University of West Bohemia
Faculty of Applied Sciences

Gaussian Process Models in System Identification and State Estimation

Ing. Jakub Prüher

A dissertation submitted for the degree of
Doctor of Philosophy

Supervisor: Doc. Ing. Ondřej Straka, Ph.D.
Department of Cybernetics

Pilsen, 2018

Declaration

I declare that the thesis has been composed by myself and that the work has not been submitted for any other degree or professional qualification. I confirm that the work submitted is my own, except where otherwise indicated. I confirm that appropriate credit has been given within this thesis where reference has been made to the work of others.

Pilsen, April 11, 2018 .....

Ing. Jakub Prüher

Acknowledgements

I would like to thank Prof. Miroslav Šimandl for giving me the support, patience and freedom to develop my ideas. I thank Doc. Ing. Ondřej Straka, Ph.D. for taking up the role as my supervisor and giving me many helpful suggestions in the course of my studies. I am grateful to Ing. Ladislav Král, Ph.D. for providing many helpful consultations and corrections for certain parts of this thesis. I am also indebted to Prof. Simo Särkkä for his enthusiasm and willingness to have me as a visiting student. Special thanks to Toni Karvonen, Filip Tronarp, Roland Hostettler, Marko Mikkonen and Kimmo Suotsalo for creating intellectually stimulating conversations and habitable office environment infused with the smell of coffee. Those six months passed way too quickly. I would like to thank my parents for providing the occasional rural escape from all the city racket.

Lastly, I would like to thank the Czech Science Foundation for supporting the project "Cooperative Approaches to Design of Nonlinear Filters" (project no. GA 16-19999J), which contributed to this thesis.

The world has achieved brilliance without wisdom, power without conscience. Ours is a world of nuclear giants and ethical infants. We know more about war than we know about peace, more about killing that we know about living.

– Omar Bradley

Education without values, as useful as it is, seems rather to make man a more clever devil.

– C. S. Lewis

... he that increaseth knowledge increaseth sorrow.

– Ecclesiastes 1:17-18

Abstract

V posledních dvou dekádách regresní modely na bázi gaussovských procesů prožily markantní rozvoj ve strojovém učení a jsou atraktivní alternativou k mnoha ustáleným regresním modelům v širokém rozsahu aplikací. Regrese s gaussovským procesem je bayesovský neparametrický model, který slučuje vysokou flexibilitu s traktabilní bayesovskou inferencí.

První aplikací GP modelů v této disertaci je identifikace nelineárních časově invariantních systémů afinních v řízení s funkcionální nejistotou. V disertaci navrhuji identifikační metodu s rekurzivním gaussovským procesem, kterou dále aplikuji ve funkcionálním duálním adaptivním řízení.

Ve druhé části disertace se soustředím na lokální nelineární sigma-bodové filtry, které aproximují netraktabilní momentové integrály numerickými kvadraturními pravidly. GP regrese hraje důležitou roli v bayesovské kvadratuře, která nahlíží na kvadraturu jako na problém pravděpodobnostní inference. Principu bayesovské kvadratury využívám ke konstrukci obecných kvadraturních momentových transformací na bázi gaussovského a studentského t-procesu, které následně aplikuji pro konstrukci sigma-bodových filtrů. Na varianci integrálu je nahlíženo jako na model integrační chyby, kterou navržené momentové transformace reflektují ve výsledných kovariancích. Finální přínos je věnován využití derivace integrované funkce ke snížení variance integrálu. Dále také dokazuji spojitosti s linearizační transformací využívané rozšířeným Kalmánovým filtrem.

Abstract

In the last two decades, Gaussian process regression models have experienced a resurgence in the machine learning community and continue to be an attractive alternative to the established regression models in wide range of application areas. Gaussian process regression is a Bayesian nonparametric model, which combines high expressiveness with tractable Bayesian inference.

The first application of GP models in this thesis is the identification of nonlinear time-invariant systems affine in control with functional uncertainty. I develop a recursive GP system identification method and apply it in a functional dual adaptive control.

In the second part of this thesis, I focus on the local nonlinear sigma-point filters, which approximate the intractable moment integrals by means of numerical quadrature rules. GP regression plays an important role in development of the Bayesian quadrature (BQ), which views numerical integration as probabilistic inference. I use the BQ approach to construct the moment transforms based on the Gaussian process quadrature and the Student's t-process quadrature. Variance of the integral is seen as a model of the integration error, which the proposed moment transformations reflect in the resulting covariances. The final contribution is devoted to utilization of derivative observations for decreasing the integral variance. Furthermore, I show connections to the linearization transform employed in the well-known extended Kalman filter.

Contents

Nomenclature

Roman Symbols

$\mathbf{C}$	Input-output covariance
$\mathbf{C}_{\mathrm{A}}$	Approximate input-output covariance
$\mathbb{C}[\cdot]$	Covariance operator
$\mathcal{D}$	Dataset of training examples
D	Dimensionality of the input
d_q	Dimensionality of the state noise
d_r	Dimensionality of the measurement noise
d_x	Dimensionality of the system state
d_z	Dimensionality of the measurement
E	Dimensionality of the output
$\mathbb{E}[\cdot]$	Expectation operator
$\mathbf{f}$	Vector of function values at training inputs $\mathbf{X}$
$\mathbf{f}^\star$	Vector of function values at test inputs $\mathbf{X}^\star$
$\mathbf{I}_D$	$D \times D$ identity matrix
K	Index of the last time step
k	Time step index
$\mathbf{K}_k$	Kalman gain at time k

$k(\mathbf{x}, \mathbf{x}')$	Covariance function (kernel)
$\mathbf{m}$	Input mean
M	Number of testing examples
$m(\mathbf{x})$	Mean function
N	Number of training examples
$\mathbf{P}$	Input covariance
$\mathbf{q}_k$	State noise
$\mathbf{r}_k$	Measurement noise
S	Number of basis vectors
$\mathbf{u}$	Vector of inducing variables
$\mathbb{V}[\cdot]$	Variance operator
$\mathbf{W}$	Quadrature weights for the transformed covariance
$\mathbf{W}_c$	Quadrature weights for the input-output covariance
$\mathbf{w}$	Quadrature weights for the transformed mean
$\mathbf{x}_{1:k}$	States from time step 1 to k
$\mathbf{X}$	Set of training inputs
$\mathbf{X}$	Matrix of sigma-points as columns.
$\mathbf{x}_i$	i-th training input
$\mathbf{x}_i^\star$	i-th test input
$\mathbf{x}_k$	State of a dynamic system at time k
$\mathbf{x}_n$	n-th sigma-point
$\mathbf{X}^\star$	Set of test inputs
$\mathbf{y}$	Vector of observed outputs corresponding to training inputs $\mathbf{X}$
$\mathbf{Y}$	Matrix of integrand evaluations

$y^\star$	Observed output corresponding to test input $\mathbf{x}^\star$
$\mathbf{z}_{1:k}$	Measurements from time step 1 to k
$\mathbf{z}_k$	Measurement of the system state at time k

Greek Symbols

$\tilde{\boldsymbol{\phi}}_i$	i-th basis vector
$\tilde{\boldsymbol{\Phi}}$	Set of basis vectors
ν_g	Degree of freedom of the Student's t-process
ε	Observation noise
$\boldsymbol{\Lambda}$	Matrix of kernel input lengthscales
$\boldsymbol{\mu}$	Output (transformed) mean
$\boldsymbol{\mu}_{\mathrm{A}}$	Approximate output (transformed) mean
$\boldsymbol{\Pi}$	Output (transformed) covariance
$\boldsymbol{\Pi}_{\mathrm{A}}$	Approximate output (transformed) covariance
σ^2	Observation noise variance
$\boldsymbol{\theta}$	Vector of kernel parameters
$\boldsymbol{\Xi}$	Matrix of unit sigma-points as columns.
$\boldsymbol{\xi}_n$	n-th unit sigma-point

Acronyms

AR	Auto-regressive
ARD	Automatic relevance determination
ARMAX	Auto-regressive moving average with exogenous input
ARX	Auto-regressive with exogenous input
AUKF	Adaptive unscented Kalman filter
BQ	Bayesian quadrature

BS-PF	Bootstrap particle filter
DDF	Central difference filter
CKF	Cubature Kalman filter
DDF	Divided difference filter
DOA	Direction of arrival
DoF	Degree of freedom
EKF	Extended Kalman filter
ELS	Extended least squares
GHKF	Gauss-Hermite Kalman filter
GHT	Gauss-Hermite transform
GP	Gaussian process
GPQ+D	Gaussian process quadrature with derivative observations
GPQ	Gaussian process quadrature
GPQKF	Gaussian process quadrature Kalman filter
GPQ-MT	Gaussian process quadrature moment transform
GPQSF	Gaussian process quadrature Student's t-filter
GP-SSM	Gaussian process state-space model
IEKF	Iterated extended Kalman filter
INC	Inclination indicator
IV	Instrumental variables
KF	Kalman filter
MA	Moving average
MC	Monte Carlo
MT	Moment transform

NARMAX	Nonlinear auto-regressive moving average with exogenous input
NLL	Negative log-likelihood
OLS	Ordinary least squares
PDF	Probability density function
PEM	Prediction error method
RBF	Radial basis function
RDR	Radar
RGP	Recursive Gaussian process
RMSE	Root mean square error
RUKF	Randomized unscented Kalman filter
SF	Student's t-filter
SIF	Stochastic integration filter
SKL	Symmetrized Kullback-Leibler divergence
SOS	Sum of squares
SR	Spherical-radial
SRT	Spherical-radial transform
S^2KF	Smart sampling Kalman filter
SSM	State space model
STD	Standard deviation
TOA	Time of arrival
TPQKF	Student's t-process quadrature Kalman filter
TPQ-MT	Student's t-process quadrature moment transform
TPQSF	Student's t-process quadrature Student's t-filter
TPQ	Student's t-process quadrature

TP	Student's t-process
UKF	Unscented Kalman filter
UNGM	Univariate nonstationary growth model
UT	Unscented transform

Chapter 1

Introduction

In the last two decades, Gaussian process regression models have experienced a resurgence in the machine learning community and continue to be an attractive alternative to the established regression models in wide range of application areas. What caused this sudden interest in regression models, that have been known in geostatistics community since the 60's? What makes them so attractive to many researchers today? To answer this, we have to look back to PhD thesis authored by Neal [1995]. Conventional wisdom at the time was that a neural network with very large number of hidden units would overfit the data and that the parameter optimization in such cases would be numerically unfeasible [Le Roux and Bengio, 2007]. Neal proposed a counterexample, which showed that Bayesian neural network with a Gaussian prior on hidden-to-output weights converges to a Gaussian process prior when the number of hidden units approaches infinity. Since expressive models are highly desirable for modeling complex functional dependencies, this was an encouraging discovery, which motivated Rasmussen and Williams to conduct further research into GP models. Gaussian process regression is a Bayesian nonparametric model, which combines high expressiveness with tractable Bayesian inference. For suitable choices of covariance functions, many conventional regression models are just special cases of GP regression [Rasmussen and Williams, 2006]. Favorable analytical properties of GP models make them the natural first choice when it comes to dealing with functional uncertainty. They were utilized for probabilistic formulation of time-frequency analysis [Turner and Sahani, 2014], predictive control [Kocijan et al., 2003], reinforcement learning [Deisenroth, 2009] and many other areas.

The first application of GP models in this thesis is in the system identification [Kocijan et al., 2005]. The system model is obtained in a primarily data-driven manner, which is in direct contrast to mathematical modeling, where the primary source of knowledge are the natural laws from physics or biology. Broadly speaking, the models

can be categorized as either *parametric*, which typically assume rigid structure with finite number of parameters, or *non-parametric*, where the model structure changes with the dataset and which can be construed as having potentially infinite number of parameters [Orbanz and Teh, 2010]. Well-known parametric structures include for example ARX and ARMAX models for linear systems [Ljung, 1999], while for non-linear systems, the NARMAX [Billings, 2013] or the various types of neural networks [Fabri and Kadirkamanathan, 2001] can be named. Methods based on frequency response and correlation analysis [Pintelon and Schoukens, 2012] are the typical examples of nonparametric methods for linear system identification. For its many favorable properties, the GP regression model is a uniquely suited candidate for the nonparametric identification of nonlinear systems. In Chapter 5, I develop a recursive GP system identification method and apply it in a functional dual adaptive control, which has been largely dominated by neural networks [Fabri and Kadirkamanathan, 2001].

Dynamic systems are widely used to model the behavior of real processes throughout the sciences. In many cases, it is useful to define a state of the system and consequently work with a state-space representation of the dynamics. When the dynamics exhibits stochasticity or can only be observed indirectly, the problem of state estimation becomes relevant. Estimating a state of the dynamic system from noisy measurements is a prevalent problem in many application areas such as aircraft guidance, GPS navigation [Grewal et al., 2007], weather forecast [Gillijns et al., 2006], telecommunications [Jiang et al., 2003] and financial time series analysis [Bhar, 2010]. When the state estimator is required to produce an estimate using only the present and past measurements, this is known as the filtering problem.

If the system dynamics and measurement functions are linear and state and measurement noises are white, additive and independent of the state initial condition, then the best linear unbiased estimator of the system state, in the sense of least mean square error, is known as the *Kalman filter* (KF) [Kalman, 1960; Kalman and Bucy, 1961]. It was soon realized that the requirement of linearity posed by the KF can be very restrictive, which naturally led to the development of filters that were able to cope with nonlinearities in the system description. The first of such was the extended KF (EKF) [Smith et al., 1962], which rests on the idea of local linearization by Taylor series expansion. A conceptually different approach is used by the unscented KF (UKF), where the idea is to approximate probability densities by a finite set of sigma-points.

I focus on the nonlinear local filters, which approximate the intractable moment integrals by means of numerical quadrature rules. GP regression plays an important role in development of the Bayesian quadrature (BQ), which views numerical integration as

probabilistic inference. A great advantage of BQ is that it provides probabilistic characterization of the integration error as part of the result. BQ fits into a broader emerging field of probabilistic numerics [Hennig et al., 2015] which views numerical computation as a process of Bayesian statistical inference, where the result is a probability measure over the solutions - not just a single solution.

The first part of the thesis covers the preliminaries. Chapter 2 outlines the problem of system identification, describes the popular parametric models and concludes with a section containing a detailed description of the GP regression. Chapter 3 is mainly devoted to introduction of the local nonlinear filtering. The transformation of moments is identified as the central problem in local nonlinear filtering and summary of the most commonly used moment transformations and corresponding filters is provided. Bayesian quadrature is discussed in the last section of the chapter. The second part of the thesis contains the contributions. Application of GPs for system identification in the context of functional dual adaptive control is covered in Chapter 5. Chapter 6 develops a general purpose moment transformation based on the Gaussian process quadrature (GPQ), which is later used for design of GPQ Kalman filter. I further propose the use of the Student's t-process quadrature (TPQ) as an alternative to GPQ. The general-purpose TPQ moment transform is defined, applied for construction of the TPQ Student's t-filter and compared with the GPQ in numerical experiments. Finally, I propose the use of derivative observations in GPQ and show connections of such quadratures to linearization.

Chapter 2

System Identification

Discovering the workings of the physical reality is one of the preeminent goals of the scientific endeavor. Human beings and other intelligent lifeforms obtain data through their qualitatively limited sensory observations and then infer a mental model of the external world on a daily basis. The mental model can thus be understood as a summary of the current state of belief about the external world.

System identification is a discipline largely developed in the control engineering community whose main objective is to infer models of dynamical systems primarily on the basis of measured data. The goals of system identification are twofold. Firstly, the objective is to build a model of the dynamical system that is a good approximation to the measured input-output data and, at the same time, minimizes the prediction error. Building on top of this, the second objective is to find a parsimonious model structure with easily interpretable parameters that reveals the underlying dynamic system characteristics. We will be predominantly concerned with building models with good predictive performance. This is also referred to as black-box or, in case more assumptions are placed on the model structure, gray-box modeling.

In our discourse, a *system* is understood to be a part of unknown physical reality, which can only be observed indirectly through noisy sensory measurements. Since many natural processes of engineering interest evolve in time, we will restrict our attention to dynamical systems, which describe how variables interact with each other in order to produce observable signals of interest, called outputs, in response to the external stimuli. Inputs are the signals that can be manipulated by the observer. Any other external stimuli are called disturbances, which represent an unpredictable influence on the system.

Before the identification can begin, it is necessary to design the experiment, which involves specifying a sufficiently rich input signal so that the obtained dataset is maxi-

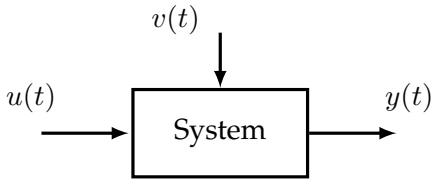

Figure 2.0.1: A diagram of a dynamical system turning the input signal $u(t)$ into an output $y(t)$. The whole system is affected by unpredictable external disturbances $v(t)$.

mally informative. Designing an ideal experiment is not always possible, though; in which case, data from the normal operation of the system has to be used.

Once the data has been collected, we have to choose a *model* set, usually on the basis of intuition and prior knowledge of the system, where every model is identified by a specific parameter values. Broadly speaking, models can be categorized as either *parametric*, which typically assume rigid structure with finite number of parameters; or *non-parametric*, where the model structure changes with the dataset and can be construed as having potentially infinite number of parameters [Orbanz and Teh, 2010]. These differences are discussed more in Section 2.2.1. The final step is to use an appropriate *identification method*, which processes the measured data in order to find optimal values of the model parameters.

In the discussion that follows, I have left out the non-parametric methods for linear systems based on frequency response, correlation, transient and spectral analyses. Interested reader is referred to [Pintelon and Schoukens, 2012] for detailed coverage of the frequency response analysis. Great overview of the non-parametric identification methods can be found in the classics by Söderström and Stoica [1989] and Ljung [1999].

2.1 Parametric Models

In this section, I briefly summarize the contemporary parametric models for linear and nonlinear systems, and also mention the commonly employed identification methods.

The Autoregressive Moving Average Model with Exogenous Input (ARMAX) is a well-known parametric model for linear systems. The form is given by

$$y(k) + \sum_{i=1}^{n_a} a_i y(k-i) = \sum_{i=1}^{n_b} b_i u(k-i) + \varepsilon(k) + \sum_{i=1}^{n_c} c_i \varepsilon(k-i), \tag{2.1.1}$$

where $y(k)$ is the system output, $u(k)$ is the input and $\varepsilon(k)$ is a noise signal with zero mean and finite variance. The quantities n_a, n_b and n_c are the maximum lags for the

relevant signals. Setting certain parameters to zero, results in several recognized special cases of the ARMAX model which are listed in the Table 2.1.1 along with the conditions. Other forms, such as Box-Jenkins model, are covered in [Billings, 2013].

Model structure	Conditions
ARMA	$b_i = 0$
ARX	$c_i = 0$
AR	$b_i = c_i = 0$
MA	$a_i = b_i = 0$

Table 2.1.1: Several widely recognized substructures of the ARMAX model.

The parameters of the ARX model can be easily estimated by the *ordinary least squares* (OLS) and the solution is available in closed-form. In general though, finding optimal parameter values for the ARMAX model, is a nonlinear regression problem requiring iterative methods, such as the *extended least squares* (ELS), the *prediction error method* (PEM) or the *instrumental variables* (IV). Note that PEM and IV are general model fitting frameworks containing concrete algorithms (such as ELS) as special cases.

The NARMAX model is a natural generalization[1] of the above in that it allows for arbitrary nonlinear combination of inputs, outputs and noise terms. The model is given by

$$\begin{aligned} y(k) = g(&y(k-1),\, y(k-2),\, \ldots,\, y(k-n_y),\\ &u(k-d-1),\, u(k-d-2),\, \ldots,\, u(k-d-n_u),\\ &\varepsilon(k-1),\, \varepsilon(k-2),\, \ldots,\, \varepsilon(k-n_\varepsilon)) + \varepsilon(k) \end{aligned} \tag{2.1.2}$$

where g is an arbitrary nonlinear function, the time delay is typically set to $d = 1$ and the quantities $n_y,\ n_u,\ n_\varepsilon$ are maximum lags of the corresponding signals. "A significant proportion of nonlinear systems can be represented by a NARMAX model including systems with exotic behaviors such as chaos, bifurcations, and subharmonics" [Billings, 2013].

The unknown nonlinearity g is most commonly parametrized by the power-form polynomial representation of given degree, which has been extensively studied and for which many identification algorithms have been developed. Polynomials are smooth functions and, thanks to the Weierstrass theorem, there are approximation guarantees for continuous functions on closed domains. However, the expansion can be ill condi-

[1] By definition, the term "nonlinear" excludes the linear case. Strictly speaking then, NARMAX *complements* ARMAX.

tioned due to the rapid growth of terms in the expansion. Rational functions [Zhu and Billings, 1993], general basis function expansions [Billings and Chen, 1989], wavelets [Wei and Billings, 2004] and neural networks [Chen et al., 1990] have been used as successful alternatives.

When the unknown nonlinear function is represented as linear-in-parameters, the identification method of choice for NARMAX models is the *orthogonal least squares* algorithm [Korenberg et al., 1988]. The solutions are typically sparse, which means that most of the terms in the expansion will be eliminated. The resulting model parsimony is a beneficial property for practical computational reasons, but it also reveals more about the underlying system dynamics (i.e. more than a black-box model).

2.2 Gaussian Process Regression

Gaussian process models [Barber, 2012; Bishop, 2007; Murphy, 2012; Rasmussen and Williams, 2006] are flexible non-parametric Bayesian models for regression and classification, that are enjoying a great surge of interest in recent decades. Gaussian process models are also at the heart of the Bayesian quadrature, a novel methodology for numerical approximation of integrals [Briol et al., 2015; Osborne et al., 2012; Rasmussen and Ghahramani, 2003], which might offer promising research directions in nonlinear filtering [Prüher and Šimandl, 2015; Särkkä et al., 2014; Särkkä et al., 2016].

The basic mathematical entity underlying these models is a stochastic Gaussian process. Formally, stochastic process is defined as a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ indexed by elements of an index set $\mathcal{X}$ [Grimmett and Stirzaker, 2001]. Although in the literature the stochastic processes are often discussed in the context of temporal settings (the index set is one-dimensional and represents time; $\mathcal{X} = \mathbb{R}^+$), note that by definition there are no restrictions placed on the index set. Thus a stochastic process can just as easily be defined on spacial domains (as it is common in geostatistics [Cressie, 1993]) as well as d_x-dimensional Euclidean spaces[2] (e.g. $\mathcal{X} = \mathbb{R}^{d_x}$). A stochastic process is characterized by specifying a finite dimensional distribution $p(f(x_1), f(x_2), \ldots, f(x_N))$ for all finite subsets $\{x_1, x_2, \ldots, x_N\}$ of an index set [Jazwinski, 1970]. For a *Gaussian process* it holds that all its finite dimensional distributions are Gaussian. Intuitively, Gaussian process (GP) can be understood as an infinite-dimensional extension of the multivariate Gaussian distribution and thus defines distribution over functions[3]. Analogously to the Gaussian distribution, a GP is fully specified

[2] A more accurate term for this mathematical entity is *random field* [Adler, 1981].

[3] Functions can be informally regarded as N-tuples of function values $\begin{bmatrix} f(x_1) & f(x_2) & \ldots & f(x_N) \end{bmatrix}^\top$ for which $N \to \infty$, i.e. as infinite-dimensional vectors.

by its mean and covariance functions. For a GP distributed random function we will write

$$f(x) \sim \mathrm{GP}\big(m(x),\, k(x, x')\big) \tag{2.2.1}$$

where $m(x)$ is the *mean function* and $k(x, x')$ is the *covariance function*. The covariance function is also often referred to as the *kernel* in machine learning literature [Rasmussen and Williams, 2006; Schölkopf and Smola, 2002]. The GP kernel $k(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric non-negative definite function; and conversely, every symmetric non-negative definite function defines a centered (zero-mean) GP [Janson, 1997]. In the following exposition, we will use the zero mean function $m(x) = 0$ because it simplifies notation.

Gaussian processes are the basis for the class of non-parametric GP regression and classification models, which are gaining on popularity in the machine learning community, because of their expressive power, flexibility and number of favorable theoretical properties [Rasmussen and Williams, 2006]. In the following, we will focus on the GP regression model as it lends itself nicely to function approximation, which is a problem prevalent in many applications, including nonlinear filtering [Särkkä, 2013; Särkkä et al., 2014; Turner and Rasmussen, 2010], system identification [Kocijan et al., 2005; Prüher and Šimandl, 2014], model predictive control [Kocijan et al., 2003], time series forecasting [Girard et al., 2003], functional dual adaptive control [Král et al., 2014] and reinforcement learning [Deisenroth, 2009; Kuss and Rasmussen, 2004; Park et al., 2013].

Gaussian process regression is a powerful non-parametric Bayesian model for solving nonlinear regression problems. Many parametric regression models, such as polynomial regression, neural networks and splines, can be obtained as a special case of the GP model [Rasmussen and Williams, 2006] by a suitable choice of the covariance function. Consider a dataset of input-output pairs $\mathcal{D} = \{(\mathbf{x}_1,\, y_1),\, \ldots,\, (\mathbf{x}_N,\, y_N)\}$, where the outputs are related to inputs through some unknown function

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathrm{N}\big(0,\, \sigma^2\big), \tag{2.2.2}$$

and are corrupted with zero-mean Gaussian noise. The goal of regression is to estimate the functional relationship between the observed data which also needs to give satisfactory predictive performance. Parametric regression models typically specify the regression function $f(\mathbf{x})$ by some parametric form as in eq. (2.2.14), which introduces rigid assumptions on the modeled relationship between inputs $\mathbf{x}$ and outputs y. The key distinguishing idea of the GP regression is that the unknown function is modeled

as a GP

$$f(\mathbf{x}) \sim \mathrm{GP}\big(0,\, k(\mathbf{x}, \mathbf{x}')\big) \tag{2.2.3}$$

where the modeling assumptions are introduced through the choice of the kernel (covariance) function $k(\mathbf{x}, \mathbf{x}')$. The zero-mean assumption is for notational and analytical convenience and does not present any serious limitation to the model flexibility, because the posterior mean is given as a weighted sum of kernels, which will become apparent later (see eq. (2.2.13)). Specifying the mean function is only important if we want to improve model's interpretability and predictive performance for inputs that lie far away from the training set. If we lack any meaningful nominal model that could be used as the mean function, one can always use any type of parametric basis function model as the mean [Rasmussen and Williams, 2006, Sec. 2.7].

From the viewpoint of Bayesian inference one could say that we have just specified a prior distribution over functions themselves. Since the unknown function in eq. (2.2.3) is a GP, the distribution for any finite subset of function values $\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) & \dots & f(\mathbf{x}_N) \end{bmatrix}^\top$ is multivariate Gaussian and will be denoted as

$$p(\mathbf{f} \,|\, \mathbf{X}, \boldsymbol{\theta}) = \mathrm{N}(\mathbf{f} \,|\, \mathbf{0},\, \mathbf{K}), \tag{2.2.4}$$

where the kernel matrix is given by pair-wise evaluations of the kernel function

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1; \boldsymbol{\theta}) & \dots & k(\mathbf{x}_1, \mathbf{x}_N; \boldsymbol{\theta}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1; \boldsymbol{\theta}) & \dots & k(\mathbf{x}_N, \mathbf{x}_N; \boldsymbol{\theta}) \end{bmatrix}. \tag{2.2.5}$$

The kernel function further depends on the parameters $\boldsymbol{\theta}$, where the dependence will be omitted to keep the notation uncluttered. For a set value of $\boldsymbol{\theta}$, we can draw samples from a GP prior. The Figure 2.2.1 shows the samples from a GP prior that uses RBF kernel. By choosing this kernel, the model operates with the assumption that the unknown function is smooth.

In the following, let $\mathbf{f}^\star = \begin{bmatrix} f(\mathbf{x}_1^\star) & \dots & f(\mathbf{x}_M^\star) \end{bmatrix}^\top$ be a vector of function values for a given *testing set* $\mathbf{X}^\star = \{\mathbf{x}_1^\star, \dots, \mathbf{x}_M^\star\}$ and let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathbf{y} = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^\top$ denote the *training set*. The GP posterior is given by the Bayes' theorem as

$$p(\mathbf{f} \,|\, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \frac{p(\mathbf{y} \,|\, \mathbf{f}, \mathbf{X}) p(\mathbf{f} \,|\, \mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{y} \,|\, \mathbf{X}, \boldsymbol{\theta})}, \tag{2.2.6}$$

where the term $p(\mathbf{y} \,|\, \mathbf{X}, \boldsymbol{\theta})$ is called marginal likelihood, which is useful for setting the

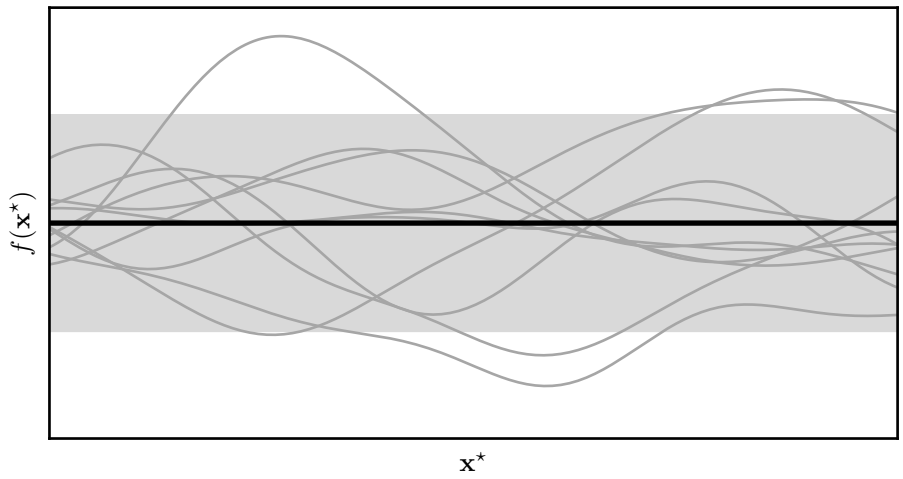

Figure 2.2.1: Gaussian process prior distribution with the RBF kernel. The mean function (bold), GP prior samples (gray), predictive uncertainty (gray band).

kernel parameters (see Section 2.2.2). For a Gaussian likelihood, that is $p\big(\mathbf{y} \,\big|\, \mathbf{f}, \sigma^2\big) = \mathrm{N}\big(\mathbf{y} \,\big|\, \mathbf{f},\, \sigma^2\mathbf{I}\big)$, the inference can be done exactly and results in a Gaussian posterior

$$p(\mathbf{f} \,|\, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \mathrm{N}\Big(\mathbf{f} \,\Big|\, \mathbf{K}\big(\mathbf{K} + \sigma^2\mathbf{I}\big)^{-1}\mathbf{y},\, \mathbf{K} - \mathbf{K}^\top\big(\mathbf{K} + \sigma^2\mathbf{I}\big)^{-1}\mathbf{K}\Big). \tag{2.2.7}$$

With the posterior distribution in hand, we can ask what function values $\mathbf{f}^\star$ the model predicts for new test inputs $\mathbf{X}^\star$. We arrive at the predictive distribution by first considering the joint distribution over the function values at the training and test inputs

$$p(\mathbf{f}, \mathbf{f}^\star \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}) = \mathrm{N}\left(\begin{bmatrix}\mathbf{f} \\ \mathbf{f}^\star\end{bmatrix} \,\middle|\, \mathbf{0},\, \begin{bmatrix}\mathbf{K} & \mathbf{K}_\star \\ \mathbf{K}_\star^\top & \mathbf{K}_{\star\star}\end{bmatrix}\right), \tag{2.2.8}$$

where $\mathbf{K}_\star$ is $N \times M$ a matrix with elements $k(\mathbf{x}_i, \mathbf{x}_j^\star)$ and $\mathbf{K}_{\star\star}$ is an $M \times M$ matrix with its elements given by $k(\mathbf{x}_i^\star, \mathbf{x}_j^\star)$. The joint predictive distribution is formed by marginalizing w.r.t. posterior distribution over functions

$$p(\mathbf{f}^\star \,|\, \mathbf{X}^\star, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) = \int p(\mathbf{f}^\star \,|\, \mathbf{f}, \mathbf{X}^\star, \mathbf{X}, \mathbf{y})\, p(\mathbf{f} \,|\, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta})\, \mathrm{d}\mathbf{f}. \tag{2.2.9}$$

Note, that $\mathbf{f}$ in eq. (2.2.9) has the same role as the parameter $\mathbf{w}$ in eq. (2.2.15), only now $\mathbf{f}$ can have arbitrarily large dimension depending on the size of the training set. Predictions of function values $\mathbf{f}^\star$ at testing inputs $\mathbf{X}^\star$ have the mean and covariance

given by

$$\mathbb{E}[\mathbf{f}^\star \mid \mathbf{X}^\star, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}] = \mathbf{K}_\star^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}, \tag{2.2.10a}$$

$$\mathbb{C}[\mathbf{f}^\star \mid \mathbf{X}^\star, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}] = \mathbf{K}_{\star\star} - \mathbf{K}_\star^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{K}_\star. \tag{2.2.10b}$$

For a single test input $\mathbf{x}^\star$ the predictive mean and variance of $f(\mathbf{x}^\star)$ are given by

$$m(\mathbf{x}^\star) = \mathbb{E}[f(\mathbf{x}^\star) \mid \mathcal{D}] = \mathbf{k}(\mathbf{x}^\star)^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}, \tag{2.2.11a}$$

$$\sigma^2(\mathbf{x}^\star) = \mathbb{V}[f(\mathbf{x}^\star) \mid \mathcal{D}] = k(\mathbf{x}^\star, \mathbf{x}^\star) - \mathbf{k}(\mathbf{x}^\star)^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{k}(\mathbf{x}^\star), \tag{2.2.11b}$$

where $\mathbf{k}(\mathbf{x}^\star) = k(\mathbf{x}^\star, \mathbf{X}) = \begin{bmatrix} k(\mathbf{x}^\star, \mathbf{x}_1) & \dots & k(\mathbf{x}^\star, \mathbf{x}_N) \end{bmatrix}^\top$. The GP mean function and the predictive variance are depicted in Figure 2.2.2. The expression (2.2.11a) for the

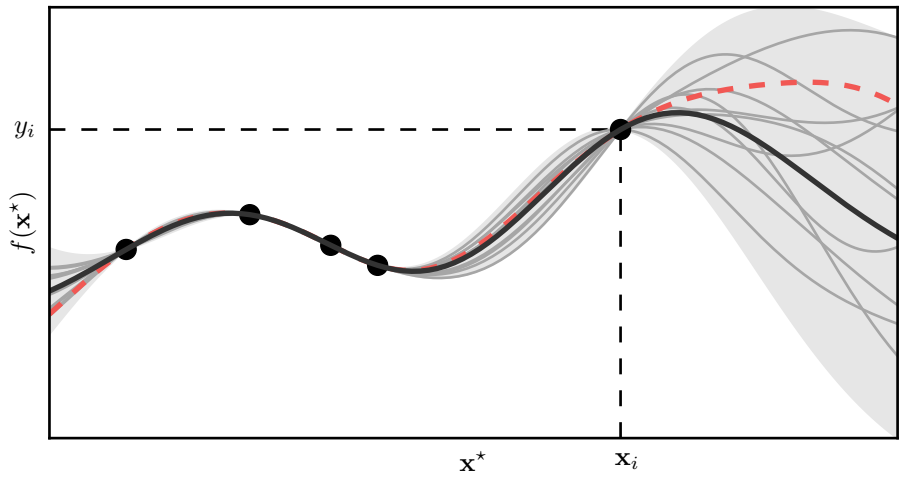

Figure 2.2.2: Gaussian process posterior distribution using RBF kernel. True underlying function (dashed), the GP posterior mean function $m(\mathbf{x}^\star)$ (bold), posterior predictive variance $\sigma^2(\mathbf{x}^\star)$ (gray area), training data (black dots) and the GP posterior samples (gray).

predictive mean can be rewritten using the substitution

$$\boldsymbol{\alpha} = \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y} \tag{2.2.12}$$

as

$$m(\mathbf{x}^\star) = \sum_{i=1}^{N} \alpha_i \, k(\mathbf{x}_i, \mathbf{x}^\star). \tag{2.2.13}$$

An interesting and useful property is that the mean function has a finite-dimensional representation, despite the fact that the mean function is an infinite dimensional object.

The insight we can draw from this is twofold [4]. First, we see that the GP prediction is a linear combination of kernel evaluations, thus (2.2.11a) is a linear predictor [Rasmussen and Williams, 2006]; and second, the more training data is used, the more complicated the expression for the posterior mean function will become, which is expected characteristic behavior of non-parametric models.

2.2.1 Relation to Parametric Models

The undisputed advantage of the GP models is that they are non-parametric. The nomenclature can be somewhat confusing, because 'non-parametric' here does not mean that the model has no parameters - in fact, non-parametric models have as much parameters as is needed (potentially infinite number of them) [Orbanz and Teh, 2010]. Another characteristic of such models is that they are not restricted by any prescribed parametric structure and their number of 'parameters' depends on the size of the dataset.

The non-parametric models stand in stark contrast to parametric models, which have a fixed structure and a finite number of parameters. To illustrate, consider a parametric Bayesian linear (in parameters) regression model

$$y = f(\mathbf{x};\, \mathbf{w}) + \varepsilon, \quad \varepsilon \sim \mathrm{N}\big(0,\, \sigma^2\big) \tag{2.2.14}$$

where the goal is to infer posterior distribution of the parameters $p(\,\mathbf{w} \,|\, \mathcal{D}\,)$ from data in the form of input-output pairs $\mathcal{D} = \{(\mathbf{x}_1, y_1),\, \ldots,\, (\mathbf{x}_N, y_N)\}$. Once the posterior over parameters is obtained, the predictive distribution $p(\,y^\star \,|\, \mathbf{x}^\star, \mathcal{D}\,)$ over an output (model response) $y^\star$ given a new input $\mathbf{x}^\star$ is given by

$$p(\,y^\star \,|\, \mathbf{x}^\star, \mathcal{D}\,) = \int p(\,y^\star \,|\, \mathbf{x}^\star,\, \mathbf{w}\,) p(\,\mathbf{w} \,|\, \mathcal{D}\,)\, \mathrm{d}\mathbf{w}. \tag{2.2.15}$$

The limitation of parametric models now becomes clearer. The following intuition from Ghahramani [2013] is especially helpful: "We can think of all models as information channels from past data $\mathcal{D}$ to future predictions $y^\star$. The parameters $\mathbf{w}$ in a parametric model constitute a bottleneck in this information channel. The complexity of the model, and the capacity of the channel, is bounded, even if the amount of observed data becomes unbounded. Parametric models are therefore not generally very flexi-

[4] Another insight is gained by using the substitution $\boldsymbol{\beta} = \mathbf{k}(\mathbf{x}^\star)^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1}$, which leads to $m(\mathbf{x}^\star) = \sum_{i=1}^{N} \beta_i\, y_i$. The prediction at $\mathbf{x}^\star$ is formed by weighted sum of noisy observations y_i, which indicates that 2.2.11a is a linear smoother.

ble." Increasing the number of parameters to infinity, would ensure that the flow of information from data to predictions is unlimited, which would essentially remove the parametric bottleneck. Indeed, constructing non-parametric models as a limiting case of parametric models is a proven strategy [Neal, 1995; Rasmussen, 1999].

2.2.2 Setting Kernel Parameters

The problem of model selection in the context of GP models refers to the optimal setting of the kernel parameters $\boldsymbol{\theta}$. Seeking an optimal value for the parameters is a frequentist resolution to the problem, whereas a fully Bayesian treatment would employ integration over parameter posterior for making predictions

$$p(\mathbf{f}^\star \mid \mathbf{X}^\star, \mathbf{X}, \mathbf{y}) = \int p(\mathbf{f}^\star \mid \mathbf{X}^\star, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{X}) \, \mathrm{d}\boldsymbol{\theta}. \tag{2.2.16}$$

A popular way of finding an optimal setting of kernel parameters is maximization of the marginal log-likelihood. Other ways of dealing with kernel parameters are outlined in [Rasmussen and Williams, 2006]. Marginal likelihood (evidence [MacKay, 2003]) for the GP regression model is given by

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y} \mid \mathbf{f}, \mathbf{X}) p(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{f}, \tag{2.2.17}$$

which is just the denominator in eq. (2.2.6) and is essentially a combination of likelihoods weighted by the prior over the 'parameter' $\mathbf{f}$. In practice, it is convenient to work with the logarithm of marginal likelihood, which, for a Gaussian likelihood, has an analytic form

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \underbrace{\mathbf{y}^\top \left(\mathbf{K}(\boldsymbol{\theta}) + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{y}}_{\text{data fit}} - \frac{1}{2} \underbrace{\log \left| \mathbf{K}(\boldsymbol{\theta}) + \sigma^2 \mathbf{I} \right|}_{\text{complexity penalty}} - \frac{N}{2} \log 2\pi. \tag{2.2.18}$$

Maximizing (2.2.18) w.r.t. $\boldsymbol{\theta}$ automatically trades-off the data fit term, which encourages complex models, with the complexity penalty term, which penalizes complex models and prevents overfitting. Gradient-based optimization algorithms, such as nonlinear conjugate gradient descent, can be used to obtain the ML-II estimate. The likelihood gradient is given by

$$\frac{\partial}{\partial \theta_i} \log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2} \operatorname{Tr} \left\{ \left(\boldsymbol{\alpha} \boldsymbol{\alpha}^\top - \mathbf{K}(\boldsymbol{\theta})^{-1} \right) \frac{\partial \mathbf{K}(\boldsymbol{\theta})}{\partial \theta_i} \right\}, \tag{2.2.19}$$

were $\boldsymbol{\alpha}$ is given by eq. (2.2.12). The marginal log-likelihood has multiple local maxima, which could be construed as different explanations of the observed data. When using this method, Wilson [2014] observed consistent model underfitting on small datasets. Proper setting of kernel parameters is thus of crucial importance for predictive performance of the GP model as illustrated in Figure 2.2.3, where mean functions of the GP with kernel $k(x, x') = \exp\left(-\frac{1}{2\lambda^2}(x - x')^2\right)$ are compared for three different values of the input lengthscale parameter λ.

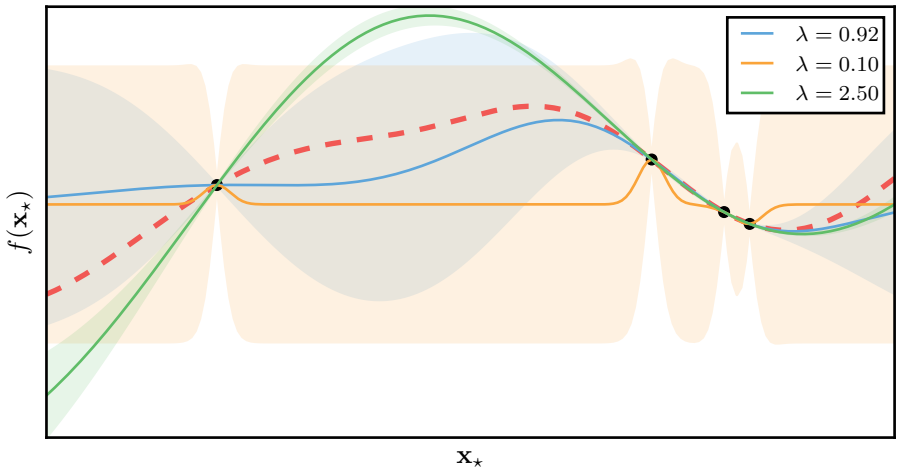

Figure 2.2.3: Predictive performance of a GP regression model with radial basis function covariance kernel is strongly affected by the values of the input lengthscale λ. For values that are too large ($\lambda = 2.50$) the predictive variance is unreasonably low (green), whereas for too small values ($\lambda = 0.10$) the posterior mean function changes too rapidly (yellow). The value $\lambda = 0.92$ was determined by marginal likelihood maximization (blue). The true function (dashed) is in red.

