Gaussian Process based Model Predictive Control

Gang Cao

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Engineering

School of Engineering and Advanced Technology
Massey University
New Zealand

February 17, 2017

Abstract

The performance of using Model Predictive Control (MPC) techniques is highly dependent on a model that is able to accurately represent the dynamical system. The data-driven modelling techniques are usually used as an alternative approach to obtain such a model when first principle techniques are not applicable. However, it is not easy to assess the quality of learnt models when using the traditional data-driven models, such as Artificial Neural Network (ANN) and Fuzzy Model (FM). This issue is addressed in this thesis by using probabilistic Gaussian Process (GP) models.

One key issue of using the GP models is accurately learning the hyperparameters. The Conjugate Gradient (CG) algorithms are conventionally used in the problem of maximizing the Log-Likelihood (LL) function to obtain these hyperparameters. In this thesis, we proposed a hybrid Particle Swarm Optimization (PSO) algorithm to cope with the problem of learning hyperparameters. In addition, we also explored using the Mean Squared Error (MSE) of outputs as the fitness function in the optimization problem. This will provide us a quality indication of intermediate solutions.

The GP based MPC approaches for unknown systems have been studied in the past decade. However, most of them are not generally formulated. In addition, the optimization solutions in existing GP based MPC algorithms are not clearly given or are computationally demanding. In this thesis, we first study the use of GP based MPC approaches in the unconstrained problems. Compared to the existing works, the proposed approach is generally formulated and the corresponding optimization problem is efficiently solved by using the analytical gradients of GP models w.r.t. outputs and control inputs. The GPMPC1 and GPMPC2 algorithms are subsequently proposed to handle the general constrained problems. In addition, through using the proposed basic and extended GP based local dynamical models, the constrained MPC problem is effectively solved in the GPMPC1 and GPMPC2 algorithms. The proposed algorithms are verified in the trajectory tracking problem of the quadrotor.

The issue of closed-loop stability in the proposed GPMPC algorithm is addressed by means of the terminal cost and constraint technique in this thesis. The stability guaranteed GPMPC algorithm is subsequently proposed for the constrained problem. By using the extended GP based local dynamical model, the corresponding MPC problem is effectively solved.

Acknowledgements

I am deeply grateful to my co-supervisor Professor Edmund M-K Lai at Auckland University of Technology who was my primary supervisor during my first three years of Ph.D study at Massey University. His supervision is great and I am always inspired by the valuable discussions with him. He spent a lot of time on my academic writing and helped me attend several international and local academic conferences. I would not be successful in my Ph.D study without his longstanding support.

I am sincerely appreciative of Dr. Fakhrul Alam who is my primary supervisor in my last year at Massey University for his sharing of ideas and inspiration on possible applications. He helped a lot when I was preparing my thesis draft.

I wish to thank the administrators and technicians of SEAT in Massey University's Albany Campus for their countless help.

Finally, I want to thank my family for their love and support.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN Artificial Neural Network

BFGS Broyden-Fletcher-Goldfarb-Shanno

CG Conjugate Gradient

CGP Convolved Gaussian Process

DGP Dependent Gaussian Process

DMC Dynamic Matrix Control

DOF Degree-of-Freedom

FM Fuzzy Model

FP-SQP Feasibility-Perturbed Sequential Quadratic Programming

GA Genetic Algorithm

GMV Generalized Minimum Variance

GP Gaussian Process

GPC Generalized Predictive Control

GPDM Gaussian Process Dynamical Model

IAE Integral Absolute Error

IDC Inverse Dynamics Control

IGP Independent Gaussian Process

KKT Karush-Kahn-Tucker

LGP Local Gaussian Process

LL Log-Likelihood

LMC Linear Model of Coregionalization

LMI Linear Matrix Inequality

LQR Linear-Quadratic Regulator

LTV Linear Time-Varying

GP-LVM Gaussian Latent Variable Model

MAE Mean Absolute Error

MAP Maximizing A Posterior

MCMC Markov Chain Monte Carlo

MFAC Model-Free Adaptive Control

MIMO Multiple-Input Multiple-Output

MISO Multiple-Input Single-Output

ML Machine learning

MLE Maximum Likelihood Estimation

MPC Model Predictive Control

mp-QP Multi-Parametric Quadratic Programs

MSE Mean Squared Error

NLL Negative value of Log-Likelihood

NLTV Nonlinear Time-Varying

NMPC Nonlinear Model Predictive Control

PCA Principal Component Analysis

PFC Predictive Functional Control

PFDL Partial Form Dynamic Linearization

PSO Particle Swarm Optimization

QP Quadratic Programming

RBFN Radial Basis Function Network

SMPC Stochastic Model Predictive Control

SQP Sequential Quadratic Programming

UAV Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 Background and Motivations

Model Predictive Control (MPC), also known as receding horizon control [1], refers to a class of computer control algorithms that predict future responses of a plant based on an explicit process model, and compute optimized control inputs by repeatedly solving a finite horizon optimization problem. The advantages of MPC mainly lie in its conceptual simplicity for multiple variable problems, and its ability to handle input and output "hard-constraints" that are commonly encountered in practice but are not well addressed in other control methods.

Figure 1.1 illustrates the MPC control strategy. At sampling time i, the future outputs $\hat{y}(i)$ within the prediction horizon $[i, i+H_p]$ are predicted by a system model based on past inputs and outputs $y(i)$ as well as future control inputs, where H_p denotes the prediction horizon. The control inputs are computed by optimizing an objective function so that the predicted output is as close to the desired "Reference Trajectory" as possible within the control horizon $[i, i + H_c]$. where H_c denotes the control horizon. Only the first term of the obtained control sequences $\mathbf{u}(\cdot)$ is subsequently applied to the real system. The whole optimization process will be repeated at next sampling time $(i + 1)$.

The performance of MPC techniques is highly dependent on a model that is able to accurately represent the dynamical system being controlled. Conventionally, such a model is mathematically derived based on theoretical analysis of the underlying physical principles of the dynamical system. For example, the dynamical models of a quadrotor are mathematically derived by the analysis of flight dynamics based on the Newton-Euler

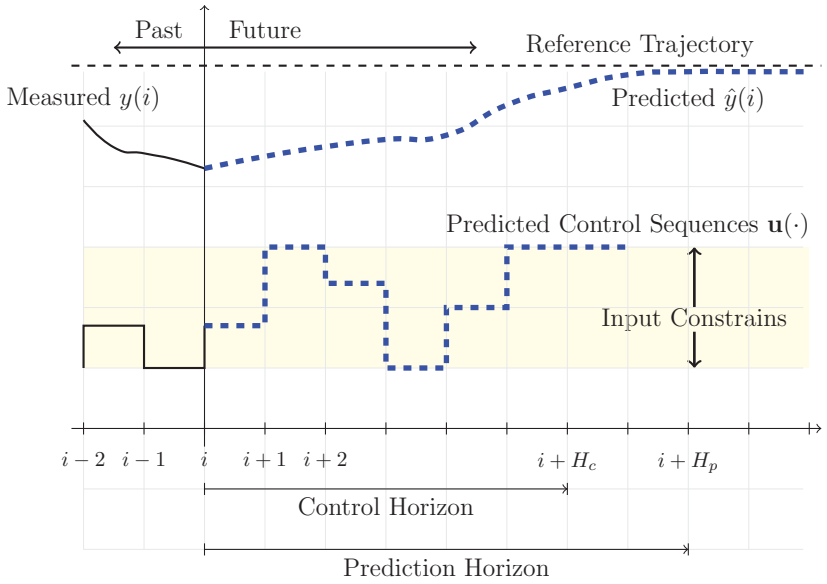


Figure 1.1: Model-based Predictive Control Strategy

or Euler-Lagrange equations [2, 3]. However, it is difficult to use this first principles approach when the system dynamics are too complex or are totally unknown. In such cases, an alternative approach is to use data-driven modelling techniques that are based on computational intelligence and machine learning.

Data-driven models are constructed entirely from empirical data. An additional benefit in using this approach is that the empirical data may capture unknown and unmodelled dynamics of the system. One of the most commonly used data-driven models are Fuzzy Model (FM). They have been used to model many types of systems including solar power generation systems [4], traffic flows [5] and quadrotors [6]. Another popular type of data-driven model is the Artificial Neural Network (ANN). For example, ANN models have also been used to model the complete dynamics of a quadrotor [7–11]. While FM and ANN are useful and versatile, assessing the quality of the models learnt from a finite amount of data is not easy. For ANN, some methods such as Maximum Likelihood Estimation (MLE) [12], bootstrap [13, 14] and Bayesian analysis techniques [15, 16] have been proposed. But these methods do not arise naturally from the model and therefore requires additional computations to implement.

More recently, probabilistic Gaussian Process (GP) models which are based on the Bayesian technique have been proposed and attracted much attention [17]. The major advantage of GP model is that the quality of the model obtained can be evalu-

ated by GP variances which are naturally obtained during the modelling and prediction processes. While the standard GP model is designed for systems with Multiple-Input Single-Output (MISO), Dependent Gaussian Process (DGP) [18] and Convolved Gaussian Process (CGP) [19] models have been proposed for Multiple-Input Multiple-Output (MIMO) systems to take into account correlations between outputs [20]. A key part of the modelling process is the learning of the model hyperparameters from data. It is typically performed by maximizing the Log-Likelihood (LL) function which leads to an unconstrained nonlinear and non-convex optimization problem. Algorithms, such as Conjugate Gradient (CG) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) are commonly used. However, the algorithms tend to be stuck in local optima, especially for CGP which has a higher problem dimension than GP. Furthermore, the LL value is not a reliable indicator for judging the quality intermediate models in the optimization process. A more effective optimization algorithm and a more physically meaningful indicator than the LL would be preferred.

The advantage of using GP model with MPC is that when multiple-step predictions are computed with uncertain inputs, the variance of the GP model, which is a measure of uncertainty, propagates naturally. A GP based MPC scheme was first introduced in [21] for nonlinear systems. It has subsequently been applied to the control of a gas-liquid separator in [22]. However these publications do not contain any discussions on the solution methods for the optimization problems involved. In addition, the cost functions that have been used in the MPC control problem are quite simple in two respects. First, the cost function has a penalty term only on the states/outputs. This reflects the control objective of minimizing the error between model outputs and target values. It also makes the optimization problems simpler to solve. However, in practice, a penalty term on the control inputs is typically needed to limit control actions to avoid over-use of the actuators. In [23], this later penalty term was added. The resulting problem is solved using a multi-parametric technique to compute an explicit look-up table of control inputs offline. But this method could be computationally demanding since the number of table entries grows exponentially with the prediction horizon as well as the input and state dimensions. Secondly, since the states/outputs of a GP model are random variables, the objective function in the optimization problem should be the expectation of cost function instead of the cost function itself [24]. This issue remains unsolved in all works mentioned above where they simply replaced the state variables by GP mean variables in the cost function. A recent work in [25] addresses this issue by using the cost function's expectation. This allows the GP variances to be directly included in the cost function.

However, only unconstrained MPC problems have been considered. To the best of this author's knowledge, constrained GP based MPC with the cost function's expectation as the objective function remains an open problem.

Stability is a key issue for control systems. The stability of GP based MPC schemes has so far been largely ignored by researchers. This issue can be approached by increasing the prediction and control horizon to infinity [2], or by introducing appropriate terminal cost and constraints [26]. In [27], a penalty term on the terminal state has been incorporated. But the authors did not provide any stability analysis. Therefore, a stability guaranteed GP based MPC method is badly needed.

1.2 Research Scope and Objectives

In view of the current status of research in GP based MPC, the main aim of this research is to develop effective GP based MPC schemes for the control of nonlinear dynamical systems with totally unknown dynamics. The unknown dynamics are modelled by GP models based on empirical input and output observations. Trajectory tracking control will be used as the focus of the control problem.

There are four specific research objectives. The first objective relates to the development of a more effective optimization algorithm and the use of an objective function that is a physically more meaningful measure of the quality of the intermediate and final solutions. The problem of getting stuck into local optimum in commonly used CG and BFGS algorithms is due to the sensitiveness to initial values. This issue was partly overcome by standard Particle Swarm Optimization (PSO) algorithms [28, 29]. However, poor initializations can lead to poor global search ability, and they exhibit slow convergence due to poor local search ability.

The second research objective is to address the deficiencies of existing research on open-loop GP based MPC schemes discussed in the previous section. The appropriate objective function which is the expected value of the cost function will be used. Effective and computationally efficient algorithms also be developed for solving the resulting unconstrained and constrained optimization problems. The algorithms developed will be tested on a quadrotor system assuming that the system dynamics are unkonwn. The third objective is therefore to verify the effectiveness of these algorithms for a trajectory tracking problem with the quadrotor.

The stabilizing of proposed open-loop GP based MPC approach is the final research objective. The use of terminal cost and constraints will be studied for this purpose. The proposed method must be theoretically proven to be stable.

1.3 Original Contributions

The original contributions in this thesis are as follows:

A new hybrid PSO algorithm is proposed for the learning of hyperparameters of CGP models. Compared to the existing standard PSO based methods for the GP models [28, 29], the proposed algorithm has better global search ability in the "exploration" phase due to the use of "multi-start" technique, and has better local search ability in the "exploitation" phase due to the use of "gradient" information. In addition, the Mean Squared Error (MSE) of the outputs has been shown to be effective as the fitness function for this learning problem. This allows us to directly assess the quality of intermediate solutions.

A GP based MPC algorithm is proposed for the unconstrained control of unknown nonlinear systems. An efficient solution has been developed to solve this optimization problem by making use of the GP gradients w.r.t. means, variances and control inputs.

Two new GP based MPC formulations, namely GPMPC1 and GPMPC2, have been developed for the constrained control of unknown nonlinear systems. They offer two different ways to include model uncertainty in the formulation. GPMPC1 introduces a variance term (which reflects the model uncertainty) into the objective function while for GPMPC2, variance is part of the state variable. Two corresponding GP local linear models are derived to relax the nonlinear, non-convex optimization problems to ones that are solvable by existing optimization methods. Simulation results showed that both GPMPC1 and GPMPC2 are equally good for solving the trajectory tracking problem for an unknown nonlinear system. Furthermore, GPMPC2 has been shown to be effective in solving the trajectory tracking problem for a quadrotor system.

Finally, a stability guaranteed GP based MPC algorithm has been proposed. This addressed an issue that has so far been largely ignored by the research community. The proposed method makes use of the terminal cost and constraint technique. The resulting constrained optimization problem has been found to be more complicated than the ones in GPMPC1 and GPMPC2. However, it can still be efficiently solved by using the active-

set approach in conjunction with the GP local linear models derived for GPMPC1 and GPMPC2. Mathematical proof of stability based on the Lyapunov theory is provided.

1.4 Thesis Outline

The rest of this thesis is organized as follows.

The literatures related to GP modelling and its applications to unknown dynamic systems and control are reviewed in Chapter 2. In Chapter 3, a new hybrid PSO algorithm is proposed for the problem of hyperparameter learning. Chapter 4 is concerned with the unconstrained tracking problem of unknown systems while Chapter 5 deals with the constrained version of the problem. GP based MPC algorithms are presented for these problems. In Chapter 6, the most effective GP based MPC scheme, known as GPMPC2, is applied to the trajectory tracking problem of the quadrotor. A GPMPC algorithm that has guaranteed stability is presented in Chapter 7. Finally, Chapter 8 concludes the thesis.

Chapter 2

Literature Review

In this chapter, GP as a data-driven modelling tool is reviewed. Following a brief overview of classical and Bayesian regression in Section 2.1, the focus is turned to Gaussian Process modelling in Section 2.2. The key issue of learning the model hyperparameters are reviewed in Section 2.3. Finally, the use of GP in the modelling and control of unknown nonlinear dynamical systems are discussed in Sections 2.4 and 2.5.

2.1 Data-driven Modelling

Data-driven modelling can be viewed as a regression problem that can be solved by classical and Bayesian techniques. This section provides an overview of these two types of techniques.

Consider a dataset $\mathcal{D}$ with N collected noisy observations of the inputs and outputs of a system,

$$\mathcal{D} = \{\mathcal{D}_i | i = 1, \cdots, N\} = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \cdots, N\} \tag{2.1}$$

where $\mathbf{x}_i \in \mathbb{R}^m, m \geq 1$ is the m-dimensional input vector, and $\mathbf{y}_i \in \mathbb{R}^n, n \geq 1$ is the n-dimensional output vector. The aim of data-driven modelling is to construct a model of the input-output relationship based on these observations such that output predictions can be made using this model.

2.1.1 Classical Regression

Parametric regression methods are first considered for this modelling problem. These approaches are usually based on the assumption that the output $\mathbf{y}_i$ are generated by a potential function $f(\mathbf{x}_i;\Theta)$ where Θ denotes a set of parameters. The problem is to find the set of parameters which perfectly explains the relationships among collected observations.

One solution refers to minimizing a cost function $L(\Theta)$. A common choice of cost function is the sum of squared errors defined by [30],

$$L(\Theta) = \sum_{i=1}^{N} \left(\mathbf{y}_i - f(\mathbf{x}_i;\Theta)\right)^2 . \tag{2.2}$$

This gives us the well known least squares model [31]. The choice of potential function is problem dependent. It can range from the polynomial functions [32] to ANN [33].

One major problem of least squares is that even though the total error is minimized, there is no guarantee that the error for any particular input is small. In addition, there is the potential problem of over-fitting where the obtained model works well for the training data but generates bad predictions for some other inputs.

An alternative approach is based on a generative noise model given by

$$\mathbf{y}_i = f(\mathbf{x}_i;\Theta) + \epsilon_i \tag{2.3}$$

where ϵ is an independently and identically distributed noise, and is usually assumed as the Gaussian white noises $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$ with noise variance σ^2. The probability density of over the observations, i.e. the likelihood, can be defined as

$$p(\mathbf{y}|\mathbf{X},\Theta,\sigma^2) = \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{x}_i,\Theta,\sigma^2) \tag{2.4}$$

where $\mathbf{y}$ denotes the set of observed outputs and $\mathbf{X}$ represents the set of observed inputs. In addition, if this probability distribution is Gaussian, then (2.4) becomes

$$p(\mathbf{y}|\mathbf{X},\Theta,\sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \mathbf{exp}(-\frac{(\mathbf{y}_i - f(\mathbf{x}_i;\Theta))^2}{2\sigma^2}). \tag{2.5}$$

The optimal model parameters are obtained by maximizing this likelihood function. This

leads to an unconstrained optimization problem which is often solved by the gradient based solutions if the function f in (2.3) is computationally integrable. CG and BFGS are two popular iterative methods for computing the numerical solution of unconstrained problems. They both require the computation of a gradient of the function. CG does not require an estimation of the Hessian matrix but usually converges slower than BFGS. One major problem is that the models learnt by maximizing the likelihood may suffer from over-fitting.

2.1.2 Bayesian Regression

Bayesian regression provides an alternative way [17, 32]. Instead of using a single model as in the classical methods, Bayesian regression considers all possible models by assuming a distribution over the parameters. In other words, for the prediction each model has a contribution, the degree of which is weighted by its posterior probability. In this way, the problem of over-fitting can be alleviated. Moreover, Bayesian regression is able to provide the full predictive distribution with important information on the amount of confidence one can have on the model predicted value for each test input.

Assuming there are J possible models $\{\mathbf{M}_j, j = 1, \cdots, J\}$ for the given dataset $\mathcal{D}$. The prior belief on $\mathbf{M}_j$ can be expressed as a probability distribution $p(\mathbf{M}_j)$. Let Θ_j be the finite number of parameters that define the model $\mathbf{M}_j$. Then the prior distribution of the model parameters can be defined as $p(\Theta_j|\mathbf{M}_j)$.

Each model makes predictions about how likely the observed data $\mathcal{D}_i = (\mathbf{x}_i, \mathbf{y}_i)$ is generated [34]. The probability distribution of predictions for the whole dataset can be expressed as,

$$p(\mathcal{D}|\mathbf{M}_j, \Theta_j) = \prod_{i=1}^{N} p(\mathcal{D}_i|\mathbf{M}_j, \Theta_j) \tag{2.6}$$

Bayesian regression then is carried out in two steps [32]:

- Calculate the probability of each possible model;
- Make a comparison between all possible models.

In the first step, the posterior distribution of the parameters for each model is obtained

from the prior distribution $p(\Theta_j|\mathbf{M}_j)$ and the likelihood $p(\mathcal{D}|\mathbf{M}_j, \Theta_j)$ by using Bayes' rule,

$$p(\Theta_j|\mathcal{D}, \mathbf{M}_j) = \frac{p(\Theta_j|\mathbf{M}_j)p(\mathcal{D}|\mathbf{M}_j, \Theta)}{p(\mathcal{D}|\mathbf{M}_j)} \tag{2.7}$$

Here, $p(\mathcal{D}|\mathbf{M}_j)$ is the marginal likelihood (or "model evidence"). It is independent of the parameters and is given by

$$p(\mathcal{D}|\mathbf{M}_j) = \int p(\Theta_j|\mathbf{M}_j)p(\mathcal{D}|\mathbf{M}_j, \Theta_j)d\Theta_j \tag{2.8}$$

The posterior distribution in (2.7) contains all the information we know about the parameters. Similar to maximizing the likelihood, Maximizing A Posterior (MAP) can produce a point estimator of parameters but employs an additional prior distribution.

The second step involves ranking the plausibility of each model based on their posterior probabilities [35]. Informally, the marginal likelihood is a likelihood of the model because its parameters have been marginalized. Thus, the posterior of the model can be defined by,

$$p(\mathbf{M}_j|\mathcal{D}) = \frac{p(\mathbf{M}_j)p(\mathcal{D}|\mathbf{M}_j)}{p(\mathcal{D})} \tag{2.9}$$

where $p(\mathcal{D})$ can be viewed as a normalising constant. Therefore (2.9) can be rewritten as

$$p(\mathbf{M}_j|\mathcal{D}) = \text{Constant} \cdot p(\mathbf{M}_j)p(\mathcal{D}|\mathbf{M}_j). \tag{2.10}$$

Three key issues should be addressed when Bayesian regression is applied to practical problems [32, 34]. The first one is related to the 'evidence' computation in the first step. Typically, the integral in (2.8) is computationally intractable. This problem can be overcome by assuming an analytically tractable Gaussian approximation to the evidence. Alternatively, Markov Chain Monte Carlo (MCMC) techniques can be used to evaluate this integral numerically [36]. However, the computational demands of both methods result are high. The second issue involves the prior. This prior reflects the subjective knowledge of the system or data to be modelled. Although the principle of Occam's Razor [37] tells us that simple models are sufficient for modelling purposes, assign the appropriate probability distribution to the prior is still an issue to be tackled. The last problem refers to the choice of candidate model set. Currently, no Bayesian criterion exists to evaluate whether the choice of models is correct. Furthermore, even though the appropriate model set is selected, making Bayesian predictions still could be complicated. This is because nonlinear functions of the parameters and its probabilistic characters often

produce analytically intractable integrals. More details on Bayesian regression can be found in [35].

2.2 Gaussian Process Models

GP has been proposed to overcome the issues when using Bayesian regression techniques [32]. Instead of assuming a particular parametric form for the latent function in (2.3) and making inference about parameters Θ, the Gaussian prior is directly imposed on the function values and inferences take place in the function space. This is much easy to implement than setting a prior on the latent function. In addition, the parameters Θ are not necessary any more therefore GP models are usually considered as non-parametric.

In the 1940's, the Wiener-Kolmogorov prediction theory was presented for the time series analysis [38]. This is considered the earliest research related to GP models as the Wiener process is the equivalent to a GP [39]. Another early stage GP model is the so-called "kriging" which is originally developed in the geostatistics field in the 1970's. This model is basically identical to the GP regression model [40], and is often applied to low-dimensional problems (two or three in most cases) in spatial statistics. In the context of statistics, the GP models were used to define the prior distributions over functions for the one-dimensional curve fitting problems in 1978 [41]. However, the potential of GP was left unnoticed until Bayesian learning methods converged with ANN in [42]. The GP model was subsequently used as the non-parametric Bayesian method to the regression problems in [43]. In [44], the GP was compared to other commonly used approaches for the nonlinear regression problem. From these works, the potential of using GP models for regression problems have been demonstrated.

2.2.1 Standard Gaussian Process Models

A standard GP is defined as a collection of random variables with a joint Gaussian distribution among any finite number of them [17].

The dataset becomes $\mathcal{D} = \{(\mathbf{x}_i, \mathrm{y}_i)|i = 1, \cdots, N\}$ if $n = 1$. In the standard GP model, it is usually assumed that $\mathrm{y}_i = f(\mathbf{x}_i) + \epsilon_i$ where $f : \mathbb{R}^m \to \mathbb{R}$ is a random function and is also called latent function, $\epsilon_i \sim \mathcal{N}(0, \varepsilon^2)$ denotes an independent Gaussian white noise with zero mean and variance σ^2. Therefore, the set of $\mathbf{f} = \{f(\mathbf{x}_1), \cdots, f(\mathbf{x}_N)\}^T$ is

a GP if the following Gaussian distribution over the latent function values satisfies,

$$p(\mathbf{f}|\mathbf{X},\mathbf{K}) = \frac{1}{L}\exp\left(-\frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^T\mathbf{K}^{-1}(\mathbf{f}-\boldsymbol{\mu})\right), \tag{2.11}$$

where L is a normalising constant, $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}^T$ denotes a dataset of training inputs and $\boldsymbol{\mu}$ is the mean value of the Gaussian distribution. In addition, $\mathbf{K} \in \mathbf{R}^{n\times n}$ represents the covariance matrix computed by a covariance function $Cov(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta})$ with well known hyperparameters $\boldsymbol{\theta}$. One key issue when using GP models is choosing an appropriate covariance function that expresses our assumptions about the latent function. This is usually described by the GP prior $p(\mathbf{f}|\boldsymbol{\theta})$ over the space of latent function. In [17], some examples of covariances functions, as well as their influence on the problem of model learning are presented.

Without loss of generality, a zero mean $\boldsymbol{\mu} \equiv 0$ is usually used in the distribution of latent functions such that,

$$\mathbf{f}|\mathbf{X},\mathbf{K} \sim \mathcal{N}(0,\mathbf{K}) \tag{2.12}$$

For an unobserved input $\mathbf{x}^*$, the joint distribution between the corresponding output value f^* and $\mathbf{f}$ is still a GP and can be subsequently obtained by,

$$\left[\begin{array}{c}\mathbf{f}\\ f^*\end{array}\right]\Bigg|\mathbf{X},\boldsymbol{\theta} \sim \mathcal{N}\left(0,\ \left[\begin{array}{cc}\mathbf{K} & \mathbf{K}_{\mathbf{f},f^*}\\ \mathbf{K}_{f^*,\mathbf{f}} & \mathbf{K}_{f^*,f^*}\end{array}\right]\right), \tag{2.13}$$

where $\mathbf{K}_{\mathbf{f},f^*} = \mathbf{K}_{f^*,\mathbf{f}}^T = [Cov(\mathbf{x}^*,\mathbf{x}_1),\cdots,Cov(\mathbf{x}^*,\mathbf{x}_N)]^T \in \mathbb{R}^{N\times 1}$ denotes the cross-covariances between observed inputs and the new input, and $\mathbf{K}_{f^*,f^*} = Cov(\mathbf{x}^*,\mathbf{x}^*)$ is the self-covariance of new input.

