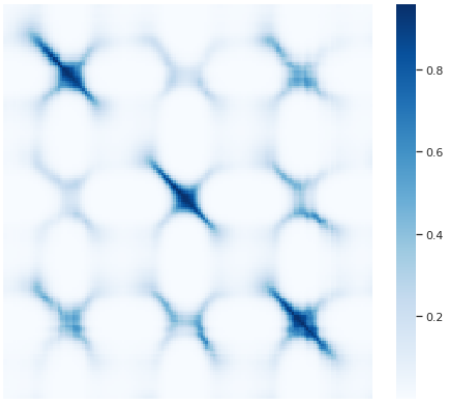
(b) $[\mathbf{B}]_{ij} = \beta(t_i, t_j)$ for the night period before training.



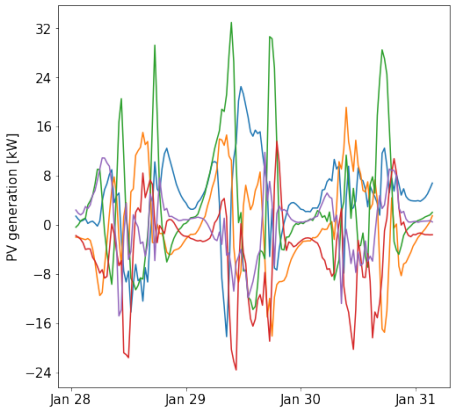
(c) $[\mathbf{C}]_{ij} = k_{\text{Process}}(\mathbf{z}_i, \mathbf{z}_i)$ for the day period before training.



(d) $[\mathbf{D}]_{ij} = k_{\text{Constant}}$ for the night period before training.

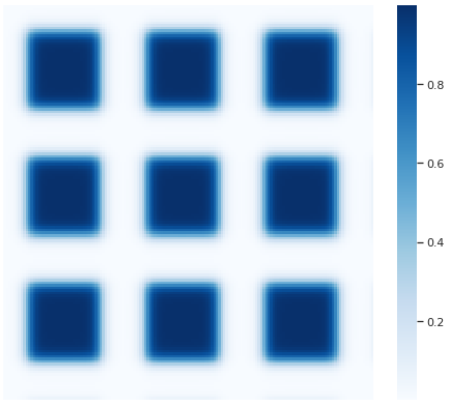


(e) $k_{\text{CP, } k_{\text{Constant}}, k_{\text{Process}}}(t_i, t_j, \mathbf{z}_i, \mathbf{z_j}) = [\mathbf{K}_{f_{1:t}, f_{1:t}}]_{ij}$ for the PV generation model before training.

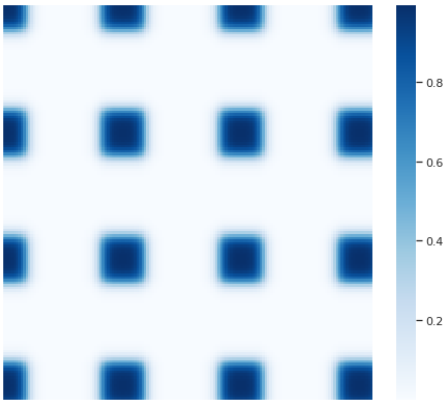


(f) 5 function samples from the prior for the PV generation model before training.

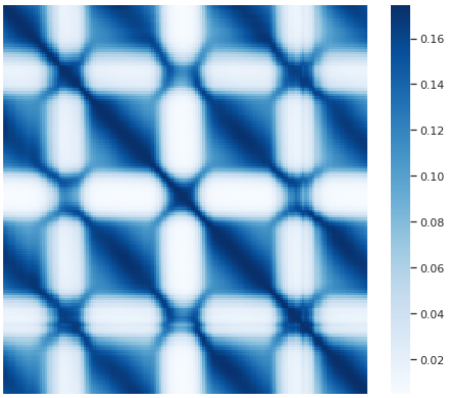
Figure 6.5: The covariance matrix of the prior for the PV generation model and the corresponding function samples before training. The data are displayed for 3 consecutive days. The PV covariance matrix is calculated as $\mathbf{K}_{f_{1:t}, f_{1:t}} = \mathbf{A} \odot \mathbf{C} + \mathbf{B} \odot \mathbf{D}$, where $\odot$ denotes the Hadamard product (elementwise product). See Appendix C.6 for more details.



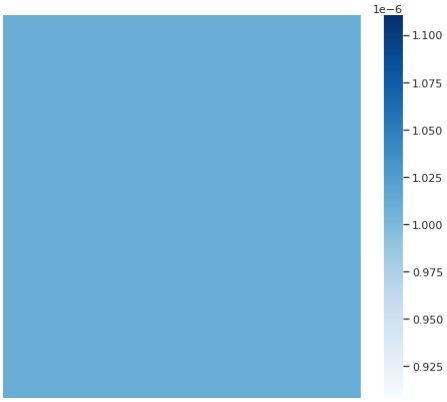
(a) $[\mathbf{A}]_{ij} = \alpha(t_i, t_j)$ for the day period after training.



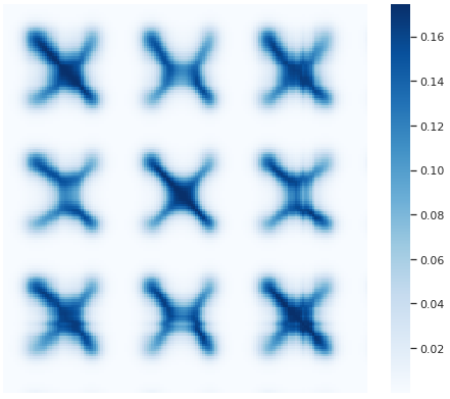
(b) $[\mathbf{B}]_{ij} = \beta(t_i, t_j)$ for the night period after training.



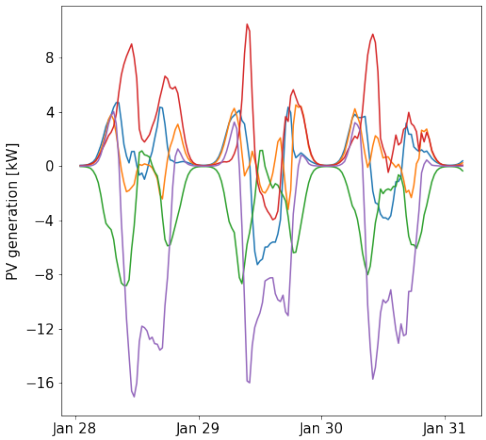
(c) $[\mathbf{C}]_{ij} = k_{\text{Process}}(\mathbf{z}_i, \mathbf{z}_i)$ for the day period after training.



(d) $[\mathbf{D}]_{ij} = k_{\text{Constant}}$ for the night period after training.



(e) $k_{\text{CP}, k_{\text{Constant}}, k_{\text{Process}}}(t_i, t_j, \mathbf{z}_i, \mathbf{z_j}) = [\mathbf{K}_{f_{1:t}, f_{1:t}}]_{ij}$ for the PV generation model after training.



(f) 5 function samples from the prior for the PV generation model after training.

Figure 6.6: The covariance matrix of the prior for the PV generation model and the corresponding function samples after training. The data are displayed for 3 consecutive days. The PV covariance matrix is calculated as $\mathbf{K}_{f_{1:t}, f_{1:t}} = \mathbf{A} \odot \mathbf{C} + \mathbf{B} \odot \mathbf{D}$, where $\odot$ denotes the Hadamard product (elementwise product). See Appendix C.6 for more details.

6.4 Short-Term Load and Photovoltaic Generation Forecast

The models for load and PV generation forecasting were evaluated on a separate test data set. Two baseline models were considered. The first one is a persistent forecast where the half-hour prediction for the time step y_{t+i} is simply a half-hour lagged observation of the respective output, i.e. y_{t+i-1}. A moving average (MA) forecast was also considered, which takes periodicity into account and can provide a probabilistic estimation. The forecasts from the MA model were obtained by averaging the data for the last week, where the forecasts were grouped by the hour of the day. Similarly, the variance of the forecasts was estimated.

Table 6.5 shows the results of the half-hour prediction. We can see that the GP-NARX models outperform the baselines for the load and PV generation. Figure 6.7 shows the scatter plot between the half-hour prediction and the observed values. We can see that the model forecast describes the observations well. The models work well even in the probabilistic sense, which is seen by the MSLL performance metric in Table 6.5. An example of the half-hour prediction can be seen in Figure 6.8 and Figure 6.9 which shows the load and PV generation forecast for two different parts of the year.

However, we are interested in forecasting the load and PV generation for the whole next day. The forecast is repeated for each time step in the day for the next 24 hours. This rolling probabilistic forecast is then used in the chance-constrained OPF where the operating limits of the PV generation are recalculated at a half-hour interval for the next 24 hours. In our case study, the forecast of the respective output is considered up to the end of the day because of the nature of the test and the training data set splits. We used a TCMC simulation approximation with $T_{\mathbb{V}} = 10^{-3}$ and 500 MC samples.

The rolling forecast in a single test day is performed as depicted in Figure 6.10. For example, in a single day we have 48 half-hour predictions, 47 1-hour predictions, and so on. In a single day, we only have 1 24-hour prediction. We consider the prediction up to 6 hours into the future, for which a reasonably large sample can be collected. Two scenarios are tested:

1. Observation available with a time delay of 1 day, i.e $\Delta t = 48 = 24$ hours;

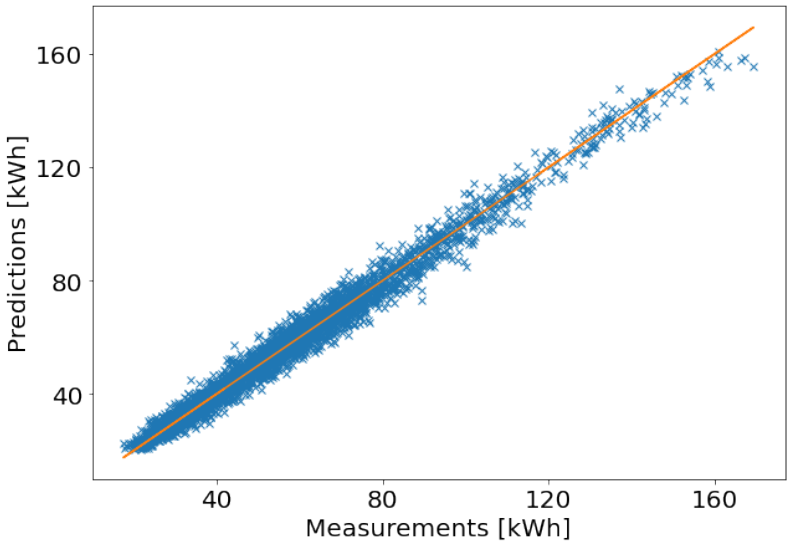
2. Observations available in real-time, i.e $\Delta t = 1 = 30$ minutes.

The results are considered from the perspective of the current time, where the forecasts are grouped by the prediction horizon and are shown in Figure 6.11.

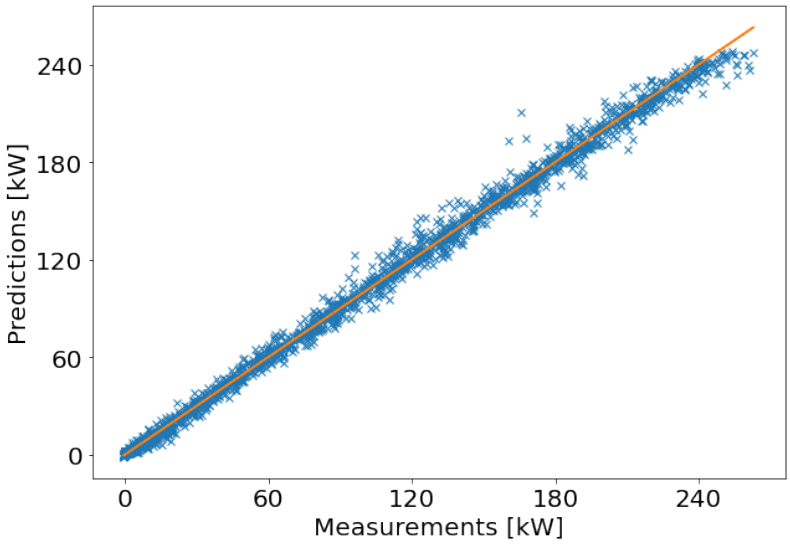
From the perspective of the model, the delayed observations effectively increase the number of predicted steps into the future. This can be seen in Figure 6.11 with worse performance metrics of the model where the data are available with a 24-hour delay, when compared to the model where the data are available in real-time. This is due to the fact that the benefits of autoregressive inputs diminish over time. After some period, the forecast is mostly deduced from the weather observations, i.e. as if the model would be trained only on the weather data. The degradation of the model performance with increased time delay of the available data justifies the need for real-time observations which are currently not available in New South Wales.

Table 6.5: Results of the half-hour prediction on the test data set. The performance metrics for the PV generation were only considered for the day period. The best results are emphasized in bold.

Output	Model	RMSE	SMSE	MSLL
Load	GP-NARX (VFE)	**581.90**	**0.018**	**-2.002**
	Persistent	826.67	0.041	-
	Moving average	1874.34	0.213	-0.413
PV generation	HGP-NARX	**840.26**	**0.004**	**-3.086**
	Persistent	2931.81	0.052	-
	Moving average	4525.75	0.122	-1.396

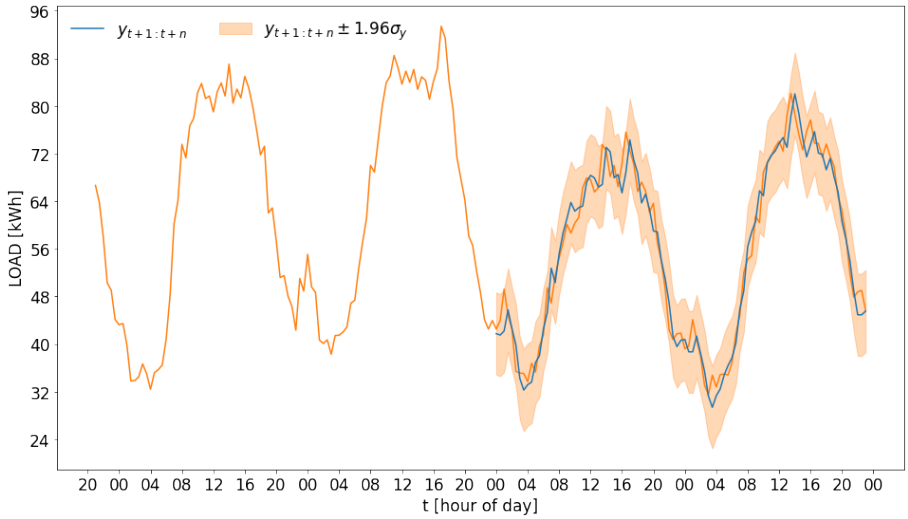


(a) Load half-hour prediction.

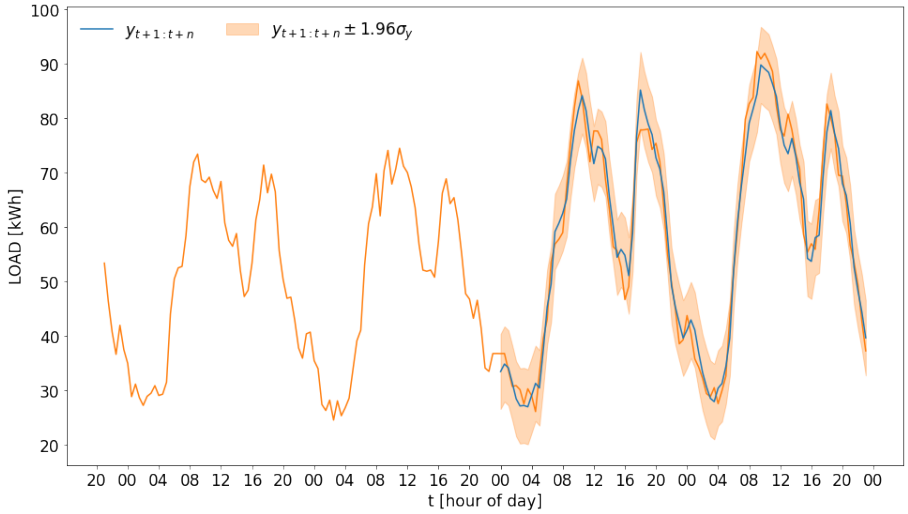


(b) PV generation half-hour prediction.

