Gaussian processes for online system identification and control of a quadrotor

Sverre Knutsen

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6085
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

The aim of this master thesis was to develop adaptive control for a quadrotor using Gaussian process regression (GP). Online regression with GPs can provide many benefits as it is a flexible non-linear regression model that can provide uncertainty measures of the estimate. However, online GP regression may also gives rise to problems. Because of the high numerical complexity of GPs, sparse approximations are necessary. To this end, the algorithm Sparse Online Gaussian Processes (SOGP) was used. Adaptive control, however, requires the ability to estimate time-varying functions which SOGP does not provide any obvious solution to. To provide time-varying estimations, a Kalman filter interpretation of the update equations in SOGP was used. This addition to the classical SOGP algorithm provided good adaption to non-stationary disturbances at the same time as it was lowering the numerical complexity compared to previously proposed algorithms for non-stationary regression with SOGP. Further, the developed time-varying SOGP regression can provide valuable estimates even under the lack of persistence of excitation, which can be a great advantage in adaptive control. The SOGP algorithm was applied to estimate a model error between a nominal model of the quadrotor and the observed dynamics. The estimated model error could then be used in model-based controllers to improve performance under uncertainties in the dynamics. Two strategies were tested: Model Reference Control and Model Predictive Control. Through these two control strategies, modeled uncertainties in the form of flap- and drag-induced effects, as well as parametric uncertainties, could be estimated online for better flight control. The performance of the proposed controllers was showed through simulations. Even if it was the aim of the thesis no real-time experiments were performed due to lack of time.

Acknowledgements

I want to thank my supervisor Marcus Greiff for enthusiastic encouragements and valuable advises, without whom this thesis would have been hard to finish. The engagement from his side has exceeded all expectations. During the course of the writing process, he has also provided very help-full advises that have improved the report substantially. I would also like to express gratitude to my main supervisor Anders Robertsson for enthusiastic help and discussion of problems during the way. Finally, I want to thank Malin Alsved for her support and understanding during times of intensive work.

Contents

1

Introduction

The quadrotor is a non-linear and unstable control system, which poses interesting and challenging control problems whose solutions typically generalize to many other aerial vehicles. Adaptive control for aerial vehicles is often of high relevance because of various disturbances such as wind and mass loads that enter the dynamics with complex state variable relations. Another source of uncertainties can originate from un-modeled dynamics in the form of permanent or temporary faults in the system. Since the dynamics are non-linear, an identification process that can estimate non-linear functions is beneficial. Parametric models could be designed to account for known uncertainties. While this might work well for some uncertainties, there is no guaranty that other uncertainties with unknown structures can be captured. If more general models are used, one might get problems with identifiability, because of the typically low order of excitation of the system. This is especially a problem in online system identification where model validation is not possible. Gaussian processes have been reported to handle situations with low excitation because of their ability to scale the regression problem with available information [Maciejowski and Yang, 2013]. With Gaussian processes' flexibility to estimate a large set of unknown but smooth non-linear functions, they could be a good candidate for non-linear adaptive control.

Gaussian processes (GP) is a nonparametric Bayesian method for the estimation of unknown functions. The theory of Gaussian processes is not new but the area has gotten new attention during the last decades. It is now a well-studied method in the Machine Learning field but has also shown promising results in other fields. This thesis will explore the use of Gaussian processes for online system identification and control of a quadrotor. The usage of GPs in the control community is rather unexplored but has recently gained more attention [Miao et al., 2018].

A GP can be seen as a distribution over functions which specifies a mean function $m(x)$ and a covariance function $k(x, x')$ as $f(x) \sim \mathcal{GP}\left(m(x), k(x, x')\right)$ of the approximated function $f(x)$. An observation $y = f(x) + \epsilon$ of the function $f(x)$ with some noise ϵ can then be used as known points in the function domain to predict $f(x)$ in the rest of the domain. The method's complexity scales as more observations are collected by adding more regressors. The GP model is therefore not determined beforehand. This is why the method is called nonparametric. However, there are still parameters to be chosen. These parameters are often called hyperparameters and define the covariance function (kernel function) $k(x, x')$ [Rasmussen and Williams, 2006]. It is, however, possible to update the hyperparameters online which could be a way of increasing the method's ability to adapting to new data.

The possible advantages of a GP control scheme are:

- Nonparametric identification: This means that the number of parameters is not fixed but grows with the number of observations and complexity. This allows for estimation of functions in a large domain without much prior assumptions and knowledge of the function.
- No assumptions on the linearity of the estimated functions are made, which is useful when dealing with the quadrotor dynamics.
- Incorporation of uncertainty/variance of predictions.

Drawbacks of using GPs in a control scheme:

- Numerical complexity of training scales as $\mathcal{O}(n^3)$, where n is the number of observations.
- The basic GP method cannot handle nonstationary functions.

To use GPs in a control setting, these problems have to be considered and dealt with. The idea of having an online identification of the dynamics is to be able to compensate for changes in the dynamics both stationary (from flight to flight) and non-stationary (changes during flight). Therefore modifications of the algorithm are needed. More advanced GP algorithms have been developed that partly can solve the above mentioned problems [Rasmussen and Williams, 2006] [Maciejowski and Yang, 2013] [Zhang and Luo, 2014]. This thesis will focus on exploring these solutions to enable both adaptive and numerically tractable control using GPs.

To benefit fully from the GP system identifications the control scheme also has to take advantage of the nonlinear nature of the GP. In the literature some different methods has been explored, e.g., MPC (Model Predictive Control) and MRAC (Model Reference Adaptive Control) [Maciejowski and Yang, 2013], [Chowdhary et al., 2013a] [Deisenroth et al., 2015], [Miao et al., 2018].

1.1 Problem formulation and Goals

The main goals of this thesis are:

- Investigation and implementations of different sparse GP methods to lower the numerical complexity.
- Investigate methods to enable non-stationary estimation for adaptivity through some sort of forgetting mechanism, which is closely linked to making the algorithm sparse.
- Find uncertainties in the dynamics of the quadrotor which could be estimated with GPs to improve flight performance.
- Implement a learning scheme where a nominal dynamic model of the quadrotor is known and where the GP is used to improve the dynamics for better flight control.
- Evaluate the method through comparison to the nominal controller, for instance MPC or MRAC, in both experiments and simulation.

1.2 Outline

The thesis will be organized as follows. Chapter 2 will first cover the background of GP regression to explain the foundation on what the sparse approximation used is based on. The sparse algorithm SOGP will then be introduced and a new extension to the algorithm be explained. The rest of Chapter 2 introduces a nominal quadrotor model, that is also extended with models of dynamic uncertainties. The proposed adaptive controllers are then introduced in Chapter 3 and the design of the adaptive element using GP is explained. The adaptive controllers are then tested in simulations in Chapter 4 and discussed in Chapter 5.

1.3 Notations

If nothing else is mentioned, the following notations will be used. Scalars will be denoted with italic font. Vectors will be denoted with lowercase bold symbols and matrices with upper case bold symbols.

Nomenclature

Acronyms

GP	Gaussian Process
GP-MPC	Gaussian Process Model Predictive Control, wherein the adaptive element is a GP
GP-MRAC	Gaussian Process Model Reference Adaptive Control, wherein the adaptive element is a GP.
KL	Kullback-Leibler removal policy for SOGP
KLI Test	Kernel Linear Independence Test. Used for deciding when to add samples to $\mathcal{BV}$
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
MRC	Model Reference Control
OP	Oldest Point removal policy for SOGP
RBF	Radial Basis Function used as kernel function
RHKS	Reproducing Kernel Hilbert Space
SOGP	Sparse Online Gaussian Process

Symbols

$\boldsymbol{q} \in \mathbb{R}^4$	Quaternion in vector form
$\boldsymbol{p} \in \mathbb{R}^3$ [m]	Position in inertial coordiante frame of quadrotor
$\boldsymbol{\eta} \in \mathbb{R}^3$ [rad]	Euler angles stacked in vector
$\boldsymbol{\omega}_{\mathcal{B}} \in \mathbb{R}^3$ [rad/s]	Body attitude rate
$\boldsymbol{\Omega} \in \mathbb{R}^4$ [rad/s]	Angular rate of rotors
m [kg]	Mass of quadrotor
g [m/s^2]	Gravitational acceleration.
$\boldsymbol{I}_{\mathcal{B}} \in \mathbb{R}^{3\times3}$ [kg·m^2]	Moment of inertia of quadrotor in body frame
$\mathcal{D}$	Set of data points added to a GP
$\mathcal{BV}$	Set of basis vectors added to a SOGP

$n_{\mathcal{D}}$	Cardinality of $\mathcal{D}$
$n_{\mathcal{BV}}$	Cardinality of $\mathcal{BV}$
$\boldsymbol{X}$	Stacked measurment locations in the $\mathcal{D}$ set
$\boldsymbol{X}_{\mathcal{BV}}$	Stacked measurment locations in the $\mathcal{BV}$ set
$\boldsymbol{y}$	Column vector of stacked noisy observations in $\mathcal{D}$ or $\mathcal{BV}$
$k(x, x')$	Kernel function
$\boldsymbol{K}$	Gram matrix, $\boldsymbol{K}_{i,j} = k(\boldsymbol{X}_i, \boldsymbol{X}_j)$
$\boldsymbol{k}_{t+1}$	Kernel vector of new input, $\boldsymbol{K}_i = k(\boldsymbol{X}_i, x_{t+1})$
k'	Kernel center value, $k' = k(x_{t+1}, x_{t+1})$
$\boldsymbol{Q}$	Inverse Gram matrix, $\boldsymbol{Q} = \boldsymbol{K}^{-1}$
σ_n^2	Measurment noise variance of observations in GP
σ_p^2	Process noise variance in GP
$\boldsymbol{\sigma}_k$	Kernel width vector of the RBF kernel function
$\boldsymbol{C}$	Inverse of the matrix $\boldsymbol{K} + \boldsymbol{I}\sigma_n$
γ_{t+1}	KLI test measure
$\mathrm{tr}(\boldsymbol{A})$	Trace of matrix $\boldsymbol{A}$
$\mathrm{sign}(a)$	Returns the sign of the number a ($\mathrm{sign}(0) = 1$)
$\hat{\boldsymbol{x}}$	The unit vector $\hat{\boldsymbol{x}} = [1, 0, 0]^T$
$\hat{\boldsymbol{y}}$	The unit vector $\hat{\boldsymbol{y}} = [0, 1, 0]^T$
$\hat{\boldsymbol{z}}$	The unit vector $\hat{\boldsymbol{z}} = [0, 0, 1]^T$

2

Background

2.1 Gaussian Processes

Gaussian processes (GPs) have been used in many fields such as spatial statistics [Ripley, 1991] and machine learning [Rasmussen and Williams, 2006] for a long time. The typical applications in these fields do not have the same demand on computational tractability as when used in online inference. This is the main reason why GP regression has not been more used in the adaptive control field.

There are different ways of looking at how a GP works: the function space view and the feature space view. In the function space view, a GP is thought of as a distribution over functions, specified by a mean and covariance denoted as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \tag{2.1}$$

where $f(\boldsymbol{x})$ is a process described by the mean function $m(\boldsymbol{x})$ and the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. Let the observation location be of dimension $\boldsymbol{x} \in \mathbb{R}^m$. This specifies the prior information of the process. But the primary interest is to use observations of an observed process together with this prior distribution over functions to get a posterior distribution over functions by using Bayes' formula.

The prior mean, $m(\boldsymbol{x})$, is often chosen to 0 and the covariance, $k(\boldsymbol{x}, \boldsymbol{x}')$, can for example be assumed to be Gaussian. Let y be some training observations in points $\mathbf{X} \in \mathbb{R}^{m \times n_\mathcal{D}}$ and let $\mathbf{f}_*$ be some test inputs in $\mathbf{X}_*$. Finally let $K(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{n_\mathcal{D} \times n_\mathcal{D}}$ be a matrix consisting of all training point pairs in $\mathbf{X}$, evaluated by $k(\boldsymbol{x}, \boldsymbol{x}')$ as

$$\boldsymbol{K}(\mathbf{X},\mathbf{Z}) = \begin{bmatrix} k(\boldsymbol{x}_1,\boldsymbol{z}_1) & k(\boldsymbol{x}_1,\boldsymbol{z}_2) & \dots & k(\boldsymbol{x}_1,\boldsymbol{z}_m) \\ k(\boldsymbol{x}_2,\boldsymbol{z}_1) & k(\boldsymbol{x}_2,\boldsymbol{z}_2) & \dots & k(\boldsymbol{x}_2,\boldsymbol{z}_m) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n,\boldsymbol{z}_1) & k(\boldsymbol{x}_n,\boldsymbol{z}_2) & \dots & k(\boldsymbol{x}_n,\boldsymbol{z}_m) \end{bmatrix} \tag{2.2}$$

$$\mathbf{X} = [\boldsymbol{x}_1,\ \boldsymbol{x}_2, \dots, \boldsymbol{x}_n] \tag{2.3}$$

$$\mathbf{Z} = [\boldsymbol{z}_1,\ \boldsymbol{z}_2, \dots, \boldsymbol{z}_m] \tag{2.4}$$

$$\tag{2.5}$$

A joint distribution can then be formed between the training and test points from the prior. Assume we have a noisy observation of the process $y = f(\boldsymbol{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ is white noise. The joint distribution is then

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(m(\mathbf{X}), \begin{bmatrix} K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & K(\mathbf{X},\mathbf{X}_*) \\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix} \right) \tag{2.6}$$

By conditioning on the stacked observations $\boldsymbol{y}$ in the joint prior distribution, the following predictive distribution in the test points $\mathbf{X}_*$ is achieved (see [Rasmussen and Williams, 2006])

$$f_* | X, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\mathbf{f}_*, \Sigma_{f_*}) \tag{2.7}$$

$$\mathbf{f}_* = K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y} \tag{2.8}$$

$$\boldsymbol{\Sigma}_{f_*} = K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}K(\mathbf{X},\mathbf{X}_*) \tag{2.9}$$

Here $\mathbf{f}_*$ is the posterior mean of the process and $\boldsymbol{\Sigma}_{f_*}$ is the posterior covariance of the posterior distribution.

The exact same result can be derived through the feature space view. This way of looking at GPs is good for understanding the role of the prior covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$, which can also be called kernel function for reasons that will become clear later. The idea is to first project the observation of the process into some feature space or function space which is spanned by the feature function $\boldsymbol{\phi}(\boldsymbol{x})$. In the one-dimensional case this function could for example be $\boldsymbol{\phi}(x) = [1, x, x^2, x^3\ \dots\ x^n]^T$ or many exponential functions centered in different locations in space. One can then adopt the following model

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T\mathbf{w} \tag{2.10}$$

$$y = f(\boldsymbol{x}) + \epsilon, \tag{2.11}$$

that can be used in a regression model.

Now, assume that a set of training sample pairs $\mathbf{X}$, $\mathbf{y}$ are available. The aim will be to predict the value of f in a new test point $\boldsymbol{x}_*$. To this end we introduce the following notations for convenience: $f_* = f(\boldsymbol{x}_*)$, $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\boldsymbol{x}_*)$ and let $\boldsymbol{\Phi}$ be the stacked $\boldsymbol{\phi}(\boldsymbol{x})$ vectors of the training observations $\mathbf{X}$. Then by assuming a Gaussian prior on the weights $\mathbf{w} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_p)$, a Bayesian update can be preformed to give a posterior distribution of the weights $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ as [Rasmussen and Williams, 2006, p. 9]

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})d\mathbf{w}} \tag{2.12}$$

This will result in a predictive distribution of the weights

$$\begin{aligned} p(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto \exp(-\frac{1}{2\sigma_n^2}(\boldsymbol{y} - \boldsymbol{\Phi}^{\boldsymbol{T}}\mathbf{w})^T(\boldsymbol{y} - \boldsymbol{\Phi}^{\boldsymbol{T}}\mathbf{w}))\exp(-\frac{1}{2}\mathbf{w}^T\boldsymbol{\Sigma}_p^{-1}\mathbf{w}) && (2.13)\\ &\propto \exp(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^T(\frac{1}{\sigma_n^2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Sigma}_p^{-1})(\mathbf{w} - \bar{\mathbf{w}})) && (2.14) \end{aligned}$$

where $\bar{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Sigma}_p^{-1})^{-1}\boldsymbol{\Phi}\boldsymbol{y}$. Extending this posterior distribution over $\mathbf{w}$ to the test inputs f_*, the distribution $p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ is formed as

$$\begin{aligned} f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(&\boldsymbol{\phi}_*^T\boldsymbol{\Sigma}_p\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\Sigma_p\boldsymbol{\Phi} + \sigma_n I)^{-1}\mathbf{y}, && (2.15)\\ &\boldsymbol{\phi}_*^T\Sigma_p\boldsymbol{\phi}_* - \boldsymbol{\phi}_*^T\boldsymbol{\Sigma}_p\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\Sigma_p\boldsymbol{\Phi} + \sigma_n^2 I)^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_p\boldsymbol{\phi}_*). && (2.16) \end{aligned}$$

By further introducing the notation $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\boldsymbol{\Sigma}_p\phi(\boldsymbol{x}')$ called the kernel function or covariance function it will be shown that we arrive at the same GP equations as in (2.7)–(2.9). This last step is often called the kernel trick and in practice it means that the feature functions, $\phi(\boldsymbol{x})$ is of secondary interest and we instead specify the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$. This can be very convenient if it is harder to work with the feature functions compared to the kernel functions which can be seen by this one-dimensional example. Assume the feature vector $\boldsymbol{\phi}$ is built up of n radial basis functions as $\boldsymbol{\phi}(x) = [\exp(\frac{-(x-c_1)^2}{2\sigma_k^2}), \exp(\frac{-(x-c_i)^2}{2\sigma_k^2}), \ . \ . \ . \ , \exp(\frac{-(x-c_n)^2}{2\sigma_k^2})]^T$, placed over the input space in the feature centres c_i. By then letting n go towards infinity it is possible to prove that this results in the Gaussian kernel [Rasmussen and Williams, 2006]

$$k(x, x') = \phi(x)^T\Sigma_p\phi(x') = \sqrt{\pi}l\sigma_p^2\exp(-\frac{(x - x')^2}{4\sigma_k^2}). \tag{2.17}$$

By now instead using this kernel function $k(x, x')$ and introducing $\boldsymbol{K}(\mathbf{X}, \mathbf{X}) = \boldsymbol{\Phi}^T\Sigma_p\boldsymbol{\Phi}$, $\boldsymbol{K}(\mathbf{X}, \mathbf{X}_*) = \boldsymbol{\Phi}^T\Sigma_p\phi_*$ and $\boldsymbol{K}(\mathbf{X}_*, \mathbf{X}_*) = \phi_*^T\Sigma_p\phi_*$, it is possible to express (2.7)–(2.9) entirely in terms of $k(x, x')$ (see (2.2)). Because the number of kernels in a regression problem is the number of available observations and the features are infinitely many, it is much simpler to work with the kernels compared to the feature space. While the regression problem is reduced to a lower dimension the result is equivalent. Note that the above example uses one-dimensional basis functions. The result can however be generalized to higher dimensions [Rasmussen and Williams, 2006].

At first glance, it can seem strange that the kernel trick can be used to lower the dimension of the regression from the number of features to the number of columns in the gram matrix (number of observations $n_{\mathcal{D}}$). It is possible because we add one kernel for every new observation instead of having infinitely many features for the regression from the start. Hence the regression is scaled with the number of input observations.

The GP framework can be summarized in the following equations

$$\mathbf{f}_*|\boldsymbol{X}, \mathbf{y}, \boldsymbol{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \Sigma_{f_*}) \tag{2.18}$$
$$\bar{\mathbf{f}}_* = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{\alpha}(\boldsymbol{X}, \mathbf{y}) \tag{2.19}$$
$$\Sigma_{f_*} = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{C}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \tag{2.20}$$
$$\boldsymbol{C} = (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \boldsymbol{I}\sigma_n^2)^{-1} \tag{2.21}$$
$$\boldsymbol{\alpha} = \boldsymbol{C}\boldsymbol{y} \tag{2.22}$$

where $\boldsymbol{\alpha}$ is a weight vector over the placed kernel centers. Both $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ only depend on the training set and can hence be pre-calculated in the training phase.

To summarize, GPs can be thought of as a regression with a changing number of regressors depending on how much information is available. This is also the reason it is called non-parametric. The GP still has parameters to be set, the expression only refers to that the kernels are not placed beforehand. A GP regression example for a simple one-dimensional function $f(x) = x^6 + x^3 + 2\sin(4\pi x)$ can be seen in Figure 2.1. For only one observation we can see that the covariance has been lowered around the observation. The width of this area relates to the width parameter σ_k in (2.23). When more samples are added the predictive mean converges towards the actual function f. The regression gets closer to the actual function when the whole interval gets evenly covered with observations. Since the observations are randomly generated over the interval $[0, 1]$ many observations are needed to get good coverage. This shows that intelligently selected observation locations are an advantage.

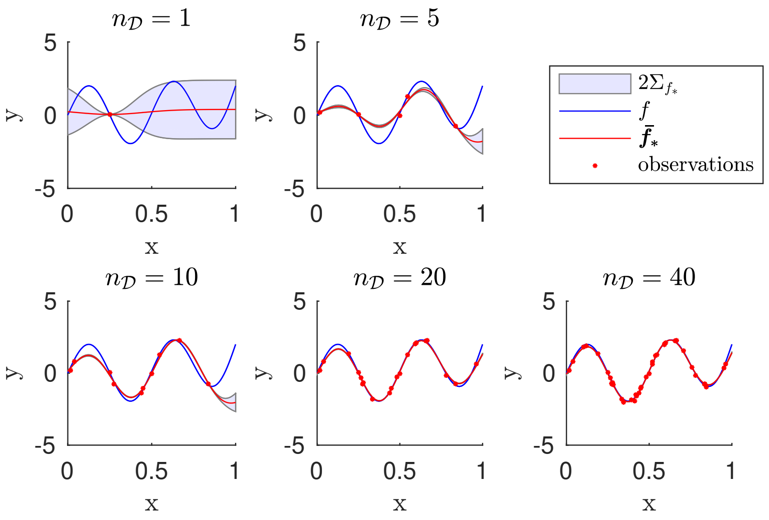

Figure 2.1 GP regression of the function $f(x) = x^6 + x^3 + 2\sin(4\pi x)$ (in blue) with the kernel $\exp(-\frac{(x-x')^2}{4\sigma_k^2})$ for $\sigma_k = 0.2$. The predictive mean of the GP regression can be seen in blue when different number of observations $n_{\mathcal{D}}$ are added. The approximate 95% confidence interval of the predictive covariance Σ_{f_*} is also shown.

Kernel

The parameters to be tuned in a GP are often called hyperparameters. They determine the characteristics of the kernel. The kernel used in this thesis will be the Gaussian kernel. This is because it results in analytic expressions and it encodes local information in a smooth way that can be used to estimate a wide range of smooth functions. The Gaussian kernel or Radial Basis Function (RBF) kernel in m dimensions is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_g^2 \exp\Big(- \frac{(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{\Sigma}_l^{-1} (\boldsymbol{x} - \boldsymbol{x}')}{2} \Big). \tag{2.23}$$

The hyperparameters are here the scaling parameter σ_g and the length scale vector $\boldsymbol{\Sigma}_l = \text{diag}(\sigma_1^2, \sigma_2^2, ..., \sigma_m^2)$. The hyperparameter σ_g is only a gain or normalization parameter of the kernel function encoding the variance of the prior distribution over functions. It is often set to 1. $\boldsymbol{\Sigma}_l$ can be interpreted as the width of the kernel, encoding the covariance of points in input space. Setting σ_i^2 small will make the kernel thin, meaning an observation only will affect points very close in the regression. Equivalently, if the σ_i^2 values are large an observation will affect the estimate in a larger region. The length scale of the RBF kernel will, therefore, have to be set according to the observed function's variation: a function with high-frequency content will call for a short kernel width and vice versa. This can be seen in the GP regression example in Figure 2.2 where the effect of different kernel widths can be seen. When a very thin kernel is chosen the number of observations is not enough to get a good estimation in the whole interval. When the kernel widths are increased the estimation gets better but in the last graph, a too wide kernel is chosen so that the GP can not capture the fast variations in the function. This shows the importance of a well-chosen kernel that fits the function to be estimated.

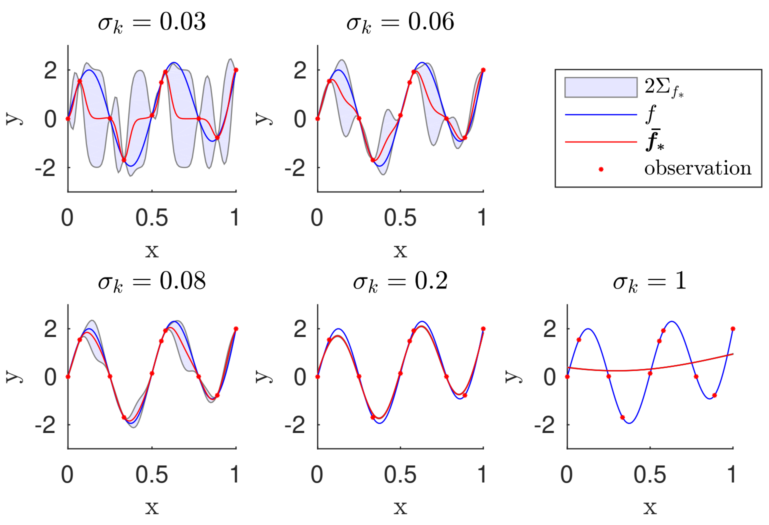

Figure 2.2 GP regression of the function $f(x) = x^6 + x^3 + 2\sin(4\pi x)$ (in blue) with the kernel $\exp(-\frac{(x-x')^2}{4\sigma_k^2})$ for different σ_k. The same observations set has been used in all graphs with the prior mean function $m(x) = 0$.

Note that the kernel function in (2.23) is a multidimensional function. In (2.17) the RBF kernel was introduced as a one-dimensional function. The results can however be extended to the multidimensional case by just replacing the kernel function $k(x, x')$ with the multidimensional kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ [Rasmussen and Williams, 2006].

If the structure of the function is known beforehand more complex kernels can also be chosen. For example, if the function is a sinusoid the kernel could be chosen to a periodic function. This can greatly improve performance since one point now will provide information in a periodic way over the input space. If however, the function is unknown, this is not possible and a more general kernel has to be used. This is where the RBF kernel comes into the picture.

An RBF kernel will only encode local information in a smooth way enabling it to cover a large range of smooth functions. It, however, comes with the downside that more kernel centers have to be collected to get an estimate over a region.

How to choose the kernel hyperparameters is not always trivial. If the function is low dimensional it can be possible to graphically make a guess but since the functions will be largely unknown this can be hard. A possibility is to use hyperparameter optimization to make a better guess. This can be done by maximizing the marginalized log probability of the posterior inference, log $p(\mathbf{y}|X, \mathbf{\Theta})$ with respect to the hyperparameters $\mathbf{\Theta} = [\theta_1, \theta_2, ..., \theta_m]$ [Rasmussen and Williams, 2006, p. 115]. This could either be done online or offline. The log marginal probability is [Rasmussen and Williams, 2006]

$$\log p(\mathbf{y}|X, \mathbf{\Theta}) = -\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \boldsymbol{I}\sigma_n^2)^{-1}\boldsymbol{y} - \frac{1}{2}\log|(\boldsymbol{K} + \boldsymbol{I}\sigma_n^2)| - \frac{n_{\mathcal{D}}}{2}\log 2\pi \tag{2.24}$$

To find the maximum of this probability with respect to the hyperparameters $\mathbf{\Theta}$ by numerical optimization methods, the gradient is needed. The derivative with respect to one hyperparameter θ_j is

$$\frac{\partial}{\partial \Theta_j}\log p(\mathbf{y}|X, \mathbf{\Theta}) = \frac{1}{2}\mathrm{tr}((\boldsymbol{a}\boldsymbol{a}^T - \boldsymbol{K}^{-1})\frac{\partial \boldsymbol{K}}{\partial \Theta_j}) \tag{2.25}$$

$$\boldsymbol{a} = \boldsymbol{K}^{-1}\boldsymbol{y} \tag{2.26}$$

where tr$(\cdot)$ denotes the trace of a matrix. The complexity of this operation is governed by the inverse of $\boldsymbol{K}$ with complexity $\mathcal{O}(n_{\mathcal{D}}^3)$. When the inverse is known, the complexity for calculating the gradient is $\mathcal{O}(n_{\mathcal{D}}^2)$ for each hyperparameter if $\boldsymbol{K}^{-1}$ is assumed to be known [Rasmussen and Williams, 2006]. Any gradient-based optimization technique could be used.

Overfitting

It is sometimes claimed that the Bayesian formalism is immune or robust against overfitting [Ghahramani, 2013, p. 5]. This is easily misunderstood since a too complex model fitted through Bayesian formalism to give a posterior prediction still can be highly overfitted. However, since the prediction is done in a probabilistic manner overfitting will be shown in the uncertainty of the posterior distribution. The model, therefore "knows" how well fitted the data is [Ghahramani, 2013]. For GPs this means that, if a too complex model is chosen, i.e., short length scale in the RBF kernel, this means that more data will be needed to get a posterior uncertainty low enough. If not enough data is found the GP will be overfitted but we will know this by looking at the posterior uncertainty (see Figure 2.2). Also, in the context of control theory, this gives a great advantage since the model uncertainty can give information about overfitting and the possibility to act more cautiously in regions of high uncertainty.

With this information, it is clear that great care has to be taken when choosing hyperparameters. For time-varying functions, we can not guarantee that the kernel actually will have a kernel width that represents the observations at all times. As described in Section **2.1** this could be solved by optimizing the hyperparameters online. However, this introduces another possibility of overfitting. Depending on how much of the state space that is explored, gradient descent of the hyperparameters can be very prone to overfitting if the data are not descriptive enough [Mohammed and Cawley, 2017].

Sparse approximations

The basic GP inference algorithm described above only allows for batch calculation with all data points. This calculation involves taking the inverse of an $n_\mathcal{D} \times n_\mathcal{D}$ matrix which has the complexity $\mathcal{O}(n_\mathcal{D}^3)$. In an online setting, this will become infeasible even for reasonably small batch sizes. Furthermore, the GP will be used in a control scheme where the same point in the state space might be visited many times. If these points are added to the GP, the Gram matrix might lose rank or be close to singular. A solution to these problems must be found. There exist many different techniques to lower this numerical complexity. The common approximation used in all of these methods is that a smaller set of observations are used instead of all observed measurements. If this smaller set of observations contain m observations the complexity is lowered from $\mathcal{O}(n_\mathcal{D}^3)$ to $\mathcal{O}(n_\mathcal{D} m^2)$ where $m << n_\mathcal{D}$ [Snelson and Ghahramani, 2006], [Csató and Opper, 2002]. How these observations are chosen differ between the methods. Optimization is for example used in [Snelson and Ghahramani, 2006] to construct new pseudo inputs representing all observations and the solution in [Csató and Opper, 2002] uses the observations themselves, but chooses them with care. This thesis will focus on the solution given in [Csató and Opper, 2002], called “Sparse Online Gaussian processes” (SOGP), which is more adapted for online inference because of its sequential nature.

Sparse Online Gaussian Processes

The idea of the SOGP algorithm is to recursively update the GP and limit the number of kernels by only adding carefully selected observations to a basis vector set, $\mathcal{BV}$. The observations $(\boldsymbol{X}_{\mathcal{BV}}, \boldsymbol{y}_{\mathcal{BV}})$ in $\mathcal{BV}$ should span the input space in a good way. The rest of the observations, not added to $\mathcal{BV}$, can be used to improve the estimate in the region spanned by $\mathcal{BV}$. The following four steps summarize the algorithm:

1. When a new observation is available, check whether the new sample location can improve the estimate according to some measure.
2. If it is beneficial to add the sample to the basis vector set $\mathcal{BV}$, this is done with a recursive update.
3. If the number of data points $n_{\mathcal{BV}}$ in $\mathcal{BV}$ exceeds a defined budget n_{max} a data point must be deleted.
4. If the new sample was not added to $\mathcal{BV}$ a "sparse update" is performed without increasing the cardinality $|\mathcal{BV}| = n_{\mathcal{BV}}$.