2.2.3 Approximations

The Gaussian process regression computes the predictions using all the available data points, which becomes computationally expensive, even prohibitive, for data sets that grow in size with time. For the standard GP regression the computational demands for prediction grow as $\mathcal{O}(N^3)$ and memory requirements grow with $\mathcal{O}(N^2)$, where N is the size of the training set. This fact can be easily seen from eqs. (2.2.11a) and (2.2.11b), where the complexity is dominated by the inversion of $N \times N$ matrix $\mathbf{K} + \sigma^2\mathbf{I}$.

For this reason, a number of approximations for the GP regression have been proposed, which typically reduce the computational demands to $\mathcal{O}(M^2N)$ where $M \ll N$. The subset of regressors approximation was proposed by Wahba [1990]. Another obvi-

ous approach is to avoid using the whole data set, but instead only use an informative subset of the data [Silverman, 1985]. Bayesian committee machine [Tresp, 2000] divides the dataset and forms the final predictive mean as a combination of the predictive means computed on each of the data partitions. Sparse greedy GP regression [Smola and Bartlett, 2001] is a representative of a greedy approximation strategy. The general idea is to maintain an active set of data points, which are added to the set only if a certain criterion is satisfied. Csató and Opper [2002] proposed the sparse online GP, which processes the data sequentially and gradually builds up a set of representative data based on information criteria. Another approach was proposed by Williams and Seeger [2001], who used the Nyström method for approximation of the kernel matrix. The sparse pseudo-input GP was proposed by Snelson and Ghahramani [2006], where the main idea is to jointly optimize the pseudo-input locations with kernel parameters.

Quiñonero Candela and Rasmussen [2005] presented a unifying framework, which recasts the problem of *approximate inference with exact prior* as a problem of *exact inference with approximate prior*. The framework rests on the idea of *inducing variables* $\mathbf{u}$, which turn out to be equivalents of, what other authors referred to as, "active set", "pseudo-inputs" or "support points". The joint Gaussian prior (2.2.8) can be written as

$$p(\mathbf{f}^\star, \mathbf{f}) = \int p(\mathbf{f}^\star, \mathbf{f}, \mathbf{u}) \, \mathrm{d}\mathbf{u} = \int p(\mathbf{f}^\star, \mathbf{f} \,|\, \mathbf{u}) p(\mathbf{u}) \, \mathrm{d}\mathbf{u}, \tag{2.2.20}$$

where $\mathbf{u} \sim \mathrm{N}(\mathbf{0}, \, \mathbf{K}_{uu})$. The fundamental approximation, which gives rise to most sparse approximations, is to assume that $\mathbf{f}^\star$ and $\mathbf{f}$ are conditionally independent given $\mathbf{u}$, such that

$$p(\mathbf{f}^\star, \mathbf{f}) \simeq q(\mathbf{f}^\star, \mathbf{f}) = \int q(\mathbf{f}^\star \,|\, \mathbf{u}) q(\mathbf{f} \,|\, \mathbf{u}) p(\mathbf{u}) \, \mathrm{d}\mathbf{u}. \tag{2.2.21}$$

Additional assumptions about the *training conditional* $q(\mathbf{f} \,|\, \mathbf{u})$ and the *test conditional* $q(\mathbf{f}^\star \,|\, \mathbf{u})$ then lead to different approximation algorithms. Among the more recent approaches, Hensman et al. [2013] worked with the idea of inducing variables and introduced a stochastic variational inference for GP regression achieving $\mathcal{O}(M^3)$ complexity, where M is the number of inducing variables. The idea behind the approach of Lázaro-Gredilla et al. [2010] is to use sparse spectral representation of the GP. Särkkä and Hartikainen [2012] proposed a solution which converts the GP regression to the state-space representation, where inference can be done using the Kalman smoother with linear $\mathcal{O}(N)$ complexity. This approach is limited in that it can only be applied to one-dimensional inputs (e.g. temporal processes). Huber [2013], drawing on the work

of Reece and Roberts [2010], proposed the recursive Gaussian Process algorithm, where the main idea is to specify a priori positions of the so-called basis vectors. The function values at the basis vectors are updated recursively together with kernel parameter [Huber, 2014]. The equations take on the form reminiscent of the Kalman smoother. This approach is not limited by the dimensionality of the input but the number of basis vectors, necessary to achieve good results, scales exponentially with dimension [Prüher and Šimandl, 2014]. Bui and Turner [2014] proposed a tree-structured GP approximation suitable for time series data.

Quiñonero Candela et al. [2007] gave an overview of the approximation methods based on the inducing inputs as well as fast matrix vector multiplication algorithms. Some methods are also discussed in [Rasmussen and Williams, 2006]. Schwaighofer and Tresp [2003] compared some of the approximations developed earlier, while Chalupka et al. [2013] proposed an evaluation framework.

Recursive Gaussian Process Regression

The recursive Gaussian process (RGP) approximation is heavily utilized in contributions to nonlinear system identification, discussed in Chapter 5, which is why it is described here in more detail. RGP was proposed by Huber [2013] as a solution to the problem of GP regression for on-line processing of sequentially arriving data. The main idea of this approximation is to use a predefined set of *basis vectors*, which are updated and effectively summarize the obtained information about the unknown function on a user-defined domain from currently available measurements. Instead of using all the data for prediction, as in the case of vanilla GP (cf. eqs. (2.2.11a) and (2.2.11b)), predictions of the RGP model are computed from the basis vectors. The number of basis vectors $S \ll N$ (where N is the number of data points) is fixed throughout the operation of the algorithm, which enables to keep the computational demands in check.

Keeping with the notation in [Huber, 2013], let $\tilde{\mathbf{\Phi}} = \begin{bmatrix} \tilde{\phi}_1 & \dots & \tilde{\phi}_S \end{bmatrix}$ denote $D \times S$ matrix of the basis vectors and $\tilde{\mathbf{f}} = f(\tilde{\mathbf{\Phi}})$ the corresponding vector of values of the unknown latent function. Because f is a GP, the distribution $p_0\Big(\tilde{\mathbf{f}}\Big) = \mathrm{N}\Big(\tilde{\mathbf{f}} \,\Big|\, \tilde{\boldsymbol{\mu}}_0,\, \tilde{\mathbf{\Sigma}}_0\Big)$ at initial time step $k = 0$ is Gaussian. For any time step $k > 0$, new set of L observations $\mathbf{y}_k = \begin{bmatrix} y_{k,1} & \dots & y_{k,L} \end{bmatrix}$ at input locations $\mathbf{X}_k = \begin{bmatrix} \mathbf{x}_{k,1} & \dots & \mathbf{x}_{k,L} \end{bmatrix}$ is processed. The goal is then to calculate posterior distribution

$$p\Big(\tilde{\mathbf{f}} \,\Big|\, \mathbf{y}_{1:k}\Big) = \mathrm{N}\Big(\tilde{\mathbf{f}} \,\Big|\, \tilde{\boldsymbol{\mu}}_k,\, \tilde{\mathbf{\Sigma}}_k\Big) \tag{2.2.22}$$

at time k, where $\mathbf{y}_{1:k} = \begin{bmatrix} \mathbf{y}_1 & \dots & \mathbf{y}_k \end{bmatrix}$, by combining the new observations $\mathbf{y}_k$ with the

distribution

$$p\Big(\tilde{\mathbf{f}} \,\Big|\, \mathbf{y}_{1:k-1}\Big) = \mathrm{N}\Big(\tilde{\mathbf{f}} \,\Big|\, \tilde{\boldsymbol{\mu}}_{k-1},\, \tilde{\boldsymbol{\Sigma}}_{k-1}\Big) \tag{2.2.23}$$

from the previous time step, which functions as a prior in Bayesian setting. Calculation of the posterior $p\Big(\tilde{\mathbf{f}} \,\Big|\, \mathbf{y}_{1:k}\Big)$ is performed in two steps:

Inference: calculating the joint prior $p\Big(\tilde{\mathbf{f}}, \mathbf{f}_k \,\Big|\, \mathbf{y}_{1:k-1}\Big)$ given the prior (2.2.23), where $\mathbf{f}_k = f(\mathbf{X}_k)$,

Update: updating the joint prior with new observations and integrating out $\mathbf{f}_k$.

I summarize the RGP algorithm below, without going further into more detail. For the purposes of this thesis, I only ever consider processing of one observation y_k at any given time. The RGP algorithm operates by means of the two sets of intertwined equations

$$\mathbf{J}_k = k(\mathbf{x}_k,\, \tilde{\boldsymbol{\Phi}})k(\tilde{\boldsymbol{\Phi}},\, \tilde{\boldsymbol{\Phi}})^{-1}, \tag{2.2.24a}$$

$$\hat{\mu}_k = m(\mathbf{x}_k) + \mathbf{J}_k\big(\tilde{\boldsymbol{\mu}}_{k-1} - m(\tilde{\boldsymbol{\Phi}})\big), \tag{2.2.24b}$$

$$\hat{\sigma}_k^2 = k(\mathbf{x}_k,\, \mathbf{x}_k) + \mathbf{J}_k\big(\tilde{\boldsymbol{\Sigma}}_{k-1} - k(\tilde{\boldsymbol{\Phi}},\, \tilde{\boldsymbol{\Phi}})\big)\mathbf{J}_k^\top, \tag{2.2.24c}$$

$$\mathbf{G}_k = \tilde{\boldsymbol{\Sigma}}_{k-1}\mathbf{J}_k^\top\big(\hat{\sigma}_k^2 + \sigma^2\big)^{-1}, \tag{2.2.25a}$$

$$\tilde{\boldsymbol{\mu}}_k = \tilde{\boldsymbol{\mu}}_{k-1} + \mathbf{G}_k(y_k - \hat{\mu}_k), \tag{2.2.25b}$$

$$\tilde{\boldsymbol{\Sigma}}_k = \tilde{\boldsymbol{\Sigma}}_{k-1} - \mathbf{G}_k\mathbf{J}_k\tilde{\boldsymbol{\Sigma}}_{k-1}, \tag{2.2.25c}$$

where

$$m(\tilde{\boldsymbol{\Phi}}) \triangleq \begin{bmatrix} m(\tilde{\boldsymbol{\phi}}_1) & \dots & m(\tilde{\boldsymbol{\phi}}_S) \end{bmatrix}^\top, \tag{2.2.26}$$

$$k(\mathbf{x}_k,\, \tilde{\boldsymbol{\Phi}}) \triangleq \begin{bmatrix} k(\mathbf{x}_k,\, \tilde{\boldsymbol{\phi}}_1) & \dots & k(\mathbf{x}_k,\, \tilde{\boldsymbol{\phi}}_S) \end{bmatrix}, \tag{2.2.27}$$

$$k(\tilde{\boldsymbol{\Phi}},\, \tilde{\boldsymbol{\Phi}}) \triangleq \begin{bmatrix} k(\tilde{\boldsymbol{\phi}}_1,\, \tilde{\boldsymbol{\phi}}_1) & \dots & k(\tilde{\boldsymbol{\phi}}_1,\, \tilde{\boldsymbol{\phi}}_S) \\ \vdots & \ddots & \vdots \\ k(\tilde{\boldsymbol{\phi}}_S,\, \tilde{\boldsymbol{\phi}}_1) & \dots & k(\tilde{\boldsymbol{\phi}}_S,\, \tilde{\boldsymbol{\phi}}_S) \end{bmatrix}. \tag{2.2.28}$$

Note, that in the above summary, we assumed non-zero GP prior mean function. The eqs. (2.2.24a) to (2.2.24c) describe how to compute the RGP model predictive mean $\hat{\mu}_k$ and variance $\hat{\sigma}_k^2$ at the test point $\mathbf{x}_k$. The eqs. (2.2.25a) to (2.2.25c) define the update step, where the first two moments of the posterior (2.2.22) are updated using the

latest function value observation y_k. For detailed explanations and derivation of the eqs. (2.2.24a) to (2.2.25c) reader is referred to the original article [Huber, 2013].

2.3 Student's t-Process Regression

An attractive alternative to the GP regression is the Student's t-process (TP) regression [Shah et al., 2014; Solin and Särkkä, 2015], which makes less restrictive assumption on the functional uncertainty and thus may provide more accurate predictive variances.

Consider a real-valued function $f\colon \mathbb{R}^D \to \mathbb{R}$ which is assigned a TP prior, denoted as $f(\mathbf{x}) \sim \mathrm{TP}(0,\, k(\mathbf{x}, \mathbf{x}'),\, \nu_g)$. This implies that for any finite collection of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ the function values are jointly Student's t-distributed

$$\begin{bmatrix} f(\mathbf{x}_1) & \cdots & f(\mathbf{x}_N) \end{bmatrix} \sim \mathrm{St}(\mathbf{0},\, \mathbf{K},\, \nu_g), \tag{2.3.1}$$

with the degrees of freedom (DoF) $\nu_g > 2$. The kernel (covariance) matrix $\mathbf{K}$ is made up of pairwise kernel evaluations, so that $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the kernel parameters. For brevity, dependence on $\boldsymbol{\theta}$ will be made explicit only when absolutely necessary. Conditioning on the function value observations $\mathbf{y} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$ at the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$, collectively denoted as $\mathcal{D} = \{(\mathbf{x}_n,\, f(\mathbf{x}_n))\}_{n=1}^N$, results in a TP posterior with predictive mean and variance [Shah et al., 2014; Solin and Särkkä, 2015]

$$\mathbb{E}_f[f(\mathbf{x}^\star) \mid \mathcal{D}] = \mathbf{k}(\mathbf{x}^\star)^\top \mathbf{K}^{-1} \mathbf{y}, \tag{2.3.2a}$$

$$\mathbb{V}_f[f(\mathbf{x}^\star) \mid \mathcal{D}] = \frac{\nu_g - 2 + \mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{\nu_g - 2 + N} \Big[k(\mathbf{x}^\star, \mathbf{x}^\star) - \mathbf{k}(\mathbf{x}^\star)^\top \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}^\star) \Big], \tag{2.3.2b}$$

where $[\mathbf{k}(\mathbf{x}^\star)]_i = k(\mathbf{x}^\star, \mathbf{x}_i)$. Evidently, the posterior mean is identical to that of the GP regression (cf. eq. (2.2.11a)), but the posterior variance (2.3.2b) has an additional data-dependent scaling coefficient. The DoF ν_g is an additional tunable parameter allowing for control of the heavy-tailed behavior of the TP. The lower the DoF, the heavier the tails and vice versa. For increasing DoF, the scaling factor becomes less dependent on the function values and eventually ($\nu_g = \infty$) the GP predictive variance is recovered, which means the GP regression can be interpreted as a special case of the TP regression.

As in the case of GP, the kernel parameters $\boldsymbol{\theta}$ and the DoF ν_g can be fitted by

maximizing the marginal likelihood, which has the form

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \nu_g) = -\frac{\nu_g + N}{2} \log\left(1 + \frac{\beta}{\nu_g - 2}\right) - \frac{1}{2}\log\left(|\mathbf{K}(\boldsymbol{\theta})|\right) - \frac{N}{2}\log\left((\nu_g - 2)\pi\right) + \log\left(\frac{\Gamma(\frac{\nu_g+N}{2})}{\Gamma(\frac{\nu_g}{2})}\right). \tag{2.3.3}$$

where $\beta = \mathbf{y}^\top \mathbf{K}(\boldsymbol{\theta})^{-1}\mathbf{y}$. The likelihood gradients are given by [Shah et al., 2014]

$$\frac{\partial}{\partial \theta_i} \log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \nu_g) = \frac{1}{2}\operatorname{Tr}\left\{\left(\frac{\nu_g + N}{\nu_g - 2 + \beta}\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}(\boldsymbol{\theta})^{-1}\right)\frac{\partial \mathbf{K}(\boldsymbol{\theta})}{\partial \theta_i}\right\}, \tag{2.3.4a}$$

$$\frac{\partial}{\partial \nu_g} \log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}, \nu_g) = -\frac{1}{2}\log\left(1 + \frac{\beta}{\nu_g - 2}\right) + \frac{(\nu_g + N)\beta}{2(\nu_g - 2)[(\nu_g - 2) + \beta]} + \psi\left(\frac{\nu_g + N}{2}\right) - \psi\left(\frac{\nu_g}{2}\right) - \frac{N}{2(\nu_g - 2)}, \tag{2.3.4b}$$

where $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$ and ψ is a digamma function [Abramowitz and Stegun, 1965]. The difference in the predictive variance is depicted in Figure 2.3.1, where the GP and TP are compared using the same values of kernel parameters. The mean functions of both models, which approximate the true underlying function, are identical. The TP is able to inflate the predictive variance due to its heavy-tailed nature, resulting in more realistic functional uncertainty given the available data.

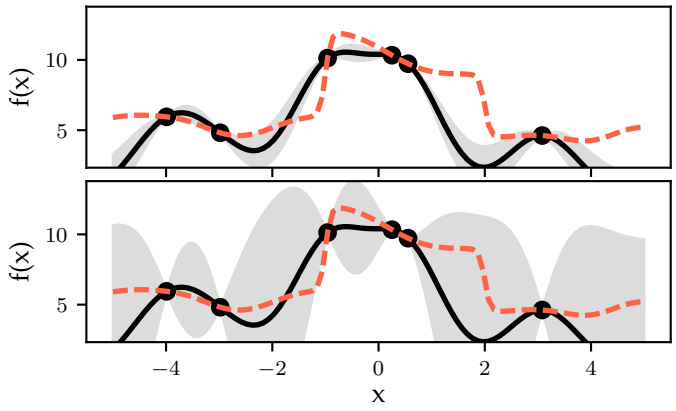

Figure 2.3.1: Comparison of predictive moments of the Gaussian process (top) and the Student's t-process (bottom) regression models using the same set of kernel parameters. The DoF of the TP model was set to $\nu_g = 10$. The true function (dashed red), the posterior mean (solid black) and predictive variance (gray band).

Chapter 3

State Estimation

In this chapter, I summarize a variety of filtering algorithms, which were originally derived under disparate assumptions. In order to simplify the exposition and unify the various filtering algorithms, I view the filters from the Bayesian perspective.

Consider an autonomous dynamic system (no exogenous input) with additive white noise. The structural description of such a system is given by the state-space model (SSM) in the form

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \quad \mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}) \tag{3.0.1a}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k, \quad \mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}) \tag{3.0.1b}$$

where $\mathbf{x}_k \in \mathbb{R}^{d_x}$ is the system state at time instant k and $\mathbf{z}_k \in \mathbb{R}^{d_z}$ is the corresponding state measurement. The state noise $\mathbf{q}_k \in \mathbb{R}^{d_x}$ with covariance $\mathbf{Q}$ and measurement noise $\mathbf{r}_k \in \mathbb{R}^{d_z}$ with covariance $\mathbf{R}$ are both considered white, mutually independent and independent of the state initial conditions $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{0}, \mathbf{P}_0^x)$. The system dynamics $\mathbf{f}(\mathbf{x}_k) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ and measurement function $\mathbf{h}(\mathbf{x}_k) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ are known vector functions. The SSM comprises a set of two equations. The first is the system equation (3.0.1a) describing the state dynamics. The system state $\mathbf{x}_k$ is hidden from the observer and only an indirect observation is possible. The way the measurement of the state $\mathbf{z}_k$ arises is modeled by measurement equation (3.0.1b). The structural description can be generalized to time-variant systems

$$\mathbf{x}_k = \mathbf{f}_k\big(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}\big), \tag{3.0.2a}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{r}_k), \tag{3.0.2b}$$

where the state noise $\mathbf{q}_k \in \mathbb{R}^{d_q}$ and measurement noise $\mathbf{r}_k \in \mathbb{R}^{d_r}$ do not necessarily

affect all state dimensions. Furthermore, the noises do not have to be additive, Gaussian or homoscedastic, so that $\mathbb{C}[\mathbf{q}_k] = \mathbf{Q}_k$ and $\mathbb{C}[\mathbf{r}_k] = \mathbf{R}_k$, where $\mathbb{C}[\cdot]$ is the covariance operator.

Alternatively, a dynamical system can be specified probabilistically by a set of conditional probability density functions (PDF),

$$\mathbf{x}_0 \sim p(\mathbf{x}_0), \tag{3.0.3a}$$

$$\mathbf{x}_k \sim p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}), \tag{3.0.3b}$$

$$\mathbf{z}_k \sim p(\mathbf{z}_k \,|\, \mathbf{x}_k). \tag{3.0.3c}$$

At the first time step $k = 0$ the state is described by the initial state PDF (3.0.3a). The transition PDF (3.0.3b) specifies how the distribution of the current state depends on the previous state. The state evolution is effectively modeled as a Markov process.[1] The measurement PDF (3.0.3c) describes the dependence of the current measurement on the current state. Owing to the Markovian conditional independence structure, which

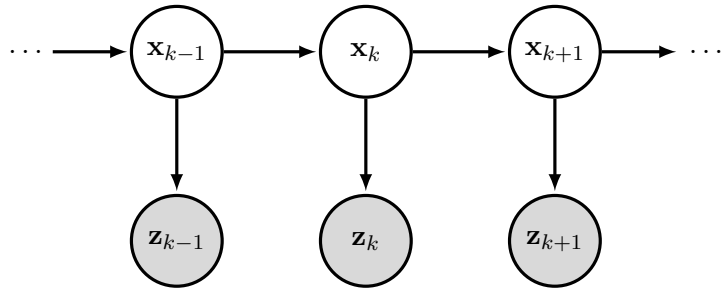

Figure 3.0.1: Graphical model depicting conditional independence structure of the SSM. The hidden state sequence is a first-order Markov chain. The observed random variables are in gray.

is encoded by the graph in Figure 3.0.1, the joint PDF over the states and measurements factorizes as

$$p(\mathbf{x}_{0:K}, \mathbf{z}_{1:K}) = p(\mathbf{x}_0) \prod_{k=1}^{K} p(\mathbf{z}_k \,|\, \mathbf{x}_k) p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}), \tag{3.0.4}$$

where $\mathbf{x}_{0:K} = \{\mathbf{x}_0, \ldots, \mathbf{x}_K\}$ and $\mathbf{z}_{1:K} = \{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ denote the set of states and measurements for the indicated range of time steps. In principle, we could utilize the Bayes rule and infer the distribution over the unknown state at all time instances given all the available measurements $p(\mathbf{x}_{0:K} \,|\, \mathbf{z}_{1:K})$. In dynamic settings, where the

[1] A discrete-time stochastic process is called Markov process if $p(\mathbf{x}_{k+1} \,|\, \mathbf{x}_k, \mathbf{x}_{k-1}) = p(\mathbf{x}_{k+1} \,|\, \mathbf{x}_k)$. In other words, the future is independent of the past given the present (all information about the past is concentrated in the present).

data arrive sequentially, this kind of inference becomes prohibitively expensive. This is because the posterior $p(\mathbf{x}_{0:K} \mid \mathbf{z}_{1:K})$ would have to be recomputed on every occasion a new measurement arrives. Utilizing the conditional independence structure of the model (see Figure 3.0.1) is thus of paramount importance in search for a tractable inference algorithm. For practical purposes several inference tasks were formulated that are conducive to efficient implementations. We talk about

- *prediction*: which is concerned with inferring the state, which is one or several steps into the *future*, given the past available measurements; $p(\mathbf{x}_k \mid \mathbf{z}_{1:k-l})$ for $1 < l < k + 1$,
- *filtering*: which is concerned with inference of the *present* state given the all currently available measurements; $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$, or
- *smoothing*: which is concerned with inference of the state at the *past* time instants given all the available measurements; $p(\mathbf{x}_k \mid \mathbf{z}_{1:K})$, where $k \leq K$.

Maybeck [1982] notes that a number of smoothing problems can be defined, but recognizes the three most important ones due to their practical applicability. These are

- *fixed-point smoothing*: The state estimate at a chosen past time of interest is recalculated given the increasing amount of data.
- *fixed-lag smoothing*: The state is estimated in a restricted moving window of past time instances.
- *fixed-interval smoothing*: The state at all past time instances is estimated given the data up to the present time.

3.1 Nonlinear Filtering

The posterior (filtering) PDF $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ of the system state at each time step is given by the following Bayesian recursive relations

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})}, \tag{3.1.1a}$$

$$p(\mathbf{z}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) \, \mathrm{d}\mathbf{x}_k, \tag{3.1.1b}$$

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1}) \, \mathrm{d}\mathbf{x}_{k-1}. \tag{3.1.1c}$$

The numerator of eq. (3.1.1a) consists of the *likelihood* term $p(\mathbf{z}_k \mid \mathbf{x}_k)$, determined from the measurement model (3.0.1b) and the predictive PDF $p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})$, which acts as a *prior* and is determined from the system dynamics (3.0.1a). The term in the denominator is a normalizing constant for the posterior, called the *marginal likelihood* (also known as *evidence* [MacKay, 2003]). The Bayesian recursive relations describe a general solution to the filtering problem, which is unfortunately intractable except for a few special cases (linear functions and Gaussian noises in eqs. (3.0.1a) and (3.0.1b)). Filtering algorithms can be divided into two categories:

- The *global filters* provide an estimate of the whole filtering PDF at each time step, which enables them to keep track of multi-modalities. This is paid for with their increased computational burden.
- The *local filters*, on the other hand, only keep track of the moments of the filtering PDF and the estimates are only valid in a limited area of the state space.

3.1.1 Gaussian Filtering

Gaussian filters fall into the category of local filters governed by the approximative assumption that the joint PDF of the state and measurement $p(\mathbf{x}_k, \mathbf{z}_k \mid \mathbf{z}_{1:k-1})$ for any time step k is Gaussian. That is

$$\begin{bmatrix} \mathbf{x}_k \\ \mathbf{z}_k \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \mathbf{m}^x_{k|k-1} \\ \mathbf{m}^z_{k|k-1} \end{bmatrix}, \begin{bmatrix} \mathbf{P}^x_{k|k-1} & \mathbf{P}^{xz}_{k|k-1} \\ \mathbf{P}^{zx}_{k|k-1} & \mathbf{P}^z_{k|k-1} \end{bmatrix} \right). \tag{3.1.2}$$

For the SSM with additive noise, given by eqs. (3.0.1a) and (3.0.1b), the predicted state mean and predicted state covariance are given in the form of expectations

$$\mathbf{m}^x_{k|k-1} = \mathbb{E}_{\mathbf{x}_{k-1}}[\mathbf{f}(\mathbf{x}_{k-1})], \tag{3.1.3a}$$

$$\mathbf{P}^x_{k|k-1} = \mathbb{E}_{\mathbf{x}_{k-1}}\left[\left(\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{m}^x_{k|k-1}\right)\left(\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{m}^x_{k|k-1}\right)^\top \right] + \mathbf{Q}_k, \tag{3.1.3b}$$

where $\mathbf{x}_{k-1} \sim \mathrm{N}\left(\mathbf{m}^x_{k-1|k-1}, \mathbf{P}^x_{k-1|k-1}\right)$. The measurement mean, measurement covariance and state-measurement cross-covariance are given by

$$\mathbf{m}^z_{k|k-1} = \mathbb{E}_{\mathbf{x}_k}[\mathbf{h}(\mathbf{x}_k)], \tag{3.1.4a}$$

$$\mathbf{P}^z_{k|k-1} = \mathbb{E}_{\mathbf{x}_k}\left[\left(\mathbf{h}(\mathbf{x}_k) - \mathbf{m}^z_{k|k-1}\right)\left(\mathbf{h}(\mathbf{x}_k) - \mathbf{m}^z_{k|k-1}\right)^\top \right] + \mathbf{R}_k, \tag{3.1.4b}$$

$$\mathbf{P}^{xz}_{k|k-1} = \mathbb{E}_{\mathbf{x}_k}\left[\left(\mathbf{x}_k - \mathbf{m}^x_{k|k-1}\right)\left(\mathbf{h}(\mathbf{x}_k) - \mathbf{m}^z_{k|k-1}\right)^\top \right], \tag{3.1.4c}$$

where $\mathbf{x}_k \sim \mathrm{N}\Big(\mathbf{m}^x_{k|k-1},\, \mathbf{P}^x_{k|k-1}\Big)$. Filtered state mean $\mathbf{m}^x_{k|k}$ and filtered state covariance $\mathbf{P}^x_{k|k}$ are determined by the conditioning formula for Gaussian densities

$$\mathbf{m}^x_{k|k} = \mathbf{m}^x_{k|k-1} + \mathbf{K}_k\big(\mathbf{z}_k - \mathbf{m}^z_k\big)^\top, \tag{3.1.5a}$$

$$\mathbf{P}^x_{k|k} = \mathbf{P}^x_{k|k-1} - \mathbf{K}_k\mathbf{P}^z_{k|k-1}\mathbf{K}_k^\top, \tag{3.1.5b}$$

$$\mathbf{K}_k = \mathbf{P}^{xz}_{k|k-1}\big(\mathbf{P}^z_{k|k-1}\big)^{-1}. \tag{3.1.5c}$$

The equations above define formal Gaussian filtering framework rather than a practical algorithm, because the moment integrals cannot be evaluated in closed-form. Equations for the filtered mean and covariance in eqs. (3.1.5a) to (3.1.5c) are reminiscent of the well-known Kalman filter. However, in the Kalman filter the moments given by eqs. (3.1.3a) to (3.1.4c) are computed analytically, because the functions in the system description in eqs. (3.0.1a) to (3.0.1b) are considered linear. Review of linear filtering theory can be found in the works of Sorenson [1970] and Kailath [1974].

The basic steps of a general nonlinear Gaussian filter are summarized by the following algorithm.

Algorithm 1: General Gaussian filter algorithm.

Input: Sequence of measurements $\big\{\mathbf{z}_k\big\}_{k=1}^K$, initial conditions $\mathbf{m}^x_{0|0}$, $\mathbf{P}^x_{0|0}$

Output: Sequence of state estimates and covariances $\big\{\mathbf{m}^x_{k|k}, \mathbf{P}^x_{k|k}\big\}_{k=1}^K$

for $k \leftarrow 1$ **to** K **do**

  `// state and measurement predictive moments`

  $\mathbf{m}^x_{k|k-1}$, $\mathbf{P}^x_{k|k-1} \leftarrow$ `MomentTransform(`$\mathbf{m}^x_{k-1|k-1}$, $\mathbf{P}^x_{k-1|k-1}$`)`

  $\mathbf{m}^z_{k|k-1}$, $\mathbf{P}^z_{k|k-1}$, $\mathbf{P}^{xz}_{k|k-1} \leftarrow$ `MomentTransform(`$\mathbf{m}^x_{k|k-1}$, $\mathbf{P}^x_{k|k-1}$`)`

  `// update`

  $\mathbf{K}_k \leftarrow \mathbf{P}^{xz}_{k|k-1}\big(\mathbf{P}^z_{k|k-1}\big)^{-1}$

  $\mathbf{m}^x_{k|k} \leftarrow \mathbf{m}^x_{k|k-1} + \mathbf{K}_k\big(\mathbf{z}_k - \mathbf{m}^z_k\big)^\top$

  $\mathbf{P}^x_{k|k} \leftarrow \mathbf{P}^x_{k|k-1} - \mathbf{K}_k\mathbf{P}^z_{k|k-1}\mathbf{K}_k^\top$

end

Moment transformations are responsible for computing predictive state and measurement moments in eqs. (3.1.3a) to (3.1.4c) and will be discussed in more detail in Section 3.2.

3.1.2 Student's t Filtering

The Gaussian assumption might become too restrictive, when the measurements are corrupted by outliers or the state disturbances are heavy-tailed. In such cases, the Student's t-distribution has become popular alternative for modeling heavy-tailed behavior, that contains the Gaussian distribution as a special limiting case. Student's t-filter for linear systems was presented by Roth et al. [2013], where it was found to increase robustness with respect to assumptions on the noise statistics in the sense of mean error. It was later extended to the non-linear and non-additive noise case by Tronarp et al. [2016].

Analogously to the Gaussian filter, the t-filter update equations [Barndorff-Nielsen et al., 1982; Roth et al., 2013] can be derived from the assumption that the state and the measurement are jointly Student's t-distributed, such that

$$\begin{bmatrix} \mathbf{x}_k \\ \mathbf{z}_k \end{bmatrix} \sim \mathrm{St}\left(\begin{bmatrix} \mathbf{m}^x_{k|k-1} \\ \mathbf{m}^z_{k|k-1} \end{bmatrix}, \begin{bmatrix} \mathbf{P}^x_{k|k-1} & \mathbf{P}^{xz}_{k|k-1} \\ \mathbf{P}^{zx}_{k|k-1} & \mathbf{P}^z_{k|k-1} \end{bmatrix}, \nu \right), \tag{3.1.6}$$

where, for convenience, I have chosen to parametrize the t-density by the covariance matrix (see Appendix A). For the general SSM in eqs. (3.0.2a) and (3.0.2b), the predicted state mean $\mathbf{m}^x_{k|k-1}$ and predicted state covariance $\mathbf{P}^x_{k|k-1}$ are computed as

$$\mathbf{m}^x_{k|k-1} = \mathbb{E}_{(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})}\big[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})\big], \tag{3.1.7a}$$

$$\mathbf{P}^x_{k|k-1} = \mathbb{E}_{(\mathbf{x}_{k-1}, \mathbf{q}_{k-1})}\Big[\big(\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) - \mathbf{m}^x_{k|k-1}\big)\big(\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) - \mathbf{m}^x_{k|k-1}\big)^\top\Big], \tag{3.1.7b}$$

with the input variable distributed according to

$$\begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{q}_{k-1} \end{bmatrix} \sim \mathrm{St}\left(\begin{bmatrix} \mathbf{m}^x_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}^x_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k \end{bmatrix}, \nu \right). \tag{3.1.8}$$

The same applies for the measurement mean $\mathbf{m}^z_{k|k-1}$, covariance $\mathbf{P}^z_{k|k-1}$ and cross-covariance $\mathbf{P}^{xz}_{k|k-1}$, which are computed as

$$\mathbf{m}^z_{k|k-1} = \mathbb{E}_{(\mathbf{x}_k, \mathbf{r}_k)}[\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k)], \tag{3.1.9a}$$

$$\mathbf{P}^z_{k|k-1} = \mathbb{E}_{(\mathbf{x}_k, \mathbf{r}_k)}\Big[\big(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \mathbf{m}^z_{k|k-1}\big)\big(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \mathbf{m}^z_{k|k-1}\big)^\top\Big], \tag{3.1.9b}$$

$$\mathbf{P}^{xz}_{k|k-1} = \mathbb{E}_{(\mathbf{x}_k, \mathbf{r}_k)}\Big[\big(\mathbf{x}_k - \mathbf{m}^x_{k|k-1}\big)\big(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \mathbf{m}^z_{k|k-1}\big)^\top\Big], \tag{3.1.9c}$$

with input variable distributed according to

$$\begin{bmatrix}\mathbf{x}_k \\ \mathbf{r}_k\end{bmatrix} \sim \mathrm{St}\left(\begin{bmatrix}\mathbf{m}^x \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{P}^x & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_k\end{bmatrix}, \nu\right). \tag{3.1.10}$$

Note that even though the state and noise are uncorrelated, under the assumption (3.1.6), they are still dependent. The filtered state is distributed according to a posterior density

$$\mathbf{x}_k \mid \mathbf{z}_{1:k} \sim \mathrm{St}\Big(\mathbf{m}^x_{k|k}, \mathbf{P}^x_{k|k}, \nu^x_{k|k}\Big), \tag{3.1.11}$$

with the statistics given by

$$\mathbf{m}^x_{k|k} = \mathbf{m}^x_{k|k-1} + \mathbf{P}^{xz}_{k|k-1}(\mathbf{P}^z_{k|k-1})^{-1}(\mathbf{z}_k - \mathbf{m}^z_{k|k-1}), \tag{3.1.12a}$$

$$\mathbf{P}^x_{k|k} = \frac{\nu - 2 + \beta}{\nu - 2 + d_z}\Big(\mathbf{P}^x_{k|k-1} - \mathbf{P}^{xz}_{k|k-1}(\mathbf{P}^z_{k|k-1})^{-1}\mathbf{P}^{zx}_{k|k-1}\Big), \tag{3.1.12b}$$

$$\beta = (\mathbf{z}_k - \mathbf{m}^z_{k|k-1})^\top(\mathbf{P}^z_{k|k-1})^{-1}(\mathbf{z}_k - \mathbf{m}^z_{k|k-1}), \tag{3.1.12c}$$

$$\nu^x_{k|k} = \nu + d_z. \tag{3.1.12d}$$

These equations constitute the t-filter measurement update rule, where $\mathbf{m}^x_{k|k}$ and $\mathbf{P}^x_{k|k}$ are the filtered state mean and covariance, respectively.

A little disconcerting feature of the update rule is highlighted by eq. (3.1.12d), which indicates that the asymptotic behavior of the Student's t-filter is indistinguishable from the Gaussian filter. With increasing DoF parameter $\nu^x_{k|k}$, the heavy-tailed behavior of Student's t-density is gradually diminished and the cherished resilience to outliers is eventually lost. To prevent this from occurring, a common solution [Roth et al., 2013; Tronarp et al., 2016] is to fix the DoF parameter to a predetermined value ν^*. In the algorithm below, we choose the moment matching approximation, so that

$$\mathrm{St}\Big(\mathbf{x}_k \,\Big|\, \mathbf{m}^x_{k|k}, \mathbf{P}^x_{k|k}, \nu^x_{k|k}\Big) \approx \mathrm{St}\Big(\mathbf{x}_k \,\Big|\, \mathbf{m}^x_{k|k}, \mathbf{P}^x_{k|k}, \nu^*\Big). \tag{3.1.13}$$

Algorithm 2: General Student's t-filter algorithm.

Input: Sequence of measurements $\{\mathbf{z}_k\}_{k=1}^K$, initial conditions $\mathbf{m}_{0|0}^x$, $\mathbf{P}_{0|0}^x$, ν^x and desired DoF ν^*

Output: Sequence of state estimates and covariances $\{\mathbf{m}_{k|k}^x, \mathbf{P}_{k|k}^x\}_{k=1}^K$

for $k \leftarrow 1$ **to** K **do**

`// state and measurement predictive moments`

$\mathbf{m}_{k|k-1}^x, \mathbf{P}_{k|k-1}^x \leftarrow$ `MomentTransform(`$\mathbf{m}_{k-1|k-1}^x, \mathbf{P}_{k-1|k-1}^x$`)`

$\mathbf{m}_{k|k-1}^z, \mathbf{P}_{k|k-1}^z, \mathbf{P}_{k|k-1}^{xz} \leftarrow$ `MomentTransform(`$\mathbf{m}_{k|k-1}^x, \mathbf{P}_{k|k-1}^x$`)`

`// state filtered moments`

$\mathbf{m}_{k|k}^x \leftarrow \mathbf{m}_{k|k-1}^x + \mathbf{P}_{k|k-1}^{xz}(\mathbf{P}_{k|k-1}^z)^{-1}(\mathbf{z}_k - \mathbf{m}_{k|k-1}^z)$

$\beta \leftarrow (\mathbf{z}_k - \mathbf{m}_{k|k-1}^z)^\top (\mathbf{P}_{k|k-1}^z)^{-1}(\mathbf{z}_k - \mathbf{m}_{k|k-1}^z)$

$\mathbf{P}_{k|k}^x \leftarrow \frac{\nu_{k|k-1}^x - 2 + \beta}{\nu_{k|k-1}^x - 2 + d_z}\Big(\mathbf{P}_{k|k-1}^x - \mathbf{P}_{k|k-1}^{xz}(\mathbf{P}_{k|k-1}^z)^{-1}\mathbf{P}_{k|k-1}^{zx}\Big)$,

`// fix DoF of the state posterior`

$\nu_{k|k}^x \leftarrow \nu^*$

end

3.2 Moment Transformations

From the above discussion, it is apparent that the core of the local nonlinear filtering is a moment transformation problem. Consider a nonlinearly transformed random variable

$$\mathbf{y} = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \sim p(\mathbf{x}), \tag{3.2.1}$$

where $\mathbf{g} : \mathbb{R}^D \to \mathbb{R}^E$ defines arbitrary nonlinear function, which can stand for either system dynamics $\mathbf{f}$ or measurement function $\mathbf{h}$, depending on which moments need to be computed. Non-additive noise can be easily subsumed into the above formulation by augmenting the state vector with the noise vector components. The goal of a moment transformation is to compute the output (transformed) moments

$$\boldsymbol{\mu} = \mathbb{E}_{\mathbf{x}}[\mathbf{g}(\mathbf{x})], \tag{3.2.2a}$$

$$\boldsymbol{\Pi} = \mathbb{E}_{\mathbf{x}}\Big[(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu})(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu})^\top\Big], \tag{3.2.2b}$$

$$\mathbf{C} = \mathbb{E}_{\mathbf{x}}\Big[(\mathbf{x} - \mathbf{m})(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu})^\top\Big], \tag{3.2.2c}$$

given the first two moments $\mathbf{m}$ and $\mathbf{P}$ of the input variable $\mathbf{x}$. Examining eqs. (3.2.2a) to (3.2.2c), it becomes evident, that all moments are just integrals of some nonlinear

function $\boldsymbol{\gamma}(\mathbf{x})$ of the form

$$\int \boldsymbol{\gamma}(\mathbf{x}) p(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{3.2.3}$$

which are intractable due to $\boldsymbol{\gamma}$ being nonlinear. Thus the whole moment transformation problem boils down to an integral approximation problem, which all local filters have to deal with, one way or another.

3.2.1 Linearization

The simplest way to approximate the moment integrals is by leveraging the Taylor series expansion around the input mean $\mathbf{m}$ to linearize $\mathbf{g}$. The first-order Taylor expansion of $\mathbf{g}$ in the neighborhood of $\mathbf{m}$ is given by

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}(\mathbf{m}) + \mathbf{G}(\mathbf{m})(\mathbf{x} - \mathbf{m}), \tag{3.2.4}$$

where $\mathbf{G}(\mathbf{m})$ denotes the Jacobian of $\mathbf{g}$ evaluated at $\mathbf{m}$. Since the approximation is valid only in the vicinity of the linearization point $\mathbf{m}$, it is not suitable for severe nonlinearities. Applying expectation and covariance operators yields approximations of the transformed moments

$$\boldsymbol{\mu} \approx \boldsymbol{\mu}_{\mathrm{A}} = \mathbf{g}(\mathbf{m}), \tag{3.2.5a}$$

$$\boldsymbol{\Pi} \approx \boldsymbol{\Pi}_{\mathrm{A}} = \mathbf{G}(\mathbf{m})\mathbf{P}\mathbf{G}(\mathbf{m})^{\top}, \tag{3.2.5b}$$

$$\mathbf{C} \approx \mathbf{C}_{\mathrm{A}} = \mathbf{P}\mathbf{G}(\mathbf{m})^{\top}. \tag{3.2.5c}$$

These equations define a general purpose moment transform, which can be viewed algorithmically as a procedure, where the input arguments are the mean $\mathbf{m}$ and covariance $\mathbf{P}$ of the input random variable $\mathbf{x}$ and the outputs are the approximations of the output mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Pi}$ and cross-covariance $\mathbf{C}$ of the transformed random variable $\mathbf{y}$. The linearization moment transform is summarized by the following algorithm

Algorithm 3: Linearization moment transform.

```
Function LinearizationMT(m, P)
    μ_A ← g(m)
    Π_A ← G(m)PG(m)^⊤
    C_A ← PG(m)^⊤
    return μ_A, Π_A, C_A
end
```

and can be utilized for all sorts of applications, one of which is local nonlinear filtering.