Figure 6.7: Scatter plot between the half-hour prediction and the observed values for the load and the PV generation model.

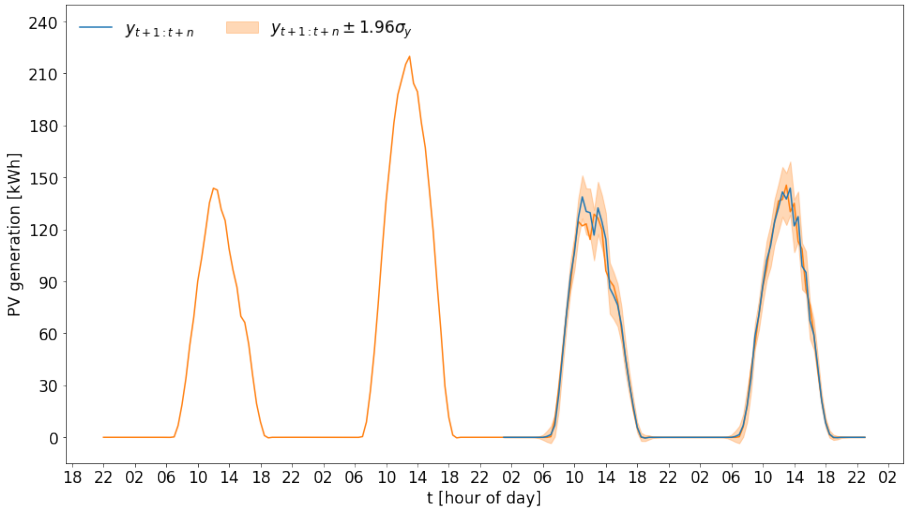


(a) Half-hour prediction of the electrical load for the noisy process $y_{t+1:t+n}$ between the 4th and 5th of February, 2019.

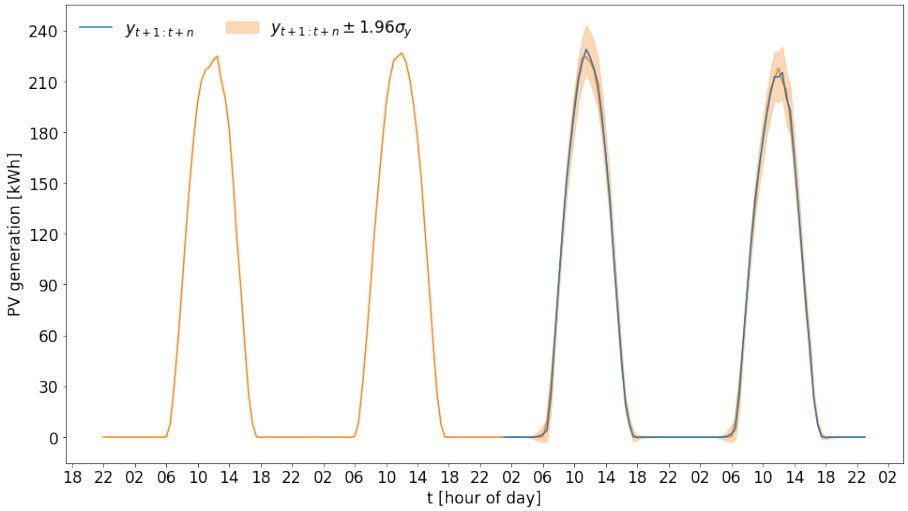


(b) Half-hour prediction of the electrical load for the noisy process $y_{t+1:t+n}$ between the 4th and 5th of July, 2019.

Figure 6.8: Example of the half-hour prediction for the electrical load on the test data set. The first two days are a part of the training data set and the following two days are a part of the test data set.

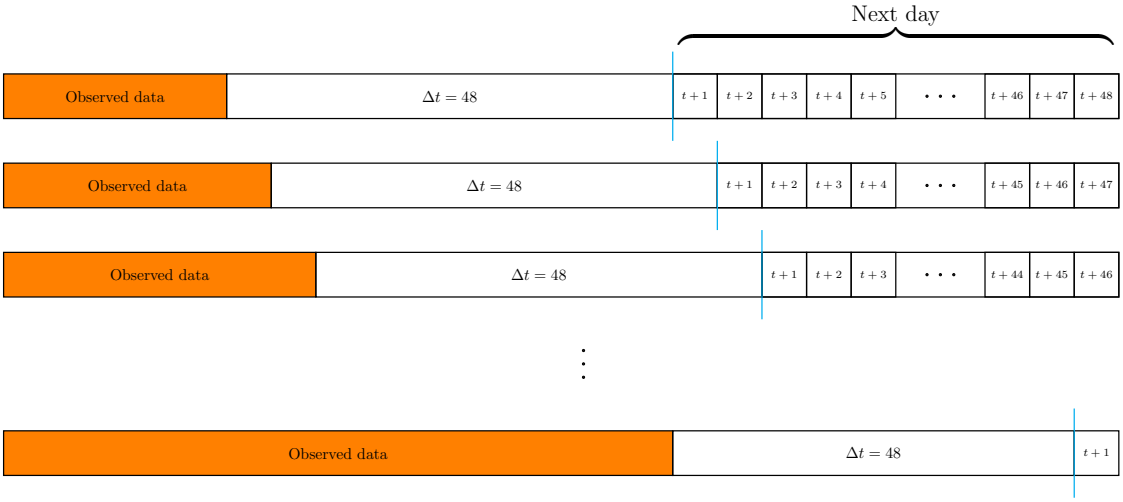


(a) Half-hour prediction of the PV generation for the noisy process $y_{t+1:t+n}$ between the 4th and 5th of April, 2019.



(b) Half-hour prediction of the PV generation for the noisy process $y_{t+1:t+n}$ between the 4th and 5th of September, 2019.

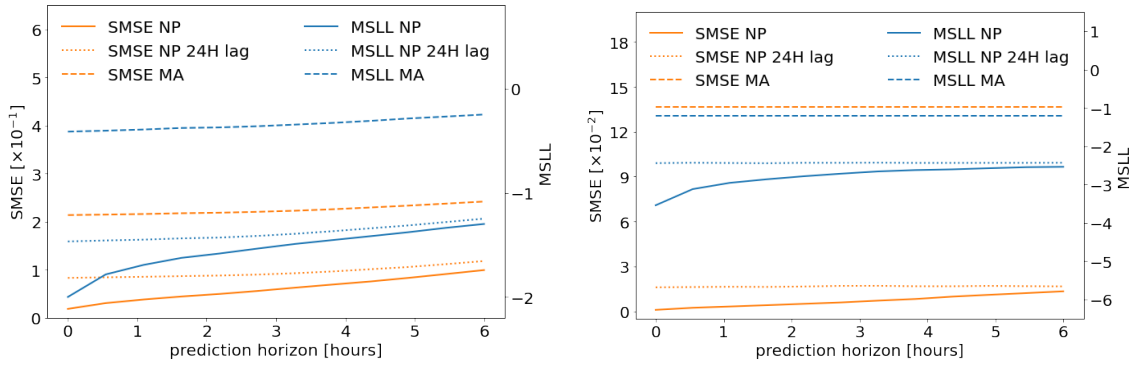
Figure 6.9: Example of the half-hour prediction for the PV generation on the test data set. The first two days are a part of the training data set and the following two days are a part of the test data set.



(a) 24-hour delayed observations.

(b) Real-time observations.

Figure 6.10: Diagram depicting the rolling forecast in testing the load and PV generation models for the next day. Note that the steps ahead are considered from the current time and not from the perspective of the model. From the perspective of the model, the delayed observations effectively increase the number of predicted steps into the future.



(a) Load forecast versus the number of predicted steps into the future.

(b) PV generation forecast versus the number of predicted steps into the future.

Figure 6.11: Load and PV generation forecast versus the number of predicted steps into the future for the SMSE and the MSLL performance metrics. The results are compared to the persistent and moving average forecast as a baseline. The performance metrics for the PV generation were only considered for the day period.

6.5 Long-Term Load and Photovoltaic Generation Forecast

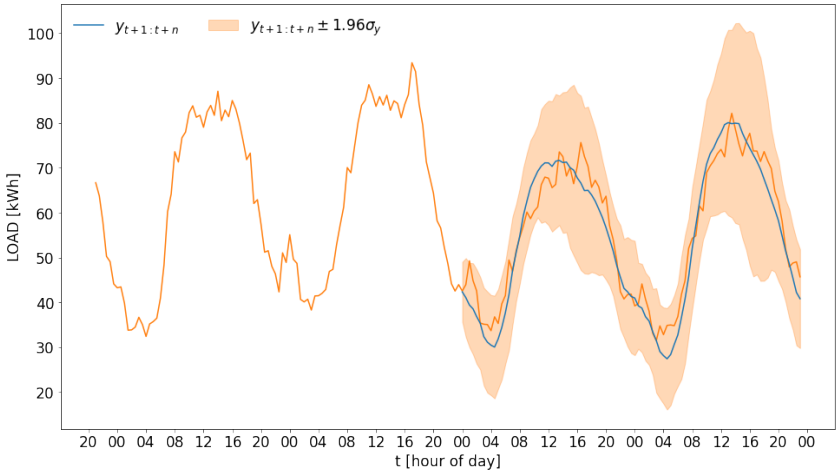
For completeness, we tested the model also for the simulation. In our case study, this is performed individually on consecutive days in the test data set. Table 6.6 shows the results for the simulation. We can see that the simulated response that best describes the respective forecasts is the one with noise propagation (NP). The simulation response describes the observation relatively well in the mean sense, as well as in the probabilistic sense. GP-NARX models outperform the baseline, especially when the forecast is validated probabilistically.

Figure 6.12 depicts the simulation response for the electrical load, whereas Figure 6.13 and Figure 6.13 depict the simulation response for the PV generation. Two consecutive days in different parts of the seasons are considered. Figure 6.13a and Figure 6.13b show the latent response in blue and the corresponding response of the process that models the heteroskedastic standard deviation in red. Figure 6.14a and Figure 6.14b show the noisy response estimated from the latent samples and the standard deviation samples. We can see how the likelihood is modeled heteroskedastically where the standard deviation, shown in red, is larger with increased PV generation. Overall, we can see that the models describe the observations well in terms of the mean value and also probabilistically.

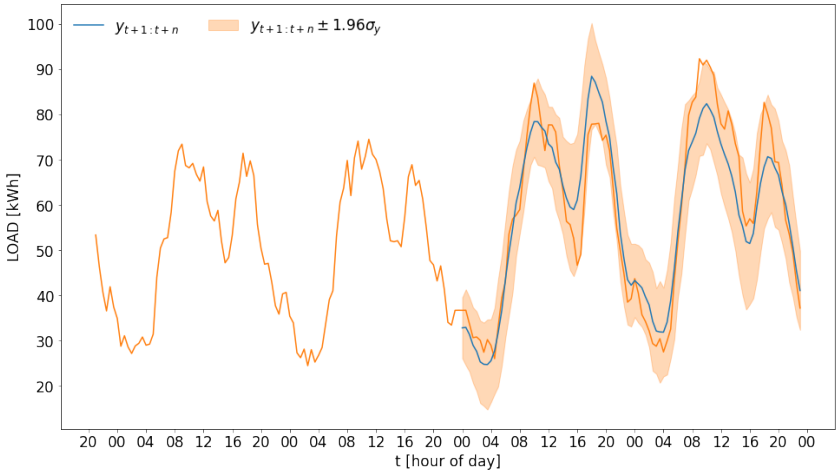
This study demonstrates the flexibility of the GP-NARX models to model large data sets in a grey-box approach. It is very important in such studies that the GP models allow for a relatively fast experimentation. We have shown this by modeling the electrical load and PV generation where the dynamics are described by a composite model, combining a NARX model with a timeseries approach. The GP approximations for the specific case were easily extended to model dynamical systems, i.e. the VFE approximation used in the load model and the SVGP approximation used in the PV generation model. The flexible covariance function design and the choice of the GP approximation are enabled by the flexible and scalable MC algorithms in the simulation.

Table 6.6: Results of the 48-hour simulation on the test data set. Simulation without NP is denoted by †, and simulation with NP is denoted by ‡. The performance metrics for the PV generation were only considered for the day period. The best results are emphasized in bold.

Output	Model	RMSE	SMSE	MSLL
Load	GP-NARX (VFE)†	1197.87	0.078	-1.187
	GP-NARX (VFE)‡	**1175.90**	**0.075**	**-1.454**
	Moving average	1874.34	0.213	-0.413
PV generation	HGP-NARX†	3892.23	0.090	-1.145
	HGP-NARX‡	**3853.19**	**0.089**	**-1.727**
	Moving average	4525.75	0.122	-1.396

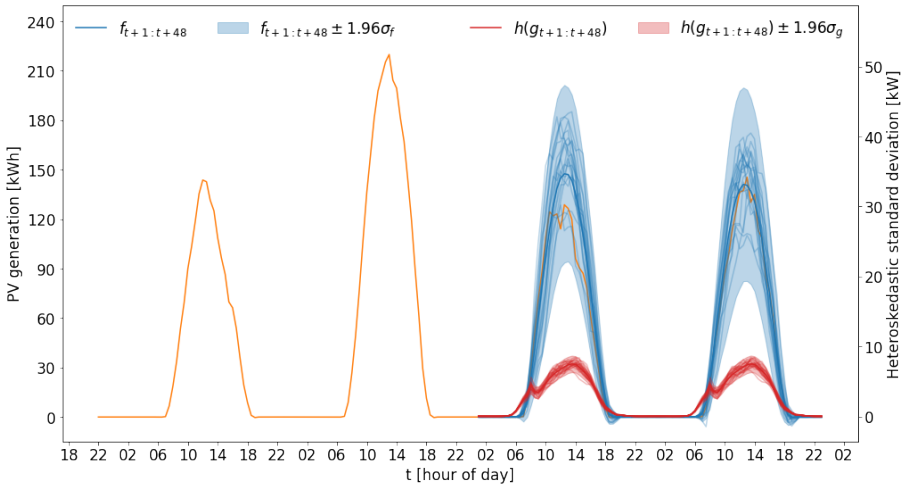


(a) 48-hour simulation of the electrical load for the noisy process $y_{t+1:t+n}$ between the 4th and 5th of February, 2019.

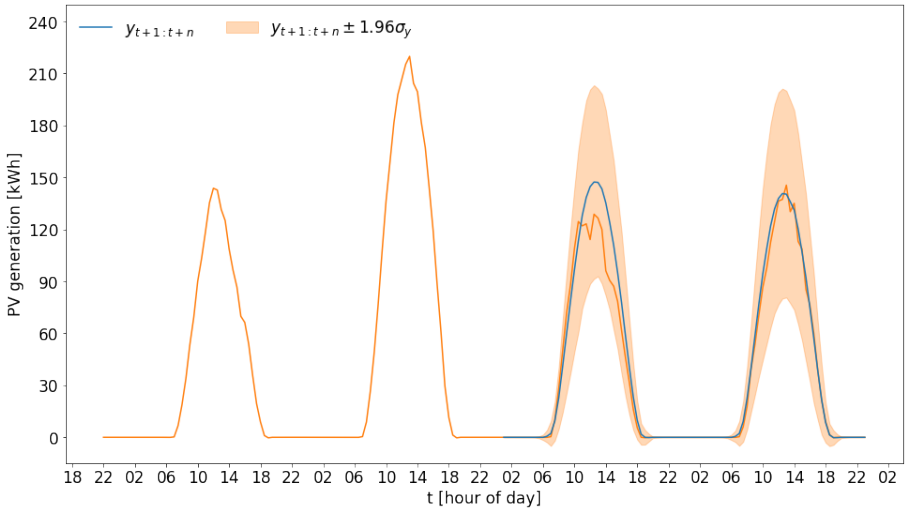


(b) 48-hour simulation of the electrical load for the noisy process $y_{t+1:t+n}$ between the 4th and 5th of July, 2019.

Figure 6.12: Example of the 48-hour simulation for the electrical load on the test data set. The first two days are a part of the training data set and the following two days are a part of the test data set.