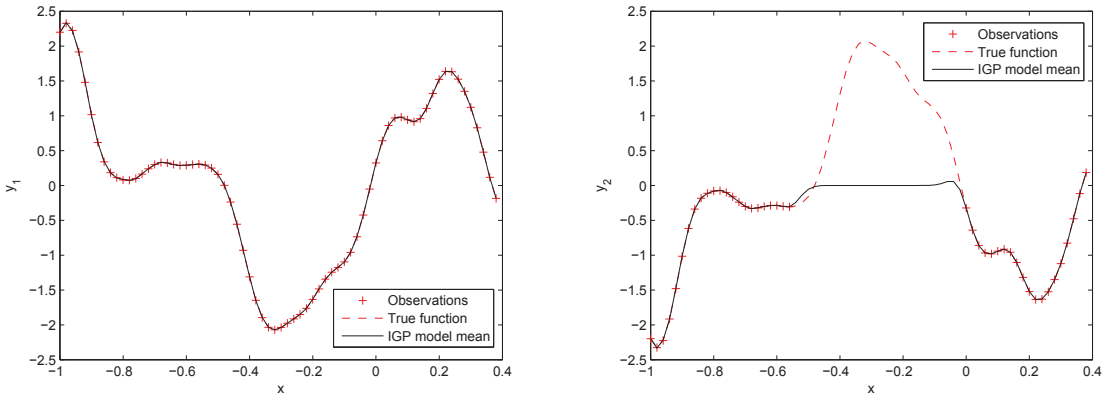
Let the latent functions be corrupted by independent noises $\boldsymbol{\epsilon} = [\epsilon_1,\cdots,\epsilon_N]^T$. The joint distribution between the observed outputs $\mathbf{y} = \{\mathbf{y}_1,\cdots,\mathbf{y}_N\}^T$ and predicted output $\mathbf{y}^*$ at unobserved $\mathbf{x}^*$ is a GP as well and can be defined by,

$$\left[\begin{array}{c}\mathbf{y}\\ \mathrm{y}^*\end{array}\right]\Bigg| \sim \mathcal{N}\left(0,\ \left[\begin{array}{cc}\mathbf{K}+\boldsymbol{\Sigma} & \mathbf{K}_{\mathbf{y},\mathrm{y}^*}\\ \mathbf{K}_{\mathrm{y}^*,\mathbf{y}} & \mathbf{K}_{\mathrm{y}^*,\mathrm{y}^*}\end{array}\right]\right), \tag{2.14}$$

where $\boldsymbol{\Sigma} = \varepsilon^2\mathbf{I}$.

Through conditioning on observed outputs $\mathbf{y}$, the predictive distribution of y^* is still a GP and can be obtained by,

$$\mathrm{y}^*|\mathbf{X},\mathbf{y},\boldsymbol{\theta},\boldsymbol{\Sigma} \sim \mathcal{N}\left(\mu(\mathbf{x}^*),\sigma^2(\mathbf{x}^*)\right), \tag{2.15}$$



2.1.a: y_1 2.1.b: y_2

Figure 2.1: Example showing the predicted outputs of IGP modelling

where the predictive mean and variance functions are specified by,

$$\begin{aligned} \mu(\mathbf{x}^*) &= \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma})^{-1}\mathbf{y} \\ \sigma^2(\mathbf{x}^*) &= \mathbf{K}_{\mathbf{f}^*,\mathbf{f}^*} - \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma})^{-1}\mathbf{K}_{\mathbf{f},\mathbf{f}^*}. \end{aligned} \tag{2.16}$$

2.2.2 GP models for Multiple Outputs

Most current applications of GP involve the modelling of MISO systems. For MIMO systems, one can either use a single multiple-output GP model or multiple independent single-output GP models with one for each output. The latter technique is known in the literature as the Independent Gaussian Process (IGP) approach [45, 46]. One example of applying IGP to a multiple response variable problem can be found in [47]. A multiple-output model is generally more complicated than the IGP, but is able to model the correlation between multiple outputs. To date, there is no consensus as to which of these two approaches produces better predictive performances.

While IGP is a simple extension of the GP modelling, there are cases where this approach is not applicable. Consider a system with single input x and two strongly coupled outputs y_1 and y_2, where $y_1 = -y_2$. The true values of y_1 and y_2 are shown as red dashed lines in Figures 2.1.a and 2.1.b respectively. Suppose the detailed knowledge of y_1 (70 observations shown as red plus signs in Figures 2.1.a) over the interval $x \in [-1, 0.4]$ is known, but a part of the observations for y_2 (27 observations shown as red plus signs in Figures 2.1.b) have been lost for $x \in (-0.55, -0.05)$. Through using the IGP approach,

Figure 2.1.a shows that y_1 is correctly modelled. However, due to the missing data, y_2 could not be modelled properly as shown in Figure 2.1.b. To address this issue, it has been proposed to use a GP based multiple-output model in [20]. The commonly used MIMO GP models are presented in the following paragraphs.

Linear Model of Coregionalization

In geostatistics, the standard GP model has been extended to multiple-task [48] and multiple-output problems [49]. This technique is known as the Linear Model of Coregionalization (LMC). In the context of LMC, each output function is expressed as the mixture of a number of different processes. Specifically, for a system with m outputs, the d^{th} output function is often synthesized from Q groups of underlying functions $u_q^i(\mathbf{x})$ [50],

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^i u_q^i(\mathbf{x}) \tag{2.17}$$

where $\mathbf{x} \in \mathbb{R}^n$ denotes the input vector and $d = 1, \cdots, m$. Within each group, the underlying functions share the same covariance function, but are independent. Here the covariance typically is defined by

$$Cov[f(\mathbf{x}), f(\mathbf{x}+\tau)] = \mathbf{K}(\tau) = \sum_{q=1}^{Q} \mathbf{A}_q k_q(\tau), \tag{2.18}$$

where $f(\mathbf{x}) = [f_1(\mathbf{x}), \cdots, f_m(\mathbf{x})]^T$. In addition, $\{\mathbf{A}_q\}_{q=1}^{Q} \in \mathbb{R}^{D\times D}$ are known as the coregionalization matrices that capture the correlations across output functions, and $k_q(\tau)$ defines a covariance that captures the correlation between inputs in the q^{th} group of output functions.

LMC models correlate the outputs based on a linear weighted sum of independent processes. This can be viewed as an 'instantaneous mixing' technique [51, 52]. It has two main limitations. The first one is that the coregionalization matrices are difficult to estimate [53]. The second one is that 'instantaneous mixing' is not often applicable in the engineering context [51] where system outputs are likely the results of convolution.

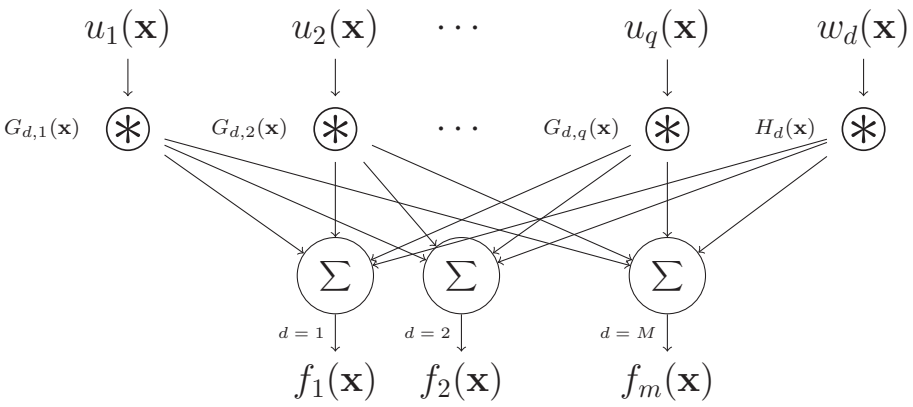


Figure 2.2: Structure of a Dependent Gaussian Process Model

Dependent Gaussian Processes

The issues of LMC can be overcome by constructing the covariances using Convolution Process (CP). This idea was first suggested in [54] and subsequently explored in [55]. In [18], CP method has been used to construct the covariance functions. Since linearly filtering a GP will produce the GP [56] again, a standard multiple-input GP can be obtained through a MISO linear filter with Gaussian white noise as input. This method is extended to MIMO systems with multiple dependent Gaussian processes. As a result, the DGP is proposed in [18] to cope with MIMO problems directly, where DGP models are produced through a convolution between Gaussian white noise and smoothing kernels.

Consider a set of m output functions $\{f_d(\mathbf{x})\}_{d=1}^{m}$. Each $f_d(\mathbf{x})$ can be obtained by a sum of two convolution integrals,

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \int_{\mathcal{X}} G_{d,q}(\mathbf{x} - \tau) u_q(\tau) d\tau + \int_{\mathcal{X}} H_d(\mathbf{x} - \tau) w_d(\tau) d\tau \tag{2.19}$$

where $G_{d,q}(\mathbf{x})$ and $H_d(\mathbf{x})$ are two smoothing kernels. The latent process $w_d(\mathbf{x})$ is a Gaussian noise and only affects the output $f_d(\mathbf{x})$. On the other hand, $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ is a set of Gaussian noise affecting all outputs. In the DGP, all Gaussian noises are assumed to be independent and with zero mean and unit variance, i.e. $w_d(\mathbf{x}), u_q(\mathbf{x}) \sim \mathcal{N}(0, 1)$. Figure 2.2 depicts the structure of a DGP model.

Then, the cross-covariance between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ can be obtained by,

$$\begin{aligned} Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = E\Bigg[&\left(\sum_{q=1}^{Q}\int_{\mathcal{X}} G_{d,q}(\mathbf{x}-\tau)u_q(\tau)d\tau + \int_{\mathcal{X}} H_d(\mathbf{x}-\tau)w_d(\tau)d\tau\right) \\ &\left(\sum_{q=1}^{Q}\int_{\mathcal{X}} G_{d',q}(\mathbf{x}'-\tau')u_q(\tau')d\tau' + \int_{\mathcal{X}} H_{d'}(\mathbf{x}'-\tau')w_{d'}(\tau')d\tau'\right)\Bigg] \\ = &\iint_{\mathcal{X}} \sum_{q=1}^{Q} G_{d,q}(\mathbf{x}-\tau)G_{d',q}(\mathbf{x}'-\tau')E[u_q(\tau), u_q(\tau')]d\tau' d\tau \\ + &\iint_{\mathcal{X}} H_d(\mathbf{x}-\tau)H'_d(\mathbf{x}'-\tau')E[w_q(\tau), w_q(\tau')]d\tau' d\tau \end{aligned} \tag{2.20}$$

In addition, based on the assumption of independent identical distribution with zero mean and unit variance, it can be known that,

$$E\left[w_q(\tau), w_q(\tau')\right] = E\left[u_q(\tau), u_q(\tau')\right] = 1 \tag{2.21}$$

and

$$E\left[\sum_{q=1}^{Q}\int_{\mathcal{X}} G_{d,q}(\mathbf{x}-\tau)u_q(\tau)d\tau \int_{\mathcal{X}} H_{d'}(\mathbf{x}'-\tau')w_{d'}(\tau')d\tau'\right] = 0 \tag{2.22}$$

Thus, (2.20) becomes,

$$\begin{aligned} Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = &\sum_{q=1}^{Q}\int_{\mathcal{X}} G_{d,q}(\mathbf{x}-\tau)G_{d',q}(\mathbf{x}'-\tau)d\tau \\ &+ \int_{\mathcal{X}} H_d(\mathbf{x}-\tau)H'_d(\mathbf{x}'-\tau)d\tau \\ = &Cov_u[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] + Cov_w[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')]\delta_{d,d'} \end{aligned} \tag{2.23}$$

where $\delta_{d,d'}$ is the Kronecker delta function with value 1 when $d = d'$ and 0 otherwise. The key to compute the covariance function in the DGP model is specifying the smoothing kernels. Gaussian kernels have been used in [18]. The results show that DGP performs great modelling ability for MIMO problem with strongly or partially correlated outputs. Recently, DGP has been applied to identify the Unmanned Aerial Vehicle (UAV) model [57, 58].

One key benefit of using DGP models is their predictive mean and variance functions remain same formulations to (2.16). More details about predictions of DGP models can be found in [30].

Convolved Gaussian Process Models

DGP can be further generalized by allowing latent processes other than Gaussian white noise. This generalization leads to the CGP model [59] that allows us to handle wider range of real problems.

Consider a system with n inputs $\mathbf{x} \in \mathbb{R}^n$ and m outputs $\mathbf{y}(\mathbf{x}) \in \mathbb{R}^m$ again. In the CGP, each output $\mathbf{y}_d(\mathbf{x})$ is modelled by,

$$\mathbf{y}_d(\mathbf{x}) = f_d(\mathbf{x}) + \epsilon_d(\mathbf{x}) \tag{2.24}$$

where $d = 1, 2, \ldots, m$ and $\epsilon_d(\mathbf{x})$ denotes an independent Gaussian white noise. The function $f_d(\mathbf{x})$ typically is defined by a linear convolution of a smoothing kernel $H_d(\mathbf{x})$ and a latent function $u(\mathbf{x})$,

$$f_d(\mathbf{x}) = \int H_d(\mathbf{x} - \tau) u(\tau) d\tau \tag{2.25}$$

The correlation between outputs is derived from the latent function $u(\mathbf{x})$ which has effects on all output functions. This latent function can be any appropriate random processes. If a Gaussian white noise is used, then resulting in a DGP model. In the CGP, a wide range of latent functions are proposed to match the modelling requirements for different physical or dynamical systems [59].

In addition, the CGP models allow using more than one type of latent function. Assuming Q groups of latent functions are considered, where for the q^{th} group, it has R_q smoothing kernels. Thus the d^{th} output function can be rewritten by,

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{k=1}^{R_q} \int H_{d,q}^k(\mathbf{x} - \tau) u_q^k(\tau) d\tau \tag{2.26}$$

Then, the covariance between different outputs $\mathbf{y}_d(\mathbf{x})$ and $\mathbf{y}_{d'}(\mathbf{x}')$ can be obtained by,

$$\begin{aligned} \mathbf{K}_{\mathbf{y}_d, \mathbf{y}_{d'}}(\mathbf{x}, \mathbf{x}') =& Cov[\mathbf{y}_d(\mathbf{x}), \mathbf{y}_{d'}(\mathbf{x}')] \\ =& Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] + Cov[\epsilon_d(\mathbf{x}), \epsilon_{d'}(\mathbf{x}')] \, \delta_{d,d'} \end{aligned} \tag{2.27}$$

where $\delta_{d,d'}$ is a Kronecker delta function thus $Cov[\epsilon_d(\mathbf{x}), \epsilon_{d'}(\mathbf{x}')] \, \delta_{d,d'}$ will lead to a diagonal matrix of noise variance $\left\{\sigma_d^2\right\}_{d=1}^m$ if it is assumed that $\epsilon_d(\mathbf{x}) \sim \mathcal{N}(0, \sigma_d^2)$, and the cross-

covariance between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ is given by,

$$\begin{aligned}
&\mathbf{K}_{f_d,f_{d'}}(\mathbf{x},\mathbf{x}') = Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] \\
&= E\left[\sum_{q=1}^{Q}\sum_{k=1}^{R_q}\int H_{d,q}^k(\mathbf{x}-\tau)u_q^k(\tau)d\tau \sum_{q=1}^{Q}\sum_{k=1}^{R_q}\int H_{d',q}^k(\mathbf{x}'-\tau')u_q^k(\tau')d\tau'\right] \\
&= \sum_{q=1}^{Q}\sum_{k=1}^{R_q} k_q(\tau,\tau')\int H_{d,q}^k(\mathbf{x}-\tau)H_{d',q}^k(\mathbf{x}'-\tau)d\tau
\end{aligned} \tag{2.28}$$

Data-driven modelling using CGP basically involves obtaining the appropriate smoothing kernels and latent functions that reflect the covariance between outputs.

As given in (2.26), the output function is a linear combination of independent random functions. Thus, if these functions are Gaussian processes, then $f_d(\mathbf{x})$ will also be a Gaussian process. In this case, the smoothing kernels can be expressed by,

$$H_{d,q}^k(\gamma) = \frac{\nu_{d,q}^k \left|\mathbf{P}_{d,q}^k\right|^{1/2}}{(2\pi)^{M/2}} \exp\left[-\frac{1}{2}(\gamma)^T\mathbf{P}_{d,q}^k(\gamma)\right] \tag{2.29}$$

where $\nu_{d,q}^k$ is a length-scale coefficient, $\mathbf{P}_{d,q}^k$ is an $n \times n$ precision matrix of the smoothing kernel. To simplify the model further, it is assumed that the covariances of latent functions $k_q(\eta)$ in every group are all same Gaussian,

$$k_q(\eta) = \frac{\upsilon_q \left|\mathbf{P}_q\right|^{1/2}}{(2\pi)^{M/2}} \exp\left[-\frac{1}{2}(\eta)^T\mathbf{P}_q(\eta)\right] \tag{2.30}$$

where υ_q is the length-scale coefficient and $\mathbf{P}_q$ is another $n \times n$ precision matrix.

To simplify the discussion again, it is assumed that $R_q = 1$ for all Q groups of latent functions. In addition, the precision matrices of the smoothing kernels are assumed to be the same for each group of latent functions. As a result, given the smoothing kernel (2.29) and latent function covariance (2.30), the covariance can be obtained by,

$$Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{q=1}^{Q} \frac{\nu_{d,q}\nu_{d',q}\upsilon_q}{(2\pi)^{M/2}\left|\mathbf{P}\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T\mathbf{P}^{-1}(\mathbf{x}-\mathbf{x}')\right] \tag{2.31}$$

where $\mathbf{P} = \mathbf{P}_\mathbf{d}^{-1} + \mathbf{P}_\mathbf{d'}^{-1} + \mathbf{P}_\mathbf{q}^{-1}$. Note that this multiple-output covariance function maintains a Gaussian form, i.e. $\mathbf{K}_{\mathbf{f_d},\mathbf{f_{d'}}}(\mathbf{x},\mathbf{x}') \sim \mathcal{N}(\mathbf{x}-\mathbf{x}'|0,\mathbf{P})$.

Then similar to standard GP models, given a set of observations $\left\{\mathbf{x}^j, \mathbf{y}_j\right\}_{j=1}^{J_d}$, where

$\sum_{d=1}^{m} J_d = N$, a Gaussian distribution can be defined on the output functions by,

$$\mathbf{y}(\mathbf{x}) \sim \mathcal{N}\left(\mu(\mathbf{x}), \mathbf{K}_{\mathbf{y},\mathbf{y}}(\mathbf{x}, \mathbf{x}')\right) \tag{2.32}$$

where the output vector $\mathbf{y}(\mathbf{x})$ is given by,

$$\mathbf{y}(\mathbf{x}) = \left[\mathbf{y}_1(\mathbf{x}), ..., \mathbf{y}_m(\mathbf{x})\right]^T \tag{2.33}$$

with the entries,

$$\mathbf{y}_d(\mathbf{x}) = \left[\mathbf{y}_d(\mathbf{x}^1), \mathbf{f_d}(\mathbf{x}^2), ..., \mathbf{f_d}(\mathbf{x}^{J_d})\right]^T \tag{2.34}$$

Without loss of generality, zero means are used. In addition, the covariance matrix $\mathbf{K}_{\mathbf{y},\mathbf{y}}(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{N \times N}$ can be obtained by using (2.28) and (2.31). Usually, the computation of such a covariance matrix is computationally expensive. Thus, some sparse approximations have been proposed to reduce the complexities of CGP [19]. Then, the marginal likelihood can be defined by,

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{y}|0, \mathbf{K}_{\mathbf{y},\mathbf{y}}) \tag{2.35}$$

The joint distribution of observed $\mathbf{y}$ and the predicted outputs $\mathbf{y}^* = \{y_1^*, \cdots, y_M^*\}$ at new input $\mathbf{x}^*$ is thus still a Gaussian and is given by,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{matrix} \mathbf{K}_{\mathbf{y},\mathbf{y}} & \mathbf{K}_{\mathbf{f},\mathbf{f}^*} \\ \mathbf{K}_{\mathbf{f}^*,\mathbf{f}} & \mathbf{K}_{\mathbf{f}^*,\mathbf{f}^*} \end{matrix}\right) \tag{2.36}$$

Finally, similar to standard GP models again, the predictive distribution is a Gaussian,

$$\mathbf{y}^*|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}, \mathbf{x}^* \sim \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)) \tag{2.37}$$

where the mean $\mu(\mathbf{x}^*)$ and variance $\sigma^2(\mathbf{x}^*)$ functions are computed by,

$$\mu(\mathbf{x}^*) = \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y} \tag{2.38a}$$

$$\sigma^2(\mathbf{x}^*) = \mathbf{K}_{\mathbf{f}^*,\mathbf{f}^*} - \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{K}_{\mathbf{f},\mathbf{f}^*} \tag{2.38b}$$

2.3 Hyperparameter Learning

Performing predictions using (2.16) for standard GP models, or using (2.38) for CGP models, the covariance matrix $\mathbf{K}$. This matrix is specified by a set of hyperparameters

$\boldsymbol{\theta}$. They are usually obtained by maximizing the log of marginal likelihood function.

In GP models, the marginal likelihood is equal to the integral over a product of the likelihood function and GP prior over the latent functions, both of which are Gaussian. Thus, the marginal likelihood is also Gaussian and defined by,

$$\begin{aligned} p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) &= \int p(\mathbf{y}|\mathbf{f},\mathbf{X},\boldsymbol{\theta})p(\mathbf{f}|\boldsymbol{\theta})d\mathbf{f} \\ &= \frac{1}{(2\pi)^{\frac{N}{2}}|\mathbf{K}_{\mathbf{y},\mathbf{y}}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y}\right) \end{aligned} \tag{2.39}$$

This marginal likelihood can be viewed as the likelihood of hyperparameters corrupted by noise so that we simply call likelihood function. A good point estimate $\hat{\boldsymbol{\theta}}$ of hyperparameters can be subsequently obtained by maximizing this likelihood function. In practice, we usually estimate the hyperparameters by maximizing the log likelihood function due to its less computation complexities. The corresponding optimization problem can be subsequently defined as,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) \tag{2.40}$$

where,

$$\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{y},\mathbf{y}}| - \frac{N}{2}\log 2\pi \tag{2.41}$$

The unconstrained optimization problem (2.40) is not easy to solve due to it is typically nonlinear and non-convex. However, in GP models, the derivatives of log likelihood function with respective to the hyperparameter $\boldsymbol{\theta}$ are mathematically analytical and can be obtained by,

$$\frac{\partial}{\partial\boldsymbol{\theta}_l}\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\frac{\partial\mathbf{K}}{\partial\boldsymbol{\theta}_l}\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y} - \frac{1}{2}\mathrm{trace}(\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\frac{\partial\mathbf{K}}{\partial\boldsymbol{\theta}_l}) \tag{2.42}$$

By using this gradient, problem (2.40) can be solved by using a BFGS and a CG algorithm [17]. However, the performance of gradient-based method is very sensitive to initial values. Therefore, these algorithms are often restarted multiple times with different initializations. Recently, PSO based solutions have been proposed in [28, 29] as an alternative method to solve the problem. It generally has better global search ability due to the use of a number of different particles. Computational demand is reduced because it does not need to compute the gradient.

2.4 Applications of GP Models

GP model has become more popular in the past decade with the development of Machine learning (ML) techniques [60–62]. In terms of unknown systems, the use of GP model mainly lies in the modelling and control of unknown nonlinear systems.

2.4.1 GP Modelling of Unknown Nonlinear System

As a non-parametric Bayesian modelling technique, the GP model has been considered as an alternative approach to the modelling of nonlinear systems that are usually represented by using the parametric methods, such as ANN and FM. In [63, 64], the GP model is used for the unknown nonlinear dynamical systems due to the quality of learnt model can be evaluated by using the GP variance naturally obtained during the modelling process. The so called Gaussian Latent Variable Model (GP-LVM) based on the GP models and Principal Component Analysis (PCA) technique is proposed in [65]. The use of the GP-LVM model allows the reduction of problem dimensionality by projecting the observed data space onto a latent space with lower dimension. The Gaussian Process Dynamical Model (GPDM) model proposed in [66] is conceptually similar to the GP-LVM but is able to accurately model the nonlinear system with limited observations. More Recently, a moving-window technique is incorporated into the GP models to handle the time-varying dynamical system in [67]. The theoretical researches on the use of GP models in the different nonlinear modelling problems can be found in [68–71].

The GP models has been used in the modelling of different industrial systems, such as the chemical and material processes [72, 73], the hydraulic system [63, 64], the wastewater treatment process [74], the concrete properties [75] and the wind turbine [76]. In addition, the GP models are also employed to identify human motion [77, 78] as well as robotic systems [79]. In the context of process controls, the GP models have been used for the process monitoring, fault diagnosis and soft sensors [67, 80–84].

2.5 GP Applications on Control

2.5.1 Inverse Dynamics Control

Inverse dynamics control (Inverse Dynamics Control (IDC)) is a control technique based on an inverse model of the nonlinear system. The inverse model, obtained by a model inversion of the nonlinear process, is directly utilized as the controller. Thus the expected output is treated as an input variable in the model, and the corresponding control action is predicted.

The GP models have been used to offline learn the inverse dynamic model of a robot system in [85]. The learnt model is subsequently applied into the inverse dynamics control of robotics in [86, 87]. In addition, through combining standard GP model and locally weighted projection regression techniques, the Local Gaussian Process (LGP) model is proposed in [88] to handle the online learning and control problems.

2.5.2 Adaptive Control

Adaptive control involves continuously adjusting the controller on-line to maintain a desired level of control performance. It is applicable when the parameters of the plant dynamic model are unknown or time-varying. Adaptive control can either be open-loop and closed-loop.

The GP model was first introduced into the adaptive control in [89] where linear and nonlinear systems are described by GP models. In addition, through considering GP variances in the cost function, obtained adaptive controllers are able to produce control actions with the robustness against output variances. This work is further improved in [90] where the multiple-step prediction technique proposed in [91] is used so that the issue of uncertainty propagation can be addressed. Another GP based adaptive controller is proposed in [92] based on the Generalized Minimum Variance (GMV) control strategy. In these works, different cost functions are used to address the issue of consideration of GP variances.

In [29], an adaptive controller is proposed based on the evolving GP models. Evolving GP refers to recursively adapting the structure of the GP model and its hyperparameters values based on some kind of evolution. The proposed technique is different from abovementioned adaptive controllers since both the model structure and model pa-

rameters are updated from measured data [29]. This technique is subsequently improved with faster on-line learning ability and is applied to the problem of predicting ozone concentration in [93].

2.5.3 Model Predictive Control

MPC is one of the most commonly used control techniques for both linear and nonlinear systems with constraints [94]. Examples of MPC algorithms that have been developed and used in practice include Dynamic Matrix Control (DMC) [95], Predictive Functional Control (PFC) [96] and Generalized Predictive Control (GPC) [97].

MPC control of dynamical nonlinear systems represented by GP models have been studied for the first time in [98]. The proposed algorithm is subsequently applied to a pH neutralization process [21], and a gas-liquid separation plant [22]. The cost function used in these works are in the form:

$$\begin{aligned}\mathcal{J}(\mathbf{x}_k) =& \sum_{i=1}^{H} \left\{ \left(\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\right)^T \mathbf{Q} \left(\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\right) \right\} \\ =& \sum_{i=1}^{H} \left\{ \left(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\right)^T \mathbf{Q} \left(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\right) \right\} = \mathcal{H}(\boldsymbol{\mu}_k)\end{aligned} \tag{2.43}$$

where $\mathbf{x}$ and $\mathbf{r}$ represents vectors of states and target values, H is the prediction horizon and $\mathbf{Q}$ is a positive definite weighting matrix. In this case, the states are simply the predicted GP mean values $\boldsymbol{\mu}$. A hard constraint on the GP variances is used to address the issue of model uncertainty.