The *extended Kalman filter* (EKF) [Smith et al., 1962], originally developed at NASA, was the first attempt to deal with the estimation of nonlinear dynamics and can be considered a standard in navigation and tracking applications to this day. The EKF is obtained by replacing the place-holder `MomentTransform()` routine in the general Gaussian filter template, in Algorithm 1, with the linearization moment transform `LinearizationMT()`, in Algorithm 3. One limitation of the EKF is that, in case of severe nonlinearities, the error incurred by the first-order Taylor approximation may result in filter divergence.

The *iterated filter*, also called iterated extended Kalman filter (IEKF), employs a technique that improves the EKF performance in case of large approximation errors [Jazwinski, 1970]. The idea is to successively refine the filtered estimate by re-linearizing the measurement function in each iteration until maximum number of iterations is reached or the user-defined tolerance is attained. Bell and Cathey [1993] have shown that the IEKF iterates are identical to the Gauss-Newton method approximating the maximum likelihood estimate of the state.

The *second-order filter* [Athans et al., 1968; Jazwinski, 1970] is a natural progression of the EKF and uses the second-order Taylor expansion for the approximation of nonlinear functions in the state-space system description. Comparison of the EKF, IEKF and second-order filter was presented by Wishner et al. [1969]. Simon [2006] notes that "Although the second-order filter often provides improved performance over the extended Kalman filter, nothing definitive can be said about its performance" and refers to Kushner [1967], who showed an example of unstable second-order filter. The implementation for higher-dimensional systems can be cumbersome due to the large number of entries required in the Hessian matrices, although efficient implementations have also been developed [Roth and Gustafsson, 2011].

A notable disadvantage of Taylor approximation is that it requires differentiability of $\mathbf{g}$. The *divided difference filters* (DDF) overcome this limitation by using a simplified multivariate extension of the Stirling's interpolation formula, which can be viewed as the Taylor series expansion where the derivatives have been replaced by the central differences [Nørgaard et al., 2000]. Šimandl and Duník [2009] showed similarities between the DDFs and the sigma-point filters. The *central difference filter* (CDF), originally introduced in the context of the quadrature-based filters [Ito and Xiong, 2000], is a special case of the DDF filter.

3.2.2 Quadrature Approximations

A more principled approach to integral approximation is to utilize numerical quadrature rules, which are the basis for an entire class of sigma-point moment transformations.

The goal of quadrature[2] is to approximate intractable definite integrals of the form

$$\mathbb{I}[\mathbf{g}] \triangleq \int_{\Omega} \mathbf{g}(\mathbf{x}) w(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{3.2.6}$$

where $w(\mathbf{x}) \geq 0$ is the *weight function* and the integration domain Ω is the support of w, that is $\Omega = \operatorname{supp}[w(\mathbf{x})] = \{\mathbf{x} \in \mathbb{R}^D : w(\mathbf{x}) \neq 0\}$. For reasons of notational brevity, integration limits will be suppressed. The problem of numerical quadrature revolves around the design of suitable weights w_n and sigma-points[3] $\mathbf{x}_n$, such that the approximation

$$\mathbb{I}[\mathbf{g}] \simeq \mathbb{Q}[\mathbf{g}] \triangleq \sum_{n=1}^{N} w_n \mathbf{g}(\mathbf{x}_n) \tag{3.2.7}$$

achieves the smallest error with minimum number of function evaluations.

In local nonlinear filtering, we often contend with integrals where the weight function is either a Gaussian $w(\mathbf{x}) = \mathrm{N}(\mathbf{x} \,|\, \mathbf{m}, \mathbf{P})$ or a Student's t-density $w(\mathbf{x}) = \mathrm{St}(\mathbf{x} \,|\, \mathbf{m}, \mathbf{\Sigma}, \nu)$. Before numerical quadratures can be leveraged, the integrals need to be converted to a form where the $w(\mathbf{x})$ is standardized density. For integrals w.r.t. an arbitrary Gaussian, this is achieved by means of a stochastic decoupling substitution $\mathbf{x} = \mathbf{m} + \mathbf{L}\boldsymbol{\xi}$, where $\mathbf{L}$ is a matrix factor, such that $\mathbf{P} = \mathbf{L}\mathbf{L}^\top$. We get

$$\mathbb{I}[\mathbf{g}] \triangleq \int \mathbf{g}(\mathbf{x}) \mathrm{N}(\mathbf{x} \,|\, \mathbf{m}, \mathbf{P}) \, \mathrm{d}\mathbf{x} = \int \mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi}) \mathrm{N}(\boldsymbol{\xi} \,|\, \mathbf{0}, \mathbf{I}) \, \mathrm{d}\boldsymbol{\xi}. \tag{3.2.8}$$

The same can be achieved for Student's t-densities with substitution $\mathbf{x} = \mathbf{m} + \mathbf{\Lambda}\boldsymbol{\xi}$, where $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top$.

A common way to characterize the degree of accuracy of a quadrature rule is to look at the highest-degree of a (multivariate) polynomial, which can be integrated exactly (i.e. $|\mathbb{I}[\mathbf{g}] - \mathbb{Q}[\mathbf{g}]| = 0$). A *monomial* is a function of the form $\prod_{d=1}^{D} x_d^{\alpha_d}$, also denoted as $\mathbf{x}^{\boldsymbol{\alpha}}$, for multi-index $\boldsymbol{\alpha} \in \mathbb{N}^D$. A D-variate polynomial $a(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ is a finite linear

[2] The D-dimensional numerical integration rules are also referred to as cubatures. However, we will use a single term "quadrature" irrespective of the dimensionality of the integration domain, since the problems are fundamentally the same.

[3] Also known as abscissas or design points.

combination of monomials

$$a(\mathbf{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \tag{3.2.9}$$

where $\mathcal{A}$ is a finite set of multi-indices. Let us define the following two notions of degrees [Cools, 1997] for multivariate polynomials.

Definition 1. The *degree* of a multivariate polynomial is defined as

$$\deg a(\mathbf{x}) \triangleq \max \{|\boldsymbol{\alpha}| : \boldsymbol{\alpha} \in \mathcal{A}\},$$

where $|\boldsymbol{\alpha}| = \sum_{d=1}^{D} \alpha^d$.

Definition 2. The *overall degree* of a multivariate polynomial is defined as

$$\operatorname{pdeg} a(\mathbf{x}) \triangleq \max \{\max \{\alpha_1, \ldots, \alpha_D\} : \boldsymbol{\alpha} \in \mathcal{A}\}.$$

Definition 3. A quadrature rule $\mathbb{Q}[\mathbf{g}]$ for an integral $\mathbb{I}[\mathbf{g}]$ has a degree r if it is exact for all polynomials of degree at most r and is not exact for at least one polynomial of degree $r + 1$.

The various sigma-point moment transforms are obtained when the moment integrals in eqs. (3.2.2a) to (3.2.2c) are approximated by the quadrature in eq. (3.2.7). The transformed moments for any sigma-point transform can be written in a general form using matrix notation as

$$\boldsymbol{\mu}_{\mathrm{A}} = \mathbf{Y}^{\top} \mathbf{w}, \tag{3.2.10a}$$

$$\boldsymbol{\Pi}_{\mathrm{A}} = (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})^{\top} \mathbf{W} (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}}), \tag{3.2.10b}$$

$$\mathbf{C}_{\mathrm{A}} = (\mathbf{X} - \mathbf{m})^{\top} \mathbf{W}_c (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}}), \tag{3.2.10c}$$

where $\mathbf{w}$, $\mathbf{W}$ and $\mathbf{W}_c$ are the quadrature weights, $\left[\mathbf{Y}^{\top}\right]_{*n} = \mathbf{g}(\mathbf{x}_n)$, $\left[(\mathbf{X} - \mathbf{m})^{\top}\right]_{*n} = \mathbf{x}_n - \mathbf{m}$ and $\left[(\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})^{\top}\right]_{*n} = \mathbf{g}(\mathbf{x}_n) - \boldsymbol{\mu}_{\mathrm{A}}$, and where $[\ \ ]_{*n}$ denotes the n-th column. Specific transforms differ in the way they define the weights for each moment and the placement of sigma-points. Significant research effort has been devoted to development of various sigma-point moment transforms over the years, which were mainly applied for construction of local nonlinear sigma-point filtering algorithms. Below, I list the most significant moment transforms and mention the filters which utilize them.

Unscented Transform

The *Unscented transform* (UT) [Julier and Uhlmann, 1996] was originally conceived of as a method for approximation of the input densities with deterministically chosen set of sigma-points. It later became recognized as a special case of the fully-symmetric quadrature rules [McNamee and Stenger, 1967].

The UT uses $N = 2D + 1$ sigma-points $\mathbf{x}_n = \mathbf{m} + \mathbf{L}\boldsymbol{\xi}_n$, where $D = \dim(\mathbf{x})$ and the unit points are defined as

$$\begin{bmatrix} \boldsymbol{\xi}_0 & \boldsymbol{\xi}_1 & \dots & \boldsymbol{\xi}_N \end{bmatrix} = \begin{bmatrix} \mathbf{0} & c\mathbf{I}_D & -c\mathbf{I}_D \end{bmatrix}, \tag{3.2.11}$$

where $c = \sqrt{D + \lambda}$ and λ is a parameter. The Figure 3.2.1 illustrates the placement of the points. The UT weights are defined as [Van der Merwe and Wan, 2001]

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \dots & w_N \end{bmatrix}^\top, \tag{3.2.12}$$

$$\mathbf{W} = \mathbf{W}_c = \operatorname{diag}\Big(\begin{bmatrix} w_0^c & w_1^c & \dots & w_N^c \end{bmatrix} \Big), \tag{3.2.13}$$

where for $n = 1, \dots, N$

$$w_0 = \frac{\lambda}{D + \lambda}, \quad w_n = \frac{1}{2(D + \lambda)}, \tag{3.2.14}$$

$$w_0^c = \frac{\lambda}{D + \lambda} + (1 - \alpha^2 + \beta), \quad w_n^c = w_n, \tag{3.2.15}$$

and the scaling parameter $\lambda = \alpha^2(D + \kappa) - D$. The parameters α and β were introduced by Julier [2002], where α determines the spread of sigma-points around the mean of $\mathbf{x}$ (usually set to a small positive value) and β is used to incorporate prior knowledge of the distribution of $\mathbf{x}$, (for the Gaussian distribution $\beta = 2$ is optimal). The parameter κ "provides an extra degree of freedom to 'fine tune' the higher order moments of the approximation, and can be used to reduce the overall prediction errors." [Julier et al., 2000]. When $\mathbf{x}$ is assumed Gaussian a good heuristic is setting $\kappa = 3 - d_x$. For state dimension $d_x > 3$, the κ is negative, which can lead to numerical instability, because the computed covariance matrices may lose positive definiteness. The *scaled UT* [Julier, 2002] was proposed to alleviate this problem.

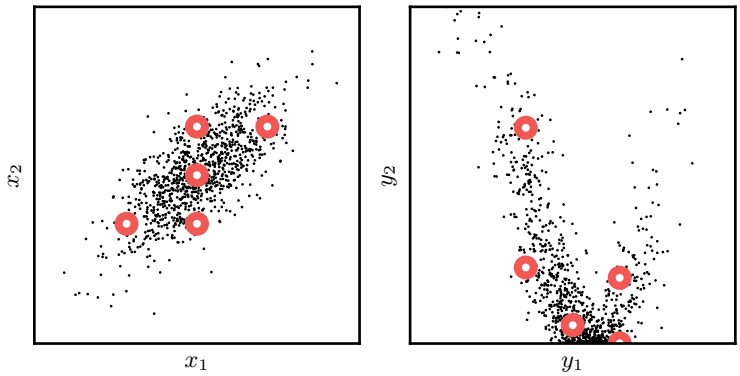

Figure 3.2.1: Left panel: samples from a Gaussian density (dots) with UT sigma-points (circles); Right panel: nonlinear transformation of the Gaussian distributed samples on the left and the corresponding sigma-point representation (circles).

Algorithm 4: Unscented moment transform.

Function `UnscentedMT`$(\mathbf{m}, \mathbf{P}, \kappa, \alpha, \beta)$

$D \leftarrow \dim(\mathbf{m})$

$\lambda \leftarrow \alpha^2(D+\kappa) - D$

$c \leftarrow \sqrt{D+\lambda}$

`// compute UT weights`

$\mathbf{w} \leftarrow \left[\frac{\lambda}{(D+\lambda)} \quad \frac{1}{2(D+\lambda)}\mathbf{1}_{2D}^\top\right]$

$\mathbf{W} \leftarrow \mathrm{diag}\Big(\left[\frac{\lambda}{D+\lambda} + 1 - \alpha^2 + \beta \quad \frac{1}{2(D+\lambda)}\mathbf{1}_{2D}^\top\right]\Big)$

$\mathbf{W}_c \leftarrow \mathbf{W}$

`// form UT sigma-points`

$\boldsymbol{\Xi} \leftarrow \left[\mathbf{0} \quad c\mathbf{I}_D \quad -c\mathbf{I}_D\right]$

$\mathbf{L} \leftarrow$ `MatrixFactor`$(\mathbf{P})$

$\mathbf{X} \leftarrow \mathbf{m} + \mathbf{L}\boldsymbol{\Xi}$

`// evaluate nonlinearity at sigma-points`

$\mathbf{Y} \leftarrow \mathbf{g}(\mathbf{X})$

`// compute transformed moments`

$\boldsymbol{\mu}_\mathrm{A} \leftarrow \mathbf{Y}^\top\mathbf{w}$

$\boldsymbol{\Pi}_\mathrm{A} \leftarrow (\mathbf{Y} - \boldsymbol{\mu}_\mathrm{A})^\top\mathbf{W}(\mathbf{Y} - \boldsymbol{\mu}_\mathrm{A})$

$\mathbf{C}_\mathrm{A} \leftarrow (\mathbf{X} - \mathbf{m})^\top\mathbf{W}_c(\mathbf{Y} - \boldsymbol{\mu}_\mathrm{A})$

return $\boldsymbol{\mu}_\mathrm{A}, \boldsymbol{\Pi}_\mathrm{A}, \mathbf{C}_\mathrm{A}$

end

The *unscented Kalman filter* (UKF) [Julier and Uhlmann, 2004; Van der Merwe and Wan, 2003] is obtained by replacing the `MomentTransform()` routine in Algorithm 1 with the unscented transform `UnscentedMT()`, in Algorithm 4. In the original publication [Julier and Uhlmann, 2004], the UKF is contrasted with the EKF and shown to provide superior performance in tracking and navigation applications. Advantage of the UKF over the EKF is that it does not require calculations of the Jacobian matrices, which is often tedious and error-prone process.

Over the years, the UKF has received much treatment in the research community and many filters contain it as a special case. Gustafsson [2010] has shown that the mean calculated by the UKF converges to that of the second-order EKF. The scaled UT [Julier, 2002] introduces additional parameters that allow for increased accuracy in the covariance computations. The reduced sigma-point filters use $d_x + 1$ points [Julier and Uhlmann, 2002] positioned at the vertices of an d_x-dimensional simplex. Van der Merwe and Wan [2003] gave an efficient square-root forms of the sigma-point filters, which, instead of updating the covariance matrix, update the Cholesky factor of the covariance matrix. The stochastic integration filter (SIF) [Duník et al., 2013] based on the third-order spherical-radial stochastic integration rule is also called randomized UKF (RUKF). Straka et al. [2014] compared the RUKF with an adaptive UKF (AUKF), which chooses the scaling and rotation of the sigma-points adaptively. Turner and Rasmussen [2010] learned the placement of sigma-points from data by a Gaussian process optimization of the log marginal likelihood of the UKF parameters. Steinbring and Hanebeck [2013] developed smart sampling KF (S^2KF), which includes the UKF as a special case, and uses a low-discrepancy Dirac mixture approximation of Gaussian densities. According to Steinbring and Hanebeck [2013]: "This approximation comprises an arbitrary number of optimally and deterministically placed samples in the relevant regions of the state space, so that the filter resolution can be adapted to either achieve high-quality results or to meet computational constraints." Similar theme is developed in [Kurz and Hanebeck, 2017]. Sandblom and Svensson [2011] designed a marginalized transform, which is based on modeling the transforming nonlinear function by a parametric Hermite polynomial model. The function model is given a Bayesian treatment and a prior distribution is placed on the model parameters, which are then integrated out during the computation of moments of the transformed distribution. This approach is related to the use of Gaussian process quadrature in sigma-point filtering [Särkkä et al., 2014].

Spherical-Radial Transform

The *spherical-radial transform* (SRT) was proposed by [Arasaratnam and Haykin, 2009] as a more numerically stable alternative to the UT that would perform better in higher dimensions. The transform is based on the third-degree spherical-radial quadrature rule proposed by Genz and Monahan [1996] and can also be seen as an instance of the fully symmetric rule [McNamee and Stenger, 1967]. The main trick employed by the SRT is conversion of the integral over a symmetric weight function (e.g. Gaussian density) from Cartesian coordinates into the spherical-radial coordinate system, which allows to exploit the weight function symmetry and ultimately reduce the number of sigma-points to $N = 2D$. The unit sigma-points are given by

$$\boldsymbol{\Xi} = \left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -1 \end{bmatrix} \right\} \tag{3.2.16}$$

with the quadrature weights defined as

$$w_n = \frac{1}{2D}, \quad \text{for} \quad n = 1, \ldots N. \tag{3.2.17}$$

The placement of SR points is shown in Figure 3.2.2. It is now evident that for $\kappa = 0$, $\alpha = 0$ and $\beta = 0$ the UT reduces to the SRT. Both transforms integrate multivariate polynomials $\{a(\mathbf{x}) : \deg a(\mathbf{x}) \leq 3\}$ exactly.

The SRT is the basis of the *cubature Kalman filter* (CKF), which can be obtained by replacing the general `MomentTransform()` in Algorithm 1 with the `SphericalRadialMT()` routine in Algorithm 5. In the original publication [Arasaratnam and Haykin, 2009] the CKF is presented as a superior alternative to the UKF mainly for the numerical accuracy and stability reasons. It is shown that for state dimensions $d_x > 3$ the stability factor of the UT, given by $\sum_n |w_n| / \sum_n w_n$, is greater than one, which signifies perturbations in numerical estimates of the moment integrals. Arasaratnam and Haykin [2009] debated the use of higher-order rules and concluded that the use of such rule in the design of CKF "may marginally improve its performance at the expense of a reduced numerical stability and an increased computational cost."

Algorithm 5: Spherical-radial moment transform.

Function `SphericalRadialMT(`$\mathbf{m}$, $\mathbf{P}$`)`
- $D \leftarrow \dim(\mathbf{m})$
- $c \leftarrow \sqrt{D}$
- `// compute SRT weights`
- $\mathbf{w} \leftarrow \frac{1}{2D}\mathbf{1}_{2D}^{\top}$
- $\mathbf{W} \leftarrow \mathrm{diag}(\,\mathbf{w}\,)$
- $\mathbf{W}_c \leftarrow \mathbf{W}$
- `// form SRT sigma-points`
- $\boldsymbol{\Xi} \leftarrow \begin{bmatrix} c\mathbf{I}_D & -c\mathbf{I}_D \end{bmatrix}$
- $\mathbf{L} \leftarrow$ `MatrixFactor(`$\mathbf{P}$`)`
- $\mathbf{X} \leftarrow \mathbf{m} + \mathbf{L}\boldsymbol{\Xi}$
- `// evaluate nonlinearity at sigma-points`
- $\mathbf{Y} \leftarrow \mathbf{g}(\mathbf{X})$
- `// compute transformed moments`
- $\boldsymbol{\mu}_{\mathrm{A}} \leftarrow \mathbf{Y}^{\top}\mathbf{w}$
- $\boldsymbol{\Pi}_{\mathrm{A}} \leftarrow (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})^{\top}\mathbf{W}(\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})$
- $\mathbf{C}_{\mathrm{A}} \leftarrow (\mathbf{X} - \mathbf{m})^{\top}\mathbf{W}_c(\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})$
- **return** $\boldsymbol{\mu}_{\mathrm{A}}, \boldsymbol{\Pi}_{\mathrm{A}}, \mathbf{C}_{\mathrm{A}}$

end

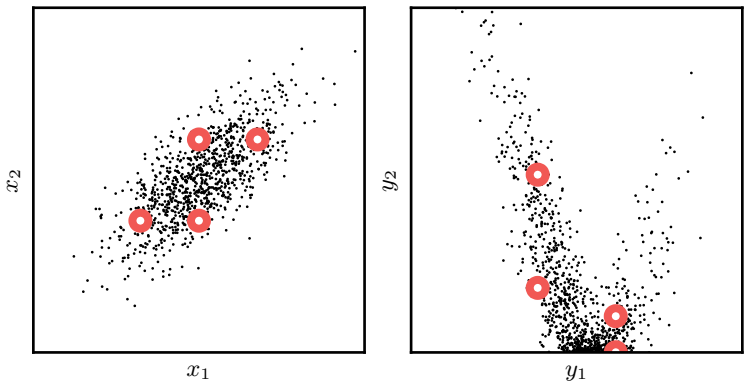

Figure 3.2.2: Left panel: samples from a Gaussian density (dots) with SR sigma-points (circles); Right panel: nonlinear transformation of the Gaussian distributed samples on the left and the corresponding sigma-point representation (circles).

Gauss-Hermite Transform

The *Gauss-Hermite transform* (GHT) is based on the well-known Gauss-Hermite quadrature rule [Gautschi, 2004] for approximating integrals with standard Gaussian weight

function. For one-dimensional integration domain the approximation is of the form

$$\int g(\xi)\mathrm{N}(\,\xi\,|\,0,\,1\,)\,\mathrm{d}\xi \approx \sum_{n=1}^{p} w_n\, g(\xi_n), \tag{3.2.18}$$

where the unit sigma-points ξ_n are chosen as the roots of the p-th degree Hermite polynomial, which can be generated by Rodrigues' formula

$$H_p(\xi) = (-1)^p \exp\left(\frac{\xi^2}{2}\right) \frac{\mathrm{d}^p}{\mathrm{d}\xi^p} \exp\left(-\frac{\xi^2}{2}\right) \tag{3.2.19}$$

and quadrature weights can be computed in closed form using the formula

$$w_n = \frac{p!}{p^2[H_{p-1}(\xi_n)]^2} \quad \text{for} \quad n = 1, \ldots, p. \tag{3.2.20}$$

Choosing the sigma-points in this way guarantees that all polynomials of the degree at most $2p-1$ will be integrated exactly.

The D-dimensional quadrature rule is constructed from one-dimensional sigma-points by the Cartesian product, which yields

$$\boldsymbol{\xi}^{(i_1, i_2, \ldots, i_D)} = \begin{bmatrix} \xi^{(i_1)} & \xi^{(i_2)} & \ldots & \xi^{(i_D)} \end{bmatrix}^\top, \tag{3.2.21}$$

for $i_d = 1, \ldots, p$ and $d = 1, \ldots, D$. This extension mechanism to higher-dimensional domains comes with a drawback. Namely, that given a p-th order quadrature rule, the number of sigma-points grows exponentially with the dimension as $N = p^D$. Figure 3.2.3 shows the sigma-point placement of the 5-th order Gauss-Hermite quadrature rule in two dimensions. Due to the larger number of points, the transformed distribution is represented more accurately than in the case of the UT (cf. Figure 3.2.1) at the expense of increased computational requirements. The weights for D-dimensional quadrature are constructed by multiplication of the one-dimensional weights

$$w_{i_1, i_2, \ldots, i_{d_x}} = w_{i_1} \times w_{i_2} \times \cdots \times w_{i_{d_x}} \tag{3.2.22}$$

Finally, the D-dimensional integral is approximated as

$$\int \mathbf{g}(\boldsymbol{\xi})\mathrm{N}(\,\boldsymbol{\xi}\,|\,\mathbf{0},\,\mathbf{I}\,)\,\mathrm{d}\boldsymbol{\xi} \approx \sum_{i_1=1}^{p} \sum_{i_2=1}^{p} \cdots \sum_{i_D=1}^{p} w_{i_1, i_2, \ldots, i_n}\, \mathbf{g}\big(\boldsymbol{\xi}^{(i_1, i_2, \ldots, i_D)}\big). \tag{3.2.23}$$

The p-th order GH rule integrates all multivariate polynomials $\{a(\mathbf{x}) : \operatorname{pdeg} a(\mathbf{x}) \leq 2p-1\}$

exactly.

Algorithm 6: Gauss-Hermite moment transform.

Function `GaussHermiteMT`($\mathbf{m}$, $\mathbf{P}$, p)

  `// form GHT sigma-points`

  $\xi_1, \ldots, \xi_p \leftarrow$ `RootHermite`(p)

  $\boldsymbol{\Xi} \leftarrow$ `CartesianPoints`$(\xi_1, \ldots, \xi_p)$

  $\mathbf{L} \leftarrow$ `MatrixFactor`$(\mathbf{P})$

  $\mathbf{X} \leftarrow \mathbf{m} + \mathbf{L}\boldsymbol{\Xi}$

  `// compute GHT weights`

  $w_1, \ldots, w_p \leftarrow \frac{p!}{p^2[H_{p-1}(\xi_1)]^2}, \ldots, \frac{p!}{p^2[H_{p-1}(\xi_p)]^2}$

  $\mathbf{w} \leftarrow$ `CartesianWeigths`$(w_1, \ldots, w_p)$

  $\mathbf{W} \leftarrow \mathrm{diag}(\mathbf{w})$

  $\mathbf{W}_c \leftarrow \mathbf{W}$

  `// evaluate nonlinearity at sigma-points`

  $\mathbf{Y} \leftarrow \mathbf{g}(\mathbf{X})$

  `// compute transformed moments`

  $\boldsymbol{\mu}_{\mathrm{A}} \leftarrow \mathbf{Y}^\top \mathbf{w}$

  $\boldsymbol{\Pi}_{\mathrm{A}} \leftarrow (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})^\top \mathbf{W} (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})$

  $\mathbf{C}_{\mathrm{A}} \leftarrow (\mathbf{X} - \mathbf{m})^\top \mathbf{W}_c (\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}})$

  return $\boldsymbol{\mu}_{\mathrm{A}}, \boldsymbol{\Pi}_{\mathrm{A}}, \mathbf{C}_{\mathrm{A}}$

end

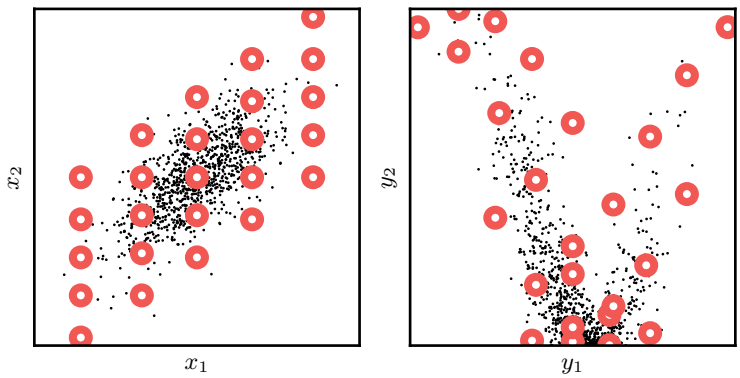

Figure 3.2.3: Left panel: samples from a Gaussian density (dots) with Gauss-Hermite sigma-points (circles); Right panel: nonlinear transformation of the Gaussian distributed samples on the left and the corresponding sigma-point representation (circles).

For single dimensional systems, the 3rd-order GH quadrature rule coincides with

the UT sigma-point rule, when $\kappa = 2, \alpha = 1, \beta = 0$. The main difference between the two algorithms is in the way they choose the locations of sigma-points and generalize to higher-dimensions. Indeed, the UT sigma-points grow linearly with the dimension $(2D + 1)$, whereas the number of GH quadrature sigma-points grows exponentially as a result of the Cartesian product construction. This is often referred to as the *curse of dimensionality*, which prohibits its use in real-world problems for dimensions higher than 5 or 6 [Wu et al., 2006]. The problem has not gone untreated in the research community and several solutions have emerged. Well-known non-product rules include Monte Carlo [Doucet et al., 2001], quasi-Monte Carlo [Guo and Wang, 2006] and the sparse-grid methods [Gerstner and Griebel, 1998; Smolyak, 1963].

3.3 Bayesian Quadrature

The early idea of Bayesian quadrature (BQ) can be traced all the way back to Poincaré's publication in 1896. The work was then later picked up by Diaconis [1988] and O'Hagan [1992]. The key feature that distinguishes the BQ from the classical quadrature is that the computation of integrals is treated as a problem of Bayesian statistical inference. This involves specifying a prior distribution over the value of the integral and inferring the posterior by conditioning on the sampled values of the function. When the integral

$$\mathbb{I}[g(\mathbf{x})] = \int g(\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x} = \mathbb{E}_{\mathbf{x}}[g(\mathbf{x})] \tag{3.3.1}$$

is viewed as a linear operator acting on a function $g(\mathbf{x})$, the distribution over the integral is induced by specifying a stochastic process model over the functions. At first the idea of introducing uncertainty over the integrand may seem strange. Consider, however, that even though the function is known, any quadrature will only ever work with finite number of function evaluations $g(\mathbf{x}_n)$ at selected sigma-points $\mathbf{x}_n$ - at any other point, the function is effectively unknown. Whereas the classical treatment of the quadrature is unable to reflect this functional uncertainty, the Bayesian treatment offers a natural resolution. In principle, any probabilistic model of the integrand could be used as a prior over the space of functions.

3.3.1 Gaussian Process Quadrature

Gaussian processes were the first to be considered as priors over functions in the BQ setting for their favorable analytical properties and they remain to be studied most

extensively to this day. In the early work, Diaconis [1988] used the Wiener process as a prior. Later, O'Hagan [1991] used a GP with the RBF covariance and called his method Bayes-Hermite quadrature. Throughout this thesis, I will refer to this family of methods as *Gaussian process quadrature* (GPQ) [Särkkä et al., 2014], which I view as a member of a broader family of BQ methods.

Applying an integral operator on a GP distributed function results in a Gaussian distribution over the value of the definite integral[4]. The Figure 3.3.1 provides a simulation-based view of the GPQ. A function is drawn from a GP posterior, integrated and the value of the integral is plotted. Repeating this process reveals that the results of integration are Gaussian distributed.

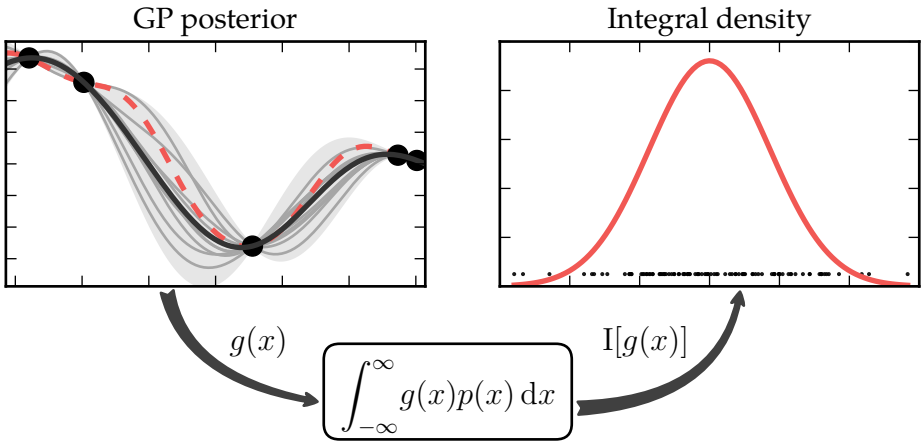

Figure 3.3.1: Simulation-based view of GP quadrature. A function is drawn from a GP posterior and integrated. Values of the integral have a Gaussian density. Left: plot of GP posterior mean (solid black) and the true function (dashed red). Quadrature only sees the true function through a set of evaluations (black dots) which implies functional uncertainty (gray area) in between the sigma-points. Right: integral values of various realizations of the GP posterior (black dots) have a Gaussian density (solid red).

The posterior mean of the integral is

$$\begin{aligned}\mathbb{E}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] &= \iint g(\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x}\,p(\,g\,|\,\mathcal{D}\,)\,\mathrm{d}g \\ &= \int\left[\int g(\mathbf{x})p(\,g\,|\,\mathcal{D}\,)\,\mathrm{d}g\right]p(\mathbf{x})\,\mathrm{d}\mathbf{x} = \mathbb{E}_{\mathbf{x}}\big[\mathbb{E}_{g|\mathcal{D}}[g(\mathbf{x})]\big],\end{aligned} \tag{3.3.2}$$

[4]This is analogous to the invariance of a Gaussian under affine transformations.

which shows that computing the posterior mean of the integral is, in fact, equivalent to integrating GP posterior mean [Rasmussen and Ghahramani, 2003], which acts as an approximation of the integrand. The GP regression is much more flexible model for approximation of arbitrary nonlinear integrands than Hermite polynomial series used in the Gauss-Hermite quadrature. The BQ is thus expected to perform better on average, when integrating a wider class of functions in comparison with classical quadrature rules, which are designed to perform with zero error on a narrow class of functions. The posterior variance of the integral is

$$\begin{aligned}\mathbb{V}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] &= \int \big[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})] - \mathbb{E}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]]\big]^2 p(\,g \,|\, \mathcal{D}\,)\,\mathrm{d}g \\ &= \iiint [g(\mathbf{x}) - \bar{g}(\mathbf{x})]\big[g(\mathbf{x}') - \bar{g}(\mathbf{x}')\big] p(\,g \,|\, \mathcal{D}\,)\,\mathrm{d}g\, p(\mathbf{x}) p(\mathbf{x}')\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}' \\ &= \iint \mathbb{C}_{g|\mathcal{D}}\big[g(\mathbf{x}), g(\mathbf{x}')\big] p(\mathbf{x}) p(\mathbf{x}')\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}', \end{aligned} \tag{3.3.3}$$

where the GP posterior mean and the GP posterior covariance are given by

$$\mathbb{E}_{g|\mathcal{D}}[g(\mathbf{x})] = \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{f} \triangleq \bar{g}(\mathbf{x}), \tag{3.3.4a}$$

$$\mathbb{C}_{g|\mathcal{D}}\big[g(\mathbf{x}), g(\mathbf{x}')\big] = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}'). \tag{3.3.4b}$$

These equations are special cases of eqs. (2.2.10a) and (2.2.10b) for zero observation noise, because in the quadrature settings we assume that the integrand can be evaluated (perfectly observed)[5]. More specifically, for $p(\mathbf{x}) = \mathrm{N}(\,\mathbf{x} \,|\, \mathbf{m},\ \mathbf{P}\,)$ and radial basis function (RBF) kernel

$$k(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\Big(-\tfrac{1}{2}(\mathbf{x} - \mathbf{x}')^\top \mathbf{\Lambda}^{-1}(\mathbf{x} - \mathbf{x}')\Big), \tag{3.3.5}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\,\begin{bmatrix}\lambda_1^2 & \ldots & \lambda_D^2\end{bmatrix}\,)$, the integrals in eqs. (3.3.2) and (3.3.3) can be evaluated

[5]Of course, nothing prevents us from modeling the function evaluations as noisy observations. In practice a small perturbation diagonal term is added for numerical stability when inverting the kernel matrix $\mathbf{K}$.

analytically[6], which results in posterior integral mean and variance

$$\mathbb{E}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \int \mathbf{k}(\mathbf{x})^\top \mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}\,)\, \mathrm{d}\mathbf{x}\, \mathbf{K}^{-1}\mathbf{f} = \mathbf{q}^\top \mathbf{K}^{-1}\mathbf{f}, \tag{3.3.6a}$$

$$\begin{aligned}\mathbb{V}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] &= \iint \Big[k(\mathbf{x},\mathbf{x}') - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}')\Big] \mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}\,) \mathrm{N}\big(\mathbf{x}' \,\big|\, \mathbf{m},\, \mathbf{P}\,\big)\; \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{x}' \\ &= \alpha^2 \big|2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\big|^{-1/2} - \mathbf{q}^\top \mathbf{K}^{-1}\mathbf{q},\end{aligned} \tag{3.3.6b}$$

where

$$[\mathbf{q}]_n = \alpha^2 \big|\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\big|^{-1/2} \exp\Big(-\tfrac{1}{2}(\mathbf{x}_n - \mathbf{m})^\top (\boldsymbol{\Lambda} + \mathbf{P})^{-1}(\mathbf{x}_n - \mathbf{m})\Big). \tag{3.3.7}$$

Defining $\mathbf{w}^\top = \mathbf{q}^\top \mathbf{K}^{-1}$ one can see that the expression for posterior mean of the integral in eq. (3.3.6a) has a form of weighted sum of function evaluations

$$\mathbb{E}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \mathbf{w}^\top \mathbf{f} = \sum_n w_n g(\mathbf{x}_n), \tag{3.3.8}$$

which is the same structure used by the classical quadrature rules in Section 3.2.2. Note that BQ does not prescribe the sigma-points $\mathbf{x}_n$. This is an additional degree of freedom not found in classical treatments of quadrature and presents an opportunity for further investigation.

One way of choosing the sigma-points would be to minimize the posterior integral variance in eq. (3.3.3) as was done by Minka [2000]. Another contribution by Huszár and Duvenaud [2012] elaborated on the equivalence of the BQ and weighted kernel herding, which is a deterministic method for choosing samples to summarize a probability distribution. Gunter et al. [2014] proposed an active learning approach for choosing sigma-points in the BQ so as to maximize the information gain from every sample. Karvonen and Särkkä [2017b] showed that the BQ weights can be computed efficiently and exactly if the density, kernel and the integration domain are fully symmetric and the sigma-point set is a union of fully symmetric point sets. For certain choices of the kernel and the sigma-points, the BQ reduces to classical quadrature rules [Karvonen and Särkkä, 2017a] and the posterior integral variance collapses to zero [Särkkä et al., 2016]. Theoretical analysis of the BQ, including the convergence proofs and behavior in misspecified settings can be found in [Bach, 2017; Briol et al., 2015; Kanagawa et al., 2016]. Connections between linearization and BQ approximation with derivative observations

[6] This also holds for Gaussian mixture densities as well as other types of kernels (e.g. polynomial).

are discussed in [Prüher and Särkkä, 2016].

3.3.2 Student's t-Process Quadrature

In Section 2.3, I mentioned the advantage of the Student's t-process over the GP. Namely, the dependence of the predictive variance on function values, which can result in more accurate and robust description of functional uncertainty, especially in low data regimes. Since the quadrature seeks to restrict the number of points, for computational reasons, I believe the t-process is ideally suited to be applied in the BQ setting.

Analogously to the GP case, the t-distribution is affine invariant, which implies that a TP posterior distribution over the integrand $g(\mathbf{x})$ induces a t-distribution over the integral $\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})] = \int g(\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x}$ with mean and variance given by

$$\mathbb{E}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \mathbf{q}^\top \mathbf{K}^{-1}\mathbf{y}, \tag{3.3.9a}$$

$$\mathbb{V}_{g|\mathcal{D}}[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \frac{\nu_g - 2 + \mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}}{\nu_g - 2 + N}\Big[\mathbb{E}_{\mathbf{x},\mathbf{x}'}\big[k(\mathbf{x},\mathbf{x}')\big] - \mathbf{q}^\top \mathbf{K}^{-1}\mathbf{q}\Big], \tag{3.3.9b}$$

where $\mathbf{q} = \mathbb{E}_{\mathbf{x}}\big[\mathbf{k}(\mathbf{x})\big]$. The mean is identical to the GPQ case, whereas the integral variance is now scaled by a data-dependent term in comparison. Integrals of general vector functions $\mathbf{g}\colon \mathbb{R}^D \to \mathbb{R}^E$ can be evaluated by applying the above equations to each function output independently with the same kernel and DoF.

Chapter 4

Thesis Goals

The main limitations of the GP regression model in the context of system identification derive mainly from the sequential nature of the data. The model in its original form is simply not suited for recursive processing, because as the dataset size grows with time, the demands for inversion of the kernel matrix grow indefinitely; not to mention the fact, that at each time step, we need to perform iterative kernel parameter optimization. The RGP algorithm appears to be a promising candidate for alleviating these ills, however a way to bound the per-iteration computational cost of the kernel parameter optimization still remains to be found. Hence, the first set of goals of this thesis revolves around leveraging advantages of the Gaussian process regression for recursive system identification.

One of the common applications of moment transforms are the local nonlinear filters, which are often plagued by overly optimistic covariance estimates. I postulate this is due to the leaking integration error in computation of the transformed moments in eqs. (3.2.2a) to (3.2.2c), which the classical moment transforms cannot account for. I do, however, recognize the fact that errors in local filters have other sources as well (chief among them being the joint Gaussianity assumption). The Bayesian statistical view of quadrature is a very promising avenue to pursue in this regard, because the integration error is treated probabilistically and thus could be more easily reflected in the moment transformation process. Since the integral variance in BQ characterizes the integration error, it is natural to ask how it could be decreased. One potential way would be to incorporate observations of the derivative of the integrand. I also suspect that, for a certain special cases of point-sets, theoretical connections could be found between the BQ with derivatives and the classical linearization based on the Taylor series. This is why, the main theme for the second set of goals, is investigation of how the statistical view of numerical quadrature could be used to improve the current state-of-the-art

moment transformations.

To summarize, the goals for this thesis are the following:

Goal 1. Leverage advantages of the Gaussian process regression for recursive system identification.

(a) Investigate suitability of the RGP algorithm for recursive system identification as means for decreasing computational demands of the original GP model.

(b) Bound the computational demands for kernel parameter optimization.

Goal 2. Improve the current state-of-the-art moment transformations with the help of Bayesian quadrature.

(a) Leverage the statistical view of quadrature for the design of general purpose moment transformations.

(b) Incorporate the approximation error in the output mean, in eq. (3.2.2a), into the moment transformation process.

(c) Use the proposed moment transformations to improve estimate quality of the nonlinear sigma-point filters; especially in terms of credibility of the covariance estimates.

Goal 3. Explore the question of derivative information in BQ as means for reducing the integral variance and attempt to find connections with classical linearization.

Chapter 5

System Identification with GP Regression Models

This chapter draws together contributions from the following articles, which are mainly concerned with the use of GP regression models for black-box non-linear system identification.

- L. Král, J. Prüher, and M. Šimandl. Gaussian Process Based Dual Adaptive Control of Nonlinear Stochastic Systems. In *22nd Mediterranean Conference of Control and Automation (MED)*, pages 1074–1079, 2014. doi: 10.1109/MED.2014.6961517
- J. Prüher and M. Šimandl. Gaussian process based recursive system identification. *Journal of Physics: Conference Series*, 570(1):1–9, 2014. doi: 10.1088/1742-6596/570/1/012002
- J. Prüher and L. Král. Functional Dual Adaptive Control with Recursive Gaussian Process Model. *Journal of Physics: Conference Series*, 659:012006, 2015. ISSN 1742-6588. doi: 10.1088/1742-6596/659/1/012006

First, I review the problem setup in Section 5.1 and then I identify fundamental drawbacks of the full GP model in Section 5.2. The Section 5.3 presents contributions mainly from [Prüher and Šimandl, 2014], where the recursive GP regression model is proposed and its overall performance compared with the standard full GP model. Finally, the Section 5.6 collects results from [Král et al., 2014; Prüher and Král, 2015] and presents performance comparison of the bicriterial dual controllers with the full GP and the recursive GP models.