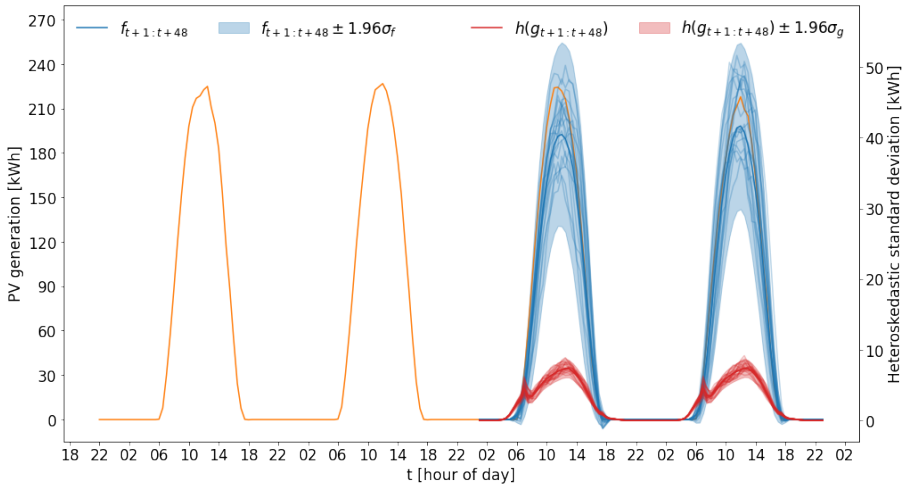


(a) 48-hour simulation of the PV generation for the latent process $f_{t+1:t+n}$ in blue and the estimated heteroskedastic likelihood standard deviation $h(g_{t+1:t+n})$ in red for the period between the 4th and 5th of April, 2019.

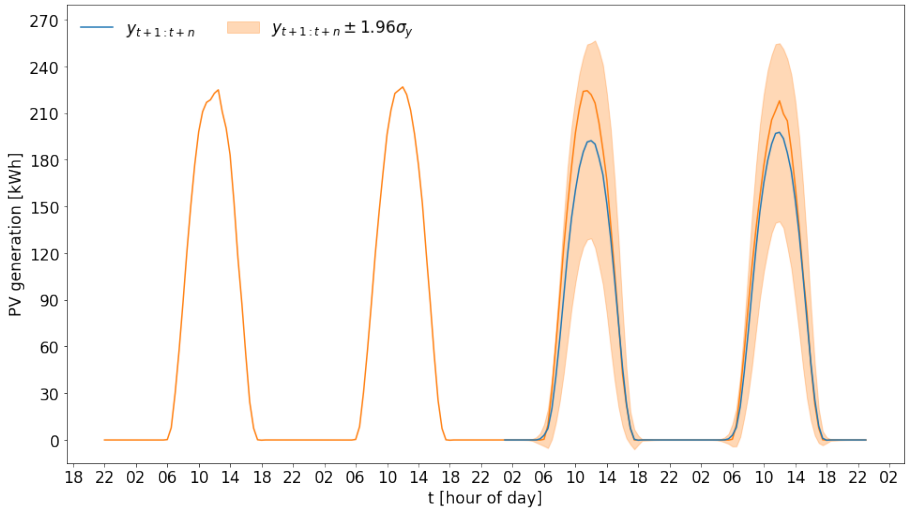


(b) 48-hour simulation of the PV generation for the noisy process $y_{t+1:t+n}$ for the period between the 4th and 5th of April, 2019.

Figure 6.13: Example of the 48-hour simulation for the PV generation model between the 4th and 5th of April, 2019. Latent process, estimate of the heteroskedastic likelihood standard deviation, and the noisy simulated response on the test data are shown. The first two days are a part of the training data set and the following two days are a part of the test data set.



(a) 48-hour simulation of the PV generation for the latent process $f_{t+1:t+n}$ in blue and the estimated heteroskedastic likelihood standard deviation $h(g_{t+1:t+n})$ in red for the period between the 4th and 5th of September, 2019.



(b) 48-hour simulation of the PV generation for the noisy process $y_{t+1:t+n}$ for the period between the 4th and 5th of September, 2019.

Figure 6.14: Example of the 48-hour simulation for the PV generation model between the 4th and 5th of April, 2019. Latent process, estimate of the heteroskedastic likelihood standard deviation, and the noisy simulated response on the test data are shown. The first two days are a part of the training data set and the following two days are a part of the test data set.

Chapter 7

Conclusions

In this thesis, we considered a probabilistic modeling approach. We believe that this is not only a reasonable, but also a necessary approach to building models from data. The uncertainty can arise from various sources, e.g.

- Real systems can be stochastic;
- We observe a finite subset of data;
- Some predictors are latent.

Quantifying the epistemic and the aleatoric uncertainty allows us to rigorously separate the different sources of stochasticity:

- The uncertainty that cannot be reduced in practice (noise);
- The uncertainty that can be reduced in practice if we observed more data.

When the assumed probabilities are Gaussian, and the system is time-invariant and linear, the desired mathematical objects quantifying uncertainty can be obtained in closed-form. However, the problems in the real world are seldom linear.

GP models are a powerful probabilistic tool. They provide analytic expressions of the marginal likelihood and of the predictive posteriors when the inputs are deterministic. The smoothness assumption, essential in nonparametric models, is incorporated through the design of the covariance function. In combination with a NARX model, the elegant properties of static GPs are preserved to model dynamical systems.

GP-NARX models are very practical to work with since the training is reduced to that of the static case and missing data can be handled trivially. GP-NARX models are, for that reason, simple to experiment with and can be scaled to large amounts of data since there is no recurrent estimation of the latent function. However, the training of GP models is computationally demanding and results in a cubic computational complexity with respect to the number of training data.

In Chapter 3, we demonstrated how the latent function values are retained in memory and considered in the joint model on sequential estimation of a static posterior. However, in static systems, there is no input/output dependency as defined by the NARX model. In the simulation of GP-NARX models, the Gaussian distribution is propagated through the nonlinear covariance function and the response cannot be obtained in closed-form. The estimate of the simulation can be obtained by MC integration. Unfortunately, the computational complexity of the ground truth numerical estimation, i.e. the FCMC simulation presented in Chapter 3, again results in cubic computational complexity with respect to

the predicted steps into the future. This restricts the vanilla GP-NARX model to a few thousand latent values, in training and in simulation.

The computational complexity with respect to the number of observed data can be efficiently reduced through sparse models. A smaller set of pseudo-inputs (when compared to the data set size) is introduced with the goal to find an appropriate set of pseudo-inputs that can effectively replace the original data. However, the computational complexity of the sparse models in simulation still depends cubically with respect to the number of predicted steps into the future. Additionally, each sparse approximation has a specific algorithm for the simulation of the respective GP-NARX model, which is not practical for two reasons:

- Errors in the implementation of tedious sequential algorithms;
- The elegant benefits of the NARX model in training are diminished with complex simulation procedures.

For the reasons above, we proposed a unified view of the simulation of GP-NARX models in Chapter 4, invariant to the choice of the sparse approximation up to the batch sample from the pseudo-point posterior. We proposed an approximation to the simulation, i.e. the TCMC simulation, where the posterior variance is used as a measure of the informational importance for the latent sample considered at the time step. If the posterior variance is lower than the user-defined threshold, the latent observation is discarded and not kept in memory. The user-defined parameter can be used to maximize the computational resources. When the parameter is selected to be infinite, the TCMC simulation corresponds to the PIMC simulation. When the parameter is selected to be the jitter that is added to covariance matrices for numerical stability, the TCMC simulation can be considered exact. For that reason, the TCMC simulation can also serve as the ground truth for future more scalable approximations.

We compared the TCMC simulation with the ground truth, i.e. the FCMC simulation, and the CIMC simulation. The analysis was conducted on a nonlinear static function and an illustrative dynamical example. We used the static function to compare the results to the closed-form estimation of the posterior. We conclude that the TCMC simulation can provide significant computational benefits when the inducing functions from the kernel are relatively smooth. The simulation can be evaluated for only a fraction of the cost of the FCMC simulation. However, the estimated posterior of the latent distribution is exact when the threshold is selected to be the numerical jitter. If the induced functions are not smooth, the computational benefits quickly diminish. In that case, the PIMC simulation is computationally tractable and should be preferred over the CIMC simulation since the estimated response using a PIMC simulation better approximates the ground truth and can be obtained in identical computational time as the CIMC simulation. But we can expect some error in the estimation of the latent uncertainty.

For the Matérn($\frac{1}{2}$) covariance function, the TCMC simulation did not provide any computational benefits. The observed latent points are not well correlated in this case, i.e. an observed latent value does not even reduce the posterior variance locally. With such a rough function assumption it is hard to justify the use of GPs and their computational complexity. The computational complexity originates from the definition that all the latent points are fully correlated. Even if these correlations are small, as with the Matérn($\frac{1}{2}$) kernel, the computational complexity persists. In this case, a more appropriate model can be used instead of GPs in the first place.

For the FITC approximation, the number of retained latent values in TCMC estimation is higher when compared to the variational approximation. We believe that this is due to the conditional independence assumption between the latent function values that belong to the training data, which partially decorrelate the aforementioned values in the

regions where pseudo-points are scarce. Consequently, the learned hyperparameters induce rougher functions which in turn increase the running time of the simulation. Given that theoretical benefits of the variational approximations are already accepted in the literature, and we empirically demonstrated the superiority of the variational approximations for the prediction of GP-NARX models in Section 3.2.2 on 10 chaotic time series, this insight provides yet another reason why the variational frameworks should be preferred over the FITC approximation.

In practice, the functions that assume a degree of smoothness are generally preferred. This is because smooth functions generalize better in the presence of noise. Determining the right amount of flexibility in a data-driven modeling approach is the core of statistical learning. In GPs, model complexity scales with training data, and the flexibility is determined automatically. However, the covariance function has to be chosen carefully.

At the end of Chapter 4, we demonstrated the use of the GP-NARX (VFE) model on two benchmark data sets. The data sets cannot be modeled by the vanilla GP-NARX due to the size of the data set. We empirically showed that the RBF and the Matérn($\frac{5}{2}$) covariance functions performed best, which were the smoothest kernels used. We demonstrated how a correlated simulation can be obtained using the TCMC simulation, which was previously not possible. The prediction and the TCMC simulation were validated on an independent data set. The data were well modeled in the mean sense which was demonstrated using the RMSE and SMSE metric, and also in the probabilistic sense, which was demonstrated using the MSLL metric. We also showed how the noise propagation can be utilized in simulation and how the posterior statistical moments can be influenced by it since the probabilities are propagated in a nonlinear fashion.

In Chapter 5, we presented a case study that dealt with modeling the local weather dynamics. We modeled the atmospheric variables in the vicinity of NPP Krško. The forecasts of the atmospheric variables are used in a system for modeling the emitted particle dispersion to advise the actions in the case of a nuclear accident. The results of the half-hour prediction, up to 12-hour prediction, and simulation were compared to the forecasts from the NWP model that was previously used in this setup. We can conclude that the GP-NARX (VFE) model performed well and outperformed the NWP forecasts even in the extreme case of the simulation. Similarly, as in the case of the two benchmarks, the smoothest kernels performed best.

In Chapter 6, we demonstrated the powerful GP-NARX framework for modeling the load and PV generation in the Greater Sydney Area. We showed how GP-NARX models can be utilized as a grey-box method and how prior knowledge can be incorporated through the covariance function design and the design of the likelihood model. For fast experimentation and flexibility, a simulation algorithm that is not constrained to the specific selection of the covariance function is of great interest. We modeled the aforementioned outputs in a nonparametric sense, where the uncertainty is quantified systematically, which is important since the probabilistic forecasts serve as inputs to the chance-constrained optimal power flow optimization problem which determines the steady state operating point that minimizes the cost of electric power while satisfying operating constraints and meeting demand. An experiment was conducted to find the effect of the 24-hour time delay on availability of the load and PV generation observations to the distribution system operators. We conclude that the results in a 24-hour delayed experiment are significantly worse which justifies the need for real-time observations and installations of smart meters.

In retrospect, the hypotheses defined in Section 1.2 were confirmed. Explicitly:

- H_1: We have shown that the CIMC simulation is a rough approximation to the simulation of GP-NARX models. Not only that it produces overly noisy simulation samples, but the posterior statistical moments are also affected since the distributions are propagated through a nonlinear function;
- H_2: The demanding computational requirements of the fully correlated GP-NARX simulation can be reduced with sparse approximated GP-NARX models. We showed with the unified view of the simulation that in the case of sparse approximations, the posterior over the training data points are replaced with the posterior of the pseudo-points. The simulation is initialized with fewer latent observations from the start. However, the computational complexity with each predicted step into the future persists and all consecutive latent function values are kept in memory;
- H_3: Pseudo-points do affect the simulation. They allow for a new approximation, namely the PIMC simulation, that does not keep the consecutive latent observations in memory. However, some latent uncertainty is still reduced which is especially evident in kernels that induce smooth functions. PIMC should be preferred over the CIMC simulation since it better approximates the ground truth and has the same computational complexity;
- H_4: An approximation to the fully correlated Monte Carlo simulation algorithm was derived for sparse approximated GP-NARX models. TCMC simulation can reduce the computational requirements and improve the estimation of the latent response when compared to the CIMC and PIMC simulation. It introduces a trade-off parameter that can maximize the computational resources and, consequently, the quality of the estimated latent response. If the threshold equals the numerical jitter, the TCMC simulation retrieves the FCMC simulation for only a fraction of the computational cost of the ground truth when kernels that induce smooth functions are used.

In future work, we aim to reconsider the assumption of the NARX model, i.e. that the outputs introduced in the input matrix are noisy. This assumption allows the GP-NARX models to be trained in a simple manner. However, the latent state is not filtered in this case. GP-SSM models filter the state but introduce a complex and analytically intractable training procedure. An interesting avenue of research would be to find the compromise between the two approaches. For the simulation of the GP-NARX model, an improvement to the computational complexity could be looked-for. An appealing solution is potentially a combination of the TCMC simulation and samplers that approximate the latent posterior with basis functions at test time. Lastly, one could seek new applications for the GP-NARX models and try to devise covariance functions for the specifics of the use case considered.

Appendix A

Gaussian and Matrix Identities

A.1 Gaussian Identities

This section will define the Gaussian identities useful for understanding and implementing Gaussian process regression models.

A.1.1 Multivariate Gaussian distribution

The multivariate Gaussian distribution is defined by

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{K}) = |2\pi\mathbf{K}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{A.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\boldsymbol{\mu} \in \mathbb{R}^n$, $\mathbf{K} \in \mathbb{R}^{n \times n}$ represent the mean vector and the covariance matrix respectively.

A.1.2 Gaussian marginals, conditionals, and linear transformations

If the joint Gaussian distribution between random variables $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x} \\ \mathbf{y}\end{bmatrix} \middle| \begin{bmatrix}\mathbf{a} \\ \mathbf{b}\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B}\end{bmatrix}\right), \tag{A.2}$$

then the marginal distribution $p(\mathbf{x})$ of a multivariate normal distribution $p(\mathbf{x}, \mathbf{y})$ is specified by

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y})\, d\mathbf{y} = \mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A}). \tag{A.3}$$

The conditional distribution $p(\mathbf{x}|\mathbf{y})$ is defined by

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mathbf{x}|\mathbf{a} + \mathbf{C}^T\mathbf{B}^{-1}(\mathbf{y}-\mathbf{b}), \mathbf{A} - \mathbf{C}^T\mathbf{B}^{-1}\mathbf{C}\right). \tag{A.4}$$

A linear transformation of a Gaussian random variable can be evaluated in closed-form and is defined by

$$p(\mathbf{x}) = \int \mathcal{N}(\mathbf{x}|\mathbf{a} + \mathbf{F}\mathbf{y}, \mathbf{A})\, \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{B})\, d\mathbf{y} = \mathcal{N}\left(\mathbf{x}|\mathbf{a} + \mathbf{F}\mathbf{b}, \mathbf{A} + \mathbf{F}\mathbf{B}\mathbf{F}^T\right). \tag{A.5}$$

A special distribution where the probability density is infinite at $\mathbf{x} = \mathbf{a}$ can be described by Dirac delta defined by

$$\delta(\mathbf{x} - \mathbf{a}) = \begin{cases} \infty, & \mathbf{x} = \mathbf{a} \\ 0, & \mathbf{x} \neq \mathbf{a} \end{cases} \tag{A.6}$$

where $\int_{-\infty}^{\infty} \delta(\mathbf{x})d\mathbf{x} = 1$. If we consider the special case in Equation (A.5) where $\mathbf{A} = 0$, then

$$p(\mathbf{x}) = \int \delta(\mathbf{x} - (\mathbf{a} + \mathbf{F}\mathbf{y}))\mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{B})\, d\mathbf{y} = \mathcal{N}\left(\mathbf{x}|\mathbf{a} + \mathbf{F}\mathbf{b}, \mathbf{F}\mathbf{B}\mathbf{F}^T\right). \tag{A.7}$$

A.1.3 Sampling from a multivariate Gaussian

A draw from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{K})$ is taken as

$$\mathbf{K} = \mathbf{L}\mathbf{L}^T, \tag{A.8a}$$

$$\text{draw } \tilde{\mathbf{z}} \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}), \tag{A.8b}$$

$$\tilde{\mathbf{x}} = \boldsymbol{\mu} + \mathbf{L}\tilde{\mathbf{z}}, \tag{A.8c}$$

where $\mathbf{L}\mathbf{L}^T$ is the Cholesky decomposition of the covariance matrix $\mathbf{K}$ defined by Equation (A.10).