A more detailed explanation of the algorithm will now follow but for a more in-depth explanation of the algorithm and its derivation see [Csató and Opper, 2002] and [Csató, 2002]. The last reference is an more extensive thesis from the same author. The results in the SOGP article [Csató and Opper, 2002] are in some places simplifications of the results in the thesis. The simplified algorithm found in [Csató and Opper, 2002] will in this document mostly be used. However, it would be relevant to investigate the more rigorous SOGP version in the future. Some parts of the proofs given in the full thesis will, however, be used in the walk-through here.

Assume at time instant t, we have a trained GP with data set $\mathcal{D}$ containing $n_{\mathcal{D}}$ observations of the function g. When a new sample is observed it needs to be added to the GP. With the basic GP algorithm, as described in the previous section, this will not be possible without making an inversion of the gram matrix for each new data point. This will become intractable very fast for real-time applications, hence the algorithm needs modification. One way would be to do a rank-1 update of the $\boldsymbol{C}$ matrix for each sample. This has been considered before in literature, see e.g., [Zhang and Luo, 2014],[Csató and Opper, 2002]. This, however, only solves a part of the problem, since a bound on the sample set is needed.

The first step will, however, be to construct a recursive update of the GP equations. To this end, a rank-1 update will be used. For a matrix $\boldsymbol{A} = \boldsymbol{B}^{-1}$ the new inverse when adding a column vector and scalar as

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}^{-1} \tag{2.27}$$

can be found as

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A} + \frac{\boldsymbol{A}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{A}}{c-\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{b}} & \frac{-\boldsymbol{A}\boldsymbol{b}}{c-\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{b}} \\ \frac{-\boldsymbol{b}^T\boldsymbol{A}}{c-\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{b}} & \frac{1}{c-\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{b}} \end{bmatrix} \tag{2.28}$$

This can be verified by using the more general block matrix inversion formula found in [Beal, 2003, p. 262] and Sherman-Morrisons' inversion formula found in [Petersen, 2012].

The inversion formula in (2.28) can be used to update $\boldsymbol{C}$ in (2.21). Fist however we will adopt a seemingly odd notation compared to the GP scheme described earlier. Let $\boldsymbol{C}$ have the opposite sign of the previously defined $\boldsymbol{C}$, making it negative definite. This is adopted from [Csató and Opper, 2002] since it is needed for compatibility with other definitions used later. This means the posterior covariance of the GP now is $\Sigma(\mathbf{x}, \mathbf{x}') = \boldsymbol{K}(\mathbf{x}, \mathbf{x}') + \boldsymbol{K}(\mathbf{X}_{\mathcal{BV}}, \mathbf{x})^T \boldsymbol{C}\boldsymbol{K}(\mathbf{X}_{\mathcal{BV}}, \mathbf{x}')$.

Introducing the notations $\boldsymbol{k}_{t+1} = \boldsymbol{K}(\boldsymbol{X}_{\mathcal{BV}}, \boldsymbol{x}_{t+1})$ and $k' = k(\boldsymbol{x}, \boldsymbol{x})$ for a single new observation $\boldsymbol{x}_{t+1}$ and using (2.28) to update $\boldsymbol{C}$ results in

$$\boldsymbol{C}_{t+1} = \begin{bmatrix} -\boldsymbol{K}_t - \boldsymbol{I}\sigma_n^2 & -\boldsymbol{k}_{t+1} \\ -\boldsymbol{k}_{t+1}^T & -k' - \sigma_n^2 \end{bmatrix}^{-1} \tag{2.29}$$

$$= \begin{bmatrix} \boldsymbol{C}_t - \frac{\boldsymbol{C}_t \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t}{k' + \sigma_n^2 + \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t \boldsymbol{k}_{t+1}} & \frac{-\boldsymbol{C}_t \boldsymbol{k}_{t+1}}{k' + \sigma_n^2 + \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t \boldsymbol{k}_{t+1}} \\ \frac{-\boldsymbol{k}_{t+1}^T \boldsymbol{C}_t}{k' + \sigma_n^2 + \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t \boldsymbol{k}_{t+1}} & \frac{-1}{k' + \sigma_n^2 + \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t \boldsymbol{k}_{t+1}} \end{bmatrix} \tag{2.30}$$

This equation can be used to update $\boldsymbol{C}$ in a recursive manner. The predictive distribution is now defined by the mean and covariance

$$\bar{f}_t(\boldsymbol{x}) = \boldsymbol{\alpha}_t^T \boldsymbol{K}(\mathbf{X}_{\mathcal{BV}}, \mathbf{x}) \tag{2.31}$$

$$\Sigma_t(\mathbf{x}, \mathbf{x}') = \boldsymbol{K}(\mathbf{x}, \mathbf{x}') + \boldsymbol{K}(\mathbf{X}_{\mathcal{BV}}, \mathbf{x})^T \boldsymbol{C}_t \boldsymbol{K}(\mathbf{X}_{\mathcal{BV}}, \mathbf{x}'), \tag{2.32}$$

where the vector $\boldsymbol{\alpha}_t$ and the matrix $\boldsymbol{C}_t$ are recursively updated variables. The recursive update of $\boldsymbol{\alpha}$ can be found by inserting (2.30) into the definition of $\boldsymbol{\alpha}$ in (2.22). The result is summarized as

$$\boldsymbol{\alpha}_{t+1} = T_{t+1}(\boldsymbol{\alpha}_t) + q_{t+1} s_{t+1} \tag{2.33}$$

$$\boldsymbol{C}_{t+1} = U_{t+1}(\boldsymbol{C}_t) + r_{t+1} \boldsymbol{s}_{t+1} \boldsymbol{s}_{t+1}^T \tag{2.34}$$

$$\boldsymbol{s}_{t+1} = T_{t+1}(\boldsymbol{C}_t \boldsymbol{k}_{t+1}) + \boldsymbol{e}_{t+1} \tag{2.35}$$

$$q_{t+1} = (y - \alpha_t^T \boldsymbol{k}_{t+1})/\sigma_x^2 \tag{2.36}$$

$$r_{t+1} = -1/\sigma_x^2 \tag{2.37}$$

$$\sigma_x^2 = \sigma_n^2 + k' + \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t \boldsymbol{k}_{t+1}^T \tag{2.38}$$

where $\boldsymbol{e}_{t+1} \in \mathbb{R}^{t+1 \times 1}$ is a unit vector with element $t+1$ set to 1, T_{t+1} is an operator introduced that extends a vector $\boldsymbol{v} \in \mathbb{R}^{t \times 1}$ to a $t+1$ dimensional vector as

$$T_{t+1}(\boldsymbol{v}) = \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} \tag{2.39}$$

and U_{t+1} is an operator introduced that extends a matrix $\boldsymbol{M} \in \mathbb{R}^{t \times t}$ to a $(t+1) \times (t+1)$ dimensional vector as

$$U_{t+1}(\boldsymbol{M}) = \begin{bmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix}. \tag{2.40}$$

The proof of the recursive update above is an alternative proof using the rank-1 update directly. Another interesting proof is given in [Csató and Opper, 2002] where the GP in the previous iteration is used as a prior distribution that is updated by a Bayesian update with one new observation. The interested reader is referred to [Csató and Opper, 2002] and [Csató, 2002].

Kernel linear independence test Up till now the only change to the GP algorithm is that it is formulated in a recursive manner. This will greatly improve the numerical complexity of adding a sample but we still need some way of limiting the number of samples in the basis vector set $\mathcal{BV}$. To this end, we need a measure of how close the new kernel function will be to the previous kernels. Let us assume that $n_{\mathcal{BV}}$ samples have been added to $\mathcal{BV}$ at certain time t. A kernel function, $k(\boldsymbol{x}, \boldsymbol{x}_{t+1})$, located at some new sample point, $\boldsymbol{x}_{t+1}$, could then possibly be described in the function space spanned by the old kernels as

$$k(\boldsymbol{x}, \boldsymbol{x}_{t+1}) \approx \sum_{i=1}^{n_{\mathcal{BV}}} a_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{2.41}$$

This is, as a linear combination of the old kernels. The weighting $\boldsymbol{a}_{t+1} = [a_1, a_2, \ . \ . \ . , a_{n_{\mathcal{BV}}}]^T$ would have to be found by some minimization and represents the projection of $k(\boldsymbol{x}, \boldsymbol{x}_{t+1})$ into the feature space spanned by the old kernels. This approximation would be good if the new sample is located at a point where old samples have been collected and added to $\mathcal{BV}$. The minimization that finds the projection is

$$\gamma_{t+1} = \min_{\boldsymbol{a}_{t+1}} \left\| k(\cdot, \boldsymbol{x}_{t+1}) - \sum_{i=1}^{n_{\mathcal{BV}}} a_i k(\cdot, \boldsymbol{x}_i) \right\|^2_{\mathcal{RHKS}}. \tag{2.42}$$

Here γ_{t+1} would represent the error between the new feature function $k(\boldsymbol{x}, \boldsymbol{x}_{t+1})$ and this feature function projected down on the space spanned by all the old feature functions. Following [Csató and Opper, 2002] we use the “reproducing kernel Hilbert space” norm ($\mathcal{RHKS}$) to denote inner product as

$$\langle g(\cdot), h(\cdot) \rangle_{\mathcal{RHKS}} = \sum_{i,j} c_i c_j k(\boldsymbol{u}_i, \boldsymbol{v}_i) \tag{2.43}$$

where g and h are two functions $g(\boldsymbol{x}) = \sum_i c_i k(\boldsymbol{x}, \boldsymbol{u}_i)$ and $h(\boldsymbol{x}) = \sum_i d_i k(\boldsymbol{x}, \boldsymbol{v}_i)$. Using this, the Least-Squares (LS) minimization in (2.42) can be written as

$$\min_{\boldsymbol{a}_{t+1}} k(x_{t+1}, x_{t+1}) + \boldsymbol{a}_{t+1}^T \boldsymbol{K} \boldsymbol{a}_{t+1} - 2\boldsymbol{a}_{t+1}^T \boldsymbol{k}_{t+1} \tag{2.44}$$

where $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}_{\mathcal{BV}}, \boldsymbol{X}_{\mathcal{BV}})$. The minimum is easily found by differentiating the expression and finding the stationary point since $\boldsymbol{K}$ is a positive definite matrix. The minimum is

$$\gamma_{t+1} = k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) - \boldsymbol{k}_{t+1}^T \boldsymbol{K}^{-1} \boldsymbol{k}_{t+1} \tag{2.45}$$

with the solution

$$\boldsymbol{a}_{t+1} = \boldsymbol{K}^{-1} \boldsymbol{k}_{t+1} = \boldsymbol{Q} \boldsymbol{k}_{t+1} \tag{2.46}$$

Here the inverse Gram matrix, $\boldsymbol{Q}$, will be expensive to compute. A better idea is to use the same rank-1 update as used for the $\boldsymbol{C}$ matrix in (2.28). This results in

$$\boldsymbol{Q}_{t+1} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k}_{t+1} \\ \boldsymbol{k}_{t+1} & k_{t+1}^* \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{Q}_t + \gamma_{t+1}^{-1} \boldsymbol{a}_{t+1} \boldsymbol{a}_{t+1}^T & -\gamma_{t+1}^{-1} \boldsymbol{a}_{t+1} \\ -\gamma_{t+1}^{-1} \boldsymbol{a}_{t+1}^T & \gamma_{t+1}^{-1} \end{bmatrix} \tag{2.47}$$

or expressed with the operator T_{t+1} and U_{t+1} as

$$\boldsymbol{Q}_{t+1} = U_{t+1}(\boldsymbol{Q}_t) + \gamma_{t+1}^{-1}(T_{t+1}(\boldsymbol{a}_{t+1}) - \boldsymbol{e}_{t+1})(T_{t+1}(\boldsymbol{a}_{t+1}) - \boldsymbol{e}_{t+1})^T. \tag{2.48}$$

The minimum of the optimization cost, γ_{t+1}, in (2.45) can be interpreted as the quadratic error between the projected kernel function and the actual one. This measure could, therefore, be used to decide whether a new kernel center in this observation point should be added to the GP. The set of recursively added kernel centers is denoted $\mathcal{BV}$ and called the basis vector set. This name originates from the fact that the kernel centers span the input space in the GP regression. Let the following inequality determine if a new sample should be added to $\mathcal{BV}$

$$\begin{aligned} \gamma_{t+1} &> \epsilon_{tol} \text{ add sample to } \mathcal{BV} \\ \gamma_{t+1} &\leq \epsilon_{tol} \text{ do not add sample to } \mathcal{BV} \end{aligned}$$

This will be referred to as the Kernel Linear Independence Test or KLI test. The tolerance ϵ_{tol} is a number $k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) \geq \epsilon_{tol} > 0$ that is user specified. Another interpretation of this is possible by noting that $\gamma_{t+1} = k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) - \boldsymbol{k}_{t+1}^T K^{-1} \boldsymbol{k}_{t+1}$ actually is the posterior covariance of a GP containing the $\mathcal{BV}$ set without measurement noise, evaluated in the new point (see (2.20)). Since the covariance is limited by the kernel function gain $k(\boldsymbol{x}, \boldsymbol{x})$, this limits the value of γ_t.

Deleting equations With the KLI Test, the size of the active set of kernel functions $\mathcal{BV}$ will be reduced compared to $\mathcal{D}$ but the problem of unbounded data is not solved. To this end, we will define a "budget" that cannot be exceeded. The budget will be defined as $|\mathcal{BV}| = n_{\mathcal{BV}} \leq n_{max}$.

The budget, however, introduces a new problem. What happens when the budget is exceeded? We do not want to stop learning from the observations. To solve this problem, samples have to be deleted when this happens. The question is then, which sample should be deleted and how is this performed.

The deletion of a sample in $\mathcal{BV}$ can be done in a similar way to the rank-1 update but reversed. The equations for this will be presented but the derivations will be left out. To make deletion possible we define partitions of $\boldsymbol{\alpha}$,$\boldsymbol{C}$ and $\boldsymbol{Q}$ as

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}^l \\ \alpha^r \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}^l & \boldsymbol{c}^r \\ \boldsymbol{c}^{rT} & c^r \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}^l & \boldsymbol{q}^r \\ \boldsymbol{q}^{rT} & q^r \end{bmatrix}. \tag{2.49}$$

where $\boldsymbol{C}^l$ and $\boldsymbol{Q}^l$ are $t \times t$ sub-matrices extracted from the $(t+1) \times (t+1)$ dimensional matrices $\boldsymbol{C}$ and $\boldsymbol{Q}$. These partitions are for simplicity showed for the case when the last sample is removed. It is, however, possible to extract an arbitrary sample in the same way. The deletion equations can be formulated as

$$\boldsymbol{\alpha}_{t-1} = \boldsymbol{\alpha}^l - \alpha^r \frac{\boldsymbol{q}^r}{q^r} \tag{2.50}$$

$$\boldsymbol{C}_{t-1} = \boldsymbol{C}^l + c^r \frac{\boldsymbol{q}^r \boldsymbol{q}^{rT}}{q^{r2}} - \frac{1}{q^r}[\boldsymbol{q}^r \boldsymbol{c}^{rT} + \boldsymbol{c}^r \boldsymbol{q}^{rT}] \tag{2.51}$$

$$\boldsymbol{Q}_{t-1} = \boldsymbol{Q}^l - \frac{\boldsymbol{q}^r \boldsymbol{q}^{rT}}{q^r}. \tag{2.52}$$

The deletion-equation for $\boldsymbol{C}$ is a bit more complex than for $\boldsymbol{Q}$ and $\boldsymbol{\alpha}$. The equation is an approximation that is derived through optimization of the KL divergence between the GP before and after the deletion [Csató, 2002].

Sparse update When the KLI test suggests that a new sample should not be added to $\mathcal{BV}$, a fruitful question would be whether it is possible to use the sample in some other way to improve the certainty of the estimate. This would imply that the GP is updated without extending the dimensionality of the $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ vectors. To do this one could first try to add the sample to $\mathcal{BV}$ and then delete the sample with the deleting equations. Since the deleting equation for $\boldsymbol{C}$ is derived from a KL-divergence optimization this should yield an approximation that is as close as possible to the recursively updated GP with respect to some cost function. Inserting (2.30) into (2.51) results in

$$\hat{\boldsymbol{C}}_{t+1} = \boldsymbol{C}_t - \frac{\boldsymbol{C}_t \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t}{\sigma_x^2} - \frac{\boldsymbol{Q}_t \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T \boldsymbol{Q}_t}{\sigma_x^2} \tag{2.53}$$

$$- \frac{\boldsymbol{Q}_t \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t}{\sigma_x^2} - \frac{\boldsymbol{C}_t \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^T \boldsymbol{Q}_t}{\sigma_x^2} \tag{2.54}$$

To denote that the new matrix $\boldsymbol{C}$, is an approximation the notation $\hat{\boldsymbol{C}}$ is used. The expression can be simplified to

$$\hat{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_t + q_{t+1} \hat{\boldsymbol{s}}_{t+1} \tag{2.55}$$

$$\hat{\boldsymbol{C}}_{t+1} = \boldsymbol{C}_t + r_{t+1} \hat{\boldsymbol{s}}_{t+1} \hat{\boldsymbol{s}}_{t+1}^T \tag{2.56}$$

$$\hat{\boldsymbol{s}}_{t+1} = \boldsymbol{C}_t \boldsymbol{k}_{t+1} + \boldsymbol{a}_{t+1} \tag{2.57}$$

This update cannot be done if the new input location is not close enough to the old kernel centers in $\mathcal{BV}$. But since the update is only done if the KLI test $\gamma_{t+1} < \epsilon_{tol}$ is fulfilled it is ensured that the approximation is good enough. What is "good enough" is determined by the user by setting ϵ_{tol}.

Deleting policy It remains to decide which sample to delete when the budget is exceeded. The problem can be solved in several ways. An appealing way of doing this when dealing with time-varying functions is to delete the oldest point (referred to as OP). In the SOGP article [Csató and Opper, 2002], another strategy is used. Instead, the kernel center contributing least to the predictive mean is removed. This requires some calculation but the computational cost is only linear in the number of data points in $\mathcal{BV}$. The idea is to calculate the difference in the predictive mean when adding a new sample compared to only performing an approximate/sparse update, i.e

$$\Delta \bar{f}_{t+1}(x) = \bar{f}_{t+1}(x) - \widehat{\bar{f}_{t+1}}(x) \tag{2.58}$$

where $\widehat{\bar{f}_{t+1}}(x)$ is the posterior mean after a sparse update. Summing all the differences for all the data points in $\mathcal{BV}$ the total error for not adding the $t+1$ data point to $\mathcal{BV}$ will be [Csató and Opper, 2002]

$$\epsilon_{t+1} = \sum_{i=1}^{n_{\mathcal{BV}}} |\Delta \bar{f}_{t+1}(\boldsymbol{x}_i)| = |q_{t+1}| \min_{\boldsymbol{a}_{t+1}} \left\| k(\cdot, x_{t+1}) - \sum_{i=1}^{n_{\mathcal{BV}}-1} a_i k(\cdot, x_{t+1}) \right\|^2 \tag{2.59}$$

$$= |q_{t+1}| \gamma_{t+1} \tag{2.60}$$

This would give the score for the latest added sample. To get approximate scores for the other samples lets first note that

$$\epsilon_{t+1} = |q_{t+1}|\gamma_{t+1} = \frac{\boldsymbol{\alpha}_{t+1}(t+1)}{\boldsymbol{Q}_{t+1}(t+1,t+1)} \tag{2.61}$$

where $\boldsymbol{\alpha}_{t+1}(t+1)$ is the element with index $t+1$ in $\boldsymbol{\alpha}_{t+1}$ and $\boldsymbol{Q}_{t+1}(t+1,t+1)$ is the element with index (row,column)$= (t+1,t+1)$ in the matrix $\boldsymbol{Q}_{t+1}$.

If the dependence on the ordering of the samples in the $\mathcal{BV}$ is neglected the score for an arbitrary sample ϵ_i is

$$\epsilon_i = \frac{\boldsymbol{\alpha}_{t+1}(i)}{\boldsymbol{Q}_{t+1}(i,i)}. \tag{2.62}$$

The score ϵ_i will be referred to as the projection induced error for data point i. Following this rule when deleting observations from the $\mathcal{BV}$ set will ensure that the kernels will span the feature space in a better way compared to the OP policy.

The whole SOGP algorithm will now be summarized in a more compact algorithm form, seen in Algorithm 1.

Time-varying SOGP

Work has previously been done on using the SOGP algorithm for uncertainty adaption in an online setting but time-varying uncertainties are rarely considered. Attempts to do this have been done by [Chowdhary et al., 2013b] by introducing a time input to the Gaussian process. This means that kernels that were added a long time ago will be deleted after a while. How fast the old kernel centers will get deleted depends on the length scale of the hyperparameter in the time dimension and the specified budget. Other attempts have also been made with other GP algorithms [Zhang and Luo, 2014] but to my knowledge, this is done by only forgetting kernel centers. This is a suboptimal solution since the idea of the SOGP algorithm is to find kernel centers that cover the function space in a good way without exceeding the budget and then learn with the sparse update over the covered area. Ideally, the centers should be placed and then not moved so that the sparse updates makes the estimate converge. However, if a kernel center is removed information will be lost and the uncertainty will spike in this area. The deletion of a kernel can, therefore, lead to drastic changes in the estimate in a real-time setting. The deletion and reallocation of kernels should, therefore, be seen as a last resort when the budget is exceeded. Another limitation of having time as an input to the SOGP is that more kernel centers must be used because of the extra dimension. Since the method scales as $\mathcal{O}(n_{\mathcal{D}}n_{\mathcal{BV}}^2)$, this is not a good idea.

Algorithm 1: SOGP algorithm

```
1  initialize $\boldsymbol{C}_t$, $\boldsymbol{Q}_t$, $\boldsymbol{X}_{\mathcal{BV}}$
2  while New observation $(\boldsymbol{x}_{t+1},\ y_{t+1})$ available do
3      $\boldsymbol{k}_{t+1} = \boldsymbol{K}(\boldsymbol{X}_{\mathcal{BV}}, \boldsymbol{x}_{t+1})$
4      $k' = k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1})$
5      $\sigma_x^2 = \sigma_n^2 + k' + \boldsymbol{k}_{t+1}^T \boldsymbol{C}_t \boldsymbol{k}_{t+1}$
6      $r_{t+1} = -1/\sigma_x^2$
7      $q_{t+1} = (y_{t+1} - \boldsymbol{\alpha}_t^T \boldsymbol{k}_{t+1})/\sigma_x^2$
8      $\gamma_{k+1} = k' - \boldsymbol{k}_{t+1}^T \boldsymbol{Q}_t \boldsymbol{k}_{t+1}$
9      if $\gamma_{k+1} > \epsilon_{tol}$ then
10         Recursive update
11         $\boldsymbol{X}_{\mathcal{BV}} = [\boldsymbol{X}_{\mathcal{BV}},\ x_{t+1}]$
12         $\boldsymbol{s}_{t+1} = T_{t+1}(\boldsymbol{C}_t \boldsymbol{k}_{t+1}) + \boldsymbol{e}_{t+1}$
13         $\boldsymbol{C}_{t+1} = U_{t+1}(\boldsymbol{C}_t) + r_{t+1} \boldsymbol{s}_{t+1} \boldsymbol{s}_{t+1}^T$
14         $\boldsymbol{l}_{t+1} = T_{t+1}(\boldsymbol{a}_{t+1}) - \boldsymbol{e}_{t+1}$
15         $\boldsymbol{Q}_{t+1} = U_{t+1}(\boldsymbol{Q}_t) + \gamma_{t+1}^{-1} \boldsymbol{l}_{t+1} \boldsymbol{l}_{t+1}^T$
16         $\boldsymbol{\alpha}_{t+1} = T_{t+1}(\boldsymbol{\alpha}_t) + q_{t+1} s_{t+1}$
17         if $n_{\mathcal{BV}} > n_{max}$ then
18             Delete one sample
19             for $i = 1 : n_{\mathcal{BV}}$ do
20                 $\epsilon_i = \frac{\boldsymbol{\alpha}_{t+1}(i)}{\boldsymbol{Q}_{t+1}(i,i)}$
21             end
22             $j = \text{argmin}_i(\epsilon_1,\ \epsilon_2, \ldots \epsilon_{n_{\mathcal{BV}}})$
23             Use the deleting equations in (2.49)–(2.52) to delete
                 sample with index $j$
24         end
25     else
26         Sparse update
27         $\hat{\boldsymbol{s}}_{t+1} = \boldsymbol{C}_t \boldsymbol{k}_{t+1} + \boldsymbol{a}_{t+1}$
28         $\boldsymbol{C}_{t+1} = \boldsymbol{C}_t + r_{t+1} \hat{\boldsymbol{s}}_{t+1} \hat{\boldsymbol{s}}_{t+1}^T$
29         $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + q_{t+1} \hat{\boldsymbol{s}}_{t+1}$
30     end
31 end
```

A reasonable question is then to ask, whether it is possible to change the sparse update to allow for time-varying regression without deleting kernels. To make this possible one can take inspiration from the classical solutions in adaptive control where forgetting factors and Kalman filters have been used traditionally [Åström and Wittenmark, 2013], [Johansson, 1993]. Let us rewrite the sparse update (2.55)–(2.57) in the following way

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \frac{\mathbf{s}_k}{\sigma_n^2 + 1 + \mathbf{k}_{k+1}^T \boldsymbol{C}_k \mathbf{k}_{k+1}} \epsilon_k \tag{2.63}$$

$$\epsilon_k = y_k - \boldsymbol{\alpha}_k^T \mathbf{k}_k \tag{2.64}$$

$$\mathbf{s}_k = (\boldsymbol{C}_k + \boldsymbol{Q})\mathbf{k}_k \tag{2.65}$$

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k - \frac{(\boldsymbol{C}_k + \boldsymbol{Q})\mathbf{k}_k \mathbf{k}_k^T (\boldsymbol{C}_k + \boldsymbol{Q})}{\sigma_n^2 + 1 + \mathbf{k}_{k+1}^T \boldsymbol{C}_k \mathbf{k}_{k+1}}. \tag{2.66}$$

These equations could be compared to a Kalman filter estimation of the weights $\boldsymbol{\alpha}$. By assuming we have noise on the parameters following a multivariate Wiener process as

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \boldsymbol{v}_k \tag{2.67}$$

$$E(\boldsymbol{v}_i \boldsymbol{v}_j^T) = \boldsymbol{R}\delta_{ij},\ \forall i, j \tag{2.68}$$

$$E(\boldsymbol{v}_i) = 0, \tag{2.69}$$

where $E(\cdot)$ denotes the expectation operator and the value of $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ij} = 1$ if $i = j$. Meaning that e_k is an independent noise sequence. Further, let observations of the function $\boldsymbol{\alpha}_k^T \mathbf{k}_k$ be of the form

$$y_k = \boldsymbol{\alpha}_k^T \mathbf{k}_k + e_k \tag{2.70}$$

$$E(e_i e_j) = \sigma_n^2 \delta_{ij},\ \forall i, j \tag{2.71}$$

$$E(e_i) = 0 \tag{2.72}$$

$$E(\boldsymbol{v}_i e_j) = 0. \tag{2.73}$$

where $\mathbf{k}_k$ is a regressor vector of the current input $\boldsymbol{x}_k$. In this case the kernel vector $\mathbf{k}_k = \boldsymbol{K}(\boldsymbol{X}_{\mathcal{BV}}, \boldsymbol{x}_k)$. These assumptions let us define an optimal Kalman filter estimation of $\boldsymbol{\alpha}_k$ as [Johansson, 1993, p. 110]

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \frac{\mathbf{s}_k}{\sigma_n^2 + \mathbf{k}_{k+1}^T \boldsymbol{P}_k \mathbf{k}_{k+1}} \epsilon_k \tag{2.74}$$

$$\epsilon_k = y_k - \alpha_k^T \mathbf{k}_k \tag{2.75}$$

$$\mathbf{s}_k = \boldsymbol{P}_k \mathbf{k}_k \tag{2.76}$$

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{\boldsymbol{P}_k \mathbf{k}_k \mathbf{k}_k^T \boldsymbol{P}_k}{\sigma_n^2 + \mathbf{k}_{k+1}^T \boldsymbol{P}_k \mathbf{k}_{k+1}} + \boldsymbol{R}. \tag{2.77}$$

Comparing this to the rewritten sparse update in (2.63)–(2.66) similarities can be seen, but it seems like the covariance matrix $\boldsymbol{P}_k$ does not appear in the same way everywhere. However, it is still possible to interpret (2.63)–(2.66) in the context of the classical Kalman filter. First, we realize that the matrix $\boldsymbol{Q}$ is actually constant during the sparse update. This allows us to chose $\boldsymbol{P}_k = \boldsymbol{C}_k + \boldsymbol{Q}$ but still only use the matrix $\boldsymbol{C}_k$ in the SOGP algorithm. Also, remember the KLI test, $\gamma_k < \epsilon_{tol}$, that determines if a sparse update is performed or not. The expression for γ_{k+1} is

$$\gamma_{k+1} = k' - \mathbf{k}_{k+1}^T Q \mathbf{k}_{k+1}. \tag{2.78}$$

With these observation lets us further rewrite the sparse update as

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \frac{\mathbf{s}_k}{\sigma_n^2 + \gamma_{k+1} + \mathbf{k}_{k+1}^T \boldsymbol{P}_k \mathbf{k}_{k+1}} \epsilon_k \tag{2.79}$$

$$\epsilon_k = y_k - \boldsymbol{\alpha}_k^T \mathbf{k}_k \tag{2.80}$$

$$\mathbf{s}_k = \boldsymbol{P}_k \mathbf{k}_k \tag{2.81}$$

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{\boldsymbol{P}_k \mathbf{k}_k \mathbf{k}_k^T \boldsymbol{P}_k}{\sigma_n^2 + \gamma_{k+1} + \mathbf{k}_{k+1}^T \boldsymbol{P}_k \mathbf{k}_{k+1}} + \boldsymbol{R} \tag{2.82}$$

$$\boldsymbol{P}_k = \boldsymbol{C}_k + \boldsymbol{Q} \tag{2.83}$$

Here the only actual change to the sparse algorithm is to add the process noise covariance matrix $\boldsymbol{R}$. With this last manipulation, the only difference compared to a Kalman filter is that we instead have a time-varying measurement noise $\sigma_n^2 + \gamma_{k+1}$. Since γ_{k+1} is a measure of the quality of the sparse approximation, this can be thought of as increasing the measurement noise if the sparse approximation is bad. However, γ_k is always bounded in the sparse update by $\gamma_k < \epsilon_{tol}$ making the approximation sufficiently good by choosing ϵ_{tol} sufficiently small. Note also that γ_k is the predicted covariance in the noise-free GP.

The only thing left, is now to define a suitable matrix $\boldsymbol{R}$. Some different solutions can here be designed depending on the application of the GP. Here two alternatives will be discussed. The simplest design of $\boldsymbol{R}$ would be a diagonal matrix, $\boldsymbol{R} = \sigma_p^2 \boldsymbol{I}$. Where σ_p^2 is a process noise variance chosen by the user. This would put an equal uncertainty on every kernel so that the uncertainty will increase the same way independent of the current observation. This design can, however, cause problems. Assume that observations are only acquired from a small subset of the input space and that $\boldsymbol{R} = \sigma_p^2 \boldsymbol{I}$ is used. The uncertainty outside this region would then grow unbounded. This could potentially cause robustness issues in control settings.