5.1 Problem Setup

All contributions in this chapter are formulated for a non-linear stochastic discrete time-invariant dynamical system given by

$$y_{k+1} = f(\mathbf{x}_k) + g(\mathbf{x}_k)u_k + e_{k+1}, \quad e_{k+1} \sim \mathrm{N}\big(0,\, \sigma_e^2\big), \tag{5.1.1}$$

where $f,\ g : \mathbb{R}^{n_y+n_u+1} \to \mathbb{R}$ are unknown non-linear functions, $u_k,\ y_k \in \mathbb{R}$ are control input and system output respectively, e_k is the noise term and

$$\mathbf{x}_k = \begin{bmatrix} y_k & \dots & y_{k-n_y} & u_{k-1} & \dots & u_{k-n_u} \end{bmatrix}^\top \in \mathbb{R}^{n_y+n_u+1} \tag{5.1.2}$$

is the state vector. Additionally, the following assumptions are considered:

A1: The non-linear functions are infinitely differentiable, i.e. $f(\mathbf{x}_k),\ g(\mathbf{x}_k) \in \mathcal{C}^\infty$.

A2: The structural parameters n_y and n_u of the system are known.

A3: The system has a globally uniformly asymptotically stable zero dynamics and $\forall \mathbf{x}_k :\ g(\mathbf{x}_k) > 0$ [Chen and Khalil, 1995].

A4: e_k is a white Gaussian noise sequence with known variance σ_e^2.

This system can be seen as a special case of the NARMAX structure and was chosen, because it is useful for functional dual adaptive control (FDAC) design [Fabri and Kadirkamanathan, 2001].

5.2 Full GP Model

This section covers identification of a non-linear system in eq. (5.1.1) by means of the GP regression algorithm in it's original formulation. I call this model *full Gaussian process* (FGP), because it works with the entire dataset at any given time. It also serves as a baseline for experiments and as a motivation for the recursive GP regression algorithm proposed in the subsequent section.

Let $\boldsymbol{\phi}_k^\top = \begin{bmatrix} \mathbf{x}_k^\top & u_k \end{bmatrix}$ be a vector of regressors of dimension $D = n_y + n_u + 2$ and let $\boldsymbol{\Phi}_k = \begin{bmatrix} \boldsymbol{\phi}_0 & \dots & \boldsymbol{\phi}_{k-1} \end{bmatrix}$ be a $D \times k$ design matrix. Denoting $h(\boldsymbol{\phi}_k) \triangleq f(\mathbf{x}_k) + g(\mathbf{x}_k)u_k$ we can write the system as

$$y_{k+1} = h(\boldsymbol{\phi}_k) + e_k, \quad e_k \sim \mathrm{N}\big(0,\, \sigma_e^2\big). \tag{5.2.1}$$

The whole idea of GP modeling is to put a Gaussian process prior on the function h, resulting in the following model

$$h \sim \mathrm{GP}(m_h(\boldsymbol{\phi}_k),\, k_h(\boldsymbol{\phi}_k, \boldsymbol{\phi}_l)), \tag{5.2.2a}$$

$$\hat{y}_{k+1} = h(\boldsymbol{\phi}_k) + e_k, \quad e_k \sim \mathrm{N}\big(0,\, \sigma_e^2\big), \tag{5.2.2b}$$

where the mean function m_h and the covariance kernel k_h need to be chosen by the designer. For our purposes, the mean function is chosen as $m_h(\boldsymbol{\phi}_k) = 0$, because it is analytically beneficial and does not take away from the generality of the model. Kernel is determined by applying the covariance operator to the model output, which yields

$$\mathbb{C}[\hat{y}_{k+1}, \hat{y}_{l+1}] = \mathbb{C}_h[h(\boldsymbol{\phi}_k), h(\boldsymbol{\phi}_l)] + \mathbb{C}[e_k, e_l], \tag{5.2.3}$$

$$= \mathbb{C}[f(\mathbf{x}_k) + g(\mathbf{x}_k)u_k, f(\mathbf{x}_l) + g(\mathbf{x}_l)u_l] + \sigma_e^2\delta_{kl}, \tag{5.2.4}$$

$$= \mathbb{C}_f[f(\mathbf{x}_k), f(\mathbf{x}_l)] + \mathbb{C}_g[g(\mathbf{x}_k), g(\mathbf{x}_l)]u_k u_l + \sigma_e^2\delta_{kl}, \tag{5.2.5}$$

$$= k_f(\mathbf{x}_k,\, \mathbf{x}_l) + k_g(\mathbf{x}_k,\, \mathbf{x}_l)u_k u_l + \sigma_e^2\delta_{kl}, \tag{5.2.6}$$

where we used the fact that the functions f and g are modeled as independent GPs and the noise e_k is white and independent of h. From the above, we can deduce that the GP covariance is given by

$$k_h(\boldsymbol{\phi}_k,\, \boldsymbol{\phi}_l) = k_f(\mathbf{x}_k,\, \mathbf{x}_l) + k_g(\mathbf{x}_k,\, \mathbf{x}_l)u_k u_l. \tag{5.2.7}$$

The same form of the covariance function was used by Sbarbaro and Murray-Smith [2003]. Since, per assumption A1, the functions f and g are smooth, we are justified in using the radial basis function (RBF) form for kernels k_f and k_g.

To summarize in more detail, the first two moments of the GP prior are given by

$$m_h(\boldsymbol{\phi}_k) = 0, \tag{5.2.8a}$$

$$k_h(\boldsymbol{\phi}_k,\, \boldsymbol{\phi}_l) = k_f(\mathbf{x}_k,\, \mathbf{x}_l) + k_g(\mathbf{x}_k,\, \mathbf{x}_l)u_k u_l, \tag{5.2.8b}$$

where the RBF kernels are given by

$$k_f(\mathbf{x}_k,\, \mathbf{x}_l) = \alpha_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{x}_l)^\top \boldsymbol{\Lambda}_f^{-1}(\mathbf{x}_k - \mathbf{x}_l)\right), \tag{5.2.9a}$$

$$k_g(\mathbf{x}_k,\, \mathbf{x}_l) = \alpha_g^2 \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{x}_l)^\top \boldsymbol{\Lambda}_g^{-1}(\mathbf{x}_k - \mathbf{x}_l)\right). \tag{5.2.9b}$$

The parameters α_f, α_g are output lengthscales. The input lengthscale matrices have the

form

$$\boldsymbol{\Lambda}_f = \operatorname{diag}\Big(\begin{bmatrix} \lambda_{f1}^2 & \dots & \lambda_{fD}^2 \end{bmatrix} \Big), \tag{5.2.10a}$$

$$\boldsymbol{\Lambda}_g = \operatorname{diag}\Big(\begin{bmatrix} \lambda_{g1}^2 & \dots & \lambda_{gD}^2 \end{bmatrix} \Big). \tag{5.2.10b}$$

For notational convenience, all kernel parameters are collated into one vector

$$\boldsymbol{\theta} = \begin{bmatrix} \alpha_f & \lambda_{f1} & \dots & \lambda_{fD} & \alpha_g & \lambda_{g1} & \dots & \lambda_{gD} \end{bmatrix}^\top. \tag{5.2.11}$$

Compared to the parametric regression models, where the assumptions on the modeled function are typically expressed in a fixed model structure, the assumptions of non-parametric GP models, expressed by the choice of covariance function, are much weaker. For instance, it is more restrictive to search for the unknown function in the space of fixed-order polynomials, rather than searching in the space of smooth functions. The RBF kernels k_f and k_f are often called *squared exponential*[1] and they are a popular choice in a many applications [Deisenroth et al., 2012; Kocijan et al., 2005]. Because the lengthscale matrices have independent lengthscale parameters for each input dimension, the kernels are said to exhibit the *automatic relevance determination* property. This is to say, that the relevance of individual input dimensions can be revealed after iterative optimization of lengthscales has taken place. Evidently, the RBF kernel is not the only choice. Since, roughly speaking, any symmetric semi-definite function of two arguments is a valid kernel [Janson, 1997], there is a plethora of covariance functions to choose from; each expressing different assumptions about the true underlying functional relationship. For comprehensive account of covariance functions and their relation to established parametric regression models see [Duvenaud, 2014; Rasmussen and Williams, 2006; Wilson, 2014].

With the help of the eqs. (2.2.11a) to (2.2.11b), which define the GP posterior predictive mean and variance, we can now determine what will the GP model output look like. Since we are dealing with a GP, the model output (at a particular time step) is a Gaussian random variable, so that $\hat{y}_{k+1} \sim \mathrm{N}\big(\hat{\mu}_{k+1},\, \hat{\sigma}_{k+1}^2\big)$, where the mean and variance are given by

$$\hat{\mu}_{k+1} \triangleq \mathbb{E}[\hat{y}_{k+1}] = \mathbf{k}_s^\top \big(\mathbf{K} + \sigma_e^2 \mathbf{I}\big)^{-1} \mathbf{y}_k, \tag{5.2.12a}$$

$$\hat{\sigma}_{k+1}^2 \triangleq \mathbb{V}[\hat{y}_{k+1}] = k_{ss} - \mathbf{k}_s^\top \big(\mathbf{K} + \sigma_e^2 \mathbf{I}\big)^{-1} \mathbf{k}_s, \tag{5.2.12b}$$

[1] As pointed out by Neil Lawrence, this terminology is incorrect, in that the kernel is not a squared exponential function but rather an *exponentiated quadratic* (i.e. $\exp\big(x^2\big) \neq \exp(x)^2$). Since this name does not quite roll off the tongue as easy, we call it the *RBF kernel*.

where

$$\mathbf{K} = k_h(\mathbf{\Phi}_k,\, \mathbf{\Phi}_k), \tag{5.2.13}$$

$$\mathbf{k}_s = \mathbf{k}_{sf} + \mathbf{k}_{sg} u_k, \tag{5.2.14}$$

$$k_{ss} = k_{ssf} + k_{ssg} u_k^2 + \sigma_e^2, \tag{5.2.15}$$

with

$$\mathbf{k}_{sf} = k_f(\mathbf{X}_k,\, \mathbf{x}_k), \qquad k_{ssf} = k_f(\mathbf{x}_k,\, \mathbf{x}_k), \tag{5.2.16}$$

$$\mathbf{k}_{sg} = k_g(\mathbf{X}_k,\, \mathbf{x}_k) \circ \mathbf{u}_k, \qquad k_{ssg} = k_g(\mathbf{x}_k,\, \mathbf{x}_k), \tag{5.2.17}$$

where $\mathbf{X}_k^\top = \begin{bmatrix}\mathbf{x}_0 & \dots & \mathbf{x}_{k-1}\end{bmatrix}^\top$ is a part of the design matrix, so that $\mathbf{\Phi}_k = \begin{bmatrix}\mathbf{X}_k^\top & \mathbf{u}_k^\top\end{bmatrix}^\top$. The variables $\mathbf{y}_k^\top = \begin{bmatrix}y_0 & \dots & y_k\end{bmatrix}$ and $\mathbf{u}_k^\top = \begin{bmatrix}u_0 & \dots & u_{k-1}\end{bmatrix}$ denote vectors containing system outputs and inputs respectively, up to time k. The symbol $\circ$ denotes element-wise product and the following shorthand notation is used

$$k(\mathbf{\Phi}_k,\, \mathbf{x}_k) \triangleq \begin{bmatrix} k(\boldsymbol{\phi}_0,\, \mathbf{x}_k) & \dots & k\big(\boldsymbol{\phi}_{k-1},\, \mathbf{x}_k\big)\end{bmatrix}^\top, \tag{5.2.18}$$

$$k(\mathbf{\Phi}_k,\, \mathbf{\Phi}_k) \triangleq \begin{bmatrix} k(\boldsymbol{\phi}_0,\, \boldsymbol{\phi}_0) & \dots & k\big(\boldsymbol{\phi}_0,\, \boldsymbol{\phi}_{k-1}\big) \\ \vdots & \ddots & \vdots \\ k\big(\boldsymbol{\phi}_{k-1},\, \boldsymbol{\phi}_0\big) & \dots & k\big(\boldsymbol{\phi}_{k-1},\, \boldsymbol{\phi}_{k-1}\big)\end{bmatrix}. \tag{5.2.19}$$

The mean of the GP model output $\hat{\mu}_{k+1}$ is taken to be the one-step prediction of the system output.

What remains at this point is to find optimal values of the kernel parameters, so that the GP model predictions can be calculated. To do that, I maximize the logarithm of marginal likelihood, given by

$$\log p(\,\mathbf{y}_k \,|\, \mathbf{\Phi}_k, \boldsymbol{\theta}\,) = -\frac{1}{2}\Big[\mathbf{y}_k^\top \big(\mathbf{K} + \sigma_e^2 \mathbf{I}\big)^{-1} \mathbf{y}_k + \log|\mathbf{K} + \sigma_e^2 \mathbf{I}| + k\log(2\pi)\Big]. \tag{5.2.20}$$

The advantage of this optimization objective is that it automatically trades-off data fit and model complexity, as described in Section 2.2.2, and therefore the resulting model is "just right" (provided enough data is available [Wilson, 2014]). The marginal likelihood objective is nonlinear in $\boldsymbol{\theta}$, hence finding the global solution may be practically unattainable (even though with increasing data set size the number of local solutions should decrease [Rasmussen and Williams, 2006]).

The FGP system identification method, that I propose in this section, works in batch

mode; which is to say, whenever new data point arrives, it is added to the dataset and the parameter optimization is started anew (from the same initial guess) at every time step.

Algorithm 7: Full Gaussian process system identification.

Input: Measurements $\mathcal{D} = \{(\boldsymbol{\phi}_k,\, y_{k+1})\}_{k=1}^{K}$, initial kernel parameters $\boldsymbol{\theta}_0$, probing gain parameter η.
Output: FGP model summarized by $(\mathcal{D},\, \boldsymbol{\theta}_K)$.

for $k \leftarrow 1$ **to** K **do**

`// Measure the new data point` $\boldsymbol{\phi}_k = \begin{bmatrix}\mathbf{x}_k^\top & u_k\end{bmatrix}$ `and update datasets.`

$\mathbf{y}_k \leftarrow \begin{bmatrix} y_0 & \dots & y_k \end{bmatrix}$

$\mathbf{u}_k \leftarrow \begin{bmatrix} u_0 & \dots & u_{k-1} \end{bmatrix}$

$\mathbf{X}_k \leftarrow \begin{bmatrix} \mathbf{x}_0 & \dots & \mathbf{x}_{k-1} \end{bmatrix}$

`// Kernel parameter optimization (starts from` $\boldsymbol{\theta}_0$`).`

$\boldsymbol{\theta}_k \leftarrow \arg\min_{\boldsymbol{\theta}} \log p(\mathbf{y}_k \,|\, \mathbf{X}_k, \boldsymbol{\theta})$

end

Algorithm 8: Function for calculating predictions of the FGP model.

Function `FGPPrediction(`$\mathcal{D}$, $\boldsymbol{\theta}^*$`)`

`// Unpack the data.`

$\begin{bmatrix}\mathbf{X}_k & \mathbf{u}_k & \mathbf{y}_k\end{bmatrix} \leftarrow \mathcal{D}$

`// Kernel evaluation.`

$k_{ssf} \leftarrow k_f\big(\mathbf{X}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$k_{ssg} \leftarrow k_g\big(\mathbf{X}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$k_{ss} \leftarrow k_{ssf} + k_{ssg} u_k^2 + \sigma_e^2$

$\mathbf{k}_{sf} \leftarrow k_f\big(\mathbf{X}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$\mathbf{k}_{sg} \leftarrow k_g\big(\mathbf{X}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big) \circ \mathbf{u}_k$

$\mathbf{k}_s \leftarrow \mathbf{k}_{sf} + \mathbf{k}_{sg} u_k$

$\mathbf{K} \leftarrow k_h\big(\mathbf{X}_k,\, \mathbf{X}_k;\, \boldsymbol{\theta}_k\big)$

`// System output prediction.`

$\hat{\mu}_{k+1} \leftarrow \mathbf{k}_s^\top \big(\mathbf{K} + \sigma_e^2 \mathbf{I}\big)^{-1} \mathbf{y}_k$

$\hat{\sigma}_{k+1}^2 \leftarrow k_{ss} - \mathbf{k}_s^\top \big(\mathbf{K} + \sigma_e^2 \mathbf{I}\big)^{-1} \mathbf{k}_s$

return $\hat{\mu}_{k+1},\, \hat{\sigma}_{k+1}^2$

end

Because of the sequential nature of the data, the full GP faces two main challenges.

1. First, consider an offline setting, where all the data are readily available. The eqs. (5.2.12a) to (5.2.12b) show that the prediction is calculated using *all* data points, which is the key distinguishing feature of non-parametric models in general. In such case, calculating model prediction calls for one-time inversion of a large covariance matrix $\mathbf{K}$, which scales unfavorably with data size (complexity of order $\mathcal{O}(k^3)$). The situation is even worse with sequential data, because the size of $\mathbf{K}$ grows with every new measurement, leading to ever increasing computational demands for matrix inversion.

2. The kernel matrix inverse is also present in the marginal likelihood objective, as is evident from eq. (5.2.20). Note that in this case, the matrix inversion needs to take place in each iteration of the optimization solver (and the parameters need to be optimized at every time step).

Naturally, these issues could be dealt with if we could devise a recursive solution, which would update kernel parameters $\boldsymbol{\theta}_k$ using the current data point $(y_k,\, \boldsymbol{\phi}_k)$ and the previous parameter estimate $\boldsymbol{\theta}_{k-1}$. Recursive parameter optimization is hinted at in [Hartikainen and Särkkä, 2010], however it assumes the index set of the GP is one-dimensional, which is too restrictive for our application. The two points, mentioned above, are the main motivation for using the recursive Gaussian process algorithm [Huber, 2013].

5.3 Recursive GP Model

The great advantage of parametric models is that they lend themselves nicely to the possibility of devising a recursive estimation algorithms, which are typically realized by some kind of nonlinear filter [Haykin, 2001]. As GPs are non-parametric models, where number of parameters increases with the dataset, developing a recursive algorithm is becoming much more complicated. Typically, it is necessary to consider some kind of GP approximation to counteract the problem of increasing data size.

Recently increased effort has been given to the design of approximate GP regression algorithms, which eliminate such drawbacks [Chowdhary et al., 2014; Csató and Opper, 2002; Huber, 2013; Ranganathan et al., 2011; Särkkä, 2013]. Särkkä [2013] proposed a solution to this problem based on Kalman filtering and smoothing applied to the state-space representation of the GP with specific covariance functions. This solution, however, works for one-dimensional inputs only and is thus inappropriate for our purposes. A very frequently used alternative is the Sparse Online Gaussian Process (SOGP)

[Chowdhary et al., 2014; Csató and Opper, 2002; Ranganathan et al., 2011], which, however, does not solve the problem of online kernel parameter learning. This section investigates usefulness of the *recursive Gaussian process* (RGP) algorithm, proposed by Huber [2013], for non-linear system identification. I rely on the brief outline of the RGP from Section 2.2.3 and present an ad-hoc procedure for kernel parameter estimation.

Assume, that we have chosen a set of S basis vectors (regressors) arranged into a design matrix structured as $\tilde{\mathbf{\Phi}} = \begin{bmatrix}\tilde{\mathbf{X}}^\top & \tilde{\mathbf{u}}^\top\end{bmatrix}^\top$. With GP prior specified by eqs. (5.2.8a) to (5.2.11), we can now derive expressions for prediction of the system output and its variance. Using the eqs. (2.2.24a) to (2.2.25c), we obtain

$$\hat{\mu}_{k+1} \triangleq \mathbb{E}[\hat{y}_{k+1}] = \mathbf{J}_k \tilde{\boldsymbol{\mu}}_{k-1}, \tag{5.3.1a}$$

$$\hat{\sigma}^2_{k+1} \triangleq \mathbb{V}[\hat{y}_{k+1}] = k_{ss} + \mathbf{J}_k\big(\tilde{\mathbf{\Sigma}}_{k-1} - \mathbf{K}\big)\mathbf{J}_k, \tag{5.3.1b}$$

where $\mathbf{J}_k = \mathbf{k}_s^\top \mathbf{K}^{-1}$ and

$$\mathbf{K} = k_h\big(\tilde{\mathbf{\Phi}},\, \tilde{\mathbf{\Phi}}\big), \tag{5.3.2}$$

$$\mathbf{k}_s = \mathbf{k}_{sf} + \mathbf{k}_{sg} u_k, \tag{5.3.3}$$

$$k_{ss} = k_{ssf} + k_{ssg} u_k^2 + \sigma_e^2, \tag{5.3.4}$$

with

$$\mathbf{k}_{sf} = k_f\big(\tilde{\mathbf{X}},\, \mathbf{x}_k\big), \qquad k_{ssf} = k_f\big(\mathbf{x}_k,\, \mathbf{x}_k\big), \tag{5.3.5}$$

$$\mathbf{k}_{sg} = k_g\big(\tilde{\mathbf{X}},\, \mathbf{x}_k\big) \circ \tilde{\mathbf{u}}, \qquad k_{ssg} = k_g\big(\mathbf{x}_k,\, \mathbf{x}_k\big), \tag{5.3.6}$$

where the symbol $\circ$ denotes the element-wise product of two vectors. The quantities $\tilde{\boldsymbol{\mu}}_k$ and $\tilde{\mathbf{\Sigma}}_k$, which are to be used to generate predictions in the next time step, are calculated by the familiar equations

$$\mathbf{G}_k = \tilde{\mathbf{\Sigma}}_{k-1}\mathbf{J}_k^\top \big(\hat{\sigma}_k^2 + \sigma^2\big)^{-1}, \tag{5.3.7a}$$

$$\tilde{\boldsymbol{\mu}}_k = \tilde{\boldsymbol{\mu}}_{k-1} + \mathbf{G}_k(y_k - \hat{\mu}_k), \tag{5.3.7b}$$

$$\tilde{\mathbf{\Sigma}}_k = \tilde{\mathbf{\Sigma}}_{k-1} - \mathbf{G}_k\mathbf{J}_k\tilde{\mathbf{\Sigma}}_{k-1}. \tag{5.3.7c}$$

5.3.1 Ad-hoc Kernel Parameter Learning Procedure

It is evident by now, that the RGP algorithm is able to reduce the GP model prediction complexity in sequential data processing. Note however, that the kernel parameters $\boldsymbol{\theta}$ still have to be determined, in order to compute model predictions (given by eqs. (5.3.1a)

to (5.3.1b)). One possibility would be to use the RGP* algorithm [Huber, 2014], which is able to learn the parameters in recursive manner in addition to recursively performing the GP regression. Nonetheless, the implementation of RGP* proved to be problematic, which is why I sought alternatives.

The proposed ad-hoc approach treats the current estimate of the latent function values $\tilde{\boldsymbol{\mu}}_k$ as perfect observations. In practical terms this entails modification of the marginal likelihood objective function eq. (5.2.20), whereby the noisy observations $\mathbf{y}_k$ are replaced with the observations $\tilde{\boldsymbol{\mu}}_k$ at basis vector locations $\tilde{\boldsymbol{\Phi}}$. The marginal likelihood is effectively approximated by

$$\log p(\mathbf{y}_k \,|\, \boldsymbol{\Phi}_k,\, \boldsymbol{\theta}\,) \approx \log p\Big(\,\tilde{\boldsymbol{\mu}}_k \,\Big|\, \tilde{\boldsymbol{\Phi}},\, \boldsymbol{\theta}\,\Big), \tag{5.3.8}$$

where the approximate marginal likelihood is given as

$$\log p\Big(\,\tilde{\boldsymbol{\mu}}_k \,\Big|\, \tilde{\boldsymbol{\Phi}},\, \boldsymbol{\theta}\,\Big) = -\frac{1}{2}\Big[\tilde{\boldsymbol{\mu}}_k^\top \mathbf{K}(\boldsymbol{\theta})^{-1}\tilde{\boldsymbol{\mu}}_k + \log\big|\mathbf{K}(\boldsymbol{\theta})\big| + S\log(2\pi)\Big], \tag{5.3.9}$$

where $\mathbf{K}(\boldsymbol{\theta}) = k(\tilde{\boldsymbol{\Phi}},\, \tilde{\boldsymbol{\Phi}};\, \boldsymbol{\theta})$. Since the number of basis vectors S is pre-defined by the user and fixed throughout the operation of the algorithm, the computational demands for evaluation of the likelihood do not grow with time. In practical implementation, the objective eq. (5.3.9) is maximized in every time step using an iterative optimization solver, where the initial guess are the kernel parameters from the previous time step. I limit the maximum number of solver iterations to keep the computational requirements in check. The notation $\mathbf{K}(\boldsymbol{\theta})$ emphasizes the fact that the kernel matrix is a function of parameters $\boldsymbol{\theta}$. The Algorithm 9 summarizes the RGP regression with an ad-hoc parameter learning for system identification.

5.4 Numerical Experiments and Evaluation

The RGP was compared with the performance of the FGP in numerical simulations on the following benchmarking example [Fabri and Kadirkamanathan, 2001] of non-linear stochastic discrete-time system

$$y_{k+1} = \frac{1.5y_k}{1+y_k^2} + (2+\cos(y_k))u_k + e_{k+1}, \tag{5.4.1}$$

Algorithm 9: Recursive Gaussian process system identification. The subroutine `RGPPrediction` is defined in the Algorithm 10.

Input: Measurements $\left\{(\boldsymbol{\phi}_k,\, y_{k+1})\right\}_{k=1}^{K}$, initial kernel parameters $\boldsymbol{\theta}_1$.
Output: RGP model summarized by $\left(\tilde{\boldsymbol{\mu}}_K,\, \tilde{\boldsymbol{\Sigma}}_K,\, \boldsymbol{\theta}_K\right)$.

`// initialization`
$\tilde{\boldsymbol{\mu}}_1 \leftarrow \mathbf{0}$
$\tilde{\boldsymbol{\Sigma}}_1 \leftarrow \mathbf{I}$
for $k \leftarrow 1$ **to** K **do**
  `// System output prediction.`
  $\hat{\mu}_{k+1}, \hat{\sigma}^2_{k+1}, \mathbf{J}_k \leftarrow$ `RGPPrediction(`$\boldsymbol{\phi}_k, \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k, \tilde{\boldsymbol{\Phi}}, \boldsymbol{\theta}_k$`)`
  `// Update RGP model.`
  $\mathbf{G}_k \leftarrow \tilde{\boldsymbol{\Sigma}}_k \mathbf{J}_k^\top \left(\hat{\sigma}^2_{k+1} + \sigma_e^2\right)^{-1}$
  $\tilde{\boldsymbol{\mu}}_{k+1} \leftarrow \tilde{\boldsymbol{\mu}}_k + \mathbf{G}_k(y_{k+1} - \hat{\mu}_{k+1})$
  $\tilde{\boldsymbol{\Sigma}}_{k+1} \leftarrow \tilde{\boldsymbol{\Sigma}}_k - \mathbf{G}_k \mathbf{J}_k \tilde{\boldsymbol{\Sigma}}_k$
  `// Kernel parameter optimization (starts from` $\boldsymbol{\theta}_k$`).`
  $\boldsymbol{\theta}_{k+1} \leftarrow \arg\min_{\boldsymbol{\theta}} \log p\Big(\, \tilde{\boldsymbol{\mu}}_{k+1} \,\Big|\, \tilde{\boldsymbol{\Phi}},\, \boldsymbol{\theta} \,\Big)$
end

Algorithm 10: Function for calculating predictions of the RGP model.

Function `RGPPrediction(`$\boldsymbol{\phi}_k, \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k, \tilde{\boldsymbol{\Phi}}, \boldsymbol{\theta}^*$`)`
  `// Unpack the data.`
  $\begin{bmatrix}\mathbf{x}_k & u_k\end{bmatrix} \leftarrow \boldsymbol{\phi}_k$
  `// Evaluate the kernel.`
  $k_{ssf} \leftarrow k_f\left(\tilde{\mathbf{X}},\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\right)$
  $k_{ssg} \leftarrow k_g\left(\tilde{\mathbf{X}},\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\right)$
  $k_{ss} \leftarrow k_{ssf} + k_{ssg} u_k^2 + \sigma_e^2$
  $\mathbf{k}_{sf} \leftarrow k_f\left(\tilde{\mathbf{X}},\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\right)$
  $\mathbf{k}_{sg} \leftarrow k_g\left(\tilde{\mathbf{X}},\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\right) \circ \tilde{\mathbf{u}}$
  $\mathbf{k}_s \leftarrow \mathbf{k}_{sf} + \mathbf{k}_{sg} u_k$
  $\mathbf{K} \leftarrow k_h\left(\tilde{\boldsymbol{\Phi}},\, \tilde{\boldsymbol{\Phi}};\, \boldsymbol{\theta}_k\right)$
  `// System output prediction.`
  $\mathbf{J}_k \leftarrow \mathbf{k}_s^\top \mathbf{K}^{-1}$
  $\hat{\mu}_{k+1} \leftarrow \mathbf{J}_k \tilde{\boldsymbol{\mu}}_k$
  $\hat{\sigma}^2_{k+1} \leftarrow k_{ss} + \mathbf{J}_k\left(\tilde{\boldsymbol{\Sigma}}_k - \mathbf{K}\right)\mathbf{J}_k^\top$
  return $\hat{\mu}_{k+1}, \hat{\sigma}^2_{k+1}, \mathbf{J}_k$
end

where the sequence e_k is a zero-mean white Gaussian noise with variance $\sigma_e^2 = 0.025$. In this case the vector of regressors $\boldsymbol{\phi}_k = \begin{bmatrix} y_{k-1} & u_{k-1} \end{bmatrix}$ is two dimensional. Kernel parameters were initialized with values $\boldsymbol{\theta}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\top$ for both algorithms. As the excitation signal u_k, I considered uniformly random noise within the interval $[-3,\, 3]$.

In case of RGP algorithm, 75 basis vectors were used to create a 15×5 grid, that covered the area of state space, where the system operates. Ad-hoc kernel parameter optimization routine was started with the estimate obtained in previous iteration. The FGP was trained using all available data up to the current time step, where the parameter optimization routine was started with initial values $\boldsymbol{\theta}_0$. We used MATLAB's `fminunc()` solver with the maximum number of iterations set to 100.

The experiments focused on the comparison of the two approaches in terms of obtained GP model error and computational demands for prediction in every time step. Root mean square error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{K_{ts}} \sum_{k=1}^{K_{ts}} (y_k - \hat{y}_k)^2} \tag{5.4.2}$$

was used to measure the error in model validation process, where y_k is the measurement of system output and $\hat{y}_k$ is the GP model prediction. Furthermore, the logarithm of average predictive variance

$$\mathrm{LOGVAR} = \log\left(\frac{1}{K_{ts}} \sum_{k=1}^{K_{ts}} \mathbb{V}[\hat{y}_k]\right) \tag{5.4.3}$$

was used to quantify the overall model prediction uncertainty. Both metrics were calculated at each of the $K = 300$ time steps using $K_{ts} = 1600$ test points, which were obtained from the system operation. The computational demand is assumed to be the sum of the time required for obtaining the current parameter estimate and the time required for computing the one-step prediction (model response). This was implemented by employing MATLAB's `tic/toc` command functionality and results recorded for every time step. 100 MC simulations were run with the results shown in figures.

As seen in Figure 5.4.1a, RMSE of the FGP is slightly lower than that of the RGP algorithm. This can be caused by the fact, that the RGP uses a *fixed amount* of predefined basis vectors for prediction. The FGP, however, makes predictions based on *all* available past data points. The main drawback of the RGP is that, when prediction is to be made further away from the area covered by the basis vectors, it's quality will be significantly

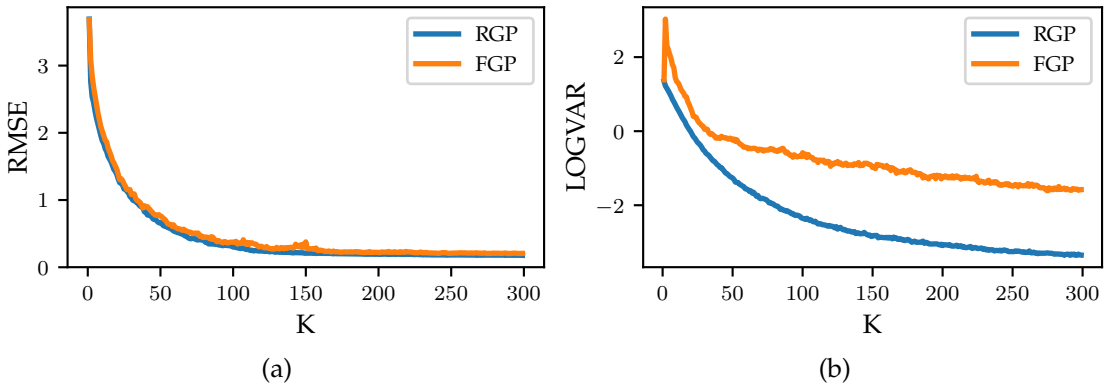

Figure 5.4.1: (a) Root mean square error. (b) Logarithm of average predictive variance.

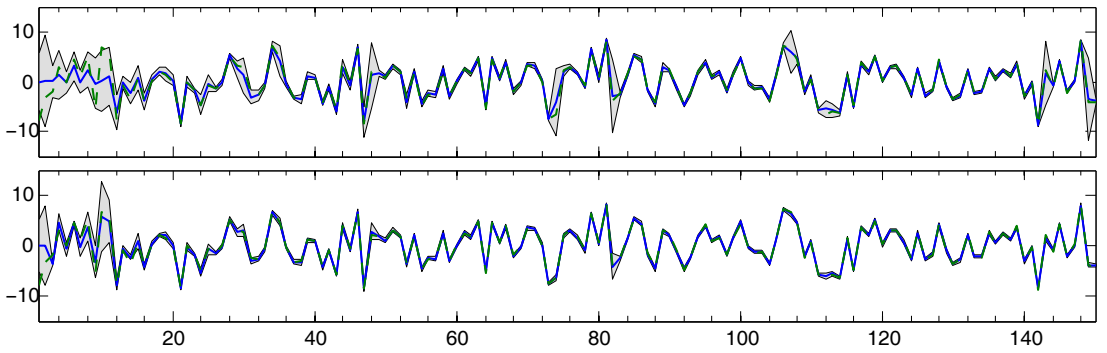

Figure 5.4.2: System response (dashed) compared to the RGP (top) and FGP (bottom) model response with the increasing amount of data used for system identification (GP model training). The grey band represents the model uncertainty.

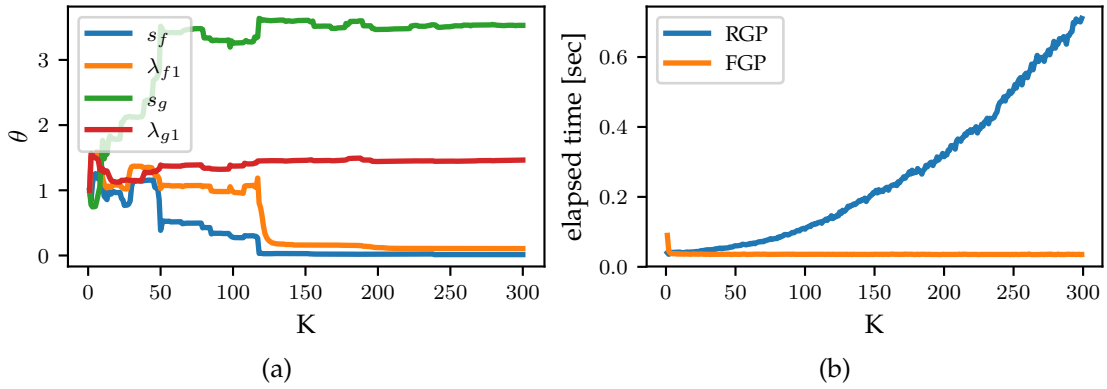

Figure 5.4.3: (a) Hyper-parameter development in time. (b) Computation time per iteration.

worse than that of the FGP.

Development of the LOGVAR in Figure 5.4.1b suggests, that predictions of the RGP algorithm are less confident than those of the FGP. Figure 5.4.2 compares the system and model response during training. Note how the model response approaches the true system response and the model uncertainty is reduced with the increasing amount of training data. Figure 5.4.3a shows converging parameter estimates. These may not always converge to the same values since the marginal likelihood has multiple local extrema. The Figure 5.4.3b demonstrates the unsurprising fact, that the RGP algorithm has constant computational demands per time step. On the contrary, time requirements of the FGP algorithm for prediction increase with each time step.

5.5 Conclusion

In this section, I proposed use of the RGP algorithm with an ad-hoc kernel parameter learning procedure for the non-linear system identification. The RGP algorithm was compared with the FGP in terms of computational demands per iteration as well as the RMSE and the logarithm of average predictive variance. In simulations I demonstrated that the RGP algorithm is clearly superior to the FGP approach in terms computational demands. Compared to the FGP, the RGP algorithm provides more conservative predictions and achieves comparable RMSE.

Main disadvantage of the RGP algorithm is that the number of basis vectors increases exponentially with the dimension of the state space (vector of regressors). Also, quality of predictions at the locations further away from the area of state space covered by the basis vectors is worse than those of the FGP, however this effect could be mitigated by suitable specification of the prior mean function.

5.6 Application: Functional Dual Adaptive Control

Cybernetics can be characterized as an inter-disciplinary scientific endeavor that seeks to design optimal control systems. Need for such controllers arises in wide array of engineering disciplines such as design of aircraft autopilots, industrial robotics and control of chemical reactors and other industrial processes.

In the real-world, dynamical properties of the controlled process are changing with time, such as, aircraft loosing its mass due to fuel consumption or mechanical properties of parts changing due to wear. An ideal control law should take these considerations into account. We can observe variety of biological systems in nature which are able to thrive under changing environmental conditions. This is in no small part due to the presence of adaptive control systems on a microscopic level, which inspired researchers to engineer control systems emulating such behavior in hopes of improving control quality of processes under varying conditions. A controller containing a mechanism for changing its own parameters is termed *adaptive*[2]. Diagram of a model-based adaptive control is depicted in Figure 5.6.1. Realistically speaking, dynamical properties of all controlled

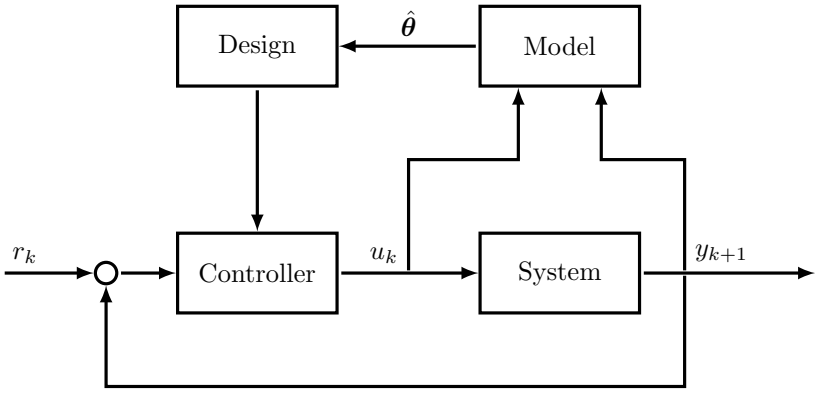

Figure 5.6.1: Adaptive control loop. The model serves as a surrogate for the unknown system, which is continuously being refined by the new incoming measurements of the input u_k and output y_{k+1}. The identification method produces the model parameter estimates, which are used for the adaptation of the controller.

processes are subject to certain degree of uncertainty. In order to deal with this problem, adaptive model-based control uses a surrogate system model with unknown parameters to summarize the state of knowledge about the controlled process up to the present time. The Model block contains an identification algorithm which produces parameter

[2] Åström and Wittenmark [1995] note that there is no generally accepted definition of adaptive system in the control community.

estimates based on observed past control inputs and noisy system response data - effectively performing system identification. Resulting parameter estimates are fed into the Design block, which decides how the controller parameters should be adapted in response to newly obtained knowledge about the controlled process.

Depending on what process information is used to produce the control action, several classes of controllers were identified. In the simplest case, the controller uses point estimates of all parameters as if they were the true values of the unknown quantities and disregards any uncertainties in the estimates. This is known as *certainty equivalence* principle, which holds in a few special cases, one of which is the well-studied linear quadratic Gaussian (LQG) control [Åström and Wittenmark, 1995, p. 360]. Due to their limiting theoretical assumptions, certainty equivalent controllers can often perform unsatisfactorily in real-world scenarios. The *cautious* controllers are more realistic, because they use additional information about the uncertainty of estimates. Typically, the magnitude of controller action is in this case inversely proportional to the model parameter uncertainty. Unfortunately, cautious controllers are susceptible to the *turn-off effect*. That is to say, whenever the model uncertainty is high, the control action will be small and consequently the amount of new information elicited from the system will decrease. Model uncertainty will therefore decrease minimally, leading again to ever so smaller control action in the subsequent time instances, which interferes with the control effort.

The *dual* controllers remove this problem from occurring by enriching the control action with a probing component. Fel'dbaum [1965] observed that in order to control an unknown process the controller design needs to find a delicate compromise between two mutually opposing requirements, which are

control (exploitation): using available process information to maximize the control quality (demands lower band-width) and

probing (exploration): perturbation of the process in order to gain more information about it (demands higher band-width).

To give an analogy, when a driver wants to improve his driving skills with a new car and, at the same time, get to his destination safely and cheaply, he needs to perform reasonably safe test maneuvers (probing) to get the feel for the car, so that he can drive better in the future, all the while keeping on the road (control). Fel'dbaum's proposed solution to the dual control problem is in the form of functional Bellman equation, which is analytically and even numerically intractable, except for a few simple cases. This is why a lot of research effort is focused on finding various approximate sub-optimal

solutions [Filatov and Unbehauen, 2004], which, nevertheless, still preserve the key aspects of the stochastic control principles (caution and probing).

Recent developments in adaptive dual control focused predominantly on the class of linear systems with unknown parameters. The *functional* dual adaptive control (FDAC) of nonlinear stochastic systems [Fabri and Kadirkamanathan, 2001; Herzallah and Lowe, 2002; Sarangapani, 2006; Sbarbaro et al., 2004] can be understood as a natural effort of extension of the adaptive control to the class of complex systems, where the *functional* relationships describing the nonlinear dynamics themselves are completely unknown. The FDAC problem is more difficult to solve, because no rigid parametric structure of the dynamics function is assumed. The original concept of the functional approach was formulated by Fabri and Kadirkamanathan [1998], where the innovation dual control (IDC) criterion, originally proposed by Milito et al. [1982], was used as a cost function together with two types of neural networks as a model for the unknown nonlinear functions. This was followed by Šimandl et al. [2005], who, instead of the IDC, utilized bicriterial cost function [Filatov and Unbehauen, 2004] and a Gaussian sum filter for parameter estimation. Further efforts tackle the individual problems, such as a practical demonstration in control of the nonholonomic wheeled mobile robot [Bugeja et al., 2009] or an extension to a more general MIMO class of nonlinear systems [Fabri and Bugeja, 2013; Král and Šimandl, 2011]. In recent years, several sub-optimal dual control methods have been applied successfully in the functional approach for their positive qualities (superior control quality and admissible computational demands) and subsequently extended in several directions [Bugeja et al., 2009; Fabri and Bugeja, 2013; Fabri and Kadirkamanathan, 1998; Král and Šimandl, 2011; Šimandl et al., 2005].