A.1.4 Expectation over a squared form

If $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{K})$, then

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] = \boldsymbol{\mu}\boldsymbol{\mu}^T + \mathbf{K}, \tag{A.9a}$$

$$\mathbb{E}\left[\mathbf{x}^T\mathbf{A}\mathbf{x}\right] = \boldsymbol{\mu}^T\mathbf{A}\boldsymbol{\mu} + \mathrm{Tr}(\mathbf{A}\mathbf{K}). \tag{A.9b}$$

A.2 Matrix Identities

A.2.1 Cholesky decomposition

Positive definite matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ can be decomposed to

$$\mathbf{K} = \mathbf{L}\mathbf{L}^T, \tag{A.10}$$

where $\mathbf{L}$ is a lower diagonal matrix with positive diagonal entries. We denote the Cholesky decomposition explicitly as $\mathbf{K} = \text{cholesky}\left(\mathbf{L}\mathbf{L}^T\right)$ throughout this thesis. The decomposition can be obtained in $\mathcal{O}\left(n^3\right)$ time.

A.2.2 Iterative Cholesky update

Assume a row/column is added to the symmetric positive definite matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$

$$\hat{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \tag{A.11}$$

and $\mathbf{L}\mathbf{L}^T$ is the Cholesky decomposition of $\mathbf{K}$. Let us consider an updated Cholesky decomposition

$$\hat{\mathbf{K}} = \hat{\mathbf{L}}\hat{\mathbf{L}}^T, \tag{A.12}$$

where

$$\hat{\mathbf{L}} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{c}^T & d \end{bmatrix}. \tag{A.13}$$

An update can be obtained in $\mathcal{O}(n^2)$ time. It is specified by

$$\mathbf{c} = \mathbf{L}^{-1}\mathbf{a}, \tag{A.14a}$$

$$d = \sqrt{b - \mathbf{c}^T\mathbf{c}}. \tag{A.14b}$$

A.2.3 Determinant of a positive-definite matrix

The determinant of a positive definite matrix K can be computed by

$$|\mathbf{K}| = |\mathbf{L}\mathbf{L}^T| = |\mathbf{L}||\mathbf{L}^T| = 2\prod_{i=1}^{n} [\mathbf{L}]_{ii}\,. \tag{A.15}$$

A.2.4 Woodbury identity

Woodbury identity defines

$$(\mathbf{A}+\mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}\mathbf{A}^{-1}, \tag{A.16}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{U} \in \mathbb{R}^{n\times k}, \mathbf{V} \in \mathbb{R}^{k\times n}$, and $\mathbf{C} \in \mathbb{R}^{k\times k}$.

Appendix B

Performance Metrics

B.1 Root Mean Squared Error

Root mean squared error (RMSE) is defined by

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_i)^2}, \tag{B.1}$$

where y_i denotes the observed value at the time step i, μ_i the predicted mean at the time step i, and N defines the number of data points.

B.2 Standardized Mean Squared Error

Standardized mean squared error (SMSE) is defined by

$$\text{SMSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \mu_i}{\sigma}\right)^2, \tag{B.2}$$

where y_i denotes the observed value at the time step i, μ_i the predicted mean at the time step i, σ is the standard deviation of the observed data $\mathbf{y} = [y_1, \ldots, y_N]$, and N is the number of data points.

B.3 Mean Standardized Log Loss

The mean standardized log loss (MSLL) is defined by standardizing the mean log loss (MLL) by subtracting the loss that would be obtained under the trivial model which predicts using a Gaussian with mean and variance of the observed data $\mathbf{y}$ [22]. MSLL is defined by

$$\begin{aligned} \text{MLL} &= \frac{1}{2N}\sum_{i=1}^{N}\left[\log(2\pi\sigma_i^2) + \left(\frac{y_i - \mu_i}{\sigma_i}\right)^2\right], \\ \text{MSLL} &= \text{MLL} - \frac{1}{2N}\sum_{i=1}^{N}\left[\log(2\pi\sigma^2) + \left(\frac{y_i - \mu}{\sigma}\right)^2\right]. \end{aligned} \tag{B.3}$$

The predicted mean at the time step i is denoted by μ_i, σ_i is the predicted standard deviation of the noisy values at the time step i, N is the number of data points, and μ, σ denote the respective mean and the standard deviation of the trivial model prediction.

B.4 Wasserstein Distance

The p-th Wasserstein distance [124] is defined by

$$W_p(\mu,\nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} d(x,y)^p d\gamma(x,y) \right)^{\frac{1}{p}}, \tag{B.4}$$

where $\Gamma(\mu,\nu)$ denotes the collection of all measures on $M \times M$ with marginals μ and ν. The 2-Wasserstein distance between two multivariate Gaussians can be obtained analytically and is defined by

$$W_2(\mu,\nu)^2 = ||\mathbf{m}_1 - \mathbf{m}_2||^2 + \mathrm{Tr}\left[\mathbf{K}_1 + \mathbf{K}_2 - 2\left(\mathbf{K}_2^{\frac{1}{2}} \mathbf{K}_1 \mathbf{K}_2^{\frac{1}{2}} \right)^{\frac{1}{2}} \right], \tag{B.5}$$

where $\mu \sim \mathcal{N}(\mu|\mathbf{m}_1, \mathbf{K}_1)$, $\nu \sim \mathcal{N}(\nu|\mathbf{m}_2, \mathbf{K}_2)$, and $\mathbf{K} = \mathbf{K}^{\frac{1}{2}}\mathbf{K}^{\frac{1}{2}}$.

Appendix C

Covariance Functions

C.1 Constant Covariance Function

A constant covariance function is defined by

$$k_{\text{Constant}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2, \tag{C.1}$$

where σ_f denotes a scaling factor.

C.2 Linear Covariance Function

A linear covariance function is defined by

$$k_{\text{Linear}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \mathbf{x}_i^T \mathbf{x}_j, \tag{C.2}$$

where σ_f denotes a scaling factor.

C.3 Radial Basis Covariance Function

A radial basis covariance (RBF) function is defined by

$$k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 e^{-\frac{1}{2}r^2}, \tag{C.3}$$

where $r = \frac{1}{l}||\mathbf{x}_i - \mathbf{x}_j||$ and l is a lengthscale parameter. Automatic Relevance Determination (ARD) covariance function weights the columns of the input $\mathbf{x}$ with their corresponding lengthscale l_d. It is defined by Equation (C.3) and

$$r = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{\Lambda}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}, \tag{C.4}$$

where $\mathbf{\Lambda}^{-1} = \text{diag}([l_1^{-2}, \ldots, l_d^{-2}])$ and d is the number of columns in $\mathbf{x}$.

C.4 Matérn Covariance Functions

Matérn covariance functions are defined by

$$k_{\text{Matérn}(\frac{1}{2})}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 e^{-r}, \tag{C.5a}$$

$$k_{\text{Matérn}(\frac{3}{2})}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 (1 + \sqrt{3}r) e^{-\sqrt{3}r}, \tag{C.5b}$$

$$k_{\text{Matérn}(\frac{5}{2})}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 (1 + \sqrt{5}r + \frac{5}{3}r^2) e^{-\sqrt{5}r}, \tag{C.5c}$$

where $r = \frac{1}{l}||\mathbf{x}_i - \mathbf{x}_j||$. Matérn covariance functions with the ARD property are defined by Equations (C.4) and (C.5).

C.5 Periodic Covariance Function

Following [125], the periodic kernel can be obtained by mapping the inputs through the transformation

$$u(x) = (\cos(2\pi\gamma^{-1}x), \sin(2\pi\gamma^{-1}x)). \tag{C.6}$$

Then

$$k_{\text{PER, k}}(\mathbf{x}_i, \mathbf{x}_j; \gamma) = k(u(\mathbf{x}_i), u(\mathbf{x}_j); \gamma). \tag{C.7}$$

The explicit form for the RBF kernel is

$$\begin{aligned} k_{\text{PER, RBF}}(\mathbf{x}_i, \mathbf{x}_j; \gamma) &= k_{\text{RBF}}(u(\mathbf{x}_i), u(\mathbf{x}_j); \gamma) \\ &= \sigma_f^2 e^{-\frac{1}{2}\sin^2(\pi r\gamma^{-1})l^{-2}}, \end{aligned} \tag{C.8}$$

where $r = ||\mathbf{x}_i - \mathbf{x}_j||$ and γ defines the period. The kernel can be trivially extended with the ARD property.

C.6 Change Point Covariance Function

Following [122], a change point kernel is defined weighting the base kernels with sigmoid functions. A single change point kernel is defined by

$$\begin{aligned} k_{\text{CP},k_1,k_2}(t_i, t_j, \mathbf{x}_i, \mathbf{x}_j) &= \alpha(t_i, t_j)k_1(\mathbf{x}_i, \mathbf{x}_j) + \beta(t_i, t_j)k_2(\mathbf{x}_i, \mathbf{x}_j), \\ \alpha(t_i, t_j) &= (1 - \sigma(t_i; s, c_0))(1 - \sigma(t_j; s, c_0)), \\ \beta(t_i, t_j) &= \sigma(t_i; s, c_0)\sigma(t_j; s, c_0). \end{aligned} \tag{C.9}$$

The sigmoid is a logistic function defined by

$$\sigma(x) = (1 + e^{-s(x-c_0)})^{-1}, \tag{C.10}$$

where s denotes the steepness parameter and c_0 the location. The change point kernel, generalized for multiple locations $\mathbf{c}^T = [c_0, c_1, \ldots, c_p]$, is defined by

$$\begin{aligned} k_{\text{CP},k_1,k_2}(t_i, t_j, \mathbf{x}_i, \mathbf{x}_j) &= \alpha(t_i, t_j)k_1(\mathbf{x}_i, \mathbf{x}_j) + \beta(t_i, t_j)k_2(\mathbf{x}_i, \mathbf{x}_j), \\ \alpha(t_i, t_j) &= (1 - \eta(t_i; s, \mathbf{c}))(1 - \eta(t_j; s, \mathbf{c})), \\ \beta(t_i, t_j) &= \eta(t_i; s, \mathbf{c})\eta(t_j; s, \mathbf{c}), \end{aligned} \tag{C.11}$$

where

$$\eta(x) = \sum_{i=0}^{\lfloor p/2 \rfloor} \sigma(x; s, c_{2i})(1 - \sigma(x; s, c_{2i+1})). \tag{C.12}$$

Appendix D

Variational Inference

D.1 KL Divergence

For probability density functions p and q, defined on the same probability space, the Kullback–Leibler (KL) divergence is defined by

$$\mathrm{KL}\left[p||q\right] = \int_{\infty}^{\infty} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} dx. \tag{D.1}$$

If $p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_1, \mathbf{K}_1\right)$ and $q(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_2, \mathbf{K}_2\right)$, then the KL divergence is explicitly defined by

$$\mathrm{KL}\left[p||q\right] = \frac{1}{2}\left(\mathrm{tr}\left[\mathbf{K}_2^{-1}\mathbf{K}_1\right] + \left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right)^T \mathbf{K}_2^{-1}\left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right) + \log\frac{|\mathbf{K}_2|}{|\mathbf{K}_1|} - k\right), \tag{D.2}$$

where k denotes the dimensionality of $\mathbf{x}$.

D.2 Jensen Inequality

If X is a random variable and ϕ is a convex function, then

$$\phi\left(\mathbb{E}\left[X\right]\right) \leq \mathbb{E}\left[\phi\left(X\right)\right]. \tag{D.3}$$

For $\phi(\cdot) = \log(\cdot)$

$$\log\left(\mathbb{E}\left[X\right]\right) \geq \mathbb{E}\left[\log\left(X\right)\right]. \tag{D.4}$$

D.3 Evidence Lower Bound

The dynamic notation of indexing the latent values by time steps can be for static problems equivalently replaced by a more common notation in the GP literature

$$\begin{aligned} \mathbf{f}_{1:t} &\mapsto \mathbf{f}, \\ \mathbf{f}_{t+1:t+n} &\mapsto \mathbf{f}_*, \\ f\left(\cdot\right) &\mapsto f\left(\cdot\right). \end{aligned} \tag{D.5}$$

Let us consider a joint distribution $p\left(\mathbf{y}_{1:t}, \mathbf{f}_{1:t}, f\left(\cdot\right), \mathbf{u}\right)$, approximated by the conditional independence assumption between the latent function values that belong to the training data set and the latent function values that belong to unobserved values,

$$p\left(\mathbf{y}_{1:t}, \mathbf{f}_{1:t}, f\left(\cdot\right), \mathbf{u}\right) \approx p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) p(\mathbf{f}_{1:t}|\mathbf{u}) p\left(f\left(\cdot\right)|\mathbf{u}\right) q(\mathbf{u}), \tag{D.6}$$

where $q(\mathbf{u})$ is a free distribution, not constrained to any parametric form. Marginalizing out all latent function values that do not belong to the training data yields

$$\begin{aligned} p\left(\mathbf{y}_{1:t}, \mathbf{f}_{1:t}, \mathbf{u}\right) &\approx \int p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) p(\mathbf{f}_{1:t}|\mathbf{u}) p\left(f\left(\cdot\right)|\mathbf{u}\right) q(\mathbf{u}) d\mathbf{f}_{\setminus\{\mathbf{f}_{1:t}\}} \\ &\approx p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) p(\mathbf{f}_{1:t}|\mathbf{u}) q(\mathbf{u}). \end{aligned} \tag{D.7}$$

The MLL is then defined by

$$\log p(\mathbf{y}_{1:t}) = \log \int\int p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t}) p(\mathbf{y}_{1:t}) d\mathbf{f}_{1:t} d\mathbf{u}, \tag{D.8a}$$

$$\log p(\mathbf{y}_{1:t}) = \log \int\int \frac{q(\mathbf{f}_{1:t}, \mathbf{u})}{q(\mathbf{f}_{1:t}, \mathbf{u})} p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t}) p(\mathbf{y}_{1:t}) d\mathbf{f}_{1:t} d\mathbf{u}, \tag{D.8b}$$

$$\log p(\mathbf{y}_{1:t}) \geq \int\int q(\mathbf{f}_{1:t}, \mathbf{u}) \log \frac{p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t}) p(\mathbf{y}_{1:t})}{q(\mathbf{f}_{1:t}, \mathbf{u})} d\mathbf{f}_{1:t} d\mathbf{u}, \tag{D.8c}$$

$$\log p(\mathbf{y}_{1:t}) \geq \log p(\mathbf{y}_{1:t}) - \mathrm{KL}[q(\mathbf{f}_{1:t}, \mathbf{u}) || p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})], \tag{D.8d}$$

where the step from Equation (D.8b) to Equation (D.8c) applies the Jensen inequality defined by Equation (D.4). The KL divergence in Equation (D.8d) can be further simplified to

$$\begin{aligned} &\mathrm{KL}[q(\mathbf{f}_{1:t}, \mathbf{u}) || p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})] = \\ &\qquad = \int\int q(\mathbf{f}_{1:t}, \mathbf{u}) \log \frac{\cancel{p(\mathbf{f}_{1:t}|\mathbf{u})} q(\mathbf{u}) p(\mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) \cancel{p(\mathbf{f}_{1:t}|\mathbf{u})} p(\mathbf{u})} d\mathbf{f}_{1:t} d\mathbf{u}. \end{aligned} \tag{D.9}$$

Taking $p(\mathbf{y}_{1:t})$ out of the expression yields

$$\begin{aligned} &\mathrm{KL}[q(\mathbf{f}_{1:t}, \mathbf{u}) || p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})] = \\ &= p(\mathbf{y}_{1:t}) - \underbrace{\left(\int\int \left[\log p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})\right] q(\mathbf{f}_{1:t}, \mathbf{u}) d\mathbf{f}_{1:t} d\mathbf{u} - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \right)}_{\text{ELBO}}, \end{aligned} \tag{D.10}$$

and finally

$$\begin{aligned} &\mathrm{KL}[q(\mathbf{f}_{1:t}, \mathbf{u}) || p(\mathbf{f}_{1:t}, \mathbf{u}|\mathbf{y}_{1:t})] = \\ &\qquad = p(\mathbf{y}_{1:t}) - \underbrace{\left(\mathbb{E}_{q(\mathbf{f}_{1:t})}[\log p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \right)}_{\text{ELBO}}, \end{aligned} \tag{D.11}$$

where we can see the equivalence between the minimization of the KL divergence (between the true and the approximated posterior) and maximization of the ELBO [38], [39], [41]. We again explicitly define the ELBO by

$$F\left(q, \mathbf{z}'_{1:m}\right) = \mathbb{E}_{q(\mathbf{f}_{1:t})}[\log p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})], \tag{D.12}$$

which will serve as a reference in the next section.