Instead, let us try to design a matrix $\boldsymbol{R}$ that only increases the covariance locally with respect to the observation location inspired by how a sample is added. To do this one can exploit the matrix structure in the covariance update in (2.77). The structure is $\boldsymbol{A}\mathbf{k}_k \mathbf{k}_k^T \boldsymbol{A}/\sigma_p^2$, for some symmetric positive definite matrix $\boldsymbol{A}$ and some fixed covariance σ_p^2. Let us assume that equal covariance will be put on all kernels according to this structure. Meaning $\boldsymbol{A} = \sigma_p^2 \boldsymbol{I}$. This would result in

$$\boldsymbol{R} = \sigma_p^4 \mathbf{k}_k \mathbf{k}_k^T / \sigma_p^2 = \sigma_p^2 \mathbf{k}_k \mathbf{k}_k^T \tag{2.84}$$

This will increase the covariance for every kernel in such a way that is not proportional to the learning but has the same structure, making the covariance far away from the observation untouched. This can increase the robustness, but it will also limit time-varying learning since you only forget locally. While this works, there are some side effects that has to be taken care off. To see this we can look at the predictive covariance of the posterior distribution $\Sigma = k' + \mathbf{k}_k^T \mathbf{C} \mathbf{k}_k$ (see (2.32)). The effect in the posterior covariance only from the process noise in one Kalman iteration would be $k' + \mathbf{k}_k^T \mathbf{R} \mathbf{k}_k = k' + \sigma_p^2 (\mathbf{k}_k^T \mathbf{k}_k)^2$. The contribution of the Kalman filter update would hence be variant with respect to the observation location. This is not wanted. Instead we define the process noise matrix as

$$\boldsymbol{R} = \sigma_p^2 \frac{\mathbf{k}_k \mathbf{k}_k^T}{(\mathbf{k}_k^T \mathbf{k}_k)^2} \tag{2.85}$$

This would give the contribution of $k' + \sigma_p^2$, which is constant with respect to the observation location. The resulting algorithm for the sparse update in (2.82) with (2.85) will be referred to as Time-varying SOGP (TV-SOGP) when used in the SOGP algorithm.

A solution where a forgetting factor was used was also tried. However, this was not performing as good and no good interpretation was found. Further, a Kalman filter will be more robust against low PE since the dynamics of $\boldsymbol{C}_k$ will grow linearly if $\boldsymbol{k}_{t+1} = \mathbf{0}$ in a Kalman filter. With a forgetting factor in the same situation, the growth would be exponential.

Stationary predictive covariance

It can be useful to know what the minimum stationary predictive covariance $\Sigma = k' + \mathbf{k}_k^T \mathbf{C} \mathbf{k}_k$ will be when using TV-SOGP. To analytically calculate this consider a TV-SOGP where only one observation $\boldsymbol{x}$ has been added and where many noisy observations in the same point in space are observed afterward. This will result in $\mathbf{k}_k = 1$ and $\mathbf{Q}_k = 1$ at all observations in $\boldsymbol{x}$. The stationary covariance will fulfil $\mathbf{C}_k = \mathbf{C}_{k+1}$. From this and (2.82) we have

$$\sigma_p^2 (\sigma_n^2 + 1 + \mathbf{k}_{k+1}^T C \mathbf{k}_{k+1}) = C \mathbf{k}_k \mathbf{k}_k^T C. \tag{2.86}$$

Using $\Sigma = k' + \mathbf{k}_k^T \mathbf{C} \mathbf{k}_k$ and simplifying, the following second order equation is found

$$\Sigma_{min}^2 - \sigma_p^2 \Sigma_{min} - \sigma_n^2 \sigma_p^2 = 0. \tag{2.87}$$

The positive solution is hence the minimum stationary covariance achievable when using the proposed process noise in (2.85).

Online hyperparameter optimization

The choice of hyperparameters is of high importance to achieve good performance in the identification of dynamic systems. Failing to do this will result in over- or under-fitting the data. If the underlying function change in an online identification there might suddenly exist parameters that are better suited. Therefore online hyperparameter optimization would be favorable. However, there are many problems in doing this. First of all the complexity of the gradient descent is high since the inverse of the gram matrix has to be recomputed in each iteration. Another problem is that the optimization problem is non-convex and could have many local optima. Even if the optimization finds the global optima there are no guarantees of the data being exciting enough to find a relevant solution.

Online adaption of hyperparameters in the SOGP algorithm has been tried before by [Grande et al., 2013]. They used stochastic gradient descent to adapt the hyperparameters and if the parameters are changed enough the hyperparameters are updated and the matrices $\boldsymbol{C}$, $\boldsymbol{Q}$ and the weights $\boldsymbol{\alpha}$ are reinitialized. In the process of doing so, information will be lost but with the hope that the estimate might get better in the future. The stochastic gradient descent consists of using the kernel centers in the SOGP algorithm but in every iteration include the current observation k in the $\mathcal{BV}$ set, temporarily, to make a gradient descent of the logarithmic marginal probability (2.24).

Since it would be desirable to avoid reinitializing $\boldsymbol{\alpha}$, the Kalman filter interpretation in TV-SOGP could possibly be very useful here, since it potentially could change the estimate in a more smooth way over time. The matrices $\boldsymbol{Q}$ and $\boldsymbol{C}$ is still a problem, however. These matrices need to be reinitialized when changing the hyperparameters. Further, the matrix $\boldsymbol{C}$ that is needed in the gradient calculation calls for one matrix inversion in every iteration. An interesting idea that has been investigated during this thesis was to use recursive updates of the matrices $\boldsymbol{Q}$ and $\boldsymbol{C}$. The hope was that this could reduce the numerical complexity. A recursive inversion of a matrix can be done if a close approximation of a matrix exists as an initialization by using approximative Newton gradient descent. This method is often called Hotelling-Bodewig algorithm [Soleymani, 2012] and is found by minimizing the strictly convex cost function

$$J(\boldsymbol{Q}) = \frac{1}{2}(\boldsymbol{Q}\boldsymbol{K}(\boldsymbol{\Theta}_k)\boldsymbol{Q} - \boldsymbol{Q}), \tag{2.88}$$

where $\boldsymbol{\Theta}_k$ is the hyperparameter vector at iteration k. The unique optimum to this cost function is $\boldsymbol{Q} = \boldsymbol{K}(\boldsymbol{\Theta}_k)^{-1}$, found at $\nabla_{\boldsymbol{Q}} J(\boldsymbol{Q}) = 0$. The gradient and hessian of $J(\boldsymbol{Q})$ is found as

$$\nabla_{\boldsymbol{Q}} J(\boldsymbol{Q}) = \boldsymbol{K}(\boldsymbol{\Theta}_k)\boldsymbol{Q} - \boldsymbol{I} \tag{2.89}$$

$$\nabla^2_{\boldsymbol{Q}} J(\boldsymbol{Q}) = \boldsymbol{K}(\boldsymbol{\Theta}_k). \tag{2.90}$$

If we assume $\boldsymbol{\Theta}_k \approx \boldsymbol{\Theta}_{k+1}$ the approximation $\boldsymbol{Q}^k \approx \boldsymbol{Q}^{k+1}$ could be done. This would imply that an approximative Newton's method can be used to construct an update of the form

$$\begin{aligned} \boldsymbol{Q}^{k+1} &= \boldsymbol{Q}^k - \nabla^2_{\boldsymbol{Q}} J(\boldsymbol{Q})^{-1} \nabla_{\boldsymbol{Q}} J(\boldsymbol{Q}) && (2.91) \\ &\approx \boldsymbol{Q}^k - \boldsymbol{Q}^k(\boldsymbol{K}(\boldsymbol{\Theta}_k)\boldsymbol{Q}^k - \boldsymbol{I}) && (2.92) \\ &= 2\boldsymbol{Q}^k - \boldsymbol{Q}^k\boldsymbol{K}(\boldsymbol{\Theta}_k)\boldsymbol{Q}^k && (2.93) \end{aligned}$$

This update could be used to iteratively compute both $\boldsymbol{Q}$ and $\boldsymbol{C}$ during the hyperparameter optimization. This method implemented with SOGP can be seen in Figure 2.3. The initial value of the hyperparameter σ_k in the one-dimensional example was chosen too large but after the optimization, the regression can be seen to follow the function much better. The example exaggerates the importance of correct hyperparameter choice to some extent since the lower hyperparameter value in the top graph also allows more kernel centers to be allocated because of the KLI test. The convergence of the length-scale hyperparameter can be seen in Figure 2.4.

While the algorithm works as expected as it finds better hyperparameters, it was found that the computational time with the iterative scheme was not performing as expected compared to a Cholesky factorization. This can be seen in Figure 2.5 where the profiling times of the inversion in the exact same experiment are compared with Cholesky factorization in Matlab. No apparent overhead can be seen with the proposed iterative inversion. The result is actually even worse since the graph shows only one Newton iteration. It was found that at least 5 iterations were needed to give reasonable accuracy. Further, it was also found that the algorithm is sensitive to ill-conditioned Gram matrices originating from low ϵ_{tol} in the KLI test. This was found to be very limiting with respect to the accuracy of the SOGP algorithm.

Using the Cholesky factorization is therefore concluded to be a better alternative. Online hyperparameter optimization will not be considered for quadcopter control. This is because it proves to be hard to find any optimum for the given optimization problem even in an off-line setting for data collected during simulation.

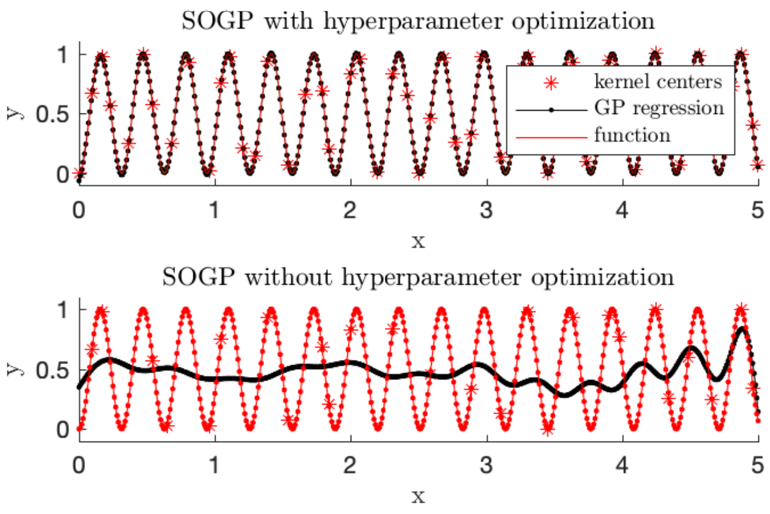

Figure 2.3 GP regression of one-dimensional sinus function with a too broad length scale hyperparameter at the start of the experiment. The top graph shows the SOGP algorithm when the hyperparameter optimizations is used over 10000 samples and the bottom graph shows the same experiment but without optimization.

SOGP for adaptive control

Machine learning strategies often focus on problems where the structure of the estimated function is not known beforehand. This calls for flexible models that can capture different structures. For this purpose, neural networks have lately been drawing much attention. Neural networks, however, have a tendency to overfit the model and need heavy optimization in the training phase. This makes them hard to use in online settings where no model validation exists. Gaussian processes, on the other hand, is a regression technique that has been used by the machine learning community for its ability to adapt to a wide range of unknown nonlinear functions but simultaneously being more robust against overfitting. This makes GPs a good candidate for the online learning of complex nonlinear functions that are not necessarily well known beforehand. This can be a great advantage in many control situations where the system is affected by unknown nonlinear uncertainties. The use of GPs in adaptive control tasks has been investigated before but mainly on stationary uncertainties [Kocijan, 2016].

The fundamental idea of adaptive control is to be able to learn changes in the dynamics of the system and be able to counteract them. Different meth-

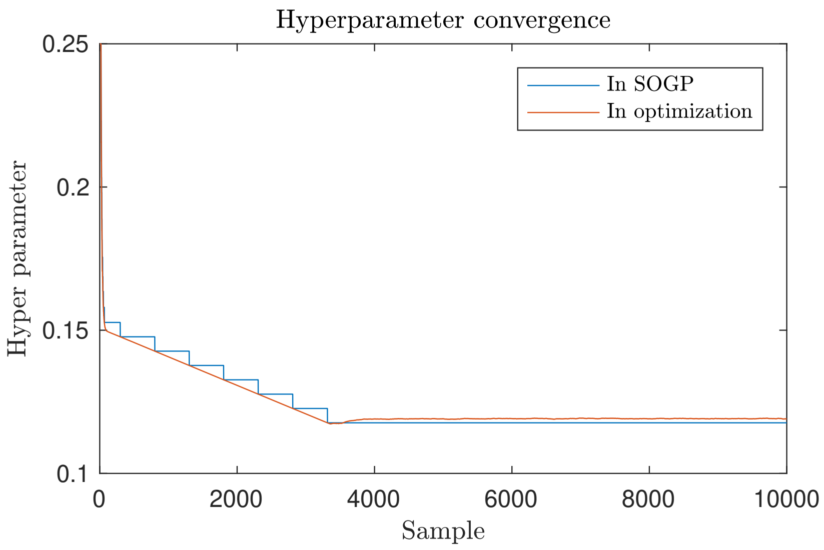

Figure 2.4 The hyperparameter σ_k convergence. The value of σ_k is not changed continuously in the SOGP but changed when the value in the hyperparameter optimization is changed more than 0.01.

ods for stochastic online regression have traditionally been used. Among these Recursive Least Square (RLS) and Kalman filters [Åström and Wittenmark, 2013], [Johansson, 1993]. Both of them incorporate the ability to forget old information and hence be able to capture new changes in the dynamics. A fundamental problem that arises when applying these methods in a control loop is the problem of Persistence of Excitation (PE) [Åström and Wittenmark, 2013]. Since old information is penalized and hence forgotten after a long time, it is of uttermost importance that new information about the system is collected continuously. If this is not the case, the estimate of the dynamical uncertainty may diverge. The adaptive control strategy will then fail and the system may become unstable.

PE is a notion of how much an input excites a dynamical system. A high order of PE in the input signal will give more information about the behavior of the system. This is often not a problem in a system identification context, where the model is to be learned only once and not online. One can simply choose the input signal to be exciting enough. In an online setting, this might not be the case. Control tasks often focus on stabilizing a system to be stationary or slowly varying which is not consistent with high order PE. High order PE typically means we want high spectral content for higher frequencies to estimate the faster dynamics. To tackle the problem of PE different strategies can be used. Either the “forgetting mechanism” has to be turned off

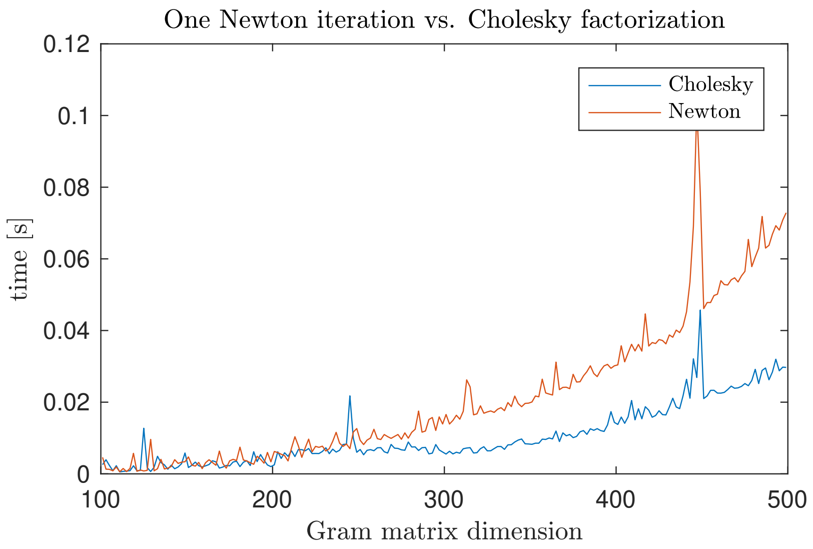

Figure 2.5 One Newton iteration (2.93) compared to Cholesky factorization and inversion for increasing Gram matrix dimension. To achieve good accuracy for the iterative method 5 iterations was at least necessary. One iteration is however used here, rendering the result even worse than in the graph.

when there is a lack of PE, or PE has to be ensured in some way. The problem of enforcing PE at the same time as stabilizing the system is often referred to as Dual control [Alpcan, 2011]. To perform dual control, information about how or what part of the dynamics that have to be excited is needed. In GP regression, which is based on Bayesian formalism, this information is easily accessible in the posterior uncertainty in the distribution over functions. This is not something that the thesis will focus on but something future research in GPs for control task could focus on.

The uncertainty could however also be used in other ways to achieve a more robust adaptive control. Since the uncertainty encodes where the estimate is thought to be less accurate, this could be used to take extra precautions in these regions. Different ways of doing this could be designed depending on which control strategy that is used. In [Grande et al., 2014] a gain was designed to lower the adaptive elements influence on the feedback if the uncertainty is high. This was done for MRAC control. If for example other model-based control strategies such as MPC or LQG are used the penalties could, for example, be chosen differently depending on the uncertainty. When using MPC even more complex solutions such as tube-based MPC could also be used [Cao et al., 2017].

2.2 Quadrotor dynamics

To model the quadrotor rigid body dynamics, the coordinate frame first has to be defined. Let the body frame be centered in the center of mass of the quadrotor. Further let the body frame be rotated and translated with respect to an inertial fixed frame (see Figure 2.6). To this end we define a rotation matrix $\boldsymbol{R}_{\mathcal{I}\mathcal{B}} \in \mathcal{SO}(3)$, rotating a vector from inertial frame to body frame, where $\mathcal{SO}(3) = \{\, \boldsymbol{R} \mid \boldsymbol{R} \in \mathbb{R}^{3\times 3},\, \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I},\, \det(\boldsymbol{R}) = 1 \,\}$ is the three dimensional rotation group. The full transformation of a point in space from inertial frame to body frame would then be

$$x_{\mathcal{B}} = \boldsymbol{R}_{\mathcal{I}\mathcal{B}} x_{\mathcal{I}} - \boldsymbol{p}, \tag{2.94}$$

where $\boldsymbol{p}$ is the position of the body frame in the inertial frame. For future use, we also define the attitude Euler angles $\boldsymbol{\eta}$ that is a parametrization of $\mathcal{SO}(3)$, the attitude rate $\boldsymbol{\omega}_{\mathcal{B}}$ which describes the rotational rate of the quadrotor in body frame and finally the rotational speed of all four rotors $\boldsymbol{\Omega}$ (see Figure 2.7). Let the following notation be used

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad \boldsymbol{s} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\eta} \end{bmatrix}, \quad \boldsymbol{\omega}_{\mathcal{B}} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad \boldsymbol{\Omega} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix}. \tag{2.95}$$

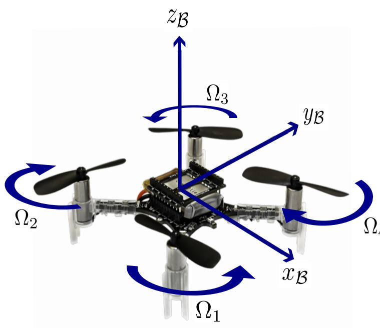

Figure 2.7 Orientation of the body frame in relation to the quadrotor body.

Rotor dynamics

Each individual rotor generates a thrust f_i and a torque τ_{M_i} around its own axis. These forces acting on the quad rotor can be approximated as [Chovancová et al., 2014]

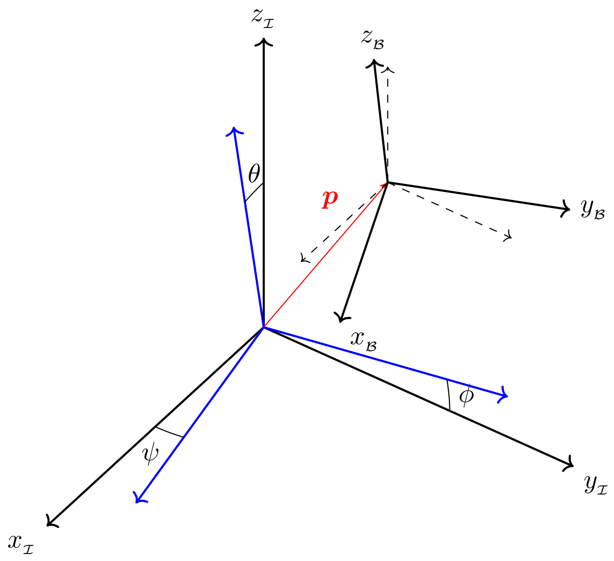

Figure 2.6 Coordinate frame definitions. Inertial fixed frame denoted with $\mathcal{I}$ subscripts and body fixed frame denoted $\mathcal{B}$. The body frame is rotated relative to the inertial frame by the Euler angles ϕ, θ and ψ as described above. Center of mass of the quadrotor is centered in the origin of the body frame.

$$f_i = k_i \Omega_i^2, \tag{2.96}$$

where k_i is some experimentally found constant. The torque τ_{M_i} can be assumed to be proportional to the squared angular rate and the angular acceleration [Chovancová et al., 2014]

$$\tau_{M_i} = j\Omega_i^2 + b\dot{\Omega}_i \approx a\Omega_i^2 \tag{2.97}$$

It can be assumed that the squared angular rate will dominate the expression. Depending on how the body frame is defined in relation to the arms of the quadrotor the combined torque acting on the body will have different expressions. Since the Crazyflie platform used in this thesis has the $x_{\mathcal{B}}$-axis oriented between motor 1 and 4 (see Figure 2.7) the following expression will define the total torques

$$\tau_B = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} kl/\sqrt{2}(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ kl/\sqrt{2}(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \\ a(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix}. \tag{2.98}$$

The total thrust will be

$$T = \sum_i f_i = k(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2). \tag{2.99}$$

Combining these equations a linear relation between the torques and forces on the system and the squared angular rates of the rotors can be formed as

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ T \end{bmatrix} = \begin{bmatrix} -kl/\sqrt{2} & -kl/\sqrt{2} & kl/\sqrt{2} & kl/\sqrt{2} \\ -kl/\sqrt{2} & kl/\sqrt{2} & kl/\sqrt{2} & -kl/\sqrt{2} \\ -a & a & -a & a \\ k & k & k & k \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}. \tag{2.100}$$

The rotors can only spin in one direction, but they have different spinning directions so that the rotor torques will cancel out in steady hover preventing the quadrotor from spinning in the ψ direction.

Euler angles representation

Using Euler angles the rotation from the inertial frame to body frame is done by a sequence of rotations around the inertial coordinate axis (see Figure 2.6). There are many definitions of Euler angles since the order of rotations can be different. The order chosen here is ZYX (ψ-θ-ϕ). The rotations around each axis can be expressed by rotation matrices as

$$\boldsymbol{R}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.101}$$

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.102}$$

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.103}$$

The full rotation sequence taking the inertial frame to the body frame is then $\boldsymbol{R}_{\mathcal{IB}} = \boldsymbol{R}(\phi)\boldsymbol{R}(\theta)\boldsymbol{R}(\psi)$. Since $\boldsymbol{R}_{\mathcal{IB}} \in \mathcal{SO}(3)$ the matrix has the property $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$, implying that the opposite rotation from body to inertial frame is $\boldsymbol{R}_{\mathcal{BI}} = \boldsymbol{R}_{\mathcal{IB}}^T = \boldsymbol{R}(\psi)^T\boldsymbol{R}(\theta)^T\boldsymbol{R}(\phi)^T$.

The model of the quadrotor is obtained from the Euler-Lagrange equations [Castillo et al., 2004]. To this end, define $\boldsymbol{F_I}$ as the external force applied to the quadrotor defined in an inertial frame and let $\boldsymbol{\tau_{\mathcal{B}}}$ be the external torque acting on the quadrotor defined in body frame. The Euler-Lagrange equations are then

$$\begin{bmatrix}\boldsymbol{F}_{\mathcal{I}} \\ \boldsymbol{\tau}_{\mathcal{B}}\end{bmatrix} = \frac{d}{dt}\frac{\partial \mathcal{L}(\boldsymbol{s})}{\partial \dot{\boldsymbol{s}}} - \frac{\partial \mathcal{L}(\boldsymbol{s})}{\partial \boldsymbol{s}} \tag{2.104}$$

Here the Lagrangian $\mathcal{L}(\boldsymbol{s})$ is the sum of kinetic translational, kinetic rotational and potential energy, respectively; $\mathcal{L}(\boldsymbol{s}) = E_{trans} + E_{rot} + E_{pot}$. To this end let the rigid body of the quadrotor be defined by its total mas m and its moment of inertia $\boldsymbol{I}_{\mathcal{B}}$. Further let the gravitational acceleration be denominated g.

The rotational energy requires extra attention. Expressed in the attitude rate $\boldsymbol{\omega}_{\mathcal{B}}$, defined in the body frame, the rotational energy is

$$E_{rot} = \frac{\boldsymbol{\omega}_{\mathcal{B}}^T \boldsymbol{I}_{\mathcal{B}} \boldsymbol{\omega}_{\mathcal{B}}}{2}. \tag{2.105}$$

Since we now want to express it in $\boldsymbol{\eta}$, the mapping between $\dot{\boldsymbol{\eta}}$ and $\boldsymbol{\omega}_{\mathcal{B}}$ has to be found. Let $\boldsymbol{W}(\eta)$ denominate the matrix transformation as $\boldsymbol{\omega}_{\mathcal{B}} = \boldsymbol{W}(\eta)\dot{\boldsymbol{\eta}}$. The rotational energy can then be rewritten as

$$E_{rot} = \frac{\dot{\boldsymbol{\eta}}^T \boldsymbol{W}(\boldsymbol{\eta})^T \boldsymbol{I}_{\mathcal{B}} \boldsymbol{W}(\boldsymbol{\eta}) \dot{\boldsymbol{\eta}}}{2} = \frac{\dot{\boldsymbol{\eta}}^T \boldsymbol{J} \dot{\boldsymbol{\eta}}}{2}. \tag{2.106}$$

The derivation of $\boldsymbol{W}(\eta)$ is left out but can be found in [Greiff, 2017]

$$\boldsymbol{W}(\eta) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\phi)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.107}$$

The full Lagrangian can then be written as [Castillo et al., 2004]

$$\mathcal{L}(\boldsymbol{s}, \dot{\boldsymbol{s}}) = \frac{m\dot{\boldsymbol{p}}^T\dot{\boldsymbol{p}}}{2} + \frac{\dot{\boldsymbol{\eta}}^T \boldsymbol{J} \dot{\boldsymbol{\eta}}}{2} - mg\hat{\boldsymbol{z}}_{\mathcal{I}} \tag{2.108}$$

By simplifying the Euler-Lagrange equations (2.104) and (2.108) the following result is found, separating the angular and positional dynamics

$$m\ddot{\boldsymbol{p}} = \boldsymbol{F}_{\mathcal{I}} - mg\hat{\boldsymbol{z}}_{\mathcal{I}} \tag{2.109}$$

$$\boldsymbol{\tau}_{\mathcal{B}} = \boldsymbol{J}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \dot{\boldsymbol{J}}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\eta}}\left(\dot{\boldsymbol{\eta}}^T \boldsymbol{J}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}\right) \tag{2.110}$$

$$= \boldsymbol{J}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} \tag{2.111}$$

The full expression of the Coriolis matrix, $\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$, is left out. For the full derivation of the equation see [Chovancová et al., 2014], [Castillo et al., 2004], [Luukkonen, 2011] or [Greiff, 2017].

The last step is now to express total force acting on the body in the inertial frame $\boldsymbol{F}_I$ with the Euler angles $\boldsymbol{\eta}$ and the body frame force $\boldsymbol{F}_B$.

$$\boldsymbol{F}_I = \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{\eta})\boldsymbol{F}_B = \boldsymbol{R}_{\mathcal{BI}} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{2.112}$$

Equation (2.109) and (2.111) will then be coupled to form

$$m\ddot{\boldsymbol{p}} = \boldsymbol{R}_{\mathcal{BI}}\boldsymbol{F}_{\mathcal{B}} - mg\hat{\boldsymbol{z}} \tag{2.113}$$

$$\tau_{\mathcal{B}} = \boldsymbol{J}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} \tag{2.114}$$

Quaternion representation

The Euler angle parametrization presented in the last section is a very intuitive way of representing the rotation but it suffers from two major downsides. One is the singularities in the inverse of $\boldsymbol{J}(\boldsymbol{\eta})$ (see (2.106)) resulting from what is often called the Gimbal lock, the other is the trigonometric evaluations needed in the rotation operations [Chovancová et al., 2014]. A better alternative is to use quaternions as a parametrization of the rotational group $\mathcal{SO}(3)$.

Before introducing the quaternion representation the underlying mathematical background will be presented briefly. For a more thorough background see [Sola, 2017]. A quaternion is a hypercomplex number consisting of three imaginary parts and one real scalar part, giving it four degrees of freedom. The quaternion can be represented in different ways, but in this document, we will adopt the vector notation in (2.116) below

$$\boldsymbol{q} = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} \tag{2.115}$$

$$\boldsymbol{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{2.116}$$

The multiplication of two quaternions follows from the distributive law and the underlying complex algebraic identities of the hypercomplex number in (2.115). The multiplication will be denoted with the Kronecker product, denoted with $\otimes$. It is often convenient to represent the quaternion in vector form as (2.116) and divide it into a real part $q_0 = \Re(\boldsymbol{q})$ and an imaginary part $\boldsymbol{q}_v = \Im(\boldsymbol{q}) = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T$. The quaternion product of two quaternions $\boldsymbol{q}$ and $\boldsymbol{r}$ can then be formed as

$$\boldsymbol{q} \otimes \boldsymbol{r} = \begin{bmatrix} q_0 r_0 - \boldsymbol{q}_v^T \boldsymbol{r}_v \\ q_0\boldsymbol{r}_v + r_0\boldsymbol{q}_v + [\boldsymbol{q}_v]_\times \boldsymbol{r}_v \end{bmatrix} \tag{2.117}$$

where $[\cdot]_\times$ is the skew symmetric matrix corresponding to

$$\boldsymbol{v}, \boldsymbol{u} \in \mathbb{R}^3 \tag{2.118}$$

$$\boldsymbol{v} \times \boldsymbol{u} = [\boldsymbol{v}]_\times \boldsymbol{u} \tag{2.119}$$

$$[\boldsymbol{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{2.120}$$

Some more identities will also be needed. The norm of a quaternion is defined as

$$\|\boldsymbol{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{2.121}$$

and the conjugate of a quaternion is defined as

$$\boldsymbol{q}^* = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix}. \tag{2.122}$$

This means that the product between the quaternion $\boldsymbol{q}$ and its conjugate will be

$$\boldsymbol{q}^* \otimes \boldsymbol{q} = \boldsymbol{q} \otimes \boldsymbol{q}^* = \begin{bmatrix} \|\boldsymbol{q}\|^2 \\ \boldsymbol{0} \end{bmatrix}. \tag{2.123}$$

From this equation the derivation of the inverse of a quaternion follows as

$$\boldsymbol{q} \otimes \boldsymbol{q}^{-1} = \boldsymbol{q} \otimes \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|^2} = \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} \tag{2.124}$$

The aim of introducing quaternions is to use it as a parametrization of $\boldsymbol{R} \in \mathcal{SO}(3)$. A rotation of a vector $\boldsymbol{x}_A \in \mathbb{R}^3$ is rotated by the rotation matrix $\boldsymbol{R}$ as

$$\boldsymbol{x}_C = \boldsymbol{R}\boldsymbol{x}_A \tag{2.125}$$

where $\boldsymbol{x}_C$ is the rotated vector in the same coordinate frame. In a quaternion parametrization of $\boldsymbol{R} \in \mathcal{SO}(3)$ a unit quaternion $\boldsymbol{q}$, $\|\boldsymbol{q}\| = 1$ will represent the rotation as

$$\begin{bmatrix} 0 \\ \boldsymbol{x}_C \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{x}_A \end{bmatrix} \otimes \boldsymbol{q}^* \tag{2.126}$$

The transformation between inertia frame and body frame can then be expressed as

$$\boldsymbol{x}_{\mathcal{I}} = \boldsymbol{R}_{\mathcal{BI}} \boldsymbol{x}_{\mathcal{B}} + \boldsymbol{p} \tag{2.127}$$

$$= \Im(\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{x}_{\mathcal{B}} \end{bmatrix} \otimes \boldsymbol{q}^*) + \boldsymbol{p}, \tag{2.128}$$

where $\boldsymbol{p}$ is the position of the origin of the body frame in the inertia frame. By using this expression instead of the Euler angles as parametrization of the rotation from body frame to inertia frame, the rigid body equations can be rewritten. The Euler-Lagrange equation is

$$\mathcal{L}(\boldsymbol{s}, \dot{\boldsymbol{s}}) = \frac{m\dot{\boldsymbol{p}}^T\dot{\boldsymbol{p}}}{2} + \frac{\boldsymbol{\omega}_{\mathcal{B}}\boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}}{2} - mg\boldsymbol{z}_{\mathcal{I}}. \tag{2.129}$$

Together with (2.104) the equation is simplified to

$$\begin{bmatrix} \boldsymbol{F}_{\mathcal{I}} \\ \boldsymbol{\tau}_{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} m\boldsymbol{I} & 0 \\ 0 & \boldsymbol{I}_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{p}}_{\mathcal{B}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ [\boldsymbol{\omega}_{\mathcal{B}}]_{\times}\boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}} \end{bmatrix} \tag{2.130}$$

To find the external force direction in the inertial frame $\boldsymbol{F}_{\mathcal{I}}$ the rotation from $\mathcal{B}$ to $\mathcal{I}$ is used as

$$\boldsymbol{F}_{\mathcal{I}} = \Im\left(\boldsymbol{q} \otimes \boldsymbol{F}_{\mathcal{B}} \otimes \boldsymbol{q}^*\right) \tag{2.131}$$

The differential relationship of the quaternion $\boldsymbol{q}$ and the attitude rate $\boldsymbol{\omega}_{\mathcal{B}}$ can be expressed as [Chovancová et al., 2014]

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{\mathcal{B}} \end{bmatrix} \tag{2.132}$$

This will effectively couple the two rows of the Euler-Lagrange equations in (2.130) forming the full dynamic equations as

$$m\ddot{\boldsymbol{p}} = \Im\left(\boldsymbol{q} \otimes \boldsymbol{F}_{\mathcal{B}} \otimes \boldsymbol{q}^*\right) - mg\hat{\boldsymbol{z}}_{\mathcal{I}} \tag{2.133}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.134}$$

$$\dot{\boldsymbol{\omega}}_{\mathcal{B}} = \boldsymbol{I}_{\mathcal{B}}^{-1}(\boldsymbol{\tau}_{\mathcal{B}} - [\boldsymbol{\omega}_{\mathcal{B}}]_{\times}\boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}) \tag{2.135}$$

Crazyflie parameters

In this thesis, the focus has been to design controllers for the crazyflie 2.0 platform. All simulations will be performed with parameters identified for the Crazyflie. The parameters used have been identified by [Greiff, 2017] and [Landry, 2015] and can be seen in the table below.