In [99], a more general formulation of GP based MPC problem is given. It is based on the following cost function that includes control inputs $\mathbf{u}$ as follows.

$$\begin{aligned}\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) =& \sum_{i=1}^{H} \left\{ \left(\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\right)^T \mathbf{Q} \left(\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\right) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right\} \\ =& \sum_{i=1}^{H} \left\{ \left(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\right)^T \mathbf{Q} \left(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\right) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right\} \\ =& \mathcal{H}(\boldsymbol{\mu}_k, \mathbf{u}_{k-1})\end{aligned} \tag{2.44}$$

where $\mathbf{R}$ denotes another positive definite weighting matrix. The states are again replaced by the GP means. However, model uncertainty in this method is addressed by treating

GP variances Σ as the slack variables of GP means in the form of constraints $\boldsymbol{\mu}_{k+i} - 2\Sigma_{k+i} \geq \mathbf{x}_{\min}$ and $\boldsymbol{\mu}_{k+i} + 2\Sigma_{k+i} \leq \mathbf{x}_{\max}$. The optimization problem are solved by a multi-parametric programming technique. The solutions are represented as an explicit look-up table of control inputs. However, the efficiency of proposed multi-parametric based approach can only be guaranteed for small size problems [100], typically 5 to 10 variables including both inputs and states/outputs, and the control and prediction horizons under 5. In addition, it also could be computationally demanding to handle the optimization problem with a more complicated cost function compared to (2.44), such as introducing a penalty term on GP variances in (2.44).

GP variances in the above formulations are not included in the cost function but form part of the constraints. GP variance can be directly incorporated into the cost function if the expectation of (2.44) is used as the cost function instead [24]. This is because

$$\begin{aligned}
&E\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] \\
=&E\left[\sum_{i=1}^{H}\left\{\left(\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\right)^T \mathbf{Q}\left(\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\right) + \mathbf{u}_{k+i-1}^T \mathbf{R}\mathbf{u}_{k+i-1}\right\}\right] \\
=&\sum_{i=1}^{H}\left\{\left(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\right)^T \mathbf{Q}\left(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\right) + \mathbf{u}_{k+i-1}^T \mathbf{R}\mathbf{u}_{k+i-1} + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i})\right\} \\
=&\mathcal{H}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \mathbf{u}_{k-1})
\end{aligned} \tag{2.45}$$

where trace$(\cdot)$ represents the trace of a matrix. Recently, this technique has been implemented in [25, 101] for unconstrained GP based MPC problems. However, its use in the constrained GP based MPC problem has still not been investigated.

There has not been much discussion on the issue of stability in the published works on GP based MPC. In [27], a penalty term on the terminal state to guarantee stability has been suggested but there is no stability analysis given. If stability is viewed as robustness against GP variances, then an Linear Matrix Inequality (LMI) based technique has been used for unconstrained control problems [101]. For the constrained case, a robust GP based MPC algorithm has recently been proposed in [102]. However, this algorithm is based on worst-case analysis. Hence, the obtained control actions are conservative. Furthermore, uncertainty is propagated in this work by using the double-seasonal Holt-Winters technique [103]. This is computationally demanding compared to the approaches in [27, 98] where the uncertainty is naturally propagated as part of the multiple step prediction process.

Chapter 3

Hybrid PSO for Hyperparameters Learning

The key issue of using the GP model is accurately learning the hyperparameters. We propose three enhanced PSO algorithms to address this issue when using the CGP model in Section 3.2. In addition, we also propose using the MSE of outputs as the fitness function when the quality of obtained model is concerned. The simulation results in Section 3.3 demonstrate the effectiveness of CGP based modelling of unknown systems, as well as the optimization performance of proposed PSO algorithms in the hyperparameter learning problem.

All simulations are conducted in the Matlab environment using the "cgpModel" toolbox we developed. It contains the implementation of the modelling and inference algorithms by using the CGP models proposed by M.A Alvarez [59]. Modelling, learning and prediction are performed at the high level by the "cgpModelCreate", "cgpModelLearn" and "cgpModelComputeMeanVar" functions. In addition to the CG, the proposed PSO learning algorithms are also included as part of the toolbox. This allows users to easily compare the performance of these two methods. The users also have a choice of two different cost functions for the PSO algorithm. One is the commonly used log-likelihood function and the other is "squared model error" function. Extensibility of this toolbox is ensured by the modular design. Additions to the current function library of learning methods, covariance functions and cost functions can easily be made.

3.1 Log-Likelihood and MSE Fitness Functions

Similar to the hyperparameters learning of standard GP models discussed in Section 2.3, the hyperparamters of a CGP can be obtained by maximizing the following LL function:

$$\log p(\mathbf{y}|\mathbf{X},\theta) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{y},\mathbf{y}}| - \frac{N}{2}\log 2\pi \tag{3.1}$$

where the covariance matrix $\mathbf{K}_{y,y}$ is defined by (2.31). In practice, we alternatively solve the problem of minimizing the Negative value of Log-Likelihood (NLL) since there are many efficient optimization algorithms available for minimization.

The minimization of the NLL again leads to an unconstrained optimization problem that can be solved by using CG techniques. However, CG algorithms often get stuck at a local optima. Therefore, one has to restart the algorithm a number of times with different initial values. Alternatively, the standard PSO technique can been employed [20]. However, the standard PSO algorithms typically converge very slowly.

In addition to the above problems, there is also the choice of objective function. When using PSO algorithms for learning hyperparameter, equation (3.1) is the natural choice as the objective function. However, there are some issues involved which we shall illustrate with the modelling of a single output nonlinear dynamic system described by the following difference equation:

$$\begin{aligned} y(k) =& 0.893y(k-1) + 0.037y^2(k-1) - 0.05y(k-2) \\ &+ 0.157u(k-1) - 0.05u(k-1)y(k-1) \end{aligned} \tag{3.2}$$

where $u(k)$ is the input and $y(k)$ is the output at time instant k. 1000 uniformly distributed input values are randomly generated within the range of $(-2, 4)$ and their corresponding outputs are computed. From these input-output data, 200 are randomly chosen for training the model. The hyperparameters of the CGP model are learned by minimizing the NLL function. To evaluate the quality of the resulting CGP model , the MSE value of 50 test points are computed by

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{y}_i - \hat{\mathbf{y}}_i(\boldsymbol{\theta})\right)^2 \tag{3.3}$$

where N is the number of test data, $\mathbf{y}_i$ are test outputs, and $\hat{\mathbf{y}}_i$ are corresponding mean values obtained by using (2.38a) given the hyperparameters $\boldsymbol{\theta}$.

Table 3.1: NLL and MSE values of two CGP models of system described by (3.2).

	Model 1	Model 2
NLL	≈ 51	≈ 242269
MSE	0.5313	0.0101

Table 3.1 shows two different CGP models that results from limiting the search range of the hyperparameters to $[0, 100]$ for Model 1 and $[0, 1]$ for Model 2. Based the MSE values, it is clear that Model 2 is able to predict the outputs more accurately than Model 1. However, the NLL value of Model 1 is much smaller than Model 2. If the NLL function is the objective function for minimization, one may conclude that Model 1 is the better model. Thus one cannot use the NLL (and hence LL) values to accurately gauge the quality of intermediate models obtained during the optimization process.

We therefore propose to minimize the MSE function (3.3) to learn CGP's hyperparameters. The optimal values of the hyperparameters are therefore obtained by

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{y}_i - \hat{\mathbf{y}}_i(\theta)\right)^2 \tag{3.4}$$

In addition, the following derivatives of MSE of outputs w.r.t. the hyperparameters can be used to accelerate the optimization process.

$$\frac{\partial}{\partial \boldsymbol{\theta}_l} MSE = -\frac{2}{N} \sum_{i=1}^{N} \left\{ \left(\mathbf{y}_i - \hat{\mathbf{y}}_i(\boldsymbol{\theta})\right) \frac{\partial \hat{\mathbf{y}}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\} \tag{3.5}$$

with

$$\frac{\partial \hat{\mathbf{y}}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}}{\partial \boldsymbol{\theta}} \mathbf{K}_{\mathbf{y},\mathbf{y}}^{-1} \mathbf{y} - \mathbf{K}_{\mathbf{f}^*,\mathbf{f}} \mathbf{K}_{\mathbf{y},\mathbf{y}}^{-1} \frac{\partial \mathbf{K}_{\mathbf{y},\mathbf{y}}}{\partial \boldsymbol{\theta}} \mathbf{K}_{\mathbf{y},\mathbf{y}}^{-1} \mathbf{y} \tag{3.6}$$

where the computation of $\frac{\partial \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathbf{K}_{\mathbf{y},\mathbf{y}}}{\partial \boldsymbol{\theta}}$ can be found in [17, 104]. This technique is in fact widely known as the least-square approach in the literature. In addition, from the viewpoint of non-Bayesian learning, minimizing the MSE is approximately equivalent to maximizing the LL. The proof of equivalence between these two learning strategies can be found in [105].

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
while $t < T_{max}$ **do**
  if $f(\mathbf{G}) \leq \xi$ **then**
    End;
  else
    for $i = 1$ *to* N_p **do**
      for $d = 1$ *to* D **do**
        Update $v_i^d(t)$ by using (3.7);
        Update $x_i^d(t)$ by using (3.9);
      end
      Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (3.10);
      Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (3.11);
    end
  end
  s $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 1: Standard PSO based Hyperparameter Learning

3.2 Enhanced PSOs for Hyperparameter Learning

PSO is an evolutionary computation technique inspired by the social behaviour of organisms [106]. Many particles are initialized simultaneously and each one represents a solution to the problem. Associated with each particle is its position in the solution space and its velocity with which it is moved to a new position. A fitness function is used to evaluate each particle and only particles that are fit enough survive in the competition. After some iterations, the particles would have explored the solution space sufficiently to arrive at the optimal or near optimal solution.

In [20, 28, 29], the standard PSO algorithm has been proven superior to gradient based CG and BFGS approaches in terms of accuracy and efficiency for the optimization problems (2.40) and (3.4). However, poor initializations can lead to poor global search ability, and they exhibit slow convergence due to poor local search ability. We shall first outline the "standard" PSO algorithm for the hyperparameter learning of CGP models. Subsequently, an enhanced algorithm that makes use of multi-start technique is described, followed by the introduction of two other new enhancements.

3.2.1 Standard PSO

Let there be a population of N_p particles, each of which, denoted by $\mathbf{x}_i = [x_i^1, \cdots, x_i^D]_{i=1,\cdots,N_p}^T \in \mathbb{R}^D$, represents a potential solution to the problem (2.40) or (3.4). Each particle also records its best position as $\mathbf{P}_i = [p_i^1, \cdots, p_i^D]^T$ and its best fitness value $V_i^{\text{pbest}} = f(\mathbf{P}_i)$, where $f(\cdot)$ denotes the fitness function and could be (3.1) or (3.3). In addition, the best position of all N_p particles is denoted by $\mathbf{G} = [g^1, \cdots, g^D]^T$ and the corresponding best fitness value is denoted by $V^{\text{gbest}} = f(\mathbf{G})$. In the iteration $t+1$, the velocity of i^{th} particle, given by $\mathbf{v}_i = [v_i^1, \cdots, v_i^D]^T$, along d^{th} dimension is updated according to the following rule,

$$v_i^d(t+1) = \omega(t) v_i^d(t) + c_1 \lambda_1 \left(p_i^d(t) - x_i^d(t)\right) + c_2 \lambda_1 \left(g^d(t) - x_i^d(t)\right) \tag{3.7}$$

where c_1 and c_2 are two acceleration factors, λ_1 and λ_2 are two random values between $[0, 1]$, $\omega(t)$ represents an inertia factor.

In general, a PSO algorithm consists of two search phases, known as "exploration" and "exploitation" respectively. They are governed by the inertia factor $\omega(t)$. The use of a larger value of $\omega(t)$ allows the particle to explore larger areas of the search space during the exploration phase. Meanwhile, a smaller value of $\omega(t)$ restricts the particle to a smaller region of the search space and allows the particle to converge to a local optimum in the exploitation phase. Thus, the inertia factor is usually reduced with time step. A commonly used $\omega(t)$ is defined by,

$$\omega(t) = \omega_{\text{end}} + (\omega_{\text{start}} - \omega_{\text{end}}) \exp(-k \times (\frac{t}{T_{\max}})) \tag{3.8}$$

where ω_{start} and ω_{end} are the pre-determined start and final values respectively, $T_{\max}$ denotes the maximum number of iterations. and the rate of decrease is governed by the constant k.

The new position of a particle can subsequently be obtained by,

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{3.9}$$

For the minimization problem (3.4), the $\mathbf{P}_i$ and V_i^{pbest} at $t+1$ iteration are updated

according to the following rule,

$$\begin{aligned} \mathbf{P}_i(t+1) &= \begin{cases} \mathbf{x}_i(t+1) & f(\mathbf{x}_i(t+1)) \leq f(\mathbf{P}_i(t)) \\ \mathbf{P}_i(t) & f(\mathbf{x}_i(t+1)) > f(\mathbf{P}_i(t)) \end{cases} \\ V_i^{\text{pbest}}(t+1) &= f(\mathbf{P}_i(t+1)) \end{aligned} \tag{3.10}$$

In addition, the $\mathbf{G}$ and V^{gbest} at $t+1$ iteration are updated by,

$$\begin{aligned} \mathbf{G}(t+1) &= \text{argmin}\left\{ f(\mathbf{P}_1(t+1)), \cdots, f(\mathbf{P}_{N_p}(t+1)), f(\mathbf{G}(t)) \right\} \\ V^{\text{gbest}}(t+1) &= f(\mathbf{G}(t+1)) \end{aligned} \tag{3.11}$$

We can also use the rules (3.10) and (3.11) when the maximization problem (2.40) becomes the minimizing the negative of LL function (3.1).

For our hyperparameter learning problem, each particle is defined by

$$\boldsymbol{\theta} = \{\boldsymbol{\theta}_{K1}, ..., \boldsymbol{\theta}_{KM}, \boldsymbol{\theta}_{L1}, ..., \boldsymbol{\theta}_{LQ}\} \tag{3.12}$$

where $\boldsymbol{\theta}_{Kd} = \{\nu_{d1}, ..., \nu_{dQ}, \mathbf{P_d}\}_{d=1,\cdots,M}$ represents the hyperparameters of smoothing kernels (2.29), and $\boldsymbol{\theta}_{Lq} = \{\upsilon_q, \mathbf{P_q}\}_{q=1,...,Q}$ are the hyperparameters of latent functions (2.30). The algorithm of standard PSO based hyperparameter learning is presented in Algorithm 1.

3.2.2 Multi-Start PSO

In the "exploration" stage of the optimization process, we want the particles to explore as much of the search space as possible. This can be achieved by setting the inertia factor $\omega(t)$ to a suitably large value which in turn is determined by ω_{start} and ω_{end} in (3.8). However, suitable values for these two constants are quite specific to each problem. Another way to achieve this objective is to diversify the swarm by introducing new particles. With the multi-start PSO proposed here, all particles will be reinitialized if the global best position $\mathbf{G}$ remains unchanged or slightly changed for a given number of iterations N_G. One may question whether the potentials of old particles should be exploited. However, in the "exploration" stage, global search is more important than local ones. It has also been proposed that only those particles that are trapped in a local optimum should be reinitialized [107]. But this approach requires checking the changes of multiple $f(\mathbf{P}_i)$ which is time-consuming. Our algorithm is therefore simpler because

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
Multi-start PSO parameters: η, N_G, $N_\eta = 0$;
while $t < T_{max}$ **do**
  if $N_\eta = N_G$ **then**
    Randomly regenerated $\boldsymbol{\theta}$;
    $N_\eta = 0$;
  else
    if $f(\mathbf{G}) \leq \xi$ **then**
      End;
    else
      for $i = 1$ *to* N_p **do**
        for $d = 1$ *to* D **do**
          Update $v_i^d(t)$ by using (3.7);
          Update $x_i^d(t)$ by using (3.9);
        end
        Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (3.10);
        Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (3.11);
      end
      if $\|f(\mathbf{G}(t)) - f(\mathbf{G}(t-1))\| \leq \eta$ **then**
        $N_\eta = N_\eta + 1$;
      else
        $N_\eta = 0$;
      end
    end
  end
  $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 2: Multi-Start PSO based Hyperparameter Learning

only the change of $f(\mathbf{G})$ need to be checked. It is described in Algorithm 2.

3.2.3 Gradient-based PSO

Standard PSO also suffers from slow convergence during the "exploitation" phase. This issue can be solved through using the gradient/derivative information especially when it approaches to the global or local optima. In this section, a gradient-based PSO is proposed for the hyperparameters learning problem by combining the standard PSO and CG algorithm. In particular, the current global best position $\mathbf{G}$ will be exploited by solving the problem 2.40) or (3.4) by using the CG algorithm. The obtained solution is subsequently used to replace the current global position in the PSO algorithm if it produces a better fitness value. Compared with the existing work in [108] where all particles are exploited by using a gradient-based method, the proposed algorithm only

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
Gradient-based PSO parameters: η, N_G, $N_\eta = 0$;
while $t < T_{max}$ **do**
  if $N_\eta = N_G$ **then**
    Initializing CG parameters, $\boldsymbol{\theta}_0 = \mathbf{G}$;
    Solving the problem (2.40) or (3.4) to obtain $\boldsymbol{\theta}^*$;
    if $f(\boldsymbol{\theta}^*) \leq f(\mathbf{G}(t))$ **then**
      $\mathbf{G}(t+1) = \boldsymbol{\theta}^*$;
    else
      $\mathbf{G}(t+1) = \mathbf{G}(t)$;
    end
    $N_\eta = 0$;
  else
    if $f(\mathbf{G}) \leq \xi$ **then**
      End;
    else
      for $i = 1$ *to* N_p **do**
        for $d = 1$ *to* D **do**
          Update $v_i^d(t)$ by using (3.7);
          Update $x_i^d(t)$ by using (3.9);
        end
        Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (3.10);
        Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (3.11);
      end
      if $\|f(\mathbf{G}(t)) - f(\mathbf{G}(t-1))\| \leq \eta$ **then**
        $N_\eta = N_\eta + 1$;
      else
        $N_\eta = 0$;
      end
    end
  end
  $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 3: Gradient-based PSO based Hyperparameter Learning

conducts gradient-based search on the current global best position if its fitness value remains unchanged or slightly changed for a specified number of iterations N_G. The computational burden of using proposed algorithm is essentially reduced. The gradient based PSO for the hyperparameter learning of CGP models is given in Algorithm 3.

3.2.4 Hybrid PSO

The multi-start method in Section 3.2.2 and the gradient-based method in Section 3.2.3 can be combined in a single PSO algorithm so that both the "exploration" and the "ex-

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
Hybrid PSO parameters: τ, η, N_G, $N_\eta = 0$;
while $t < T_{max}$ **do**
  if $N_\eta = N_G$ **then**
    if $t \leq \tau \times T_{max}$ **then**
      Randomly regenerated $\boldsymbol{\theta}$;
    else
      Initializing CG parameters, $\boldsymbol{\theta}_0 = \mathbf{G}$;
      Solving the problem (2.40) or (3.4) to obtain $\boldsymbol{\theta}^*$;
      if $f(\boldsymbol{\theta}^*) \leq f(\mathbf{G}(t))$ **then**
        $\mathbf{G}(t+1) = \boldsymbol{\theta}^*$;
      else
        $\mathbf{G}(t+1) = \mathbf{G}(t)$;
      end
    end
    $N_\eta = 0$;
  else
    if $f(\mathbf{G}) \leq \xi$ **then**
      End;
    else
      for $i = 1$ *to* N_p **do**
        for $d = 1$ *to* D **do**
          Update $v_i^d(t)$ by using (3.7);
          Update $x_i^d(t)$ by using (3.9);
        end
        Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (3.10);
        Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (3.11);
      end
      if $\|f(\mathbf{G}(t)) - f(\mathbf{G}(t-1))\| \leq \eta$ **then**
        $N_\eta = N_\eta + 1$;
      else
        $N_\eta = 0$;
      end
    end
  end
  $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 4: Hybrid PSO based Hyperparameter Learning

ploitation" phases of the optimization process are enhanced. This leads to the proposed hybrid PSO algorithm. In particular, the multi-start technique is first used such that the search space can be well covered. When the number of iterations N_G reaches a given proportion η of maximum iteration number, the optimization process is considered to have approached near global or local optima. The algorithm subsequently switches to the use of gradient-based technique. This allows a faster convergence rate due to the nature of using gradient-based solution compared to the use of rules (3.7) and (3.9). The

Table 3.2: Parameters used in the simulations

Symbol	Description	Quantity
N_p	PSO population	20
$\mathbf{T}_{\max}$	Maximum Iterations	500
c_1,c_2	Acceleration Factors	1.5
ω_{start}	Start Inertial Factor	0.4
ω_{end}	End Inertial Factor	0.9
k	Shape Control Factor	0.8
CG Restarts	Restart Times	20×500
$\|\Delta\xi\|$	Minimum Fitness Variation	10^{-5}
$\nu_{d,i}, \upsilon_q$ α_i, β_j	Coefficients Search Range $\mathbf{P_d}$, $\mathbf{P_q}$ Elements Search Range	$[0, 100]$ for LTV
		$[0, 100]$ for NLTV with "Step"
		$[0, 1]$ for NLTV with "Curve"

proposed hybrid PSO is conceptually simple and allows to adjust the proportion η to suit the problem. The use of hybrid PSO in the problem of CGP models' hyperparameter learning is given in Algorithm 4.

3.3 Simulation Results

The optimization performances of proposed PSO based algorithms for CGP hyperparameters learning are demonstrated in the modelling of non-trivial MISO and MIMO systems. The proposed PSOs are compared with the CG algorithm. In addition, both the NLL and MSE are used as the fitness function, respectively. The simulations are repeated 50 times with same training and test samples. The results in terms of MSE value, convergence rate and computer run-time are averaged values.

All simulations are performed on a computer with a 3.40GHz Intel® Core™ 2 Duo CPU with 16 GB RAM, using Matlab® version 8.1. In addition, the parameters related to the CGP and PSO models in the simulations are listed in the Table 3.2. Note that the number of restarts for CG is designed to give a fair comparison with PSO with the given population size and number of iterations.

3.3.1 Standard PSO with MSE Fitness

We study the effectiveness and benefit of PSO based hyperparameters learning method. The proposed PSO with MSE fitness is compared to existing NLL fitness based PSO, as well as commonly used CG algorithm. The convergence of proposed PSO is not discussed due to it is not our concern in this simulation.

Single-Output Modelling

The numerical system used in the simulation is described by the following difference equation,

$$\begin{aligned} y(k) =& 0.893y(k-1) + 0.037y^2(k-1) - 0.05y(k-2) \\ &+ 0.157u(k-1) - 0.05u(k-1)y(k-1) \end{aligned} \tag{3.13}$$

where $u(k)$ is the input and $y(k)$ is the output. Although this dynamical system has only 1 input and 1 output, the CGP modelling inputs will be $u(k-1)$, $y(k-1)$ and $y(k-2)$, making it a 3-input and 1-output model. Only a single output is used here for modelling to simplify the comparison. In addition, we randomly chose 1000 inputs in $u \sim \mathcal{U}(-2, 4)$ and apply them into the system. This allows us collect 1000 observations including inputs, states and outputs.

First we aim to verify the effectiveness of minimizing the model errors by using standard PSO in the hyperparameters learning problem. The 50 simulations are performed with the same 200 training and 50 test data randomly selected from the 1000 observations. To study the influence of PSO population size, the simulations are also independently performed for each population size of 10, 25, 50 and 100. The obtained results are given in Table 3.3 in terms of Mean Absolute Error (MAE) and average variance (Var) values, where PSO/1 represents the proposed approach and PSO/2 denotes the PSO with NLL fitness. Overall, the both two approaches produce equally good CGP model due to close MAE and Var values. In addition, the results also suggest that a value of 25 to 50 may be a good choice of PSO population size. This is because a bigger size normally requires much more runtime (exponentially increasing).