5.6.1 Bicriterial Control

We start with the formalization of the bicriterial approach to adaptive control design, first introduced by Filatov and Unbehauen [2004]. Later, we design dual adaptive controller based on the full and recursive GP models, discussed previously. Our contribution differs from [Šimandl et al., 2005] and [Král and Šimandl, 2011] in two main aspects: (a) we use a non-parametric GP model instead of the parametric neural network model, (b) the resulting dual control respects the GP model prediction uncertainties. From another point of view, this contribution is an extension of the GP-based cautious controller [Sbarbaro et al., 2004] by the duality property.

The key idea of the bicriterial approach is based on the cost function exploiting two separate criteria. Each criterion introduces one of the mutually opposing goals between estimation and control (*caution* and *probing*). The optimal control action is obtained by

first minimizing eq. (5.6.1) to solve for cautious control component, which is then later used to determine the probing component by subsequent minimization of the second criterion in eq. (5.6.3). The first of the criteria has the following form

$$J_k^c = \mathbb{E}\Big[(r_{k+1} - y_{k+1})^2 \;\Big|\; \mathbf{I}^k \Big], \tag{5.6.1}$$

where $\mathbf{I}^k$ is the information state containing all measurable system inputs and outputs available up to time instant k and r_{k+1} is a bounded reference signal. The *cautious control* action, resulting from the minimization

$$u_k^c = \arg\min_{u_k} J_k^c \tag{5.6.2}$$

respects uncertainties in the knowledge of the unknown functions in the system description (5.1.1). The second criterion is chosen as

$$J_k^a = -\mathbb{E}\Big[(y_{k+1} - \hat{y}_{k+1})^2 \;\Big|\; \mathbf{I}^k \Big], \tag{5.6.3}$$

where $\hat{y}_{k+1}$ is a one-step model prediction of the system output. The single term in the criterion penalizes deviation of the future system output as predicted by the model $\hat{y}_{k+1}$ and the actual system output y_{k+1}. It therefore forces reduction of uncertainty in the system model and thus effectively encourages exploration. Finally, the *dual control* is obtained by minimizing

$$u_k = \arg\min_{u_k \in \Omega_k} J_k^a, \tag{5.6.4}$$

where $\Omega_k = [u_k^c - \delta,\, u_k^c + \delta]$ defines a symmetrical Pareto set centered around the cautious control u_c as a region of permissible values of u_k, which represents an efficient trade-off between the criteria J_k^a and J_k^c. The presence of the design parameter δ_k, which determines the region Ω_k, stems from the reasoning that it is necessary to enrich the cautious control with a probing signal proportional to the uncertainty of the model. The situation is illustrated in Figure 5.6.2.

Looking at the criterion in eq. (5.6.3) it becomes evident that a vital part of the control design is a system model for generating predictions $\hat{y}_{k+1}$. Previously, in Sections 5.2 and 5.3, I described how the full GP and the recursive GP can be used as system models. In the following two sections, we utilize these models and identification algorithms for the design of a dual adaptive controller.

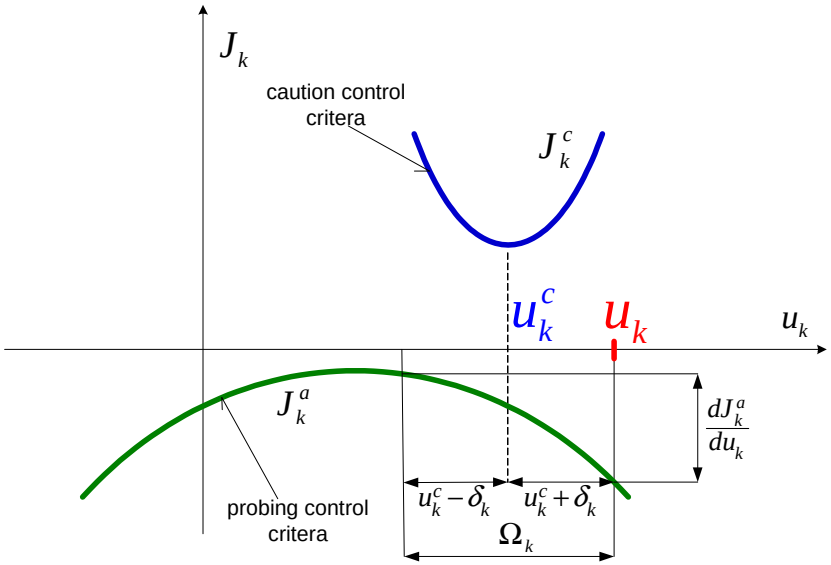

Figure 5.6.2: The core principle behind bicriterial dual control design. First, the cautious control component is determined as a minimizer of J_k^c. In the subsequent stage, the final dual control is determined by minimizing J_k^a over a set of permissible control actions, whose boundaries are parametrized by the u_k^c.

5.6.2 Control Design with Full GP

Using the kernel structure in eqs. (5.2.14) to (5.2.17), the GP model prediction and variance can be written as

$$\hat{\mu}_{k+1} = \hat{f}_{k+1} + \hat{g}_{k+1} u_k, \tag{5.6.5a}$$
$$\hat{\sigma}^2_{k+1} = \mu_k u_k^2 - \nu_k u_k + c_k + \sigma_e^2, \tag{5.6.5b}$$

where

$$\hat{f}_{k+1} \triangleq \mathbf{k}_{sf}^\top \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \mathbf{y}_k, \tag{5.6.6a}$$
$$\hat{g}_{k+1} \triangleq \mathbf{k}_{sg}^\top \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \mathbf{y}_k, \tag{5.6.6b}$$
$$\mu_k \triangleq k_{ssg} - \mathbf{k}_{sg} \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \mathbf{k}_{sg}^\top, \tag{5.6.6c}$$
$$c_k \triangleq k_{ssf} - \mathbf{k}_{sf} \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \mathbf{k}_{sf}^\top, \tag{5.6.6d}$$
$$\nu_k \triangleq 2\mathbf{k}_{sg} \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \mathbf{k}_{sf}^\top. \tag{5.6.6e}$$

The criterion in eq. (5.6.1) can be written as

$$J_k^c = \mathbb{E}\Big[(r_{k+1} - y_{k+1})^2 \;\Big|\; \mathbf{I}^k \Big] = \Big(r_{k+1} - \mathbb{E}\Big[y_{k+1} \;\Big|\; \mathbf{I}^k \Big] \Big)^2 \tag{5.6.7}$$

$$= \Big(r_{k+1} - (\hat{f}_{k+1} + \hat{g}_{k+1} u_k) \Big)^2 + \mu_k u_k^2 - \nu_k u_k + c_k, \tag{5.6.8}$$

where we used the approximation $\mathbb{E}\big[y_{k+1} \;\big|\; \mathbf{I}^k \big] \approx \mathbb{E}\big[\hat{y}_{k+1} \;\big|\; \mathbf{I}^k \big]$. Setting derivative w.r.t. the control action to zero

$$\frac{\mathrm{d}J_k^c}{\mathrm{d}u_k} = -2\Big(r_{k+1} - \hat{f}_{k+1} - \hat{g}_{k+1} u_k \Big) \hat{g}_{k+1} + 2\mu_k u_k - \nu_k = 0 \tag{5.6.9}$$

and solving for u_k, yields the cautious control component, given by

$$u_k^c = \frac{\big(r_{k+1} - \hat{f}_{k+1} \big) \hat{g}_{k+1} + \frac{1}{2}\nu_k}{\hat{g}_{k+1}^2 + \mu_k}. \tag{5.6.10}$$

The second criterion in eq. (5.6.3) can be expressed as

$$J_k^a = -\mu_k u_k^2 + \nu_k u_k - c_k. \tag{5.6.11}$$

For a non-parametric GP model it is suitable to modify the probing component design [Filatov and Unbehauen, 2004; Král and Šimandl, 2008] so that it respects the uncertainty in the model predictions. The main idea of the proposed modification is as follows. Efficiency of the system excitation is represented by the shape of the criteria J_k^a in the neighborhood of u_k^c as is illustrated by Figure 5.6.2. If J_k^a has a slope close to zero, the desired decrease of the criterion value could not be achieved even with large probing amplitude. In such cases, the criterion J_k^a changes negligibly with perturbations in u_k and thus it is not beneficial to enhance the control signal with probing. Otherwise, when the slope of J_k^a is significant, an appropriate excitation of the system could be achieved even with small perturbations in the control u_k. In this case, it makes sense to introduce the probing signal as it can significantly accelerate uncertainty reduction in the model.

With these considerations in mind, it is suitable to choose the domain Ω as symmetric and bounded with a constant, so that $\Omega = [u_k - \delta, u_k + \delta]$, where δ is a design parameter. Since eq. (5.6.4) defines a constrained optimization problem with a concave objective $J_k^a(u_k)$, the extreme is found on the boundary of the domain Ω. For dual control, we

can therefore write

$$u_k = u_k^c + \eta \operatorname{sign}\left(J_k^a(u_k^c + \delta_k) - J_k^a(u_k^c - \delta_k)\right), \tag{5.6.12}$$

where $\eta > 0$ is a design parameter, which represents gain of the probing. The expression in brackets evaluates to

$$\left.\frac{\mathrm{d}J_k^a}{\mathrm{d}u_k}\right|_{u_k=u_k^c} \propto J_k^a(u_k^c + \delta_k) - J_k^a(u_k^c - \delta_k) = 2\delta(-2\mu_k u_k^c + \nu_k) \tag{5.6.13}$$

where the multiplicative constant 2δ is subsumed into the design parameter η in eq. (5.6.12). Note that the derivative is a function of μ_k and ν_k that quantify uncertainty of the model functions $\hat{f}$ and $\hat{g}$. Consequently, a higher amplitude of the probing signal is generated in case of high model uncertainty and vice versa. Finally, the resulting bicriterial dual control action is given by

$$u_k = u_k^c + \eta(-2\mu_k u_k^c + \nu_k), \tag{5.6.14}$$

where the cautious component u_k^c is given by eq. (5.6.10). The proposed bicriterial dual full Gaussian process (BD-FGP) adaptive control algorithm is summarized in Algorithm 11. Note that the system output prediction on lines 14 to 17 is unnecessary to generate the control action.

5.6.3 Control Design with Recursive GP

The control design with the RGP model is very similar. We start with the same form for GP predictive mean and variance in eqs. (5.6.5a) and (5.6.5b), with the difference that now we define

$$\hat{f}_{k+1} \triangleq \mathbf{k}_{sf}^\top \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \tilde{\boldsymbol{\mu}}_k, \tag{5.6.15a}$$

$$\hat{g}_{k+1} \triangleq \mathbf{k}_{sg}^\top \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \tilde{\boldsymbol{\mu}}_k, \tag{5.6.15b}$$

$$\mu_k \triangleq k_{ssg} + \mathbf{k}_{sg}^\top \boldsymbol{\Gamma} \mathbf{k}_{sg}, \tag{5.6.15c}$$

$$c_k \triangleq k_{ssf} + \mathbf{k}_{sf}^\top \boldsymbol{\Gamma} \mathbf{k}_{sf}, \tag{5.6.15d}$$

$$\nu_k \triangleq 2\mathbf{k}_{sg}^\top \boldsymbol{\Gamma} \mathbf{k}_{sf}, \tag{5.6.15e}$$

$$\boldsymbol{\Gamma} \triangleq \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1} \left(\tilde{\boldsymbol{\Sigma}}_k - \mathbf{K} - \sigma_e^2 \mathbf{I}\right) \left(\mathbf{K} + \sigma_e^2 \mathbf{I}\right)^{-1}. \tag{5.6.15f}$$

Algorithm 11: FGP-based bicriterial dual control.

Input: Measurements $\{(\phi_k,\, y_{k+1})\}_{k=1}^{K}$, initial kernel parameters $\boldsymbol{\theta}_0$, probing gain parameter η.

Output: Sequence of control actions $\{u_k\}_{k=1}^{K}$.

for $k \leftarrow 1$ **to** K **do**

`// Kernel parameter optimization (starts from` $\boldsymbol{\theta}_0$`).`

$\boldsymbol{\theta}_k \leftarrow \arg\min_{\boldsymbol{\theta}} \log p(\,\mathbf{y}_k \,|\, \mathbf{X}_k,\, \boldsymbol{\theta}\,)$

`// Kernel evaluation.`

$k_{ssf} \leftarrow k_f\big(\mathbf{x}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$k_{ssg} \leftarrow k_g\big(\mathbf{x}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$\mathbf{k}_{sf} \leftarrow k_f\big(\mathbf{X}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$\mathbf{k}_{sg} \leftarrow k_g\big(\mathbf{X}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big) \circ \mathbf{u}_k$

$\mathbf{K} \leftarrow k_h\big(\boldsymbol{\Phi}_k,\, \boldsymbol{\Phi}_k;\, \boldsymbol{\theta}_k\big)$

`// Control action.`

$\hat{f}_{k+1} \leftarrow \mathbf{k}_{sf}^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\mathbf{y}_k$

$\hat{g}_{k+1} \leftarrow \mathbf{k}_{sg}^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\mathbf{y}_k$

$\mu_k \leftarrow k_{ssg} - \mathbf{k}_{sg}^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\mathbf{k}_{sg}$

$\nu_k \leftarrow 2\mathbf{k}_{sg}^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\mathbf{k}_{sf}$

$u_k^c \leftarrow \frac{\big(r_{k+1} - \hat{f}_{k+1}\big)\hat{g}_{k+1} + \frac{1}{2}\nu_k}{\hat{g}_{k+1}^2 + \mu_k}$

$u_k \leftarrow u_k^c + \eta(-2\mu_k u_k^c + \nu_k)$

`// System output prediction (if needed).`

$k_{ss} \leftarrow k_{ssf} + k_{ssg}u_k^2 + \sigma_e^2$

$\mathbf{k}_s \leftarrow \mathbf{k}_{sf} + \mathbf{k}_{sg}u_k$

$\hat{\mu}_{k+1} \leftarrow \mathbf{k}_s^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\mathbf{y}_k$

$\hat{\sigma}_{k+1}^2 \leftarrow k_{ss} - \mathbf{k}_s^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\mathbf{k}_s$

end

The cautious control component is given as

$$u_k^c = \frac{\left(r_{k+1} - \hat{f}_{k+1}\right)\hat{g}_{k+1} - \frac{1}{2}\nu_k}{\hat{g}_{k+1}^2 + \mu_k}. \tag{5.6.16}$$

The dual control is then

$$u_k = u_k^c + \eta \,\mathrm{sign}\,(2\mu_k u_k^c + \nu_k). \tag{5.6.17}$$

Notice the change of signs, when compared with eqs. (5.6.10) and (5.6.14), which is due to the fact that the quantities in eqs. (5.6.15a) and (5.6.15f) are defined differently.

The Algorithm 12 summarizes operation of the proposed bicriterial dual recursive Gaussian process (BD-RGP) adaptive controller. Note, that unlike in the BD-FGP case, the RGP model predictions on lines 25 to 29 are necessary for calculating the control action.

5.6.4 Numerical Experiments and Evaluation

The proposed bicriterial dual controllers were tested on the synthetic example [Fabri and Kadirkamanathan, 2001] of non-linear stochastic discrete-time system

$$y_{k+1} = \frac{1.5y_k}{1 + y_k^2} + (2 + \cos(y_k))u_k + e_{k+1}, \tag{5.6.18}$$

where e_{k+1} is a white Gaussian noise with variance $\sigma_e^2 = 0.0025$. In this case, the vector of regressors $\boldsymbol{\phi}_k = \begin{bmatrix} y_k & u_k \end{bmatrix}^\top$ is two-dimensional. The bicriterial dual controller with the full Gaussian process model (BD-FGP) as well as the recursive GP model (BD-RGP) used the same set of initial kernel parameter values $\boldsymbol{\theta} = \log\left(\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}\right)^\top$. The set of basis vectors for the BD-RGP was initialized as a rectangular grid of 7×5 points in the interval $[-2,\, 2] \times [-1,\, 1]$.

The main motivation for designing the BD-RGP algorithm was to alleviate the prohibitive computational demands of the BD-FGP. The Figure 5.6.3 is a result of averaging execution time measured in each control loop iteration of both algorithms over 100 Monte Carlo runs. Clearly, the demands of the BD-RGP algorithm remain constant, thus making it superior alternative to the BD-FGP in this regard.

The other experiment was focused on the assessment of control quality, which was

Algorithm 12: RGP-based bicriterial dual control.

Input: Measurements $\{(\boldsymbol{\phi}_k,\, y_{k+1})\}_{k=1}^{K}$, set of basis vectors $\tilde{\boldsymbol{\Phi}}$, initial kernel parameters $\boldsymbol{\theta}_0$, probing gain parameter η.

Output: Sequence of control actions $\{u_k\}_{k=1}^{K}$.

`// Initialization.`

$\tilde{\boldsymbol{\mu}}_0 \leftarrow \mathrm{Uniform}(0,\, 1)$

$\tilde{\boldsymbol{\Sigma}}_0 \leftarrow k(\tilde{\boldsymbol{\Phi}}, \tilde{\boldsymbol{\Phi}};\, \boldsymbol{\theta}_0)$

$\hat{\mu}_1 \leftarrow e_1$

$\hat{\sigma}_1^2 \leftarrow e_1$

$\mathbf{K} \leftarrow k(\tilde{\boldsymbol{\Phi}}, \tilde{\boldsymbol{\Phi}};\, \boldsymbol{\theta}_0)$

$\mathbf{k}_s \leftarrow k_f(\tilde{\mathbf{X}}, \mathbf{x}_0;\, \boldsymbol{\theta}_0) + k_g(\tilde{\mathbf{X}}, \mathbf{x}_0;\, \boldsymbol{\theta}_0)e_1$

$\mathbf{J}_1 \leftarrow \mathbf{k}_s\mathbf{K}^{-1}$

for $k \leftarrow 1$ **to** K **do**

`// RGP model update.`

$\mathbf{G}_k \leftarrow \tilde{\boldsymbol{\Sigma}}_{k-1}\mathbf{J}_k^{\top}\left(\hat{\sigma}_k^2 + \sigma_e^2\right)^{-1}$

$\tilde{\boldsymbol{\mu}}_k \leftarrow \tilde{\boldsymbol{\mu}}_{k-1} + \mathbf{G}_k(y_k - \hat{\mu}_k)$

$\tilde{\boldsymbol{\Sigma}}_k \leftarrow \tilde{\boldsymbol{\Sigma}}_{k-1} - \mathbf{G}_k\mathbf{J}_k\tilde{\boldsymbol{\Sigma}}_{k-1}$

`// Kernel parameter optimization (starts from` $\boldsymbol{\theta}_{k-1}$`).`

$\boldsymbol{\theta}_k \leftarrow \arg\min_{\boldsymbol{\theta}} \log p\Big(\, \tilde{\boldsymbol{\mu}}_k \,\Big|\, \tilde{\boldsymbol{\Phi}},\, \boldsymbol{\theta}\,\Big)$

`// Kernel evaluation.`

$k_{ssf} \leftarrow k_f\big(\mathbf{x}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$k_{ssg} \leftarrow k_g\big(\mathbf{x}_k,\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$\mathbf{k}_{sf} \leftarrow k_f\big(\tilde{\mathbf{X}},\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big)$

$\mathbf{k}_{sg} \leftarrow k_g\big(\tilde{\mathbf{X}},\, \mathbf{x}_k;\, \boldsymbol{\theta}_k\big) \circ \tilde{\mathbf{u}}$

$\mathbf{K} \leftarrow k_h\big(\tilde{\boldsymbol{\Phi}},\, \tilde{\boldsymbol{\Phi}};\, \boldsymbol{\theta}_k\big)$

`// Control action.`

$\hat{f}_{k+1} \leftarrow \mathbf{k}_{sf}^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\tilde{\boldsymbol{\mu}}_k$

$\hat{g}_{k+1} \leftarrow \mathbf{k}_{sg}^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\tilde{\boldsymbol{\mu}}_k$

$\boldsymbol{\Gamma} \leftarrow \big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}\big(\tilde{\boldsymbol{\Sigma}}_k - \mathbf{K} - \sigma_e^2\mathbf{I}\big)\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}$

$\mu_k \leftarrow k_{ssg} + \mathbf{k}_{sg}^{\top}\boldsymbol{\Gamma}\mathbf{k}_{sg}$

$\nu_k \leftarrow 2\mathbf{k}_{sg}^{\top}\boldsymbol{\Gamma}\mathbf{k}_{sf}$

$u_k^c \leftarrow \frac{\left(r_{k+1} - \hat{f}_{k+1}\right)\hat{g}_{k+1} - \frac{1}{2}\nu_k}{\hat{g}_{k+1}^2 + \mu_k}$

$u_k \leftarrow u_k^c + \eta(2\mu_k u_k^c + \nu_k)$

`// System output prediction.`

$k_{ss} \leftarrow k_{ssf} + k_{ssg}u_k^2 + \sigma_e^2$

$\mathbf{k}_s \leftarrow \mathbf{k}_{sf} + \mathbf{k}_{sg}u_k$

$\mathbf{J}_k \leftarrow \mathbf{k}_s^{\top}\big(\mathbf{K} + \sigma_e^2\mathbf{I}\big)^{-1}$

$\hat{\mu}_{k+1} \leftarrow \mathbf{J}_k\tilde{\boldsymbol{\mu}}_k$

$\hat{\sigma}_{k+1}^2 \leftarrow k_{ss} + \mathbf{J}_k\big(\tilde{\boldsymbol{\Sigma}}_k - \mathbf{K}\big)\mathbf{J}_k^{\top}$

end

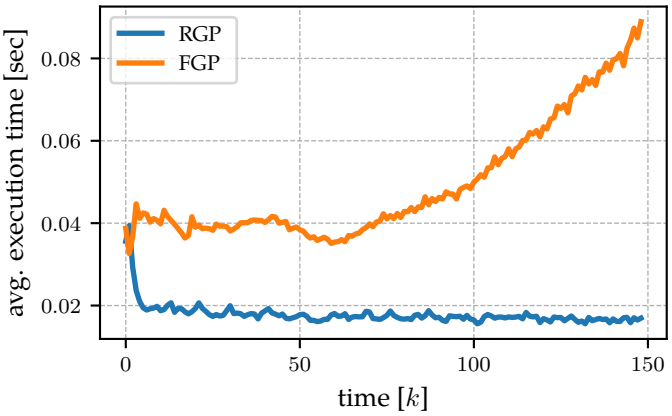

Figure 5.6.3: Average execution time per time step of the BD-FGP and BD-RGP control algorithms. The computational requirements of the BD-RGP controller remain constant, while the demands BD-FGP controller continue to grow indefinitely with every time step.

measured by

$$J_m = \frac{1}{K} \sum_{k=m}^{K} (y_{k+1} - r_{k+1})^2, \tag{5.6.19}$$

where $m \geq 1$ is the offset, which is meant to counteract the initial transition effects in the system response. We compared the control quality of the BD-FGP controller, which uses the exact non-recursive GP model, with the BD-RGP controller, which uses the approximate recursive GP model. Ideally, we would like the quality of the approximate BD-RGP to remain close enough to the exact BD-FGP. Each controller was simulated for $K = 60$ time steps and the reference signal r_k was chosen as a square wave for $k < 30$ and a sine wave for $30 \leq k \leq 60$.

Table 5.6.1 compares the control quality of the BD-FGP and the BD-RGP controllers for offsets $m = 1$, which evaluates the criterion using the whole trajectory, and $m = 15$, which ignores the first few steps to assess the behavior after the GP model adaptation. The quality of the BD-RGP is two orders of magnitude worse than that of the BD-FGP, when evaluating eq. (5.6.19) over the whole trajectory. This is caused by the longer adaptation of the RGP model. However, the values of the offset criteria indicate that this adaptation quickly disappears (after about 15 time steps) and the BD-RGP control quality is then close to the BD-FGP. The reason for this behavior is the fact, that the RGP model is an approximation of the exact FGP model, and as such takes longer to adapt (slower convergence). This intuition is corroborated by our earlier results published in [Prüher and Šimandl, 2014].

Figure 5.6.4 compares the two dual controllers in terms of the system output response

Controller	$\hat{J}_1$	$\mathbb{V}\left[\hat{J}_1\right]$	$\hat{J}_{15}$	$\mathbb{V}\left[\hat{J}_{15}\right]$
BD-FGP	1.08e-01	2.02e-04	1.12e-02	3.79e-05
BD-RGP	1.07e+01	9.66e+00	1.42e-02	1.01e-04

Table 5.6.1: Control quality of the bicriterial dual controller based on full (BD-FGP) and recursive Gaussian process (BD-RGP) model. Results are averaged over 100 Monte Carlo simulations. Variance estimates of the means of the criteria were obtained by bootstrap method.

to the reference signal. The output response for the BD-RGP is more erratic at the start, which explains the results in the Table 5.6.1, but then stays close to the BD-FGP output response. Figure 5.6.5 shows the evolution of the FGP and the RGP model predictive

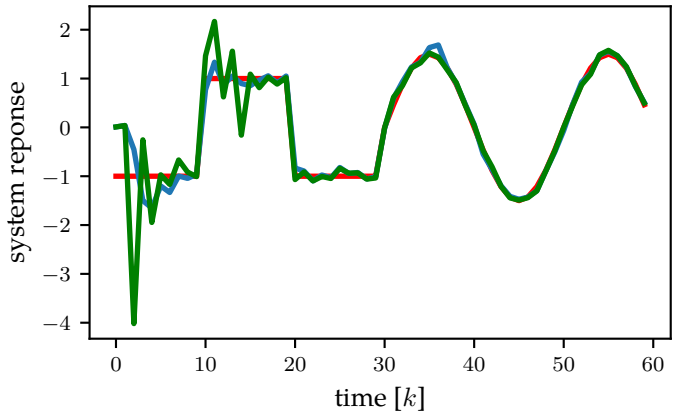

Figure 5.6.4: Comparison of the typical system output response y_{k+1} to a reference signal r_k for the BD-FGP and BD-RGP controllers. The controller using the approximate recursive GP model (BD-RGP) exhibits slower adaptation than the BD-FGP.

uncertainty. The RGP model provides more conservative predictive variances across the whole trajectory, which is a desirable behavior considering the RGP is an approximate model.

5.6.5 Conclusion

In this chapter, I addressed the **Goal 1** set out in Chapter 4. I proposed the use of the RGP regression model for reducing computational demands of the BD controller based on the full GP model, when applied for control of non-linear discrete time-invariant stochastic systems with functional uncertainties. The GP regression models are applied for the approximation of unknown functions in the system description. We utilize an approximation to the marginal likelihood, which makes it possible to keep the computational demands fixed when learning the kernel parameters. We derived the

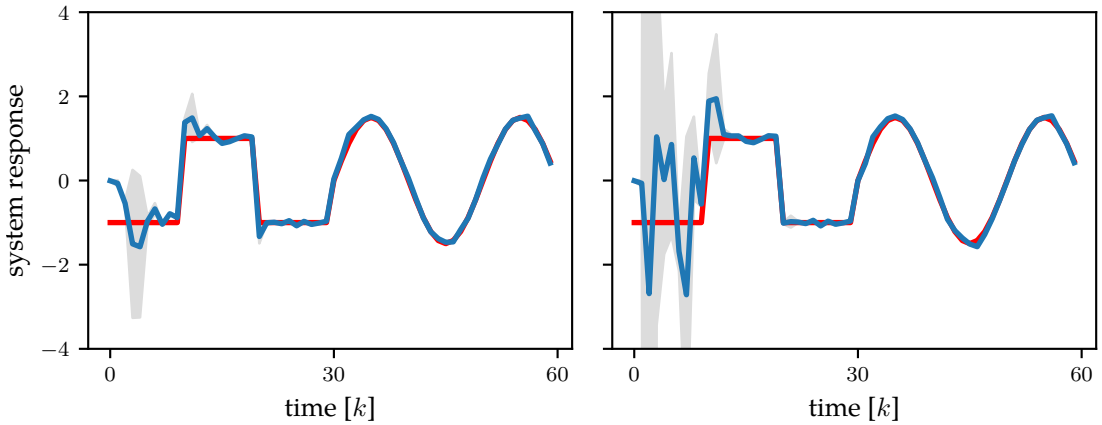

Figure 5.6.5: Comparison of model variances (gray band); the RGP model predictive variance (right) is more conservative than that of the FGP model (left). Since RGP model is an approximation of the FGP, this is a desirable behavior.

bicriterial dual controller with RGP model, whose computational demands do not increase with time. Even thought the BD-RGP controller takes longer to adapt, the control quality stays close to the baseline BD-FGP controller.

Chapter 6

Bayesian Quadrature Moment Transforms

Contributions in this chapter can be found in the following author's publications:

- J. Prüher and M. Šimandl. Bayesian Quadrature in Nonlinear Filtering. In *12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 01, pages 380–387, July 2015
- J. Prüher and M. Šimandl. Bayesian Quadrature Variance in Sigma-point Filtering. In *Informatics in Control, Automation and Robotics, 12th International Conference, Colmar, Alsace, France, 21-23 July, 2015 Revised Selected Papers*, volume 370 of *Lecture Notes in Electrical Engineering*. Springer International Publishing, 2016
- J. Prüher and O. Straka. Gaussian Process Quadrature Moment Transform. *IEEE Transactions on Automatic Control*, Pre-print(99):1–1, 2017. ISSN 0018-9286. doi: 10.1109/TAC.2017.2774444
- J. Prüher, F. Tronarp, T. Karvonen, S. Särkkä, and O. Straka. Student-t Process Quadratures for Filtering of Non-linear Systems with Heavy-tailed Noise. In *20th International Conference on Information Fusion (Fusion)*, pages 1–8, July 2017. doi: 10.23919/ICIF.2017.8009742

Throughout this chapter I rely heavily on the exposition of the state estimation preliminaries in Chapter 3 and mainly that of Bayesian quadrature in Section 3.3.

6.1 General Bayesian Quadrature Moment Transform

The purpose of this section is to present an abstracted general BQ moment transform framework that underlies both of the proposed moment transforms; namely, the GPQ as well as the TPQ. I assume the familiar setup from eq. (3.2.1), where a nonlinear transformation

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \tag{6.1.1}$$

is considered, with an input random variable $\mathbf{x}$ being distributed according to $p(\mathbf{x})$. Some sort of probabilistic model of the integrand $g(\mathbf{x})$ will be further assumed, in the sense that both the mean $\mathbb{E}_g[g(\mathbf{x})]$ and the variance $\mathbb{V}_g[g(\mathbf{x})]$ exist.

Let us first consider a case when the nonlinearity in eq. (3.2.1) is a scalar function $g(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$. Note, that in the following derivations we omit the conditioning on data and use the shorthand notation $\mathbb{E}_g[g(\mathbf{x})] \triangleq \mathbb{E}_g[g(\mathbf{x}) \mid \mathcal{D}]$, $\mathbb{V}_g[g(\mathbf{x})] \triangleq \mathbb{V}_g[g(\mathbf{x}) \mid \mathcal{D}]$. Since the source of uncertainty is now, not only in the input $\mathbf{x}$, but in the nonlinearity g as well, the transformed moments also need to reflect this fact. The general BQ transform approximates the moments as follows

$$\mathbb{E}[y] = \mathbb{E}_{\mathbf{x}}[g(\mathbf{x})] \approx \mathbb{E}_{g,\mathbf{x}}[g(\mathbf{x})] \tag{6.1.2a}$$

$$\mathbb{V}[y] = \mathbb{V}_{\mathbf{x}}[g(\mathbf{x})] \approx \mathbb{V}_{g,\mathbf{x}}[g(\mathbf{x})] \tag{6.1.2b}$$

$$\mathbb{C}[\mathbf{x}, y] = \mathbb{C}_{\mathbf{x}}[\mathbf{x}, g(\mathbf{x})] \approx \mathbb{C}_{g,\mathbf{x}}[\mathbf{x}, g(\mathbf{x})] \tag{6.1.2c}$$

where, using the law of total expectation and variance, we can further write

$$\mathbb{E}_{g,\mathbf{x}}[g(\mathbf{x})] = \mathbb{E}_g[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \mathbb{E}_{\mathbf{x}}[\mathbb{E}_g[g(\mathbf{x})]], \tag{6.1.3a}$$

$$\mathbb{V}_{g,\mathbf{x}}[g(\mathbf{x})] = \mathbb{E}_g[\mathbb{V}_{\mathbf{x}}[g(\mathbf{x})]] + \mathbb{V}_g[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] \tag{6.1.3b}$$

$$= \mathbb{E}_{\mathbf{x}}[\mathbb{V}_g[g(\mathbf{x})]] + \mathbb{V}_{\mathbf{x}}[\mathbb{E}_g[g(\mathbf{x})]], \tag{6.1.3c}$$

$$\mathbb{C}_{g,\mathbf{x}}[\mathbf{x}, g(\mathbf{x})] = \mathbb{E}_{\mathbf{x}}[\mathbf{x}\mathbb{E}_g[g(\mathbf{x})]] - \mathbb{E}_{\mathbf{x}}[\mathbf{x}]\mathbb{E}_{g,\mathbf{x}}[g(\mathbf{x})]. \tag{6.1.3d}$$

The eq. (6.1.3a) shows that the mean of the integral is equivalent to integrating the mean function. Since the variance decompositions in eqs. (6.1.3b) to (6.1.3c) are equivalent, both can be used to achieve the same goal. The form (6.1.3c) was utilized in the derivation of the GP-ADF [Deisenroth et al., 2012], which relies on the solution to the problem of prediction with GPs at uncertain inputs [Girard et al., 2003]. So, even though these results were derived to solve a seemingly different problem, by using the form (6.1.3c), the uncertainty of the mean integral (as seen in the last term of eq. (6.1.3b)) is implicitly reflected in the resulting covariance. Furthermore, the form (6.1.3c) is

preferable, because it is more amenable to analytical expression and implementation. In a deterministic case, when the *integrand* variance $\mathbb{V}_g[g(\mathbf{x})] = 0$ and the *integral* variance $\mathbb{V}_g[\mathbb{E}_\mathbf{x}[g(\mathbf{x})]] = 0$, the eqs. (6.1.3a) to (6.1.3d) fall back to the classical expressions given by eqs. (3.2.2a) to (3.2.2c). Compared to the deterministic case, the transformed BQ variance is inflated by the uncertainty in g.

Extension to the case of vector functions $\mathbf{g}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^E$ is fairly straightforward. The BQ transformed moments now become

$$\boldsymbol{\mu}_\mathrm{A} = \mathbb{E}_{\mathbf{g},\mathbf{x}}[\mathbf{g}(\mathbf{x})] \tag{6.1.4a}$$
$$= \mathbb{E}_\mathbf{g}[\mathbb{E}_\mathbf{x}[\mathbf{g}(\mathbf{x})]] = \mathbb{E}_\mathbf{x}[\mathbb{E}_\mathbf{g}[\mathbf{g}(\mathbf{x})]], \tag{6.1.4b}$$
$$\boldsymbol{\Pi}_\mathrm{A} = \mathbb{C}_{\mathbf{g},\mathbf{x}}[\mathbf{g}(\mathbf{x})] \tag{6.1.4c}$$
$$= \mathbb{E}_\mathbf{g}[\mathbb{C}_\mathbf{x}[\mathbf{g}(\mathbf{x})]] + \mathbb{C}_\mathbf{g}[\mathbb{E}_\mathbf{x}[\mathbf{g}(\mathbf{x})]] \tag{6.1.4d}$$
$$= \mathbb{E}_\mathbf{x}[\mathbb{C}_\mathbf{g}[\mathbf{g}(\mathbf{x})]] + \mathbb{C}_\mathbf{x}[\mathbb{E}_\mathbf{g}[\mathbf{g}(\mathbf{x})]], \tag{6.1.4e}$$
$$\mathbf{C}_\mathrm{A} = \mathbb{C}_{\mathbf{g},\mathbf{x}}[\mathbf{x},\, \mathbf{g}(\mathbf{x})] \tag{6.1.4f}$$
$$= \mathbb{E}_\mathbf{x}\left[\mathbf{x}\mathbb{E}_\mathbf{g}[\mathbf{g}(\mathbf{x})]^\top\right] - \mathbb{E}_\mathbf{x}[\mathbf{x}]\mathbb{E}_\mathbf{x}[\mathbb{E}_\mathbf{g}[\mathbf{g}(\mathbf{x})]]. \tag{6.1.4g}$$

6.2 Gaussian Process Quadrature Moment Transform

I propose a moment transform based on the GPQ (see Section 3.3.1). First, I define a general algorithm, which works with any kernel function, and then give relations for a GPQ transform based on the popular RBF kernel. As a remark, consider the fact that a vector function $\mathbf{g}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^E$ can be written as

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g^1(\mathbf{x}) & \dots & g^e(\mathbf{x}) & \dots & g^E(\mathbf{x}) \end{bmatrix}. \tag{6.2.1}$$

Throughout this section I will assume that

A1: the input variable is Gaussian distributed with $\mathbf{x} \sim \mathrm{N}(\mathbf{m},\, \mathbf{P})$,

A2: the stochastic decoupling substitution

$$\mathbf{x} = \mathbf{m} + \mathbf{L}\boldsymbol{\xi} \quad \text{where} \quad \mathbf{P} = \mathbf{L}\mathbf{L}^\top \quad \text{and} \quad \boldsymbol{\xi} \sim \mathrm{N}(\mathbf{0},\, \mathbf{I})$$

in the integrals has taken place, and

A3: the integrand is GP distributed according to $g^e(\boldsymbol{\xi}) \sim \mathrm{GP}\big(0,\, k(\boldsymbol{\xi},\, \boldsymbol{\xi}';\, \boldsymbol{\theta})\big)$ for $e = 1, \dots, E$.

The last assumption essentially means that a single GP[1] is used to model every output dimension of the integrand.

Expressions for the GPQ transformed moments are derived by plugging in the GP predictive moments from eqs. (2.2.11a) to (2.2.11b) into the general expressions in eqs. (6.1.4b) to (6.1.4g). The transformed mean in eq. (6.1.4b) thus becomes

$$\boldsymbol{\mu}_{\mathrm{A}} = \mathbb{E}_{\boldsymbol{\xi}}[\mathbb{E}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]] = \mathbf{Y}^{\top}\mathbf{K}^{-1}\mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\big] = \mathbf{Y}^{\top}\mathbf{w}, \tag{6.2.2}$$

where the e-th column of $\mathbf{Y}$, given by $\begin{bmatrix} y_1^e & \dots & y_N^e \end{bmatrix}^{\top}$, comprises function values of the e-th output of $\mathbf{g}(\mathbf{x})$ and $[\mathbf{K}]_{nm} = k(\boldsymbol{\xi}_n, \boldsymbol{\xi}_m)$. Using eq. (6.1.4e), the transformed covariance can be written as

$$\begin{aligned}
\boldsymbol{\Pi}_{\mathrm{A}} &= \mathbb{C}_{\boldsymbol{\xi}}[\mathbb{E}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]] + \mathbb{E}_{\boldsymbol{\xi}}[\mathbb{C}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]] \\
&= \mathbb{E}_{\boldsymbol{\xi}}\big[\mathbb{E}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]\mathbb{E}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]^{\top}\big] - \boldsymbol{\mu}_{\mathrm{A}}\boldsymbol{\mu}_{\mathrm{A}}^{\top} + \mathbb{E}_{\boldsymbol{\xi}}[\mathbb{C}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]] \\
&= \mathbf{Y}^{\top}\mathbf{K}^{-1}\mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\mathbf{k}^{\top}(\boldsymbol{\xi}')\big]\mathbf{K}^{-1}\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}}\boldsymbol{\mu}_{\mathrm{A}}^{\top} + \sigma^2\mathbf{I},
\end{aligned} \tag{6.2.3}$$

where

$$\sigma^2 = \mathbb{E}_{\boldsymbol{\xi}}\big[\sigma_g^2(\boldsymbol{\xi})\big] = \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi})] - \operatorname{tr}\left(\mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\mathbf{k}^{\top}(\boldsymbol{\xi})\big]\mathbf{K}^{-1}\right). \tag{6.2.4}$$

The diagonal matrix in the last term of eq. (6.2.3) reflects the fact that the outputs of $\mathbf{g}$, as seen in eq. (6.2.1), are not correlated. Finally, the cross-covariance in eq. (6.1.4g) becomes

$$\begin{aligned}
\mathbf{C}_{\mathrm{A}} &= \mathbb{E}_{\boldsymbol{\xi}}\big[\boldsymbol{\xi}\mathbb{E}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]^{\top}\big] - \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi}]\mathbb{E}_{\boldsymbol{\xi}}[\mathbb{E}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]], \\
&= \mathbf{L}\mathbb{E}_{\boldsymbol{\xi}}\big[\boldsymbol{\xi}\mathbf{k}^{\top}(\boldsymbol{\xi})\big]\mathbf{K}^{-1}\mathbf{Y}.
\end{aligned} \tag{6.2.5}$$

In summary, the proposed general GPQ-based Gaussian approximation of the joint distribution of $\mathbf{x}$ and the transformed random variable $\mathbf{y} = \mathbf{g}(\mathbf{x})$, where $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$, is given by

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{A}} \end{bmatrix}, \begin{bmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{A}} \\ \mathbf{C}_{\mathrm{A}}^{\top} & \boldsymbol{\Pi}_{\mathrm{A}} \end{bmatrix}\right), \tag{6.2.6}$$

where the transformed moments are computed by the Algorithm 13. Note, that the unit sigma-points $\boldsymbol{\xi}_n$ can be chosen arbitrarily, because, unlike the classical quadrature, the BQ does not prescribe any. The advantage to using the decoupling substitution (see eq. (3.2.8)) is that it allows to formulate the GPQ weights as independent of the input moments, which becomes crucial in filtering applications.

[1] same values of kernel parameters

Algorithm 13: Gaussian process quadrature moment transform. The function takes the input mean $\mathbf{m}$ and covariance $\mathbf{P}$ together with the kernel parameters $\boldsymbol{\theta}$ and unit sigma-points $\boldsymbol{\xi}_n$ arranged in columns of a matrix $\boldsymbol{\Xi}$ to produce the transformed moments $\boldsymbol{\mu}_\mathrm{A},\ \boldsymbol{\Pi}_\mathrm{A},\ \mathbf{C}_\mathrm{A}$.