Table 2.1 System model parameters for the Crazyflie platform.

Parameter	Value
k	$2.2 \cdot 10^{-1}$ [kg·m/rad^2]
l	0.046 [m]
a	10^{-9} [kg·m^2/rad^2]
$\boldsymbol{I}_{\mathcal{B}}$	$\begin{bmatrix} 2.3951 & 0 & 0 \\ 0 & 2.3951 & 0 \\ 0 & 0 & 3.2347 \end{bmatrix} 10^{-5}$ [kg·m^2]
m	0.027 [kg]

Uncertainty classification

The presented dynamics above do not model many other physical effects such as viscous friction and dynamics of the rotor blades. All these unmodelled dynamics will be referred to as uncertainties with different characteristics. Some commonly used classification of uncertainties will now be explained to enable further discussion.

The model uncertainties can be divided into three different sub-categories: unstructured uncertainties, structured uncertainties and parametric uncertainties [Lee et al., 2001]. The unstructured uncertainties can, for example, be external disturbances/uncertainties with an unknown source. The structured uncertainties, on the other hand, are uncertainties which depend on a known source. The external disturbances such as wind acting on the quadcopter or the so-called near ground effect could be viewed as structured uncertainties if the wind speed and proximity to the ground were known. Since this often is not the case, however, they need to be modelled as unstructured uncertainties. Other internal model uncertainties could also arise if the dynamical system changes over time. This could, for example, be mass or inertia changes due to external loads attached to the quadrotor. Such changes of the modelled dynamics are called parametric uncertainties since they can be modelled as parameter changes in the nominal quadrotor model.

So what uncertainties will be most prominent in the quadrotor dynamics? The answer to this question will, of course, depend on the intended application. Outdoor use will probably be dominated by wind and drag disturbances and indoor use will be dominated by smaller effects in modeling errors but also near ground effects. Commonly discussed uncertainties for quadrotors in the literature are, e.g., near ground effect [Bernard et al., 2017], wind disturbances, and drag/flapping forces [Luukkonen, 2011]. These uncertainties are often neglected in the attitude dynamics. But it can be highly relevant to take them into account in some cases. For example when flying with unsymmetrical loads or under strong wind conditions. But the flight performance could probably be improved by incorporating drag and flapping forces under normal conditions too.

In the two following sections, uncertainties in both the attitude and position dynamics will be discussed more thoroughly.

Uncertainties in quadrotor dynamics

If the Euler angle parametrization is used, the attitude dynamic equation will be (2.114). The attitude dynamics will in this representation contain more parameters than the moment of inertia tensor. This is because the Coriolis matrix $\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ in (2.114) will capture more dynamics that originates from the attitude subsystem (2.134) relating the rotation and attitude rate. However, the differential relation between the rotation and the attitude rate in (2.134) will have no uncertain parameters but is a fundamental trigonometric relationship. This makes the separation in the quaternion parametrization (2.134) and (2.134) better suited for uncertainty approximations compared to (2.114).

The only possible parameter uncertainties in the modeled attitude rate dynamics is therefore the moment of inertia. Other uncertainties will of course be present but we proceed with the assumption that the quadrotor follows the dynamics in eq. (2.135) and that a nominal approximate model of the attitude subsystem is

$$\dot{\boldsymbol{\omega}} = \hat{\boldsymbol{I}}_{\mathcal{B}}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]_{\times}\hat{\boldsymbol{I}}_{\mathcal{B}}\boldsymbol{\omega}) \tag{2.136}$$

Where $\hat{\boldsymbol{I}}_{\mathcal{B}}$ is a close approximation of the true inertial tensor $\boldsymbol{I}_{\mathcal{B}}$.

Other uncertainties in the attitude dynamics would be completely unmodeled in the nominal model. This could be, e.g., effects of wind giving moment forces on the quadrotor, lateral speed giving moment forces, gyroscopic effects and near ground effects. All of the above-mentioned uncertainties would be unstructured with respect to the nominal inputs $\boldsymbol{\omega}$ and $\boldsymbol{\tau}$. They could be modeled as structured if also wind, lateral speed, and the position were incorporated as inputs. Due to numerical complexity, it would not be feasible to have all of these inputs. This is the downside of nonparametric approaches compared to parametric ones. However, this does not mean the GP will not be able to capture any of these uncertainties, only that a stationary solution not will be found when these uncertainties vary. Implying that the adaptation might be able to capture slowly varying uncertainties in the rotor dynamics, like a voltage drop over time or a slow wind gust. This correction will of course not be modeled as a voltage change but rather a change in unstructured disturbances.

There has been done extensive research in the rotor dynamics of rotary wing aircraft in the past. It has mostly focused on single rotor helicopters but work has also been done for the very similar dynamics of multirotor UAVs. See for example [Kai et al., 2017] and [Bangura et al., 2016] for a more in-depth introduction. Since this project does not focus on the exact modeling of these uncertainties the results will only be described briefly to facilitate the analysis of input variables to the adaptive element.

The dynamics presented above for the quadrotor does only take into account the force and torque generated by the rotors from their direct perpendicular thrust. However, in forward motion and under strong wind other aerodynamics effects will produce forces and torques on the UAV. Well-known effects are the gyroscopic torque, the drag and flap induced torques and drag and flapping induced forces. In high velocities, drag from the viscous friction of the UAV body will also affect the dynamics. To summarize, the dynamics could be altered in the following way

$$m\ddot{\boldsymbol{p}} = \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{F}_{\mathcal{B}} - mg\hat{\boldsymbol{z}} + \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{F}_a \tag{2.137}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.138}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{I}_{\mathcal{B}}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]_{\times}\boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega} + \boldsymbol{\tau}_g + \boldsymbol{\tau}_a), \tag{2.139}$$

where $\boldsymbol{F}_a = \boldsymbol{F}_{flap} + \boldsymbol{F}_{drag}$ are the lumped flap and drag induced forces respectively. $\boldsymbol{\tau}_g$ is the gyroscopic torque and the term $\boldsymbol{\tau}_a$ is the lumped aerodynamic torques from flapping and drag.

The induced drag from the blades appears when the UAV is exposed to wind due to high velocities or air movement. The relative air movement will make one blade advancing the wind and the other retreating, which creates different lifts from the blades. This will introduce both parasitic torques lumped into $\boldsymbol{\tau}_a$ and forces $\boldsymbol{F}_{drag}$. To describe these effects some notations will be introduced. Let an external wind velocity $\boldsymbol{v}_w$, defined in inertial frame, be present and let $\boldsymbol{v}_a^i = \boldsymbol{R}_{\mathcal{IB}}(\dot{\boldsymbol{p}} - \boldsymbol{v}_w) + \boldsymbol{\omega} \times \boldsymbol{d}_i$ be the relative wind velocity at rotor i in body frame. Further let $\boldsymbol{d}_i$ denote the position of rotor i in body frame and finally we introduce a matrix transformation $\boldsymbol{\pi}_{\hat{z}} = \boldsymbol{I} - \hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^T$ projecting a vector into the xy-plane. The forces induced from the drag effect can now be expressed as

$$\boldsymbol{F}_{drag} = \sum_{i=1}^{4} \boldsymbol{F}_{drag}^i = \sum_{i=1}^{4} \sqrt{T_i} c_{d1} \boldsymbol{\pi}_{\hat{z}} \boldsymbol{v}_a^i \tag{2.140}$$

The torque generated from the drag effect will consist of the different lifts on the blades, a drag dampening effect and the torques generated by the drag forces

$$\boldsymbol{\tau}_{drag} = -\text{sign}(\Omega_i)\sqrt{T_i} c_{d2} \boldsymbol{\pi}_{\hat{z}} \boldsymbol{v}_a^i - \sqrt{T_i} c_{d3} \boldsymbol{\pi}_{\hat{z}} \boldsymbol{\omega} + d_i \times \boldsymbol{F}_{drag}^i \tag{2.141}$$

Note that Ω_i always is positive for the Crazyflie meaning $\text{sign}(\Omega_i) = 1$, always. The difference in lift between the advancing blade and retreating blade will also cause another effect if the rotor blades are flexible. Because of the different lifts, the blade will be bent so that the rotation plane will be tilted away from the wind. This is called flapping and causes a force that is not perpendicular to the body xy-plane. This will cause additional torques and forces that have a similar structure to the drag effect. Let c_1, c_2, c_3 and c_4 be positive constants, then the flapping force from one rotor can be modelled by [Kai et al., 2017]

$$\begin{aligned}\boldsymbol{F}^i_{flap} &= -\sqrt{T_i}c_1\boldsymbol{\pi}_{\hat{\boldsymbol{z}}}\boldsymbol{v}^i_a + \operatorname{sign}(\Omega_i)\sqrt{T_i}c_2\hat{\boldsymbol{z}} \times \boldsymbol{v}^i_a \qquad (2.142)\\ &- \operatorname{sign}(\Omega_i)\sqrt{T_i}c_3\boldsymbol{\pi}_{\hat{\boldsymbol{z}}}\boldsymbol{\omega} - \sqrt{T_i}c_4\hat{\boldsymbol{z}} \times \boldsymbol{\omega} \qquad (2.143)\end{aligned}$$

The total flapping force $\boldsymbol{F}_{flap} = \sum_{i=1}^4 \boldsymbol{F}^i_{flap}$ have two components. One is the translational speed induced flap and the other is the rotational induced flap. The first will have a structure similar to $\boldsymbol{F}_{drag}$. The second component is the rotational induced force which depends on $\boldsymbol{\omega}$, the angular speed.

The last effect is the gyroscopic torque $\boldsymbol{\tau}_g$ which can be expressed as

$$\boldsymbol{\tau}_g = -\boldsymbol{I}_{blade}\sum_{i=1}^{4}\boldsymbol{\omega} \times \Omega_i e_3 \qquad (2.144)$$

where $\boldsymbol{I}_{blade}$ is the moment of inertia of one rotor. This expression for the gyroscopic torque assumes the rotor as completely rigid. However, most quadrotors have non-rigid rotors which decreases the gyroscopic effect significantly which comes with the cost of more flapping induced forces.

Since the flapping and drag effect has a similar structure the induced forces will be lumped together to form the following simplified system of equations

$$\begin{aligned}m\ddot{\boldsymbol{p}} &= \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{F}_B - mg\hat{\boldsymbol{z}}\\ &\quad + \sum_{i=1}^{4}\sqrt{T_i}\left(\boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{A}\boldsymbol{R}_{\mathcal{IB}}(\boldsymbol{q})\left(\dot{\boldsymbol{p}} - \boldsymbol{v}_w\right) - \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{B}\omega\right) \qquad (2.145)\\ \dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix} \qquad (2.146)\\ \dot{\boldsymbol{\omega}} &= \boldsymbol{I}_{\mathcal{B}}^{-1}\left(\boldsymbol{\tau} - [\boldsymbol{\omega}]_\times\boldsymbol{I}_B\boldsymbol{\omega} - \boldsymbol{\tau}^r_g + \sum_{i=1}^{4}\sqrt{T_i}\left(\boldsymbol{C}\boldsymbol{R}_{\mathcal{IB}}(\boldsymbol{q})(\dot{\boldsymbol{p}} - \boldsymbol{v}_w) - \boldsymbol{D}\boldsymbol{\omega}\right)\right) \qquad (2.147)\end{aligned}$$

Here $\boldsymbol{\tau}^r_g$ is the residual gyroscopic effect assuming the rotors are flexible and $\boldsymbol{v}_w$ is external wind speeds in the inertial frame. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and

$\boldsymbol{D}$ are constant and depends upon aerodynamic coefficients and geometries of the blades. The structures of the matrices are $\boldsymbol{A} = a\boldsymbol{\pi}_{\hat{z}}$, $\boldsymbol{B} = b[\hat{\boldsymbol{z}}]_\times$, $\boldsymbol{C} = c[\hat{\boldsymbol{z}}]_\times$ and $\boldsymbol{D} = d_1\boldsymbol{\pi}_{\hat{z}} + d_2\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^T$. Exact values of a, b, c, d_1 and d_2 are not known (see [Kai et al., 2017]). The model will however only be used to apply uncertainties with realistic structure in simulation. It is further possible, to do the simplification that all rotors will have approximately the same thrust during near hover conditions for the position dynamics. This will make the sum in (2.145) disappear and be replaced by a multiplication of $2\sqrt{T}$, where $T = 4T_i$ is the total thrust.

Another effect that also will be present even during indoor flight is the near-ground effect. This effect will increase the force from the rotors in close proximity to ground [Sanchez-Cuevas et al., 2017]. Since the effect will differ depending on ground surfaces and the exact distance to the ground a dedicated sensor is typically required to model this uncertainty as structured. The velocities of the system will typically be small close to the ground, making it possible to model the near-ground effect as unstructured if a sufficiently fast time-varying adaption is used.

3

Controller Design

3.1 Full controller structure

To provide an overview of the proposed controller, the full cascade controller structure will here be presented before each controller is explained in more detail. A schematic figure of the full controller structure can be seen in Figure 3.1. The rotor feed-forward controller will be the inner controller, receiving trust and torque commands which are translated to rotor angular rates. The MRAC controller takes attitude commands $\boldsymbol{q}_{ref}$ and $\boldsymbol{\omega}_{ref}$ and sends desired torques to the rotor feed-forward controller. The MPC will, in turn, give commands in desired attitude angles and rates to the MRC/MRAC to follow the given position trajectory. To ease the work for the MPC, a trajectory generator is used to calculate a feasible trajectory according to the nominal dynamics. The different controllers will now be explained in detail.

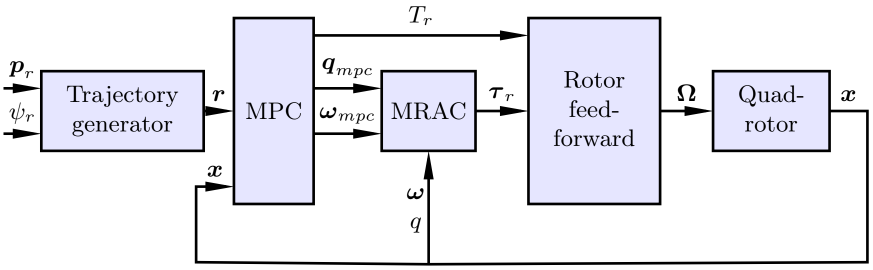

Figure 3.1 Structure of the full controller with the proposed MPC. The subscripts r denote a reference variable.

3.2 Rotor feed-forward

The rotor control implemented in the Crazyflie is a simple feed-forward controller that uses the input voltage that gives the desired output in stationarity. Since no phase compensation is done the lowpass nature of the rotor dynamics from voltage to thrust will be present in the dynamics. Further, the rotor speed is not measurable, which creates a problem when calculating the error dynamics. The difference between two time-series that have different phase delay will create an error increasing with higher frequency content. This makes it necessary to account for the rotor dynamics in the nominal model when the error dynamics are calculated. An approximative identification of the dynamics has been performed by [Greiff, 2017]. This model can be used to design a unit gain filter with the same phase delay properties. In a real-time implementation, the phase delay would have to be taken care of. An even better solution would be to use angular speed sensors in the rotors to construct a feedback controller. This is, however, not possible for all quadrotor platforms. An alternative solution could be to construct a more complex feed-forward controller by dynamic inversion of the rotor model to lower the phase-delay.

In simulations, it will be assumed that the rotor controller only is a thrust to input voltage mapping that does not reduce the phase delay, as on the Crazyflie platform. However, when the model error is calculated the true thrust and torques are used.

3.3 Inversion based MRAC

Due to the small inertial tensor of the Crazyflie (see Table 2.1) the attitude dynamics will be too fast to use sample times low enough for off-board control. Hence an onboard implementation on the quadrotor will be necessary. Model Reference Control (MRC) is a control scheme that will be computationally tractable and at the same time can use the GP estimated model for control. This section introduces a non-linear inversion based MRC scheme, modified to fit the attitude dynamics in quaternion form.

Assume first that a system $\mathcal{S}$ with state vector $\boldsymbol{x} = [\boldsymbol{x}_1,\ \boldsymbol{x}_2]^T \in \mathbb{R}^{2n_x}$, $\boldsymbol{x}_1 \in \mathbb{R}^{n_x}$, $\boldsymbol{x}_2 \in \mathbb{R}^{n_x}$ and input vector $\boldsymbol{u} \in \mathbb{R}^{n_u}$ has the form

$$\mathcal{S} : \begin{cases} \dot{\boldsymbol{x}}_1 &= \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_2 &= f(\boldsymbol{x}) + b(\boldsymbol{x})\boldsymbol{u}. \end{cases} \tag{3.1}$$

Further, assume that approximations of the functions f and b, denoted $\hat{f}$ and $\hat{b}$ are known and that $\hat{b}^{-1}$ has no singularities. It is then possible to construct

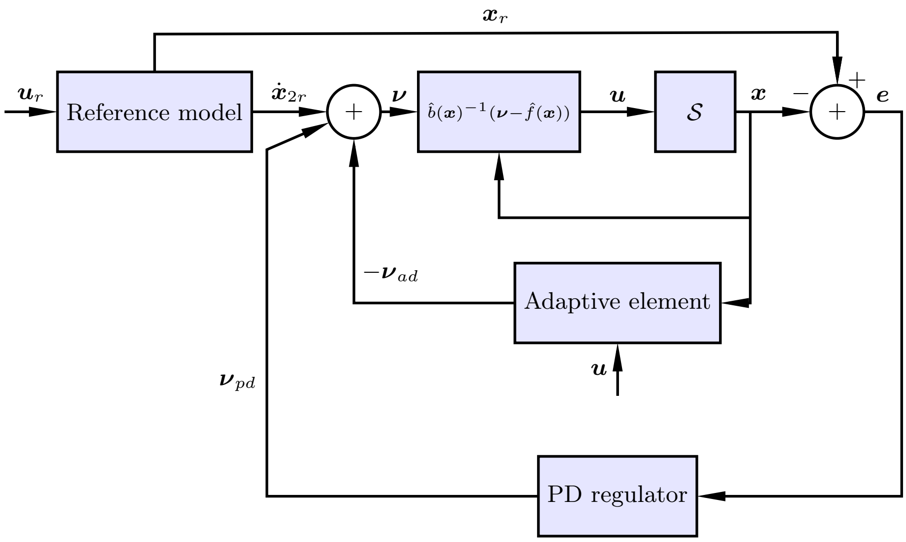

Figure 3.2 Blockdiagram of the inversion based MRAC. The notation $\mathcal{M}^{-1}$ symbolizes the dynamic inversion in (3.2).

a dynamic inversion that can be used as input to $\mathcal{S}$ as

$$\boldsymbol{u} = \hat{b}(\boldsymbol{x})^{-1}(\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})). \tag{3.2}$$

Using this feedback rule will result in

$$\begin{aligned}
\dot{\boldsymbol{x}}_2 &= f(\boldsymbol{x}) + b(\boldsymbol{x})\boldsymbol{u} && (3.3)\\
&= f(\boldsymbol{x}) + b(\boldsymbol{x})\hat{b}(\boldsymbol{x})^{-1}(\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})) && (3.4)\\
&= f(\boldsymbol{x}) + b(\boldsymbol{x})\hat{b}(\boldsymbol{x})^{-1}(\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})) + (\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})) - (\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})) && (3.5)\\
&= f(\boldsymbol{x}) + (b(\boldsymbol{x}) - \hat{b}(\boldsymbol{x}))\hat{b}(\boldsymbol{x})^{-1}(\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})) + (\boldsymbol{\nu} - \hat{f}(\boldsymbol{x})) && (3.6)\\
&= f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) + (b(\boldsymbol{x}) - \hat{b}(\boldsymbol{x}))\boldsymbol{u} + \boldsymbol{\nu} && (3.7)\\
&:= \Delta(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\nu} && (3.8)
\end{aligned}$$

where $\boldsymbol{\nu}$ is a reference acceleration, referred to as the virtual control signal (see Figure 3.2). A designer chosen reference model is then used to generate a trajectory to be followed by the controller as

$$\begin{aligned}
\dot{\boldsymbol{x}}_{1r} &= \boldsymbol{x}_{2r} && (3.9)\\
\dot{\boldsymbol{x}}_{2r} &= f_r(\boldsymbol{x}_r) + b_r(\boldsymbol{x}_r)\boldsymbol{u}_r, && (3.10)
\end{aligned}$$

In order to follow the trajectory we define the tracking error

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{x}_{1r} - \boldsymbol{x}_1 \\ \boldsymbol{x}_{2r} - \boldsymbol{x}_2 \end{bmatrix} \Rightarrow \dot{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{x}_{2r} - \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_{2r} - \Delta(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{\nu} \end{bmatrix} \tag{3.11}$$

By now choosing the virtual input $\boldsymbol{\nu}$ to

$$\boldsymbol{\nu} = \dot{\boldsymbol{x}}_{2r} + \underbrace{\boldsymbol{K}_1(\boldsymbol{x}_{1r} - \boldsymbol{x}_1) + \boldsymbol{K}_2(\boldsymbol{x}_{2r} - \boldsymbol{x}_2)}_{\boldsymbol{\nu}_{pd}(\boldsymbol{x}_r, \boldsymbol{x})} - \boldsymbol{\nu}_{ad}(\boldsymbol{x}, \boldsymbol{u}), \tag{3.12}$$

we have the full MRAC controller depicted in Figure 3.2 where $\dot{\boldsymbol{x}}_{2r}$ is the reference trajectory, $\boldsymbol{\nu}_{pd}(\boldsymbol{x}_r, \boldsymbol{x})$ is a proportional-derivative compensator and $\boldsymbol{\nu}_{ad}(\boldsymbol{x}, \boldsymbol{u})$ an adaptive element term. Using this virtual input in (3.8) will result in the tracking error dynamics

$$\dot{\boldsymbol{e}} = \underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{K}_1 & -\boldsymbol{K}_2 \end{bmatrix}}_{\boldsymbol{M}} \boldsymbol{e} - \begin{bmatrix} 0 \\ \boldsymbol{I} \end{bmatrix} \Big(\boldsymbol{\nu}_{ad}(\boldsymbol{x}, \boldsymbol{u}) - \Delta(\boldsymbol{x}, \boldsymbol{u}) \Big) \tag{3.13}$$

By choosing $\boldsymbol{K}_1 \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{K}_2 \in \mathbb{R}^{n_x \times n_x}$ to make $\boldsymbol{M} \in \mathbb{R}^{2n_x \times 2n_x}$ Hurwitz the error dynamics will be globally asymptotically stable under the assumption $\nu_{ad}(\boldsymbol{x}, \boldsymbol{u}) - \Delta(\boldsymbol{x}, \boldsymbol{u}) = 0$. Consequently, the idea is to design $\nu_{ad}(\boldsymbol{x}, \boldsymbol{u})$ to approach $\Delta(\boldsymbol{x}, \boldsymbol{u})$ through the GP-framework. Note that a necessary assumption for ν_{ad} to be able to cancel the model error Δ is that the model error is stationary in the sense that $\nu_{ad}(\boldsymbol{x}, \boldsymbol{u}) = \Delta(\boldsymbol{x}, \boldsymbol{u})$ has a fixed point solution. This implies that the model error cannot be fully cancelled under non-stationary disturbances and model errors [Chowdhary et al., 2013a].

Quaternion based MRAC

In the previous section, it was assumed that the system contained a simple integrator that is fundamentally known. This would be an appropriate form if the Euler angle representation would be used (see (2.114)). The Euler angle representation, however, can give rise to singular rotations at well defined Euler angles, making it a suboptimal parametrization of SO(3). A better option is to use quaternions. The attitude dynamics was introduced in (2.134) and (2.135) and are restated here

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{3.14}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{I}_{\mathcal{B}}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]_\times \boldsymbol{I}_{\mathcal{B}} \boldsymbol{\omega}) := f(\boldsymbol{\omega}) + b(\cdot)\boldsymbol{\tau} \tag{3.15}$$

From these equations, it is clear that the dynamics no longer contains a simple integrator but some non-linear dynamics in the top equation (3.14). These non-linear dynamics are however fundamental with no unknown parameters. The dynamics describe the relationship between the rotation parametrised by $\boldsymbol{q}$ and the attitude rate $\boldsymbol{\omega}$. This means that the MRAC scheme in the previous section can be applied in a similar way. To do this, we will first construct a PD compensator. As $\boldsymbol{q}$ and $-\boldsymbol{q}$ correspond to the same element of $\boldsymbol{R}(q) = \boldsymbol{R}(-q) \in \mathcal{SO}(3)$ a difference between a reference quaternion $\boldsymbol{q}_r$ and $\boldsymbol{q}$ is not relevant. Instead the quaternion difference defined as $\Delta\boldsymbol{q} = \boldsymbol{q}^* \otimes \boldsymbol{q}_r$, will be used. If the demanded rotation is more than π radians away, the smallest rotation possible will instead be the conjugated one [Fresk and Nikolakopoulos, 2013]. This situation can be resolved by looking at the sign of $\Re\{\boldsymbol{q}^* \otimes \boldsymbol{q}_r\}$. The full PD feedback can then be designed as

$$\boldsymbol{\tau}_{pd} = [\boldsymbol{K}_q^{pd} \; \boldsymbol{K}_\omega^{pd}] \begin{bmatrix} \mathrm{sign}(\Re\{\Delta\boldsymbol{q}\})\Im\{\Delta\boldsymbol{q}\} \\ \boldsymbol{\omega}_r - \boldsymbol{\omega} \end{bmatrix} \tag{3.16}$$

It now remains to design the feedback parameters $\boldsymbol{K}_q^{pd}$ and $\boldsymbol{K}_\omega^{pd}$ to make the error dynamics sufficiently fast. A stability proof for the suggested feedback can be found in [Brescianini et al., 2013]. Similar to the previous section this proof only holds for the case when the true system is known.

Let us now assume that an approximation of the true system in (3.15), defined by an approximate inertia tensor $\hat{\boldsymbol{I}}_\mathcal{B}$, is known. A dynamic inversion according to (3.2) will then give the control law

$$\boldsymbol{\tau} = \hat{\boldsymbol{I}}_\mathcal{B}\boldsymbol{\nu} + [\boldsymbol{\omega}]_\times \hat{\boldsymbol{I}}_\mathcal{B}\boldsymbol{\omega}, \tag{3.17}$$

where $\boldsymbol{\nu}$ can be designed similarly to the previous section. To this end we need to design a reference model that can generate a trajectory $\dot{\boldsymbol{\omega}}_r$. Let us introduce a reference model with states $\boldsymbol{q}_r$ and $\boldsymbol{\omega}_r$ and design the pseudo control law as

$$\boldsymbol{\nu} = \dot{\boldsymbol{\omega}}_r + \boldsymbol{\tau}_{\boldsymbol{pd}} - \boldsymbol{\nu}_{ad}. \tag{3.18}$$

To facilitate reference commands form the MPC positional controller, that will be designed later, the reference model must take reference commands in form of an attitude rate and a rotation, denoted $\boldsymbol{w}_{mpc}$ and $\boldsymbol{q}_{mpc}$ respectively. The constructed reference model, with states $\boldsymbol{q}_r$ and $\boldsymbol{\omega}_r$, also needs to produce a reference trajectory that can be followed by the true dynamics. To achieve this the dynamic model (3.15) will be used with the non-linear feedback law $\boldsymbol{\tau}_{pd}$ in (3.16). Let a reference quaternion denoted $\boldsymbol{q}_{mpc}$ and a reference attitude rate denoted $\boldsymbol{\omega}_{mpc}$ be available from an outer position controller. With this, a modified feedback law can be introduced as

$$\boldsymbol{\tau} = \hat{\boldsymbol{I}}_\mathcal{B}\boldsymbol{\tau}_{pd} + [\boldsymbol{\omega}]_\times \hat{\boldsymbol{I}}_\mathcal{B}\boldsymbol{\omega}. \tag{3.19}$$

where $\boldsymbol{\tau}_{pd}$ is defined as (3.16) with $\boldsymbol{q}_{mpc}$ and $\boldsymbol{\omega}_{mpc}$ as reference inputs. If $\hat{\boldsymbol{I}}_\mathcal{B}$ in (3.19) only is a diagonal matrix this only scales the feedback. The term

$[\boldsymbol{\omega}]_\times \hat{\boldsymbol{I}}_\mathcal{B}\boldsymbol{\omega}$ is used to cancel the same term in (3.15). The resulting closed loop will then define a feasible reference model as

$$\dot{\boldsymbol{q}}_r = \frac{1}{2}\boldsymbol{q}_r \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_r \end{bmatrix} \tag{3.20}$$

$$\dot{\boldsymbol{\omega}}_r = \boldsymbol{K}_q^r \text{sign}(\Re\{\Delta\boldsymbol{q}\})\Im\{\Delta\boldsymbol{q}\} + \boldsymbol{K}_\omega^r(\boldsymbol{\omega}_{mpc} - \boldsymbol{\omega}_r), \tag{3.21}$$

where $\Delta\boldsymbol{q} = \boldsymbol{q}_r^* \otimes \boldsymbol{q}_{mpc}$, $\boldsymbol{K}_q^r$ and $\boldsymbol{K}_\omega^r$ are positive definite diagonal matrices specified by a user, determining the convergence rate of the reference model. By this definition, it can be ensured that the generated trajectory is feasible under the assumption that no saturations are active in the actuation.