Next, we want to determine the effect of the hyperparameters search space. Two different cases are considered here. In the first case, it is assumed that a prior knowledge

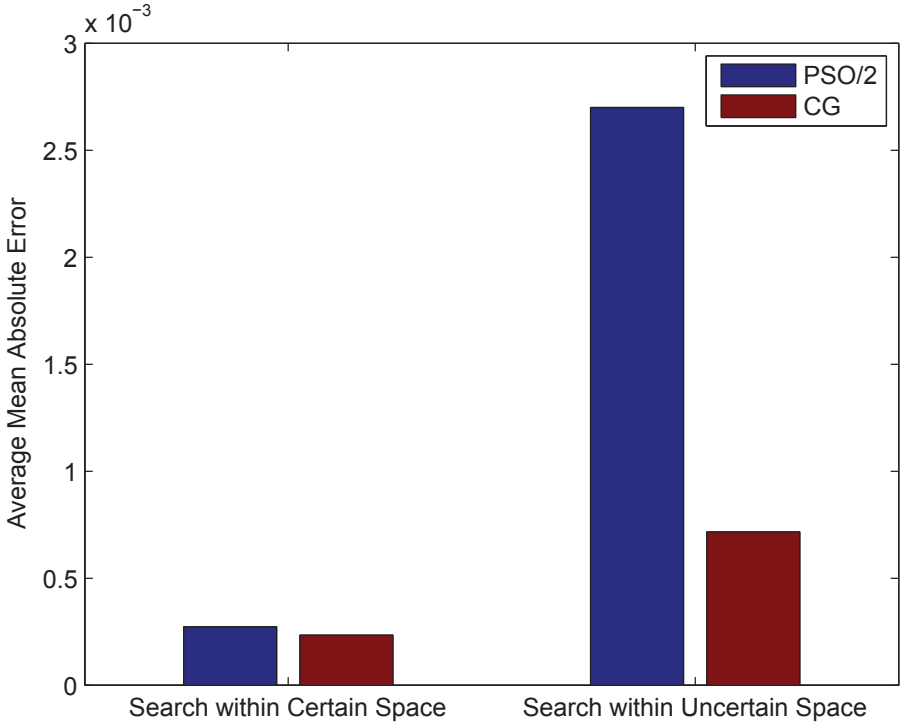


Figure 3.1: Obtained MAE in the single-output dynamical system modelling over 50 runs

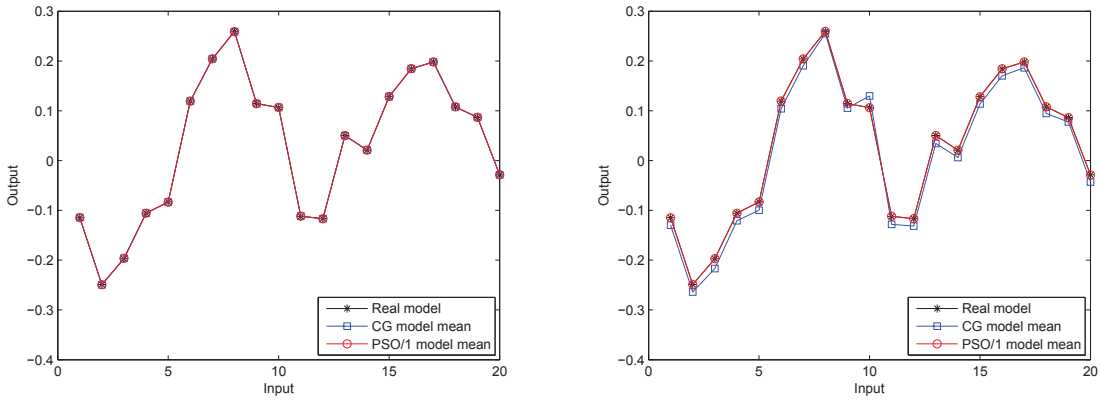
Table 3.3: Comparison of two PSOs with different population sizes

N_p	MAE		Var	
	PSO/1	PSO/2	PSO/1	PSO/2
10	0.2297	0.2355	1.52e-02	2.44e-02
25	0.0054	0.0047	9.05e-03	3.64e-03
50	0.0022	0.0021	8.66e-03	4.37e-03
100	0.0011	0.0012	9.14e-03	9.74e-04

of value ranges for the parameters in (3.12) is available. Specifically, they are

$$\begin{aligned} \nu_{d,i}, \upsilon_q &\in \{1, 2\} \\ \alpha_i, \beta_j &\in \{0, 1\} \end{aligned} \tag{3.14}$$

where α_i and β_j are the elements of the diagonal precision matrices $\mathbf{P_d}$ and $\mathbf{P_q}$ respectively. In the second case, we do not assume any prior knowledge of the value ranges. The obtained MAE values using the proposed PSO are given in Figure 3.1 and are compared to those using CG. The results show that the learnt CGP models by using PSO and CG perform equally good when the search space is well defined. Figure 3.2.a confirms that



3.2.a: Well defined search range

3.2.b: Not well defined search range

Figure 3.2: Predicted outputs in the single-output simulations

Table 3.4: Results of Linear Relationship

	MAE		SE	
	PSO	CG	PSO	CG
$\mathbf{y}_1$	1.41E-04	4.46E-04	0084	0.0327
$\mathbf{y}_2$	1.11E-04	2.18E-04	0.147	0.0446

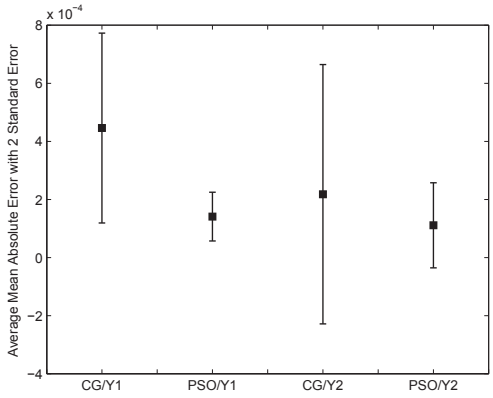
Table 3.5: Results of Nonlinear Relationship

	MAE		SE	
	PSO	CG	PSO	CG
$\mathbf{y}_1$	4.41E-04	5.36E-04	0.0065	0.043
$\mathbf{y}_2$	3.57E-04	8.11E-04	0.0064	0.0356

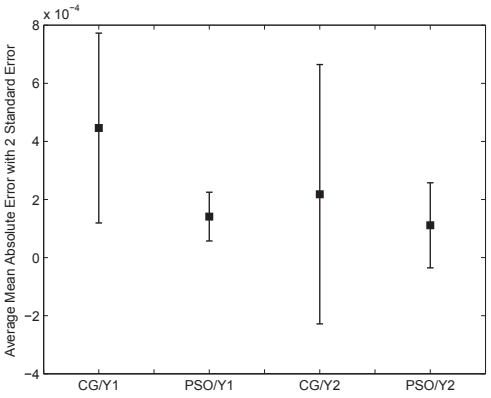
the predicted outputs are very close to the actual values. However, when the search space is not well constrained, the proposed PSO outperforms CG by a wide margin. Figure 3.2.b shows that while the predicted CGP outputs by the PSO are still very close to the actual values, there are some clear deviations with by the CG.

3.3.2 Two-output Modelling

Systems with multiple-outputs can be modelled in two different ways. One is to use multiple single-output models and the other is to provide a single model for all outputs at the same time. While the first approach is often simpler, the latter approach is able to capture correlation between outputs. For example, a robot arm system with multiple degrees of freedom has multiple outputs that are strongly correlated. Another example is



3.3.a: $\mathbf{y}_2 = -\mathbf{y}_1$



3.3.b: $\mathbf{y}_2 = \mathbf{exp}(\mathbf{y}_1)$

Figure 3.3: MIMO dynamical system modelling results: MAE and 2 standard errors (divided by 0.01) over 50 runs

the prediction of steel mechanical properties in [109], where the yield and tensile strength are predicted from the chemical compositions and grain size. These two "outputs" are highly correlated.

We shall continue to use the dynamical system in (3.13). Since it has only one output y (denoted y_1 here), a second output y_2 will be created which is a function of y_1. Two such functions are considered, one linear and the other nonlinear, given by $\mathbf{y}_2 = -\mathbf{y}_1$ and $\mathbf{y}_2 = \mathbf{exp}(\mathbf{y}_1)$ respectively. Two different sets of training data, each has 200 samples, are selected from the 1000 observations. The test data consists of 50 samples which are different from the training samples.

Tables 3.4 and 3.5 show the performances of using PSO and CG. The same results are also shown in Figure 3.3 with an indication of the deviations. For outputs $\mathbf{y}_1$ and $\mathbf{y}_2$ in both two MIMO systems, the CGP models learnt by using PSO exhibit smaller MAE and standard error (SE) values.

3.3.3 Enhanced PSO Algorithms

Next, we demonstrate the optimization performances of proposed enhanced PSO algorithms for CGP hyperparameters learning problem. Three enhanced PSOs are compared with the standard PSO and CG algorithms in the modelling two non-trivial MIMO systems, with respect to the model accuracy, runtime as well as convergence.

LTV System Modelling

Consider a 2-input-2-output Linear Time-Varying (LTV) system [110] defined by,

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(t) \cdot \mathbf{x}(t) + \mathbf{B}(t) \cdot \mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}(t) \cdot \mathbf{x}(t) + \mathbf{D}(t) \cdot \mathbf{u}(t) \end{aligned} \tag{3.15}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ are defined as:

$$\begin{aligned} \mathbf{A}(t) &= \begin{bmatrix} 0.3 - 0.9\Gamma_{1t} & 0.1 & 0.7\Gamma_{2t} \\ 0.6\Gamma_{1t} & 0.3 - 0.8\Gamma_{2t} & 0.01 \\ 0.5 & 0.15 & 0.6 - 0.9\Gamma_{1t} \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \mathbf{C} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \mathbf{D} = 0.1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{3.16}$$

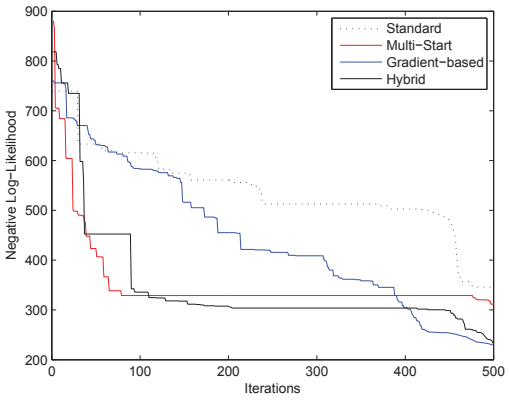
Matrix $\mathbf{A}$ has time-varying parameters $\Gamma_{1t} = \sin(10t)$ and $\Gamma_{2t} = \cos(10t)$. The two control inputs are given by $u_1(t) = 0.5\sin(12t)$ and $u_2(t) = \cos(7t)$. They have zero initial conditions.

Using a sampling interval of $0.05s$, 200 data records which include the inputs, states and outputs are generated. Out of these data, 60 samples are selected randomly for training the CGP models, and all 200 samples are employed for testing.

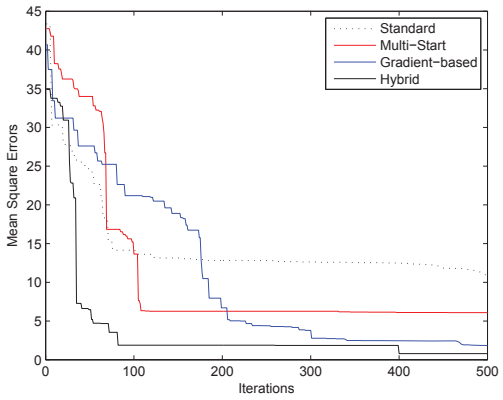
First, the results of using LL fitness function are discussed. Table 3.6 shows the MSE of the predicted outputs of the obtained CGP models. These results show that, the gradient and hybrid PSO algorithms are able to produce smaller MSE than the other two PSO algorithms. However, the runtime of the gradient PSO is longer than the other three PSO algorithms due to gradient computations. It is also interesting to note that the CG algorithm took significantly longer to converge to the solution than PSO algorithms.

The convergence behaviour of the PSO algorithms with LL fitness function is shown in Figure 3.4.a, averaged over 50 runs. It shows that both the hybrid and multi-start algorithms converges to a near-optimal solution much faster than the other two alternatives. Thus it can be concluded that these two algorithms perform better global search ability at the early stages (approximately before 150 iterations) than standard and gradient PSO.

Next we discuss the results of using MSE fitness. Due to similar MSE values and runtime of 4 PSO algorithms are produced, therefore they are not shown here. The convergence behaviour of the PSO algorithms with MSE fitness function averaged over



3.4.a: Log-Likelihood Fitness



3.4.b: MSE Fitness

Figure 3.4: Convergence behaviour of the four PSO algorithms in modelling the LTV system

Table 3.6: CGP model accuracies over 50 runs for the LTV system.

	PSO				CG
	Standard	Gradient	Multi-Start	Hybrid	
MSE of y_1	0.4413	0.1159	0.3882	0.1975	0.2235
MSE of y_2	0.4934	0.1699	0.3806	0.1556	0.2473
Time	≈18s	≈23s	≈18s	≈19s	$\gg$ 5min

50 runs are given in Figure 3.4.b. The results show again that, overall the hybrid PSO has the best optimization performance than other 3 PSO algorithms due to the global search ability as well as multi-start PSO and local search ability as well as gradient PSO.

NLTV System Modelling

The second simulation involves the CGP modelling of a Nonlinear Time-Varying (NLTV) system controlled by a Partial Form Dynamic Linearization (PFDL) based Model-Free Adaptive Control (MFAC) controller with the same parameters as in [111]. The 4-input

and 2-output numerical system is described by,

$$\begin{aligned}
x_{11}(k+1) =& \frac{x_{11}(k)^2}{1+x_{11}(k)^2} + 0.3x_{12}(k) \\
x_{12}(k+1) =& \frac{x_{11}(k)^2}{1+x_{12}(k)^2+x_{21}(k)^2+x_{22}(k)^2} + a(k)u_1(k) \\
x_{21}(k+1) =& \frac{x_{21}(k)^2}{1+x_{21}(k)^2} + 0.2x_{22}(k) \\
x_{22}(k+1) =& \frac{x_{21}(k)^2}{1+x_{11}(k)^2+x_{12}(k)^2+x_{22}(k)^2} + b(k)u_2(k) \\
y_1(k+1) =& x_{11}(k+1) + 0.005 * \text{rand}(1) \\
y_2(k+1) =& x_{21}(k+1) + 0.005 * \text{rand}(1)
\end{aligned} \tag{3.17}$$

where the time-varying parameters are given by,

$$a(k) = 1 + 0.1\sin(2\pi k/1500) \tag{3.18a}$$

$$b(k) = 1 + 0.1\cos(2\pi k/1500) \tag{3.18b}$$

This system is to track two trajectories. One involves a "Step" trajectory given by,

$$y_1^*(k) = \begin{cases} 0.4 & k \leq 500 \\ 0.7 & 500 < k \leq 1000 \\ 0.5 & 1000 < k \leq 1500 \end{cases} \tag{3.19a}$$

$$y_2^*(k) = \begin{cases} 0.6 & k \leq 300 \\ 0.8 & 300 < k \leq 700 \\ 0.7 & 700 < k \leq 1200 \\ 0.5 & 1200 < k \leq 1500 \end{cases} \tag{3.19b}$$

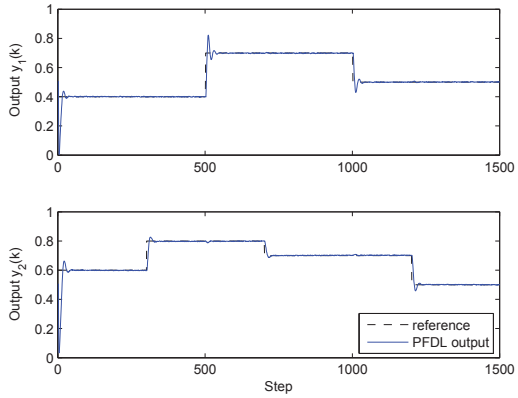
the other is "Curve" trajectory specified by,

$$y_1^*(k) = 0.75\sin(\frac{\pi k}{8}) + 0.5\cos(\frac{\pi k}{4}) \tag{3.20a}$$

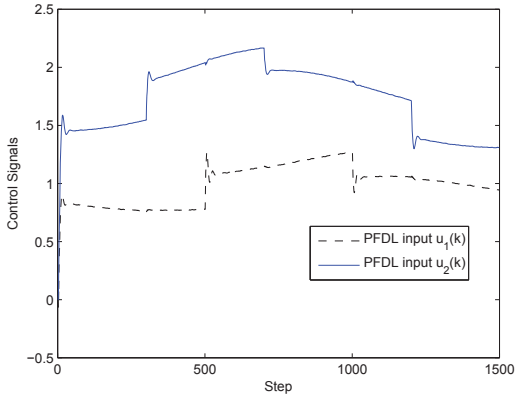
$$y_2^*(k) = 0.5\cos(\frac{\pi k}{8}) + 0.5\sin(\frac{\pi k}{4}) \tag{3.20b}$$

The initial values of the system are [112]: $x_{11}(1) = x_{11}(2) = x_{21}(1) = x_{21}(2) = 0.5$, $x_{12}(1) = x_{12}(2) = x_{22}(1) = x_{22}(2) = 0$, and $u_1(1) = u_1(2) = u_2(1) = u_2(2) = 0$. The reference outputs and inputs for the two trajectories are shown in Figures 3.5.

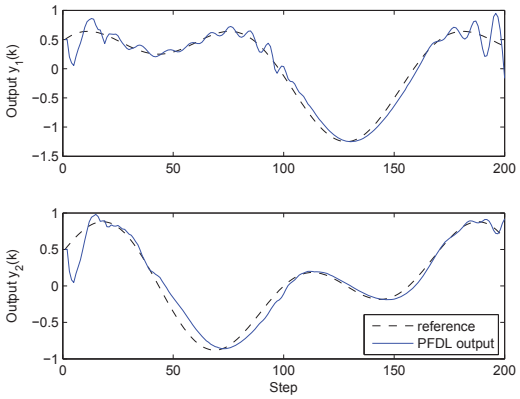
First, we discuss the results of using LL as fitness function. Table 3.7 shows the results of CGP modelling of the NLTV system tracking both the "Step" and "Curve" trajectories



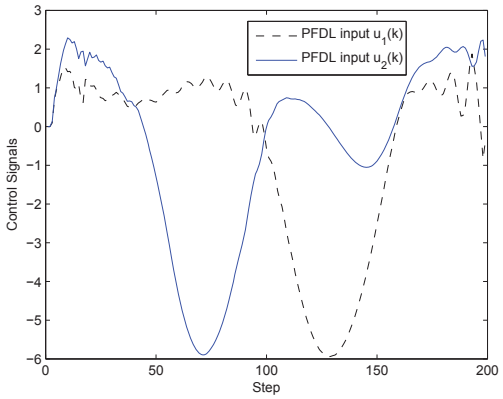
3.5.a: "Step" Trajectory–Outputs $y(k)$



3.5.b: "Step" Trajectory–Inputs $u(k)$



3.5.c: "Curve" Trajectory–Outputs $y(k)$

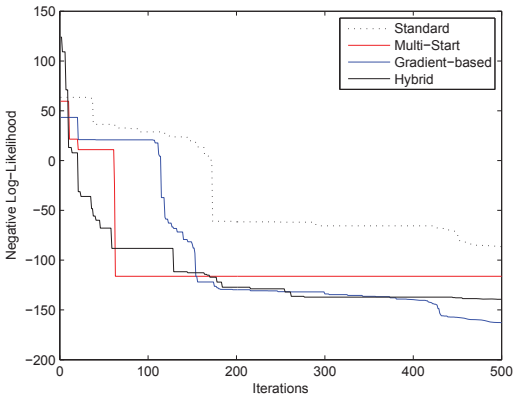


3.5.d: "Curve" Trajectory–Inputs $u(k)$

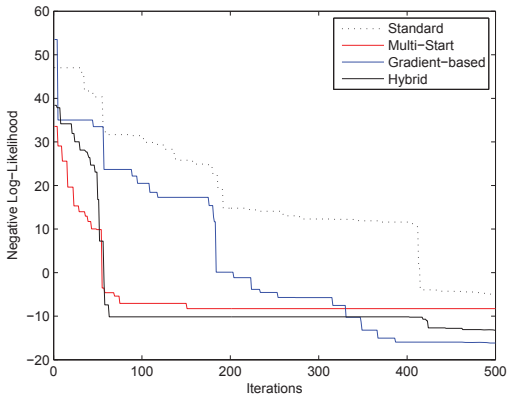
Figure 3.5: Reference PFDL inputs and outputs for the two trajectories

using the five different algorithms for hyperparameter learning. These results are average of 50 independent simulations. All three enhanced PSO algorithms outperform the standard PSO as well as CG. Specifically, the hybrid PSO produces the second best model accuracy (slightly bigger than the best one of gradient PSO), and while the runtime is comparable to the standard and multi-start PSO. Figure 3.6.a shows the convergence behaviour of the various PSO algorithms for the "Step" trajectory. In this case, the hybrid and multi-start PSO converge faster than the other PSO algorithms initially and the hybrid algorithm eventually achieves a smaller model error. The similar convergence behaviours of the "Curve" trajectory can be found in Figure 3.6.b .

Next, the convergence behaviours of PSO algorithms when using MSE fitness are discussed. For the "Step" trajectory simulation, three enhanced PSOs approach the near-



3.6.a: "Step" Trajectory



3.6.b: "Curve" Trajectory

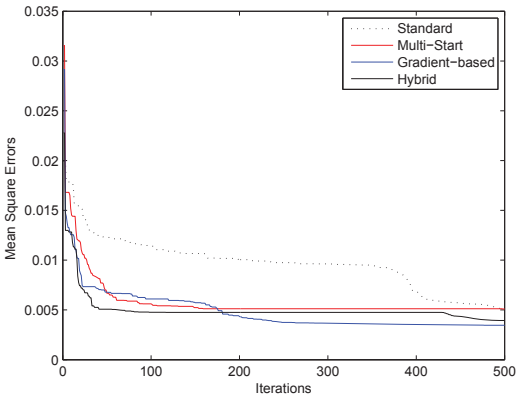
Figure 3.6: Convergence behaviour of the four PSO algorithms with LL fitness in modelling the NLTV system

Table 3.7: CGP model accuracies over 50 runs for the NLTV system.

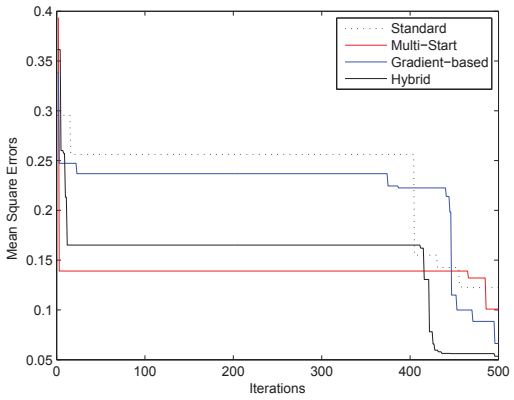
	PSO				CG
	Standard	Gradient	Multi-Start	Hybrid	
"Step" Trajectory					
MSE of y_1	0.8378	0.0084	0.0179	0.0124	0.1221
MSE of y_2	0.2218	0.0062	0.0337	0.0127	0.1273
Time	≈28s	≈40s	≈28s	≈31s	≫ 5min
"Curve" Trajectory					
MSE of y_1	0.3083	0.0417	0.1594	0.0633	0.1541
MSE of y_2	0.1627	0.0402	0.1098	0.0516	0.2333
Time	≈27s	≈39s	≈27s	≈29s	≫ 5min

optima faster than standard PSO algorithm while gradient and hybrid PSOs eventually achieve the smaller model errors. In addition, for the "Curve" trajectory simulation, three enhanced PSOs perform better optimization ability than standard PSO algorithms again. Specifically, the multi-start and hybrid PSOs converge faster while the gradient and hybrid PSOs are able to produce smaller MSE values.

The trade-off between the size of training data and the model accuracy is demonstrated using the hybrid PSO algorithm. The results, given in Table 3.8, are again averaged over 50 simulations. The training data are uniformly chosen from the control interval shown in Figures 3.5.b and 3.5.d. As expected, the model accuracy improves as the training data size increases. However, the algorithm runtime increases exponentially with data size. For a data size of 100, the model error and PSO runtime for both tra-



3.7.a: "Step" Trajectory



3.7.b: "Curve" Trajectory

Figure 3.7: Convergence behaviour of the four PSO algorithms with MSE fitness in modelling the NLTV system

Table 3.8: Effects of training data size on model error and hybrid PSO runtime.

Training Data Size	Average MSE		Time
	y_1	y_2	
"Step" Trajectory			
20	0.0377	0.0511	≈12s
40	6.1475e-04	7.611e-04	≈17s
100	1.1292e-04	1.3543e-04	≈31s
200	1.3411e-05	1.8854e-05	≈110s
"Curve" Trajectory			
25	0.0562	0.0665	≈14s
50	0.0031	0.0032	≈18s
75	0.0012	0.0011	≈23s
100	1.1712e-04	1.9201e-04	≈29s

jectories are similar. Furthermore, the results also show that the system with piecewise constant outputs ("Step" trajectory) can be modelled with far fewer training data compared with the one with smooth outputs ("Curve" trajectory). This is despite the jump discontinuities in the outputs.

3.4 Conclusion

The hyperparameters of the GP models are conventionally learnt by minimizing the NLL function. This typically leads to an unconstrained nonlinear non-convex optimization

problem that is usually solved by using the CG algorithm. Three enhanced PSO algorithms have been proposed in this chapter to improve the hyperparameter learning for CGP models of MIMO systems. They make use of gradient information and also combine it with the multi-start technique. Using a LTV and a NLTV system, we have shown that these algorithms are more effective in avoiding getting stuck in local optima. Hence they are able to produce more accurate models of the systems. Results showed that the hybrid PSO algorithm allows the faster convergence and produces the more accurate models. These algorithms also may make use of the MSE of the outputs rather than the LL function as the fitness function for optimization. This enables us to assess the quality of intermediate solutions more directly.

Chapter 4

Unconstrained Model Predictive Control Using Gaussian Process Models

The focus of this chapter is on unconstrained MPC of unknown systems modelled by GP models. As an effective implementation is described in Section 4.1. The problem is solved efficiently by using a gradient based solution. The simulation results in Section 4.2 demonstrate that the proposed solution is effective and computationally efficient.

4.1 Unconstrained MPC based on GP Models

4.1.1 Unknown Dynamical System Modelling using GP

Consider a dynamical system with states $\mathbf{x} \in \mathbb{R}^n$ and controls $\mathbf{u} \in \mathbb{R}^m$ which are related by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, k) + \mathbf{w}_k \tag{4.1}$$

where k is the integer index of time, $f(\cdot)$ is an unknown nonlinear time-varying function, and $\mathbf{w} \in \mathbb{R}^n$ represents Gaussian noise with zero mean and variance $\mathbf{\Sigma}_w$. To model this system using GP model, we use the state-control tuples $\tilde{\mathbf{x}}_k = (\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}^{n+m}$ and state differences $\delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k \in \mathbb{R}^n$ are used as training inputs and targets respectively [113, 114]. This approach can be advantageous when changes in $\delta\mathbf{x}$ are less than changes in $\mathbf{x}$. When there are multiple targets, a separate GP model can be trained for each independent target.

A GP model is completely specified by its mean and covariance function [17]. Without loss of generality, it is usually assume that the prior mean is zero and the squared exponential covariance is computed by $\mathbf{K}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \sigma_s^2 \exp(-\frac{1}{2}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)^T \boldsymbol{\Lambda}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)) + \sigma_n^2$, where σ_s^2, σ_n^2 and entries of matrix $\boldsymbol{\Lambda}$ are hyperparameters of a GP model. Thus, given D training inputs $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_D]$ and corresponding training targets $\mathbf{y} = [\Delta\mathbf{x}_1, \cdots, \Delta\mathbf{x}_D]^T$ at the sampling time k, the joint distribution between training targets and a test target $\Delta\mathbf{x}_k^*$ at a training input $\tilde{\mathbf{x}}_k^*$ follows a Gaussian distribution

$$p\begin{pmatrix} \mathbf{y} \\ \Delta\mathbf{x}_k^* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{matrix} \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I} & \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k^*) \\ \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}}) & \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{x}}_k^*) \end{matrix}\right) \tag{4.2}$$

Furthermore, through restricting the joint distribution to only contain those targets that agree with collected observations, we can obtain the Gaussian posterior distribution with mean and variance function

$$m(\tilde{\mathbf{x}}_k^*) = E_f[\Delta\mathbf{x}_k^*] = \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{y} \tag{4.3a}$$

$$\begin{aligned} \sigma^2(\tilde{\mathbf{x}}_k^*) = Var_f[\Delta\mathbf{x}_k^*] &= \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{x}}_k^*) \\ &\quad - \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k^*) \end{aligned} \tag{4.3b}$$

where $\mathbf{K}_\sigma = \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I}$. Then it is known that the state at next sampling time also is a GP

$$p(\mathbf{x}_{k+1}) \sim \mathcal{N}(\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}) \tag{4.4}$$

where

$$\boldsymbol{\mu}_{k+1} = \mathbf{x}_k + m(\tilde{\mathbf{x}}_k^*) \tag{4.5a}$$

$$\boldsymbol{\Sigma}_{k+1} = \sigma^2(\tilde{\mathbf{x}}_k^*) \tag{4.5b}$$

The approaches to learn hyperparameters $\boldsymbol{\theta} = [\sigma_s, \sigma_n, \mathrm{vec}(\boldsymbol{\Lambda})]$ are presented in Chapter 3, where $\mathrm{vec}(\cdot)$ denotes vectorization of given matrix.

4.1.2 Uncertainty propagation

The obtained GP model makes one-step ahead prediction by using (4.3). For multiple-step predictions, the conventional way is to perform multiple one-step ahead predictions using estimated mean values iteratively. However, the uncertainties induced by each successive prediction are not taken into account. This issue has been shown to be important

in [91] with a time-series prediction task.

The uncertainty propagation problem can be dealt with by assuming that the joint distribution of the training input at sample time k is uncertain and follows a Gaussian distribution

$$p(\tilde{\mathbf{x}}_k) = p\big(\mathbf{x}_k, \mathbf{u}_k\big) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k) \tag{4.6}$$

with the mean and variance

$$\tilde{\boldsymbol{\mu}}_k = \Big[\boldsymbol{\mu}_k, E\big[\mathbf{u}_k\big]\Big]^T \tag{4.7a}$$

$$\tilde{\boldsymbol{\Sigma}}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k & Cov\big[\mathbf{x}_k, \mathbf{u}_k\big] \\ Cov\big[\mathbf{u}_k, \mathbf{x}_k\big] & Var\big[\mathbf{u}_k\big] \end{bmatrix} \tag{4.7b}$$

where $E\big[\mathbf{u}_k\big]$ and $Var\big[\mathbf{u}_k\big]$ are mean and variance of system controls, $Cov\big[\mathbf{x}_k, \mathbf{u}_k\big] = E\big[\mathbf{x}_k\mathbf{u}_k\big] - \boldsymbol{\mu}_k E\big[\mathbf{u}_k\big]$.