D.4 Collapsed Evidence Lower Bound

The derivation is reproduced from [126] where some additional information is added in between the steps of the original derivation. For generality let us assume the likelihood $p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}) \sim \mathcal{N}(\mathbf{y}_{1:t}|\mathbf{f}_{1:t}, \mathbf{\Lambda}_{1:t})$ where

$$\mathbf{\Lambda}_{1:t} = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_t^2 \end{bmatrix} \tag{D.13}$$

represents the heteroskedastic noise. If we again consider the ELBO defined in Equation (D.12) and explicitly write out the KL divergence and the expectations by their definition, we get

$$F\left(q, \mathbf{z}'_{1:m}\right) = \int\int \left[\log\, p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})\right] q(\mathbf{f}_{1:t}, \mathbf{u}) d\mathbf{f}_{1:t} d\mathbf{u} + \int q(\mathbf{u})\, \log\, \frac{p(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u} \tag{D.14a}$$

$$= \int q(\mathbf{u}) \left(\underbrace{\int \left[\log\, p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})\right] p(\mathbf{f}_{1:t}|\mathbf{u}) d\mathbf{f}_{1:t}}_{\log\, G(\mathbf{u}, \mathbf{y}_{1:t})} + \log\, \frac{p(\mathbf{u})}{q(\mathbf{u})} \right) d\mathbf{u}. \tag{D.14b}$$

The inner integral can be solved analytically

$$\log\, G\left(\mathbf{u}, \mathbf{y}_{1:t}\right) = \int \left[\log\, p(\mathbf{y}_{1:t}|\mathbf{f}_{1:t})\right] p(\mathbf{f}_{1:t}|\mathbf{u}) d\mathbf{f}_{1:t} = \tag{D.15a}$$

$$= \int \left[\text{ const. } -\frac{1}{2}\text{Tr}\left[\mathbf{\Lambda}_{1:t}^{-1}\left(\mathbf{y}_{1:t}\mathbf{y}_{1:t}^T - 2\mathbf{y}_{1:t}\mathbf{f}_{1:t}^T + \mathbf{f}_{1:t}\mathbf{f}_{1:t}^T\right)\right]\right] p(\mathbf{f}_{1:t}|\mathbf{u}) d\mathbf{f}_{1:t} \tag{D.15b}$$

$$= \text{ const. } -\frac{1}{2}\text{Tr}\left[\mathbf{\Lambda}_{1:t}^{-1}\left(\mathbf{y}_{1:t}\mathbf{y}_{1:t}^T - 2\mathbf{y}_{1:t}\boldsymbol{\alpha}_{1:t}^T + \boldsymbol{\alpha}_{1:t}\boldsymbol{\alpha}_{1:t}^T + \mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}\right)\right], \tag{D.15c}$$

where

$$\begin{aligned} \text{const.} &= -\frac{n}{2}\log\,(2\pi) - \frac{1}{2}\log\,|\mathbf{\Lambda}_{1:t}| \\ \boldsymbol{\alpha}_{1:t} &= \mathbb{E}\left[\mathbf{f}_{1:t}|\mathbf{u}\right] = \mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{u} \\ \mathbf{Q}_{f_{1:t},f_{1:t}} &= \mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{1:t}} \end{aligned} \tag{D.16}$$

The expectation of a Gaussian over a squared form $\int \left[\mathbf{f}_{1:t}\mathbf{f}_{1:t}^T\right] p(\mathbf{f}_{1:t}|\mathbf{u}) d\mathbf{f}_{1:t}$ is defined by Equation (A.9a). The expression from Equation (D.15) then simplifies to

$$\log\, G\left(\mathbf{u}, \mathbf{y}_{1:t}\right) = \log\, \left[\mathcal{N}(\mathbf{y}_{1:t}|\boldsymbol{\alpha}_{1:t}, \mathbf{\Lambda}_{1:t})\right] - \frac{1}{2}\text{Tr}\left[\mathbf{\Lambda}_{1:t}^{-1}\left(\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}\right)\right]. \tag{D.17}$$

ELBO can now be rewritten to

$$F\left(q, \mathbf{z}'_{1:m}\right) = \int q(\mathbf{u}) \log\, \frac{\mathcal{N}(\mathbf{y}_{1:t}|\boldsymbol{\alpha}_{1:t}, \mathbf{\Lambda}_{1:t}) p(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u} - \frac{1}{2}\text{Tr}\left[\mathbf{\Lambda}_{1:t}^{-1}\left(\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}\right)\right]. \tag{D.18}$$

The optimal variational distribution is determined by maximizing Equation (D.18) with respect to the variational distribution $q(\mathbf{u})$, and constraining the variational distribution to $\int q(\mathbf{u})d\mathbf{u} = 1$. The optimization problem is then defined by

$$\begin{aligned} &\frac{\partial \left[F\left(q, \mathbf{z}'_{1:m}\right) + \lambda(\int q(\mathbf{u})d\mathbf{u} - 1)\right]}{\partial q(\mathbf{u})} = 0 \\ =&\underbrace{\frac{\partial \left[F\left(q, \mathbf{z}'_{1:m}\right) + \lambda(\int q(\mathbf{u})d\mathbf{u} - 1)\right]}{\partial \mathbf{u}}}_{=0} \left(\frac{\partial q(\mathbf{u})}{\partial \mathbf{u}}\right)^{-1} = 0 \end{aligned} \tag{D.19}$$

If we consider

$$\begin{aligned} \frac{\partial \left[F\left(q, \mathbf{z}'_{1:m}\right) + \lambda(\int q(\mathbf{u})d\mathbf{u} - 1)\right]}{\partial \mathbf{u}} &= 0 \\ \frac{\partial \left[\int q(\mathbf{u}) \log \frac{\mathcal{N}(\mathbf{y}_{1:t}|\boldsymbol{\alpha}_{1:t}, \boldsymbol{\Lambda}_{1:t})p(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u} + \lambda \int q(\mathbf{u})d\mathbf{u} + \text{cost.}\right]}{\partial \mathbf{u}} &= 0 \\ q(\mathbf{u}) \left(\log \frac{\mathcal{N}(\mathbf{y}_{1:t}|\boldsymbol{\alpha}_{1:t}, \boldsymbol{\Lambda}_{1:t})p(\mathbf{u})}{q(\mathbf{u})} + \lambda\right) &= 0 \end{aligned} \tag{D.20}$$

The optimal variational distribution is then defined by

$$q(\mathbf{u}) = Z \cdot \mathcal{N}(\mathbf{y}_{1:t}|\boldsymbol{\alpha}_{1:t}, \boldsymbol{\Lambda}_{1:t})p(\mathbf{u}), \tag{D.21}$$

where Z is a normalizing constant and $\lambda = \log Z$. Only at this step we see that the optimal variation distribution is Gaussian. Explicitly, the free variational distribution is specified by

$$\begin{aligned} q(\mathbf{u}) =& Z \cdot \mathcal{N}(\mathbf{y}_{1:t}|\boldsymbol{\alpha}_{1:t}, \boldsymbol{\Lambda}_{1:t})p(\mathbf{u}) = \\ = Z \cdot \exp&\left(-\frac{1}{2}\mathbf{u}^T(\mathbf{K}_{u,u}^{-1} + \mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}_{1:t}^{-1}\mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1})\mathbf{u} + \mathbf{y}_{1:t}^T\boldsymbol{\Lambda}_{1:t}^{-1}\mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{u}\right), \end{aligned} \tag{D.22}$$

which simplifies to

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{K}_{u,u}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}_{1:t}^{-1}\mathbf{y}_{1:t}, \mathbf{K}_{u,u}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{u,u}), \tag{D.23}$$

where $\boldsymbol{\Sigma} = \mathbf{K}_{u,u} + \mathbf{K}_{f_{1:t},u}\boldsymbol{\Lambda}_{1:t}^{-1}\mathbf{K}_{u,f_{1:t}}$. The collapsed ELBO is in this case defined by

$$F(\mathbf{z}'_{1:m}) = \log\ \left[\mathcal{N}\left(\mathbf{y}_{1:t}|\mathbf{0}, \mathbf{Q}_{f_{1:t},f_{1:t}} + \boldsymbol{\Lambda}_{1:t}\right)\right] - \frac{1}{2}\text{Tr}\left[\boldsymbol{\Lambda}_{1:t}^{-1}\left(\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}\right)\right]. \tag{D.24}$$

In the special case where the likelihood noise is homoskedastic, i.e. $\boldsymbol{\Lambda}_{1:t} = \mathbf{I}\sigma_n^2$, the free variational distribution simplifies to

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\sigma_n^{-2}\mathbf{K}_{u,u}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t}, \mathbf{K}_{u,u}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{u,u}), \tag{D.25}$$

where $\boldsymbol{\Sigma} = \mathbf{K}_{u,u} + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}$. The collapsed ELBO for the homoskedastic case is then defined by

$$F(\mathbf{z}'_{1:m}) = \log\ \left[\mathcal{N}\left(\mathbf{y}_{1:t}|\mathbf{0}, \mathbf{Q}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)\right] - \frac{1}{2\sigma_n^2}\text{Tr}\left[\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{Q}_{f_{1:t},f_{1:t}}\right]. \tag{D.26}$$

Appendix E

Posteriors in GP-NARX Models

In this section of the Appendix, we will consider numerically practical definitions of the predictive and pseudo-point posteriors that are required for implementing the simulation of GP-NARX models in practice. The noisy prediction only adds the σ_n^2 to the variance of the latent prediction. Most of the definitions are taken from the software implementations in the GPflow toolbox [91].

E.1 Predictive Posteriors

E.1.1 GP-NARX

The mean of the predictive posterior is defined by

$$\begin{aligned}\mathbb{E}[f_{t+1}|\mathbf{y}_{1:t}] &= \mathbf{K}_{f_{t+1},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}}+\mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}_{1:t}\\ &= \mathbf{K}_{f_{t+1},f_{1:t}}\mathbf{L}_{1:t}^{-T}\mathbf{L}_{1:t}^{-1}\mathbf{y}_{1:t}.\end{aligned} \tag{E.1}$$

The variance of the predictive posterior is defined by

$$\begin{aligned}\mathbb{V}[f_{t+1}|\mathbf{y}_{1:t}] &= \mathbf{K}_{f_{t+1},f_{t+1}}-\mathbf{K}_{f_{t+1},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}}+\mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f_{1:t},f_{t+1}}\\ &= \mathbf{K}_{f_{t+1},f_{t+1}}-\mathbf{K}_{f_{t+1},f_{1:t}}\mathbf{L}_{1:t}^{-T}\mathbf{L}_{1:t}^{-1}\mathbf{K}_{f_{1:t},f_{t+1}},\end{aligned} \tag{E.2}$$

where

$$\mathbf{L}_{1:t}\mathbf{L}_{1:t}^{T} = \text{cholesky}\left(\mathbf{K}_{f_{1:t},f_{1:t}}+\mathbf{I}\sigma_n^2\right). \tag{E.3}$$

E.1.2 GP-NARX (FITC)

The mean of the predictive posterior is defined by

$$\begin{aligned}\mathbb{E}[f_{t+1}|\mathbf{y}_{1:t}] &= \mathbf{K}_{f_{t+1},u}\left(\mathbf{K}_{u,u}+\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t}\\ &= \mathbf{K}_{f_{t+1},u}\left(\mathbf{L}_u\mathbf{L}_u^{T}+\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t}\\ &= \mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\left(\mathbf{I}+\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}\right)^{-1}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t}\\ &= \mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t}.\end{aligned} \tag{E.4}$$

The variance of the predictive posterior is defined by

$$\begin{aligned}\mathbb{V}[f_{t+1}|\mathbf{y}_{1:t}] &= \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{t+1}} \\ &\quad + \mathbf{K}_{f_{t+1},u}\left(\mathbf{K}_{u,u} + \mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{t+1}} \\ &= \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\left(\mathbf{I} - \mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\right)\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{t+1}},\end{aligned} \tag{E.5}$$

where

$$\begin{aligned}\mathbf{L}_u\mathbf{L}_u^T &= \text{cholesky}\left(\mathbf{K}_{u,u}\right), \\ \mathbf{L}_B\mathbf{L}_B^T &= \text{cholesky}\left(\mathbf{I} + \mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}\right), \\ \mathbf{\Lambda} &= \text{diag}\left[\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{1:t}} + \mathbf{I}\sigma_n^2\right].\end{aligned} \tag{E.6}$$

E.1.3 GP-NARX (VFE)

The mean of the predictive posterior is defined by

$$\begin{aligned}\mathbb{E}[f_{t+1}] &= \sigma_n^{-2}\mathbf{K}_{f_{t+1},u}\left(\mathbf{K}_{u,u} + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t} \\ &= \sigma_n^{-2}\mathbf{K}_{f_{t+1},u}\left(\mathbf{L}_u\mathbf{L}_u^T + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t} \\ &= \sigma_n^{-2}\mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\left(\mathbf{I} + \sigma_n^{-2}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}\right)^{-1}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t} \\ &= \sigma_n^{-2}\mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t}.\end{aligned} \tag{E.7}$$

The variance of the predictive posterior is defined by

$$\begin{aligned}\mathbb{V}[f_{t+1}] &= \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{t+1}} \\ &\quad + \mathbf{K}_{f_{t+1},u}\left(\mathbf{K}_{u,u} + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{t+1}} \\ &= \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\left(\mathbf{I} - \mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\right)\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{t+1}},\end{aligned} \tag{E.8}$$

where

$$\begin{aligned}\mathbf{L}_u\mathbf{L}_u^T &= \text{cholesky}\left(\mathbf{K}_{u,u}\right), \\ \mathbf{L}_B\mathbf{L}_B^T &= \text{cholesky}\left(\mathbf{I} + \sigma_n^{-2}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}\right).\end{aligned} \tag{E.9}$$

E.1.4 GP-NARX (SVGP)

For the whitened representation of the inducing point posterior as defined by Equation (3.24), the mean of the predictive posterior is defined by

$$\mathbb{E}[f_{t+1}] = \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{m}, = \mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\hat{\mathbf{m}} \tag{E.10}$$

The variance of the predictive posterior is then defined by

$$\begin{aligned}\mathbb{V}[f_{t+1}] &= \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{t+1}} + \mathbf{K}_{f_{t+1},u}\mathbf{K}_{u,u}^{-1}\mathbf{\Phi}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{t+1}} \\ &= \mathbf{K}_{f_{t+1},f_{t+1}} - \mathbf{K}_{f_{t+1},u}\mathbf{L}_u^{-T}\left(\mathbf{I} - \hat{\mathbf{L}}_\phi\hat{\mathbf{L}}_\phi^T\right)\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{t+1}},\end{aligned} \tag{E.11}$$

where

$$\mathbf{L}_u\mathbf{L}_u^T = \text{cholesky}\left(\mathbf{K}_{u,u}\right) \tag{E.12}$$

and $\hat{\mathbf{m}}, \hat{\mathbf{L}}_\phi$ represent the whitened parameters, determined by optimization, and otherwise defined

$$\begin{aligned} \hat{\mathbf{m}} &= \mathbf{L}_u^{-1}\mathbf{m}, \\ \hat{\mathbf{L}}_\phi &= \mathbf{L}_u^{-1}\mathbf{L}_\phi^{-1}, \end{aligned} \tag{E.13}$$

where $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{\Phi}) = \mathcal{N}\left(\mathbf{u}|\mathbf{m}, \mathbf{L}_\phi\mathbf{L}_\phi^T\right)$.