Function `GPQMT(`$\mathbf{m}$, $\mathbf{P}$, $\boldsymbol{\theta}$, $\boldsymbol{\Xi}$`)`

  `// Form sigma-points.`

  $\mathbf{L} \leftarrow$ `MatrixFactor(`$\mathbf{P}$`)`

  $\mathbf{X} \leftarrow \mathbf{m} + \mathbf{L}\boldsymbol{\Xi}$

  `// Evaluate nonlinearity at sigma-points.`

  $\mathbf{Y} \leftarrow \mathbf{g}(\mathbf{X})$

  `// Evaluate kernel expectations.`

  $\bar{k} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi}\,; \boldsymbol{\theta})]$

  for $n \leftarrow 1$ **to** N **do**

    $[\mathbf{q}]_n \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

    $[\mathbf{R}]_{*n} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi} k(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

    for $m \leftarrow 1$ **to** N **do**

      $[\mathbf{Q}]_{nm} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta}) k(\boldsymbol{\xi}, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})]$

      $[\mathbf{K}]_{nm} \leftarrow k(\boldsymbol{\xi}_n, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})$

    end

  end

  `// Compute GPQ weights.`

  $\mathbf{w} \leftarrow \mathbf{K}^{-1}\mathbf{q}$

  $\mathbf{W} \leftarrow \mathbf{K}^{-1}\mathbf{Q}\mathbf{K}^{-1}$

  $\mathbf{W}_c \leftarrow \mathbf{R}\mathbf{K}^{-1}$

  `// Compute transformed moments.`

  $\sigma^2 \leftarrow \bar{k} - \mathrm{tr}\left(\mathbf{Q}\mathbf{K}^{-1}\right)$

  $\boldsymbol{\mu}_\mathrm{A} \leftarrow \mathbf{Y}^\top \mathbf{w}$

  $\boldsymbol{\Pi}_\mathrm{A} \leftarrow \mathbf{Y}^\top \mathbf{W}\mathbf{Y} - \boldsymbol{\mu}_\mathrm{A}\boldsymbol{\mu}_\mathrm{A}^\top + \sigma^2\mathbf{I}$

  $\mathbf{C}_\mathrm{A} \leftarrow \mathbf{L}\mathbf{W}_c\mathbf{Y}$

  return $\boldsymbol{\mu}_\mathrm{A}, \boldsymbol{\Pi}_\mathrm{A}, \mathbf{C}_\mathrm{A}$

end

Evidently, the GPQ moment transform hinges upon the kernel expectations as seen in lines 5 to 10. Since we are already using one quadrature to approximate the moments, it is thus preferable that these expectations be analytically tractable. A list of tractable kernel-density pairs is provided in [Briol et al., 2015]. A popular choice in many applications is an RBF kernel, expectations of which are summarized below.

Theorem 1 (GPQ transform with RBF kernel)**.** *Assuming a change of variables has taken place in the Gaussian weighted integrals given by eq.* (3.2.8) *and the kernel is of the form*

$$k(\boldsymbol{\xi}, \boldsymbol{\xi}') = \alpha^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\xi}')^\top \boldsymbol{\Lambda}^{-1}(\boldsymbol{\xi} - \boldsymbol{\xi}')\right), \tag{6.2.7}$$

where α is a scaling parameter and $\boldsymbol{\Lambda} = \operatorname{diag}\left(\begin{bmatrix}\lambda_1^2 & \dots & \lambda_D^2\end{bmatrix}\right)$ is a lengthscale, then the expectations given in lines 5 to 10 of Algorithm 13 take on the form

$$[\mathbf{q}]_i = C_1 \exp\left(-\tfrac{1}{2}\boldsymbol{\xi}_i^\top (\boldsymbol{\Lambda} + \mathbf{I})^{-1}\boldsymbol{\xi}_i\right), \tag{6.2.8a}$$

$$[\mathbf{Q}]_{ij} = C_2 \exp\left(-\tfrac{1}{2}\left(\boldsymbol{\xi}_i^\top \boldsymbol{\Lambda}^{-1}\boldsymbol{\xi}_i + \boldsymbol{\xi}_j \boldsymbol{\Lambda}^{-1}\boldsymbol{\xi}_j - \mathbf{z}_{ij}^\top \left(2\boldsymbol{\Lambda}^{-1} + \mathbf{I}\right)^{-1}\mathbf{z}_{ij}\right)\right), \tag{6.2.8b}$$

$$[\mathbf{R}]_{*j} = C_1 \exp\left(-\tfrac{1}{2}\boldsymbol{\xi}_j^\top (\boldsymbol{\Lambda} + \mathbf{I})^{-1}\boldsymbol{\xi}_j\right)(\boldsymbol{\Lambda} + \mathbf{I})^{-1}\boldsymbol{\xi}_j, \tag{6.2.8c}$$

$$\bar{k} = \alpha^2 \tag{6.2.8d}$$

where $C_1 = \alpha^2\left|\boldsymbol{\Lambda}^{-1} + \mathbf{I}\right|^{-\frac{1}{2}}$, $C_2 = \alpha^4\left|2\boldsymbol{\Lambda}^{-1} + \mathbf{I}\right|^{-\frac{1}{2}}$ and $\mathbf{z}_{ij} = \boldsymbol{\Lambda}^{-1}(\boldsymbol{\xi}_i + \boldsymbol{\xi}_j)$.

Proof. The expressions can be derived by writing the RBF kernel as a Gaussian density and making use of the formulas for the product of two Gaussian densities (and the normalizing constant). Derivations are confined to the appendix in Appendix A, so as not to detract from the flow of the text (see also [Deisenroth, 2009]). □

An important requirement is for moment transforms to produce a valid covariance matrices. Theorem 3, given below, states that the proposed GPQ transform always produces a positive semi-definite covariance matrix. For the proof, I use the following lemma.

Lemma 2. *For any $m \times n$ matrix $\mathbf{X}$ and a positive definite $n \times n$ matrix $\mathbf{A}$, the matrix $\mathbf{X}\mathbf{A}\mathbf{X}^\top$ is positive semi-definite.*

Proof. See [Horn and Johnson, 1990, Observation 7.1.6, p. 399]. □

In the following, let $\mathbf{A} \succeq 0 \;\Leftrightarrow\; \mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0,\ \forall \mathbf{x} \in \mathbb{R}^n$ for any $n \times n$ matrix $\mathbf{A}$.

Theorem 3. *The GPQ transformed covariance on the line 31 of Algorithm 13 is positive semi-definite.*

Proof. Using the expressions for the GPQ weights on lines 26 to 28 of Algorithm 13, we can write

$$\mathbf{\Pi} = \mathbf{Y}^\top \mathbf{K}^{-1}\big(\mathbf{Q} - \mathbf{q}\mathbf{q}^\top\big)\mathbf{K}^{-1}\mathbf{Y} + \sigma^2\mathbf{I} = \mathbf{Z}^\top\widetilde{\mathbf{Q}}\mathbf{Z} + \sigma^2\mathbf{I} = \widetilde{\mathbf{\Pi}} + \sigma^2\mathbf{I}, \tag{6.2.9}$$

where $\widetilde{\mathbf{\Pi}} = \mathbf{Z}^\top\widetilde{\mathbf{Q}}\mathbf{Z},\ \mathbf{Z} = \mathbf{K}^{-1}\mathbf{Y}$ and $\widetilde{\mathbf{Q}} = \mathbf{Q} - \mathbf{q}\mathbf{q}^\top$. We recognize that

$$\widetilde{\mathbf{Q}} = \mathbb{C}[\mathbf{k}(\mathbf{x})] = \mathbb{E}\big[\mathbf{k}(\mathbf{x})\mathbf{k}^\top(\mathbf{x})\big] - \mathbb{E}[\mathbf{k}(\mathbf{x})]\mathbb{E}[\mathbf{k}(\mathbf{x})]^\top. \tag{6.2.10}$$

From the property of covariance matrices it follows that $\widetilde{\mathbf{Q}} \succeq 0$. The Lemma 2 implies that $\widetilde{\mathbf{\Pi}} = \mathbf{Z}^\top\widetilde{\mathbf{Q}}\mathbf{Z} \succeq 0$ for any matrix $\mathbf{Z}$. Finally, since $\sigma^2 \geq 0$, we have that $\mathbf{\Pi} = \widetilde{\mathbf{\Pi}} + \sigma^2\mathbf{I} \succeq 0$. □

6.2.1 Choice of Kernel Parameters

It is now evident that the GPQ transformed moments depend on the kernel parameters, which need to be set prior to computing the weights. A typical approach in the GP regression would be to optimize the kernel parameters by marginal likelihood (evidence) maximization. However, in the BQ setting this method would likely yield unreliable parameter estimates due to the inherently limited amount of available data. For these reasons, I resorted to a manual choice of the parameter values, which were mostly informed by the prior knowledge of the integrated function.

In the following theorem, I prove the independence of the GPQ weights on the kernel scaling parameter.

Theorem 4 (Kernel scaling independence)**.** *Assume a scaled version of a kernel is used, so that $\bar{k}(\mathbf{x}, \mathbf{x}') = c \cdot k(\mathbf{x}, \mathbf{x}')$, then the weights of the GPQ transform given on lines 26 to 28 of Algorithm 13 are independent of the scaling parameter c.*

Proof. Define a scaled kernel matrix $\mathbf{K}' = c\mathbf{K}$, and scaled kernel expectations $[\mathbf{q}']_n = c\mathbb{E}_\mathbf{x}[k(\mathbf{x}, \mathbf{x}_n)],\ [\mathbf{Q}']_{nm} = c^2\mathbb{E}_\mathbf{x}[k(\mathbf{x}, \mathbf{x}_n)k(\mathbf{x}, \mathbf{x}_m)],\ [\mathbf{R}']_{*m} = c\mathbb{E}_\mathbf{x}[\mathbf{x}k(\mathbf{x}, \mathbf{x}_m)]$. Plugging

into the expressions for the GPQ weights given on the lines 26 to 28, we get

$$\mathbf{w}' = \mathbf{q}'\big(\mathbf{K}'\big)^{-1} = cc^{-1}\mathbf{q}\mathbf{K}^{-1} = \mathbf{w}, \tag{6.2.11a}$$

$$\begin{aligned}\mathbf{W}' &= \big(\mathbf{K}'\big)^{-1}\mathbf{Q}'\big(\mathbf{K}'\big)^{-1} \\ &= c^2c^{-2}\mathbf{K}^{-1}\mathbf{Q}\mathbf{K}^{-1} = \mathbf{W},\end{aligned} \tag{6.2.11b}$$

$$\mathbf{W}_c' = \mathbf{R}'\big(\mathbf{K}'\big)^{-1} = cc^{-1}\mathbf{R}\mathbf{K}^{-1} = \mathbf{W}_c. \tag{6.2.11c}$$

□

Corollary 5. *The kernel scaling affects only the additive term in the transformed covariance, which becomes* $\sigma^2 = c\big[\bar{k} - \mathrm{tr}\left(\mathbf{Q}\mathbf{K}^{-1}\right)\big]$. *The transformed mean on the line 30 and cross-covariance on the line 32 are unaffected by the scaling.*

Since the scaling has no effect on the GPQ weights, the main attention is given to the input lengthscales. The form of $\mathbf{\Lambda}$ in the RBF kernel formulation above exhibits, so called, automatic relevance determination (ARD). That is to say, by optimizing the lengthscales ℓ_d, dimensions contributing most to the variability in the data can be discovered, where a small ℓ_d would indicate high relevance of the d-th dimension and vice versa. As an initial guideline, I considered a priori knowledge of the nonlinear transformations encountered in the experiments. Whenever a certain output is observed to be linearly dependent on d-th input, the lengthscale parameter ℓ_d should be set to a large value relative to all other lengthscales $\{\ell_e : e \neq d\}$. Conversely, when strong nonlinearity is encountered a relatively small value of lengthscale should be selected. A several proposed nonlinearity measures [Duník et al., 2016] could be utilized as an aid for assessing severity of the nonlinearity in question.

Further point to consider, are the numerical issues with calculating the kernel matrix inverse. Choosing a lengthscale that is too high would cause the kernel matrix to become singular. Generally speaking, the smaller the distance between the sigma-points, the lower is the upper bound on the numerically allowed lengthscale.

With these guidelines in mind, the parameters can be further fine-tuned by evaluating relevant performance criteria, which in all cases involves running preliminary simulations. As an example of this process, consider Figure 6.2.3 which illustrates sensitivity of the GPQ filter performance to changing lengthscale.

6.2.2 Experiments

The proposed GPQ moment transform is first tested on a polar-to-Cartesian coordinate transformation, while the later experiments focus on its applications in the nonlinear

filtering. In all cases, the GPQ transform uses the RBF kernel given by eq. (6.2.7). Since the sigma-point locations are not prescribed and their choice is entirely arbitrary, I used the point-sets of the classical rules mentioned in Section 3.2 for all examples.

The *Gaussian process quadrature Kalman filter* (GPQKF) is obtained by replacing the `MomentTransform()` routine in Algorithm 1 with the GPQ transform `GPQMT()`, in Algorithm 13. The GPQKF is an umbrella acronym for all nonlinear Kalman filters based on the GPQ regardless of which point-set they use.

Mapping from Polar to Cartesian Coordinates

The conversion from polar to Cartesian coordinates is a ubiquitous nonlinearity appearing in radar sensors or laser range finders and is given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}. \tag{6.2.12}$$

Since the mapping is conditionally linear (for a fixed θ) and we use a kernel with ARD in the moment transform, we can exploit this fact and set the kernel lengthscales to $\mathbf{\Lambda} = \operatorname{diag}\big(\begin{bmatrix}60 & 6\end{bmatrix}\big)$ while the scaling was set to $\alpha = 1$. Note, that we set the lengthscale corresponding to the range to a relatively large value. This is because the larger lengthscales in the kernel correspond to a slower variation in the approximated function.

I compared the performance of the spherical radial transform (SR), which is basis of the cubature Kalman filter [Arasaratnam and Haykin, 2009], and the GPQ transform with SR points (GPQ-SR) for 100 different input moments. The 10 different positions on a spiral in polar coordinates were chosen as input means $\mathbf{m}_i = \begin{bmatrix} r_i & \theta_i \end{bmatrix}$. For each mean I assigned 10 different input covariance matrices $\mathbf{P}_j = \operatorname{diag}\big(\begin{bmatrix}\sigma_r^2 & \sigma_{\theta,j}^2\end{bmatrix}\big)$, where $\sigma_r = 0.5\,\mathrm{m}$ and azimuth standard deviations were uniformly placed in the interval $\sigma_{\theta,j} \in [6^\circ,\, 36^\circ]$ for $j = 1, \ldots, 10$. Figure 6.2.1 depicts the input means in polar coordinates. As a measure of an agreement between the approximate moments $(\boldsymbol{\mu}_\mathrm{A},\, \mathbf{\Pi}_\mathrm{A})$ and the ground-truth moments $(\boldsymbol{\mu},\, \mathbf{\Pi})$ I used the symmetrized KL-divergence of two Gaussian densities given by

$$\begin{aligned} \mathrm{SKL} &= \frac{1}{2}\{\mathbb{KL}[\mathrm{N}(\mathbf{y}\,|\,\boldsymbol{\mu},\, \mathbf{\Pi})\,\|\,\mathrm{N}(\mathbf{y}\,|\,\boldsymbol{\mu}_\mathrm{A},\, \mathbf{\Pi}_\mathrm{A})] + \mathbb{KL}[\mathrm{N}(\mathbf{y}\,|\,\boldsymbol{\mu}_\mathrm{A},\, \mathbf{\Pi}_\mathrm{A})\,\|\,\mathrm{N}(\mathbf{y}\,|\,\boldsymbol{\mu},\, \mathbf{\Pi})]\} \\ &= \frac{1}{4}\Big\{(\boldsymbol{\mu} - \boldsymbol{\mu}_\mathrm{A})^\top \mathbf{\Pi}^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}_\mathrm{A}) + (\boldsymbol{\mu}_\mathrm{A} - \boldsymbol{\mu})^\top \mathbf{\Pi}_\mathrm{A}^{-1}(\boldsymbol{\mu}_\mathrm{A} - \boldsymbol{\mu}) \\ &\qquad + \operatorname{tr}(\mathbf{\Pi}^{-1}\mathbf{\Pi}_\mathrm{A}) + \operatorname{tr}(\mathbf{\Pi}_\mathrm{A}^{-1}\mathbf{\Pi}) - 2E\Big\}, \end{aligned} \tag{6.2.13}$$

where $E = \dim(\mathbf{y})$. The ground truth transformed mean and covariance were computed

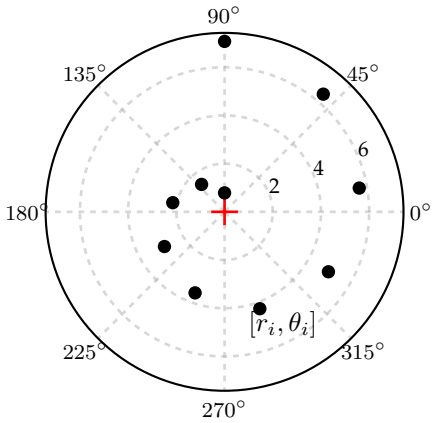

Figure 6.2.1: Input means are placed on a spiral. For each input mean $\mathbf{m}_i = \begin{bmatrix} r_i & \theta_i \end{bmatrix}$ (black dot) the radius variance is fixed at $\sigma_r = 0.5\,\mathrm{m}$ and 10 different azimuth variances are considered so that $\sigma_\theta \in [6°, 36°]$.

using the Monte Carlo method with $10\,000$ samples. Two SKL scores were considered; the average over means and an average over azimuth variances.

The Figure 6.2.2 shows the SKL score calculated for each configuration on the spiral. The left pane of Figure 6.2.2 shows results for individual means averaged over the

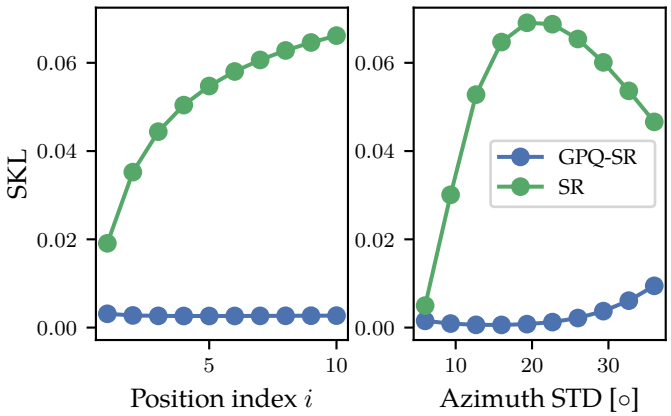

Figure 6.2.2: Performance comparison of the spherical radial (SR) and GPQ with SR points (GPQ-SR) moment transforms in terms of averaged symmetrized KL-divergence. Left: average over a range of azimuth variances; Right: average over the range of input means (positions on the spiral).

azimuth variances, whereas the right pane displays averaged SKL over the means. In both cases our proposed moment transform outperforms the classical quadrature transform with the same SR point-set.

Univariate Non-Stationary Growth Model

The performance of nonlinear sigma-point filters based on the GPQ transform was first tested in [Prüher and Šimandl, 2016] on a univariate non-stationary growth model (UNGM), where the system dynamics and the observation model are given by

$$x_k = \frac{1}{2}x_{k-1} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2\,k) + q_{k-1}, \tag{6.2.14a}$$

$$z_k = \frac{1}{20}x_{k-1}^2 + r_k, \tag{6.2.14b}$$

with the state noise $q_{k-1} \sim \mathrm{N}(0,\, 10)$, the measurement noise $r_k \sim \mathrm{N}(0,\, 1)$ and initial conditions $x_{0|0} \sim \mathrm{N}(0,\, 5)$. This model is frequently used as a benchmark for the particle filtering algorithms [Gordon et al., 1993; Kitagawa, 1996].

For this problem, all of the considered GPQKFs used the same kernel scaling $\alpha = 1$. The lengthscale was set to $\lambda = 3.0$ for the UT, $\lambda = 0.3$ for SR and GH-5, and $\lambda = 0.1$ for all higher-order GH sigma-points. The GPQKFs that used the UT and GH sigma-points of order 5, 7, 10, 15 and 20 were compared with their classical quadrature-based counterparts, namely, the UKF and the GHKF of the same orders. The UKF operated with $\kappa = 0$. We performed 100 simulations, each for $K = 500$ time steps. As a baseline for comparison, I used the bootstrap filter with stratified resampling and $10\,000$ particles (BS-PF).

For evaluation of the filter performance, I used the root-mean-square error (RMSE)

$$\mathrm{RMSE} = \left(\frac{1}{K} \sum_{k=1}^{K} \|\mathbf{x}_k - \mathbf{m}_{k|k}^x\|^2 \right)^{1/2}. \tag{6.2.15}$$

to measure the overall difference between the state estimate $\mathbf{m}_{k|k}^x$ and the true state $\mathbf{x}_k$ across all time steps. Since the BQ-based MTs, are primarily focused on incorporating additional uncertainty by inflating the estimated covariance, I used the inclination indicator (INC) [Li and Zhao, 2006] as a metric which takes into account the estimated state covariance. The indicator is given by

$$\mathrm{INC} = \frac{10}{K} \sum_{k=1}^{K} \log_{10} \frac{\big(\mathbf{x}_k - \mathbf{m}_{k|k}^x\big)^\top \big(\mathbf{P}_{k|k}^x\big)^{-1} \big(\mathbf{x}_k - \mathbf{m}_{k|k}^x\big)}{\big(\mathbf{x}_k - \mathbf{m}_{k|k}^x\big)^\top \boldsymbol{\Sigma}_k^{-1} \big(\mathbf{x}_k - \mathbf{m}_{k|k}^x\big)}, \tag{6.2.16}$$

where $\boldsymbol{\Sigma}_k$ is the sample mean-square-error (MSE) matrix, which can be computed from samples of the true system state trajectories. When the indicator is $\mathrm{INC} = 0$ the estimator is said to be *balanced*, which is to say that the estimated covariance is on average equal

Point set	N	GPQ	Classical
BS-PF	10000	–	5.657 ± 0.032
SR	2	6.157 ± 0.071	13.652 ± 0.253
UT	3	7.124 ± 0.131	7.103 ± 0.130
GH5	5	8.371 ± 0.128	10.466 ± 0.198
GH7	7	8.360 ± 0.043	9.919 ± 0.215
GH10	10	7.082 ± 0.038	8.035 ± 0.193
GH15	15	6.944 ± 0.048	8.224 ± 0.188
GH20	20	6.601 ± 0.058	7.406 ± 0.193

Table 6.2.1: The average root-mean-square error.

Point set	N	GPQ	Classical
BS-PF	10000	–	9.211 ± 0.196
SR	2	3.328 ± 0.026	56.570 ± 2.728
UT	3	4.970 ± 0.343	5.306 ± 0.481
GH5	5	4.088 ± 0.064	14.722 ± 0.829
GH7	7	4.045 ± 0.017	12.395 ± 0.855
GH10	10	3.530 ± 0.012	7.565 ± 0.534
GH15	15	3.468 ± 0.014	7.142 ± 0.557
GH20	20	3.378 ± 0.017	5.664 ± 0.488

Table 6.2.2: The average negative log-likelihood.

to the true state MSE matrix. For $\mathrm{INC} > 0$ the estimator is said to be *optimistic*, while for $\mathrm{INC} < 0$ it is considered *pessimistic*. Finally, the negative log-likelihood (NLL) of the state estimate $\mathbf{m}^x_{k|k}$ and covariance $\mathbf{P}^x_{k|k}$

$$\mathrm{NLL} = \frac{1}{2}\Big[\log\left|2\pi\mathbf{P}^x_{k|k}\right| + \left(\mathbf{x}_k - \mathbf{m}^x_{k|k}\right)^\top \left(\mathbf{P}^x_{k|k}\right)^{-1}\left(\mathbf{x}_k - \mathbf{m}^x_{k|k}\right)\Big] \tag{6.2.17}$$

was used to measure the overall model fit [Gelman et al., 2013].

Tables 6.2.1 to 6.2.3 show average values of the performance criteria across simulations with bootstrapped estimates of ± 2 standard deviations [Wasserman, 2007]. Note that N in the tables denotes the number of sigma-points. As evidenced by the results in Table 6.2.1, the Bayesian quadrature achieves superior RMSE performance for all sigma-point sets. In the classical quadrature case the performance improves with increasing number of sigma-points used. Table 6.2.2 shows that the performance of GPQKF is clearly superior in terms of NLL, which indicates that the estimates produced by the GPQ-based filters are better representations of the unknown true state development.

Point set	N	GPQ	Classical
BS-PF	10000	–	-12.838 ± 0.016
SR	2	1.265 ± 0.010	18.585 ± 0.045
UT	3	0.363 ± 0.108	0.897 ± 0.088
GH5	5	4.549 ± 0.013	9.679 ± 0.068
GH7	7	4.638 ± 0.006	8.409 ± 0.076
GH10	10	2.520 ± 0.006	5.315 ± 0.058
GH15	15	2.331 ± 0.008	5.424 ± 0.059
GH20	20	1.654 ± 0.007	4.105 ± 0.055

Table 6.2.3: The average inclination indicator.

The self-assessment of the filter performance is more credible in the case of GPQ, as indicated by lower inclination ν in the Table 6.2.3. This indicates that the GPQ-based filters are more conservative in their covariance estimates - a consequence of including additional uncertainty (integral variance), which the classical quadrature-based filters do not employ. Also note, that the variance of all the evaluated criteria for GPQ-based filters is mostly an order of magnitude lower.

Sensitivity of the performance for the GPQKF with UT sigma-points to changing kernel lengthscale is shown in Figure 6.2.3.

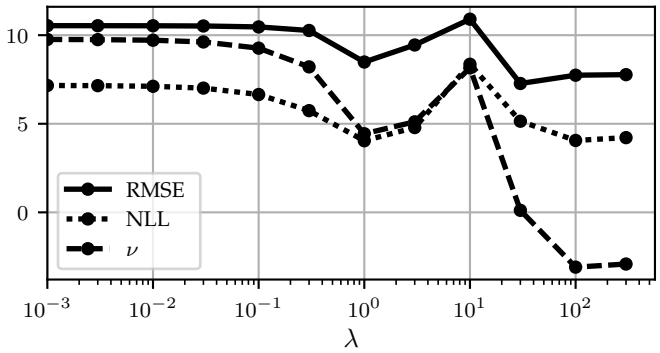

Figure 6.2.3: Sensitivity of GPQKF performance (using UT sigma-points) to changes in the lengthscale parameter λ. The choice $\lambda = 3$ minimizes RMSE and yields nearly optimal inclination ν.

Target Tracking

As a more application oriented example, I considered a target tracking scenario adopted from [Athans et al., 1968; Julier et al., 2000]. A spherical object is falling down from high altitude entering the Earth's atmosphere with a high velocity. The nonlinear dynamics

is described by the following set of differential equations

$$\dot{p}(t) = -v(t) + q_1(t), \tag{6.2.18a}$$
$$\dot{v}(t) = -v^2(t)\theta(t)\exp\left(-\gamma p(t)\right) + q_2(t), \tag{6.2.18b}$$
$$\dot{\theta}(t) = q_3(t), \tag{6.2.18c}$$

where $\gamma = 0.164$ is a constant and the system state $\mathbf{x} = \begin{bmatrix} p & v & \theta \end{bmatrix}$ consists of position (altitude) p, velocity v and a constant ballistic parameter θ. The zero-mean state noise is characterized by $\mathbb{E}\left[\mathbf{q}(t)\mathbf{q}(s)^\top\right] = \mathbf{Q}\delta(t-s)$, where $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}$. The range measurements are produced at discrete time intervals by a radar positioned at the altitude of $30\,\mathrm{km}$ and $30\,\mathrm{km}$ horizontally to the vertical path of the falling object. Thus the observation model is

$$y(k) = \sqrt{s_x^2 + (s_y - p(k))^2} + r(k), \tag{6.2.19}$$

where $(s_x,\ s_y)$ is the radar position. The measurements were generated with frequency $10\,\mathrm{Hz}$ and the measurement noise is zero-mean with variance $\sigma_y^2 = 9.2903 \times 10^{-4}\,\mathrm{km}^2$. The mean and covariance of the system initial condition were set to

$$\mathbf{x}_0 = \begin{bmatrix} 90\,\mathrm{km} & 6\,\mathrm{km\,s^{-1}} & 1.5 \end{bmatrix} \tag{6.2.20}$$
$$\mathbf{P}_0 = \mathrm{diag}\Big(\begin{bmatrix} 0.0929\,\mathrm{km}^2 & 1.4865\,\mathrm{km}^2\,\mathrm{s}^{-2} & 10^{-4} \end{bmatrix} \Big) \tag{6.2.21}$$

while the filter used different initial state estimate

$$\mathbf{m}_{0|0}^x = \begin{bmatrix} 90\,\mathrm{km} & 6\,\mathrm{km\,s^{-1}} & 1.7 \end{bmatrix} \tag{6.2.22}$$
$$\mathbf{P}_{0|0}^x = \mathrm{diag}\Big(\begin{bmatrix} 0.0929\,\mathrm{km}^2 & 1.4865\,\mathrm{km}^2\,\mathrm{s}^{-2} & 10 \end{bmatrix} \Big) \tag{6.2.23}$$

which implies a lighter object than in reality.

In the experiments, I focused on the comparison of the proposed GPQKF with the UT points and the UKF, because this filter was previously used in [Julier et al., 2000] to demonstrate its superiority over the EKF on the same tracking problem. The parameters of the UKF were set to $\kappa = 0,\ \alpha = 1,\ \beta = 2$, following the recommended heuristics [Särkkä, 2013]. The GPQKF used different kernel parameters for the dynamics, $\alpha_f = 0.5,\ \mathbf{\Lambda}_f = \mathrm{diag}\big(\begin{bmatrix} 10 & 10 & 10 \end{bmatrix} \big)$, and the measurement nonlinearity, $\alpha_h = 0.5,\ \mathbf{\Lambda}_h = \mathrm{diag}\big(\begin{bmatrix} 15 & 20 & 20 \end{bmatrix} \big)$. All filters operated with a discrete-time model obtained by the

Euler approximation with step size $\Delta t = 0.1\,\mathrm{s}$. The discretized model is given by

$$p(k+1) = p(k) - \Delta t\, v(k) + q_1(k), \tag{6.2.24a}$$
$$v(k+1) = v(k) - \Delta t\, v^2(k)\theta(k)\exp(-\gamma p(k)) + q_2(k), \tag{6.2.24b}$$
$$\theta(k+1) = \theta(k) + q_3(k). \tag{6.2.24c}$$

I generated 100 truth trajectories by simulating the continuous-time dynamics, given by the eqs. (6.2.18a) to (6.2.18c), for 30 time steps by 4th-order Runge-Kutta scheme and computed the average RMSE and inclination indicator ν for both tested filters. Figure 6.2.4 shows realizations of the altitude and velocity trajectories along with the average trajectory. Note, that when the object is passing directly in front of the radar at approximately $t = 10\,\mathrm{s}$ (i.e. altitude $30\,\mathrm{km}$), the system is almost unobservable.

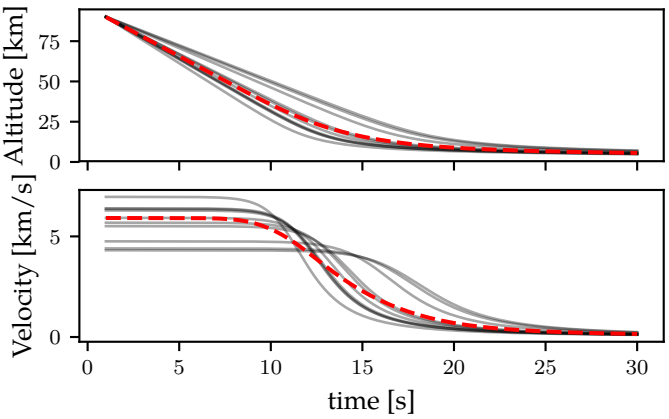

Figure 6.2.4: Altitude and velocity evolution in time. Trajectory realizations (black) and the average trajectory (red). The highest deceleration occurs in the period from 10 to 20 seconds.

Figure 6.2.5 depicts the RMSE for each time step averaged over trajectory simulations. The RMSE of the GPQKF tends to be better for all state vector components. The biggest difference is evident in the RMSE of the ballistic parameter where GPQKF shows significantly better performance during the period of the greatest deceleration. Overall, the UKF shows signs of an unbalanced estimator as evidenced from Figure 6.2.6, where the inclination ν rises significantly above zero, indicating excessive optimism. The GPQKF manages to stay mostly balanced (ν wobbles around zero) with the exception of velocity, where it tips toward pessimism towards the end of the trajectory. This behaviour is mostly likely caused by the inclusion of additional functional uncertainty in the transformed covariance as shown in eqs. (6.1.4d) and (6.1.4e).

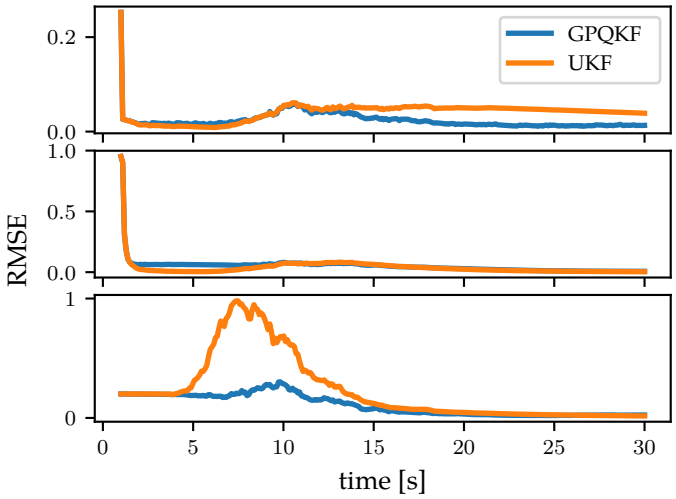

Figure 6.2.5: Evolution of the average RMSE in time for the GPQKF with the UT points and the UKF. From top to bottom: position, velocity and ballistic parameter.

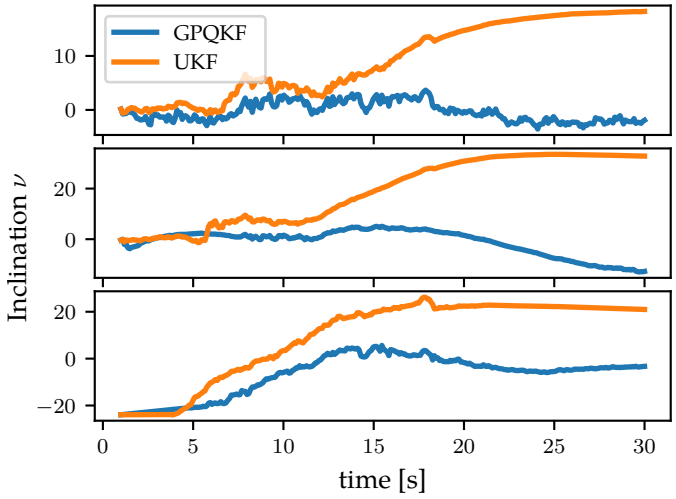

Figure 6.2.6: Evolution of the average inclination in time for the GPQKF with the UT points and the UKF. From top to bottom: position, velocity and ballistic parameter.

6.3 Student's t-Process Quadrature Moment Transform

Motivated by findings in [Shah et al., 2014] where it was concluded that "TP has many if not all of the benefits of GPs, but with increased modeling flexibility at no extra cost.", my aim is to leverage the TP regression model for the design of a Student's t-process quadrature moment transform (TPQ-MT). I propose a moment transformation for moments of the Student's t-distributed random variables, which uses the Student's t-process quadrature. This contribution can be regarded as an extension of the previous

work by Tronarp et al. [2016] with the BQ philosophy.

A common way of parameterizing the multivariate Student's t-density is via the scale matrix $\mathbf{\Sigma}$, which is related to the covariance by $\mathbf{\Sigma} = \frac{\nu}{\nu-2}\mathbf{P}$. Since we are transforming *moments*, I find the alternative parametrization via the covariance matrix more convenient and use it throughout this section. Both parameterizations are reviewed in Appendix A.

Consider again the familiar setup of the moment transformation problem in eqs. (3.2.1) to (3.2.2c). Throughout this section we will assume that

A1: the input variable is Student's t-distributed with $\mathbf{x} \sim \mathrm{St}(\mathbf{m}, \mathbf{P}, \nu)$,

A2: the stochastic decoupling substitution

$$\mathbf{x} = \mathbf{m} + \mathbf{L}\boldsymbol{\xi} \quad \text{where} \quad \mathbf{P} = \mathbf{L}\mathbf{L}^\top \quad \text{and} \quad \boldsymbol{\xi} \sim \mathrm{St}(\mathbf{0}, \mathbf{I}, \nu)$$

in the integrals has taken place, and

A3: the integrand is TP distributed according to $g^e(\boldsymbol{\xi}) \sim \mathrm{TP}\big(0,\, k(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}),\, \nu\big)$ for $e = 1, \ldots, E$.

The last assumption essentially means that a single TP is used to model every output dimension of the integrand.

In order to derive the TPQ transformed moments for $\mathbf{x} \sim \mathrm{St}(\mathbf{m}, \mathbf{P}, \nu)$, I use the familiar stochastic decoupling substitution $\mathbf{x} = \mathbf{m} + \mathbf{L}\boldsymbol{\xi}$, which allows for casting the expectations in terms of a standard t random variable $\boldsymbol{\xi} \sim \mathrm{St}(\mathbf{0}, \mathbf{I}, \nu)$, so that

$$\mathbb{E}_{\mathbf{x}}[\mathbf{g}(\mathbf{x})] = \mathbb{E}_{\boldsymbol{\xi}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})] \approx \mathbb{E}_{\mathbf{g},\boldsymbol{\xi}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})], \tag{6.3.1}$$

$$\mathbb{C}_{\mathbf{x}}[\mathbf{g}(\mathbf{x})] = \mathbb{C}_{\boldsymbol{\xi}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})] \approx \mathbb{C}_{\mathbf{g},\boldsymbol{\xi}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})], \tag{6.3.2}$$

$$\mathbb{C}_{\mathbf{x}}[\mathbf{x}, \mathbf{g}(\mathbf{x})] = \mathbb{C}_{\boldsymbol{\xi}}[\boldsymbol{\xi}, \mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})] \approx \mathbb{C}_{\mathbf{g},\boldsymbol{\xi}}[\boldsymbol{\xi}, \mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})], \tag{6.3.3}$$

where $\mathbf{P} = \mathbf{L}\mathbf{L}^\top$. Relying on the same line of reasoning from Section 6.2, the TPQ transformed moments become

$$\boldsymbol{\mu}_{\mathrm{A}} = \mathbf{Y}^\top \mathbf{K}^{-1} \mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\big], \tag{6.3.4}$$

$$\mathbf{\Pi}_{\mathrm{A}} = \mathbf{Y}^\top \mathbf{K}^{-1} \mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\mathbf{k}(\boldsymbol{\xi})^\top\big] \mathbf{K}^{-1}\mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}}\boldsymbol{\mu}_{\mathrm{A}}^\top + \mathbf{S}, \tag{6.3.5}$$

$$\mathbf{C}_{\mathrm{A}} = \mathbf{L}\mathbb{E}_{\boldsymbol{\xi}}\big[\boldsymbol{\xi}\mathbf{k}(\boldsymbol{\xi})^\top\big] \mathbf{K}^{-1}\mathbf{Y}. \tag{6.3.6}$$

where

$$\mathbf{S} = \mathbb{E}_{\boldsymbol{\xi}}[\mathbb{C}_{\mathbf{g}}[\mathbf{g}(\mathbf{m} + \mathbf{L}\boldsymbol{\xi})]] = \mathrm{diag}\big(\begin{bmatrix} s_{\mathrm{TP}}^1 & \dots & s_{\mathrm{TP}}^E \end{bmatrix} \big), \tag{6.3.7}$$

$$s_{\mathrm{TP}}^e = \gamma_e \Big[\mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi})] - \mathrm{tr}\left(\mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\mathbf{k}(\boldsymbol{\xi})^\top\big] \mathbf{K}^{-1} \right) \Big], \tag{6.3.8}$$

$$\gamma_e = (\nu_g - 2 + \mathbf{y}_e^\top \mathbf{K}^{-1} \mathbf{y}_e)/(\nu_g - 2 + N). \tag{6.3.9}$$

Immediately, we observe that the moments are largely similar to those of the GPQ-MT in eqs. (6.2.2), (6.2.3) and (6.2.5). The only difference is that the expected model variance $\mathbf{S}$ is no longer isotropic, because the TP predictive variance depends on the realizations of the output variable (see eq. (2.3.2b)).

In summary, the general Student's t-process quadrature approximation to the joint distribution of $\mathbf{x} \sim \mathrm{St}(\mathbf{m}, \mathbf{P}, \nu)$ and a transformed random variable $\mathbf{y} = \mathbf{g}(\mathbf{x})$ is given by

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathrm{St}\left(\begin{bmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{A}} \end{bmatrix}, \begin{bmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{A}} \\ \mathbf{C}_{\mathrm{A}}^\top & \boldsymbol{\Pi}_{\mathrm{A}} \end{bmatrix}, \nu \right) \tag{6.3.10}$$

where the transformed moments are computed by the Algorithm 14.

The transform is general in a sense that it can, in principle, operate with any kernel. The RBF kernel in eq. (6.2.7) admits a closed-form evaluation of the expectations in eqs. (6.3.4) to (6.3.6) for a Gaussian distributed input variable. However, for the Student's t-distributed inputs, we have been unable to find any reasonable kernel admitting closed-form solution and thus we resorted to standard Monte Carlo approximations, given by

$$\mathbb{E}_{\boldsymbol{\xi}}[\mathbf{k}(\boldsymbol{\xi})] \approx \frac{1}{I} \sum_{i=1}^{I} \mathbf{k}(\boldsymbol{\xi}_i), \tag{6.3.11}$$

$$\mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{k}(\boldsymbol{\xi})\mathbf{k}(\mathbf{x})(\boldsymbol{\xi})^\top\big] \approx \frac{1}{I} \sum_{i=1}^{I} \mathbf{k}(\boldsymbol{\xi}_i)\mathbf{k}(\boldsymbol{\xi}_i)^\top, \tag{6.3.12}$$

$$\mathbb{E}_{\boldsymbol{\xi}}\big[\boldsymbol{\xi}\mathbf{k}(\boldsymbol{\xi})^\top\big] \approx \frac{1}{I} \sum_{i=1}^{I} \boldsymbol{\xi}_i\mathbf{k}(\boldsymbol{\xi}_i)^\top, \tag{6.3.13}$$

where $\boldsymbol{\xi}_i$ are samples from $\boldsymbol{\xi} \sim \mathrm{St}(\mathbf{0}, \mathbf{I}, \nu)$. Since decoupling is used in the moment integrals, the kernel expectations do not depend on the moments of the input density and only need to be pre-computed once, which significantly reduces computational cost, especially, when the TPQ-MT is utilized in filtering algorithms.

Algorithm 14: Student's t-process quadrature moment transform. The function takes the input mean $\mathbf{m}$ and covariance $\mathbf{P}$ together with the kernel parameters $\boldsymbol{\theta}$, the unit sigma-points $\boldsymbol{\xi}_n$ arranged in columns of a matrix $\boldsymbol{\Xi}$ and a DoF parameter ν_g to produce the transformed moments $\boldsymbol{\mu}_{\mathrm{A}},\ \boldsymbol{\Pi}_{\mathrm{A}},\ \mathbf{C}_{\mathrm{A}}$.