The exact predictive distribution of the training target then can be obtained by integrating over the training input distribution

$$p(\Delta\mathbf{x}_k^*) = \int p(f(\tilde{\mathbf{x}}_k^*)|\tilde{\mathbf{x}}_k^*)p(\tilde{\mathbf{x}}_k^*)d\tilde{\mathbf{x}}_k^* \tag{4.8}$$

However, this integration is analytically intractable due to a nonlinear mapping of a GP distribution results in a non-Gaussian distribution. One possible solution is using the Monte-Carlo approach to obtain a numerical approximation. In [115], a moment-matching based approach is proposed to obtain an analytical Gaussian approximation. Its the mean and variance at an uncertain input are computed by the laws of iterated expectations and conditional variances respectively

$$m(\tilde{\mathbf{x}}_k^*) = E_{\tilde{\mathbf{x}}_k^*}\Big[E_f\big[\Delta\mathbf{x}_k^*\big]\Big] \tag{4.9a}$$

$$\sigma^2(\tilde{\mathbf{x}}_k^*) = E_{\tilde{\mathbf{x}}_k^*}\Big[Var_f\big[\Delta\mathbf{x}_k^*\big]\Big] + Var_{\tilde{\mathbf{x}}_k^*}\Big[E_f\big[\Delta\mathbf{x}_k^*\big]\Big] \tag{4.9b}$$

The further elaborations of (4.9) can be found in Appendix A. Then the state distribution at sampling time $k+1$ in (4.5) becomes

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + m(\tilde{\mathbf{x}}_k^*) \tag{4.10a}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{k+1} =& \boldsymbol{\Sigma}_k + \sigma^2(\tilde{\mathbf{x}}_k^*) \\ &+ Cov\big[\mathbf{x}_k, \Delta\mathbf{x}_k\big] + Cov\big[\Delta\mathbf{x}_k, \mathbf{x}_k\big]\end{aligned} \tag{4.10b}$$

where the computation of $Cov\big[\mathbf{x}_k, \Delta\mathbf{x}_k\big]$ can be found in Appendix B.

The computational complexity of GP inference (4.9) is $\mathcal{O}(D^2n^2(n+m))$. Thus, GP is normally only suitable for problems with limited dimensions (under 12 as suggested by most publications) and limited size of training data. For problems with higher dimensions, a sparse approximation technique is often used to reduce the computational burden when using GP models. A review of commonly used sparse approximations can be found in [116].

4.1.3 GP based MPC

Consider an unconstrained MPC optimal control problem with the following objective function

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) \tag{4.11}$$

where the cost function $\mathcal{J}$ is given by

$$\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) = \sum_{i=1}^{H} \left\{ (\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q} (\mathbf{x}_{k+i} - \mathbf{r}_{k+i}) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right\} \tag{4.12}$$

Here, $\mathbf{r}$ denotes the target reference, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ are positive definite weighting matrices, and the prediction horizon H is assumed to be same as the control horizon.

If the dynamical system (4.1) is represented by GP models, the predictions of $\mathbf{x}_k$ are stochastic. Hence the MPC is a stochastic one and (4.11) becomes [117, 118]

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} E[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})] \tag{4.13}$$

The expected value of the cost function can be derived as

$$\begin{aligned} E[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})] &= E\left[\sum_{i=1}^{H} \left\{ (\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q} (\mathbf{x}_{k+i} - \mathbf{r}_{k+i}) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right\} \right] \\ &= \sum_{i=1}^{H} E\left[(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q} (\mathbf{x}_{k+i} - \mathbf{r}_{k+i}) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right] \end{aligned} \tag{4.14}$$

Since the controls have to be deterministic in practice, the joint distribution of the state-

control tuple at sample time k is then given by

$$p(\tilde{\mathbf{x}}_k) = p\begin{pmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_k \\ \mathbf{u}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_k & Cov[\mathbf{x}_k, \mathbf{u}_k] \\ Cov[\mathbf{u}_k, \mathbf{x}_k] & Var[\mathbf{u}_k] \end{bmatrix}\right) \tag{4.15}$$

where $Cov[\mathbf{x}_k, \mathbf{u}_k]$, $Cov[\mathbf{u}_k, \mathbf{x}_k]$ and $Var[\mathbf{u}_k]$ are zero. As given in Appendix D, the cost function (4.14) can be simplified to

$$\begin{aligned} &E[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})] \\ &= \sum_{i=1}^{H} \left\{ E\Big[(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q} (\mathbf{x}_{k+i} - \mathbf{r}_{k+i})\Big] + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right\} \\ &= \sum_{i=1}^{H} \left\{ (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q} (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}) + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \right\} \end{aligned} \tag{4.16}$$

This simplification essentially transformed the stochastic cost function into a deterministic one. Therefore most linear and nonlinear optimization methods can be used to solve the problem.

4.1.4 Gradient Based Optimization

Solving (4.13) is computationally demanding. The computational complexity of the one-step moment matching in (4.9) alone requires $\mathcal{O}(D^2 n^2 (n+m))$ operations. With the complexities of both hyperparameters learning and GP inferences, only problems with limited dimensions (under 12 as suggested by most publications) and limited size of training data can make use of GP based MPC. In this section, we shall describe our gradient-based method to solve this problem that is able to reduce the computational burden significantly.

Assuming $h(z) = \mathbb{E}[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})]$, the optimization problem (4.13) can be described in a condensed form as

$$z^* = \arg\min_{z \in \mathbb{Z}} h(z) \tag{4.17}$$

with initial guess $z_0 \subseteq \mathbb{R}^m$, and $h(\cdot)$ is a value-based differentiable function over the whole solution domain $\mathbb{Z} \subseteq \mathbb{R}^m$. z^* denotes an optimal solution that satisfies $\nabla_z h(z^*) = 0$ and $\nabla_z^2 h(z^*) \geq 0$. Note that optimization approaches using second-order derivatives $\nabla_z^2 h(\cdot)$, such as Newton's method that using second-order derivatives to construct a Hessian matrix, can improve the accuracy but is computational demanding. Therefore we only use the first-order derivative $\nabla_z h(\cdot)$ to keep the algorithm simple, even though both

derivatives are available when using GP models [113].

The optimal solution z^* can be obtained by iteratively conducting a linear or steepest descent search

$$z(i+1) = z(i) + \alpha_s \nabla_z h(z(i)) \tag{4.18}$$

until finding one that satisfies $h(z(i)) - h(z^*) \geq \epsilon$, where ϵ is a predefined tolerance, and α_s is search step size. A way to tune this step size can be found in [119]. Using this method, suboptimal solutions to (4.13) can still be found even if it is non-convex.

The key issue in implementing this gradient-based method on problem (4.13) is computing the gradients that are derivatives of the value function w.r.t. controls. Numerical methods such as finite difference [120] are often used to approximate the gradients. They are easy to implement but may lead to poor gradients due to the nature of approximation methods [121]. Fortunately, with the use of GP models to represent the dynamical system, the gradients can be readily obtained analytically without the need for numerical approximations.

Let

$$\mathcal{H}_i = (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q} (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}) + \text{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}) + \mathbf{u}_{k+i-1}^T \mathbf{R} \mathbf{u}_{k+i-1} \tag{4.19}$$

Then from (4.16), $\mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] = \sum_{i=1}^{H} \mathcal{H}_i$. The gradients $\frac{d\mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right]}{d\mathbf{u}_{k-1}}$ can be obtained by using the chain-rule,

$$\frac{d}{d\mathbf{u}_{k-1}} \mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] = \sum_{i=1}^{H} \frac{d\mathcal{H}_i}{d\mathbf{u}_{k+i-1}} \tag{4.20}$$

and

$$\frac{d\mathcal{H}_i}{d\mathbf{u}_{k+i-1}} = \frac{\partial \mathcal{H}_i}{\partial \boldsymbol{\mu}_{k+i}} \frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \mathbf{u}_{k+i-1}} + \frac{\partial \mathcal{H}_i}{\partial \boldsymbol{\Sigma}_{k+i}} \frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \mathbf{u}_{k+i-1}} + \frac{\partial \mathcal{H}_i}{\partial \mathbf{u}_{k+i-1}} \tag{4.21}$$

where $\frac{\partial \mathcal{H}_k}{\partial \boldsymbol{\mu}_{k+i}}, \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{\Sigma}_{k+i}}$ and $\frac{\partial \mathcal{H}_k}{\partial \mathbf{u}_{k-1}}$ can be easily obtained. Also,

$$\begin{aligned} \frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \mathbf{u}_{k+i-1}} &= \frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}} \frac{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}} \\ \frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \mathbf{u}_{k+i-1}} &= \frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}} \frac{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}} \end{aligned} \tag{4.22}$$

InitializationLearning GP Models, H, $\mathbf{r}_k$, $\mathbf{Q}, \mathbf{R}$. ***Initialization***:
Maximum iterations $N = 1000$,
$\epsilon = 1.0 \times 10^{-6}$,
initial inputs $\mathbf{u}_0$ and optimal controls $\mathbf{u}^* = \mathbf{u}_0$;
for $i = 1$ *to* N **do**
  if $\mathbb{E}[\mathcal{J}(\mathbf{u}_i)] \leq \epsilon$ **then**
    $\mathbf{u}^* = \mathbf{u}_i$;
    End Loop;
  else
    Calculate gradients $\dfrac{d\mathbb{E}\left[\mathcal{J}(\mathbf{u}_i)\right]}{d\mathbf{u}_{i-1}}$ using (4.21);
    Update step length α_s according to [119];
    Update controls $\mathbf{u}_{i+1} = \mathbf{u}_i + \alpha_s \dfrac{d\mathbb{E}\left[\mathcal{J}(\mathbf{u}_i)\right]}{d\mathbf{u}_{i-1}}$;
    $i = i + 1$;
  end
end
Output: Optimal controls $\mathbf{u}^*$.

Algorithm 5: Analytical gradient based optimization method

where $\dfrac{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}}$ and $\dfrac{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}}$ can be easily obtained as well. More details about computations of $\dfrac{\partial \boldsymbol{\mu}_{k+i}}{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}}$ and $\dfrac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}}$ can be found in Appendix C. Algorithm 5 summarizes our proposed optimization method using analytical gradients at each iteration of MPC optimization.

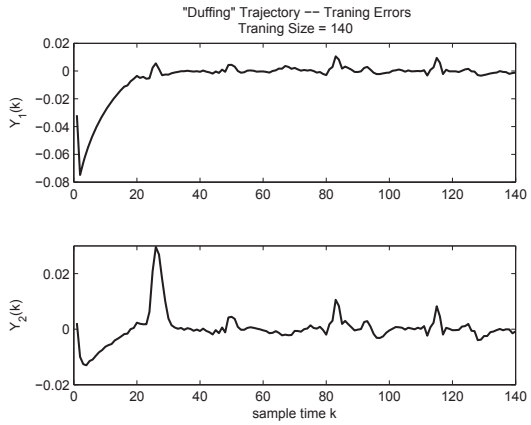
4.2 Simulation Results

The performance of proposed GP based MPC algorithm is verified by simulations on the trajectory tracking problems of MIMO LTV and NLTV systems. All simulations are independently conducted 50 times.

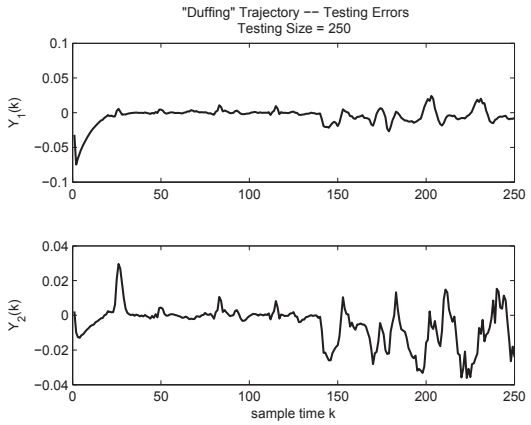
4.2.1 Numerical Simulations of LTV System

The 2-inputs and 2-outputs LTV numerical example used here is given as follows,

$$\dot{\mathbf{x}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} a(t) & 1 \\ b(t) & 0 \end{bmatrix} \mathbf{u} + \mathbf{w}_k \tag{4.23}$$



4.1.a: Learning Errors



4.1.b: Testing Errors

Figure 4.1: GP Modelling results of unknown LTV system in the "Duffing" trajectory tracking problem

where $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]^T$ and $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2]^T$ denotes states and inputs. The numerical system is corrupted by a Gaussian noises $\mathbf{w} \sim \mathcal{N}(0, 0.01)$.

"Duffing" Trajectory Tracking

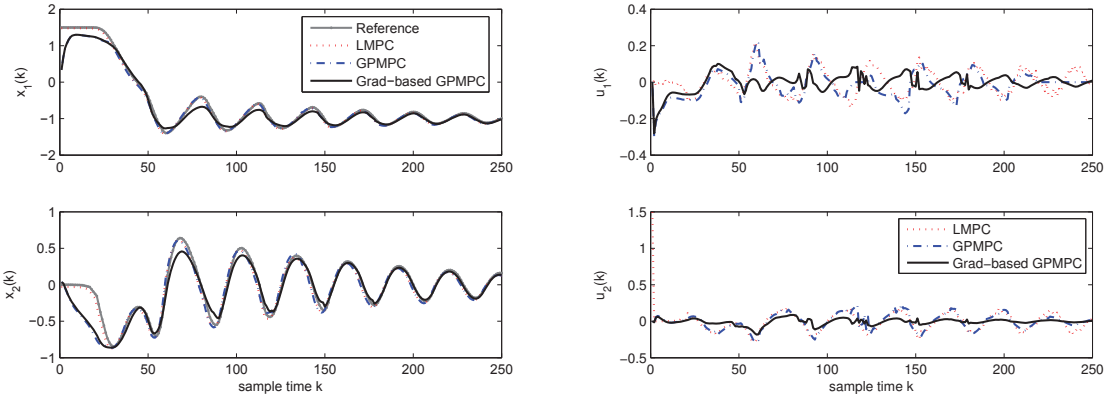
In the first simulation, the time-varying parameters are defined by,

$$a(t) = 1 + \sin\left(\frac{2\pi t}{1500}\right) \tag{4.24a}$$

$$b(t) = \cos\left(\frac{2\pi t}{1500}\right) \tag{4.24b}$$

To collect observations, the system firstly is controlled to follow the so called "Duffing" trajectory (shown as grey dotted line in Figure 4.2.a) through using a linear MPC approach proposed in [122]. Then, 250 observations including states and controls are collected. We use 140 and all of them to train and test GP models, respectively. As a result, the overall learning process takes approximated 0.5 seconds. Meanwhile, the training MSE is 2.2338×10^{-4} while the test MSE is 3.4091×10^{-4}. These results demonstrate a well modelling performance of using GP models. The learning and testing errors over the sampling time is given in Figure 4.1.a and 4.1.b, respectively.

The learned GP models are then used to predict future system responses in the tracking problem. Theoretically, a long enough prediction horizon H is required to guarantee stability and feasibility of using MPC scheme [123]. The approach to estimate required H



4.2.a: Controlled States

4.2.b: Control Inputs

Figure 4.2: Simulation results of using GP based MPC in the "Duffing" trajectory tracking problem

is discussed in [124]. However, a longer time also may be required to solve MPC optimization problems because computational burdens are increased. The consideration of this issue is especially important when using data-driven GP models due to their computational issues. The prediction and control horizon both are defined as 1 to make a trade-off in this simulation. In addition, the MPC optimization problem (4.11) is solved by the derivative-free Nelder-Mead approach which attempts to minimize a scalar-valued nonlinear function of multiple variables using only function values [125]. The controlled states and control inputs in the "Duffing" trajectory tracking problem are given in Figure 4.2.a and 4.2.b, respectively. Where "Grad-based GPMPC" denotes the proposed MPC approach using analytical gradients, while "GPMPC" is GP based MPC without using gradients. Meanwhile, the simulation result of using linear MPC (shown as "LMPC" in the figures) is used as a reference. Based on controlled states given in Figure 4.2.a, "LMPC" based on the exact numerical model produces the best controlled states that closely follow the trajectory. Over the whole trajectory, the tracking MSE is 0.0098 on x_1, and 6.9009×10^{-4} on x_2. In addition, "GPMPC" is able to perform as well as "LMPC" after first 25 sample time because of close control inputs shown in Figure 4.2.b and tracking MSE values. In particular, we obtain 7.979×10^{-4} and 6.921×10^{-4} MSE values on x_1 and x_2 when using "GPMPC", they are quite close to 3.771×10^{-4} and 4.686×10^{-4} of using "LMPC". Finally, through using "Grad-based GPMPC" approach, the controlled states overall follow the trajectory even though we obtain the bigger tracking MSE, i.e. 0.0202 on x_1 and 0.0175 on x_2, than others. Probably, this is mainly because larger average predicted variances are produced when using "Grad-based GPMPC" approach

as shown in Figure 4.5.a. Particularly, they are 0.01 on x_1 and 0.0127 on x_2 than 0.0077 and 0.0071 of using "GPMPC".

However, it takes over 70 seconds to iteratively compute 250 controls when using "GPMPC" in the first tracking problem. By using proposed "Grad-based GPMPC" approach, the required time is reduced to approximately 35 seconds. This demonstrates that optimization problem can be solved more efficiently through using "Grad-based GPMPC" approach.

4.2.2 "Lorenz" Trajectory Tracking

In the second simulation, the control task is tracking a 2D "Lorenz" trajectory, shown in Figure 4.4.a. The time-varying parameters are now defined by,

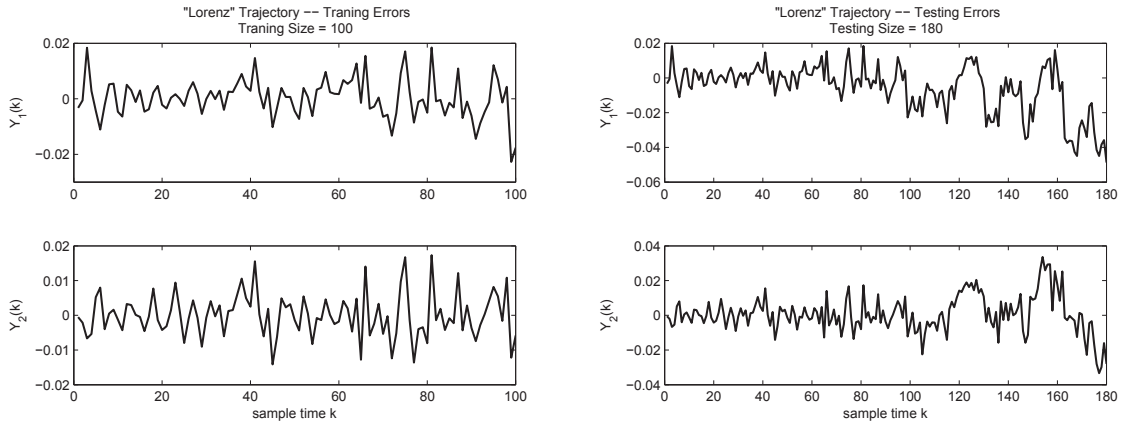
$$a(t) = \frac{1 - \exp(-t^2)}{t} \tag{4.25a}$$

$$b(t) = 1 \tag{4.25b}$$

Similarly, we use a linear MPC strategy firstly to conduct the "Lorenz" trajectory tracking task. In this way, we collect 180 state-control observations. To learn GP models, 100 and all of them are used for training and testing, respectively. The overall learning process takes approximately 0.57s. In addition, we obtain 8.7979×10^{-5} training and 4.0674×10^{-4} testing MSE values. The learning and testing errors over the sampling time in the "Lorenz" trajectory tracking problem is given in Figure 4.3.a and 4.3.b, respectively.

As well, in the MPC, we define prediction horizon as $H = 1$. And the derivative-free Nelder-Mead and proposed algorithms are used to solve the optimization problem.

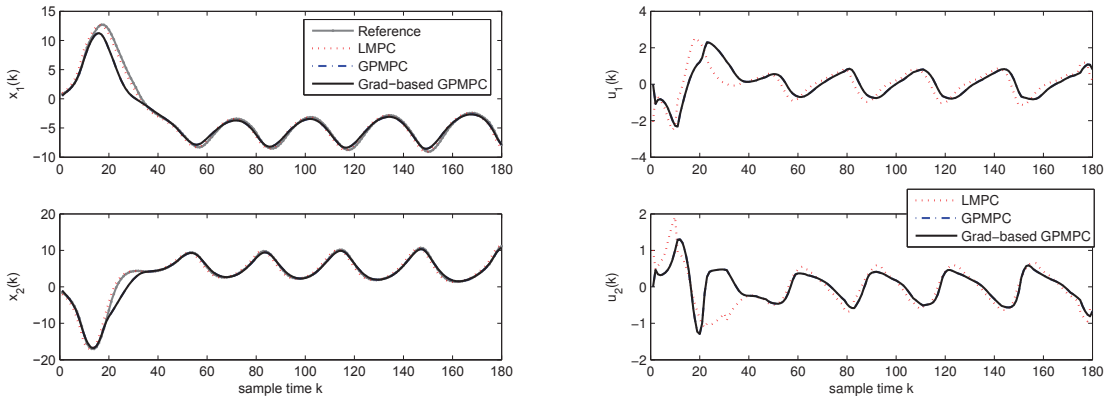
As shown in Figure 4.4.a, controlled states by using "LMPC" can closely follow the "Lorenz" trajectory when the exact dynamical model is available. In this situation, the tracking MSE is only 0.0043 on x_1 and 0.0151 on x_2. The dynamical system can follow the trajectory as well by using proposed "GPMPC" and "Grad-based GPMPC" algorithms when exact model of the system is unknown and learnt by GP models. According to simulation results, two proposed approaches are equally able to produce controlled states that are overall close to target trajectory, even though bigger tracking MSE values they produced than those of "LMPC". In particular, The tracking MSE of using "GPMPC" is 1.0013 on x_1 and 0.9778 on x_2 that are close to 1.0012 and 0.9772 when using "Grad-based



4.3.a: Learning Errors

4.3.b: Testing Errors

Figure 4.3: GP Modelling results of unknown LTV system in the "Lorenz" trajectory tracking problem

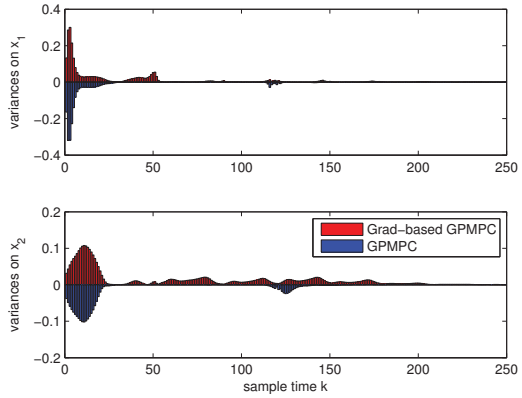


4.4.a: Controlled States

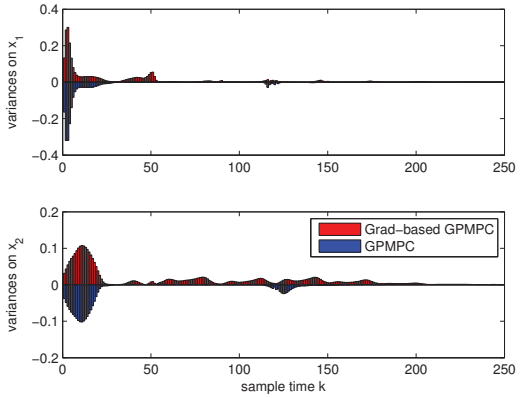
4.4.b: Control Inputs

Figure 4.4: Simulation results of using GP based MPC in the "Lorenz" trajectory tracking problem

GPMPC" approach. In addition, as shown in Figure 4.4.b, control inputs of using these two approaches are approximately equal as well. The same situation again happens when we compute average predicted variances given in Figure 4.5.b. The obtained average variance when using "GPMPC" is 1.8112 on x_1 and 0.9505 on x_2, and is 1.8118 on x_1 and 0.9504 on x_2 when using "Grad-based GPMPC" approach. Those results all demonstrate an approximately equal performance when we use proposed algorithms in the "Lorenz" trajectory tracking problem. However, to obtain approximately equal 180 controls in this simulation, "GPMPC" approach requires over 37 seconds while "Grad-based GPMPC" only takes approximately 16 seconds. This again demonstrates that



4.5.a: "Duffing" Trajectory



4.5.b: "Lorenz" Trajectory

Figure 4.5: Uncertainty propagation over the sampling time in the trajectory tracking problems of the LTV system

"Grad-based GPMPC" outperforms than "GPMPC" with respect to the computational efficiency.

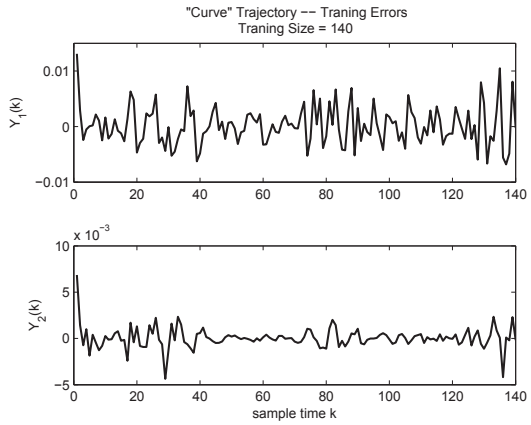
4.2.3 Numerical Simulations of NLTV System

The nonlinear system in Section 3.3 are used as the NLTV numerical example here.

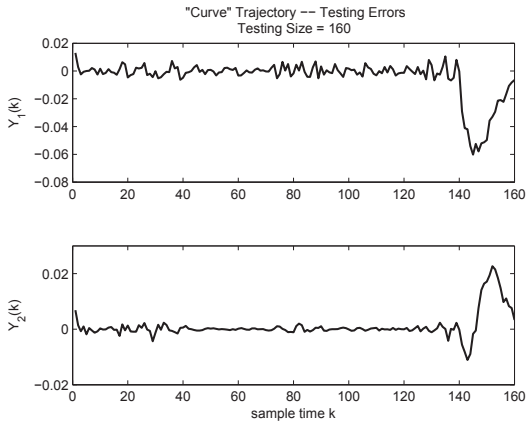
"Curve" Trajectory

In the first simulation, we use the same time-varying parameters defined in (3.18) to track the "Curve" trajectory given in (3.20) in Section 3.3.