3.4 Exploiting the predictive uncertainty

Since a GP with RBF kernels only learns locally, the model error estimate quality may be very poor when the system ventures into new regions. This is a major stability issue when using GPs for adaptive control. A GP, however, provides possible solutions to this problem, since the posterior covariance of the GP provides a measure of uncertainty in the function estimate. This could potentially be used to provide the controller with information on how cautious it should be. In an inversion-based MRAC there are no natural ways of directly incorporating the covariance estimate but a solution that has been used in this setting before can be found in [Grande et al., 2014]. Assume the true system is $\dot{x} = f(x,u)$ and that a nominal approximate model is $\dot{x} = \hat{f}(x,u)$. Further assume a GP estimate of the model error $\Delta(x,u) = f(x,u) - \hat{f}(x,u)$ is $\Delta_{GP}(x,u)$. The idea is to scale the estimated model error with a gain depending on the uncertainty measure in the relevant point $\Sigma(z,z)$, $z = [x,u]^T$ (see (2.32)). To do this, a prior certainty on the nominal approximate model $\hat{f}$ is specified as a constant σ_f^2. With this, a heuristic approach to scale $\Delta_{GP}(x,u)$ could be designed as

$$\nu_{ad}(x,u) = \frac{\sigma_f^2}{\Sigma(z,z) + \sigma_f^2}\Delta_{GP}(x,u) \tag{3.22}$$

$$= \rho\Delta_{GP}(x,u), \tag{3.23}$$

which results in $0 < \rho < 1$, where ρ will be close to zero when $\Sigma(z,z) \gg \sigma_f$ and close to one when $\Sigma(z,z) \ll \sigma_f$. The MRAC law in (3.12) could then be designed as $\nu = \dot{x}_{rm} + \nu_{pd} - \rho\Delta_{GP}(x,u)$. In practice, however, the gain ρ will jump around when the GP is learning. To this end, a low-pass filter can be used to smooth the time-varying gain ρ as

$$\rho_{t+1} = \rho_t + \epsilon_{gain}\left(\frac{\sigma_f^2}{\Sigma(z,z) + \sigma_f^2} - \rho_t\right) \tag{3.24}$$

where $\epsilon \in [0,1)$ is chosen by the user, determining the low-pass character of ρ.

A change in this method compared to the original one, outlined above, as proposed in [Grande et al., 2014] will however be done. Since the TV-SOGP does not converge to zero covariance because of the process noise, the gain ρ will never reach one. To this end, we can look at the minimum stationary covariance for the TV-SOGP. It can be found by solving (2.87) to give

$$\sigma_{min}^2 = \frac{\sigma_p^2}{2} + \sqrt{\frac{\sigma_p^4}{4} + \sigma_p^2\sigma_n^2}. \tag{3.25}$$

A heuristic approach to make the gain ρ closer to one in stationarity is to change it to

$$\rho = \frac{\sigma_f^2}{\Sigma(z,z) - \sigma_{min}^2 + \sigma_f^2} \tag{3.26}$$

before low-pass filtering it.

3.5 Position MPC

To plan and track a positional trajectory, Model Predictive Control (MPC) will be used. This is because of several reasons. First of all, this thesis focuses on using an identified dynamic model online to control the quadrotor. This calls for a model based controller where the model can be changed online. One control method where this is possible is inversion-based MRAC which has been described in the previous section. The problem with this method for the positional dynamics is that an exact inversion of the nominal dynamics is not possible because the dynamic equation is overdetermined. This is not a problem in the MPC method since optimization is used to find the optimal input.

MPC is a finite receding horizon control problem, where the nominal dynamic model is used to predict the first H time steps given an initial state $\boldsymbol{x}_0$ at time t. The number of prediction steps into the future H is often called the horizon. Given a cost function of the states, $\boldsymbol{X} = [\ \boldsymbol{x}_1,\ ...\ \boldsymbol{x}_H]$ and inputs, $\boldsymbol{U} = [\boldsymbol{u}_0,\ \boldsymbol{u}_1,\ ...\ \boldsymbol{u}_{H-1}]$ an optimization over both $\boldsymbol{U}$ and $\boldsymbol{X}$ is performed at every time instant beginning from the initial state $\boldsymbol{x}_0$. The solution to this optimization then gives the optimal trajectory H steps ahead. In the MPC setting only the first control action $\boldsymbol{u}_0$ is actuated in the plant. The next control action will be given by solving a similar optimization problem at the next time-step.

The advantage of MPC is that it is easy to incorporate constraints on the states and controls to guarantee saturations in control signals are not violated and that infeasible parts of the state space are not visited. To ensure that the optimal control problem will be feasible under constraints, a sufficiently large horizon H is needed.

MPC for a discrete non-linear time invariant system can be formulated as an Optimal Control Problem (OCP) with constraints. To this end let $\mathcal{X}$ be the set of all feasible $\boldsymbol{x}_t \in \mathbb{R}^n$ and $\mathcal{U}$ be the set of all feasible $\boldsymbol{u}_t \in \mathbb{R}^m$. Further let $\boldsymbol{Q}$, $\boldsymbol{Q}_f \in \mathbb{R}^{n \times n}$ and $\boldsymbol{R} \in \mathbb{R}^{m \times m}$ be positive definite matrices. The OCP can then be formulated as

$$\min_{U,X} \sum_{i=1}^{H-1} \boldsymbol{x}_i^T \boldsymbol{Q} \boldsymbol{x}_i + \boldsymbol{x}_H^T \boldsymbol{Q}_f \boldsymbol{x}_H + \sum_{i=0}^{H-1} \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i \tag{3.27}$$

$$\text{subject to} \tag{3.28}$$

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{3.29}$$

$$\boldsymbol{x}_t \in \mathcal{X} \tag{3.30}$$

$$\boldsymbol{u}_t \in \mathcal{U} \tag{3.31}$$

If the constraints permit this would drive the system to the origin by virtue of the matrices $\boldsymbol{Q}$, $\boldsymbol{Q}_f$ and $\boldsymbol{R}$ being positive definite. Since the quadrotor dynamics is highly non-linear the general OCP formulation above would be advantageous. OCPs of nonlinear dynamics, however, generally result in non-convex problems, and the numerical complexity increases compared to Quadratic Programming (QP) problems for linear systems. This can result in an intractable solution for real-time purposes and convergence to a global optimum is no longer guaranteed. An example of a software tool that handles the intractability of such formulations is the ACADO toolkit that builds upon the Real-Time Iterating scheme (RTI) [Houska et al., 2011]. This solution only uses one numerical optimization step at every sample time. This means sufficiently fast sample time is needed to ensure convergence but it will never be guaranteed. The ACADO toolbox as a whole incorporates more than just the RTI scheme, but it only supports code generation for RTI.

The different ways of representing rotation in the quadrotor dynamics have different advantages as explained earlier. The quaternion formulation uses a vector of dimension four with the added constraint of having unit L2-norm. This is a non-convex constraint on the quaternion and cannot be incorporated in a convex OCP. The RTI scheme in ACADO could, however, be used to solve such non-convex and non-linear problems. This was tested during the course of this thesis, but it suffered from poor convergence. The Euler angles, on the other hand, do not suffer from non-convex constraints but instead, have problems with singularities. However, since the aim of this thesis is not to fly at extreme attitudes the Euler formalism will be used to model the dynamical constraint in the MPC. The rest of the thesis will, therefore, focus on linear MPC using linear approximations of the quadrotor dynamics parameterized in Euler angles.

Existing methods for Linear Time Invariant systems (LTI) using Quadratic Programming (QP) are very efficient, and there exist many tools that facilitate code generation of optimized QP solvers. To use these, the non-linear dynamics of the quadrotor would have to be linearized. However, a linearization around the hover point would only be a decoupled double integrator, which would likely result in bad performance. To improve the performance a linearization around the current state $\boldsymbol{x}_0$ could be done in every sample instance. This approximation will be called a Piece-Wise Affine (PWA) approximation. This makes it possible to use QP optimization tools for efficient code generation that can be used for real-time control while simultaneously taking advantage of the non-linear model.

MPC for a discrete LTI system resulting from the PWA approximation can be formulated as a Quadratic Programming OCP with constraints. To this end, define $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\boldsymbol{G} \in \mathbb{R}^{n\times 1}$ and a reference trajectory $[\boldsymbol{r}_1,\ \boldsymbol{r}_2,\ ...\ \boldsymbol{r}_H] \in \mathbb{R}^{n\times H}$. A QP optimization over the trajectory can then be formulated as

$$\begin{aligned}\min_{U,X}\ & \sum_{i=1}^{H-1} (\boldsymbol{x}_i - \boldsymbol{r}_i)^T \boldsymbol{Q} (\boldsymbol{x}_i - \boldsymbol{r}_i) + (\boldsymbol{x}_H - \boldsymbol{r}_H)^T \boldsymbol{Q}_f (\boldsymbol{x}_H - \boldsymbol{r}_H) \qquad (3.32)\\ & + \sum_{i=0}^{H-1} \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i\end{aligned}$$

subject to

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{G}, && t = 1, 2...H-1 && (3.33)\\ \boldsymbol{F}\boldsymbol{x}_t &< \boldsymbol{b}, && t = 1, 2...H && (3.34)\\ \boldsymbol{E}\boldsymbol{u}_t &< \boldsymbol{c}, && t = 1, 2...H-1 && (3.35)\end{aligned}$$

where $\boldsymbol{F}$ and $\boldsymbol{E}$ are matrices and $\boldsymbol{b}$ and $\boldsymbol{c}$ are vectors with dimensions corresponding to the number of constraints. The formulation in (3.5)-(3.35) is easily implemented in QP code generation tools such as CVXGEN [Mattingley and Boyd, 2012] and μAO-MPC [Zometa et al., 2013]. CVXGEN was chosen for this project. The CVXGEN tool makes it possible to use sparse matrices in the constraints to exploit structure in the dynamics and also change parameters online so that the PWA system approximation can be used.

Dynamic model

A modification can be made to the dynamic equations in (2.113)-(2.114) to speed up the optimization problem. The MPC will be operating at a much longer time-scale compared to the attitude controller since the magnitude of the inertia tensor is small (see Table 2.1). This makes it reasonable to assume that including the attitude dynamics in the MPC will have little effect. Just excluding the attitude dynamics in (2.114) would give a linearised system of order six. Another possibility is to replace the attitude dynamics with a simple integrator. This has several advantages: the reference input to the attitude controller will be smoothened from a zero order hold signal to linear interpolation and a very sparse structure of the dynamics is achieved.

The proposed nonlinear dynamic model used in the MPC is

$$m\ddot{\boldsymbol{p}} = \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{\eta})\hat{\boldsymbol{z}}T - mg\hat{\boldsymbol{z}} \tag{3.36}$$

$$\dot{T} = T_{slew} \tag{3.37}$$

$$\ddot{\boldsymbol{\eta}} = \dot{\boldsymbol{\eta}}_{slew} \tag{3.38}$$

where T_{slew} and $\dot{\boldsymbol{\eta}}_{slew}$ will be the output of the MPC. The above dynamics is the nominal dynamics used to define the dynamical constraint in the QP. However, since the aim is to incorporate uncertainties such as the drag and flap effects estimated by a GP, a term proportional to $\dot{\boldsymbol{p}}$ must also be incorporated in (3.36). This could be done in a general manner as

$$m\ddot{\boldsymbol{p}} = \boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{\eta})\hat{\boldsymbol{z}}T - mg\hat{\boldsymbol{z}} - \boldsymbol{D}\dot{\boldsymbol{p}}, \tag{3.39}$$

where $\boldsymbol{D} \in \mathbb{R}^{3\times 3}$ is a positive definite matrix.

Linearisation and discretization The linearisation can be performed in an arbitrary point spanned by the states and control inputs and does not have to be a stationary point. Introducing the state vector $\boldsymbol{x} = [\boldsymbol{p}, \dot{\boldsymbol{p}}, \boldsymbol{\eta}, T, \dot{\boldsymbol{\eta}}]^T$ and the input vector $\boldsymbol{u} = [T_{slew},\ \dot{\boldsymbol{\eta}}_{slew}]^T$, the linearisation results in an LTI system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_l\boldsymbol{x} + \boldsymbol{B}_l\boldsymbol{u} + \boldsymbol{G}_l \tag{3.40}$$

where the exact matrices can be found in Section **A.1**.

The outlined LTI system above needs to be discretized to allow for real-time MPC. The standard zero-order hold discretization of the system in (3.40) takes the form of a difference equation as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{u}_k + \boldsymbol{G}_d \tag{3.41}$$

The matrices $\boldsymbol{A}_d$,$\boldsymbol{B}_d$ and $\boldsymbol{G}_d$ can be found by a series expansion for fast real-time calculation as

$$\boldsymbol{A}_d = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{\boldsymbol{A}_l^i \Delta t^i}{i!} \tag{3.42}$$

$$\boldsymbol{B}_d = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{\boldsymbol{A}_l^i \Delta t^{i+1}}{(i+1)!} \boldsymbol{B}_l \tag{3.43}$$

$$\boldsymbol{G}_d = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{\boldsymbol{A}_l^i \Delta t^{i+1}}{(i+1)!} \boldsymbol{G}_l \tag{3.44}$$

The derivation of the zero-order hold approximation can be found in Section **A.2**. By using the symbolic toolbox of Matlab the non zero elements after convergence in the matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ were found (see Figure 3.5). These sparse matrix maps can be used in the CVXgen tool to significantly lower the complexity of the constraints in the optimization problem.

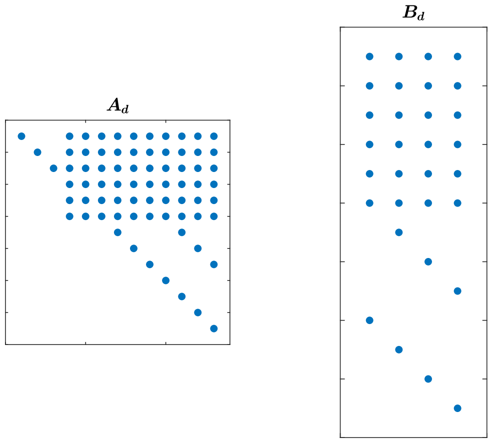

Figure 3.3 Non-zero elements after convergence by discretization with (3.42)-(3.43).

Constraints

Since the aim of the thesis is not to do obstacle avoidance or path planning, state constraints on $\boldsymbol{x}_k$ in (3.41) will not be considered. Constraints on the inputs could, however, be beneficial to avoid saturation. The total thrust T will be limited by the maximum angular speed of the rotors which is approximately $\Omega_{max} = 2500$ rad/s for the Crazyflie [Greiff, 2017]. This yields a constraint

$$0 \leq T \leq 4k\Omega_{max}^2 \tag{3.45}$$

The resulting constraints on the torques are harder to design since the attitude dynamics is approximated with an integrator. Because of this we do not have access to the torques inside the MPC. To incorporate the constraints an approximation has to be made. Remember the attitude dynamics in the Euler angle representation (2.114) and the definition of $\boldsymbol{J}(\boldsymbol{\eta})$. If the quadrotor is assumed to be close to hover the angles ϕ and θ will be small while ψ can be an arbitrary angle. The matrix $\boldsymbol{W}(\boldsymbol{\eta})$ will then be close to an identity matrix resulting in the approximation $\boldsymbol{J}(\boldsymbol{\eta}) \approx \boldsymbol{I}_{\mathcal{B}}$ if $\phi \approx \theta \approx 0, \forall\psi$. This would imply $\boldsymbol{\tau}_{\mathcal{B}} = \boldsymbol{J}(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} \approx \boldsymbol{I}_{\mathcal{B}}\boldsymbol{\eta}_{slew}$, where the approximation of $\boldsymbol{C}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} = 0$ has been used. From (2.98) it can be seen that the maximum torque is

$$\boldsymbol{\tau}_{max} = \begin{bmatrix} \sqrt{2}kl\Omega_{max}^2 \\ \sqrt{2}kl\Omega_{max}^2 \\ 2a\Omega_{max}^2 \end{bmatrix} \tag{3.46}$$

The torques and total thrust are however coupled. Thus, if the max trust is active, no torques can be generated at all. To consider this one can subtract the hover rotor speed, Ω_{hover}, of approximately 1800 rad/s^2 from Ω_{max} when calculating (3.46). The available rotor speed for generating a torque would then be $\Omega_{max} - \Omega_{hover}$.

A reasonable constraint could therefore be

$$|\dot{\boldsymbol{\eta}}_{slew}| \leq \boldsymbol{I}_{\mathcal{B}}^{-1}\boldsymbol{\tau}_{max} = \boldsymbol{I}_{\mathcal{B}}^{-1} \begin{bmatrix} \sqrt{2}kl(\Omega_{max} - \Omega_{hover})^2 \\ \sqrt{2}kl(\Omega_{max} - \Omega_{hover})^2 \\ 2a(\Omega_{max} - \Omega_{hover})^2 \end{bmatrix} \tag{3.47}$$

Because of the coupling and that the reasoning is extremely approximative, the constraints outlined above should be more restrictive when operating in real time.

Trajectory generation

To exploit the nonlinear model a trajectory can be generated if a six times differentiable position trajectory and a given yaw angle trajectory is given. If the dynamic model in (3.36)–(3.38) is used, the velocity $\dot{\boldsymbol{p}}$ and acceleration $\ddot{\boldsymbol{p}}$ can be obtained through direct numerical differentiation. Now it only remains to solve the dynamic equation (3.36) for the Euler angles. The solution when the yaw angle ψ is known can be found as [Limaverde Filho et al., 2016]

$$T = m\sqrt{\ddot{p}_x^2 + \ddot{p}_y^2 + (\ddot{p}_z + g)^2} \tag{3.48}$$

$$\theta = \tan^{-1}\Big(\frac{\ddot{p}_x\cos(\psi) + \ddot{p}_y\sin(\psi)}{\ddot{p}_z + g}\Big) \tag{3.49}$$

$$\phi = \tan^{-1}\Big(\frac{\sin(\theta)(\ddot{p}_y\cos(\psi) - \ddot{p}_x\sin(\psi))}{\ddot{p}_x\cos(\psi) + \ddot{p}_y\sin(\psi)}\Big) \tag{3.50}$$

Adaptive MPC

There are several ways of using GP model identification in an MPC. The problem of using offline identified GP models in MPC has been more thoroughly investigated than online identification [Kocijan, 2016]. Since a GP gives a nonlinear model it is natural to use it directly in a nonlinear MPC. Another possibility is linearising the GP and using the QP MPC formulation. Here we will use the latter alternative, making use of the PWA approximation.

Many interesting extensions can be made when using GPs in MPC schemes. Since GP regression gives an uncertainty measure it facilitates for example "cautious MPC". This MPC formulation can make use of information on the certainty of a function estimate and change the constraints to ensure stability [Hewing and Zeilinger, 2017]. Similar strategies could also be constructed with e.g. tube-based MPC [Cao et al., 2017]. These MPC formulations could also be very interesting in online GP regression and adaptive MPC where the robustness of the resulting control scheme is a priority.

GPs with their ability to identify a wide range of functions without much knowledge about their structure has also drawn attention by the fault tolerant control field [Maciejowski and Yang, 2013]. Fault tolerant control aims to detect faults such as actuator degradation or a complete failure and overcome them. In [Maciejowski and Yang, 2013] it is proposed to use online GP identification in a linear adaptive MPC formulation to detect and identify the behavior of faults. A similar strategy will be used here.

Exploiting the predictive uncertainty As described, there are many ways of incorporating the uncertainty estimate in an MPC. Both cautious MPC and tube MPC are intuitive ways of doing it. They, however, suffer from large computational complexity since the uncertainty propagates over the horizon. Further, tube MPC is often too restrictive and approximations have to be made to make the optimization problem feasible [Cao et al., 2017]. These limitations make it complicated to implement. This document will not focus on such methods but we only want to point to the possibility of such methods. Instead, the proposed ρ gain will be used to limit the influence of the adaptive element in the same way as in the GP-MRAC.

Linearisation of GP To use the estimated GP regression in the QP based MPC a linearization of the GP has to be performed. Since the mean function is a sum of weighted exponential functions this is straightforward. The expression will be left out.

4

Adaptive element

To use GPs to represent the adaptive element in both MPC and MRAC, some considerations have to be done. Because the number of kernel centers needed scales exponentially with the number of input dimensions, it might not be possible to have all state variables as inputs. Consequently, a carefully selected subset of input variables has to be chosen. Furthermore, the online calculation of the dynamic model error could be done in different ways. Phase delays in both dynamics and discrete time low-pass filters can cause unacceptable errors. This also requires attention.

Choosing inputs and outputs to GP

Which inputs that might be beneficial to use will now be discussed. The discussion will be divided into two parts. The first discussing concerns the uncertainties that could be modeled in the attitude dynamics (MRAC) and second in the positional dynamics (MPC). Some analysis of the structure of the uncertainties presented in Section **2.2** will also be made to select good input variables.

Attitude dynamics The moment of inertia when the body frame axis is aligned with the quadrotor arms is a diagonal matrix. However, it could be reasonable to assume this approximation is not perfect and that the off-diagonal element could be non zero because of attachments of components. Furthermore, the inertia could also change when the battery is placed in different ways. Attachments and battery placements that are not symmetrical will also contribute with torques that are dependent on the orientation $\boldsymbol{q}$ of the quadrotor. This could also be modeled if $\boldsymbol{q}$ is introduced as an input.

The model error originating from a parametric uncertainty in the approximate inertia tensor $\hat{\boldsymbol{I}}_{\mathcal{B}}$ will result in a model error

$$\boldsymbol{\Delta}(\boldsymbol{\omega}, \boldsymbol{\tau}) = \boldsymbol{I}_{\mathcal{B}}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]_{\times}\boldsymbol{I}_{\mathcal{B}}\boldsymbol{\omega}) - \hat{\boldsymbol{I}}_{\mathcal{B}}^{-1}(\boldsymbol{\tau} - [\boldsymbol{\omega}]_{\times}\hat{\boldsymbol{I}}_{\mathcal{B}}\boldsymbol{\omega}). \tag{4.1}$$

To estimate $\boldsymbol{\Delta}(\boldsymbol{\omega}, \boldsymbol{\tau}) = [\Delta_x, \Delta_y, \Delta_z]^T$ with GPs, an initial guess of the hyperparameters is needed especially if no hyperparameter optimization is used.

To determine a reasonable guess the analytic model error is illustrated in Figure 4.1 below, when the actual moment of inertia $\boldsymbol{I}_{\mathcal{B}}$ is an identity matrix and the approximate inertia is $\hat{\boldsymbol{I}}_{\mathcal{B}} = \boldsymbol{I}_{\mathcal{B}} + \boldsymbol{\delta}\boldsymbol{\delta}^T$ where $\boldsymbol{\delta} = [0.1, 0.2, 0.15]^T$. Since this is a 6D function it has been illustrated in 15 graphs corresponding to four dimensions being locked (i.e., the 6D room is intersected by 2D planes).

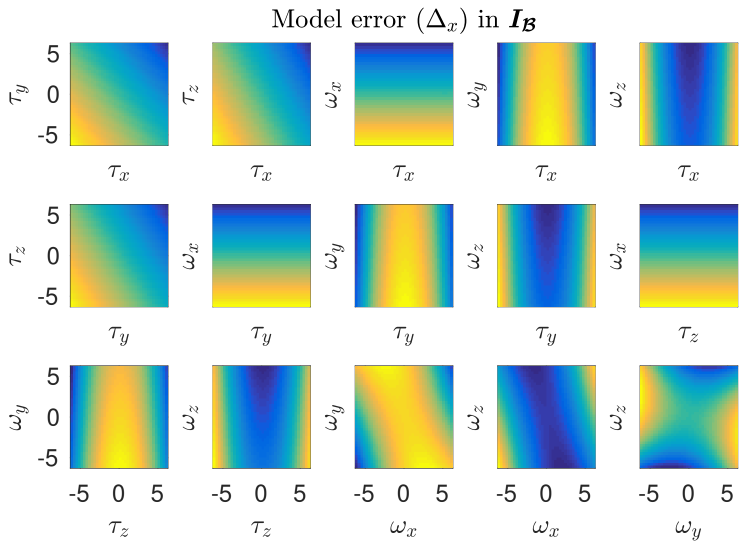

Figure 4.1 Model error Δ_x ploted in two-dimensional planes intersecting the six dimensional room.

To get a good value of the hyperparameters, an optimization of the marginal likelihood can be done. The optimization is, however, non-convex and can be hard to solve. For a spherical RBF kernel function with one hyperparameter, σ_k (see (2.17)), the cost is one-dimensional. An optimization with Matlab's 'fminunc' for non-linear problems results in the maximum likelihood for $\sigma_k \approx 15$. The optimization over the marginal likelihood can in many cases be hard since it involves inversion of the gram matrix. If the kernels are placed too close, the gram matrix might become ill-conditioned. This means great care has to be taken when choosing observations to optimize over. In Figure 4.1 the model error was plotted in the region that is believed to be visited under normal flight of the quadrotor. Because of the slowly varying function, it was however found that the kernel width of $\sigma_k \approx 15$ is optimal, which is much wider. The high value of $\sigma_k \approx 15$ is good since it means that we can hope to estimate the model error with very few kernel centers. However, there is no guarantee that other uncertainties will result in the same conclusion.

Other dynamical model uncertainties could also be modeled by the adaptive element. A simplified version of commonly left out aerodynamics can be seen in (2.145)–(2.147). The gyroscopic term (for the rigid rotor case) (2.144) depends on the attitude rate and the rotor speeds. The attitude rate is easily available and should be incorporated since it is the nominal input of the modeled dynamics. The rotor speeds cannot be measured but one can make approximations of what they are. Assuming the quadrotor is in near hover state the total trust will be approximately mg. This together with the linear equation system in (2.100) facilitates calculation of the rotor angular rate. It is, therefore, possible that the gyroscopic uncertainty can be approximated if the torque τ is incorporated as an input. The drag and flap terms are a bit more complex as they depend on the angular orientation $\boldsymbol{q}$, the lateral speed $\dot{\boldsymbol{p}}$, the rotor trusts T_i, the relative wind speed v_w and finally the angular rate $\boldsymbol{\omega}$. The only variable included in the nominal dynamics is $\boldsymbol{\omega}$. The rotor thrusts, T_i could possibly be approximated by the adaptive element if τ is known according to the reasoning above. However, it would be infeasible to incorporate the rest of the variables in a GP scheme. The drag and flap uncertainty also consists of one term $\sum_{i=1}^{4} \sqrt{T_i}\boldsymbol{D}\boldsymbol{\omega}$ that could be approximated with the nominal inputs. This is the induced drag and flap torque from angular rates. However, we cannot hope to estimate the rest of the uncertainties in a structured way.

In summary, the nominal inputs $\boldsymbol{\omega}$ and $\boldsymbol{\tau}$ will be necessary for all uncertainty estimations in a structured way. This already involves six input variables, which are on the limit of what could be feasible in an online setting. The gyroscopic term could be approximated but the effect is often very small and the approximation is of high uncertainty. The translational drag and flap uncertainty is complex with many variables that can not be incorporated. Finally the rotational flap and drag uncertainty $\sum_{i=1}^{4} \sqrt{T_i}\boldsymbol{D}\boldsymbol{\omega}$ could be possible to estimate in a structured way with the nominal inputs.

Note that all of the reasoning above assumes that T_i is known. This would imply that T would have to be incorporated in the adaptive element. Assuming $T = mg$ will only be a reasonable approximation in near hover state where the uncertainties described are very small. Since the attitude dynamics is a reasonably fast dynamics it is not reasonable to assume the GP can capture much of unstructured uncertainties by a time-varying estimation.

Position dynamics In the position dynamics (2.145), the only parametric uncertainties that could arise from a poor estimation of parameters in the nominal dynamics is the mass of the quadrotor m and the gravitational constant that should be well known. Further, it is possible that a voltage drop in the battery gives a lower thrust than expected. But this is an unstructured uncertainty since it actually changes the rotor dynamics.

With some assumptions, it is possible to account for the lateral drag and flap effect with a reasonably small set of input variables. From equation (2.145) the uncertainties not modelled in the nominal dynamics is

$$\sum_{i=1}^{4} \sqrt{T_i}\left(\boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{A}\boldsymbol{R}_{\mathcal{IB}}(\boldsymbol{q})(\dot{\boldsymbol{p}}-\boldsymbol{v}_w) - R(\boldsymbol{q})\boldsymbol{B}\boldsymbol{\omega}\right). \tag{4.2}$$

The nominal inputs to the position dynamics in (3.39), which is used in the MPC formulation are $\boldsymbol{q}$, T and $\dot{\boldsymbol{p}}$. With these inputs, it will be hard to model the second term consisting of the rotational induced drag. The first term originating from the lateral drag, however, has inputs that could be feasible to model. Assuming that the quadrotor is close to hovering, meaning that $T_1 \approx T_2 \approx T_3 \approx T_4 \approx T/4$ and $\boldsymbol{B} = 0$, the uncertainty can be written as

$$2\sqrt{T}\left(\boldsymbol{R}_{\mathcal{BI}}(\boldsymbol{q})\boldsymbol{A}\boldsymbol{R}_{\mathcal{IB}}(\boldsymbol{q})(\dot{\boldsymbol{p}}-\boldsymbol{v}_w)\right). \tag{4.3}$$

In this expression, the only unknown variable is the wind speed $\boldsymbol{v}_w$ in the inertial frame. Assuming that the wind is slowly time-varying the adaptive element could learn this as an unstructured uncertainty. The adaptive element input variables would then be $\boldsymbol{q}$, $\dot{\boldsymbol{p}}$ and T.

By exploiting the structure of the matrix $\boldsymbol{A} = \boldsymbol{I} - \hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^T$ that extracts the x and y component of the body lateral motion and assuming the quadrotor is close to hovering it is a reasonable assumption that the z component of $\dot{\boldsymbol{p}}$ will contribute least to the drag and flap effect. It can, therefore, be disregarded under not too aggressive flights.

4.1 Online calculation of model error (Δ)

Generally, if we want to estimate the model error in a scalar system

$$\dot{x} = f(x,u) \tag{4.4}$$

with respect to the approximate model $\hat{f}(x,u)$ the dynamic model error would be $\Delta = \dot{x} - \hat{f}(x,u)$. To evaluate this error both x and $\dot{x}$ has to be known. In the best of worlds, $\dot{x}$ can be measured but this is often not the case. Instead, a numerical differentiation of x has to be performed. When numerically differentiating the measured x an anti-aliasing and noise suppression filter has to be used. This, however, introduces a phase delay that can cause errors when taking the difference $\Delta = \dot{x} - \hat{f}(x,u)$. To solve this both signals have to be filtered through the same filter. Let p denote the differentiation operator and T denote a time constant that is smaller than half of the sampling frequency of x. Then

$$\Delta = \frac{p}{(p/T+1)^2}x - \frac{1}{(p/T+1)^2}\hat{f}\big(x,u\big) \tag{4.5}$$

will give an estimate of Δ that has no error resulting from phase delays in the anti-aliasing filter. However, the model error will with this model error calculation contain the dynamics of the implemented filter. Since this is the case it is important that the model error is calculated in a much higher sampling rate than the actual control sampling rate to minimize the effect of low-pass filter dynamics in the model.

It should be noted that problems still can arise if there exist other low pass characteristics in the actuation dynamics itself. This is the case of the quadrotor process where u is a desired thrust while the actual trust will be achieved when the rotors have reached their stationary angular speed. Even if the rotor dynamics is very fast compared to the controlled positional or attitude dynamics the phase delay might cause unacceptable errors when calculating the difference in (4.5). The best solution to this would, of course, be to measure u directly. The other alternative is to identify the low pass character and filter the desired input u with this filter.

In the attitude dynamics, the derivative of the attitude rate $\dot{\boldsymbol{\omega}}$ is not measurable so the method outlined above is needed. For the position dynamics, however, we have access to the accelerometer measurements. Let the accelerometer measurement be denoted $\boldsymbol{a}_{acc}$. It can be expressed as [Mueller et al., 2015]

$$\boldsymbol{a}_{acc} = \boldsymbol{R}_{\mathcal{IB}}(\ddot{\boldsymbol{p}} + \hat{\boldsymbol{z}}g). \tag{4.6}$$

If then the true rotation is known the acceleration $\ddot{\boldsymbol{p}}$ is measurable. The rotation will then have to be observed through a Kalman filter. Since the acceleration is known the filtering in (4.5) will not be needed. The difference between the model acceleration and the true acceleration can be done directly. An anti-aliasing filter can, however, be necessary.

An alternative way of estimating $\ddot{\boldsymbol{p}}$ would be to use an Extended Kalman filter. Implying that model error states has to be included in the Kalman filter state vector. This would help to improve the estimates of all states under large model errors due to e.g., wind disturbances and additionally providing estimates of the model error.