Function `TPQMT(`$\mathbf{m}$, $\mathbf{P}$, $\boldsymbol{\theta}$, $\boldsymbol{\Xi}$, ν_g`)`

  `// Form sigma-points.`

  $\mathbf{L} \leftarrow$ `MatrixFactor(`$\mathbf{P}$`)`

  $\mathbf{X} \leftarrow \mathbf{m} + \mathbf{L}\boldsymbol{\Xi}$

  `// Evaluate nonlinearity at sigma-points.`

  $\mathbf{Y} \leftarrow \mathbf{g}(\mathbf{X})$

  `// Evaluate kernel expectations.`

  $\bar{k} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi}\,; \boldsymbol{\theta})]$

  for $n \leftarrow 1$ **to** N **do**

    $[\mathbf{q}]_n \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

    $[\mathbf{R}]_{*n} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi} k(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

    for $m \leftarrow 1$ **to** N **do**

      $[\mathbf{Q}]_{nm} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta}) k(\boldsymbol{\xi}, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})]$

      $[\mathbf{K}]_{nm} \leftarrow k(\boldsymbol{\xi}_n, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})$

    end

  end

  `// Compute TPQ weights.`

  $\mathbf{w} \leftarrow \mathbf{K}^{-1}\mathbf{q}$

  $\mathbf{W} \leftarrow \mathbf{K}^{-1}\mathbf{Q}\mathbf{K}^{-1}$

  $\mathbf{W}_c \leftarrow \mathbf{R}\mathbf{K}^{-1}$

  `// Compute transformed moments.`

  $\sigma^2 \leftarrow \bar{k} - \mathrm{tr}\left(\mathbf{Q}\mathbf{K}^{-1}\right)$

  for $e = 1$ **to** E **do**

    $[\mathbf{S}]_{ee} \leftarrow \frac{\nu_g - 2 + \mathbf{y}_e^\top \mathbf{K}^{-1} \mathbf{y}_e}{\nu_g - 2 + N} \sigma^2$

  end

  $\boldsymbol{\mu}_{\mathrm{A}} \leftarrow \mathbf{Y}^\top \mathbf{w}$

  $\boldsymbol{\Pi}_{\mathrm{A}} \leftarrow \mathbf{Y}^\top \mathbf{W} \mathbf{Y} - \boldsymbol{\mu}_{\mathrm{A}} \boldsymbol{\mu}_{\mathrm{A}}^\top + \mathbf{S}$

  $\mathbf{C}_{\mathrm{A}} \leftarrow \mathbf{L}\mathbf{W}_c\mathbf{Y}$

  return $\boldsymbol{\mu}_{\mathrm{A}}, \boldsymbol{\Pi}_{\mathrm{A}}, \mathbf{C}_{\mathrm{A}}$

end

6.3.1 Experiments

In this section, I compare the performance of the proposed TPQ-based Student's t-filters and the Student's t-filter introduced by Tronarp et al. [2016], which is based on classical quadrature. The *Student's t-process quadrature Kalman filter* (TPQKF) is obtained by replacing the `MomentTransform()` routine in Algorithm 2 with the TPQ transform `TPQMT()`, in Algorithm 14. Performance is assessed with the help of the familiar RMSE (eq. (6.2.15)) and INC (eq. (6.2.16)) metrics.

Univariate Non-Stationary Growth Model

In the first numerical illustration, I consider the univariate non-stationary growth model (UNGM), which is often used for benchmarking purposes [Gordon et al., 1993; Kitagawa, 1996]. The system is given by the following set of equations

$$x_k = 0.5x_{k-1} + \frac{25x_{k-1}}{1 + x_{k-1}^2} + 8\cos(1.2k) + q_{k-1}, \tag{6.3.14}$$

$$z_k = 0.05x_k^2 + r_k. \tag{6.3.15}$$

The initial conditions were drawn from $x_0 \sim \mathrm{N}(0,\, 1)$. Outliers in the state noise q_k and measurement noise r_k were simulated with Gaussian mixtures, such that

$$q_k \sim 0.8\mathrm{N}\big(0,\, \sigma_q^2\big) + 0.2\mathrm{N}\big(0,\, 10\sigma_q^2\big), \tag{6.3.16}$$

$$r_k \sim 0.8\mathrm{N}\big(0,\, \sigma_r^2\big) + 0.2\mathrm{N}\big(0,\, 100\sigma_r^2\big), \tag{6.3.17}$$

where $\sigma_q^2 = 10$ and $\sigma_r^2 = 0.01$. I simulated 500 trajectories for 250 time steps, which were used for evaluation of the RMSE and INC performance metrics. All tested filters used a state-space model with an initial condition distributed according to $x_0 \sim \mathrm{St}(0,\, 1,\, \nu)$ and the following noise statistics

$$q_k \sim \mathrm{St}\big(0,\, \sigma_q^2,\, \nu\big), \tag{6.3.18}$$

$$r_k \sim \mathrm{St}\big(0,\, \sigma_r^2,\, \nu\big), \tag{6.3.19}$$

where $\nu = 4$.

I compared the RMSE and INC of the proposed TPQSF with the SF [Tronarp et al., 2016], the UKF [Julier and Uhlmann, 2004] and a Student's t-filter using the GPQ MT [Prüher and Šimandl, 2016] (abbreviated GPQSF). The TPQSF and the GPQSF will be collectively referred to as the BQ filters. Student's t-filters used the same 3rd

	RMSE	STD	INC	STD
UKF	8.6924	0.1517	3.0012	0.1539
SF	17.4461	0.6236	51.8733	0.4417
TPQSF($\nu_g = 3$)	7.5683	0.1091	1.5837	0.1561
TPQSF($\nu_g = 4$)	6.8323	0.1044	2.3384	0.1713
TPQSF($\nu_g = 10$)	6.1423	0.0154	5.6910	0.0324
TPQSF($\nu_g = 100$)	7.4399	0.1550	12.5120	0.1676
TPQSF($\nu_g = 500$)	7.5709	0.1546	13.2104	0.1623
GPQSF	7.6766	0.1554	13.5926	0.1595

Table 6.3.1: Performance of TPQSF compared in terms of average RMSE and INC. Standard deviations of the criteria were estimated by bootstrapping. For increasing DOF parameter ν_g of the TP regression model the performance approaches that of the GPQSF.

degree fully symmetric sigma-point set with $\kappa = 0$ and the filter DoF fixed at $\nu = 4$. The kernel parameters for all BQ filters were set to $\boldsymbol{\theta}_f = \begin{bmatrix} 3 & 1 \end{bmatrix}$ and $\boldsymbol{\theta}_h = \begin{bmatrix} 3 & 3 \end{bmatrix}$. Table 6.3.1 reports MC simulation averages of both metrics along with bootstrapped variances [Wasserman, 2007] of the averages (using $10\,000$ samples). It is evident that the TPQSFs can outperform all the classical filters (UKF, SF) as well as the GPQSF in terms of both metrics. The values of INC, being closer to zero, indicate increased estimate credibility. For increasing DoF of the Student's t-process model, we observe the performance of TPQSFs approaching that of GPQSF, which is an expected behavior, since TPQSF with $\nu_g = \infty$ is equivalent to GPQSF.

Radar Tracking with Glint Noise

As a second illustration, I consider tracking of a moving object where the range and bearing measurements are corrupted with glint noise. I adopted the example from Arasaratnam et al. [2007], where the tracking scenario is described by the following state-space model

$$\mathbf{x}_k = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} \tau^2/2 & 0 \\ \tau & 0 \\ 0 & \tau^2/2 \\ 0 & \tau \end{bmatrix} \mathbf{q}_{k-1}, \tag{6.3.20}$$

$$\mathbf{z}_k = \begin{bmatrix} \sqrt{x_k^2 + y_k^2} \\ \operatorname{atan2}(y_k, x_k) \end{bmatrix} + \mathbf{r}_k, \tag{6.3.21}$$

with the system state being defined as $\mathbf{x}_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k \end{bmatrix}$. The state components x_k and y_k are the Cartesian coordinates of the moving object and the pair $(\dot{x}_k,\ \dot{y}_k)$ stands for the velocity in the respective directions. During simulations, the discretization interval was $\tau = 0.5\,\mathrm{s}$, the initial state was drawn from $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{m}_0^x,\ \mathbf{P}_0^x)$ with

$$\mathbf{m}_0 = \begin{bmatrix} 10\,000\,\mathrm{m} & 300\,\mathrm{ms}^{-1} & 1000\,\mathrm{m} & -40\,\mathrm{ms}^{-1} \end{bmatrix}, \tag{6.3.22}$$

$$\mathbf{P}_0 = \mathrm{diag}\big(\begin{bmatrix} 10\,000\,\mathrm{m}^2 & 100\,\mathrm{m}^2\mathrm{s}^{-2} & 10\,000\,\mathrm{m}^2 & 100\,\mathrm{m}^2\mathrm{s}^{-2} \end{bmatrix} \big). \tag{6.3.23}$$

The state noise is Gaussian distributed, such that $\mathbf{q}_k \sim \mathrm{N}(\mathbf{0},\ \mathbf{Q})$ with covariance $\mathbf{Q} = \mathrm{diag}\big(\begin{bmatrix} 50\,\mathrm{m}^2\mathrm{s}^{-4} & 5\,\mathrm{m}^2\mathrm{s}^{-4} \end{bmatrix} \big)$. The glint noise in the measurements is modeled by a Gaussian mixture

$$\mathbf{r}_k \sim (1-\beta)\mathrm{N}(\mathbf{0},\ \mathbf{R}_1) + \beta\mathrm{N}(\mathbf{0},\ \mathbf{R}_2) \tag{6.3.24}$$

with $\mathbf{R}_1 = \mathrm{diag}\big(\begin{bmatrix} 50\,\mathrm{m}^2 & 0.4\,\mathrm{mrad}^2 \end{bmatrix} \big)$ and $\mathbf{R}_2 = \mathrm{diag}\big(\begin{bmatrix} 5000\,\mathrm{m}^2 & 16\,\mathrm{mrad}^2 \end{bmatrix} \big)$, where β is the glint noise probability.

As in the UNGM experiment, I compared the performance of the proposed TPQSF with the SF, the standard UKF and the GPQSF. The UKF used $\kappa = 0$, following the usual heuristic recommendation. For the TPQSF, I considered two settings of the TP model DoF parameter, $\nu_g = 2.2$ and $\nu_g = 4$. All of the Student's t-filters assumed that the initial state, the state noise and the measurement noise were characterized by the Student's t-distribution, such that

$$\mathbf{x}_0 \sim \mathrm{St}\big(\mathbf{m}_{0|0}^x,\ \mathbf{P}_{0|0}^x,\ \nu^x\big), \tag{6.3.25}$$

$$\mathbf{q}_k \sim \mathrm{St}\big(\mathbf{0},\ \mathbf{Q},\ \nu^q\big), \tag{6.3.26}$$

$$\mathbf{r}_k \sim \mathrm{St}\big(\mathbf{0},\ \mathbf{R},\ \nu^r\big) \tag{6.3.27}$$

where the initial filtered mean and covariance were

$$\mathbf{m}_{0|0}^x = \begin{bmatrix} 10\,175\,\mathrm{m} & 295\,\mathrm{ms}^{-1} & 980\,\mathrm{m} & -35\,\mathrm{ms}^{-1} \end{bmatrix}, \tag{6.3.28}$$

$$\mathbf{P}_{0|0}^x = \mathbf{P}_0, \tag{6.3.29}$$

with the DoF parameters $\nu^x = 1000$, $\nu^q = 1000$ and $\nu^r = 4.0$. The kernel parameters for the BQ filters were set to $\boldsymbol{\theta}_f = \begin{bmatrix} 1 & 100 & 100 & 100 & 100 \end{bmatrix}$ for the dynamics model and $\boldsymbol{\theta}_h = \begin{bmatrix} 0.05 & 10 & 100 & 10 & 100 \end{bmatrix}$ for the measurement model.

The filter performance was evaluated by simulating 1000 trajectories, each 100 time steps long, and computing the Monte Carlo averages of the performance scores. Figure 6.3.1 shows box-plots of the time-averaged RMSE scores. The left pane shows

that the UKF and SF have more extreme outliers than the proposed TPQSF, while in the right pane we see that the classical SF is better in terms of median RMSE. It is worth noting that because TPQ-based filters have a tunable DoF parameter, they were able to achieve improved median RMSE over the GPQ-based filter. From Figure 6.3.2, showing

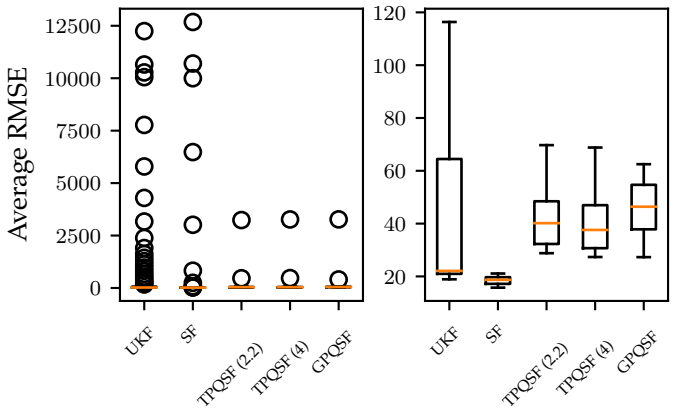

Figure 6.3.1: Overall filter RMSEs shown with outliers (left) and a detail without outliers (right). The proposed TPQ-based filters have less extreme outliers, whereas the median RMSE favors the classical quadrature-based SF.

the time-averaged INC score, we can deduce that the BQ filters provide more balanced estimates on average, whereas the classical filters are excessively optimistic in their estimates. This behavior is in accordance with the expectations, because the BQ filters account for the additional functional uncertainty as described in Section 6.1. Table 6.3.2

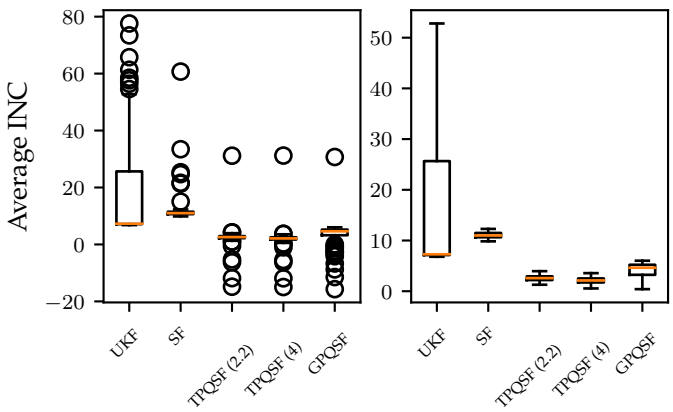

Figure 6.3.2: Overall filter INCs shown with (left) and without (right) outliers. The proposed TPQ-based filters display improved INC with most outliers in the pessimistic direction, whereas the UKF and SF are excessively optimistic.

shows the mean of the overall average RMSE and INC along with the their standard

deviations, which were estimated by bootstrapping with $10\,000$ samples. Evidently, TPQSFs drastically improve the mean of the overall average RMSE and, as mentioned previously, provide much more balanced state estimates.

	RMSE	STD	INC	STD
UKF	803.99	231.62	18.37	1.80
SF	457.49	200.88	12.22	0.58
TPQSF($\nu_g = 2.2$)	77.29	32.17	2.39	0.38
TPQSF($\nu_g = 4$)	75.54	31.95	1.92	0.39
GPQSF	81.04	32.17	3.52	0.45

Table 6.3.2: Overall RMSEs for the radar tracking example. The average RMSE favours the TPQ-based filters. Our proposed filters also give more balanced state estimates on average, as shown by the inclination indicator (INC) being closer to zero.

6.4 Conclusion

In this chapter, I have shown how a Bayesian view of quadrature can be leveraged for the design of general purpose moment transform, which meets the **Goal 2** from Chapter 4. Unlike the classical transforms, the proposed GPQ and TPQ-MTs are able to acknowledge the limited extent of knowledge of the integrated function when evaluated at finite number of evaluation points and thus they can account for the integration error incurred in computing the mean, by inflating the transformed covariance. The underlying models in the proposed transforms are non-parametric, which brings a number of advantages. Namely, the transform is not restricted by polynomial assumptions on the integrand (unlike the classical methods) and it quantifies predictive uncertainty, which eventually translates into integral uncertainty. Should a situation arise when we do not trust the function evaluations for any reason, the models have the capability to account for noise in function evaluations. The proposed moment transforms are entirely general, in that the equations hold for any kernel and input density; however, analytically tractable kernel-density pairs are preferable. I showed that the transform may outperform classical transforms on a coordinate conversion and nonlinear sigma-point filtering examples. In all experiments, the filters based on the BQ give more realistic estimates of the covariance, hence are better at self-assessing their estimation error.

Chapter 7

GPQ Moment Transforms with Derivative Observations

The quadratures seen up to this point relied only on the function values. When the integrand is differentiable, it might be beneficial to consider incorporating the derivative evaluations into the quadrature rule for improving the integral approximation.

This chapter proposes the use of function derivatives as a way of decreasing the GPQ integral variance. While the use of derivatives in BQ is not a completely new idea, their use has not yet been systematically analyzed in the present literature. I design a general GPQ moment transformation which uses gradients as additional source of information about the integrand and also reveal connections between the proposed transform and the linearization transform in Algorithm 3, which is a center piece of the well-known EKF. Contributions in this chapter can be found in the following author's publication:

- J. Prüher and S. Särkkä. On the Use of Gradient Information in Gaussian Process Quadratures. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Sept 2016. doi: 10.1109/MLSP.2016.7738903

7.1 Gradient Observations in GP Quadrature

The use of derivative observations in GP regression is a special case of using linear operator observations, which has been previously discussed, for example, in [Murray-Smith and Pearlmutter, 2005; Särkkä, 2011]. I briefly review the problem setup and state the main result, which is later applied to incorporate derivatives into GPQ.

First, let's consider the GP regression problem on a scalar function $g : \mathbb{R}^D \to \mathbb{R}$,

where, in addition to observing function values $g(\mathbf{x}_n)$ for every input $\mathbf{x}_n$, we also observe gradients $\boldsymbol{\nabla} g(\mathbf{x}_n)$. We could look at it as having two observation models, so that

$$y_n = g(\mathbf{x}_n) + \varepsilon, \qquad \varepsilon \sim \mathrm{N}\big(\mathbf{0},\, \sigma^2\big), \tag{7.1.1}$$

$$\boldsymbol{\delta}_n = \mathbf{d}(\mathbf{x}_n) + \boldsymbol{\varepsilon}_d, \qquad \boldsymbol{\varepsilon}_d \sim \mathrm{N}\big(\mathbf{0},\, \sigma_d^2 \mathbf{I}\big), \tag{7.1.2}$$

where $\mathbf{d} : \mathbb{R}^D \to \mathbb{R}^D$ is a gradient of function $g(\mathbf{x})$ defined as

$$\mathbf{d}(\mathbf{x}) \coloneqq \boldsymbol{\nabla} g(\mathbf{x}) = \begin{bmatrix} \frac{\partial g}{\partial x_1} & \cdots & \frac{\partial g}{\partial x_D} \end{bmatrix}^\top. \tag{7.1.3}$$

Since gradient is a linear operator acting on a GP distributed function g, the $\mathbf{d}$ is also a GP. This fact is an infinite dimensional analogue of the affine invariance property of Gaussian random variables. With gradient observations incorporated, our dataset is now $\mathcal{D} = \{(\mathbf{x}_n,\, y_n,\, \boldsymbol{\delta}_n)\}_{n=1}^N$.

In order to distinguish covariances, the following notation is adopted

$$k_{gg}(\mathbf{x}, \mathbf{x}') = \mathbb{C}\big[g(\mathbf{x}), g(\mathbf{x}')\big], \tag{7.1.4}$$

$$k_{gd}(\mathbf{x}, \mathbf{x}') = \mathbb{C}\big[g(\mathbf{x}), \mathbf{d}(\mathbf{x}')\big], \tag{7.1.5}$$

$$k_{dd}(\mathbf{x}, \mathbf{x}') = \mathbb{C}\big[\mathbf{d}(\mathbf{x}), \mathbf{d}(\mathbf{x}')\big]. \tag{7.1.6}$$

Assuming the observations are exact (zero noise), the GP predictive mean and variance are given by

$$\mathbb{E}_g[g(\mathbf{x}) \mid \mathcal{D}] = \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1} \mathbf{y}, \tag{7.1.7}$$

$$\mathbb{V}_g[g(\mathbf{x}) \mid \mathcal{D}] = k_{gg}(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}), \tag{7.1.8}$$

where all observations are arranged in a $(N + ND \times 1)$ vector

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_g \\ \mathbf{y}_d \end{bmatrix}, \quad \text{where} \quad \mathbf{y}_g = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \text{and} \quad \mathbf{y}_d = \begin{bmatrix} \boldsymbol{\delta}_1 \\ \vdots \\ \boldsymbol{\delta}_N \end{bmatrix}. \tag{7.1.9}$$

The layout of the block matrices is

$$\mathbf{k}(\mathbf{x}) = \begin{bmatrix} \mathbf{k}_{gg}(\mathbf{x}) \\ \mathbf{k}_{gd}(\mathbf{x}) \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{gg} & \mathbf{K}_{gd}^\top \\ \mathbf{K}_{gd} & \mathbf{K}_{dd} \end{bmatrix}, \tag{7.1.10}$$

where

$$\mathbf{k}_{gd}(\mathbf{x}) = \sum_{n=1}^{N} \mathbf{e}_n \otimes k_{gd}(\mathbf{x}, \mathbf{x}_n), \tag{7.1.11}$$

$$\mathbf{K}_{gd} = \sum_{m,n=1}^{N} \mathbf{e}_m \mathbf{e}_n^\top \otimes k_{gd}(\mathbf{x}_m, \mathbf{x}_n), \tag{7.1.12}$$

$$\mathbf{K}_{dd} = \sum_{m,n=1}^{N} \mathbf{e}_m \mathbf{e}_n^\top \otimes k_{dd}(\mathbf{x}_m, \mathbf{x}_n), \tag{7.1.13}$$

where $\otimes$ denotes Kronecker product. Since

$$k_{gd}(\mathbf{x}, \mathbf{x}_n) \coloneqq \frac{\partial}{\partial \mathbf{x}'} k_{gg}\big(\mathbf{x}, \mathbf{x}'\big)\Big|_{\mathbf{x}'=\mathbf{x}_n}, \tag{7.1.14}$$

$$k_{dd}(\mathbf{x}_m, \mathbf{x}_n) \coloneqq \frac{\partial^2}{\partial \mathbf{x} \partial \mathbf{x}'} k_{gg}\big(\mathbf{x}, \mathbf{x}'\big)\Big|_{\mathbf{x}=\mathbf{x}_m, \mathbf{x}'=\mathbf{x}_n} \tag{7.1.15}$$

are a $(D \times 1)$ vector and $(D \times D)$ matrix respectively, the $\mathbf{K}_{gd}$ is $(DN \times N)$ block matrix and $\mathbf{K}_{dd}$ is $(DN \times DN)$ block matrix. The Figure 7.1.1b shows the reduced predictive variance of GP regression fit when gradient observations are used. The same kernel parameters are used in both cases.

Finally, with the augmented expressions (7.1.7) and (7.1.8) in hand, incorporating derivative information into the GPQ is very straightforward. Plugging eqs. (7.1.7) and (7.1.8) into eqs. (3.3.2) and (3.3.3), the expressions for first two moments of the integral are obtained

$$\mathbb{E}_g[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \mathbb{E}_{\mathbf{x}}\big[\mathbf{k}(\mathbf{x})\big]^\top \mathbf{K}^{-1}\mathbf{y} = \mathbf{q}^\top \mathbf{K}^{-1}\mathbf{y}, \tag{7.1.16}$$

$$\mathbb{V}_g[\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]] = \bar{k} - \mathbf{q}^\top \mathbf{K}^{-1}\mathbf{q}. \tag{7.1.17}$$

Since $\mathbf{y}$ contains gradient observations as well, the quadrature weights are stacked in the following layout

$$\mathbf{w} = \begin{bmatrix} w_1 & \dots & w_N & (\mathbf{w}_1^d)^\top & \dots & (\mathbf{w}_N^d)^\top \end{bmatrix} \tag{7.1.18}$$

where $\mathbf{w}_N^d \in \mathbb{R}^D$. This means, that eq. (7.1.16) is a quadrature rule of the form

$$\mathbb{Q}[g] = \sum_{n=1}^{N} w_n g(\mathbf{x}_n) + \sum_{n=1}^{N} (\mathbf{w}_n^d)^\top \mathbf{d}(\mathbf{x}_n). \tag{7.1.19}$$

The Figure 7.1.1c illustrates the reduction in integral variance when the additional gradient observations are included. Both the GPQ and GPQ with gradient observations (GPQ+D) produce distributions centered on the true value of the integral.

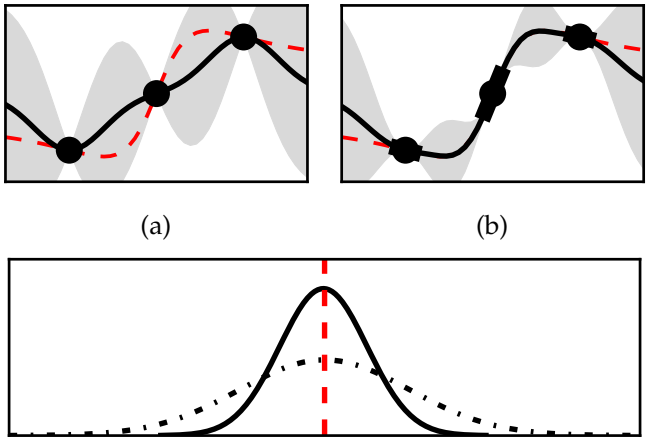

Figure 7.1.1: Approximation of the true function (dashed) with the GP mean function (solid). (a) Approximation using function observations (dots) only. (b) Approximation using function values and gradient observations (line segments). (c) Densities over the value of the integral. The integral variance of GPQ+D (solid) is visibly smaller than that of GPQ (dash dot). Both densities concentrate near the true value of the integral (dashed).

7.2 GPQ Transforms with Gradients

Mathematical derivation of the GPQ+D moment transform follows the same template as the previous BQ transforms in Sections 6.2 and 6.3. I take the expressions for GPQ+D integral mean and variance from eqs. (7.1.16) and (7.1.17), and substitute them into the general expressions for the BQ transformed moments in eqs. (6.1.4b), (6.1.4e) and (6.1.4g).

The only remaining issue is the extension of results from the previous section to the case of vector functions $\mathbf{g} : \mathbb{R}^D \to \mathbb{R}^E$. This is easily accomplished by extending the matrix $\mathbf{y}$ with additional columns, each of which contains all observations of one

particular output of $\mathbf{g}(\mathbf{x})$. For clarity, layout of the observation matrix is as follows

$$\mathbf{Y} = \begin{bmatrix} y_1^1 & \dots & y_1^e & \dots & y_1^E \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_n^1 & \dots & y_n^e & \dots & y_n^E \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_N^1 & \dots & y_N^e & \dots & y_N^E \\ \boldsymbol{\delta}_1^1 & \dots & \boldsymbol{\delta}_1^e & \dots & \boldsymbol{\delta}_1^E \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{\delta}_n^1 & \dots & \boldsymbol{\delta}_n^e & \dots & \boldsymbol{\delta}_n^E \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{\delta}_N^1 & \dots & \boldsymbol{\delta}_N^e & \dots & \boldsymbol{\delta}_N^E \end{bmatrix}, \tag{7.2.1}$$

where n and e index sigma-points and outputs, respectively.

To summarize, the proposed Gaussian approximation of the joint distribution of $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ and the transformed random variable $\mathbf{y} = \mathbf{g}(\mathbf{x})$ based on the GPQ with derivative observations (GPQ+D), is given by

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \mathbf{m} \\ \boldsymbol{\mu}_\mathrm{A} \end{bmatrix}, \begin{bmatrix} \mathbf{P} & \mathbf{C}_\mathrm{A} \\ \mathbf{C}_\mathrm{A}^\top & \boldsymbol{\Pi}_\mathrm{A} \end{bmatrix} \right), \tag{7.2.2}$$

where the transformed moments are computed by the Algorithm 15.

7.2.1 Connection to Linearization Transform

General GPQ+D moment transform is formulated for *arbitrary* kernel and sigma-point sets. Below, I give proofs that GPQ+D reduces to linearized transform for two particular choices of the kernel. In the derivations, I use the centered variant of the GPQ+D transform, which uses a substitution $\mathbf{x} = \mathbf{m} + \boldsymbol{\eta},\ \boldsymbol{\eta} \sim \mathrm{N}(\mathbf{0}, \mathbf{P})$ in the Gaussian integrals in eqs. (3.2.2a) to (3.2.2c). Thus GP models the function $\tilde{\mathbf{g}}(\boldsymbol{\eta}) = \mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{m} + \boldsymbol{\eta})$, so that $\tilde{\mathbf{g}}(\boldsymbol{\eta}) \sim \mathrm{GP}(0,\, k(\boldsymbol{\eta}, \boldsymbol{\eta}'))$. Expressions for the mean and covariance of the centered GPQ+D, on the lines 30 and 31 of Algorithm 15 respectively, remain formally the same, except for the input-output covariance, which becomes

$$\mathbf{C}_\mathrm{A} = \mathbf{W}_c \mathbf{Y}. \tag{7.2.3}$$

In order for the GPQ+D to be equivalent to the linearization transform, clearly, the expressions for the mean and covariance on the lines 30 and 31 of the Algorithm 15 and

Algorithm 15: Gaussian process quadrature moment transform with additional derivative observations of the integrand. The function takes the input mean $\mathbf{m}$ and covariance $\mathbf{P}$ together with the kernel parameters $\boldsymbol{\theta}$ and unit sigma-points $\boldsymbol{\xi}_n$ arranged in columns of a matrix $\boldsymbol{\Xi}$ to produce the transformed moments $\boldsymbol{\mu}_\mathrm{A},\ \boldsymbol{\Pi}_\mathrm{A},\ \mathbf{C}_\mathrm{A}$.

Function `GPQDMT(`$\mathbf{m}$, $\mathbf{P}$, $\boldsymbol{\theta}$, $\boldsymbol{\Xi}$`)`

`// Form sigma-points.`

$\mathbf{L} \leftarrow$ `MatrixFactor(`$\mathbf{P}$`)`

$\mathbf{X} \leftarrow \mathbf{m} + \mathbf{L}\boldsymbol{\Xi}$

`// Evaluate nonlinearity and its gradient at sigma-points.`

$\mathbf{Y}_g \leftarrow \mathbf{g}(\mathbf{X})$

$\mathbf{Y}_d \leftarrow \boldsymbol{\nabla}\mathbf{g}(\mathbf{X})$

$\mathbf{Y} \leftarrow \begin{bmatrix} \mathbf{Y}_g \\ \mathbf{Y}_d \end{bmatrix}$

`// Evaluate kernel expectations.`

$\bar{k} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k_{gg}(\boldsymbol{\xi}, \boldsymbol{\xi}\,; \boldsymbol{\theta})]$

for $n \leftarrow 1$ **to** N **do**

$\left[\mathbf{q}_{gg}\right]_n \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k_{gg}(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

$[\mathbf{R}_g]_{*n} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi} k_{gg}(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

$\mathbf{K}_{gd} \leftarrow \mathbf{K}_{gd} + \mathbf{e}_n \otimes k_{gd}(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})$

$\mathbf{q}_{gd} \leftarrow \mathbf{q}_{gd} + \mathbf{e}_n \otimes \mathbb{E}_{\boldsymbol{\xi}}[k_{gd}(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

$\mathbf{R}_d \leftarrow \mathbf{R}_d + \mathbf{e}_n \otimes \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi} k_{gd}(\boldsymbol{\xi}, \boldsymbol{\xi}_n\,; \boldsymbol{\theta})]$

for $m \leftarrow 1$ **to** N **do**

$[\mathbf{K}_{gg}]_{nm} \leftarrow k_{gg}(\boldsymbol{\xi}_n, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})$

$\left[\mathbf{Q}_{gg}\right]_{nm} \leftarrow \mathbb{E}_{\boldsymbol{\xi}}[k_{gg}(\boldsymbol{\xi}, \boldsymbol{\xi}_m\,; \boldsymbol{\theta}) k_{gg}(\boldsymbol{\xi}, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})]$

$\mathbf{K}_{dd} \leftarrow \mathbf{K}_{dd} + \mathbf{e}_n \mathbf{e}_m^\top \otimes k_{dd}(\boldsymbol{\xi}_n, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})$

$\mathbf{Q}_{gd} \leftarrow \mathbf{Q}_{gd} + \mathbf{e}_n \mathbf{e}_m^\top \otimes \mathbb{E}_{\boldsymbol{\xi}}[k_{gg}(\boldsymbol{\xi}_n, \boldsymbol{\xi}\,; \boldsymbol{\theta}) k_{gd}(\boldsymbol{\xi}, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})]$

$\mathbf{Q}_{dd} \leftarrow \mathbf{Q}_{dd} + \mathbf{e}_n \mathbf{e}_m^\top \otimes \mathbb{E}_{\boldsymbol{\xi}}[k_{dg}(\boldsymbol{\xi}_n, \boldsymbol{\xi}\,; \boldsymbol{\theta}) k_{gd}(\boldsymbol{\xi}, \boldsymbol{\xi}_m\,; \boldsymbol{\theta})]$

end

end

$\mathbf{K} \leftarrow \begin{bmatrix} \mathbf{K}_{gg} & \mathbf{K}_{gd}^\top \\ \mathbf{K}_{gd} & \mathbf{K}_{dd} \end{bmatrix}$

$\mathbf{q} \leftarrow \begin{bmatrix} \mathbf{q}_{gg} \\ \mathbf{q}_{gd} \end{bmatrix}$

$\mathbf{Q} \leftarrow \begin{bmatrix} \mathbf{Q}_{gg} & \mathbf{Q}_{gd}^\top \\ \mathbf{Q}_{gd} & \mathbf{Q}_{dd} \end{bmatrix}$

$\mathbf{R} \leftarrow \begin{bmatrix} \mathbf{R}_g & \mathbf{R}_d \end{bmatrix}$

`// Compute GPQ+D weights.`

$\mathbf{w} \leftarrow \mathbf{K}^{-1}\mathbf{q}$

$\mathbf{W} \leftarrow \mathbf{K}^{-1}\mathbf{Q}\mathbf{K}^{-1}$

$\mathbf{W}_c \leftarrow \mathbf{R}\mathbf{K}^{-1}$

`// Compute transformed moments.`

$\sigma^2 \leftarrow \bar{k} - \mathrm{tr}\left(\mathbf{Q}\mathbf{K}^{-1}\right)$

$\boldsymbol{\mu}_\mathrm{A} \leftarrow \mathbf{Y}^\top \mathbf{w}$

$\boldsymbol{\Pi}_\mathrm{A} \leftarrow \mathbf{Y}^\top \mathbf{W}\mathbf{Y} - \boldsymbol{\mu}_\mathrm{A}\boldsymbol{\mu}_\mathrm{A}^\top + \sigma^2\mathbf{I}$

$\mathbf{C}_\mathrm{A} \leftarrow \mathbf{L}\mathbf{W}_c\mathbf{Y}$

return $\boldsymbol{\mu}_\mathrm{A}, \boldsymbol{\Pi}_\mathrm{A}, \mathbf{C}_\mathrm{A}$

end

the cross-covariance in eq. (7.2.3) should be equal to eqs. (3.2.5a) to (3.2.5c) respectively. This can be achieved when GPQ+D uses single sigma-point $\mathbf{x}_0 = \mathbf{m}$ ($\boldsymbol{\eta}_0 = \mathbf{0}$), in which case $\mathbf{Y} = \begin{bmatrix}\mathbf{g}(\mathbf{m}) & \mathbf{G}(\mathbf{m})\end{bmatrix}^\top$ and the weights are given by

$$\mathbf{w} = \begin{bmatrix}1\\ \mathbf{0}\end{bmatrix},\ \mathbf{W} = \begin{bmatrix}1 & \mathbf{0}^\top\\ \mathbf{0} & \mathbf{P}\end{bmatrix},\ \mathbf{W}_c = \begin{bmatrix}0 & \mathbf{0}^\top\\ \mathbf{0} & \mathbf{P}\end{bmatrix}. \tag{7.2.4}$$

Additionally, $\sigma^2 = 0$ must hold.

The linearized transform uses derivative at one point to approximate the nonlinearity with a tangent line. The same can be achieved with GP regression using affine kernel and gradient observations.

Theorem 6 (GPQ+D with affine kernel)**.** *Let the kernel be given as*

$$k_{gg}(\boldsymbol{\eta}, \boldsymbol{\eta}') = \sigma_0^2 + \boldsymbol{\eta}^\top \boldsymbol{\Sigma} \boldsymbol{\eta}'$$

with $\boldsymbol{\Sigma} = \operatorname{diag}\left(\begin{bmatrix}\sigma_1^2 & \dots & \sigma_D^2\end{bmatrix}\right)$*. Assume that one sigma-point* $\boldsymbol{\eta}_0 = \mathbf{0}$ *is used. Then the general GPQ+D transform reduces to the linearized transform and the variance of the mean integral is zero.*

Proof. The kernel derivatives are

$$k_{gd}(\boldsymbol{\eta}, \boldsymbol{\eta}') = \boldsymbol{\Sigma}\boldsymbol{\eta}, \tag{7.2.5}$$

$$k_{dd}(\boldsymbol{\eta}, \boldsymbol{\eta}') = \boldsymbol{\Sigma}. \tag{7.2.6}$$

After evaluating the kernel at $\boldsymbol{\eta}_0$ and its expectations, we find that

$$\mathbf{q} = \begin{bmatrix}\sigma_0^2\\ \mathbf{0}\end{bmatrix},\ \mathbf{K}^{-1} = \begin{bmatrix}\sigma_0^{-2} & \mathbf{0}^\top\\ \mathbf{0} & \boldsymbol{\Sigma}^{-1}\end{bmatrix},\ \mathbf{Q} = \begin{bmatrix}\sigma_0^4 & \mathbf{0}^\top\\ \mathbf{0} & \boldsymbol{\Sigma}\mathbf{P}\boldsymbol{\Sigma}\end{bmatrix} \tag{7.2.7}$$

plugging into the expressions for weight matrices $\mathbf{w},\ \mathbf{W}$ and $\mathbf{W}_c$ from Algorithm 15, we obtain the linear transform weights given by eq. (7.2.4). Expected GP variance reduces to $\sigma^2 = \sigma_0^2 + \operatorname{tr}(\mathbf{P}\boldsymbol{\Sigma}) - (\sigma_0^2 + \operatorname{tr}(\mathbf{P}\boldsymbol{\Sigma})) = 0$. Plugging $\mathbf{q}$ and $\mathbf{K}^{-1}$ into eq. (7.1.17), it is easily verified that the variance of the mean integral is zero. □

In a similar fashion, we can use the radial basis function (RBF) kernel with infinite lengthscale to obtain linearized transform.

Theorem 7 (GPQ+D with RBF kernel and $\lambda \to \infty$)**.** *Let*

$$k_{gg}(\boldsymbol{\eta}, \boldsymbol{\eta}') = \alpha^2 \exp\left(-\tfrac{1}{2}(\boldsymbol{\eta} - \boldsymbol{\eta}')^\top \boldsymbol{\Lambda}^{-1} (\boldsymbol{\eta} - \boldsymbol{\eta}')\right)$$

where $\boldsymbol{\Lambda} = \lambda^2\mathbf{I}$ and λ is the lengthscale parameter. Assume that one sigma-point $\boldsymbol{\eta}_0 = \mathbf{0}$ is used. Then the GPQ+D transform reduces to the linear transform for $\lambda \to \infty$ and variance of the mean integral is $\alpha^2 - 1$.

Proof. The kernel derivatives are

$$k_{gd}(\boldsymbol{\eta}, \boldsymbol{\eta}') = \boldsymbol{\Lambda}^{-1}(\boldsymbol{\eta} - \boldsymbol{\eta}')k_{gg}(\boldsymbol{\eta}, \boldsymbol{\eta}') \tag{7.2.8}$$

$$k_{dd}(\boldsymbol{\eta}, \boldsymbol{\eta}') = \left[\mathbf{I} - \boldsymbol{\Lambda}^{-1}(\boldsymbol{\eta} - \boldsymbol{\eta}')(\boldsymbol{\eta} - \boldsymbol{\eta}')^\top\right]\boldsymbol{\Lambda}^{-1}k_{gg}(\boldsymbol{\eta}, \boldsymbol{\eta}') \tag{7.2.9}$$

After computing all the necessary kernel expectations on the lines 7 to 19 of Algorithm 15, we find that

$$\mathbf{q}^\top = \begin{bmatrix}\alpha^2\left|\mathbf{P}\boldsymbol{\Lambda}^{-1} + \mathbf{I}\right|^{-1/2} & \mathbf{0}\end{bmatrix} \tag{7.2.10}$$

$$\mathbf{K}^{-1} = \begin{bmatrix}\alpha^{-2} & \mathbf{0}^\top \\ \mathbf{0} & \alpha^{-2}\boldsymbol{\Lambda}\end{bmatrix} \tag{7.2.11}$$

$$\mathbf{Q} = \begin{bmatrix}b & \mathbf{0}^\top \\ \mathbf{0} & b\boldsymbol{\Lambda}^{-1}(2\boldsymbol{\Lambda}^{-1} + \mathbf{P}^{-1})^{-1}\boldsymbol{\Lambda}^{-1}\end{bmatrix} \tag{7.2.12}$$

and $b = \alpha^4\left|2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\right|^{-1/2}$. The weights are

$$\mathbf{w} = \left|\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\right|^{-1/2}\begin{bmatrix}1 & \mathbf{0}^\top\end{bmatrix}^\top, \tag{7.2.13}$$

$$\mathbf{W} = \left|2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\right|^{-1/2}\begin{bmatrix}1 & \mathbf{0}^\top \\ \mathbf{0} & (2\boldsymbol{\Lambda}^{-1} + \mathbf{P}^{-1})^{-1}\end{bmatrix}, \tag{7.2.14}$$

$$\mathbf{W}_c = \left|\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\right|^{-1/2}\begin{bmatrix}0 & \mathbf{0}^\top \\ \mathbf{0} & (\boldsymbol{\Lambda}^{-1} + \mathbf{P}^{-1})^{-1}\end{bmatrix}. \tag{7.2.15}$$

Taking a limit for $\lambda \to \infty$, we obtain the linear transform weights in eq. (7.2.4). Expected GP variance is $\sigma^2 = \alpha^2 - b\left[\alpha^{-2} + \operatorname{tr}\left((2\boldsymbol{\Lambda}^{-1} + \mathbf{P}^{-1})^{-1}\boldsymbol{\Lambda}^{-1}\right)\right]$ for which $\lim_{\lambda\to\infty}\sigma^2 = 0$. Plugging in $\mathbf{q}$ and $\mathbf{K}^{-1}$ into eq. (7.1.17), the variance of the mean integral becomes

$$\mathbb{V}_g[\mathbb{E}_{\boldsymbol{\eta}}[g(\boldsymbol{\eta})]] = \alpha^2\left|2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\right|^{-1/2} - \left|\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\right|^{-1},$$

which, for $\lambda \to \infty$, approaches $\alpha^2 - 1$. □

7.3 Numerical Experiments

In all numerical experiments, I tested decoupled variants of GPQ and GPQ+D moment transforms. Both transforms operate with the RBF kernel and use spherical-radial unit sigma-points. The kernel parameters λ and α were set to good heuristic values for each experiment independently. Both transforms were compared against the spherical-radial transform (SR) [Arasaratnam and Haykin, 2009].