E.1.5 HGP-NARX

The mean of the noise corrupted predictive posterior is defined by

$$\begin{aligned} \mathbb{E}[y_{t+1}] &= \int y_{t+1}\mathbb{E}_{q(f_{t+1},g_{t+1})}\left[p(y_{t+1}|f_{t+1},g_{t+1})\right]dy_{t+1} \\ &= \mathbb{E}_{q(f_{t+1},g_{t+1})}\left[\int y_{t+1}p(y_{t+1}|f_{t+1},g_{t+1})dy_{t+1}\right] \\ &= \mathbb{E}_{q(f_{t+1},g_{t+1})}\left[f_{t+1}\right] \\ &= \mu_{f_{t+1}}. \end{aligned} \tag{E.14}$$

The variance of the noise corrupted predictive posterior is defined by

$$\begin{aligned} \mathbb{V}[y_{t+1}] &= \mathbb{E}[y_{t+1}^2] - \mathbb{E}[y_{t+1}]^2 \\ &= \mathbb{E}[y_{t+1}^2] - \mu_{f_{t+1}}^2 \\ &= \mathbb{E}_{q(g_{t+1})}\left[h(g_{t+1})^2\right] + \sigma_{f_{t+1}}^2, \end{aligned} \tag{E.15}$$

where the last step follows from

$$\begin{aligned} \mathbb{E}[y_{t+1}^2] &= \int y_{t+1}^2\mathbb{E}_{q(f_{t+1},g_{t+1})}\left[p(y_{t+1}|f_{t+1},g_{t+1})\right]dy_{t+1} \\ &= \mathbb{E}_{q(f_{t+1},g_{t+1})}\left[\int y_{t+1}^2p(y_{t+1}|f_{t+1},g_{t+1})dy_{t+1}\right] \\ &= \mathbb{E}_{q(f_{t+1},g_{t+1})}\left[\mathbb{V}[y_{t+1}|f_{t+1},g_{t+1}] + \mathbb{E}[y_{t+1}|f_{t+1},g_{t+1}]^2\right] \\ &= \mathbb{E}_{q(f_{t+1},g_{t+1})}\left[h(g_{t+1})^2 + f_{t+1}^2\right] \\ &= \mathbb{E}_{q(g_{t+1})}\left[h(g_{t+1})^2\right] + \sigma_{f_{t+1}}^2 + \mu_{f_{t+1}}^2. \end{aligned} \tag{E.16}$$

The latent posterior mean $\mu_{f_{t+1}}$ and latent posterior variance $\sigma_{f_{t+1}}^2$ are defined by Equation (E.10) and Equation (E.11). Similarly, the posterior of $g_{t+1} \sim \mathcal{N}\left(g_{t+1}|\mu_{g_{t+1}}, \sigma_{g_{t+1}}^2\right)$ is specified. The posteriors of the pseudo-points are defined in Section E.2.4.

E.2 Pseudo-Point Posterior

E.2.1 GP-NARX

In the GP-NARX model, the pseudo-point posterior is replaced with the predictive posterior at the training point locations, i.e. $q(\mathbf{u}) = p(\mathbf{f}_{1:t}|\mathbf{y}_{1:t})$. The mean and variance are defined by

$$\begin{aligned} \mathbb{E}[\mathbf{f}_{1:t}|\mathbf{y}_{1:t}] &= \mathbf{K}_{f_{1:t},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{y}_{1:t} \\ &= \mathbf{K}_{f_{1:t},f_{1:t}}\mathbf{L}_{1:t}^{-T}\mathbf{L}_{1:t}^{-1}\mathbf{y}_{1:t}. \end{aligned} \tag{E.17}$$

The variance of the predictive posterior is defined by

$$\begin{aligned}\mathbb{V}[\mathbf{f}_{1:t}|\mathbf{y}_{1:t}] &= \mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right)^{-1}\mathbf{K}_{f_{1:t},f_{1:t}} \\ &= \mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},f_{1:t}}\mathbf{L}_{1:t}^{-T}\mathbf{L}_{1:t}^{-1}\mathbf{K}_{f_{1:t},f_{1:t}},\end{aligned} \tag{E.18}$$

where

$$\mathbf{L}_{1:t}\mathbf{L}_{1:t}^T = \text{cholesky}\left(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2\right). \tag{E.19}$$

E.2.2 GP-NARX (FITC)

The mean of the pseudo-point posterior, i.e. $q(\mathbf{u}|\mathbf{y}_{1:t})$, is defined by

$$\begin{aligned}\mathbb{E}[\mathbf{u}|\mathbf{y}_{1:t}] &= \mathbf{K}_{u,u}\left(\mathbf{K}_{u,u} + \mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t} \\ &= \mathbf{L}_u\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{y}_{1:t}.\end{aligned} \tag{E.20}$$

The variance of the pseudo-point posterior is defined by

$$\begin{aligned}\mathbb{V}[\mathbf{u}|\mathbf{y}_{1:t}] &= \mathbf{K}_{u,u}\left(\mathbf{K}_{u,u} + \mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,u} \\ &= \mathbf{L}_u\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^T,\end{aligned} \tag{E.21}$$

where

$$\begin{aligned}\mathbf{L}_u\mathbf{L}_u^T &= \text{cholesky}\left(\mathbf{K}_{u,u}\right), \\ \mathbf{L}_B\mathbf{L}_B^T &= \text{cholesky}\left(\mathbf{I} + \mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}\right), \\ \mathbf{\Lambda} &= \text{diag}\left[\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,f_{1:t}} + \mathbf{I}\sigma_n^2\right].\end{aligned} \tag{E.22}$$

E.2.3 GP-NARX (VFE)

The mean of the pseudo-point posterior, i.e. $q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})$, is defined by

$$\begin{aligned}\mathbb{E}[\mathbf{u}] &= \sigma_n^{-2}\mathbf{K}_{u,u}\left(\mathbf{K}_{u,u} + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t} \\ &= \mathbf{L}_u\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^{-1}\sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{y}_{1:t}.\end{aligned} \tag{E.23}$$

The variance of the pseudo-point posterior is defined by

$$\begin{aligned}\mathbb{V}[\mathbf{u}] &= \mathbf{K}_{u,u}\left(\mathbf{K}_{u,u} + \sigma_n^{-2}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\right)^{-1}\mathbf{K}_{u,u} \\ &= \mathbf{L}_u\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^T,\end{aligned} \tag{E.24}$$

where

$$\begin{aligned}\mathbf{L}_u\mathbf{L}_u^T &= \text{cholesky}\left(\mathbf{K}_{u,u}\right), \\ \mathbf{L}_B\mathbf{L}_B^T &= \text{cholesky}\left(\mathbf{I} + \sigma_n^{-2}\mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}\right).\end{aligned} \tag{E.25}$$

E.2.4 GP-NARX (SVGP)

For the whitened representation, the mean of the pseudo-point posterior, i.e. $q(\mathbf{u}) \approx p(\mathbf{u}|\mathbf{y}_{1:t})$, is defined by

$$\begin{aligned} \mathbb{E}[\mathbf{u}] &= \mathbf{m} \\ &= \mathbf{L}_u^{-1}\hat{\mathbf{m}}. \end{aligned} \tag{E.26}$$

The variance is then defined by

$$\begin{aligned} \mathbb{V}[\mathbf{u}] &= \mathbf{\Phi} \\ &= \mathbf{L}_u^{-1}\hat{\mathbf{L}}_\phi\hat{\mathbf{L}}_\phi^T\mathbf{L}_u^{-T}, \end{aligned} \tag{E.27}$$

where

$$\mathbf{L}_u\mathbf{L}_u^T = \text{cholesky}\left(\mathbf{K}_{u,u}\right) \tag{E.28}$$

and $\hat{\mathbf{m}}, \hat{\mathbf{L}}_\phi$ represent the whitened parameters as defined in Equation (E.13).

Appendix F

Simulation Algorithms

F.1 CIMC Simulation

F.1.1 GP-NARX

Algorithm F.1: Algorithm for a single sample in the CIMC simulation of the vanilla GP-NARX model. NP denotes noise propagation.

Data: $\mathcal{D} = \{\mathbf{Z}_{1:t}, \mathbf{y}_{1:t}\}$: training data
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+2-n_a:t+1} \sim p(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{y}_{1:t})$;
$\tilde{y}_{t+1} \sim p(y_{t+1}|f_{t+1})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\tilde{f}_{t+1}]^T$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\tilde{y}_{t+1}]^T$;
$i \leftarrow t+2$;
$\mathbf{L}_{1:t}\mathbf{L}_{1:t}^T \leftarrow \text{cholesky}(\mathbf{K}_{f_{1:t},f_{1:t}} + \mathbf{I}\sigma_n^2)$;
$\mathbf{L}_{1:t}^{-1} \leftarrow \mathbf{L}_{1:t} \backslash \mathbf{I}$;
$\boldsymbol{\alpha} \leftarrow \mathbf{L}_{1:t}^{-T}\mathbf{L}_{1:t}^{-1}$;
$\boldsymbol{\beta} \leftarrow \boldsymbol{\alpha}\mathbf{y}_{1:t}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\mu \leftarrow \mathbf{K}_{f_i,f_{1:t}}\boldsymbol{\beta}$;
  $\sigma^2 \leftarrow \mathbf{K}_{f_i,f_i} - \mathbf{K}_{f_i,f_{1:t}}\boldsymbol{\alpha}\mathbf{K}_{f_{1:t},f_i}$;
  $\tilde{f}_i \sim p(f_i|\mathcal{D}, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu, \sigma^2)$;
  $\tilde{y}_i \sim p(y_i|f_i)$;
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.1.2 GP-NARX (FITC)

Algorithm F.2: Algorithm for a single sample in the CIMC simulation of the GP-NARX (FITC) model. NP denotes noise propagation.

Data: $\mathcal{D} = \{\mathbf{Z}_{1:t}, \mathbf{y}_{1:t}\}$: training data
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+2-n_a:t+1} \sim q(\mathbf{f}_{t+2-n_a:t+1}|\mathbf{y}_{1:t})$;
$\tilde{y}_{t+1} \sim p(y_{t+1}|f_{t+1})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\tilde{f}_{t+1}]^T$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\tilde{y}_{t+1}]^T$;
$i \leftarrow t+2$;
$\mathbf{L}_u\mathbf{L}_u^T \leftarrow \text{cholesky}(\mathbf{K}_{u,u})$;
$\mathbf{L}_u^{-1} \leftarrow \mathbf{L}_u \backslash \mathbf{I}$;
$\boldsymbol{\alpha} \leftarrow \mathbf{L}_u^{-T}\mathbf{L}_u^{-1}$;
$\boldsymbol{\Lambda} \leftarrow \text{diag}[\mathbf{K}_{f_{1:t},f_{1:t}} - \mathbf{K}_{f_{1:t},u}\boldsymbol{\alpha}\mathbf{K}_{u,f_{1:t}} + \mathbf{I}\sigma_n^2]$;
$\mathbf{B} = \mathbf{I} + \mathbf{L}_u^{-1}\mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}^{-1}\mathbf{K}_{f_{1:t},u}\mathbf{L}_u^{-T}$;
$\mathbf{L}_B\mathbf{L}_B^T \leftarrow \text{cholesky}(\mathbf{B})$;
$\mathbf{L}_B^{-1} \leftarrow \mathbf{L}_B \backslash \mathbf{I}$;
$\boldsymbol{\beta} \leftarrow \mathbf{L}_u^{-T}\mathbf{L}_B^{-T}\mathbf{L}_B^{-1}\mathbf{L}_u^{-1}$;
$\boldsymbol{\gamma} \leftarrow \boldsymbol{\beta}\mathbf{K}_{u,f_{1:t}}\boldsymbol{\Lambda}^{-1}\mathbf{y}_{1:t}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\mu \leftarrow \mathbf{K}_{f_i,u}\boldsymbol{\gamma}$;
  $\sigma^2 \leftarrow \mathbf{K}_{f_i,f_i} - \mathbf{K}_{f_i,u}\boldsymbol{\alpha}\mathbf{K}_{u,f_i} + \mathbf{K}_{f_i,u}\boldsymbol{\beta}\mathbf{K}_{u,f_i}$;
  $\tilde{f}_i \sim p(f_i|\mathcal{D}, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu, \sigma^2)$;
  $\tilde{y}_i \sim p(y_i|f_i)$;
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.1.3 GP-NARX (VFE)

Algorithm F.3: Algorithm for a single sample in the CIMC simulation of the GP-NARX (VFE) model. NP denotes noise propagation.

Data: $\mathcal{D} = \{\mathbf{Z}_{1:t}, \mathbf{y}_{1:t}\}$: training data
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+2-n_a:t+1} \sim q(\mathbf{f}_{t+2-n_a:t+1})$;
$\tilde{y}_{t+1} \sim p(y_{t+1}|f_{t+1})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\tilde{f}_{t+1}]^T$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\tilde{y}_{t+1}]^T$;
$i \leftarrow t+2$;
$\mathbf{L}_u \mathbf{L}_u^T \leftarrow \text{cholesky}(\mathbf{K}_{u,u})$;
$\mathbf{L}_u^{-1} \leftarrow \mathbf{L}_u \backslash \mathbf{I}$;
$\mathbf{B} = \mathbf{I} + \mathbf{L}_u^{-1} \mathbf{K}_{u,f_{1:t}} \sigma_n^{-2} \mathbf{K}_{f_{1:t},u} \mathbf{L}_u^{-T}$;
$\mathbf{L}_B \mathbf{L}_B^T \leftarrow \text{cholesky}(\mathbf{B})$;
$\mathbf{L}_B^{-1} \leftarrow \mathbf{L}_B \backslash \mathbf{I}$;
$\boldsymbol{\alpha} \leftarrow \mathbf{L}_u^{-T} \mathbf{L}_u^{-1}$;
$\boldsymbol{\beta} \leftarrow \mathbf{L}_u^{-T} \mathbf{L}_B^{-T} \mathbf{L}_B^{-1} \mathbf{L}_u^{-1}$;
$\boldsymbol{\gamma} \leftarrow \sigma_n^{-2} \boldsymbol{\beta} \mathbf{K}_{u,f_{1:t}} \mathbf{y}_{1:t}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\mu \leftarrow \mathbf{K}_{f_i,u} \boldsymbol{\gamma}$;
  $\sigma^2 \leftarrow \mathbf{K}_{f_i,f_i} - \mathbf{K}_{f_i,u} \boldsymbol{\alpha} \mathbf{K}_{u,f_i} + \mathbf{K}_{f_i,u} \boldsymbol{\beta} \mathbf{K}_{u,f_i}$;
  $\tilde{f}_i \sim q(f_i|\mathcal{D}, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu, \sigma^2)$;
  $\tilde{y}_i \sim p(y_i|f_i)$;
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.1.4 GP-NARX (SVGP)

Algorithm F.4: Algorithm for a single sample in the CIMC simulation of the GP-NARX (SVGP) model. NP denotes noise propagation.