5

Simulations

The first part of this chapter will give two simulation examples of the proposed TV-SOGP algorithm. The first simulation compares TV-SOGP to another time varying SOGP algorithm previously proposed by [Chowdhary et al., 2013b]. This shows the benefits of TV-SOGP and validates the ability to adapt to time-varying functions. The second experiment compares TV-SOGP to parametric regression with a linear Kalman filter under low PE and unidentifiability due to feedback. The experiment shows that TV-SOGP avoids these problems by local learning and forgetting encoded by the RBF kernel. In the second part of the chapter, the TV-SOGP is applied to the quadrotor problem together with the attitude controller and position controller described in Section **3**. First, the GP-MRAC scheme is shown under parametric inertia tensor uncertainties together with strong wind disturbances entering the dynamics through flap and drag effects. Because of the low content of high frequencies in the model error it was possible to use wide kernels that facilitates real-time performance for sampling times sufficiently short for the attitude controller. The focus will then be directed to the GP-MPC scheme for the rest of the chapter. Two simulations will be done: one with a periodic trajectory in space and one with a random trajectory designed to visit a larger region of the state space. By visiting a large region of the state space it is shown that the SOGP scheme can be used under normal flight without the need of unfeasibly many kernel centers. To show the time-varying ability of TV-SOGP in an adaptive control setting different process noise values σ_p is compared for the periodic trajectory and lastly the TV-SOGP and MPC is profiled to show the real-time performance of the method.

5.1 SOGP simulations

TV-SOGP Simulation example A simulation example of the proposed TV-SOGP algorithm is given with a time-varying polynomial function $f(t,x,y) = (0.3\cos(\pi t/150) + 0.7)(y^2 - x^2)$ in a two-dimensional space (x, y).

Observations are collected along a circular trajectory that can be seen in Figure 5.2. The estimation in the current point over time can be seen in Figure 5.1, where the basic SOGP algorithm, the proposed time input in [Chowdhary et al., 2013b] and the TV-SOGP ()Kalman filter interpretation) are compared. The "agent" moving along the trajectory in Figure 5.2 observes noisy measurements $f_{obs} = f(t, x, y) + \epsilon$, $\epsilon \sim \mathcal{N}(0, 0.01)$. The agent tries to estimate the function value in the current point by using previous observations. It is possible to see that the basic SOGP converges to a stationary estimate while the TV-SOGP has a time-varying estimate following the function f better. Another important thing to note is the noisy estimate at the start of the experiment for all methods. They originate from drastic changes when the SOGP adds new kernels. Using the strategy of [Chowdhary et al., 2013b] with a time input to the GP, these oscillations will persist throughout the experiment while forgetting old kernels. This can be seen in the bottom graph of Figure 5.1 where spikes in the estimate can be seen throughout the whole experiment.

Note that the experiment in Figure 5.1 is designed to show the limitations of the proposed algorithm of [Chowdhary et al., 2013b]. The limitations mainly occur when a small budget together with a large ϵ_{tol} is chosen. Choosing a longer length scale parameter in the time dimension would give a smoother but slower estimate in the same way as a smaller process noise could be chosen in the Kalman filter. The main advantage of our proposed algorithm is however that no extra input has to be added. This means significantly fewer kernels have to be allocated. The sparse update is in itself also less computationally expensive than the algorithm proposed in [Chowdhary et al., 2013b].

It could be argued that the proposed Kalman filter interpretation is more easily tuned and used since the forgetting speed is only determined by one parameter σ_p. This is in contrast to when using the time input, where both the budget and the kernel width in the time dimension will affect the forgetting speed.

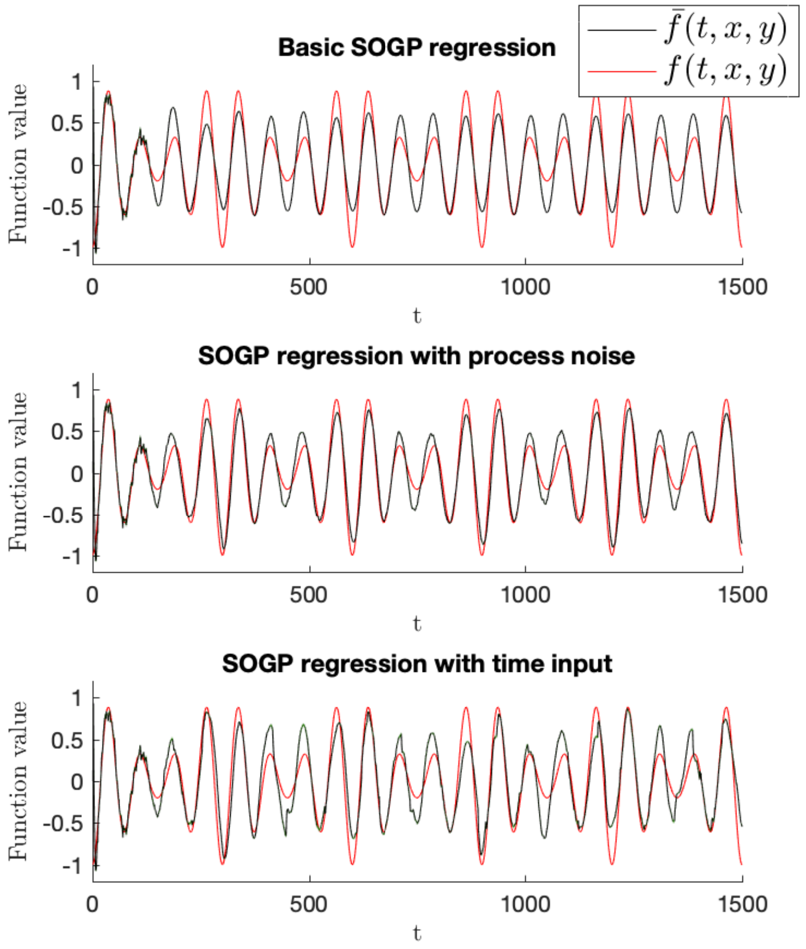

Figure 5.1 In the top graph, the basic SOGP is used on a noisy observation of the time varying function (see Figure 5.2). The true function f is shown without noise. In the second graph, the SOGP regression with the proposed Kalman filter update is used on a time-varying function. In the bottom graph the SOGP is used with the time t as an input. The predictive mean $\bar{f}$ (in black) is calculated before the observation in the relevant point is added. The values $\sigma_n = \sqrt{E(\epsilon^2)} = 0.1$ and $\epsilon_{tol} = 0.01$ were used. The length-scale hyperparameter was $\sigma_k = 1$ in the x and y dimensions, and the length-scale hyperparameter in the time dimension was chosen to 1000 for the bottom graph. The maximum budget was chosen to 20 and the process noise in the middle graph was chosen to $\sigma_p = 0.008$.

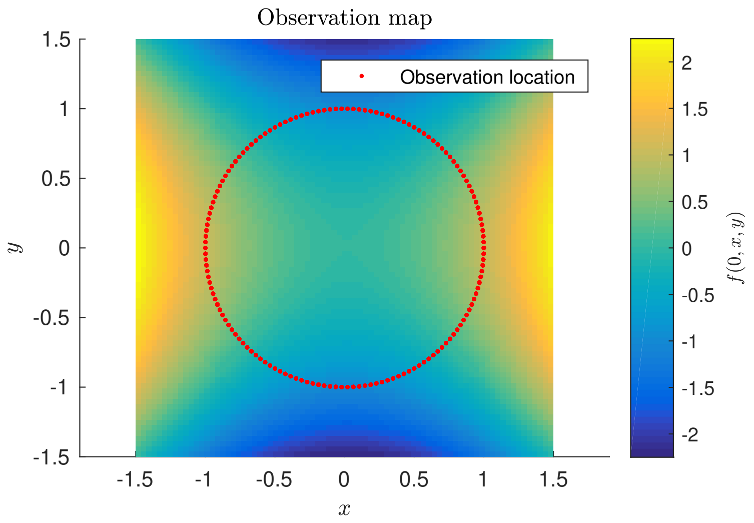

Figure 5.2 The underlying two-dimensional function observed in the simulation seen in Figure 5.1. The polynomial function is $f(0, x, y) = y^2 - x^2$. The observations of the function f along the circle trajectory is plotted as red dots.

Persistence of excitation and TV-SOGP

Since a GP encodes information locally and places kernel centers in the input space according to what has been observed we will see that the PE is not needed to get valuable information about the underlying function that can be used in adaptive control. The problem when using parametric models for identification online is that sufficient excitation of the system is necessary to get a valuable estimate of the underlying function. This can be seen in the following example. Let s denote the Laplace variable and assume we have a first-order linear system

$$G(s) = \frac{1}{s+0.1} \tag{5.1}$$

$$X(s) = G(s)U(s) \tag{5.2}$$

A zero-order-hold approximation of $G(s)$ with the sample time $\Delta t = 1/20$ s can be found as

$$x_{k+1} = 0.9950 x_k + 0.0499 u_k, \tag{5.3}$$

The system can then be controlled by a proportional controller $u_k = -1.5(x_k + v_k)$ in discrete time, where v_k is zero mean white measurement noise with standard deviation $\sigma_n = 0.05$. For an initial value of $x_0 = 5$ the system response can be seen in Figure 5.3.

If now a system identification would be done with the parametric model $x_{k+1} = a_{est} x_k + b_{est} u_k$ it could be possible to identify $a = 0.9950$ and $b = 0.0499$ if the data is sufficiently exciting enough. The data form Figure 5.3 however has a relationship between the input and output $u_k = -1.5(x_k + v_k)$ that makes the parameters a and b unidentifiable. Intuitively, this is because all observations will lay on a line in the input space (x_k, u_k) which makes it impossible to estimate the function over the whole two-dimensional plane. To use a time-varying regression such as a Kalman filter would also need PE over the whole interval, which can be seen to be far from true for the data in Figure 5.3. To show that this is a Kalman regression according to (2.74)–(2.77) with regressor $\boldsymbol{k}_k = [x_k, u_k]^T$ and process noise $R = \sigma_p^2 \boldsymbol{I}$, $\sigma_p = 0.01$ can be seen in Figure 5.4. It is clear that the estimate cannot recover the parameters a and b because of the feedback relation. Another problem is that the covariance matrix $\boldsymbol{P}$ of the estimates of a and b will grow unbounded over time as can be seen in the top graph of Figure 5.5 because of the lack of PE.

While the problems outlined above are impossible to get around for estimates of a and b we will now see that GP models and especially TV-SOGP get around the problem by forming a regressor vector that always will be non-zero if a prediction is done after updating the TV-SOGP. This will make

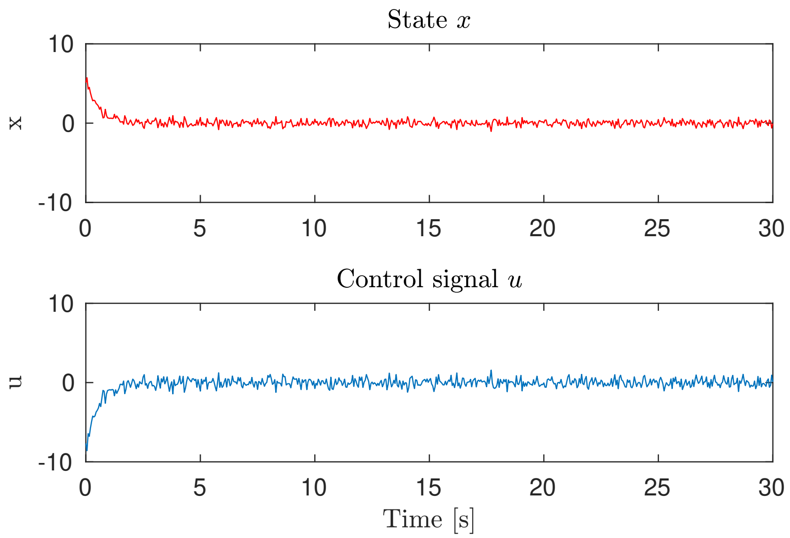

Figure 5.3 Initial value simulation of $G(s)$ (see (5.1)) with initial value $x_0 = 5$. Data sample time is $\Delta t = 1/20$ s.

regressor covariance matrix $\boldsymbol{P}_{GP} = \boldsymbol{C} + \boldsymbol{P}$ bounded. Intuitively, it can be seen since the kernel centers only will be placed in parts of the input space that is visited. The regressor vector $\boldsymbol{k}_k$ will hence always be non-zero in the neighborhood of the observations. A TV-SOGP regression was done for the same data seen in Figure 5.3 and as can be seen in the bottom graph of Figure 5.5 the regressor covariance matrix $\boldsymbol{C} + \boldsymbol{P}$ will remain bounded. When the cardinality of $\mathcal{BV}$ has converged it can be seen that the covariance even decreases until convergence. The discrete jumps in the covariance originate from the allocation of new kernel centers and the decrease afterward originates from the sparse Kalman update. To visualize the local learning of the SOGP the covariance over the input space is plotted in Figure 5.6. All allocated kernel centers can be seen together with the feedback line $u_k = -1.5x_k$. The SOGP has hence only learned the function values along this line but effectively avoiding the problem of low PE and the unidentifiable parameters a and b. It is also clear that local process noise designed in (2.85) does not make the covariance grow in other areas of the state space. This can be seen since almost all observations is collected in $(x_k, u_k) = (0, 0)$. If the process noise addition would not be local the covariance would be larger in the kernel centers far from $(x_k, u_k) = (0, 0)$. This is the reason the covariance $\boldsymbol{C} + \boldsymbol{P}$ can stay bounded even for time-varying regression under lack of PE.

The properties of TV-SOGP outlined above can be very useful in an adaptive control setting since the regression always gives valuable informa-

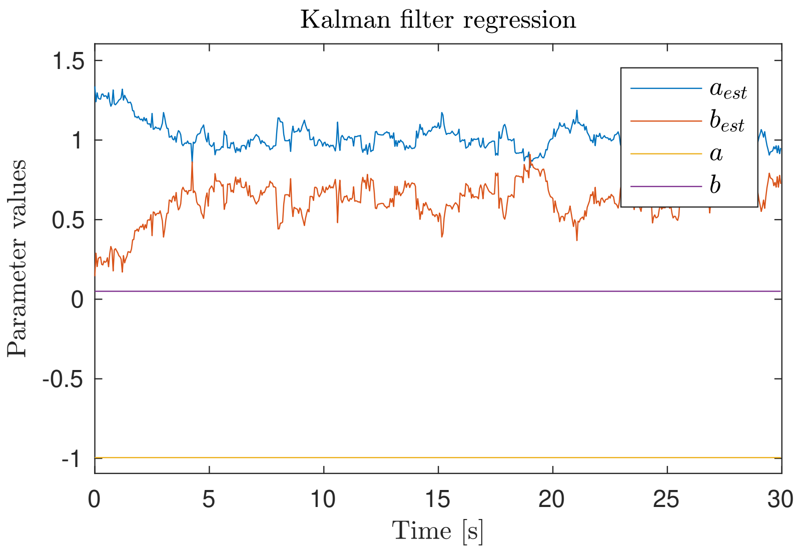

Figure 5.4 Kalman filter regression (see (2.74)–(2.77)) of data in Figure 5.3 with regression model $x_{k+1} = [a_{est}, b_{est}][x_k, u_k]^T = \boldsymbol{\alpha}^T \boldsymbol{k}_k$, process noise $\sigma_p = 0.01$ and measurement noise $\sigma_n = 0.05$.

tion from the local data even if no estimation over the whole input space is achieved. Since controlled systems often only visit limited parts of the input space this is not always a problem. The problem is however that the SOGP regression is much more computationally expensive than parametric regression.

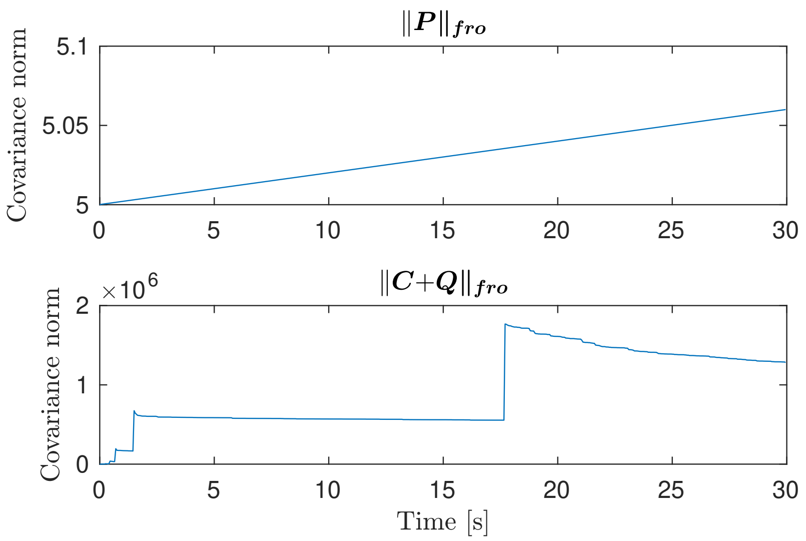

Figure 5.5 The Frobenius norm of the covariance matrices for both the Kalman filter regression (top) and the TV-SOGP regression (bottom).

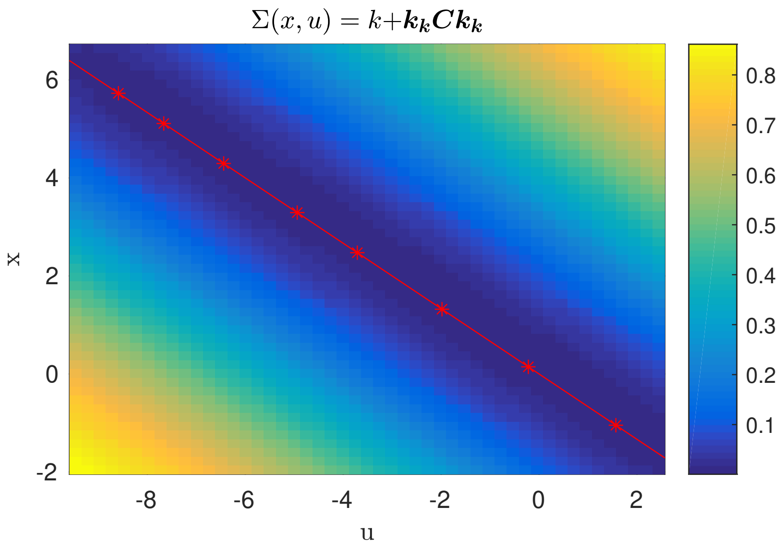

Figure 5.6 Predictive covariance (see 2.32) of the TV-SOGP regression over the covered input space (x_k, u_k).

5.2 Quadrotor simulations

Attitude Model Reference Adaptive Control

To show the performance of the MRAC scheme for the attitude dynamics it is separated from the positional dynamics. This, however, raises the question of what the total thrust would be during the simulation since the uncertainties described under Section **2.2** will depend on the individual trusts T_i during the simulation. To this end the total thrust close to the value at hover will be used $T = mg + 0.01\sin(\pi t)$. This trust will not affect the MRC controller but add unstructured effects in the model error for the GP to handle. This choice of the signal can be somewhat validated in MPC simulations in the next section. This only aims to make the model error unstructured with respect to the GP inputs $[\omega_x,\ \omega_y,\ \omega_z,\ \tau_x,\ \tau_y,\ \tau_z]$ as in a real-world problem.

One simulation will be shown with two uncertainty changes. The first change will occur in the moment of inertia and the second in an external wind speed entering through the flap and drag dynamics (see Figure 5.7). The input quaternion reference is a randomly generated steps in three dimensions that is low-pass filtered. The legend in Figure 5.7 shows the color coding used for all three-dimensional vectors plotted below. An approximative low-pass character rotor dynamics are included. This introduces a phase delay between the command torque and the actual torque. This will limit the performance of the controller. It also makes the calculated model error unacceptably inaccurate. Because of this, the torque is assumed to be known exactly when calculating the model error. This would imply that the rotor speeds are measured, which is not possible for the Crazyflie platform.

The uncertainty matrices $\boldsymbol{C}$ and $\boldsymbol{D}$ in (2.147) were chosen to

$$\boldsymbol{D} = \begin{bmatrix} 0 & -0.1 & 1 \\ 0.1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} 10^{-8} \tag{5.4}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} 10^{-7}. \tag{5.5}$$

These values are believed to be exaggerated compared to their true values, but chosen so that the performance of the controller can be evaluated in extreme conditions.

The settings and parameter choices used in the MRAC and TV-SOGP can be found in Table 5.1 and 5.2, respectively. The GP-MRAC scheme will here be tested during changes in the moment of inertia and a wind disturbance. The changes will take place at different times. The changes can be seen in Figure 5.7. The performance of the GP-MRAC tracking error can be seen in Figure 5.9 and should be compared to the MRC under the same disturbances

Table 5.1 Parameters choices for MRAC

Parameter	Value
Ts (sample time)	1/300 [s]
$\boldsymbol{K}_q^r$	diag([20, 20 ,20])
$\boldsymbol{K}_w^r$	diag([50, 50 ,50])
$\boldsymbol{K}_q^{pd}$	diag([10, 10 ,10])
$\boldsymbol{K}_w^{pd}$	diag([7, 7 ,7])

Table 5.2 Parameters choices for TV-SOGP. To make $\boldsymbol{\omega}$ and $\boldsymbol{\tau}$ on the same order of magnitude, $\boldsymbol{\tau}$ is multiplied by 10^5.

Parameter	Value
input variables	$[\omega_x,\ \omega_y,\ \omega_z,\ \tau_x 10^5,\ \tau_y 10^5,\ \tau_z 10^5]$
σ_k	$\begin{bmatrix}15 & 15 & 15 & 15 & 15 & 15\end{bmatrix}$
Ts (sample time)	1/300 s
ϵ_{tol}	0.001
n_{max} (budget)	60
σ_n	0.05
σ_p	0.0001

which can be seen in Figure 5.10. It can be seen that the disturbance in the form of a moment of inertia change does not give a stationary error. The error following from a constant wind will however result in a stationary error seen in Figure 5.10. Introducing the GP adaptive element however resolves the stationary error, seen in Figure 5.9.

In Figure 5.11 the estimated model error can be seen together with actual model error. Even in stationary small deviations between the estimated model error and the true value can be seen. This is because of the unstructured nature of the flap and drag uncertainty and possibly because a large length scale of the kernel is chosen. Note that the speed $\dot{\boldsymbol{p}}$ is assumed to be zero here. Introducing $\dot{\boldsymbol{p}}$ would add even more unstructured uncertainty in the model error.

The number of kernel centers can be seen in Figure 5.12. The budget of 60 is not reached. This is due to the wide kernel chosen. The choice of this wide kernel is necessary for making the GP-MRAC feasible but it works since the underlying model error function is slowly varying with respect to the covered state space. When studying the projection error in the bottom graph of Figure 5.12 and comparing the peaks in the error to the events of adding new observations to $\mathcal{BV}$ in the top graph, it is possible to see that the KLI test efficiently keeps the projection error below the specified value $\epsilon_{tol} = 0.01$. This is done by adding new kernels when the KLI test is violated.

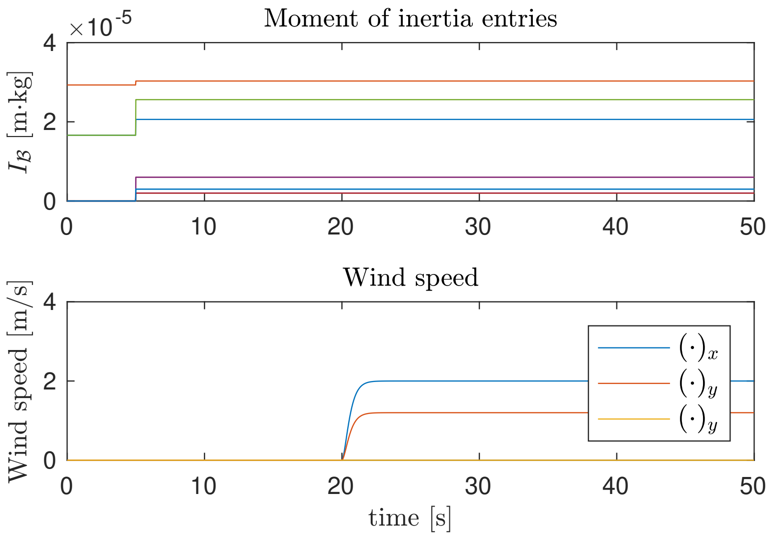

Figure 5.7 Top graph shows the moment of inertia matrix entries over time. The bottom graph shows the wind speed in 3 dimensions.

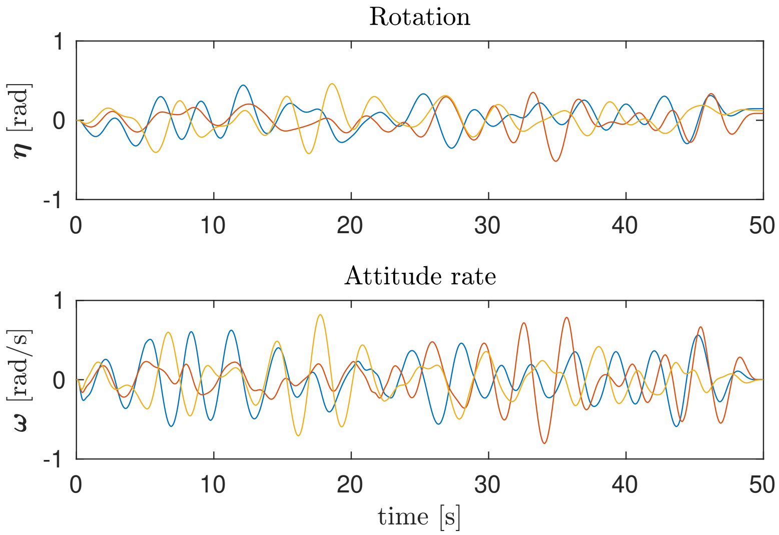

Figure 5.8 The states $\boldsymbol{\eta}$ and $\boldsymbol{\omega}$ when following the random trajectory during adaptive control with GP-MRAC.

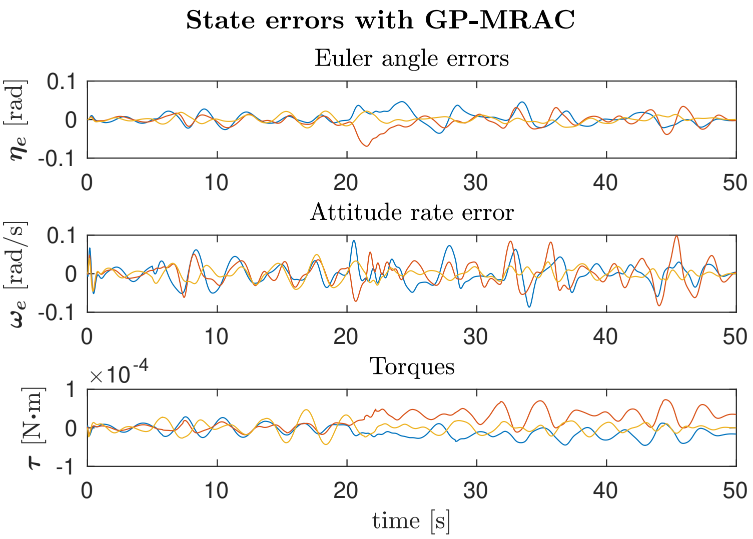

Figure 5.9 State errors for MRAC with GP adaptive element. Top graph shows the error between the reference model and the actual Euler angles. The middle graph shows the error between the reference model attitude rates and the actual attitude rate. The bottom graph shows the control torques.

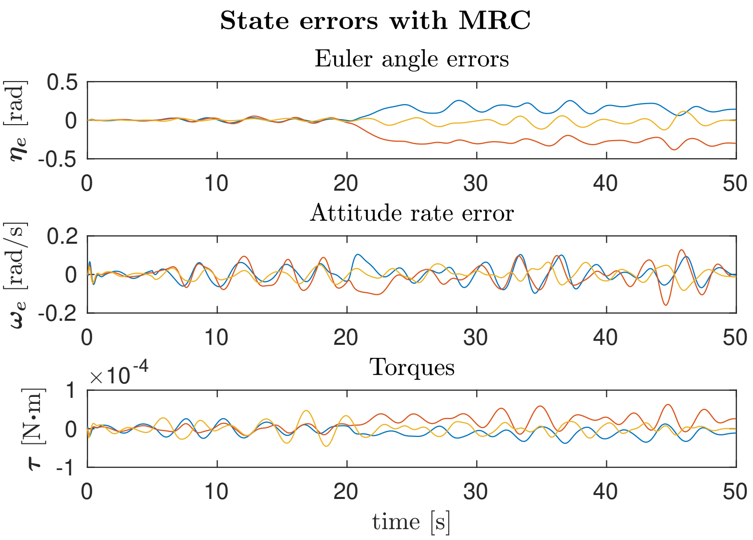

Figure 5.10 State errors for MRC without GP adaptive element.

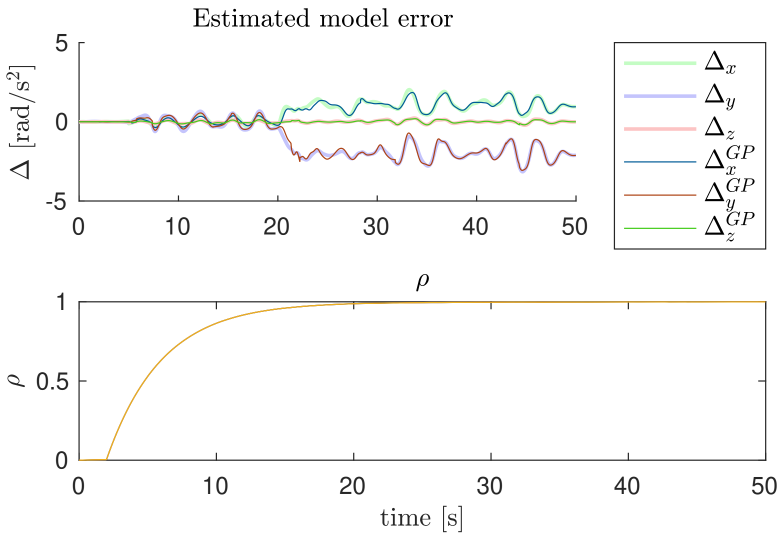

Figure 5.11 The top graph shows the GP estimated model error. The bottom graph shows the low-pass filtered ρ gain.

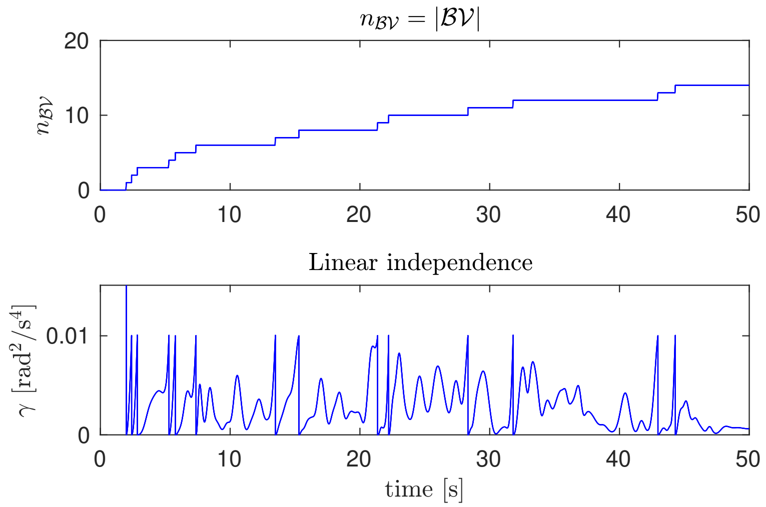

Figure 5.12 SOGP data over time. The top graph shows the number of observations added to $\mathcal{BV}$ and the bottom graph shows γ_{k+1}. The tolerance ϵ_{tol} is 0.01 which can be seen on the highest peaks.