7.3.1 Analytical example

In the first experiment, I considered a simple example, where the transformed moments can be computed exactly. A well known fact is that a random variable given by sum of squares function (SOS)

$$z = g_{\mathrm{SOS}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{x}, \quad \text{where} \quad \mathbf{x} \sim \mathrm{N}(\mathbf{0}, \mathbf{I}_D) \tag{7.3.1}$$

has density $z \sim \chi^2(D)$ with mean D and variance $2D$. Kernel parameters of both GPQ-based transforms were $\alpha = 1$ and $\lambda = 10$. It is apparent from the Table 7.3.1, that,

D	1	5	10	25
Mean				
True	1.00	5.00	10.00	25.00
SR	1.00	4.95	10.00	25.00
GPQ	1.00	5.00	10.00	25.02
GPQ+D	0.99	5.00	9.89	24.49
Variance				
True	2.00	10.00	20.00	50.00
SR	0.00	0.00	0.00	0.00
GPQ	0.00	0.01	0.05	0.78
GPQ+D	**1.92**	**9.61**	**19.16**	**46.44**

Table 7.3.1: Comparison of transformed mean and variance for increasing dimension D computed by the SR, GPQ and GPQ+D moment transforms.

while SR and GPQ transforms capture the mean fairly accurately, they completely fail to capture the variance of transformed random variable. GPQ+D is the only transform that comes close to the true transformed variances. Table 7.3.2 demonstrates that by including gradient information the variance of the mean integral decreases. Figure 7.3.1 shows the GP regression fit to eq. (7.3.1) for $D = 1$. The ordinary GP regression fit, in

Figure 7.3.1a, fails to capture the true function, whereas by including the derivative information the fit improves significantly as seen in Figure 7.3.1b. This is because we are using symmetric sigma-point set and eq. (7.3.1) is an even function.

(a) (b)

Figure 7.3.1: (a) Approximation used by GPQ. (b) Approximation used by GPQ+D.

D	1	5	10	25
GPQ	1.14e-08	5.56e-08	3.31e-07	4.62e-06
GPQ+D	**6.62e-09**	**3.16e-08**	**2.75e-07**	**4.27e-06**

Table 7.3.2: Comparison of variance of the mean integral for GPQ and GPQ+D. Overall, including derivative information decreases variance

7.3.2 Sensor network measurements

The second experiment is inspired by Gustafsson and Hendeby [2012], who considered nonlinear measurements commonly encountered in sensor networks. These are, time of arrival (TOA), direction of arrival (DOA) and received signal strength (RSS) given by

$$g_{\text{TOA}}(\mathbf{x}) = \|\mathbf{x}\|_2, \tag{7.3.2}$$

$$g_{\text{DOA}}(\mathbf{x}) = \operatorname{atan2}(x_1, x_2), \tag{7.3.3}$$

$$g_{\text{RSS}}(\mathbf{x}) = 10 - 20\log_{10}(\|\mathbf{x}\|_2^2), \tag{7.3.4}$$

where $\operatorname{atan2}$ is the four-quadrant variant of atan. As an example of vector function, I considered radar measurements (RDR) which arise as a mapping of range r and bearing θ to Cartesian coordinates given by

$$\mathbf{g}_{\text{RDR}}(\mathbf{x}) = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}. \tag{7.3.5}$$

The symmetrized KL-divergence was used to measure the distance between the baseline moments of the Gaussian distribution, computed by Monte Carlo transform with $20\,000$

samples, and the transformed distribution computed by SR, GPQ and GPQ+D respectively. The covariance of the input distribution was randomly generated and results were averaged over 100 MC simulations. Since DOA and RDR functions are limited to two-dimensional inputs, I tested for $D = 2$ only. Table 7.3.4 shows that including gradient observations in GPQ can improve the average symmetrized KL-divergence; in case of TOA, even by two orders of magnitude. Values of the kernel parameters for

	TOA	RSS	DOA	RDR
λ	3.0	0.2	2.0	5.0
α	10	10	1	1

Table 7.3.3: Values of the RBF kernel parameters.

individual functions are summarized in Table 7.3.3.

	SR	GPQ	GPQ+D
TOA	2.74e-02	3.37e-01	**4.61e-03**
RSS	4.48e+00	4.76e-01	**4.70e-01**
DOA	5.48e-03	5.99e-03	**1.80e-03**
RDR	6.48e-01	7.07e-01	**2.94e-01**

Table 7.3.4: Comparison of the SR, GPQ and GPQ+D moment transforms in terms of symmetrized KL-divergence performance.

7.4 Conclusion

In this chapter I analyzed the use of gradient observations in GP quadratures and designed general moment transformations based on GP quadrature with gradients. The proposed transforms were tested on a range of functions arising in the sensor network applications. Using Monte Carlo simulations, I have shown that including additional gradient observations improve the accuracy of computed moments as measured by the symmetrized KL-divergence. Finally, I gave proofs for the two limit cases when the proposed transform reduces to the linear transform based on Taylor series, which meets the final goal of this thesis set out in Chapter 4.

Chapter 8

Conclusion

This thesis focused on the application of the Gaussian process regression models for the recursive system identification and applications of the Bayesian quadrature for design of the moment transformations, commonly found in state estimation algorithms.

The early chapters were focused on the preliminaries necessary for building up the background to the contributions presented in the later chapters. The contemporary system identification algorithms for linear as well as nonlinear systems were reviewed and detailed explanation of the non-parametric Gaussian process regression model was presented. The current state-of-the-art in the local nonlinear filtering was reviewed next, where the moment transforms were identified as the key component. The Bayesian quadrature was presented as a novel alternative view to the numerical integration offering many advantages over the classical methods.

Leveraging advantages of the Gaussian process regression for recursive system identification was the main theme of the **Goal 1** stated in Chapter 4 as:

(a) *Investigate suitability of the RGP algorithm for recursive system identification as means for decreasing computational demands of the original GP model.*

(b) *Bound the computational demands for kernel parameter optimization.*

In Chapter 5, I applied the recursive GP algorithm to system identification and proposed an ad-hoc kernel parameter optimization procedure, which significantly reduced the per-iteration computational cost. The proposed RGP identification algorithm was compared with the full GP in the system identification experiments. Even though the RGP is an approximation, I found it gave overall comparable performance to the FGP under mild assumptions on the boundedness of the state trajectory. The RGP identification was later applied in the functional adaptive control loop. Despite its slower adaptation, the

experiments showed that the RGP control can achieve comparable performance in terms of tracking error. With it's current basis vector placement scheme, the proposed RGP control algorithm is practically applicable to lower-order systems. For higher-order systems, an alternative placement scheme, which scales favorably with the dimension of the state vector, would have to be developed.

Chapter 6 focused on the moment transforms based on the Bayesian quadrature and addressed the **Goal 2**, which revolved around improvement of the current state-of-the-art moment transformations. The sub-goals were stated as follows:

(a) *Leverage the statistical view of quadrature for the design of general purpose moment transformations and incorporate the approximation error in the output mean, in eq.* (3.2.2a)*, into the moment transformation process.*

(b) *Use the proposed moment transformations to improve estimate quality of the nonlinear sigma-point filters; especially in terms of credibility of the covariance estimates.*

The problem of incorporating the integral variance was elegantly solved for any probabilistic model of the integrand. I utilized the GP and the TP regression, because they are analytically convenient, and designed the corresponding general purpose moment transforms. Employing the BQ-based transforms in both the Gaussian as well as the Student's t-filters resulted in significantly improved credibility of the covariance estimates, thus lending credence to the suspicions raised in the Chapter 4 about the leaking integration error. For higher dimensions of the system state it is becoming increasingly difficult to come up with good initial guesses for the kernel parameters that yield competitive results, which is why I would recommend these filters for systems with lower state dimension.

The Chapter 7 was concerned with the design of GPQ transforms that can make use of the information about derivatives of the integrand and thus addressed the final **Goal 3** of this thesis, formulated as:

(a) *Explore the question of derivative information in BQ as means for reducing the integral variance and attempt to find connections with classical linearization.*

Perhaps not so surprising theoretical connection, which I proved in Chapter 7, is that the GPQ transform with derivatives can be reduced to familiar linearization for suitable choices of kernels and sigma-points. Including additional derivative information helps reduce the integral variance and, as shown experimentally, the derivatives play a decisive role in situations when the integrand is an even function and the sigma-point set is symmetric.

8.1 Challenges and Future Work

The Bayesian approach to quadrature is not without its challenges. One peculiarity is that the point-sets are not prescribed by the rule, which can either be viewed as an opportunity, because they could be optimized to minimize the integral variance; or a burden, because the additional degree of freedom introduces ambiguity regarding the choice of criteria for point selection. More importantly, the proposed BQ transforms assume that we can guess suitable values of the parameters, which is certainly possible if the analytic form of the integrand is known a priori, as is the case in nonlinear filtering. The standard evidence maximization method for fitting GPs works well if the number of points is large enough, but produces unsatisfactory estimates for minimal point-sets, such as those used in local filters. A principled method for optimizing the parameters for small point-sets still remains to be found and presents an interesting challenge for future research.

In the GPQ and TPQ, the choice of the kernel plays a decisive role, because it encodes assumptions on the integrand and also affects computations of the quadrature weights through the kernel expectations. Ideally, we need to choose the kernel such that the induced assumptions about the integrand are acceptable for the given application and the kernel expectations are tractable. Meeting both criteria at the same time is very hard, if not impossible in some cases, which is why the use of approximate kernel expectations might become inevitable. The proposed moment transforms relied on the commonly used RBF kernel for tractability reasons. The induced assumption is that the integrand is infinitely differentiable (smooth), which may be unacceptable for some applications.

In spite of these challenges, I hope to have shown that accounting for the integration error in local filters is a worthy pursuit, which will spur further research.

Appendix A

Appendix

A.1 Multivariate Gaussian Probability Density Function

$$\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}) = |2\pi\mathbf{P}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\top}\mathbf{P}^{-1}(\mathbf{x}-\mathbf{m})\right) \tag{A.1.1}$$

Product of Two Gaussian Probability Density Functions

According to [Petersen and Pedersen, 2012] the formula for the product of two Gaussian densities can be written as

$$\mathrm{N}(\mathbf{x} \,|\, \mathbf{m}_1,\, \mathbf{P}_1)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m}_2,\, \mathbf{P}_2) = \mathrm{N}(\mathbf{m}_1 \,|\, \mathbf{m}_2,\, \mathbf{P}_1 + \mathbf{P}_2)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}) \tag{A.1.2}$$

where

$$\mathbf{m} = \mathbf{P}\big(\mathbf{P}_1^{-1}\mathbf{m}_1 + \mathbf{P}_2^{-1}\mathbf{m}_2\big) \tag{A.1.3}$$

$$\mathbf{P} = \big(\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1}\big)^{-1} \tag{A.1.4}$$

A.2 Multivariate Student's t-density Function

In the following, let $\mathbf{x} \in \mathbb{R}^D$ be a Student's t-distributed random variable. Multivariate Student's t-density can be defined in various ways, see [Kotz and Nadarajah, 2004].

Scale parametrization

This is the most commonly encountered parametrization.

$$\mathrm{St}(\mathbf{x}\,|\,\mathbf{m},\,\boldsymbol{\Sigma},\,\nu) = \frac{\Gamma\left(\frac{\nu+D}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)|\nu\pi\boldsymbol{\Sigma}|^{\frac{1}{2}}}\left(1+\frac{1}{\nu}(\mathbf{x}-\mathbf{m})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{m})\right)^{-\frac{\nu+D}{2}} \tag{A.2.1}$$

Covariance parametrization

Plugging the relationship between the scale and covariance matrices $\boldsymbol{\Sigma} = \frac{\nu-2}{\nu}\mathbf{P}$ into eq. (A.2.1), yields

$$\mathrm{St}(\mathbf{x}\,|\,\mathbf{m},\,\mathbf{P},\,\nu) = \frac{\Gamma\left(\frac{\nu+D}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)|(\nu-2)\pi\mathbf{P}|^{\frac{1}{2}}}\left(1+\frac{1}{\nu-2}(\mathbf{x}-\mathbf{m})^{\top}\mathbf{P}^{-1}(\mathbf{x}-\mathbf{m})\right)^{-\frac{\nu+D}{2}}. \tag{A.2.2}$$

A.3 RBF Kernel

For $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$, the radial basis function kernel (RBF) is given by

$$k(\mathbf{x},\mathbf{x}') = \alpha^2 \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^{\top}\boldsymbol{\Lambda}^{-1}(\mathbf{x}-\mathbf{x}')\right), \tag{A.3.1}$$

with $\boldsymbol{\Lambda} = \mathrm{diag}\big(\begin{bmatrix}\lambda_1 & \dots & \lambda_D\end{bmatrix}\big)$, where α and λ_d are parameters. Comparing eqs. (A.1.1) and (A.3.1), the RBF kernel can be written with the help of Gaussian PDF as

$$k(\mathbf{x},\mathbf{x}') = \alpha^2|2\pi\boldsymbol{\Lambda}|^{\frac{1}{2}}\mathrm{N}\big(\mathbf{x}\,\big|\,\mathbf{x}',\,\boldsymbol{\Lambda}\big) \tag{A.3.2}$$

RBF Kernel Expectations w.r.t. Gaussian Density

This section contains derivations of the RBF kernel expectations w.r.t. an arbitrary Gaussian PDF.

$$[\mathbf{q}]_n = \mathbb{E}_{\mathbf{x}}[k(\mathbf{x}, \mathbf{x}_n)] = \int k(\mathbf{x}, \mathbf{x}_n)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P})\,\mathrm{d}\mathbf{x} \tag{A.3.3}$$

$$= \alpha^2 |2\pi\mathbf{\Lambda}|^{\frac{1}{2}} \int \mathrm{N}(\mathbf{x} \,|\, \mathbf{x}_n,\, \mathbf{\Lambda})\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P})\,\mathrm{d}\mathbf{x} \tag{A.3.4}$$

$$= \alpha^2 |2\pi\mathbf{\Lambda}|^{\frac{1}{2}} \mathrm{N}(\mathbf{x}_n \,|\, \mathbf{m},\, \mathbf{\Lambda} + \mathbf{P}) \int \mathrm{N}(\mathbf{x} \,|\, \boldsymbol{\mu},\, \mathbf{\Sigma})\,\mathrm{d}\mathbf{x} \tag{A.3.5}$$

$$= \alpha^2 |2\pi\mathbf{\Lambda}|^{\frac{1}{2}} \mathrm{N}(\mathbf{x}_n \,|\, \mathbf{m},\, \mathbf{\Lambda} + \mathbf{P}) \tag{A.3.6}$$

Using the definition of the Gaussian density in eq. (A.1.1), the expression simplifies to

$$[\mathbf{q}]_n = \alpha^2 \left|\mathbf{\Lambda}^{-1} + \mathbf{I}\right|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\mathbf{x}_n - \mathbf{m})^\top (\mathbf{\Lambda} + \mathbf{I})^{-1}(\mathbf{x}_n - \mathbf{m})\right) \tag{A.3.7}$$

$$[\mathbf{R}]_{*n} = \mathbb{E}_{\mathbf{x}}[\mathbf{x}k(\mathbf{x}, \mathbf{x}_n)] = \int \mathbf{x}k(\mathbf{x}, \mathbf{x}_n)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P})\,\mathrm{d}\mathbf{x} \tag{A.3.8}$$

$$= \alpha^2 |2\pi\mathbf{\Lambda}|^{\frac{1}{2}} \mathrm{N}(\mathbf{x}_n \,|\, \mathbf{m},\, \mathbf{\Lambda} + \mathbf{P}) \int \mathbf{x}\mathrm{N}(\mathbf{x} \,|\, \boldsymbol{\mu},\, \mathbf{\Sigma})\,\mathrm{d}\mathbf{x} \tag{A.3.9}$$

$$= [\mathbf{q}]_n \boldsymbol{\mu} = [\mathbf{q}]_n (\mathbf{\Lambda} + \mathbf{I})^{-1}\mathbf{x}_n, \tag{A.3.10}$$

where $[\mathbf{q}]_n$ is defined above.

$$
\begin{aligned}
[\mathbf{Q}]_{nm} &= \mathbb{E}_{\mathbf{x}}[k(\mathbf{x}, \mathbf{x}_n)k(\mathbf{x}, \mathbf{x}_m)] = \int k(\mathbf{x}, \mathbf{x}_n)k(\mathbf{x}, \mathbf{x}_m)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}\,)\,\mathrm{d}\mathbf{x} \\
&= \alpha^4 \big|(2\pi)^2\boldsymbol{\Lambda}^2\big|^{\frac{1}{2}} \int \mathrm{N}(\mathbf{x} \,|\, \mathbf{x}_n,\, \boldsymbol{\Lambda}\,)\mathrm{N}(\mathbf{x} \,|\, \mathbf{x}_m,\, \boldsymbol{\Lambda}\,)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}\,)\,\mathrm{d}\mathbf{x} \\
&= \alpha^4 \big|(2\pi)^2\boldsymbol{\Lambda}^2\big|^{\frac{1}{2}} \mathrm{N}(\mathbf{x}_n \,|\, \mathbf{x}_m,\, 2\boldsymbol{\Lambda}\,) \int \mathrm{N}\big(\mathbf{x} \,\big|\, \mathbf{z}_{nm},\, \tfrac{1}{2}\boldsymbol{\Lambda}\big)\mathrm{N}(\mathbf{x} \,|\, \mathbf{m},\, \mathbf{P}\,)\,\mathrm{d}\mathbf{x} \\
&= \alpha^4 \big|(2\pi)^2\boldsymbol{\Lambda}^2\big|^{\frac{1}{2}} \mathrm{N}(\mathbf{x}_n \,|\, \mathbf{x}_m,\, 2\boldsymbol{\Lambda}\,)\mathrm{N}\big(\mathbf{z}_{nm} \,\big|\, \mathbf{m},\, \tfrac{1}{2}\boldsymbol{\Lambda} + \mathbf{P}\big) \int \mathrm{N}(\mathbf{x} \,|\, \cdot,\, \cdot\,)\,\mathrm{d}\mathbf{x} \\
&= \alpha^4 \big|(2\pi)^2\boldsymbol{\Lambda}^2\big|^{\frac{1}{2}} |2\pi 2\boldsymbol{\Lambda}|^{-\frac{1}{2}} \big|2\pi(\tfrac{1}{2}\boldsymbol{\Lambda} + \mathbf{P})\big|^{-\frac{1}{2}} \\
&\qquad \times \exp\Big(-\tfrac{1}{2}(\mathbf{x}_n - \mathbf{x}_m)^\top \tfrac{1}{2}\boldsymbol{\Lambda}^{-1}(\mathbf{x}_n - \mathbf{x}_m)\Big) \\
&\qquad \times \exp\Big(-\tfrac{1}{2}(\mathbf{z}_{nm} - \mathbf{m})^\top (\tfrac{1}{2}\boldsymbol{\Lambda} + \mathbf{P})^{-1}(\mathbf{z}_{nm} - \mathbf{m})\Big) \\
&= \alpha^4 \big|2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\big|^{-\frac{1}{2}} \qquad \text{(A.3.11)} \\
&\times \exp\Big(-\tfrac{1}{2}(\mathbf{x}_n - \mathbf{x}_m)^\top \tfrac{1}{2}\boldsymbol{\Lambda}^{-1}(\mathbf{x}_n - \mathbf{x}_m) - \tfrac{1}{2}(\mathbf{z}_{nm} - \mathbf{m})^\top (\tfrac{1}{2}\boldsymbol{\Lambda} + \mathbf{P})^{-1}(\mathbf{z}_{nm} - \mathbf{m})\Big) \qquad \text{(A.3.12)}
\end{aligned}
$$

where $\mathbf{z}_{nm} = \frac{1}{2}(\mathbf{x}_n + \mathbf{x}_m)$. Deisenroth et al. [2009, p.22] gives a simplified form of the exponent, so that

$$
[\mathbf{Q}]_{nm} = \alpha^4 \big|2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}\big|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2}\Big[\boldsymbol{\zeta}_n^\top \boldsymbol{\Lambda}^{-1}\boldsymbol{\zeta}_n + \boldsymbol{\zeta}_m^\top \boldsymbol{\Lambda}^{-1}\boldsymbol{\zeta}_m - \mathbf{z}_{nm}^\top \mathbf{R}^{-1}\mathbf{P}\mathbf{z}_{nm}\Big]\Big) \qquad \text{(A.3.13)}
$$

where $\boldsymbol{\zeta}_n = \mathbf{x}_n - \mathbf{m}$ and $\mathbf{R} = 2\boldsymbol{\Lambda}^{-1}\mathbf{P} + \mathbf{I}$.

Bibliography

M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover Publications, 1965. ISBN 978-0486612720.

R. J. Adler. *Geometry of Random Fields*. Wiley, 1981. ISBN 978-0898716931.

I. Arasaratnam and S. Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.

I. Arasaratnam, S. Haykin, and R.J. Elliott. Discrete-Time Nonlinear Filtering Algorithms Using Gauss-Hermite Quadrature. *Proceedings of the IEEE*, 95(5):953–977, 2007. ISSN 0018-9219. doi: 10.1109/JPROC.2007.894705.

K. J. Åström and B. Wittenmark. *Adaptive Control*. Prentice Hall, 2nd edition edition, 1995. ISBN 978-0201558661.

M. Athans, R. Wishner, and A. Bertolini. Suboptimal State Estimation for Continuous-Time Nonlinear Systems from Discrete Noisy Measurements. *IEEE Transactions on Automatic Control*, 13(5):504–514, 1968. doi: 10.1109/TAC.1968.1098986.

Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research*, 18(1):714–751, January 2017. ISSN 1532-4435.

D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012. ISBN 978-0-521-51814-7.

O. Barndorff-Nielsen, J. Kent, and M. Sørensen. Normal variance-mean mixtures and z distributions. *International Statistical Review*, (50):145–159, 1982.

Bradley M. Bell and Frederick W. Cathey. Iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993. doi: 10.1109/9.250476.

R. Bhar. *Stochastic filtering with applications in finance*. World Scientific, 2010. ISBN 978-981-4304-85-6.

S. A. Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domans*. Wiley, 2013.

S. A. Billings and S. Chen. Extended model set, global data and threshold model identification of severely non-linear systems. *International Journal of Control*, 50(5): 1897–1923, 1989. doi: 10.1080/00207178908953473.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007. ISBN 978-0387310732.

F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic. Probabilistic Integration: A Role for Statisticians in Numerical Analysis? *ArXiv e-prints*, December 2015.

M. K. Bugeja, S. G. Fabri, and L. Camilleri. Dual adaptive control of mobile robots using neural networks. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 39(1): 129–141, 2009.

T. D. Bui and R. E. Turner. Tree-structured Gaussian Process Approximations. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2213–2221. Curran Associates, Inc., 2014.

K. Chalupka, C. K. I. Williams, and I. Murray. A Framework for Evaluating Approximation Methods for Gaussian Process Regression. *The Journal of Machine Learning Research*, 14:333–350, 2013.

F. C. Chen and H. K. Khalil. Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Transactions on Automatic Control*, 40(5):791–801, 1995.

S. Chen, S. A. Billings, and P. M. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990. doi: 10.1080/00207179008934126.

G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela. Bayesian Nonparametric Adaptive Control Using Gaussian Processes. *Neural Networks and Learning Systems, IEEE Transactions on*, 26(3):537–550, 2014. doi: 10.1109/TNNLS.2014.2319052.

R. Cools. Constructing cubature formulae: the scince behind the art. *Acta Numerica*, pages 1–54, 1997.

N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1993. ISBN 978-0471002550.

L. Csató and M. Opper. Sparse Online Gaussian Processes. *Neural Computation*, 14(3): 641–668, 2002.

M. P. Deisenroth. *Efficient Reinforcement Learning Using Gaussian Processes*. PhD thesis, KIT, 2009.

M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck. Analytic moment-based Gaussian process filtering. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8. ACM Press, 2009. doi: 10.1145/1553374.1553403.

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*, 57(7):1865–1871, 2012. doi: 10.1109/TAC.2011.2179426.

P. Diaconis. Bayesian numerical analysis. In S. S. Gupta and J. O. Berger, editors, *Statistical Decision Theory and Related Topics IV*, pages 163–175. Springer, 1988.

A. Doucet, N. De Freitas, and N. Gordon. Sequential Monte Carlo Methods in Practice. *Springer New York*, pages 178–195, 2001. ISSN 1530-888X. doi: 10.1198/tech.2003.s23.

J. Duník, O. Straka, and M. Šimandl. Stochastic Integration Filter. *IEEE Transactions on Automatic Control*, 58(6):1561–1566, 2013.

J. Duník, O. Straka, M. Mallick, and E. Blasch. Survey of Nonlinearity and Non-Gaussianity Measures for State Estimation. In *19th International Conference on Information Fusion*, pages 1845–1852, July 2016.

D. K. Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

S. G. Fabri and M. K. Bugeja. Unscented Transform-based Dual Adaptive Control of Nonlinear MIMO Systems. In *Proceedings of 13th European Control Conference*, pages 392–397, 2013. Zürich.

S. G. Fabri and V. Kadirkamanathan. Dual Adaptive Control of Nonlinear Stochastic Systems using Neural Networks. *Automatica*, 34(2):245–253, 1998.

S. G. Fabri and V. Kadirkamanathan. *Functional Adaptive Control: An Intelligent Systems Approach*. Springer-Verlag, 2001.

A. A. Fel'dbaum. *Optimal Control Systems*. Academic Press, New York, 1965.

N. M. Filatov and H. Unbehauen. *Adaptive Dual Control*. Springer-Verlag, 2004.

W. Gautschi. *Orthogonal Polynomials: Computation and Approximation*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2004. ISBN 978-0198506720.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. Chapman and Hall/CRC, 3rd edition, 2013. ISBN 978-1439840955.

A. Genz and J. Monahan. Stochastic Integration Rules for Infinite Regions. *SIAM Journal on Scientific Computation*, 19:426—-439, 1996. doi: 10.1137/S1064827595286803.

Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18(3-4):209–232, 1998. doi: 10.1023/A:1019129717644.

Z. Ghahramani. Bayesian nonparametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A*, 371(1984):1–27, 2013. ISSN 1364-503X. doi: 10.1098/rspa.00000000.

S. Gillijns, O.B. Mendoza, J. Chandrasekar, B.L.R. De Moor, D.S. Bernstein, and A. Ridley. What is the ensemble kalman filter and how well does it work? In *American Control Conference, 2006*, page 6, June 2006. doi: 10.1109/ACC.2006.1657419.

A. Girard, C. E. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith. Gaussian Process Priors With Uncertain Inputs Application to Multiple-Step Ahead Time Series Forecasting. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 545–552. MIT Press, 2003.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993.

M. S. Grewal, L. R. Weill, and A. P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. Wiley, 2007. ISBN 978-0-470-09971-1.

G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 3rd edition, 2001. ISBN 978-0198572220.

T. Gunter, M. A. Osborne, R. Garnett, P. Hennig, and S. J. Roberts. Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2789–2797. Curran Associates, Inc., 2014.

Dong Guo and Xiaodong Wang. Quasi-monte carlo filtering in nonlinear dynamic systems. *Signal Processing, IEEE Transactions on*, 54(6):2087–2098, 2006. doi: 10.1109/TSP.2006.873585.

F. Gustafsson. Praticle Filter Theory and Practice with Positioning Applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–81, 2010.

F. Gustafsson and G. Hendeby. Some Relations Between Extended and Unscented Kalman Filters. *IEEE Transactions on Signal Processing*, 60(2):545–555, 2012.

J. Hartikainen and S. Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 379–384, 2010. doi: 10.1109/MLSP.2010.5589113.

S. Haykin. *Kalman Filtering and Neural Networks*. Wiley, 2001.

P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015. ISSN 1364-5021. doi: 10.1098/rspa.2015.0142.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data. In *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013*, pages 282–290, 2013.

R. Herzallah and D. Lowe. A novel approach to modelling and exploiting uncertainty in stochastic control systems. In *Artificial Neural Networks - ICANN 2002*, pages 801–806. Springer Berlin Heidelberg, 2002.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990. ISBN 978-0-521-38632-6.

M. F. Huber. Recursive Gaussian process regression. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3362–3366. IEEE, 2013. doi: 10.1109/ICASSP.2013.6638281.

M. F. Huber. Recursive Gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45:85–91, 2014. doi: 10.1016/j.patrec.2014.03.004.

F. Huszár and D. Duvenaud. Optimally-Weighted Herding is Bayesian Quadrature. In *Uncertainty in Artificial Intelligence - Proceedings of the 28th Conference, UAI 2012*, pages 377–386, 2012.

K. Ito and K. Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, 2000. doi: 10.1109/9.855552.

S. Janson. *Gaussian Hilbert Spaces*. Cambridge University Press, 1997. ISBN 978-0521057202.

A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970. ISBN 978-0-486-46274-5.

T. Jiang, N.D. Sidiropoulos, and G.B. Giannakis. Kalman filtering for power estimation in mobile communications. *Wireless Communications, IEEE Transactions on*, 2(1):151–161, 2003. doi: 10.1109/TWC.2002.806386.

S. J. Julier. The Scaled Unscented Transformation. In *Proceedings of the 2002 American Control Conference*, pages 4555–4559. IEEE, 2002.

S. J. Julier and J. K. Uhlmann. A General Method for Approximating Nonlinear Transformations of Probability Distributions. Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford, 1996.

S. J. Julier and J. K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. doi: 10.1109/JPROC.2003.823141.

S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

S.J. Julier and J.K. Uhlmann. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. In *American Control Conference, 2002. Proceedings of the 2002*, volume 2, pages 887–892 vol.2, 2002. doi: 10.1109/ACC.2002.1023128.

T. Kailath. A View of Three Decades of Linear Filtering Theory. *IEEE Transactions on Information Theory*, 20(2):146–181, 1974. doi: 10.1109/TIT.1974.1055174.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

R. E. Kalman and R. S. Bucy. New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95, 1961. doi: 10.1115/1.3658902.

Motonobu Kanagawa, Bharath K. Sriperumbudur, and Kenji Fukumizu. Convergence guarantees for kernel-based quadrature rules in misspecified settings. In *Advances in Neural Information Processing Systems 29*, pages 3288–3296. Curran Associates, Inc., 2016.

T. Karvonen and S. Särkkä. Classical quadrature rules via Gaussian processes. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017a. ISBN 9781509063413.

T. Karvonen and S. Särkkä. Fully symmetric kernel quadrature. *arXiv:1703.06359*, 2017b.

G. Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar. Predictive control with Gaussian process models. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, pages 352–356, 2003. ISBN 078037763X. doi: 10.1109/EURCON.2003.1248042.

J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamic Systems*, 11 (4):411–424, 2005. doi: 10.1080/13873950500068567.

M. Korenberg, S. A. Billings, Y. P. Liu, and P. J. McIlroy. Orthogonal parameter estimation algorithm for non-linear stochastic systems. *International Journal of Control*, 48(1):193–210, 1988. doi: 10.1080/00207178808906169.

S. Kotz and S. Nadarajah. *Multivariate T-Distributions and Their Applications*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511550683.

L. Král and M. Šimandl. Functional adaptive control for multi-input multi-output systems. In *Proceedings of the 17th IFAC World Congress*. Seoul: IFAC, 2008.

L. Král and M. Šimandl. Functional adaptive controller for multivariable stochastic systems with dynamic structure of neural network. *Int. J. Adapt. Control Signal Process.*, 25:949–964, 2011.

L. Král, J. Prüher, and M. Šimandl. Gaussian Process Based Dual Adaptive Control of Nonlinear Stochastic Systems. In *22nd Mediterranean Conference of Control and Automation (MED)*, pages 1074–1079, 2014. doi: 10.1109/MED.2014.6961517.

G. Kurz and U. D. Hanebeck. Linear regression kalman filtering based on hyperspherical deterministic sampling. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 977–983, Dec 2017. doi: 10.1109/CDC.2017.8263785.

H Kushner. Approximations to Optimal Nonlinear Filters. *Automatic Control, IEEE Transactions on*, 12(5):546–556, 1967. doi: 10.1109/TAC.1967.1098671.

M. Kuss and C. E. Rasmussen. Gaussian Processes in Reinforcement Learning. In S. Thrun, L. K. Saul, and P. B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 751–758. MIT Press, 2004.

M. Lázaro-Gredilla, J. Quiñonero Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse Spectrum Gaussian Process Regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

N. Le Roux and Y. Bengio. Continuous neural networks. January 2007.

X. R. Li and Z. Zhao. Measuring Estimator's Credibility: Noncredibility Index. In *Information Fusion, 2006 9th International Conference on*, pages 1–8, 2006. doi: 10.1109/ICIF.2006.301770.

L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition edition, 1999. ISBN 9780137156498.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. ISBN 978-0521642989.

P. S. Maybeck. *Stochastic Models, Estimation and Control: Volume 2*. Academic Press, 1982. ISBN 978-0124110434.

J. McNamee and F. Stenger. Construction of fully symmetric numerical integration formulas. *Numerische Mathematik*, 10(4):327–344, 1967. ISSN 0029599X. doi: 10.1007/BF02162032.

R. Milito, C. S. Padilla, R. A. Padilla, and D. Cadorin. An inovations approach to dual control. *IEEE Transactions on Automatic Control*, 27(1):133–137, 1982.

T. P. Minka. Deriving Quadrature Rules from Gaussian Processes. Technical report, Statistics Department, Carnegie Mellon University, Tech. Rep, 2000.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. ISBN 978-0-262-01802-9.

R. Murray-Smith and B. A. Pearlmutter. *Transformations of Gaussian Process Priors*, pages 110–123. Springer, 2005. ISBN 978-3-540-31728-9. doi: 10.1007/11559887_7.

R. M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

M. Nørgaard, N. K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36:1627–1638, 2000.

A. O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29 (3):245–260, 1991. doi: 10.1016/0378-3758(91)90002-V.

A. O'Hagan. Some Bayesian Numerical Analysis. *Bayesian Statistics*, 4:345–363, 1992.

P. Orbanz and Y. W. Teh. *Bayesian Nonparametric Models*, pages 81–89. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_66.

M. A. Osborne, C. E. Rasmussen, D. K. Duvenaud, R. Garnett, and S. J. Roberts. Active Learning of Model Evidence Using Bayesian Quadrature. In *Advances in Neural Information Processing Systems*, pages 46–54, 2012.

S. Park, S. K. Mustafa, and K. Shimada. Learning Based Robot Control with Sequential Gaussian Process. In *Robotic Intelligence In Informationally Structured Space (RiiSS), 2013 IEEE Workshop on*, pages 120–127. Ieee, 2013. doi: 10.1109/RiiSS.2013.6607939.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`. Version 20121115.

R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach*. Wiley-IEEE Press, 2nd edition edition, 2012. ISBN 978-0-470-64037-1.

H. Poincaré. *Calcul des probabilités*. Paris, Gauthier-Villars, 1896.

J. Prüher and L. Král. Functional Dual Adaptive Control with Recursive Gaussian Process Model. *Journal of Physics: Conference Series*, 659:012006, 2015. ISSN 1742-6588. doi: 10.1088/1742-6596/659/1/012006.

J. Prüher and S. Särkkä. On the Use of Gradient Information in Gaussian Process Quadratures. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Sept 2016. doi: 10.1109/MLSP.2016.7738903.

J. Prüher and O. Straka. Gaussian Process Quadrature Moment Transform. *IEEE Transactions on Automatic Control*, Pre-print(99):1–1, 2017. ISSN 0018-9286. doi: 10.1109/TAC.2017.2774444.

J. Prüher and M. Šimandl. Gaussian process based recursive system identification. *Journal of Physics: Conference Series*, 570(1):1–9, 2014. doi: 10.1088/1742-6596/570/1/012002.

J. Prüher and M. Šimandl. Bayesian Quadrature in Nonlinear Filtering. In *12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 01, pages 380–387, July 2015.

J. Prüher and M. Šimandl. Bayesian Quadrature Variance in Sigma-point Filtering. In *Informatics in Control, Automation and Robotics, 12th International Conference, Colmar, Alsace, France, 21-23 July, 2015 Revised Selected Papers*, volume 370 of *Lecture Notes in Electrical Engineering*. Springer International Publishing, 2016.

J. Prüher, F. Tronarp, T. Karvonen, S. Särkkä, and O. Straka. Student-t Process Quadratures for Filtering of Non-linear Systems with Heavy-tailed Noise. In *20th International Conference on Information Fusion (Fusion)*, pages 1–8, July 2017. doi: 10.23919/ICIF.2017.8009742.

J. Quiñonero Candela and C. E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

J. Quiñonero Candela, C. E. Rasmussen, and C. K. I. Williams. Approximation Methods for Gaussian Process Regression. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*, chapter 9, pages 203–223. MIT Press, 2007.

A. Ranganathan, M. H. Yang, and J. Ho. Online Sparse Gaussian Process Regression and Its Applications. *IEEE Transactions on Image Processing*, 20(2):391–404, 2011.

C. E. Rasmussen. The Infinite Gaussian Mixture Model. In S. A. Solla, T. K. Leen, and K. R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 554–560. MIT Press, 1999.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN 978-0-262-18253-9.

S. Reece and S. Roberts. An Introduction to Gaussian Processes for the Kalman Filter Expert. In *Proceedings of the 13th International Conference on Information Fusion*, pages 1–9. IEEE, 2010.

M. Roth and F. Gustafsson. An efficient implementation of the second order extended Kalman filter. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–6, 2011.

M. Roth, E. Özkan, and F. Gustafsson. A student's t filter for heavy tailed process and measurement noise. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5770–5774, May 2013. doi: 10.1109/ICASSP.2013.6638770.

F. Sandblom and L. Svensson. Marginalized Sigma-Point Filtering. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8, 2011.

J. Sarangapani. *Neural Network Control of Nonlinear Disrete-Time Systems*. Taylor & Francis, 2006.

S. Särkkä. Linear operators and stochastic partial differential equations in Gaussian process regression. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 151–158. Springer, 2011.

S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, New York, 2013. ISBN 978-1-107-61928-9.

S. Särkkä and J. Hartikainen. Infinite-dimensional Kalman filtering approach to spatio-temporal Gaussian process regression. *International Conference on Artificial Intelligence and Statistics*, pages 993–1001, 2012.

S. Särkkä, J. Hartikainen, L. Svensson, and F. Sandblom. Gaussian Process Quadratures in Nonlinear Sigma-Point Filtering and Smoothing. In *Information Fusion, 2014 17th International Conference on*, pages 1–8, 2014.

S. Särkkä, J. Hartikainen, L. Svensson, and F. Sandblom. On the relation between Gaussian process quadratures and sigma-point methods. *Journal of Advances in Information Fusion*, 11:31–46, June 2016.

D. Sbarbaro and R. Murray-Smith. Self-tuning Control of Non-linear Systems Using Gaussian Process Prior Models. In R. Murray-Smith and R. Shorten, editors, *Switching*

and Learning in Feedback Systems, pages 140–157. Springer, Berlin Heidelberg, 2003. doi: 10.1007/978-3-540-30560-6_6.

D. Sbarbaro, R. Murray-Smith, and A. Valdes. Multivariable generalized minimum variance control based on artificial neural networks and Gaussian process models. In *Advances in Neural Networks - ISNN 2004, International Symposium on Neural Networks, Dalian, China, August 19-21, 2004, Proceedings, Part II*, pages 52–58, 2004. doi: 10.1007/978-3-540-28648-6_8.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002. ISBN 978-0-262-19475-4.

A. Schwaighofer and V. Tresp. Transductive and Inductive Methods for Approximate Gaussian Process Regression. In *Advances in Neural Information Processing Systems 15*, pages 977–984. MIT Press, 2003.

A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t Processes as Alternatives to Gaussian Processes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33, pages 877–885, 2014.

B. W. Silverman. Some Aspects of the Spline Smoothing Approach to Non-parametric Regression Curve Fitting (with discussion). *Journal of Royal Statistical Society B*, 47(1): 1–52, 1985.

D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Blackwell, 2006. ISBN 978-0471708582.

G. L. Smith, S. F. Schmidt, and L. A. McGee. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. Technical report, NASA Ames Research Center: Technical Report R-135, 1962.

A. J. Smola and P. Bartlett. Sparse Greedy Gaussian Process Regression. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

S.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. 148(5):1042–1045, 1963.

E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.

T. Söderström and P. Stoica. *System Identification*. Prentice Hall, 1989.

A. Solin and S. Särkkä. State space methods for efficient inference in Student-t process regression. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2015)*, volume 38 of *JMLR Workshop and Conference Proceedings*, pages 885–893, 2015.

H. W. Sorenson. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum*, pages 63–68, 1970.

J. Steinbring and U. D. Hanebeck. S2KF : The Smart Sampling Kalman Filter. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 2089–2096, 2013.

O. Straka, J. Duník, M. Šimandl, and E. Blasch. Comparison of Adaptive and Randomized Unscented Kalman Filter Algorithms. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–8, 2014.

V. Tresp. A Bayesian Commitee Machine. *Neural Computation*, 12:2719–2741, 2000.

F. Tronarp, R. Hostettler, and S. Särkkä. Sigma-point Filtering for Nonlinear Systems with Non-additive Heavy-tailed Noise. In *19th International Conference on Information Fusion*, pages 1859–1866, July 2016.

R. Turner and C. E. Rasmussen. Model Based Learning of Sigma Points in Unscented Kalman Filtering. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, pages 178–183, 2010. doi: 10.1109/MLSP.2010.5589003.

R. E. Turner and M. Sahani. Time-Frequency Analysis as Probabilistic Inference. *IEEE Transactions on Signal Processing*, 62(23):6171–6183, 2014. doi: 10.1109/TSP.2014.2362100.

R. Van der Merwe and E. Wan. Efficient Derivative-Free Kalman Filters for Online Learning. In *European Symposium on Artificial Neural Networks, ESANN*, number April, pages 205–210, 2001.

R. Van der Merwe and E. Wan. Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models. In *Proceedings of the Workshop on Advances in Machine Learning*, 2003.

M. Šimandl and J. Duník. Derivative-free estimation methods: New results and performance analysis. *Automatica*, 45(7):1749–1757, 2009. doi: 10.1016/j.automatica.2009.03.008.

M. Šimandl, L. Král, and P. Hering. Neural network based bicriterial dual control of nonlinear systems. In *Proceedings of the 16th IFAC World Congress*. Prague: IFAC, 2005.

G. Wahba. Finite-Dimensional Approximating Subspaces. In *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics, chapter 7, pages 95–99. SIAM, 1990. doi: dx.doi.org/10.1137/1.9781611970128.ch7.

L. Wasserman. *All of Nonparametric Statistics*. Springer, 2007. ISBN 978-0387251455.

H. L. Wei and S. A. Billings. A unified wavelet-based modelling framework for non-linear system identification: the wanarx model structure. *International Journal of Control*, 77(4):351–366, 2004. doi: 10.1080/00207170420001976 22.

C. K. I. Williams and M. Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

A. G. Wilson. *Covariance Kernels for Automatic Pattern Discovery and Extrapolation with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

R. P. Wishner, J. A. Tabaczynski, and M. Athans. A comparison of three non-linear filters. *Automatica*, 5(4):487–496, 1969.

Y. Wu, D. Hu, M. Wu, and X. Hu. A Numerical-Integration Perspective on Gaussian Filters. *IEEE Transactions on Signal Processing*, 54(8):2910–2921, 2006.

Q. M. Zhu and A. Billings. Parameter estimation for stochastic nonlinear rational models. *International Journal of Control*, 57(2):309–333, 1993. doi: 10.1080/00207179308934390.