Data: $\mathcal{D} = \{\mathbf{Z}_{1:t}, \mathbf{y}_{1:t}\}$: training data
Data: $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\tilde{\mathbf{m}}, \tilde{\mathbf{L}}_\phi)$: $\tilde{\mathbf{m}}, \tilde{\mathbf{L}}_\phi$ are the whitened parameters determined by optimization
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+2-n_a:t+1} \sim q(\mathbf{f}_{t+2-n_a:t+1})$;
$\tilde{y}_{t+1} \sim p(y_{t+1}|f_{t+1})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\tilde{f}_{t+1}]^T$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\tilde{y}_{t+1}]^T$;
$i \leftarrow t+2$;
$\mathbf{L}_u \mathbf{L}_u^T \leftarrow \text{cholesky}(\mathbf{K}_{u,u})$;
$\mathbf{L}_u^{-1} \leftarrow \mathbf{L}_u \backslash \mathbf{I}$;
$\boldsymbol{\alpha} \leftarrow \mathbf{L}_u^{-T} \mathbf{L}_u^{-1}$;
$\boldsymbol{\beta} \leftarrow \mathbf{L}_u^{-T} \tilde{\mathbf{L}}_\Phi \tilde{\mathbf{L}}_\Phi^T \mathbf{L}_u^{-1}$;
$\boldsymbol{\gamma} \leftarrow \mathbf{L}_u^{-T} \tilde{\mathbf{m}}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\mu \leftarrow \mathbf{K}_{f_i,u} \boldsymbol{\gamma}$;
  $\sigma^2 \leftarrow \mathbf{K}_{f_i,f_i} - \mathbf{K}_{f_i,u} \boldsymbol{\alpha} \mathbf{K}_{u,f_i} + \mathbf{K}_{f_i,u} \boldsymbol{\beta} \mathbf{K}_{u,f_i}$;
  $\tilde{f}_i \sim q(f_i|\mathcal{D}, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu, \sigma^2)$;
  $\tilde{y}_i \sim p(y_i|f_i)$;
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.2 PIMC Simulation

Algorithm F.5: Algorithm for a single sample in the PIMC simulation of the GP-NARX models. NP denotes noise propagation.

Data: $\mathbf{Z}'_{1:m}$: pseudo-input locations
Data: $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{\Phi})$: posterior at the pseudo-input locations $\mathbf{Z}'_{1:m}$ for the respective GP-NARX model
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+2-n_a:t+1} \sim q(\mathbf{f}_{t+2-n_a:t+1})$;
$\tilde{y}_{t+1} \sim p(y_{t+1}|f_{t+1})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\tilde{f}_{t+1}]^T$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\tilde{y}_{t+1}]^T$;
$i \leftarrow t+2$;
$\tilde{\mathbf{u}} \sim q(\mathbf{u})$;
$\mathcal{D} \leftarrow \{\mathbf{Z}'_{1:m}, \tilde{\mathbf{u}}\}$;
$\mathbf{L}_u \mathbf{L}_u^T \leftarrow \text{cholesky}(\mathbf{K}_{u,u})$;
$\mathbf{L}_u^{-1} \leftarrow \mathbf{L}_u \backslash \mathbf{I}$;
$\boldsymbol{\alpha} \leftarrow \mathbf{L}_u^{-T} \mathbf{L}_u^{-1}$;
$\boldsymbol{\beta} \leftarrow \boldsymbol{\alpha}\tilde{\mathbf{u}}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\mu \leftarrow \mathbf{K}_{f_i,u}\boldsymbol{\beta}$;
  $\sigma^2 \leftarrow \mathbf{K}_{f_i,f_i} - \mathbf{K}_{f_i,u}\boldsymbol{\alpha}\mathbf{K}_{u,f_i}$;
  $\tilde{f}_i \sim p(f_i|\mathcal{D}, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu, \sigma^2)$;
  $\tilde{y}_i \sim p(y_i|f_i)$;
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.3 TCMC Simulation

Algorithm F.6: Algorithm for a single sample in the TCMC simulation of the GP-NARX models. The pseudo-point posteriors for the respective GP-NARX model are defined in Section E.2. PIMC simulation corresponds to the variance threshold $T_{\mathbb{V}} = \infty$. FCMC simulation corresponds to the variance threshold $T_{\mathbb{V}} = 0$.

Data: $T_{\mathbb{V}}$: variance threshold
Data: $\mathbf{Z}'_{1:m}$: pseudo-input locations
Data: $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{\Phi})$: posterior at the pseudo-input locations $\mathbf{Z}'_{1:m}$ for the respective GP-NARX model
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+1-n_a:t} \sim q(\mathbf{f}_{t+1-n_a:t})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\,]$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\,]$;
$i \leftarrow t+1$;
$\mathbf{K}_{f,f} \leftarrow \mathbf{K}_{u,u}$;
$\mathbf{L}\mathbf{L}^T \leftarrow \text{cholesky}(\mathbf{K}_{f,f})$;
$\tilde{\mathbf{u}} \sim q(\mathbf{u})$;
$\mathbf{Z} \leftarrow \mathbf{Z}'_{1:m}, \boldsymbol{\eta} \leftarrow \tilde{\mathbf{u}}, \mathcal{D} \leftarrow \{\mathbf{Z}, \boldsymbol{\eta}\}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\boldsymbol{\alpha} \leftarrow \mathbf{L} \backslash \mathbf{K}_{f_i,\eta}$;
  $\boldsymbol{\beta} \leftarrow \mathbf{L} \backslash \boldsymbol{\eta}$;
  $\mu \leftarrow \boldsymbol{\alpha}^T \boldsymbol{\beta}$;
  $\sigma^2 \leftarrow \mathbf{K}_{f_i,f_i} - \boldsymbol{\alpha}^T \boldsymbol{\alpha}$;
  $\tilde{f}_i \sim p(f_i|\mathcal{D}, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu, \sigma^2)$;
  $\tilde{y}_i \sim p(y_i|f_i)$;
  if $\sigma^2 \geq T_{\mathbb{V}}$ **then**
    $\mathbf{Z}^T \leftarrow [\mathbf{Z}^T, \tilde{\mathbf{z}}_i^T], \boldsymbol{\eta}^T \leftarrow [\boldsymbol{\eta}^T, \tilde{f}_i], \mathcal{D} \leftarrow \{\mathbf{Z}, \boldsymbol{\eta}\}$;
    $\mathbf{a} \leftarrow \mathbf{K}_{f,f_i}, b \leftarrow \mathbf{K}_{f_i,f_i}$;
    $\mathbf{K}_{f,f} \leftarrow \begin{bmatrix} \mathbf{K}_{f,f} & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix}$;
    $\mathbf{L} \leftarrow$ update cholesky$(\mathbf{L}, \{\mathbf{a}, b\})$ (Algorithm F.8);
  end
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.4 Heteroskedastic TCMC Simulation

Algorithm F.7: Algorithm for a single sample in the TCMC simulation of the HGP-NARX model. The pseudo-point posteriors for the respective GP-NARX model are defined in Section E.2.

Data: $T_{\mathbb{V}}$: variance threshold
Data: $\mathbf{Z}_f, \mathbf{Z}_g$: pseudo-input locations
Data: $q(\mathbf{u}_f) = \mathcal{N}(\mathbf{u}_f|\mathbf{m}_f, \mathbf{\Phi}_f), q(\mathbf{u}_g) = \mathcal{N}(\mathbf{u}_g|\mathbf{m}_g, \mathbf{\Phi}_g)$: posterior at the pseudo-input locations $\mathbf{Z}_f, \mathbf{Z}_g$ for the respective latent process
Result: $\tilde{\mathbf{f}}_{t+1:t+n}$: sample of the latent trajectory for the process
Result: $\tilde{\mathbf{y}}_{t+1:t+n}$: sample of the noisy trajectory
$\tilde{\mathbf{f}}_{t+1-n_a:t} \sim q(\mathbf{f}_{t+1-n_a:t})$;
$\tilde{\mathbf{f}}_{t+1:t+n} \leftarrow [\,]$;
$\tilde{\mathbf{y}}_{t+1:t+n} \leftarrow [\,]$;
$i \leftarrow t+1$;
$\mathbf{K}_{f,f} \leftarrow \mathbf{K}_{u_f,u_f}, \mathbf{K}_{g,g} \leftarrow \mathbf{K}_{u_g,u_g}$;
$\mathbf{L}_f\mathbf{L}_f^T \leftarrow \text{cholesky}(\mathbf{K}_{f,f}), \mathbf{L}_g\mathbf{L}_g^T \leftarrow \text{cholesky}(\mathbf{K}_{g,g})$;
$\tilde{\mathbf{u}}_f \sim q(\mathbf{u}_f), \tilde{\mathbf{u}}_g \sim q(\mathbf{u}_g)$;
$\boldsymbol{\eta}_f \leftarrow \tilde{\mathbf{u}}_f, \boldsymbol{\eta}_g \leftarrow \tilde{\mathbf{u}}_g$;
$\mathcal{D}_f \leftarrow \{\mathbf{Z}_f, \boldsymbol{\eta}_f\}, \mathcal{D}_g \leftarrow \{\mathbf{Z}_g, \boldsymbol{\eta}_g\}$;
while $i \leq t+n$ **do**
  if *NP* **then**
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{y}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  else
    $\tilde{\mathbf{z}}_i^T \leftarrow [\tilde{\mathbf{f}}_{i-n_a:i-1}^T, \mathbf{x}_{i-n_b:i-1}^T]$;
  end
  $\boldsymbol{\alpha}_f \leftarrow \mathbf{L}_f \backslash \mathbf{K}_{f_i,\eta_f}, \boldsymbol{\alpha}_g \leftarrow \mathbf{L}_g \backslash \mathbf{K}_{g_i,\eta_g}$;
  $\boldsymbol{\beta}_f \leftarrow \mathbf{L}_f \backslash \boldsymbol{\eta}_f, \boldsymbol{\beta}_g \leftarrow \mathbf{L}_g \backslash \boldsymbol{\eta}_g$;
  $\mu_f \leftarrow \boldsymbol{\alpha}_f^T\boldsymbol{\beta}_f, \mu_g \leftarrow \boldsymbol{\alpha}_g^T\boldsymbol{\beta}_g$;
  $\sigma_f^2 \leftarrow \mathbf{K}_{f_i,f_i} - \boldsymbol{\alpha}_f^T\boldsymbol{\alpha}_f, \sigma_g^2 \leftarrow \mathbf{K}_{g_i,g_i} - \boldsymbol{\alpha}_g^T\boldsymbol{\alpha}_g$;
  $\tilde{f}_i \sim p(f_i|\mathcal{D}_f, \tilde{\mathbf{z}}_i) = \mathcal{N}(f_i|\mu_f, \sigma_f^2)$;
  $\tilde{g}_i \sim p(g_i|\mathcal{D}_g, \tilde{\mathbf{z}}_i) = \mathcal{N}(g_i|\mu_g, \sigma_g^2)$;
  $\tilde{y}_i \sim p(y_i|f_i, g_i)$;
  if $\sigma_f^2 \geq T_{\mathbb{V}}$ **then**
    $\mathbf{Z}_f^T \leftarrow [\mathbf{Z}_f^T, \tilde{\mathbf{z}}_i^T], \boldsymbol{\eta}_f^T \leftarrow [\boldsymbol{\eta}_f^T, \tilde{f}_i], \mathcal{D}_f \leftarrow \{\mathbf{Z}_f, \boldsymbol{\eta}_f\}$;
    $\mathbf{a}_f \leftarrow \mathbf{K}_{f,f_i}, b_f \leftarrow \mathbf{K}_{f_i,f_i}$;
    $\mathbf{K}_{f,f} \leftarrow \begin{bmatrix} \mathbf{K}_{f,f} & \mathbf{a}_f \\ \mathbf{a}_f^T & b_f \end{bmatrix}$;
    $\mathbf{L}_f \leftarrow$ update cholesky$(\mathbf{L}_f, \{\mathbf{a}_f, b_f\})$ (Algorithm F.8);
  end
  if $\sigma_g^2 \geq T_{\mathbb{V}}$ **then**
    $\mathbf{Z}_g^T \leftarrow [\mathbf{Z}_g^T, \tilde{\mathbf{z}}_i^T], \boldsymbol{\eta}_g^T \leftarrow [\boldsymbol{\eta}_g^T, \tilde{g}_i], \mathcal{D}_g \leftarrow \{\mathbf{Z}_g, \boldsymbol{\eta}_g\}$;
    $\mathbf{a}_g \leftarrow \mathbf{K}_{g,g_i}, b_g \leftarrow \mathbf{K}_{g_i,g_i}$;
    $\mathbf{K}_{g,g} \leftarrow \begin{bmatrix} \mathbf{K}_{g,g} & \mathbf{a}_g \\ \mathbf{a}_g^T & b_g \end{bmatrix}$;
    $\mathbf{L}_g \leftarrow$ update cholesky$(\mathbf{L}_g, \{\mathbf{a}_g, b_g\})$ (Algorithm F.8);
  end
  $\tilde{\mathbf{f}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{f}}_{t+1:t+n}^T, \tilde{f}_i]$;
  $\tilde{\mathbf{y}}_{t+1:t+n}^T \leftarrow [\tilde{\mathbf{y}}_{t+1:t+n}^T, \tilde{y}_i]$;
  $i \leftarrow i+1$;
end

F.5 Cholesky Update

Algorithm F.8: Algorithm for the iterative update of the Cholesky factor. Updates the Cholesky factor based on a row/column addition to the covariance matrix.

Data: $\mathbf{L} \in \mathbb{R}^{m \times m}$: cached Cholesky factor
Data: $\{\mathbf{a}, b\}$: column addition to the covariance matrix as in Section (A.2.2)
Result: $\hat{L} \in \mathbb{R}^{(m+1)\times(m+1)}$: updated cholesky factor
$\boldsymbol{\alpha} \leftarrow \mathbf{L} \backslash \mathbf{a}$;
$\beta \leftarrow \sqrt{b - \boldsymbol{\alpha}^T \boldsymbol{\alpha}}$;
$\hat{\mathbf{L}} \leftarrow \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \boldsymbol{\alpha}^T & \beta \end{bmatrix}$;

Appendix G

Used hardware specifications

In the experiments presented in this thesis we used the following system specifications:

- CPU: Intel(R) Core(TM) i7-8700K CPU 3.70GHz,
- GPU: Quadro P4000 8GB GDDR5 [GP104GL],
- RAM: 2x16GB DIMM DDR4 3200 MHz,
- OS: Ubuntu 18.04.5 LTS,
- Software: GPflow 2.4.0, Tensorflow 2.4.0.

References

[1] L. Ljung, "System identification," in *Signal analysis and prediction*, Springer, 1998, pp. 163–173.

[2] T. P. Bohlin, *Practical grey-box process identification: theory and applications*. Springer Science & Business Media, 2006.

[3] J. Sjöberg, Q. Zhang, L. Ljung, *et al.*, "Nonlinear black-box modeling in system identification: A unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[4] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[5] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[6] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[7] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[8] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, 1995.

[9] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[11] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[12] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[13] J. Abonyi, "Fuzzy model identification," in *Fuzzy model identification for control*, Springer, 2003, pp. 87–164.

[14] I. Škrjanc, J. A. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey," *Information Sciences*, vol. 490, pp. 344–368, 2019.

[15] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on bayesian neural networks—a tutorial for deep learning users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[16] H. Wang and D.-Y. Yeung, "Towards Bayesian deep learning: A framework and some existing methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3395–3408, 2016.

[17] G. Gregorčič and G. Lightbody, "Gaussian process approach for modelling of nonlinear systems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 4-5, pp. 522–533, 2009.

[18] A. Capone, A. Lederer, and S. Hirche, "Gaussian process uniform error bounds with unknown hyperparameters for safety-critical applications," in *International Conference on Machine Learning*, PMLR, 2022, pp. 2609–2624.

[19] Y. Yi and G. Verbic, "Operating envelopes under probabilistic electricity demand and solar generation forecasts," *arXiv preprint arXiv:2207.09818*, 2022.

[20] P. Boškoski, M. Gašperin, D. Petelin, and Đ. Juričić, "Bearing fault prognostics using Rényi entropy based features and Gaussian process models," *Mechanical Systems and Signal Processing*, vol. 52, pp. 327–337, 2015.

[21] D. G. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 52, no. 6, pp. 119–139, 1951.

[22] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[23] J. Kocijan, *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[24] M. N. Gibbs, "Bayesian Gaussian processes for regression and classification," Ph.D. dissertation, Citeseer, 1998.

[25] D. J. MacKay, D. J. Mac Kay, *et al.*, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[26] S. Zhang, T. J. Rogers, and E. J. Cross, "Gaussian process based grey-box modelling for SHM of structures under fluctuating environmental conditions," in *European Workshop on Structural Health Monitoring*, Springer, 2020, pp. 55–66.

[27] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*, PMLR, 2016, pp. 370–378.

[28] A. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Artificial intelligence and statistics*, PMLR, 2013, pp. 207–215.

[29] A. Damianou, "Deep Gaussian processes and variational propagation of uncertainty," Ph.D. dissertation, University of Sheffield, 2015.

[30] H. Salimbeni and M. Deisenroth, "Doubly stochastic variational inference for deep Gaussian processes," *Advances in neural information processing systems*, vol. 30, 2017.