Again, to collect the observations, we first perform the "Curve" trajectory tracking task on the deterministic system (3.17) by using the classical Nonlinear Model Predictive Control (NMPC) [126]. The simulation results also are used as a reference when we discuss the control performance. Consequently, the 160 observations including states and controls are collected in the range of $[0, 6\pi]$. We use 140 and all of them to train and test GP models. As a result, it takes approximately 1.4 seconds to learn all separate GP models. In addition, the training MSE is 4.3360×10^{-5}, while the testing MSE is 3.2930×10^{-4}. These results demonstrate a well modelling performance of using GP models. The



4.6.a: Learning Errors



4.6.b: Testing Errors

Figure 4.6: GP Modelling results of unknown NLTV system in the "Curve" trajectory tracking problem

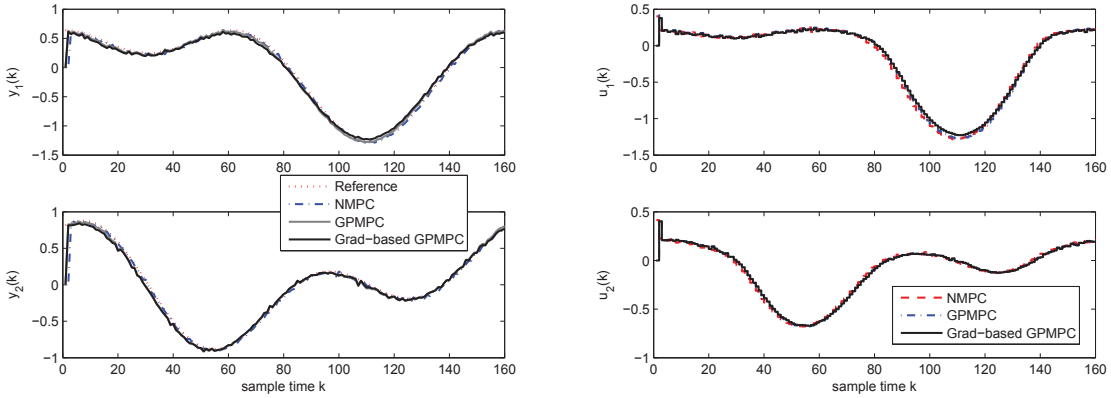
learned hyperparameters are given as follows,

$$\boldsymbol{\theta}_1 = [-0.21, 0.62, 1.20, 2.89, -0.52, 0.75, -0.33, -5.47]^T$$
$$\boldsymbol{\theta}_2 = [0.10, 5.40, 3.59, 1.33, -0.33, 2.97, 0.55, -6.60]^T$$
$$\boldsymbol{\theta}_3 = [0.96, 2.75, -0.04, 1.47, 3.49, -0.50, -0.17, -5.07]^T$$
$$\boldsymbol{\theta}_4 = [3.96, 5.21, -0.41, 3.95, 3.17, -0.36, 0.81, -6.55]^T$$

And the learning and testing errors for the unknown NLTV system over the sampling time in the "Curve" trajectory tracking problem is given in Figure 4.6.a and 4.6.b, respectively.

Then, the learnt GP models are used in the MPC control problems. Again, the prediction and control horizon both are specified as 1 to make a trade-off of the computational efficiency and complexity of the problem. In addition, the MPC optimization problem are solved by using derivative-free approach and proposed gradient-based algorithm.

The simulation results of the "Curve" trajectory tracking problem are given in Figure (4.7.a) and (4.7.b). The control inputs of using NMPC are denoted as "NMPC" in the figures. By using the exact system model, "NMPC" can produce controlled outputs that closely follow the desired trajectory. The overall tracking MSE of "NMPC" is 0.0063 on y_1 and 0.0093 on y_2. The proposed "GPMPC" and "Grad-based GPMPC" perform equally well when the unknown system is learnt by GP models. As shown in Figure (4.7.a), controlled outputs of the proposed algorithms closely follow the trajectory as well. The obtained tracking MSE values are even better. They are 0.0045 and 0.0057



4.7.a: Controlled States

4.7.b: Control Inputs

Figure 4.7: Simulation results of using GP based MPC in the "Curve" trajectory tracking problem

when using "GPMPC", and 0.0055 and 0.0060 when using "Grad-based GPMPC" on y_1 and y_2, respectively. The obtained control inputs in this simulation are given in Figure (4.7.b).

In addition, Figure (4.10.a) shows the propagated uncertainties in this simulation, where higher variance values denote less beliefs on model predictions. This also indicates the quality of learned models during the policy planning process.

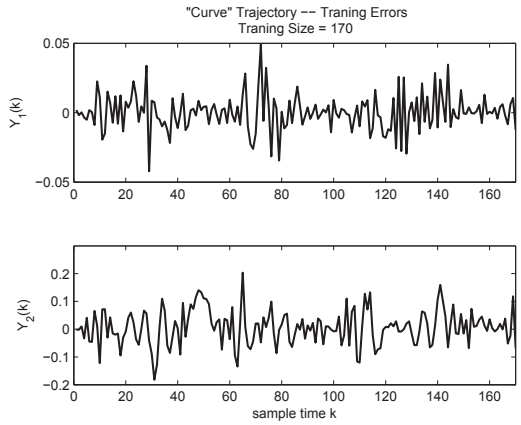
4.2.4 "Lorenz" Trajectory

The second simulation involves tracking a "Lorenz" trajectory (shown as red dotted lines in Figure (4.9.a)). The time-varying parameters are specified by,

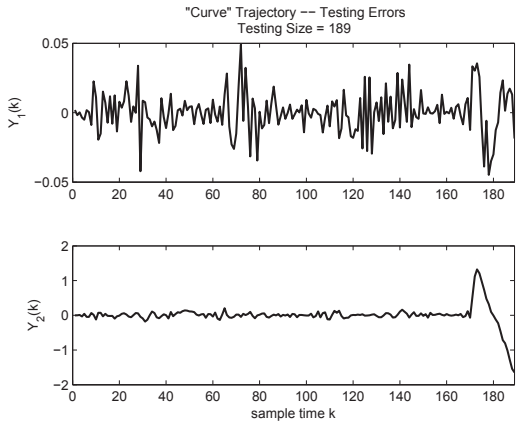
$$a(k) = 10 + 0.5\sin(k) \tag{4.26a}$$

$$b(k) = 10/(1 + \exp(-0.05k)) \tag{4.26b}$$

In the second "Lorenz" tracking problem, we collect 189 state-control observations. To learn GP models, 170 and all of them are used for training and testing, respectively. The overall learning process takes approximately 1.7 seconds. Consequently, we obtain 0.0398 training and 0.3356 testing MSE values. This shows the good modelling performance of



4.8.a: Learning Errors



4.8.b: Testing Errors

Figure 4.8: GP Modelling results of unknown NLTV system in the "Lorenz" trajectory tracking problem

using GP models again. The learned hyperparameters are given by

$$\boldsymbol{\theta}_1 = [1.37, 2.94, 5.32, 5.83, 4.76, 5.41, 1.68, -4.01]^T$$
$$\boldsymbol{\theta}_2 = [1.91, 3.69, 5.64, 4.41, 0.17, 2.23, 2.71, -2.85]^T$$
$$\boldsymbol{\theta}_3 = [3.66, 2.72, 2.39, 3.01, 1.98, 1.99, 1.61, -4.72]^T$$
$$\boldsymbol{\theta}_4 = [3.27, 3.07, 4.33, 3.80, 4.55, 2.45, 3.83, -2.06]^T$$

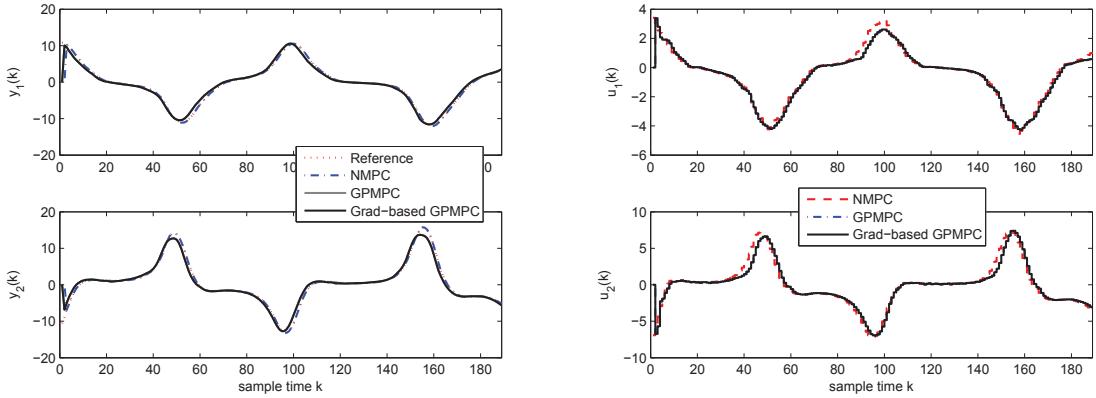
Figure 4.8.a and 4.8.b give the learning and testing errors for the unknown NLTV system over the sampling time in the "Lorenz" trajectory tracking problem, respectively.

In the "Lorenz" trajectory tracking problem, the two proposed algorithms perform good control abilities as well. As shown in Figure (4.9.a), the controlled outputs y_1 and y_2 again are closed to the desired trajectory and results of "NMPC". The corresponding tracking MSE are 1.2274 and 1.0071 when using "PMPC", and 1.0175 and 1.4039 when using "G-PMPC", compared to 1.2274 and 1.0071 of "NMPC". The control actions of using these three methods in the "Lorenz" tracking problem are given in Figure (4.9.b).

The propagated uncertainty in this simulation is given in Figure 4.10.b.

Computational Efficiency

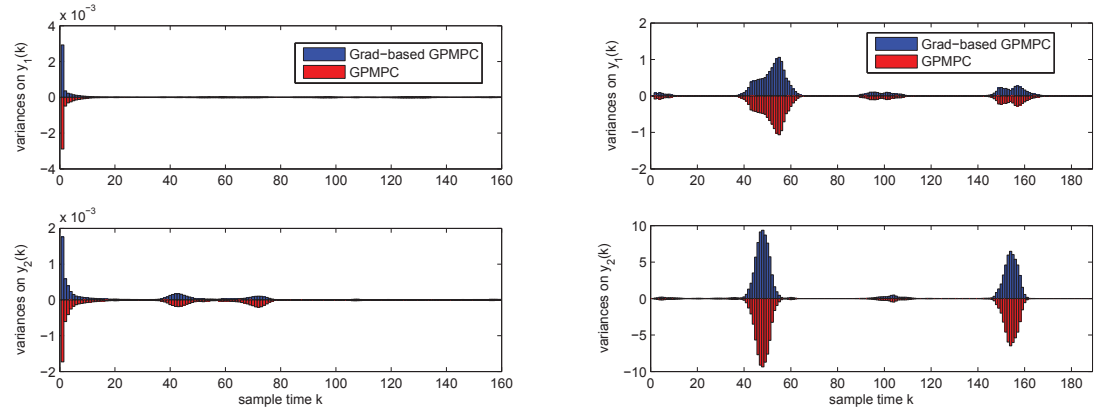
The benefit of using the proposed gradient based solution to the optimization problem is validated by the following comparison. In the "Curve" simulation. "GPMPC" requires



4.9.a: Controlled States

4.9.b: Control Inputs

Figure 4.9: Simulation results of using GP based MPC in the "Lorenz" trajectory tracking problem



4.10.a: "Curve" Trajectory

4.10.b: "Lorenz" Trajectory

Figure 4.10: Uncertainty propagation over the sampling time in the trajectory tracking problems of the NLTV system

over 70 seconds to iteratively obtain 160 control actions, while it only takes approximately 35 seconds when using "Grad-based GPMPC". Similarly, to obtain 189 control inputs in the "Lorenz" trajectory tracking problem, it takes over 130 seconds when using "GPMPC", and approximately requires only 71 seconds when using "Grad-based GPMPC". This comparison demonstrates a significant improvement on computational efficiency when using the analytical gradients.

4.3 Conclusion

The GP based MPC strategy is proposed to handle the unconstrained control problem of unknown systems. The proposed algorithm allows directly taking model uncertainties naturally obtained during GP model predictions into account when computing the MPC control inputs. This essentially reduces the computational burdens of most existing probabilistic model based Stochastic Model Predictive Control (SMPC) approaches. In addition, through using analytical gradients that are available when using GP models, the optimization problem is solved more efficiently compared with derivative-free solutions. The simulation results on the trajectory tracking problem of numerical LTV and NLTV systems demonstrate good modelling and control performances of the proposed GP based MPC, as well as the computational efficiency of proposed gradient based optimization algorithm.

Chapter 5

Constrained Model Predictive Control Using Gaussian Process Models

The proposed GP based MPC approach in Chapter 4 is only for unconstrained problems. In this chapter, two GP based MPC algorithms GPMPC1 and GPMPC2 are proposed where there are input and state constraints. The constrained GP based SMPC problem is first formulated so that the GP variances are directly included in the cost function. Then the constrained stochastic problem is relaxed to a deterministic one by specifying the confidence level. In the GPMPC1 algorithm, the resulting nonlinear MPC problem is usually non-convex, and solved by using a Sequential Quadratic Programming (SQP) based method and the basic GP based local model in this paper. Similar to commonly used methods in [21, 99], the GP variances are considered as a slack variable in the state constraints. The nonlinear MPC problem is further reformulated to a convex optimization problem in the GPMPC2 algorithm by using the extended GP based local model. The resulting MPC problem is efficiently solved by using an active-set method in this paper.

The proposed GP based local dynamical models are presented in Section 5.1, and the proposed GP based MPC algorithms GPMPC1 and GPMPC2 are presented in Section 5.2 and 5.3 respectively. The stability of the proposed algorithms are analysed in Section 5.4. The simulation results on the trajectory tracking problems of numerical nonlinear systems in Section 5.5 demonstrate the performance of the proposed algorithms.

5.1 GP Based Local Dynamical Models

When dealing with the control of nonlinear systems, it is common practice to obtain local linearized models of the system around operating points. The main purpose is to reduce the computation burden involved in the nonlinear control problem. The difference here is that the model of the system is probabilistic rather than deterministic. There are many alternate ways by which a GP model could be linearized.

In [101], a GP based local dynamical model allows standard robust control methods to be used on the partially unknown system directly. Another GP based local dynamical model is proposed in [114] to integrate GP model with dynamic programming. In these two cases, the nonlinear optimization problems considered are *unconstrained*.

In this section, we shall present two different GP based local models. They will be applied to the *constrained* nonlinear problems presented in Section 5.2 and 5.3, respectively.

5.1.1 Basic GP based Local Model

Linearization can be done based on the mean values in the GP model. In this case we replace the state vector $\mathbf{x}_k$ by its mean $\boldsymbol{\mu}_k$. Then (4.1) becomes

$$\boldsymbol{\mu}_{k+1} = \mathcal{F}(\boldsymbol{\mu}_k, \mathbf{u}_k) \tag{5.1}$$

Let $(\boldsymbol{\mu}_k^*, \mathbf{u}_k^*)$ be the operating point at which the linearized model is to be obtained. Given that $\Delta\boldsymbol{\mu}_k = \boldsymbol{\mu}_k - \boldsymbol{\mu}_k^*$ and $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^*$ are small, from 5.1, we have

$$\Delta\boldsymbol{\mu}_{k+1} = \frac{\partial\mathcal{F}}{\partial\boldsymbol{\mu}_k}\Delta\boldsymbol{\mu}_k + \frac{\partial\mathcal{F}}{\partial\mathbf{u}_k}\Delta\mathbf{u}_k \tag{5.2a}$$

$$= \frac{\partial\boldsymbol{\mu}_{k+1}}{\partial\boldsymbol{\mu}_k}\Delta\boldsymbol{\mu}_k + \frac{\partial\boldsymbol{\mu}_{k+1}}{\partial\mathbf{u}_k}\Delta\mathbf{u}_k \tag{5.2b}$$

where $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k}$ and $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k}$ are the Jacobian state and input matrices respectively. Using the chain rule, we get

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k} \tag{5.3a}$$

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k} \tag{5.3b}$$

where $\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k}$, $\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k}$, $\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k}$, $\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k}$ can be easily obtained based on (4.7). Elaborations of $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}$ and $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}$ can be found in Appendix C.

5.1.2 Extended GP based Local Model

Model uncertainties are characterized by the variances. However, the basic local model derived above only involves the mean values. The extended local model aims to take into account model uncertainties. Similar to what we have done to derive the basic model, we replace the state vector $\mathbf{x}_k$ in (4.1) by $\mathbf{s}_k = [\boldsymbol{\mu}_k, \mathbf{vec}(\sqrt{\boldsymbol{\Sigma}_k})]^T \in \mathbb{R}^{n+n^2}$ which shall be known as the "extended state". Here, $\mathbf{vec}(\cdot)$ denotes the vectorization of a matrix [1]. Hence (4.1) becomes

$$\mathbf{s}_{k+1} = \mathcal{F}'(\mathbf{s}_k, \mathbf{u}_k) \tag{5.4}$$

Linearizing at the operating point $(\mathbf{s}_k^*, \mathbf{u}_k^*)$ where $\mathbf{s}_k^* = [\boldsymbol{\mu}_k^*, \mathbf{vec}(\sqrt{\boldsymbol{\Sigma}_k^*})]^T$, we have

$$\Delta \mathbf{s}_{k+1} = \frac{\partial \mathcal{F}'}{\partial \mathbf{s}_k}\Delta \mathbf{s}_k + \frac{\partial \mathcal{F}'}{\partial \mathbf{u}_k}\Delta \mathbf{u}_k \tag{5.5}$$

Here, $\Delta \mathbf{s}_k = \mathbf{s}_k - \mathbf{s}_k^*$ and $\Delta \mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^*$. The Jacobian matrices are

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{s}_k} = \begin{bmatrix} \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k} & \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \\ \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\mu}_k} & \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \end{bmatrix} \in \mathbb{R}^{(n+n^2)\times(n+n^2)} \tag{5.6a}$$

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{u}_k} = \begin{bmatrix} \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k} \\ \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \mathbf{u}_k} \end{bmatrix} \in \mathbb{R}^{(n+n^2)\times m} \tag{5.6b}$$

[1] $\boldsymbol{\Sigma}_k$ is a real symmetric matrix therefore can be diagonalized. The square root of a diagonal matrix can simply be obtained by computing the square roots of diagonal entries.

with the entries given by

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\Sigma}_k} \frac{\partial \boldsymbol{\Sigma}_k}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \tag{5.7a}$$

$$\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\mu}_k} \tag{5.7b}$$

$$\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} = \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\Sigma}_k} \frac{\partial \boldsymbol{\Sigma}_k}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \tag{5.7c}$$

$$\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \mathbf{u}_k} = \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \mathbf{u}_k} \tag{5.7d}$$

Since $\frac{\partial \sqrt{\boldsymbol{\Sigma}_k}}{\partial \boldsymbol{\Sigma}_k} = \frac{1}{2\sqrt{\boldsymbol{\Sigma}_k}}$ and $\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} = \frac{1}{2\sqrt{\boldsymbol{\Sigma}_{k+1}}}$, they can be expressed as

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k} \tag{5.8a}$$

$$\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k} \tag{5.8b}$$

$$\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k} \tag{5.8c}$$

$$\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \mathbf{u}_k} = \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k} \tag{5.8d}$$

$\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k}$ and $\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k}$ can be easily obtained based on (4.7). Elaborations of $\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}$ and $\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}$ can be found in Appendix C.

5.2 GPMPC1 Algorithm

5.2.1 MPC Trajectory Tracking Problem Formulation

A discrete-time nonlinear dynamical system defined by (4.1) is required to track a trajectory given by $\{\mathbf{r}_k\}$ for $k = 1, 2, \cdots$. Using MPC with a prediction horizon $H \geq 1$, the

optimal control sequence can be obtained by solving the following problem:

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}, \mathbf{r}_k) \tag{5.9a}$$

$$\text{s.t. } \mathbf{x}_{k+i|k} = f(\mathbf{x}_{k+i-1|k}, \mathbf{u}_{k+i-1}) \tag{5.9b}$$

$$\mathbf{x}_{\min} \leq \mathbf{x}_{k+i|k} \leq \mathbf{x}_{\max} \tag{5.9c}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i-1} \leq \mathbf{u}_{\max} \tag{5.9d}$$

$$i = 1, \cdots, H$$

where only the first control action $\mathbf{u}_k$ of the resulting control sequence $\mathbf{u}(\cdot) = [\mathbf{u}_k, \cdots, \mathbf{u}_{k+H-1}]^T$ is applied to the system at time k. $\mathbf{x}_{\min} \leq \mathbf{x}_{\max}$ and $\mathbf{u}_{\min} \leq \mathbf{u}_{\max}$ are the upper and lower bounds of the system states and control inputs, respectively.

In the rest of this chapter, the cost function $\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}, \mathbf{r}_k)$ shall be rewritten as $\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})$ for brevity. The quadratic cost function given by

$$\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) = \sum_{i=1}^{H} \left\{ \left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 \right\} \tag{5.10}$$

will be used. Here, $\left\| \cdot \right\|_{\mathbf{Q}}$ and $\left\| \cdot \right\|_{\mathbf{R}}$ denote the two 2-norms weighted by positive definite matrices $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{R} \in \mathbb{R}^{m \times m}$ respectively. The control horizon will be assumed to be equal to the prediction horizon.

5.2.2 Problem Reformulation based on GP

We assume that the system function $f(\cdot)$ is unknown and it is replaced by a GP model. Consequently, problem (5.9) becomes a stochastic one [117],

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \tag{5.11a}$$

$$\text{s.t.} \quad p(\mathbf{x}_{k+1}|\mathbf{x}_k) \sim \mathcal{N}(\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}) \tag{5.11b}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i-1} \leq \mathbf{u}_{\max} \tag{5.11c}$$

$$p\big\{\mathbf{x}_{k+i|k} \geq \mathbf{x}_{\min}\big\} \geq \eta \tag{5.11d}$$

$$p\big\{\mathbf{x}_{k+i|k} \leq \mathbf{x}_{\max}\big\} \geq \eta \tag{5.11e}$$

where η denotes a confidence level. For $\eta = 0.95$, the chance constraints (5.11d) and (5.11e) are equivalent to

$$\boldsymbol{\mu}_{k+i} - 2\boldsymbol{\Sigma}_{k+i} \geq \mathbf{x}_{\min} \tag{5.12a}$$

$$\boldsymbol{\mu}_{k+i} + 2\boldsymbol{\Sigma}_{k+i} \leq \mathbf{x}_{\max} \tag{5.12b}$$

Using (5.10) as the cost function, we get

$$\begin{aligned} & E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \\ &= E\Big[\sum_{i=1}^{H} \Big\{ \big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2 \} \Big] \\ &= \sum_{i=1}^{H} E\Big[\big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big] \\ &= \sum_{i=1}^{H} \Big\{ E\Big[\big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2\Big] + E\Big[\big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big] \Big\} \end{aligned} \tag{5.13}$$

In practice, the controls are deterministic. Hence, $E\big[\mathbf{u}_k^2\big] = \mathbf{u}_k^2$ and (5.13) becomes

$$\begin{aligned} & E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \\ =& \sum_{i=1}^{H} \Big\{ \big\|\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2 + trace\big(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}\big) \Big\} \\ =& h\left(\boldsymbol{\mu}_k, \mathbf{u}_{k-1}\right) \end{aligned} \tag{5.14}$$

Now we have a deterministic cost function which involve the model variance $\boldsymbol{\Sigma}$ which allows model uncertainties to be explicitly included in the computation of the optimized controls.

5.2.3 Nonlinear Optimization Solution

With the cost function (5.13) and the state constraints (5.12), the original stochastic optimization problem (5.11) has been relaxed to a deterministic nonlinear optimization problem. But it is typically non-convex. This is usually solved by derivative-based approaches, such as Lagrange multipliers [127] which is based on first-order derivatives (gradient), and SQP and interior-point algorithms based on second-order derivatives (Hessians matrix) [128]. When the derivative of the cost function is unavailable or is too difficult to compute, it could be approximated iteratively by a sampling method [129,

130]. Alternatively, evolutionary algorithms, such as PSO [131] and Genetic Algorithm (GA) [132], could be used to solve the problem. This approach is able to handle general constrained optimization problems. However, there is no guarantee that the solutions obtained are near optimum. A review of nonlinear optimization techniques for the MPC problem can be found in [128].

In this section, the constrained nonlinear optimization problem (5.11) is solved by using the Feasibility-Perturbed Sequential Quadratic Programming (FP-SQP) algorithm proposed in [133], where a sequence of iterates can be generated by solving the trust-region SQP sub-problem at each iteration, and the feasibility of each iterate is retained by perturbing the resulting step. One benefit of using this algorithm is the SQP sub-problems can be guaranteed feasible by using the proper trust-regions. In addition, this algorithm allows simply using the objective function as the merit that typically consist of the constraint violations with estimated Lagrange multipliers besides the objective function. Finally, the FP-SQP algorithm is defined with proved local and global convergences.

Considering the following general form of a constrained nonlinear optimization problem:

$$\min_{\mathbf{z}} h(\mathbf{z}) \qquad \mathbf{z} \in \mathbb{R}^{n+m} \tag{5.15a}$$

$$\text{s.t.} \quad c(\mathbf{z}) = 0 \tag{5.15b}$$

$$d(\mathbf{z}) \leq 0 \tag{5.15c}$$

where $h : \mathbb{R}^{n+m} \to \mathbb{R}$ is the objective function, $c : \mathbb{R}^{n+m} \to \mathbb{R}^n$ and $d : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ represents the corresponding equality and inequality constraints, respectively. FP-SQP generates a sequence of feasible solutions $\{\mathbf{z}^j\}_{j=0,1,2,\cdots}$ by splitting the original problem into several Quadratic Programming (QP) sub-problems. In particular, a step $\Delta\mathbf{z}^j$ from current iterate $\mathbf{z}^j$ to the next one $\mathbf{z}^{j+1}$ can be obtained by solving the following QP subproblem:

$$\min_{\Delta\mathbf{z}^j} \nabla h(\mathbf{z}^j)^T \Delta\mathbf{z}^j + \frac{1}{2} \Delta\mathbf{z}^{jT} \mathbf{H}^j \Delta\mathbf{z}^j \tag{5.16a}$$

$$\text{s.t. } c(\mathbf{z}^j) + \nabla c(\mathbf{z}^j)^T \Delta\mathbf{z}^j = 0 \tag{5.16b}$$

$$d(\mathbf{z}^j) + \nabla d(\mathbf{z}^j)^T \Delta\mathbf{z}^j \leq 0 \tag{5.16c}$$

under the trust-region constraint

$$\left\|\Delta \mathbf{z}^j\right\| \leq \gamma^j \tag{5.17}$$

where $\nabla h(\cdot)$ denotes the first-order derivative of the objective function at $\mathbf{z}^j$, $\nabla c(\cdot)$ and $\nabla d(\cdot)$ are two linearised Jacobian matrices at the $\mathbf{z}^j$. The matrix $\mathbf{H}^j \in \mathbb{R}^{(n+m)\times(n+m)}$ is an exact or approximated Lagrangian Hessian matrix and γ^j represents the trust-region radius. To guarantee the feasibility of $\Delta \mathbf{z}^j$, its corresponding perturbation $\Delta \tilde{\mathbf{z}}^j$ which satisfies the following conditions need to be computed:

$$\mathbf{z}^j + \Delta \tilde{\mathbf{z}}^j \in \mathbf{\Pi} \tag{5.18a}$$

$$\frac{1}{2}\left\|\Delta \mathbf{z}\right\|_2 \leq \left\|\tilde{\Delta \mathbf{z}}\right\|_2 \leq \frac{3}{2}\left\|\Delta \mathbf{z}\right\|_2 \tag{5.18b}$$

where $\mathbf{\Pi}$ denotes the feasible points set of problem (5.15). A method to obtain such a perturbation is proposed in [134]. An acceptability value of $\Delta \mathbf{z}^j$ defined by:

$$\rho^j = \frac{h(\mathbf{z}^{j+1}) - h(\mathbf{z}^j)}{-\nabla h(\mathbf{z}^j)^T \Delta \mathbf{z}^j - \frac{1}{2}\Delta \mathbf{z}^{jT} \mathbf{H}^j \Delta \mathbf{z}^j} \tag{5.19}$$

If this value is not acceptable, then the trust-region radius γ^j will need to be adjusted. An adaptive method to adjust γ^j can be found in [135]. The complete FP-SQP algorithm is described in Algorithm 6.