Adaptive position MPC

Two trajectory simulations will here be shown to demonstrate the performance of the GP-MPC controller under structured flap and drag uncertainties and unstructured wind disturbances. The first simulation shows a figure-8 trajectory that visualizes the accuracy achieved. This periodic trajectory is designed to visualize the learning over time in a good way. However, since the state space is only visited along the specific trajectory it does not show if the GP-MPC would work in a setting where a larger part of the state space is visited. To this end, a second simulation is also shown where a random trajectory is simulated. The trajectory is generated through low-pass filtered steps in three dimensions. This also shows that the GP-MPC scheme can be feasible for more general flight maneuvers. Both of the simulations are designed so that the TV-SOGP design is necessary for the adaption to wind changes.

The simulations are done with an approximate model of the rotors consisting of a simple low-pass filter. Since the rotor dynamics are included it adds a low-pass dynamic that delays the actual trust from the given thrust command. The model errors in the attitude dynamics are neglected but the flap and drag in the position dynamics are included with the parameters $\boldsymbol{A} = \text{diag}([2\ 2\ 0])10^{-5}$ and $\boldsymbol{B} = \text{diag}([0\ 0\ 1])10^{-7}$. The values are only chosen to give visible model errors and are not claimed to be realistic in magnitude.

The designed MRC controller is used to control the attitude dynamics, but with slightly different settings because the adaptive element is turned off. Since the GP computational time was the main reason for having a longer sample time of 1/300 in the GP-MRAC simulation a shorter sample time of 1/500 will now be used. The MRC controller will also have slightly different parameters seen in Table 5.4. The input to the GP adaptive element was chosen to $[v_x,\ v_y,\ \phi,\ \theta,\ \psi,\ T]$ with the corresponding hyperparameter length scale vector [0.5, 0.5, 0.3, 0.3, 0.3, 0.07]. All parameter choices for the MRC and the MPC can be seen in Tables 5.3 and 5.4, respectively.

Table 5.3 Parameters choices for MPC

Parameter	Value
H	15
Ts (sample time)	0.1 [s]
$\boldsymbol{Q}$	diag([10, 10, 10, 15, 15, 15, 10, 10, 10, 10, 10, 10, 10])
$\boldsymbol{Q}_f$	diag([10, 10, 10, 15, 15, 15, 10, 10, 10, 10, 10, 10, 10])
$\boldsymbol{R}$	diag([3, 3 ,3 ,3])

Table 5.4 Parameters choices for MRC

Parameter	Value
Ts (sample time)	1/500 [s]
$\boldsymbol{K}_q^{ref}$	diag([20, 20 ,20])
$\boldsymbol{K}_w^{ref}$	diag([50, 50 ,50])
$\boldsymbol{K}_q^{pd}$	diag([10, 10 ,10])
$\boldsymbol{K}_w^{pd}$	diag([5, 5 ,5])

Figure-8 trajectory The following simulations show the quadrotor following a figure-8 movement under sudden wind disturbance (see Figure 5.13). The simulations are done without any noise and the wind load disturbance is smooth in a way that the GP model error estimation can be analyzed easily. The legend in Figure 5.13 shows the color coding used for all three-dimensional vectors plotted below. The SOGP adaptive element is here designed so that the budget will not be reached and a small kernel width is used compared to what is believed to be optimal. This choice is to show the performance of the TV-SOGP when no reallocation of kernels is done. All SOGP parameters can be seen in Table 5.5.

Table 5.5 Parameters choices for SOGP

Parameter	Value
input variables	$[v_x,\ v_y,\ \phi,\ \theta,\ \psi,\ T]$
σ_k	[0.5, 0.5, 0.3, 0.3, 0.3, 0.07]
Ts (sample time)	0.1 [s]
ϵ_{tol}	0.001
n_{max} (budget)	200
σ_n	0.05
σ_p	0.01

In Figure 5.14 the quadrotor trajectory can be seen with and without the GP adaptive element during the wind disturbance. It is a clear improvement when using GP estimation. The chosen process noise of $\sigma_p = 0.01$ also makes the controller attenuate the wind disturbance very fast (see Figure 5.16). This makes the quadrotor following the reference trajectory closely without venturing into new regions of the state space. This can be seen in Figure 5.15 where the number of kernel centers is approximately 60 after convergence. This can also be seen in the middle graph where γ is kept under ϵ_{tol} during most of the experiment. Adding new kernels online is always a sensitive thing in a controlled setting as will be seen later. When adding new kernel centers at the start of the experiment the gain ρ makes the influence of the GP adaptive element small. This makes the initial transients from noisy GP estimation smaller. The influence of the gain during the simulation is minimal.

In Figure 5.17 the estimated model error can be seen. The adaption to the model error change after the wind disturbance is very fast. It is questionable whether the high process noise compared to the measurement noise is reasonable in a real-time implementation. At the same time, it is hard to compare the process noise choice to parametric Kalman filter regression. Since forgetting only affects the close neighborhood, it is reasonable that higher process noise is necessary. The question of whether the forgetting is too high or not will also depend on the measurement noise on the model error observation. In this simulation, no measurement noise is present.

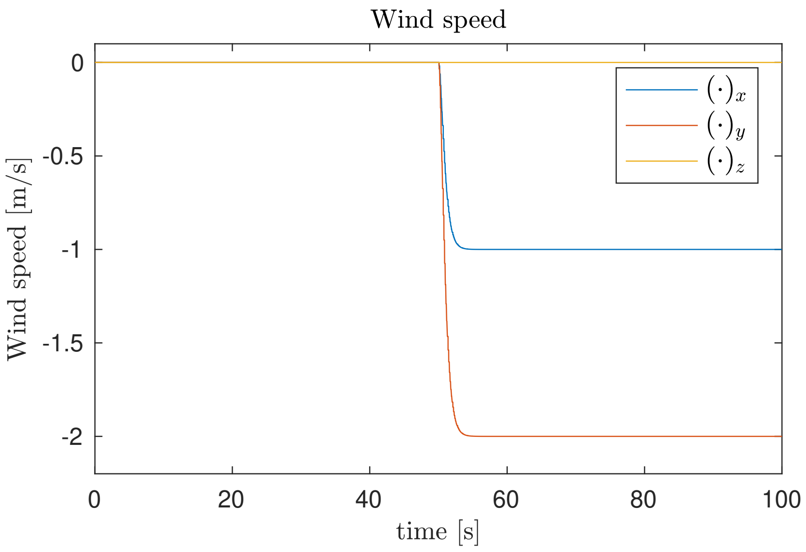

Figure 5.13 Wind speed during figure-8 flight trajectory.

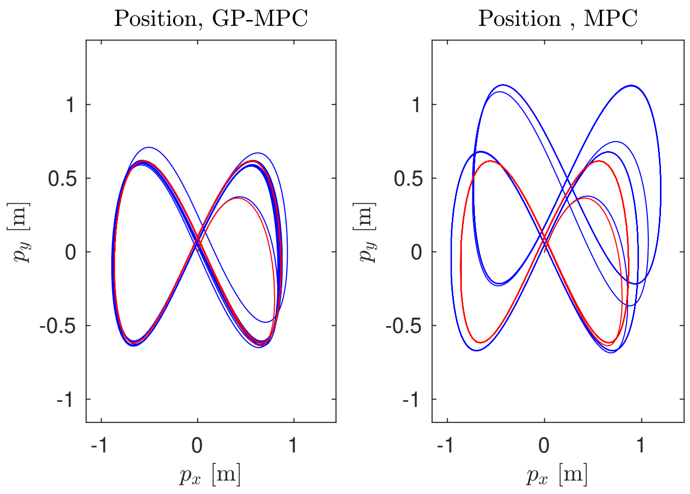

Figure 5.14 Simulation trajectories for adaptive GP-MPC (left) and MPC (right) without adaptive element under wind disturbances seen in Figure 5.13. Red line represents the desired trajectory and the blue line represents the actual simulated trajectory.

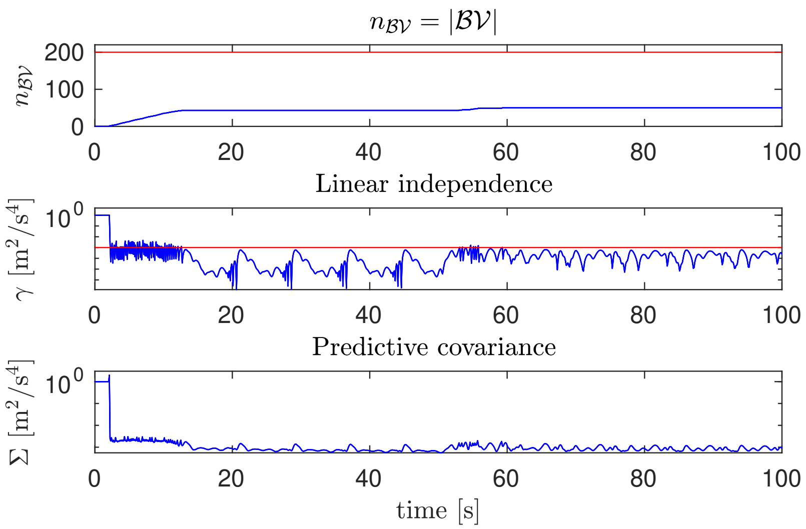

Figure 5.15 GP The top graph shows the number of samples in $\mathcal{BV}$ in blue and the maximum limit of the set (budget) in read. The second graph shows the KLI test value γ_t in blue (logarithmic scale) and the tolerance in ϵ_{tol} in red. The bottom graph shows the predictive covariance in the current state space point (logarithmic scale).

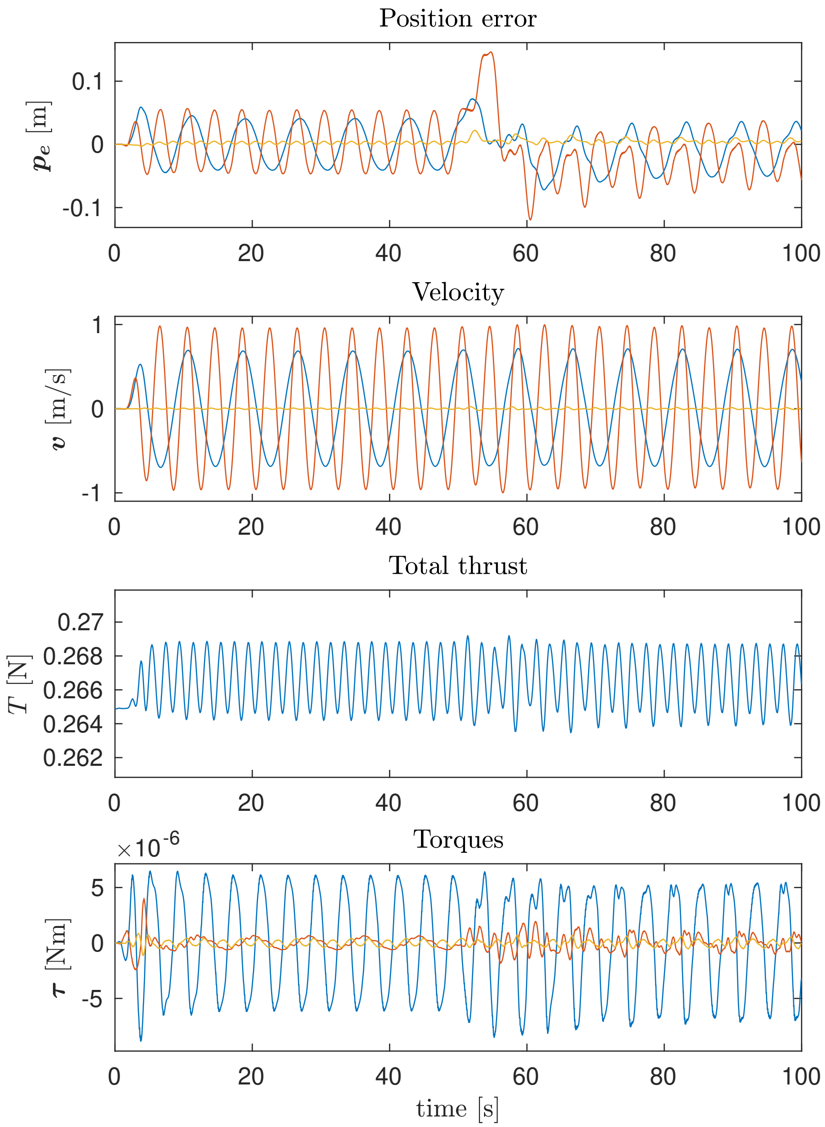

Figure 5.16 The top graph shows the difference between reference trajectory and the actual position. The second graph show the actual speed. In the third graph the total trust is shown and lastly the the bottom graph shows the torques.

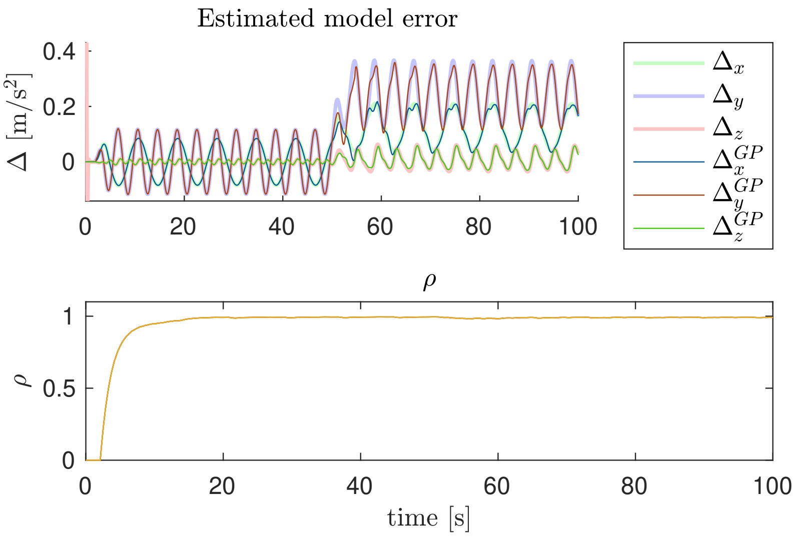

Figure 5.17 The top graph shows the estimated model error and the actual model error. The bottom graph shows the ρ gain which limits the adaptive element gain.

Process noise validation To show the importance of the designed process noise in the TV-SOGP the same experiment with a figure-8 trajectory was performed with different process noise σ_p. The same wind disturbance as in Figure 5.13 was used. The result for $\sigma_p = 0.01$, $\sigma_p = 0.001$ and $\sigma_p = 0$ can be seen in Figure 5.18. It is clear that higher process noise makes the reaction to a wind disturbance step faster. For $\sigma_p = 0.001$ the reaction is a bit slower, but other things can also be seen in the resulting estimate. If one looks closely on the estimated model error it possible to see small oscillations around the time of the wind disturbance step. This is not because of the Kalman filter in the sparse update but because the quadrotor ventures into new regions and allocates new kernel centers. When the kernel centers are added the estimate may change drastically in the current point. This is an effect that always will be present when venturing into new regions of the state space, but it will be larger when the speed of the measurement "agent" is fast compared to the sampling rate. In the bottom graphs where no process noise is present at all, these oscillations get even worse and the system is on the verge of getting unstable. Note that the system would not become unstable because of the wind disturbance if the adaptive element was turned off altogether (as can be seen in Figure 5.14). The oscillations in the GP estimate is the most likely cause for the aggressive behavior for $\sigma_p = 0$.

The reason for the two first experiments (with $\sigma_p = 0.01$ and $\sigma_p = 0.01$) being stable is hence that the forgetting is rapid enough to prevent the system from venturing into new areas and hence preventing oscillations in the estimate.

The oscillations can, however, be prevented to some extent by other methods than high process noise. In Figure 5.19 the KLI projection error γ and the dimension of the SOGP $n_{\mathcal{BV}}$ can be seen for the same experiment with different process noises. The graphs suggest that smaller process noise leads to a higher exploration of the state space since the dimension increases much more at the time of the disturbance entrance. In the projection error graph, something even more interesting happens. The projection error γ exceeds the tolerance ϵ_{tol} by far, especially in the last graph. This means that during one sample time the system has ventured too far out into new regions without adding new kernels, suggesting that the sample time is too long compared to the kernel width. This is the main reason for the estimation becoming noisy as can be seen in Figure 5.18. If a shorter sample time is not wanted this suggests a greater kernel width. This shows another danger with too short kernel widths other than overfitting the underlying function.

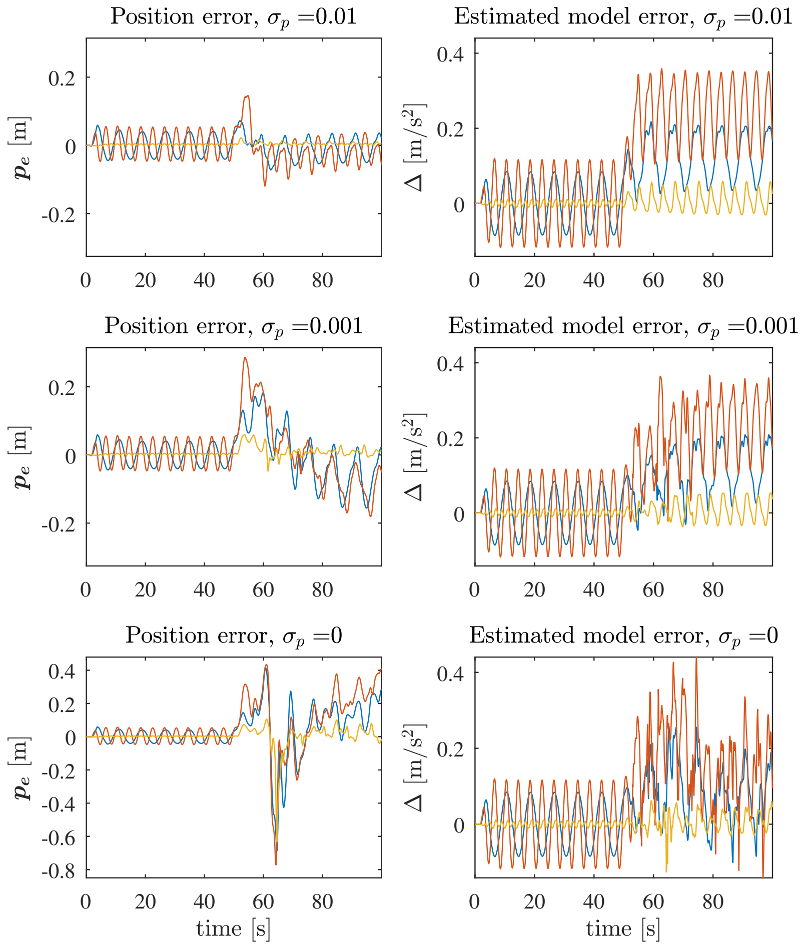

Figure 5.18 Comparison of different choices of process noise during the figure-8 simulation. Note the different scale in the position error graph for $\sigma_p = 0$.

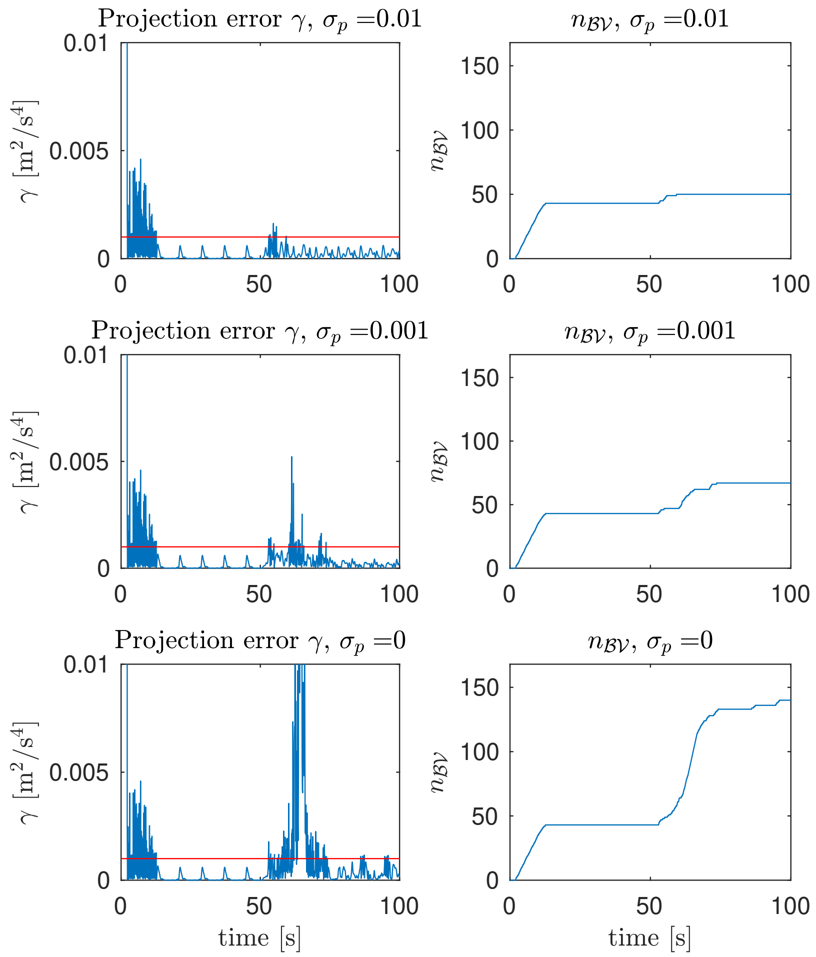

Figure 5.19 Comparison of the projection error γ and the cardinality of $\mathcal{BV}$ for different choices of process noise during the figure-8 simulation.

Random trajectory In this section simulation results from a flight over a randomly generated trajectory during wind disturbances are shown. The random trajectory is chosen to show that the GP-MPC scheme is not only feasible for periodic trajectories.

In previous GP-MPC simulations, the kernel width was chosen short compared to what was believed to be optimal. A hyperparameter optimization of the simulation data in this experiment has been tried but it renders in solutions of the length scale hyperparameter $\sigma_k > 300$. At hyperparameter values of these magnitudes, the cost function will be ill-conditioned since the distance between observations will be too small, suggesting that the data is not exciting enough to perform hyperparameter optimization. What could be concluded from the optimization is that the gradient is positive at the initial value chosen as the hyperparameter vector in Table 5.5. A positive gradient suggests that the length-scale parameter should be increased. However, since the optimum could not be found with the considered quasi-Newton algorithms, the hyper-parameters were tuned manually (see Table 5.6). The higher kernel length scale will cause the GP to allocate fewer kernel centers. If no bound is put on the $\mathcal{BV}$ set, the simulations below would render in $n_{\mathcal{BV}} \approx 60$. However, to make the simulation more interesting a budget of $n_{max} = 20$ was chosen so that the SOGP algorithm has to dynamically reallocate kernels centers. All other parameters of the SOGP for this simulation can be found in Table 5.6. The wind is now modeled as a Wiener process added to a step function (see Figure 5.20). This is a more realistic wind disturbance that will be harder to follow for the GP. It can, however, be seen that the position error is kept down in a good way in Figure 5.23 considering the rather aggressive wind changes. From Figure 5.23 it is also clear that the control signals are smooth even with rather fast and noisy changes in the wind.

Table 5.6 Parameters choices for SOGP

Parameter	Value
input variables	$[v_x,\ v_y,\ \phi,\ \theta,\ \psi,\ T]$
σ_k	[2.5, 2.5, 1.5, 1.5, 1.5, 0.1]
Ts (sample time)	0.1 [s]
ϵ_{tol}	0.001
n_{max} (budget)	20
σ_n	0.05
σ_p	0.001

In the graph of the estimated model error in Figure 5.21, it can be seen that the estimated model error follows the error, quickly adapting to the abrupt change in error induced by the change in wind speed. Most of the high-frequency content is however smoothed out. In the same figure, the gain ρ can be seen to stabilize close to one.

In the top graph of Figure 5.22 showing the cardinality of $\mathcal{BV}$, it can be seen that $|\mathcal{BV}| = n_{\mathcal{BV}}$ is kept below 20 s. This forces the SOGP to reallocate kernel centers dynamically after $t \approx 20$. The reallocation will happen each time γ hits the tolerance ϵ_{tol}. This happens several times as can be seen in Figure 5.22.

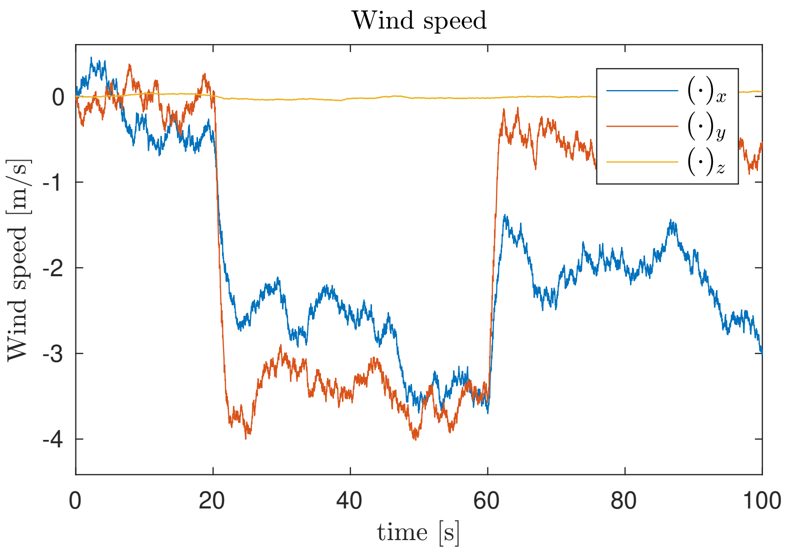

Figure 5.20 Wind disturbance in form of steps and an additive Wiener process with noise covariance matrix diag$([16, 16, 0])$.

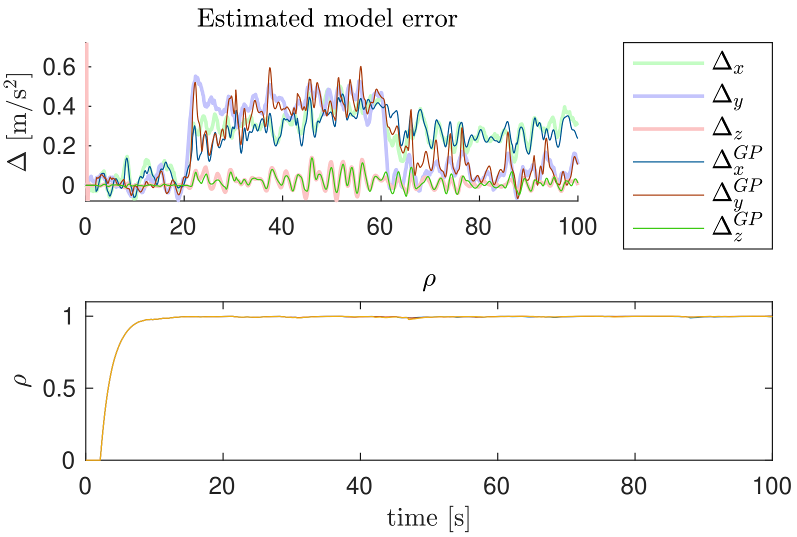

Figure 5.21 In the top graph a comparison between the model error and the estimated model error can be seen. The model error seen by the MPC is scaled with the gain in the bottom graph.

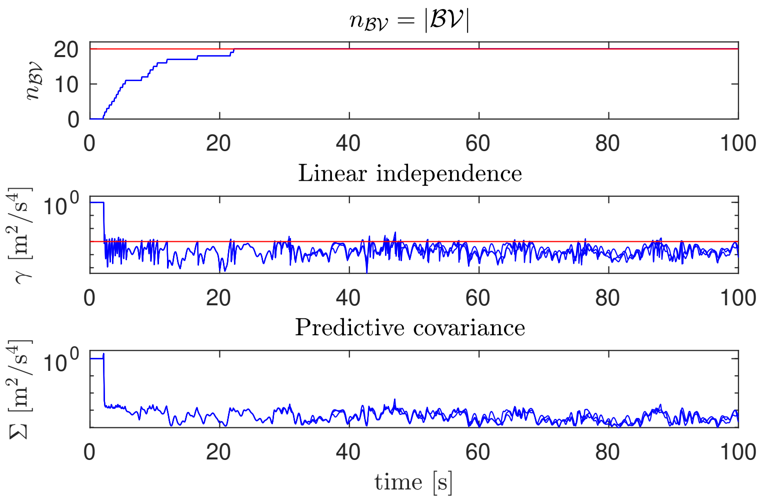

Figure 5.22 GP data along the trajectory. The top graph shows the number of samples in $\mathcal{BV}$ with the budget marked in red. The middle graph shows the KLI test measure γ togheter with the limiting tolerance ϵ_{tol} marked in red. The bottom graph shows the predictive covariance over time.

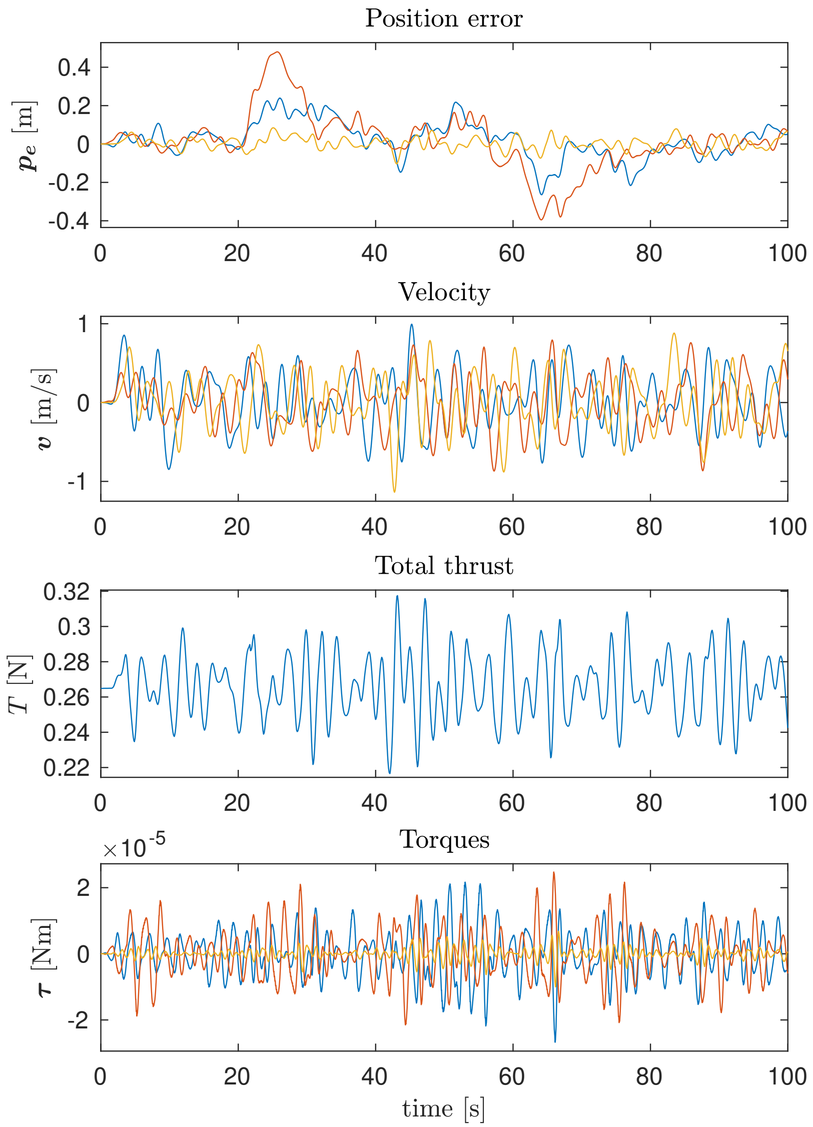

Figure 5.23 States and controls during the experiment. The top graph shows the error between the reference position and the actual position.

Profiling To validate the controller for real-time schemes the computation times of the MPC and the SOGP are here profiled. The histogram for two different choices of the size of the set $\mathcal{BV}$ and the MPC can be seen in Figure 5.24. The profiling time for SOGP only shows the result for one GP regression. In the GP-MPC three GP regressions are used, but as can be seen, the computation time of one SOGP iteration never exceeds 0.004 s if the budget is chosen to 200. The total computation time of both the MPC and the SOGP will therefore with very high probability not exceed 0.015 s. Since the sample time of the MPC was chosen to 1/10 s this is well within the deadline. The code for SOGP was implemented in Matlab and the generated MPC C-code was called through a MEX file in Matlab. The computer used was a Mac 1.5 GHz Intel Core i5 processor with 4 GB memory.