[31] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[32] Y. Shen, M. Seeger, and A. Ng, "Fast Gaussian process regression using kd-trees," *Advances in neural information processing systems*, vol. 18, 2005.

[33] V. Morariu, B. Srinivasan, V. C. Raykar, R. Duraiswami, and L. S. Davis, "Automatic online tuning for fast Gaussian summation," *Advances in neural information processing systems*, vol. 21, 2008.

[34] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[35] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse Gaussian process approximations," *Advances in neural information processing systems*, vol. 29, 2016.

[36] M. W. Seeger, C. K. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *International Workshop on Artificial Intelligence and Statistics*, PMLR, 2003, pp. 254–261.

[37] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, 2005.

[38] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," *arXiv preprint arXiv:1309.6835*, 2013.

[39] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," in *Artificial Intelligence and Statistics*, PMLR, 2015, pp. 351–360.

[40] S. Sun, J. Shi, A. G. Wilson, and R. Grosse, "Scalable variational Gaussian processes via harmonic kernel decomposition," *arXiv preprint arXiv:2106.05992*, 2021.

[41] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Artificial intelligence and statistics*, PMLR, 2009, pp. 567–574.

[42] T. D. Bui, J. Yan, and R. E. Turner, "A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 3649–3720, 2017.

[43] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured Gaussian processes (KISS-GP)," in *International conference on machine learning*, PMLR, 2015, pp. 1775–1784.

[44] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[45] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka, "Batched large-scale Bayesian optimization in high-dimensional spaces," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 745–754.

[46] M. Mutny and A. Krause, "Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[47] D. Calandriello, L. Carratino, A. Lazaric, M. Valko, and L. Rosasco, "Gaussian process optimization with adaptive sketching: Scalable and no regret," in *Conference on Learning Theory*, PMLR, 2019, pp. 533–557.

[48] R. Frigola, "Bayesian time series learning with Gaussian processes," Ph.D. dissertation, University of Cambridge, 2015.

[49] R. W. Hamming, *Digital filters*. Courier Corporation, 1998.

[50] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification*, Springer, 2001, pp. 547–577.

[51] M. P. Deisenroth, R. D. Turner, M. F. Huber, U. D. Hanebeck, and C. E. Rasmussen, "Robust filtering and smoothing with Gaussian processes," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1865–1871, 2011.

[52] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," *Advances in neural information processing systems*, vol. 26, 2013.

[53] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational Gaussian process state-space models," *Advances in neural information processing systems*, vol. 27, 2014.

[54] J. Kocijan and D. Petelin, "Output-error model training for Gaussian process models," in *Adaptive and Natural Computing Algorithms*, A. Dobnikar, U. Lotrič, and B. Šter, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 312–321.

[55] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[56] A. Girard and 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.

[57] A. McHutchon and C. Rasmussen, "Gaussian process training with input noise," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[58] H. Bijl, T. B. Schön, J.-W. van Wingerden, and M. Verhaegen, "System identification through online sparse Gaussian process regression with input noise," *IFAC Journal of Systems and Control*, vol. 2, pp. 1–11, 2017.

[59] N. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," *Advances in neural information processing systems*, vol. 16, 2003.

[60] M. Titsias and N. D. Lawrence, "Bayesian Gaussian process latent variable model," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 844–851.

[61] A. C. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational inference for uncertainty on the inputs of Gaussian process models," *arXiv preprint arXiv:1409.2287*, 2014.

[62] J. Quinonero-Candela, A. Girard, and C. E. Rasmussen, "Prediction at an uncertain input for Gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting," 2003.

[63] J. Q. Candela, A. Girard, J. Larsen, and C. E. Rasmussen, "Propagation of uncertainty in Bayesian kernel models-application to multiple-step ahead forecasting," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, IEEE, vol. 2, 2003, pp. II–701.

[64] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[65] A. Girard, C. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," *Advances in neural information processing systems*, vol. 15, 2002.

[66] P. Groot, P. Lucas, and P. Bosch, "Multiple-step time series forecasting with sparse Gaussian processes," 2011.

[67] T. Krivec, J. Kocijan, M. Perne, B. Grašič, M. Božnar, and P. Mlakar, "Data-driven method for the improving forecasts of local weather dynamics," *Engineering Applications of Artificial Intelligence*, vol. 105, p. 104 423, 2021. DOI: 10.1016/j.engappai.2021.104423.

[68] T. Beckers and S. Hirche, "Prediction with approximated Gaussian process dynamical models," *IEEE Transactions on Automatic Control*, pp. 1–1, 2021. DOI: 10.1109/TAC.2021.3131988.

[69] T. Krivec, G. Papa, and J. Kocijan, "Simulation of variational Gaussian process NARX models with GPGPU," *ISA transactions*, vol. 109, pp. 141–151, 2021. DOI: 10.1016/j.isatra.2020.10.011.

[70] A. Indrayan and R. K. Malhotra, *Medical biostatistics*. Chapman and Hall/CRC, 2017.

[71] B. Settles, "Active learning literature survey," 2009.

[72] F. Olsson, "A literature survey of active machine learning in the context of natural language processing," 2009.

[73] M. Prince, "Does active learning work? A review of the research," *Journal of engineering education*, vol. 93, no. 3, pp. 223–231, 2004.

[74] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[75] J. Mockus, *Bayesian approach to global optimization: theory and applications*. Springer Science & Business Media, 2012, vol. 37.

[76] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[77] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[78] L. Daston, *Classical probability in the Enlightenment*. Princeton University Press, 1995.

[79] P. S. de Laplace, *Théorie analytique des probabilités*. Courcier, 1820, vol. 7.

[80] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[81] D. Duvenaud, "Automatic model construction with Gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[82] A. G. d. G. Matthews, J. Hensman, R. Turner, and Z. Ghahramani, "On sparse variational methods and the Kullback-Leibler divergence between stochastic processes," pp. 231–239, 2016.

[83] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," 1970.

[84] M. D. Hoffman, A. Gelman, *et al.*, "The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[85] M. K. Titsias, N. Lawrence, and M. Rattray, "Markov chain Monte Carlo algorithms for Gaussian processes," *Inference and Estimation in Probabilistic Time-Series Models*, vol. 9, p. 298, 2008.

[86] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[87] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in Pytorch," 2017.

[88] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.

[89] S. P. Smith, "Differentiation of the Cholesky algorithm," *Journal of Computational and Graphical Statistics*, vol. 4, no. 2, pp. 134–147, 1995.

[90] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[91] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, *et al.*, "GPflow: A Gaussian process library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, no. 40, pp. 1–6, Apr. 2017. [Online]. Available: `http://jmlr.org/papers/v18/16-537.html`.

[92] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[93] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[94] M. A. Alvarez, L. Rosasco, N. D. Lawrence, *et al.*, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[95] E. V. Bonilla, K. Chai, and C. Williams, "Multi-task Gaussian process prediction," *Advances in neural information processing systems*, vol. 20, 2007.

[96] M. van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman, "A framework for interdomain and multioutput Gaussian processes," *arXiv preprint arXiv:2003.01115*, 2020.

[97] J. Hensman, A. G. Matthews, M. Filippone, and Z. Ghahramani, "MCMC for variationally sparse Gaussian processes," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[98] I. Murray and R. P. Adams, "Slice sampling covariance hyperparameters of latent Gaussian models," *Advances in neural information processing systems*, vol. 23, 2010.

[99] F. W. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge university press, 2010.

[100] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[101] H. Salimbeni, S. Eleftheriadis, and J. Hensman, "Natural gradients in practice: Non-conjugate variational inference in Gaussian process models," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 689–697.

[102] J. C. Sprott and J. C. Sprott, *Chaos and time-series analysis*. Oxford University Press Oxford, 2003, vol. 69.

[103] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," pp. 11–30, 2015.

[104] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[105] L. a. Golshani, "The rate of entropy for gaussian processes," *Journal of Statistical Research of Iran*, vol. 12, no. 1, 2015. DOI: `10.18869/acadpub.jsri.12.1.71`. eprint: `http://jsri.srtc.ac.ir/article-1-24-en.pdf`. [Online]. Available: `http://jsri.srtc.ac.ir/article-1-24-en.html`.

[106] L. Ljung, Q. Zhang, P. Lindskog, and A. Juditski, "Estimation of grey box and black box models for non-linear circuit data," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 399–404, 2004.

[107] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *2013 European Control Conference (ECC)*, IEEE, 2013, pp. 2933–2938.

[108] J. Noël and M. Schoukens, "Hysteretic benchmark with a dynamic nonlinearity," in *Workshop on nonlinear system identification benchmarks*, 2016, pp. 7–14.

[109] J.-P. Noël, A. F. Esfahani, G. Kerschen, and J. Schoukens, "A nonlinear state-space approach to hysteresis identification," *Mechanical Systems and Signal Processing*, vol. 84, pp. 171–184, 2017.

[110] P. Mlakar, M. Z. Božnar, B. Grašič, and B. Breznik, "Integrated system for population dose calculation and decision making on protection measures in case of an accident with air emissions in a nuclear power plant," *Science of The Total Environment*, vol. 666, pp. 786–800, 2019.

[111] M. Z. Božnar, B. Grašič, A. P. de Oliveira, J. Soares, and P. Mlakar, "Spatially transferable regional model for half-hourly values of diffuse solar radiation for general sky conditions based on perceptron artificial neural networks," *Renewable Energy*, vol. 103, pp. 794–810, 2017.

[112] J. Carpentier, "Contribution a l'etude du dispatching economique," *Bulletin de la Societe Francaise des Electriciens*, vol. 3, no. 1, pp. 431–447, 1962.

[113] W. A. Bukhsh, A. Grothey, K. I. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4780–4788, 2013.

[114] K. Petrou, M. Z. Liu, A. T. Procopiou, L. F. Ochoa, J. Theunissen, and J. Harding, "Operating envelopes for prosumers in LV networks: A weighted proportional fairness approach," in *2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, IEEE, 2020, pp. 579–583.

[115] K. Petrou, A. T. Procopiou, L. Gutierrez-Lagos, *et al.*, "Ensuring distribution network integrity using dynamic operating limits for prosumers," *IEEE Transactions on Smart Grid*, vol. 12, no. 5, pp. 3877–3888, 2021.

[116] L. Blackhall, "On the calculation and use of dynamic operating envelopes," *Australian National Univerity, Canberra, ACT*, 2020.

[117] R. Skagestad, "Electricity demand forecasting with Gaussian process regression," M.S. thesis, NTNU, 2018.

[118] M. Blum and M. Riedmiller, "Electricity demand forecasting using Gaussian processes," in *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, Citeseer, 2013.

[119] A. Dahl and E. Bonilla, "Scalable Gaussian process models for solar power forecasting," in *International Workshop on Data Analytics for Renewable Energy Integration*, Springer, 2017, pp. 94–106.

[120] A. Dahl and E. V. Bonilla, "Grouped Gaussian processes for solar power prediction," *Machine Learning*, vol. 108, no. 8, pp. 1287–1306, 2019.

[121] G. M. Masters, *Renewable and efficient electric power systems*. John Wiley & Sons, 2013.

[122] J. Lloyd, D. Duvenaud, R. Grosse, J. Tenenbaum, and Z. Ghahramani, "Automatic construction and natural-language description of nonparametric regression models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, 2014.

[123] A. D. Saul, J. Hensman, A. Vehtari, and N. D. Lawrence, "Chained Gaussian processes," in *Artificial Intelligence and Statistics*, PMLR, 2016, pp. 1431–1440.

[124] Wikipedia, *Wasserstein metric — Wikipedia, The Free Encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Wasserstein%20metric&oldid=1089055041`, [Online; accessed 03-June-2022], 2022.

[125] D. J. MacKay *et al.*, "Introduction to Gaussian processes," *NATO ASI series F computer and systems sciences*, vol. 168, pp. 133–166, 1998.

[126] M. K. Titsias, "Variational model selection for sparse Gaussian process regression," *Report, University of Manchester, UK*, 2009.

Bibliography

Publications Related to the Thesis

Journal Articles

T. Krivec, J. Kocijan, M. Perne, B. Grašič, M. Božnar, and P. Mlakar, "Data-driven method for the improving forecasts of local weather dynamics," *Engineering Applications of Artificial Intelligence*, vol. 105, p. 104 423, 2021. DOI: 10.1016/j.engappai.2021.104423.

T. Krivec, G. Papa, and J. Kocijan, "Simulation of variational Gaussian process NARX models with GPGPU," *ISA transactions*, vol. 109, pp. 141–151, 2021. DOI: 10.1016/j.isatra.2020.10.011.

Other Publications

Journal Articles

T. Krivec, D. Gradišar, M. Glavan, and G. Mušič, "Obdelava kompleksnih dogodkov pri spremljanju proizvodnega procesa," *Ventil : revija za fluidno tehniko in avtomatizacijo*, vol. 25, no. 1, pp. 46–53, 2019.

Conference Paper

T. Krivec, N. Hvala, and J. Kocijan, "Integrated theoretical and data-driven Gaussian process NARX model for the simulation of effluent concentrations in wastewater treatment plant," in *19th IFAC Symposium on System Identification SYSID, Padova, Italy, 13-16 July*, ser. IFAC papersOnline, vol. 54, 2021, pp. 714–719. DOI: 10.1016/j.ifacol.2021.08.445.

T. Krivec and J. Kocijan, "Model globokih Gaussovskih procesov za identifikacijo procesa Bouc-Wen = Deep Gaussian process model for the identification of the Bouc-Wen proces," in *Zbornik devetindvajsete mednarodne Elektrotehniške in računalniške konference ERK = Proceedings of the Twenty-ninth International Electrotechnical and Computer Science Conference ERK*, A. Žemva and A. Trost, Eds., ser. Zbornik, Elektrotehniške in računalniške konference (Online), vol. 29, Slovenska sekcija IEEE, = Slovenian Section IEEE, 2020, pp. 185–188.

J. M. Rožanec, T. Krivec, V. Keršič, *et al.*, "Bicycle sharing systems meet AI: Forecasting bicycles availability and decision-making.," in *33rd International Scientific Conference : September 21-23, Dubrovnik, Croatia : proceedings*, Faculty of Organization, Informatics, University of Zagreb. Central European Conference on Information, and Intelligent Systems, 2022, pp. 365–370.

T. Krivec, D. Gradišar, M. Glavan, and G. Mušič, "Zaznavanje anomalij v proizvodnem procesu preko obdelave kompleksnih dogodkov = Fault detection in production plant using complex event processing," in *Zbornik enajste konference AIG'19 Avtomatizacija v industriji in gospodarstvu, 9. in 10. april, Maribor, Slovenija*, N. Muškinja and B. Tovornik, Eds., Društvo avtomatikov Slovenije, 2019, pp. 34–39.

T. Krivec and J. Kocijan, "Gaussian process regression for big data," in *Knjiga povzetkov = Book of abstracts*, M. Topole, T. Turk Dermastia, M. Dežman, *et al.*, Eds., Ljubljana: Mednarodna podiplomska šola Jožefa Stefana = Jožef Stefan International Postgraduate School, Inštitut Jožef Stefan = Jožef Stefan Institute, 2019, p. 44.

T. Krivec and L. Žnidaršič, "Sparse Gaussian process regression for system identification," in *Workshop on Nonlinear System Identification Benchmarks, April 10-12*, Eindhoven, The Netherlands, 2019, p. 37.

Biography

The author of this thesis was born on the 8th of January 1993 in Novo mesto. He received a Bachelor's Degree in Electrical Engineering at the University of Ljubljana in 2015. He received a Master's Degree in Electrical Engineering (Control systems and computer engineering) in 2018 at the aforementioned university. During his studies, he was awarded the Zois scholarship, was the recipient of the Jožef Stefan Institute scholarship, and got the Dean's recognition for his overall study success. He was awarded the best master thesis of the year by the Technological Network of Process Control Technology (PCT).

Since October 2018, he has been working as a young researcher at the Jožef Stefan Institute, Department of Systems and Control. He is currently enrolled in Information and Communication Technologies program at the Jožef Stefan International Postgraduate School. His research area is the modeling of dynamical systems, more precisely using Gaussian process models for system identification. Since 2020, he has been junior fellow of the American Slovenian Education Foundation (ASEF).

To date, he published 3 original scientific articles, has 4 scientific conference contributions, and 2 scientific conference contribution abstracts.