5.2.4 Application to GPMPC1

Applying FP-SQP to the GPMPC1 problem (5.11), it should be noted that the constraints (5.12) are linear. Therefore it is possible to simply use $\Delta \tilde{\mathbf{z}}^j = \Delta \mathbf{z}^j$. The next iterate then can be obtained by

$$\mathbf{z}^{j+1} = \mathbf{z}^j + \Delta \mathbf{z}^j \tag{5.20}$$

Expressing the cost function (5.14) as (5.15a), define $\mathbf{z}_k = [\boldsymbol{\mu}_k^T, \mathbf{u}_{k-1}^T]^T \in \mathbb{R}^{n+m}$. Hence,

$$h(\mathbf{z_k}) = \sum_{i=1}^{H} \left\{ \mathbf{z}_{k+i}^T \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & \mathbf{R} \end{bmatrix} \mathbf{z}_{k+i} + \operatorname{trace}\{\mathbf{Q}\mathbf{\Sigma}_{k+i}\} \right\} \tag{5.21}$$

One key issue in using FP-SQP is the local linearisation at $\Delta \mathbf{z}^j$. The basic GP based

Initialization
feasible point $\mathbf{z}^0 \in \mathbf{\Pi}$,
Hessian matrix $\mathbf{H}^0$,
trust region upper bound $\gamma_{\max} > 0$,
initial trust region radius $\gamma^0 = \|\nabla h(\mathbf{z}^0)\|$
$\tau = 0$, $0 < \tau_1 < \tau_2 < 1$
for $j = 0, 1, 2, \cdots, J < \infty$ **do**
  Obtain step $\Delta\mathbf{z}^j$ by solving the problem (5.16);
  if $\nabla h(\mathbf{z}^j)^T \Delta\mathbf{z}^j + \frac{1}{2}\Delta\mathbf{z}^{jT}\mathbf{H}^j\Delta\mathbf{z}^j = 0$ **then**
    Stop;
  else
    Update ρ^j by using (5.19);
    Update $\mathbf{z}^{j+1}$, τ:

$$\mathbf{z}^{j+1} = \begin{cases} \mathbf{z}^j + \Delta\mathbf{z}^j, & \rho^j \geq \tau_1 \\ \mathbf{z}^j, & \text{otherwise} \end{cases}$$

$$\tau = \begin{cases} \dfrac{\|\Delta\mathbf{z}^j\|}{\|\nabla h(\mathbf{z}^{j+1}) - h(\mathbf{z}^j)\|}, & \rho^j \geq \tau_1 \\ \tau/4, & \text{otherwise} \end{cases}$$

    Update trust region radius:

$$\gamma^{j+1} = \begin{cases} \min\left\{\tau\|\nabla h(\mathbf{z}^{j+1})\|, \gamma_{\max}\right\}, & \rho^j \geq \tau_2 \\ \tau\|\nabla h(\mathbf{z}^{j+1})\|, & \text{otherwise} \end{cases}$$

    Update Hessian matrix $\mathbf{H}^{j+1}$ by using (5.24);
    $j = j + 1$;
  end
end

Algorithm 6: The Feasibility-Perturbed Sequential Quadratic Programming used in the GPMPC1 algorithm

local model derived in Section 5.1 shall be used to derive the QP subproblem as,

$$\min_{\Delta\mathbf{z}_k, \Delta\mathbf{\Sigma}_k} \sum_{i=1}^{H} \Big\{ \frac{\partial h}{\partial \mathbf{z}_{k+i}} \Delta\mathbf{z}_{k+i} + \frac{1}{2}\Delta\mathbf{z}_{k+i}^T \mathbf{H}_k \Delta\mathbf{z}_{k+i} \tag{5.22a}$$

$$+ \operatorname{trace}\big\{\mathbf{Q}(\mathbf{\Sigma}_{k+i} + \Delta\mathbf{\Sigma}_{k+i})\big\}\Big\} \tag{5.22b}$$

$$\text{s.t.} \quad \Delta\boldsymbol{\mu}_{k+i+1} = \mathbf{A}_{k+i}\Delta\boldsymbol{\mu}_{k+i} + \mathbf{B}_{k+i}\Delta\mathbf{u}_{k+i} \tag{5.22c}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i} + \Delta\mathbf{u}_{k+i} \leq \mathbf{u}_{\max} \tag{5.22d}$$

$$\boldsymbol{\mu}_{k+i} + \Delta\boldsymbol{\mu}_{k+i} - 2(\mathbf{\Sigma}_{k+i} + \Delta\mathbf{\Sigma}_{k+i}) \geq \mathbf{x}_{\min} \tag{5.22e}$$

$$\boldsymbol{\mu}_{k+i} + \Delta\boldsymbol{\mu}_{k+i} + 2(\mathbf{\Sigma}_{k+i} + \Delta\mathbf{\Sigma}_{k+i}) \leq \mathbf{x}_{\max} \tag{5.22f}$$

$$\|\Delta\mathbf{z}_{k+i}\| \leq \gamma \tag{5.22g}$$

Note that $\mathbf{A}_{k+i} = \dfrac{\partial \boldsymbol{\mu}_{k+i+1}}{\partial \boldsymbol{\mu}_{k+i}}$ and $\mathbf{B}_{k+i} = \dfrac{\partial \boldsymbol{\mu}_{k+i+1}}{\partial \mathbf{u}_{k+i}}$ are the two Jacobian matrices of the basic GP based local model (5.1).

The computation of the Hessian matrix $\mathbf{H}_k$ of the Lagrangian in (5.16) is another key issue when using the FP-SQP algorithm. The exact Hessian matrix is usually obtained by

$$\mathbf{H}_k = \nabla^2 h(\mathbf{z}_k) + \sum_{i=1}^{n} \alpha_i \nabla^2 c(\mathbf{z}_k) + \sum_{i=1}^{n+m} \beta_i \nabla^2 d(\mathbf{z}_k) \tag{5.23}$$

where α and β are two Lagrange multipliers applied to the equality and the inequality constraints respectively. This allows rapid local convergence but requires the second-order derivatives $\nabla^2 c(\mathbf{z}_k)$ which are generally not available. When the system is represented by a GP model, these derivatives are mathematically computable [2] but are computationally expensive to obtain. In addition, the exact Hessian matrix may be not positive definite. In our work, $\mathbf{H}_k$ is approximately updated by using a Quasi-Newton Hessian approximation based on the BFGS, which guarantees a convex quadratic problem (5.22). The update equation is given by

$$\mathbf{H}_{k+1} = \mathbf{H}_k - \frac{\mathbf{H}_k \Delta \mathbf{z}_k \Delta \mathbf{z}_k^T \mathbf{H}_k}{\Delta \mathbf{z}_k^T \mathbf{H}_k \Delta \mathbf{z}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \Delta \mathbf{z}_k} \tag{5.24}$$

where $\Delta \mathbf{z}_k = \mathbf{z}_{k+1} - \mathbf{z}_k$ and $\mathbf{y}_k = \boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k$.

5.3 GPMPC2 Algorithm

GPMPC1 introduced in the last section has two main disadvantages. First, model uncertainty was introduced through the variance term into the objective function in (5.14). But this is an indirect way to handle model uncertainties. A more direct approach is to introduce the variance into that state variable. This can be done through the use of the extended GP based local model (5.5). In this way, the variances are directly handled in the optimization process.

Another disadvantage of GPMPC1 is that the MPC optimization problem (5.9) is non-convex. Due to the recursive nature of SQP optimizations, the process could be time consuming. With GPMPC2, the non-convex problem is relaxed to a convex one,

[2] As shown in [113], the first-order derivatives are functions of $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Sigma}}$, the second-order derivatives therefore can be obtained by using the chain-rule.

making it much easier to solve. Sensitivity to initial conditions is reduced and in most cases exact solutions can be obtained [128]. This convex optimization problem can be solved offline by using Multi-Parametric Quadratic Programs (mp-QP) [136] where the explicit solutions are computed as a lookup table of nonlinear controllers. An example can be found in [99]. However, the size of the table grows exponentially with the number of states. Hence it is only suitable for problems with less than 5 states [137]. Using the extended GP based local model (5.5), the problem can be solved efficiently by an online active-set algorithm.

5.3.1 Problem Reformulation using Extended GP Local Model

Based on the extended local model (5.5) in Section 5.1, define the state variable as

$$\begin{aligned} \mathbf{Z}_{k+1} &= \left[\mathbf{s}_{k+1|k}, \cdots, \mathbf{s}_{k+H|k}\right]^T \in \mathbb{R}^{H(n+n^2)} \\ &= [\boldsymbol{\mu}_{k+1}, \sqrt{\boldsymbol{\Sigma}_{k+1}}, \cdots, \boldsymbol{\mu}_{k+H}, \sqrt{\boldsymbol{\Sigma}_{k+H}}]^T \end{aligned} \tag{5.25}$$

Also, let

$$\mathbf{U}_k = [\mathbf{u}_k, \cdots, \mathbf{u}_{k+H-1}]^T \in \mathbb{R}^{Hm} \tag{5.26}$$

$$\mathbf{r}^*_{k+1} = [\mathbf{r}_{k+1}, \mathbf{0}, \cdots, \mathbf{r}_{k+H}, \mathbf{0}]^T \in \mathbb{R}^{H(n+n^2)} \tag{5.27}$$

Problem (5.11) then becomes

$$\min_{\mathbf{U}} \left\{ \left\| \mathbf{Z}_{k+1} - \mathbf{r}^*_{k+1} \right\|^2_{\tilde{\mathbf{Q}}} + \left\| \mathbf{U}_{k+1} \right\|^2_{\tilde{\mathbf{R}}} \right\} \tag{5.28a}$$

$$\text{s.t.} \quad \mathbf{I}_{Hn}\mathbf{x}_{\min} \leq \mathbf{M}_z \mathbf{Z}_{k+1} \leq \mathbf{I}_{Hn}\mathbf{x}_{\max} \tag{5.28b}$$

$$\mathbf{I}_{Hm}\mathbf{u}_{\min} \leq \mathbf{U}_{k+1} \leq \mathbf{I}_{Hm}\mathbf{u}_{\max} \tag{5.28c}$$

where $\tilde{\mathbf{Q}} = \text{diag}\{[\mathbf{Q}, \text{diag}\{\mathbf{vec}(\mathbf{Q})\}, \cdots, \mathbf{Q}, \text{diag}\{\mathbf{vec}(\mathbf{Q})\}]\} \in \mathbb{R}^{H(n+n^2)\times H(n+n^2)}$, $\tilde{\mathbf{R}} = \text{diag}\{[\mathbf{R}, \cdots, \mathbf{R}]\} \in \mathbb{R}^{Hm\times Hm}$, $\mathbf{I}_a \in \mathbb{R}^a$ is the identity vector, and

$$\mathbf{M}_z = \begin{bmatrix} \mathbf{I}_n^T & 2\mathbf{I}_{n^2}^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_n^T & 2\mathbf{I}_{n^2}^T & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_n^T & 2\mathbf{I}_{n^2}^T \end{bmatrix} \in \mathbb{R}^{H\times H(n+n^2)} \tag{5.29}$$

Let $\mathbf{T}_u \in \mathbb{R}^{Hm\times Hm}$ be a lower triangular matrices with unit entries. Then,

$$\mathbf{U}_k = \mathbf{I}_{Hm}\mathbf{u}_{k-1} + \mathbf{T}_u \Delta \mathbf{U}_k \tag{5.30}$$

$\Delta \mathbf{Z}_{k+1}$ can be expressed as

$$\Delta \mathbf{Z}_{k+1} = \tilde{\mathbf{A}} \Delta \mathbf{s}_k + \tilde{\mathbf{B}} \Delta \mathbf{U}_k \tag{5.31}$$

based on the extended local model, with the state and control matrices given by

$$\tilde{\mathbf{A}} = \left[\mathbf{A}, \mathbf{A}^2, \cdots, \mathbf{A}^H \right]^T \in \mathbb{R}^{H(n+n^2)} \tag{5.32a}$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{A}^{H-1}\mathbf{B} & \mathbf{A}^{H-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix} \in \mathbb{R}^{H(n+n^2) \times Hm} \tag{5.32b}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the two Jacobian matrices (5.6) and (5.7) respectively. The corresponding state variables $\mathbf{Z}_{k+1}$ is therefore obtained by,

$$\mathbf{Z}_{k+1} = \mathbf{s}_k + \mathbf{T}_z \left(\tilde{\mathbf{A}} \Delta \mathbf{s}_k + \tilde{\mathbf{B}} \Delta \mathbf{U}_k \right) \tag{5.33}$$

where $\mathbf{T}_z \in \mathbb{R}^{H(n+n^2) \times H(n+n^2)}$ denotes a lower triangular matrix with unit entries.

Then, based on the (5.30) and (5.33), the problem (5.28) can be expressed in a more compact form as

$$\min_{\Delta \mathbf{U}} \frac{1}{2} \left\| \Delta \mathbf{U}_k \right\|_{\mathbf{\Phi}}^2 + \boldsymbol{\psi}^T \Delta \mathbf{U}_k + \mathbf{C} \tag{5.34a}$$

$$\text{s.t. } \Delta \mathbf{U}_{\min} \leq \begin{bmatrix} \mathbf{T}_u \\ \mathbf{T}_z \tilde{\mathbf{B}} \end{bmatrix} \Delta \mathbf{U}_k \leq \Delta \mathbf{U}_{\max} \tag{5.34b}$$

where

$$\mathbf{\Phi} = \tilde{\mathbf{B}}^T \mathbf{T}_z^T \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{B}} + \mathbf{T}_u^T \tilde{\mathbf{R}} \mathbf{T}_u \in \mathbb{R}^{Hm \times Hm} \tag{5.35a}$$

$$\boldsymbol{\psi} = 2(\mathbf{s}_k \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{B}} + \Delta \mathbf{s}_k \tilde{\mathbf{A}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{B}} - \mathbf{r}_{k+1}^* \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{B}} + \mathbf{u}_{k-1} \tilde{\mathbf{R}} \mathbf{T}_u) \in \mathbb{R}^{Hm} \tag{5.35b}$$

$$\mathbf{C} = (\mathbf{s}_k^2 + \mathbf{r}_{k+1}^*) \tilde{\mathbf{Q}} + 2\mathbf{s}_k \Delta \mathbf{s}_k \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{A}} + \Delta \mathbf{s}_k^2 \tilde{\mathbf{A}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{A}} \tag{5.35c}$$

$$- 2\mathbf{r}_{k+1}^* (\mathbf{s}_k \tilde{\mathbf{Q}} - \Delta \mathbf{s}_k \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{A}}) + \mathbf{u}_{k-1}^2 \tilde{\mathbf{R}} \tag{5.35d}$$

$$\Delta \mathbf{U}_{\min} = \begin{bmatrix} \mathbf{I}_{Hm} (\mathbf{u}_{\min} - \mathbf{u}_{k-1}) \\ \mathbf{I}_{H(n+n^2)} (\mathbf{x}_{\min} - \mathbf{s}_k - \mathbf{T}_z \tilde{\mathbf{A}} \Delta \mathbf{s}_k) \end{bmatrix} \tag{5.35e}$$

$$\Delta \mathbf{U}_{\max} = \begin{bmatrix} \mathbf{I}_{Hm} (\mathbf{u}_{\max} - \mathbf{u}_{k-1}) \\ \mathbf{I}_{H(n+n^2)} (\mathbf{x}_{\max} - \mathbf{s}_k - \mathbf{T}_z \tilde{\mathbf{A}} \Delta \mathbf{s}_k) \end{bmatrix} \tag{5.35f}$$

Since $\tilde{\mathbf{Q}}, \tilde{\mathbf{R}}, \mathbf{T}_z$ and $\mathbf{T}_u$ are positive definite, $\mathbf{\Phi}$ is also positive definite. Hence (5.34) is a constrained QP problem and is strictly convex. The solution will therefore be unique

Initialization
the feasible point $\Delta\mathbf{U}_k^0 \in \mathbf{\Pi}_{\Delta\mathbf{U}}$;
the working set $\mathcal{W}^0 = \mathcal{A}(\Delta\mathbf{U}_k^0)$;
for $j = 0, 1, 2, \cdots$ **do**
  Compute the $\boldsymbol{\delta}^j$ and $\boldsymbol{\lambda}_k^*$ by solving the linear equations (5.42);
  if $\boldsymbol{\delta}^j = 0$ **then**
    $\lambda_k^* = \min_{i \in \mathcal{W}_k^j \cap \mathcal{I}} \boldsymbol{\lambda}_{k,i}^*$,
    $p = \operatorname{argmin}_{i \in \mathcal{W}_k^j \cap \mathcal{I}} \boldsymbol{\lambda}_{k,i}^*$
    if $\lambda_k^* \geq 0$ **then**
      $\Delta\mathbf{U}_k^* = \Delta\mathbf{U}_k^j$;
      Stop.
    else
      $\mathcal{W}_k^{j+1} = \mathcal{W}_k^j \setminus p$;
      $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j$;
    end
  else
    Compute the step length κ^j by (5.46),
    $q = \operatorname{argmin}_{i \in \mathcal{B}(\Delta\mathbf{U}_k^j)} \dfrac{\tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^j}{\tilde{\mathbf{G}}_i \boldsymbol{\delta}^j}$
    if $\kappa^j < 1$ **then**
      $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j + \kappa^j \boldsymbol{\delta}^j$;
      $\mathcal{A}(\Delta\mathbf{U}_k^{j+1}) = \mathcal{A}(\Delta\mathbf{U}_k^j) \cup q$;
    else
      $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j$;
      $\mathcal{A}(\Delta\mathbf{U}_k^{j+1}) = \mathcal{A}(\Delta\mathbf{U}_k^j)$;
    end
  end
end

Algorithm 7: Active set method for solving the GPMPC2 problem

and satisfies the Karush-Kahn-Tucker (KKT) conditions (see Appendix E).

5.3.2 Quadratic Programming Solution

The optimization problem (5.34) can be solved by an active-set method proposed in [138]. It iteratively seeks an active (or working) set of constraints and solve an equality constrained QP problem until the optimal solution is found. The advantage of this method

is that accurate solutions can still be obtained even when they may be ill-conditioning or degenerated. In addition, it is conceptually simple and easy to implement. We also use a warm-start technique to accelerate the optimization process substantially.

Let $\mathbf{G} = [\mathbf{T}_u, \mathbf{T}_z\tilde{\mathbf{B}}]^T$, the constraint (5.34b) can be written as

$$\begin{bmatrix} \mathbf{G} \\ -\mathbf{G} \end{bmatrix} \Delta\mathbf{U} \leq \begin{bmatrix} \Delta\mathbf{U}_{\max} \\ -\Delta\mathbf{U}_{\min} \end{bmatrix} \tag{5.36}$$

Ignoring the constant term $\mathbf{C}$, problem (5.34) becomes

$$\min_{\Delta\mathbf{U}} \frac{1}{2} \|\Delta\mathbf{U}_k\|^2_{\boldsymbol{\Phi}} + \boldsymbol{\psi}^T \Delta\mathbf{U}_k \tag{5.37a}$$

$$\text{s.t.} \quad \tilde{\mathbf{G}}\Delta\mathbf{U}_k \leq \tilde{\Delta}_{\mathbf{U}} \tag{5.37b}$$

where $\tilde{\mathbf{G}} = [\mathbf{G}, -\mathbf{G}]^T \in \mathbb{R}^{2H(m+n+n^2)\times Hm}$ and $\tilde{\Delta}_{\mathbf{U}} = [\Delta\mathbf{U}_{\max}, -\Delta\mathbf{U}_{\min}]^T \in \mathbb{R}^{2H(m+n+n^2)}$.

Let $\boldsymbol{\Pi}_{\Delta\mathbf{U}}$ be the set of feasible points, and $\mathcal{I} = \{1, \cdots, 2H(m+n+n^2)\}$ be the constraint index set. For a feasible point $\Delta\mathbf{U}_k^* \in \boldsymbol{\Pi}_{\Delta\mathbf{U}}$, the index set for the active set of constraints is defined as

$$\mathcal{A}(\Delta\mathbf{U}_k^*) = \{i \subseteq \mathcal{I} | \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^* = \tilde{\Delta}_{\mathbf{U},i}\} \tag{5.38}$$

where $\tilde{\mathbf{G}}_i$ is the i^{th} row of $\tilde{\mathbf{G}}$ and $\tilde{\Delta}_{\mathbf{U},i}$ is the i^{th} row of the $\tilde{\Delta}_{\mathbf{U}}$. The inactive set is therefore given by

$$\begin{aligned} \mathcal{B}(\Delta\mathbf{U}_k^*) &= \mathcal{I} \setminus \mathcal{A}(\Delta\mathbf{U}_k^*) \\ &= \{i \subseteq \mathcal{I} | \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^* < \tilde{\Delta}_{\mathbf{U},i}\} \end{aligned} \tag{5.39}$$

Given any iteration j, the working set $\mathcal{W}_k^j$ contains all the equality constraints plus the inequality constraints in the active set. The following QP problem subject to the equality constraints w.r.t. $\mathcal{W}_k^j$ is considered given the feasible points $\Delta\mathbf{U}_k^j \in \boldsymbol{\Pi}_{\Delta\mathbf{U}}$:

$$\min_{\boldsymbol{\delta}^j} \frac{1}{2} \left\|\Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j\right\|^2_{\boldsymbol{\Phi}} + \boldsymbol{\psi}^T(\Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j) \tag{5.40a}$$

$$= \min_{\boldsymbol{\delta}^j} \frac{1}{2} \left\|\boldsymbol{\delta}^j\right\|^2_{\boldsymbol{\Phi}} + (\boldsymbol{\psi} + \boldsymbol{\Phi}\Delta\mathbf{U}_k^j)^T\boldsymbol{\delta}^j \tag{5.40b}$$

$$+ \underbrace{\frac{1}{2} \left\|\Delta\mathbf{U}_k^j\right\|^2_{\boldsymbol{\Phi}} + \boldsymbol{\psi}^T\Delta\mathbf{U}_k^j}_{\text{constant}}$$

$$\text{s.t.} \quad \tilde{\mathbf{G}}_i(\Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j) = \tilde{\Delta}_{\mathbf{U},i}, i \in \mathcal{W}_k^j \tag{5.40c}$$

This problem can be simplified by ignoring the constant term to:

$$\min_{\boldsymbol{\delta}^j} \frac{1}{2} \left\| \boldsymbol{\delta}^j \right\|_{\boldsymbol{\Phi}}^2 + (\boldsymbol{\psi} + \boldsymbol{\Phi}\Delta\mathbf{U}_k^j)^T \boldsymbol{\delta}^j \tag{5.41a}$$

$$= \min_{\boldsymbol{\delta}^j} \frac{1}{2} {\boldsymbol{\delta}^j}^T \boldsymbol{\Phi} \boldsymbol{\delta}^j + (\boldsymbol{\psi} + \boldsymbol{\Phi}\Delta\mathbf{U}_k^j)^T \boldsymbol{\delta}^j \tag{5.41b}$$

$$\text{s.t.} \quad \tilde{\mathbf{G}}_i \boldsymbol{\delta}^j = \tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^j, i \in \mathcal{W}_k^j \tag{5.41c}$$

By applying the KKT conditions to problem (5.41), we can obtain the following linear equations:

$$\underbrace{\begin{bmatrix} \boldsymbol{\Phi} & \tilde{\mathbf{G}}_{\mathcal{A}}^T \\ \tilde{\mathbf{G}}_{\mathcal{A}} & \mathbf{0} \end{bmatrix}}_{\text{Lagrangian Matrix}} \begin{bmatrix} \boldsymbol{\delta}^j \\ \boldsymbol{\lambda}_k^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\psi} - \boldsymbol{\Phi}\Delta\mathbf{U}_k^j \\ \tilde{\Delta}_{\mathbf{U},\mathcal{A}} - \tilde{\mathbf{G}}_{\mathcal{A}}\Delta\mathbf{U}_k^j \end{bmatrix} \tag{5.42}$$

where $\boldsymbol{\lambda}_k^* \in \mathbb{R}^{2H(m+n+n^2)}$ denotes the vector of Lagrangian multipliers, $\tilde{\mathbf{G}}_{\mathcal{A}} \subseteq \tilde{\mathbf{G}}$ and $\tilde{\Delta}_{\mathbf{U},\mathcal{A}} \subset \tilde{\Delta}_{\mathbf{U}}$ are the weighting matrix and the upper bounds of the constraints w.r.t. $\mathcal{W}_k^j$. Let the inverse of Lagrangian matrix be denoted by

$$\begin{bmatrix} \boldsymbol{\Phi} & \tilde{\mathbf{G}}_{\mathcal{A}}^T \\ \tilde{\mathbf{G}}_{\mathcal{A}} & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{L}_2^T \\ \mathbf{L}_2 & \mathbf{L}_3 \end{bmatrix} \tag{5.43}$$

If this inverse exists, then the solution is given by

$$\boldsymbol{\delta}^j = -\mathbf{L}_1(\boldsymbol{\psi} + \boldsymbol{\Phi}\Delta\mathbf{U}_k^j) + \mathbf{L}_2^T(\tilde{\Delta}_{\mathbf{U},\mathcal{A}} - \tilde{\mathbf{G}}_{\mathcal{A}}\Delta\mathbf{U}_k^j) \tag{5.44a}$$

$$\boldsymbol{\lambda}_k^* = -\mathbf{L}_2(\boldsymbol{\psi} + \boldsymbol{\Phi}\Delta\mathbf{U}_k^j) + \mathbf{L}_3(\tilde{\Delta}_{\mathbf{U},\mathcal{A}} - \tilde{\mathbf{G}}_{\mathcal{A}}\Delta\mathbf{U}_k^j) \tag{5.44b}$$

where

$$\mathbf{L}_1 = \boldsymbol{\Phi}^{-1} - \boldsymbol{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T(\tilde{\mathbf{G}}_{\mathcal{A}}\boldsymbol{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T)^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}\boldsymbol{\Phi}^{-1} \tag{5.45a}$$

$$\mathbf{L}_2 = \boldsymbol{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T(\tilde{\mathbf{G}}_{\mathcal{A}}\boldsymbol{\Phi}^{-1} \tag{5.45b}$$

$$\mathbf{L}_3 = -(\tilde{\mathbf{G}}_{\mathcal{A}}\boldsymbol{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T)^{-1} \tag{5.45c}$$

If $\boldsymbol{\delta}^j \neq 0$, then the set of feasible points $\Delta\mathbf{U}_k^j$ fails to minimize problem (5.37). In this case, the next set of feasible point is computed for the next iteration by $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j + \kappa^j\boldsymbol{\delta}^j$ with step size

$$\kappa^j = \min\left\{1, \min_{i \in \mathcal{B}(\Delta\mathbf{U}_k^j)} \frac{\tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i\Delta\mathbf{U}_k^j}{\tilde{\mathbf{G}}_i\boldsymbol{\delta}^j}\right\} \tag{5.46}$$

If $\kappa^j < 1$, the inequality constraint with index $q = \underset{i \in \mathcal{B}(\Delta \mathbf{U}_k^j)}{\operatorname{argmin}} \dfrac{\tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i \Delta \mathbf{U}_k^j}{\tilde{\mathbf{G}}_i \boldsymbol{\delta}^j}$ should be "activated", giving the working set $\mathcal{W}_k^{j+1} = \mathcal{W}_k^j \cup q$. Otherwise, we have $\mathcal{W}_k^{j+1} = \mathcal{W}_k^j$.