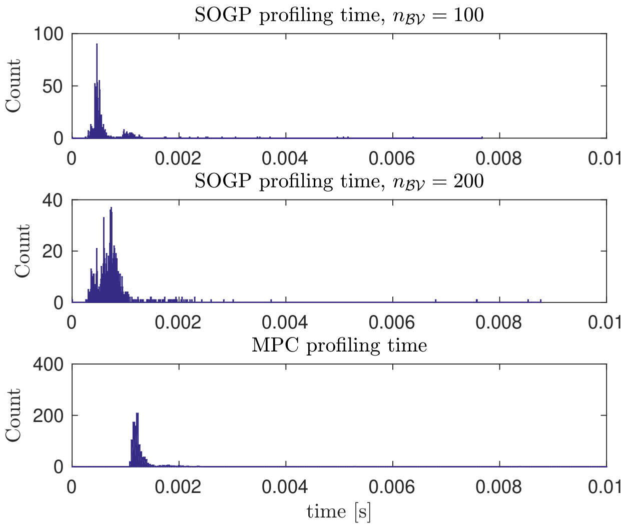

Figure 5.24 Profiling times for the SOGP in the top two histograms and for the MPC in the bottom histogram. The profiling of the SOGP includes both prediction and update for two different choices of budget on $\mathcal{BV}$.

6 Discussion

The aim of the thesis was to investigate different sparse formulations of GPs. The main focus has however been a single sparse approximation, namely SOGP. The reason for this is that the nature of this approximation is sequential, which is beneficial for online learning. To my knowledge, other sparse approximations do not aim to learn online but rather reduce the complexity of the learning in a batch setting (see for example [Snelson and Ghahramani, 2006]).

While the few previous articles on adaptive control with GPs [Maciejowski and Yang, 2013], [Chowdhary et al., 2013a], [Grande et al., 2014] have incorporated time-varying learning in different ways, it was found that improvements could be done. Different sparse approximations allow for different solutions to the problem of time-varying functions. Previous work has used methods resulting in a sliding window of allocated kernels. In [Maciejowski and Yang, 2013] a batch strategy was used so that the batch could be selected as a sliding window. In the more related work of [Grande et al., 2014] a time input was used. The time input essentially results in the same strategy since the budget of SOGP will be exceeded faster and the oldest sample will be deleted. While these methods use a very intuitive way of incorporating forgetting, the work done in this thesis proves that forgetting can be done in a numerically much less expensive way. This is especially true if a very slow forgetting is wanted. A longer sliding window would thus be needed which in turn means that the number of kernels must be increased too. This is completely avoided in the TV-SOGP algorithm.

Another interesting feature is the locality in the forgetting of the TV-SOGP, which does not change the behavior of the GP to the same extent as a time input does. Since the information in the whole state space is not penalized by the process noise of one Kalman update, an interesting question is how the sensitivity to low PE is affected. Since the TV-SOGP uses a forgetting mechanism that only affects the covariance locally, information far away from the current observation in the input space is preserved to a larger extent. This, in turn, means that information that was collected a long time ago does not necessarily get “thrown away”. Only information about regions in state space where new information is collected is forgotten. The local forgetting should be held in contrast to parametric regression where one regressor often has a global influence. This effect can be seen in Section **5.1**. This feature of TV-SOGP is a great advantage when designing adaptive controllers. Since PE is not needed for the TV-SOGP estimate to converge in the region where samples have been collected the controller gets enough information to control the plant in the close surroundings of the observations. If instead a parametric model would be used the covariance will diverge over time making the adaptive controller very sensitive to low PE.

The dynamics of the quadrotor was investigated by using a nominal model commonly used in literature for control and simulation of quadrotors. The aim of the thesis is, however, to perform adaptive control of the quadrotor dynamics with unknown uncertainties. Parametric uncertainties in the nominal dynamics constitute one such example. However, since one of the benefits of GP regression is its flexibility to estimate functions of unknown structure it is an advantage to show the ability to adapt to completely un-modeled dynamics. To this end, flap and drag uncertainties were investigated. The flap and drag effects are also very relevant, especially for outdoor-flight where wind might be present. When analyzing what inputs should be included to model these uncertainties, it was realized that a full model of the drag and flap effects could not be hoped to be identified online with a GP model. The reason for this is that the GP regression problem scales exponentially in complexity with the input dimensions. Already for the nominal dynamic estimation, an input dimension of six was needed. This dimension size is believed to be on the verge of what is feasible for online control of the quadrotor. This casts light on the big limitations of GP regression. Since a bound on the $\mathcal{BV}$ set is needed, allocating enough kernel centers to cover the input space will in many cases not be possible. In the model error plotted in Figure 4.1, six input dimensions are needed. If only three kernel centers would be placed along each dimension in the visible grid, this would lead to $3^6 = 729$ kernel centers. This would hardly be feasible to run at 10 Hz together with the MPC on a desktop computer. The question is then if the dynamic allocation strategy used in SOGP can be used to estimate the function only locally in the current region of state space. The possibility to learn new parts of the state space when the budget is exceeded will imply that information in another part of the state space is forgotten. This possibility, to dynamically allocate new kernels is important and analysis of the behavior of the SOGP in these situations could be investigated further. This thesis has mainly been focusing on the problem where a sufficiently small part of the state space is visited to stay within the budget. In Figure 5.18 this problem is indirectly seen as oscillations in the estimate. Designing robust control while venturing into new state space regions needs to be investigated further. The simple attempt to use the predicted covariance to design the gain ρ (as proposed in [Grande et al., 2014]) showed that it might help in the start of the experiment where new kernels are allocated. The usefulness of the gain when venturing into new regions is however very limited. As can be seen in all simulations, the gain is almost constant over the experiment, even for the simulation in Figure 5.21 when the system dynamically reallocates kernels when venturing into new regions. The reason for this is that the parameter σ_f expressing the certainty of the nominal model is chosen high. This is unfortunately necessary when using the TV-SOGP algorithm if a value of $\rho = 1$ is wanted in stationarity. This is mainly because the predictive covariance indirectly will be limited by ϵ_{tol}, since the covariance is very related to the projection error γ. This means that the variation of the covariance always will be small if the SOGP succeeds to allocate kernels sufficiently fast. The gain ρ will, therefore, be of little help during slow explorations but could help if the kernel width is too narrow compared to the exploration speed, as was the case for the experiment in Figure 5.19.

Another problem is that the gain needs to be low-pass filtered, which introduces a phase delay that will limit the estimate's ability to quickly react to changes in the uncertainty. The MPC framework could, however, provide a solution to this since predictions along the horizon could be used to predict future estimation uncertainties, introducing an overhead. This is a possibility that could be tested in future work. The main limitation of the gain ρ is that it only restricts the influence of the adaptive element. Intuitively a better alternative could be to change the controller to be more cautious in low certainty situations as discussed in Section **3.5**.

It is in general hard to give stability proofs for adaptive controllers. Further, this thesis has focused on adaptive controllers for unstable non-linear systems. This combines two complex fields and it is often hard to give any guarantees of stability. If the amplitude of the model error is not known it can be questionable whether any stability analysis is possible at all. This thesis does therefore not aim to give any proof but instead design controllers and verify them through simulations and real-time implementation. No implementation of the designed adaptive controllers could, however, be implemented in real time because of lack of time. This is a serious limitation in the validation of the proposed controllers. Simulation environments do only incorporate what is known of the system and real-time experiments will have to deal with un-modeled uncertainties, colored noise and load disturbances of largely unknown structure. This might, first of all, make the nominal controller perform worse than expected and further the model error calculation could be a problem. If the model error not can be assumed to represent the true error the hope of using it in adaptive control to result in a stable closed loop is rather naive. For the case of the position control, the model error can be measured through the accelerometer. This measurement is however obtained in body frame and needs to be rotated to the inertial frame. Hence good estimates of the rotation are needed, which assumes the rotations are observable through other measurements. In the case of the Crazyflie, these measurements consist of gyroscope and ultra-wideband position measurements [Greiff, 2017].

The model error calculation on the Crazyflie platform is however believed to be a serious problem. This is because the rotor controller only consists of a feedforward controller that sets a PWM signal to the motors that give the desired behavior in stationarity. This means that rather large phase delays will be present. When making the difference in (4.5) this might lead to significant errors. It is believed that this could be a serious problem in real-time implementations. Even in simulations, it proved to be hard to solve the model error calculation. The simulations done is assuming that the real values of the thrust are available. Without out this, the performance will be significantly reduced of the adaptive controller.

The GP-MRAC scheme has previously been validated in real time for a quadrotor [Grande et al., 2014]. This raises the question of how the problem of phase delay in the rotor dynamics was solved there. No explanation can be found in the article. A possible reason is that rotor controllers use feedback to lower the phase delay. It is, therefore, possible that a different platform than the Crazyflie could be very beneficial. The only solution available on the Crazyflie would be to run a model of the rotor dynamics on the quadrotor to approximate the real rotor speeds. If the model provided in [Greiff, 2017] is accurate enough for this would have to be investigated.

6.1 Limitations

The work done during the course of the thesis covers much of the background theory of the proposed controllers and GP regression. The validations are however more limited and much remains to be done. The simulations could be complemented with noise models and observers to more realistically simulate the Crazyflie platform and validate the controller under noise and load disturbances.

To fully validate the GP-MPC scheme a real-time implementation would be needed. The simulations made here, only prove the effectiveness of the proposed adaptive controller in a simulation environment. In a real-time setting, the controller needs to tackle colored noise and other disturbances. To this end, good observers have to be designed. Since the Crazyflie platform has a well developed open source firmware where this is included this has not been a focus of this thesis. The most pressing issue with the real-time implementation is believed to be the accuracy of the model error calculation on the Crazyflie platform as explained earlier.

The proposed controllers are in some sense not designed to perform as well as possible. They are designed to show the performance of the adaptive element. This means that no integral action has been included in any of the controllers. This is especially a limitation in the GP-MPC. Since the identified dynamics are linearised the controller sometimes showed signs of not driving the stationary error to zero. Another reason for this might be that the nominal dynamics are used to generate the reference trajectory. To this end, integral action could be added to further improve the controller.

While GPs for adaptive control certainly have possible advantages such as the ability estimate a wide range of unknown functions, providing predictive uncertainty measures and the ability to identify usable models with very little data there are also limitations. The most pressing is the complexity that scales exponentially with the inputs. However, for the model errors in the simulations of this document, the underlying function proved to have mostly low-frequency content allowing very wide kernels. In some situations when the model error structure is known, it can instead be very beneficial to use a parametric regression instead of GPs. These methods will, however, have problems for low order of PE which TV-SOGP does not have.

As seen when venturing into new regions of the state space the SOGP is forced to allocate more kernels as described earlier in the discussion. The stability and robustness in these situations are not well explored in this work. It was observed that it might be a big problem in some situations because of noisy estimates.

7

Conclusions

TV-SOGP The work on GP-MRAC by [Grande et al., 2014] includes some attempts of doing time-varying estimation by using time as an input to the SOGP. These attempts have some limitations and do not exploit the locality in GP learning (see Section **2.1**). In addition, the $\mathcal{BV}$ set, has to be larger when having one additional input, increasing the dimensionality and numerical complexity significantly. The main contribution of this thesis has been to design a time-varying SOGP that does not suffer from the above-mentioned limitations. This was done by introducing a time-varying Kalman filter in the "sparse update" and enables time-varying learning without deleting kernel centers, resulting in more smooth learning which is beneficial in a control setting (see Section **5.1**). It was also shown that the TV-SOGP performed better in terms of smooth adaption compared to a SOGP with a time input in situations where the budget is chosen too restrictive. The main advantage of TV-SOGP is, however, that one less input is needed. This will greatly improve the numerical complexity of the algorithm since a smaller $\mathcal{BV}$ set is needed.

One of the greatest advantages of TV-SOGP compared to parametric models in an adaptive control setting is that there is no need for PE to get valuable estimates of the dynamics locally in the input space. The price for this is, however, higher computational complexity compared to parametric models.

A problem that was observed with SOGPs in the adaptive element was the sometimes noisy learning of the model error. This is however not a problem originating from high process noise in the Kalman filter in TV-SOGP, but rather from the SOGP algorithm itself. Since the kernel centers are placed online a new kernel will possibly change the estimation drastically if the new observation is far from the old ones. This is something that is a clear disadvantage of GP learning in a control setting. The effect can, however, be reduced by choosing a large length scale hyperparameter if the underlying function permits or increasing the sample time. Even if the drastic changes may lead to better estimation it might make the system unstable. Even worse is that these discontinuities in the estimation happen in sensitive situations when the dynamics venture into new regions of the state space, see Figure 5.18. Using the uncertainty prediction to solve this problem is an interesting direction of possible future work.

Controller structure The simulations of the proposed GP-MRAC scheme for the attitude dynamics showed that the GP adaptive element efficiently could attenuate wind load disturbances but also adapt to structured uncertainties in the inertia tensor, drag, and flap dynamics. Since the plant model is perfectly known in simulations, tuning of the hyperparameters can easily be done. In a real-time setting however it could be hard to collect data that is exciting enough for hyperparameter optimization. In this specific problem, the length scale hyperparameters had to be chosen large to allow for a tractable SOGP running at 300 Hz. The large length scale was however validated through hyperparameter optimization of the parametric inertia tensor model error. Since the problem of numerical tractability often is a more pressing issue, this will call for wider kernels that might be suboptimal. This is especially true for controllers that must run at high rates. It is questionable if the GP-MRAC scheme can run onboard the Crazyflie platform in a sufficiently high sample rate for the attitude controller. The results here only show that it is feasible to run with a desktop computer in the loop.

Designing an inversion-based MRAC for the position dynamics is a suboptimal solution in the sense that the dynamic model is overdetermined and hence not invertible. This would call for an approximate inversion which does not draw full benefit from the nominal model. Instead, a solution using MPC to control the positional dynamics was used. The MPC controller was designed to operate at 10 Hz. The rather slow sample rate allows for more possibilities and freedom to choose the hyperparameters. The simulations suggest that reasonably large part of the state space can be estimated through GP regression with a sufficiently small $\mathcal{BV}$ budget to run on a desktop computer in real-time. The simulations also show that the adaptive controller is good at attenuating changes in the dynamics. For this to be possible a large process noise was chosen compared to the measurement noise. It is possible that lower process noise has to be chosen in a real-time controller. Since the forgetting is only local it is, however, reasonable to expect that a lager process noise is needed compared to a parametric Kalman regression.

The simulations here show that GP-MPC has the potential to provide adaptivity and can be made numerically tractable for many problems if the number of inputs is limited. This is especially true for GPs representing distributions of functions with low-frequency content.

7.1 Future work

To clearly state interesting directions for future work, a brief summary of the points made in earlier sections is given below.

- As pointed out in [Maciejowski and Yang, 2013] a sophisticated way of using the uncertainty would instead be to change the optimization problem to be more cautious in situations where there are high uncertainties in the function estimate. Earlier work has been done exploiting this in offline identified GP dynamic models (see for example [Cao et al., 2017]). The problem does, however, suffer from the propagation of uncertainties that become unfeasible in many real-time settings. To my knowledge, no attempts to use these methods have been done for online GP identification.

- Since GP regression incorporates prediction uncertainties a possibility that is interesting could be to use the uncertainty function for intelligently exciting the system to gain information in uncertain regions of the state space. This is a problem often referred to as dual control. GPs for dual control have been investigated before [Alpcan, 2011], but the field remains an open research question.

- To my knowledge, no GP-MPC has yet been tested in a real-time setting. The results in this thesis, however, show that it could be feasible in a real-time setting for reasonably large problems.

A

Model appendix

A.1 MPC model Linearisation

The linearisation of the nominal model designed for the MPC (see 3.36–3.38) results in an LTI system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_l \boldsymbol{x} + \boldsymbol{B}_l \boldsymbol{u} + \boldsymbol{G}_l \tag{A.1}$$

$$\boldsymbol{A}_l = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I}^{3\times 3} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{D}/m & \boldsymbol{A}_\eta & \boldsymbol{A}_T \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}^{4\times 4} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{13\times 13} \tag{A.2}$$

$$\boldsymbol{B}_l = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{I}^{4\times 4} \end{bmatrix} \in \mathbb{R}^{13\times 4} \tag{A.3}$$

$$\boldsymbol{G}_l = \begin{bmatrix} \dot{\boldsymbol{p}}_0 \\ \ddot{\boldsymbol{p}}_0 \\ \dot{\boldsymbol{\eta}}_0 \\ \boldsymbol{0}^{4\times 4} \end{bmatrix} \in \mathbb{R}^{13\times 1} \tag{A.4}$$

and

$$\boldsymbol{A}_\eta = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \tag{A.5}$$

$$A_{11} = \frac{T_0}{m}(\cos(\phi)\sin(\psi) - \cos(\psi)\sin(\phi)\sin(\theta)) \tag{A.6}$$

$$A_{12} = \frac{T_0}{m}\cos(\phi)\cos(\psi)\cos(\theta) \tag{A.7}$$

$$A_{13} = \frac{T_0}{m}(\cos(\psi)\sin(\phi) - \cos(\phi)\sin(\psi)\sin(\theta)) \tag{A.8}$$

$$A_{21} = -\frac{T_0}{m}(\cos(\phi)\cos(\psi) + \sin(\phi)\sin(\psi)\sin(\theta)) \tag{A.9}$$

$$A_{22} = \frac{T_0}{m}\cos(\phi)\cos(\theta)\sin(\psi) \tag{A.10}$$

$$A_{23} = \frac{T}{m}(\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)) \tag{A.11}$$

$$A_{31} = -\frac{T_0}{m}\cos(\theta)\sin(\phi) \tag{A.12}$$

$$A_{32} = -\frac{T_0}{m}\cos(\phi)\sin(\theta) \tag{A.13}$$

$$A_{33} = 0 \tag{A.14}$$

$$\boldsymbol{A}_T = \begin{bmatrix} \frac{(\sin(\phi)\sin(\psi)+\cos(\phi)\cos(\psi)\sin(\theta))}{m} \\ \frac{-(\cos(\psi)\sin(\phi)-\cos(\phi)\sin(\psi)\sin(\theta))}{m} \\ \frac{\cos(\phi)\cos(\theta)}{m} \end{bmatrix} \tag{A.15}$$

Note the zeros in the vector $\boldsymbol{G}_l$, corresponding to linearising around $\boldsymbol{u} = \boldsymbol{0}$. This is not a restriction since this part of the dynamics is linear. The expressions are derived with Matlab's symbolic toolbox.

A.2 Discretization

The discretization of an LTI system of the form (A.1) through zero-order hold takes the form of a difference equation as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{u}_k + \boldsymbol{G}_d \tag{A.16}$$

Zero-order hold discretization assumes that the input signal is piece-wise constant between samples to calculate the next sample prediction. By using the integrating factor $e^{-\boldsymbol{A}_l t}$ the linear dynamics in (3.40) become

$$e^{-\boldsymbol{A}_l t}\dot{\boldsymbol{x}}(t) - e^{-\boldsymbol{A}_l t}\boldsymbol{A}_l\boldsymbol{x}(t) = \frac{d}{dt}(e^{-\boldsymbol{A}_l t}\boldsymbol{A}_l\boldsymbol{x}) = e^{-\boldsymbol{A}_l t}(\boldsymbol{B}_l\boldsymbol{u}(\tau) + \boldsymbol{G}_l). \tag{A.17}$$

The fundamental theorem of calculus then gives

$$e^{-\boldsymbol{A}_l t}\boldsymbol{A}_l\boldsymbol{x}(t) = e^{-\boldsymbol{A}_l t_0}\boldsymbol{A}_l\boldsymbol{x}(t_0) + \int_{t_0}^{t} e^{-\boldsymbol{A}_l t}(\boldsymbol{B}_l\boldsymbol{u}(\tau) + \boldsymbol{G}_l)d\tau. \tag{A.18}$$

By using the zero-order hold approximation of $\boldsymbol{u}$ over an interval $\Delta t = t_{k+1} - t_k$ and introducing the substitution $\tau = s + t_0$ the following discretization is achieved

$$x(t_{k+1}) = e^{\boldsymbol{A}_l \Delta t}\boldsymbol{x}_k + \int_0^{\Delta t} e^{\boldsymbol{A}_l \tau}\boldsymbol{B}_l d\tau \boldsymbol{u}_k + \int_0^{\Delta t} e^{\boldsymbol{A}_l \tau}\boldsymbol{G}_l d\tau. \tag{A.19}$$

Identification of terms gives

$$\boldsymbol{A}_d = e^{\boldsymbol{A}_l \Delta t} \tag{A.20}$$

$$\boldsymbol{B}_d = \int_0^{\Delta t} e^{\boldsymbol{A}_l \tau} d\tau \boldsymbol{B}_l \tag{A.21}$$

$$\boldsymbol{G}_d = \int_0^{\Delta t} e^{\boldsymbol{A}_l \tau} d\tau \boldsymbol{G}_l. \tag{A.22}$$

The calculation of the matrix exponential can be done by a series expansion for fast calculation in a real-time setting

$$e^{\boldsymbol{A}_l \Delta t} = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{\boldsymbol{A}_l^i \Delta t^i}{i!} \tag{A.23}$$

$$\int_0^{\Delta t} e^{\boldsymbol{A}_l \tau} d\tau = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{\boldsymbol{A}_l^i \Delta t^{i+1}}{(i+1)!}. \tag{A.24}$$

Bibliography

Alpcan, T. (2011). “Dual control with active learning using Gaussian process regression”. *arXiv preprint arXiv:1105.2211*. (Visited on 2019-05-27).

Åström, K. J. and B. Wittenmark (2013). *Adaptive control*. Dover publications, Mineola, New York. ISBN: 0486462781.

Bangura, M., M. Melega, R. Naldi, and R. Mahony (2016). “Aerodynamics of rotor blades for quadrotors”. *arXiv preprint arXiv:1601.00733*. (Visited on 2019-05-27).

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. PhD thesis. The Gatsby Computational Neuroscience Unit, University of London, 17 Queen Square, London.

Bernard, D. D. C., F. Riccardi, M. Giurato, and M. Lovera (2017). “A dynamic analysis of ground effect for a quadrotor platform”. *IFAC-PapersOnLine* **50**:1, pp. 10311–10316.

Brescianini, D., M. Hehn, and R. D’Andrea (2013). *Nonlinear quadrocopter attitude control*. Tech. rep. 7387. Institute for Dynamic Systems and Control, ETH Zurich, Switzerland.

Cao, G., E. M.-K. Lai, and F. Alam (2017). “Gaussian process model predictive control of unknown non-linear systems”. *IET Control Theory & Applications* **11**:5, pp. 703–713.

Castillo, P., R. Lozano, and A. Dzul (2004). “Stabilization of a mini-rotorcraft having four rotors”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* (Sept. 28–Oct. 2, 2004). Vol. 3. 8359372. Sendai, Japan, pp. 2693–2698. DOI: 10.1109/IROS.2004.1389815.

Chovancová, A., T. Fico, L. Chovanec, and P. Hubinsk (2014). “Mathematical modelling and parameter identification of quadrotor (a survey)”. *Procedia Engineering* **96**, pp. 172–181.

Chowdhary, G., H. A. Kingravi, J. P. How, and P. A. Vela (2013a). “A Bayesian nonparametric approach to adaptive control using Gaussian processes”. In: *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on* (Dec. 10–13, 2013). IEEE. Florence, Italy, pp. 874–879.

Chowdhary, G., H. A. Kingravi, J. P. How, and P. A. Vela (2013b). “Bayesian nonparametric adaptive control of time-varying systems using Gaussian processes”. In: *2013 American Control Conference* (June 17–19, 2013). IEEE. Washington, DC, USA, pp. 2655–2661. DOI: 10.1109/ACC.2013.6580235.

Csató, L. (2002). *Gaussian processes: iterative sparse approximations.* PhD thesis. Department of Computer Science & Applied Mathematics, Aston University, Birmingham, United Kingdom.

Csató, L. and M. Opper (2002). “Sparse on-line Gaussian processes”. *Neural computation* **14**:3, pp. 641–668.

Deisenroth, M. P., D. Fox, and C. E. Rasmussen (2015). “Gaussian processes for data-efficient learning in robotics and control”. *IEEE Transactions on pattern analysis and machine intelligence* **37**:2, pp. 408–423.

Fresk, E. and G. Nikolakopoulos (2013). “Full quaternion based attitude control for a quadrotor”. In: *European Control Conference (ECC),* (July 17–19, 2013). IEEE. Zurich, Switzerland, pp. 3864–3869.

Ghahramani, Z. (2013). “Bayesian non-parametrics and the probabilistic approach to modelling”. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **371**:1984, p. 20110553.

Grande, R. C., G. Chowdhary, and J. P. How (2013). “Nonparametric adaptive control using Gaussian processes with online hyperparameter estimation”. In: *52nd IEEE Conference on Decision and Control* (Dec. 10–13, 2013). IEEE. Florence, Italy, pp. 861–867.

Grande, R. C., G. Chowdhary, and J. P. How (2014). “Experimental validation of Bayesian nonparametric adaptive control using Gaussian processes”. *Journal of Aerospace Information Systems* **11**:9, pp. 565–578.

Greiff, M. (2017). *Modelling and control of the crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation.* MSc thesis TFRT-6026. Department of Automatic Control, Lunds University, Lund, Sweden.

Hewing, L. and M. N. Zeilinger (2017). “Cautious model predictive control using Gaussian process regression”. *arXiv preprint arXiv:1705.10702.*

Houska, B., H. J. Ferreau, and M. Diehl (2011). “ACADO toolkit—an open-source framework for automatic control and dynamic optimization”. *Optimal Control Applications and Methods* **32**:3, pp. 298–312.

Johansson, R. (1993). *System modeling and identification Prentice Hall information and system sciences series.* Englewood Cliffs, NJ: Prentice Hall. ISBN: 0134823087.

Kai, J.-M., G. Allibert, M.-D. Hua, and T. Hamel (2017). "Nonlinear feedback control of quadrotors exploiting first-order drag effects". *IFAC-PapersOnLine* **50**:1, pp. 8189–8195.

Kocijan, J. (2016). *Modelling and control of dynamic systems using Gaussian process models.* Springer, Cham, Switzerland. ISBN: 9783319210216. URL: `https://link.springer.com/book/10.1007%2F978-3-319-21021-6` (visited on 2019-05-27).

Landry, B. (2015). *Planning and control for quadrotor flight through cluttered environments.* MSc thesis 932228932. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA. URL: `https://dspace.mit.edu/handle/1721.1/100608`.

Lee, H. J., J. B. Park, and G. Chen (2001). "Robust fuzzy control of nonlinear systems with parametric uncertainties". *IEEE Transactions on fuzzy systems* **9**:2, pp. 369–379.

Limaverde Filho, J. O. d. A., T. S. Lourenço, E. Fortaleza, A. Murilo, and R. V. Lopes (2016). "Trajectory tracking for a quadrotor system: a flatness-based nonlinear predictive control approach". In: *2016 IEEE Conference on Control Applications (CCA)* (Sept. 19–22, 2016). IEEE. Buenos Aires, Argentina, pp. 1380–1385.

Luukkonen, T. (2011). *Modelling and control of quadcopter.* Aalto University. URL: `http://sal.aalto.fi/publications/pdf-files/eluu11_public.pdf` (visited on 2019-06-10).

Maciejowski, J. M. and X. Yang (2013). "Fault tolerant control using Gaussian processes and model predictive control". In: *Control and Fault-Tolerant Systems (SysTol), 2013 Conference on* (Oct. 9–11, 2013). IEEE. Nice, France, pp. 1–12.

Mattingley, J. and S. Boyd (2012). "CVXGEN: a code generator for embedded convex optimization". *Optimization and Engineering* **13**:1, pp. 1–27.

Miao, L., G. Chowdhary, B. Castra da Silva, L. Shih-Yuan, and J. How (2018). "Gaussian processes for learning and control: a tutorial with examples." *IEEE Control Systems Magazine* **38**:5, pp. 53–86. ISSN: 1066-033X. DOI: `10.1109/MCS.2018.2851010`.

Mohammed, R. O. and G. C. Cawley (2017). "Over-fitting in model selection with Gaussian process regression". In: *International Conference on Machine Learning and Data Mining in Pattern Recognition.* Springer, pp. 192–205.

Mueller, M. W., M. Hamer, and R. D'Andrea (2015). "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)* (May 26–30, 2015). IEEE. Seattle, WA, USA, pp. 1730–1736.

Petersen, K. (2012). *The matrix cookbook*. Version: November 15, 2012. Technical University of Denmark, Lyngby, Denmark. URL: `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=3274` (visited on 2019-05-27).

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Mass. : MIT Press, cop. 2006. ISBN: 026218253X.

Ripley, B. D. (1991). *Statistical inference for spatial processes*. Cambridge university press, Cambridge, United Kingdom. ISBN: 0521352347.

Sanchez-Cuevas, P., G. Heredia, and A. Ollero (2017). "Characterization of the aerodynamic ground effect and its influence in multirotor control". *International Journal of Aerospace Engineering* **2017**:1823056.

Snelson, E. and Z. Ghahramani (2006). "Sparse Gaussian processes using pseudo-inputs". In: Weiss, Y. et al. (Eds.). *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, pp. 1257–1264. URL: `http://papers.nips.cc/paper/2857-sparse-gaussian-processes-using-pseudo-inputs.pdf`.

Sola, J. (2017). "Quaternion kinematics for the error-state Kalman filter". *arXiv preprint arXiv:1711.02508*. (Visited on 2019-05-27).

Soleymani, F. (2012). "A rapid numerical algorithm to compute matrix inversion". *International Journal of Mathematics and Mathematical Sciences* **2012**:134653.

Zhang, Y. and G. Luo (2014). "Fast algorithm for non-stationary Gaussian process prediction". In: *Twenty-Eighth AAAI Conference on Artificial Intelligence* (July 27–31, 2014). Vol. 800. Québec, Canada, pp. 3150–3151.

Zometa, P., M. Kögel, and R. Findeisen (2013). "muAO-MPC: a free code generation tool for embedded real-time linear model predictive control". In: *Proc. American Control Conference (ACC), 2013* (June 17–19, 2013). Washington D.C., USA, pp. 5340–5345. DOI: `10.1109/ACC.2013.6580668`.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* June 2019 *Document Number* TFRT- 6085
Author(s) Sverre Knutsen	*Supervisor* Marcus Greiff, Dept. of Automatic Control, Lund University, Sweden Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Gaussian processes for online system identification and control of a quadrotor

Abstract

The aim of this master thesis was to develop adaptive control for a quadrotor using Gaussian process regression (GP). Online regression with GPs can provide many benefits as it is a flexible non-linear regression model that can provide uncertainty measures of the estimate. However, online GP regression may also gives rise to problems. Because of the high numerical complexity of GPs, sparse approximations are necessary. To this end, the algorithm Sparse Online Gaussian Processes (SOGP) was used. Adaptive control, however, requires the ability to estimate time-varying functions which SOGP does not provide any obvious solution to. To provide time-varying estimations, a Kalman filter interpretation of the update equations in SOGP was used. This addition to the classical SOGP algorithm provided good adaption to non-stationary disturbances at the same time as it was lowering the numerical complexity compared to previously proposed algorithms for nonstationary regression with SOGP. Further, the developed time-varying SOGP regression can provide valuable estimates even under the lack of persistence of excitation, which can be a great advantage in adaptive control. The SOGP algorithm was applied to estimate a model error between a nominal model of the quadrotor and the observed dynamics. The estimated model error could then be used in model-based controllers to improve performance under uncertainties in the dynamics. Two strategies were tested: Model Reference Control and Model Predictive Control. Through these two control strategies, modeled uncertainties in the form of flap- and drag-induced effects, as well as parametric uncertainties, could be estimated online for better flight control. The performance of the proposed controllers was showed through simulations. Even if it was the aim of the thesis no real-time experiments were performed due to lack of time.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-110	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/