Alternatively, if the solution gives $\boldsymbol{\delta}^j = 0$, then the current feasible points $\Delta \mathbf{U}_k^j$ could be the optimal solution. This can be verified by checking the Lagrangian multiplier $\lambda_k^* = \min_{i \in \mathcal{W}_k^j \cap \mathcal{I}} \boldsymbol{\lambda}_{k,i}^*$. If $\lambda_k^* \geq 0$, the optimal solution of the (5.37) at sampling time k is found. Otherwise, this inequality constraint indexed by $p = \underset{i \in \mathcal{W}_k^j \cap \mathcal{I}}{\operatorname{argmin}} \boldsymbol{\lambda}_{k,i}^*$ should be removed from the current working set, giving us $\mathcal{W}_k^{j+1} = \mathcal{W}_k^j \setminus p$. Algorithm 7 summarizes the active set algorithm used in the GPMPC2.

Implementation Issues

The key to solving the linear equations (5.42) is the inverse of the Lagrangian matrix. However, $\tilde{\mathbf{G}}_{\mathcal{A}}$ is not always full ranked. Thus the Lagrangian matrix is not always invertible. This problem can be solved by decomposing $\tilde{\mathbf{G}}_{\mathcal{A}}$ using QR factorization technique, giving us $\mathbf{G}_{\mathcal{A}}^T = \mathcal{Q} \begin{bmatrix} \mathcal{R} & \mathbf{0} \end{bmatrix}^T$ where $\mathcal{R} \in \mathbb{R}^{m_1 \times m_1}$ is an upper triangular matrix with $m_1 = \operatorname{rank}(\tilde{\mathbf{G}}_{\mathcal{A}})$. $\mathcal{Q} \in \mathbb{R}^{Hm \times Hm}$ is an orthogonal matrix that can be further decomposed to $\mathcal{Q} = [\mathcal{Q}_1 \ \mathcal{Q}_2]$ where $\mathcal{Q}_1 \in \mathbb{R}^{Hm \times m_1}$ and $\mathcal{Q}_2 \in \mathbb{R}^{Hm \times (Hm - m_1)}$. Thus, $\mathbf{G}_{\mathcal{A}}^T = \mathcal{Q} = \mathcal{Q}_1 \mathcal{R}$ and

$$\mathbf{L}_1 = \mathcal{Q}_2 (\mathcal{Q}_2^T \boldsymbol{\Phi} \mathcal{Q}_2)^{-1} \mathcal{Q}_2^T \tag{5.47a}$$

$$\mathbf{L}_2 = \mathcal{Q}_1 {\mathcal{R}^{-1}}^T - \mathbf{L}_1 \boldsymbol{\Phi} \mathcal{Q}_1 {\mathcal{R}^{-1}}^T \tag{5.47b}$$

$$\mathbf{L}_3 = \mathcal{R}^{-1} \mathcal{Q}_1^T \boldsymbol{\Phi} \mathbf{L}_2 \tag{5.47c}$$

The second issue relates to using the appropriate warm-start technique to improve the convergence rate of the active-set method. For GPMPC2, since the changes in the state between two successive sampling instants are usually quite small, we simply use the previous $\Delta \mathbf{U}_k^*$ as the starting point $\Delta \mathbf{U}_{k+1}^0$ for the next sampling time $k+1$. This warm-start technique is usually employed in MPC optimizations because of its proven effectiveness [137].

5.4 Stability Analysis

The stability of the closed-loop controller is not guaranteed because the MPC problem is open-loop. This can be demonstrated by the stability analysis of the proposed algorithms.

In particular, for the MPC problem (5.9) in the GPMPC1 algorithm, the objective (5.10) can be directly used as the Lyapunov function. Therefore, it can be known that

$$\mathcal{V}^*(k) = \sum_{i=1}^{H} \left\{ \left\| \Delta\boldsymbol{\mu}^*_{k+i} \right\|^2_{\mathbf{Q}} + \left\| \mathbf{u}^*_{k+i-1} \right\|^2_{\mathbf{R}} + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}^*_{k+i}) \right\} \tag{5.48}$$

where $\Delta\boldsymbol{\mu}^*_{k+i} = \boldsymbol{\mu}^*_{k+i} - \mathbf{r}_{k+i}$, $\mathbf{u}^*$ is the optimal control inputs, and $\boldsymbol{\mu}^*_{k+i}$ and $\boldsymbol{\Sigma}^*_{k+i}$ represent the corresponding optimal means and variances of the GP model at time k. The Lyapunov function at time $k+1$ is subsequently obtained by,

$$\begin{aligned}
&\mathcal{V}(k+1) && \text{(5.49a)}\\
&= \sum_{i=1}^{H} \left\{ \| \Delta\boldsymbol{\mu}_{k+1+i} \|^2_{\mathbf{Q}} + \| \mathbf{u}_{k+i} \|^2_{\mathbf{R}} + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1+i}) \right\} && \text{(5.49b)}\\
&= \mathcal{V}^*(k) - \left\| \Delta\boldsymbol{\mu}^*_{k+1} \right\|^2_{\mathbf{Q}} - \| \mathbf{u}^*_k \|^2_{\mathbf{R}} - \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}^*_{k+1}) && \text{(5.49c)}\\
&+ \| \Delta\boldsymbol{\mu}_{k+1+H} \|^2_{\mathbf{Q}} + \| \mathbf{u}_{k+H} \|^2_{\mathbf{R}} + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1+H})
\end{aligned}$$

It is easy to know that $\mathcal{V}^*(k+1) \leq \mathcal{V}(k+1)$ due to the nature of the optimization. Furthermore, the following inequality can be obtained,

$$\begin{aligned}
\mathcal{V}^*(k+1) \leq & \mathcal{V}(k+1) && \text{(5.50a)}\\
\leq & \mathcal{V}^*(k) + \| \Delta\boldsymbol{\mu}_{k+1+H} \|^2_{\mathbf{Q}} && \text{(5.50b)}\\
& + \| \mathbf{u}_{k+H} \|^2_{\mathbf{R}} + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1+H})
\end{aligned}$$

because of $\left\| \Delta\boldsymbol{\mu}^*_{k+1} \right\|^2_{\mathbf{Q}} \geq 0$, $\| \mathbf{u}^*_k \|^2_{\mathbf{R}} \geq 0$ and $\mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}^*_{k+1}) \geq 0$. The stability result of the problem (5.28) in the GPMPC2 algorithm can be obtained in the same way.

The result in (5.50) shows that, to guarantee the stability, additional terminal constraints on the means and variances of the GP model, as well as the control inputs are

required such that,

$$\boldsymbol{\mu}_{k+H+1|k} - \mathbf{r}_{k+H+1} = 0 \tag{5.51a}$$

$$\boldsymbol{\Sigma}_{k+H+1|k} = 0 \tag{5.51b}$$

$$\mathbf{u}_{k+H|k} = 0 \tag{5.51c}$$

However, it should be noted that, these newly added constraints altered the optimization problem. Hence its feasibility will need to be analysed. Another approach to provide the guaranteed stability is by introducing a terminal cost into the objective function [1]. This issue will be revisited in Chapter 7.

5.5 Simulation Results

The proposed algorithms are applied to three trajectory tracking problems of a MIMO nonlinear system with time-varying parameters. For each problem, the simulation is independently repeated 50 times. The average simulation results of the 50 trials are shown here.

The modelling performance of proposed algorithm is evaluated by computing the consumed time and training MSE defined as follows,

$$\text{MSE} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{y}_k - \mathbf{r}_k)^2$$

where N denotes the terminal time index of the tracking problem and is specified as 189 in this section. In addition, the control performance is demonstrated in terms of computed inputs and outputs, as well as the Integral Absolute Error (IAE) value that describes the set-point tracking errors over the time. It is defined as follow,

$$\text{IAE} = \int_0^N |\mathbf{y}_k - \mathbf{r}_k|$$

Finally, to demonstrate the efficiency of using the GPMPC2 algorithm, the runtime of solving the MPC optimization problems of using the proposed algorithms is compared.

5.5.1 Nonlinear Numerical Example

The MIMO nonlinear numerical system adopted from [139] is given as follows,

$$\begin{aligned}
x_1(k+1) &= \frac{x_1(k)^2}{1+x_1(k)^2} + 0.3x_2(k) \\
x_2(k+1) &= \frac{x_1(k)^2}{1+x_2(k)^2+x_3(k)^2+x_4(k)^2} \\
&\quad + a(k)u_1(k) \\
x_3(k+1) &= \frac{x_3(k)^2}{1+x_3(k)^2} + 0.2x_4(k) \\
x_4(k+1) &= \frac{x_3(k)^2}{1+x_1(k)^2+x_2(k)^2+x_4(k)^2} \\
&\quad + b(k)u_2(k) \\
y_1(k+1) &= x_1(k+1) + \omega_1 \\
y_2(k+1) &= x_3(k+1) + \omega_2
\end{aligned} \tag{5.52}$$

where x_1, x_2, x_3 and x_4 are system states, u_1, u_2 and y_1, y_2 denote system inputs and outputs, respectively. $\omega_1, \omega_2 \sim \mathcal{N}(0, 0.01)$ denote two Gaussian white noises. In addition, the time-varying parameters $a(k)$ and $b(k)$ are specified by,

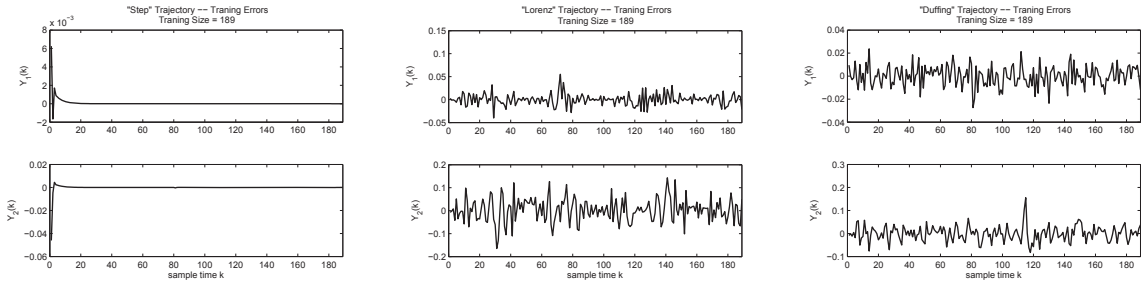
$$a(k) = 10 + 0.5\sin(k) \tag{5.53a}$$

$$b(k) = \frac{10}{1+\exp(-0.05k)} \tag{5.53b}$$

The nonlinear numerical system is steer to follow the so called "Step", "Lorenz" and "Duffing" trajectories shown as red dots in Figure 5.2.a, 5.3.a and 5.4.a respectively. The system inputs are subjected to $0.25 \leq u_1(k) \leq 1.75, 0.5 \leq u_2(k) \leq 2.5$ for the "Step" trajectory, $-4 \leq u_1(k) \leq 4, -7 \leq u_2(k) \leq 7$ for the "Lorenz" trajectory and $-0.5 \leq u_1(k) \leq 0.25, -0.5 \leq u_2(k) \leq 0.2$ for the "Duffing" trajectory.

To generate the observations, each trajectory tracking problem is first solved by using an NMPC strategy proposed in [126]. 189 observations including inputs, states and outputs are consequently collected and are used to learn the GP models for each trajectory tracking problem.

The MPC parameters in the simulations are given as: initial states $\mathbf{x}_0 = [0, 0, 0, 0]^T$ and initial control inputs $\mathbf{u}_0 = [0, 0]^T$, weighting matrix $\mathbf{R} = \text{diag}\{[27000, 21000]\}$ and



5.1.a: "Step" 5.1.b: "Lorenz" 5.1.c: "Duffing"

Figure 5.1: Training errors over the sampling time in the trajectory tracking simulations

Table 5.1: Simulation Results of learning the unknown nonlinear system by using GP models

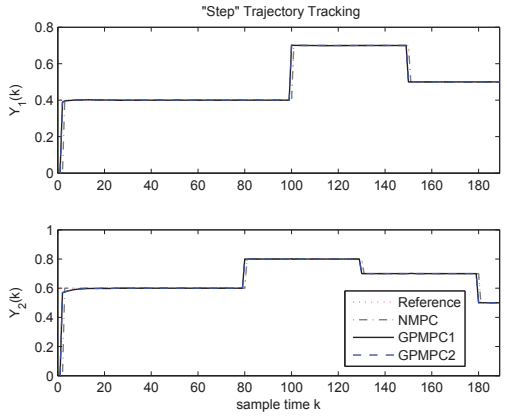
Trajectory	MSE Values	Approximate Runtime (seconds)	
		GP Models	RBFN
"Step"	9.9114e-05	2.17	1.67
"Lorenz"	0.0196	1.91	14.1
"Duffing"	0.0012	2.01	6.38

$\mathbf{Q} = \mathbf{I}_{4\times4}$, sampling frequency $f_s = 1$Hz. In addition, it is required to specify the horizon H. Theoretically, a long enough H is necessary to guarantee the stability of MPC controllers. However, the complexity of MPC problem increases exponentially with the horizon's growth. Meanwhile, when using the GP models, variances are also propagated over the horizon. Thus, it is usually proposed to chose the H to make a trade-off between the control performance and computational complexity. In this section, it is defined that $H = 10$.

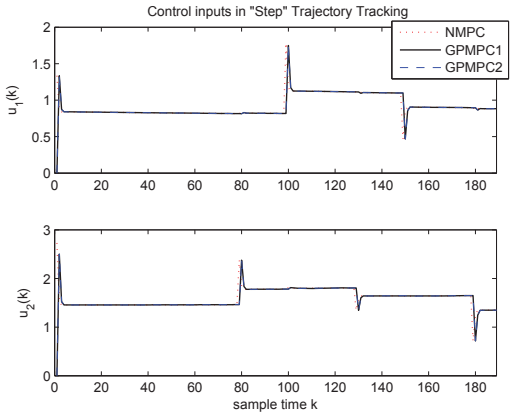
5.5.2 Unknown System Learning Results

Table 5.1 describes the obtained MSE values of learning the unknown nonlinear system by using GP models in these three trajectory tracking problems. The corresponding training errors over the 189 samples are given in Figure 5.1.a, 5.1.b and 5.1.c respectively. These results show that the nonlinear system (5.52) is accurately learnt by using the GP models due to small MSE values and tracking errors.

In addition, we compare the runtime required to obtain the equal MSE values when learning the unknown nonlinear system by using the GP models and Radial Basis Func-

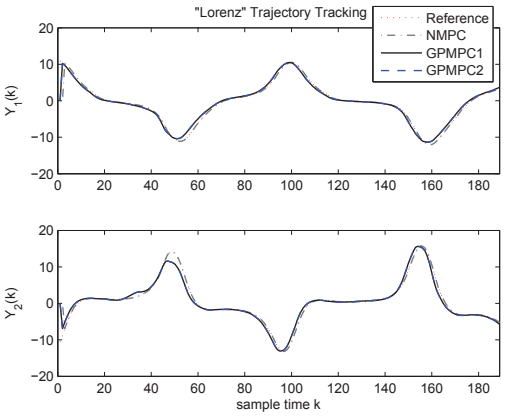


5.2.a: Controlled Outputs

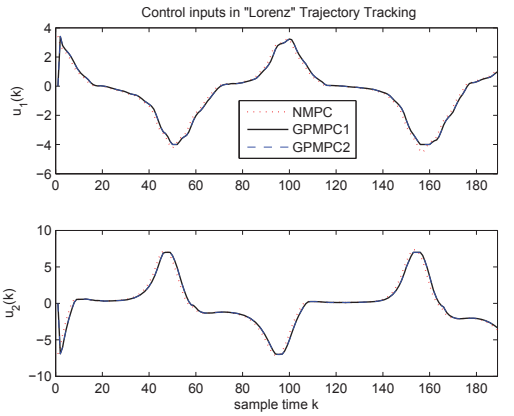


5.2.b: Control Inputs

Figure 5.2: Simulation result of tracking the "Step" trajectory using the proposed algorithms



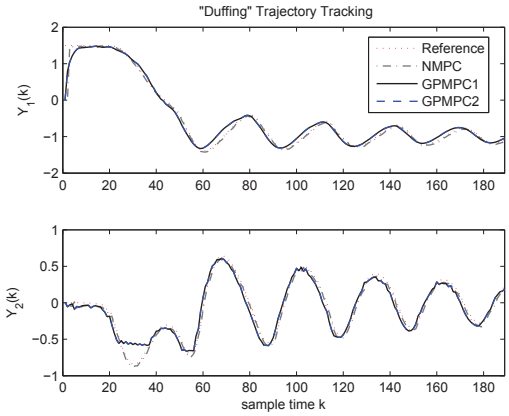
5.3.a: Controlled Outputs



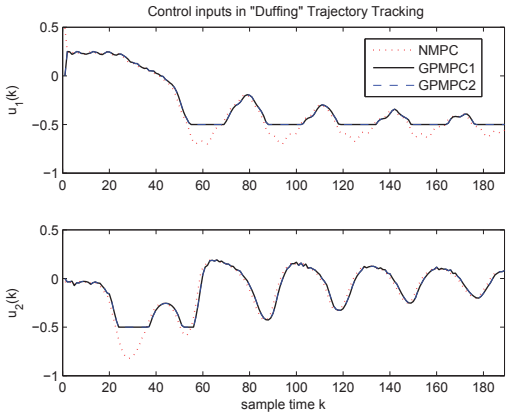
5.3.b: Control Inputs

Figure 5.3: Simulation result of tracking the "Lorenz" trajectory using the proposed algorithms

tion Network (RBFN), even though the training method and cost function used in these two approaches are not same. The resultant runtimes are given in Table 5.1. It seems that the GP models require less runtime to learn the unknown nonlinear system with the same accuracy than the RBFN.



5.4.a: Controlled Outputs



5.4.b: Control Inputs

Figure 5.4: Simulation result of tracking the "Duffing" trajectory using the proposed algorithms

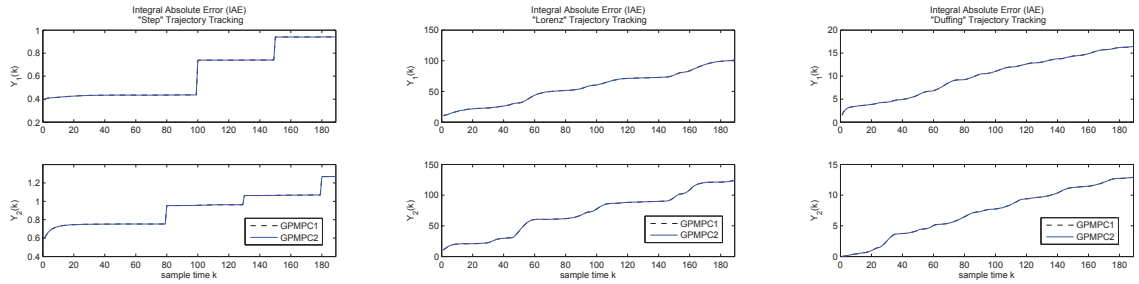
5.5.3 Unknown System Control Results

The resultant controlled outputs and control inputs by using the GPMPC1 and GPMPC2 algorithms for the trajectory tracking problems are given in Figure 5.2, 5.3 and 5.4. For the "Step" and "Lorenz" trajectories, the effectivenesses of both algorithms are obvious since they both produced outputs closed to the desired ones. In addition, both the algorithms exhibit equally good control performances in these two trajectory tracking problems as indicated by the very close control inputs shown in the Figure 5.2.b and 5.3.b.

For the "Duffing" trajectory, the both algorithms are able to produce the controlled outputs $y_1(k)$ overall following the desired ones with constrained control inputs $u_1(k)$ shown in Figure 5.4.a. In addition, the controlled outputs $y_2(k)$ have obvious errors to the desired outputs at the beginning 60 time steps. This is probably due to the control inputs $u_2(k)$ are constrained shown in Figure 5.4.b. The control outputs subsequently follow the reference closely.

As given in Figure 5.5, the equal control performances of the proposed algorithms also can be found by comparing the IAE values obtained in the three trajectory tracking problems.

Table 5.2 approximately compares the GPMPC1 and GPMPC2's runtime of computing 189 control inputs in the "Step", "Lorenz" and "Duffing" trajectories. The results demonstrate the better efficiency of using GPMPC2 algorithm to deal with the GP



5.5.a: "Step" 5.5.b: "Lorenz" 5.5.c: "Duffing"

Figure 5.5: Integral Absolute Errors (IAE) over the sampling time in the trajectory tracking simulations

Table 5.2: Runtime required to compute 189 control inputs by using the proposed algorithms in the trajectory tracking simulations

Trajectory	Runtime (seconds)	
	"GPMPC1"	"GPMPC2"
"Step"	33.09	5.93
"Lorenz"	30.28	5.11
"Duffing"	27.11	4.86

based MPC optimization problem than the GPMPC1 algorithm.

5.6 Conclusion

Two GP based MPC algorithms GPMPC1 and GPMPC2 are proposed to handle the trajectory tracking problem of the unknown systems in this chapter. GP models are used to identify the unknown system offline due to their ability of explicitly evaluating uncertainties of the obtained model by using GP variances. The general GP based MPC problem based on GP model is subsequently formulated such that the model uncertainties are directly included in the cost function. In the GPMPC1 algorithm, this general GP based MPC problem is originally stochastic but is relaxed to a deterministic non-convex nonlinear optimization problem through specifying a confidence level. The resultant problem is subsequently solved effectively by using the FP-SQP method based on the basic GP based local dynamical model. In addition, by using the extended GP based local dynamical model, the GPMPC2 algorithm further relaxes the non-convex optimization problem to a convex one. The simplified question is solved by using the active-set method. The main difference between the proposed algorithms lies in the way they handle the

model uncertainties. In the GPMPC1 algorithm, model uncertainties are treated as the slack variables of GP mean constraints and are included in the objective function as the penalty terms. This is considered as a "soften" technique to the "hard" state constraints. Another approach is used with the uncertainties is treated as partial entries of the state vector in the GPMPC2 algorithm. This allows directly dealing with the model uncertainties when computing the optimized control inputs. Based on the simulation results, the proposed algorithms perform equally good in the trajectory tracking problem of the unknown nonlinear system. However, the GPMPC2 algorithm seems superior due to the shorter time is required in the GP based MPC optimization problems compared with the GPMPC1 algorithm.

Chapter 6

Quadrotor Control using GPMPC

MPC trajectory tracking problem of the quadrotor is studied in this chapter. The translational and rotational dynamics of the quadrotor are assumed to be unknown and are modelled by GP techniques. A hierarchical scheme based on the GPMPC2 algorithm is used to control the quadrotor. The translational and rotational subsystems are given in Section 6.1. The proposed GPMPC2 based hierarchical control scheme is presented in Section 6.2. Section 6.3 reports the simulation results.

6.1 Quadrotor Dynamical Equations

A quadrotor helicopter (quadrotor for short) basically has four rotors in a cross configuration structure. Four propellers are connected to the corresponding motors through reduction gears, and the rotation axe of each propeller is fixed and parallel to others. When the quadrotor works, four rotors are divided into two groups. The first group consists of front and gear rotors (rotor 2 and 4 respectively as shown in Figure 6.1) that rotate counter-clockwise. The remaining right and left rotors (rotor 1 and 3 respectively as shown in Figure 6.1) that rotate clockwise belongs to the second group.

Four independent thrusts, F_1, F_2, F_3 and F_4 as shown in Figure 6.1 are generated by corresponding rotors. Working together, they control movements with six coupled Degree-of-Freedom (DOF) – three translational x, y, z and three rotational ϕ, θ, ψ. The speed differences between rotors in the same group will produce corresponding torques to drive roll or pitch movements, as shown in Figure 6.2.b and Figure 6.2.c respectively, Further, the speed difference between these two torques will produce the yaw torque, as shown

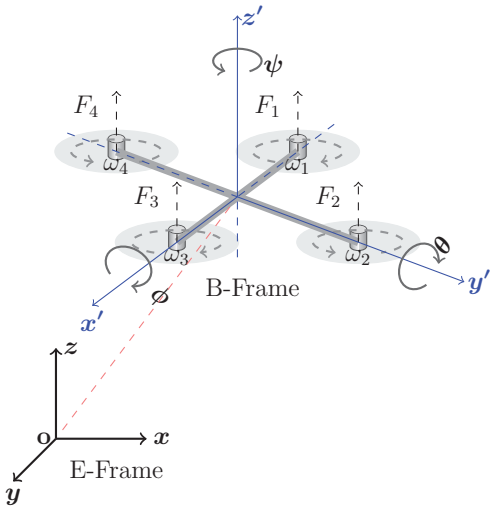


Figure 6.1: Quadrotor Body-Inertial Frame

in Figure 6.2.d. In addition, Figure 6.2.a shows same speed over four rotors will keep the quadrotor hovering. Otherwise, increasing (or decreasing) the same amount of speed over all rotors will allow the quadrotor reach a certain height.

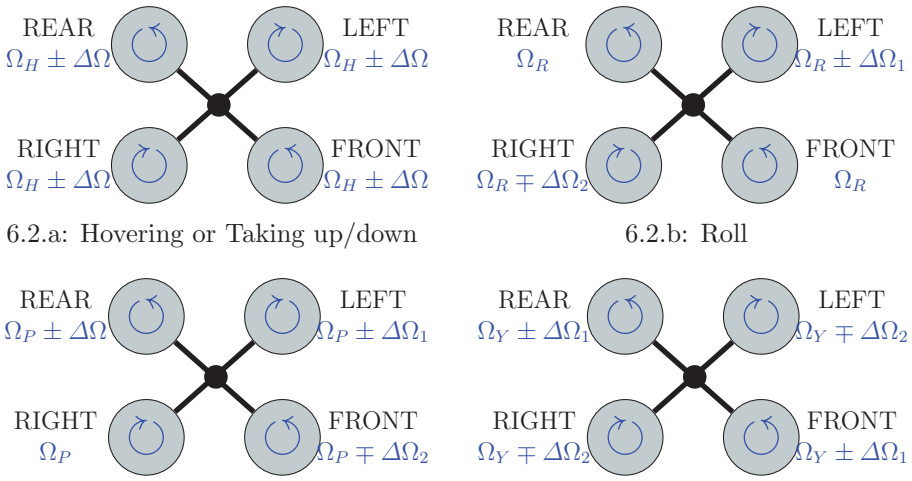
In this section, the quadrotor is basically considered as a 6 DOF rigid body where the entire quadrotor is represented as a mass with inertia and autogyroscopics driven by gravity and controlled torques and forces. In addition, two reference frames are defined as shown in Figure 6.1, including earth inertial reference frame (E-frame for short) and Body-fixed reference frame(B-frame for short).

The dynamical equations of a quadrotor can be obtained by using the Newton-Euler formalism based on forces/torques analysis of a rigid body, or energy-based Lagrange-Euler formalism. Assuming $\boldsymbol{\xi}^E[m] = [x, y, z]^T$ and $\boldsymbol{\eta}^E[rad] = [\phi, \theta, \psi]^T$ are the position and attitude vectors of the quadrotor in the E-frame, where ϕ, θ and ψ are rotation angles around x, y and z-axis, respectively. The generalized coordinate vector can be subsequently defined by,

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\xi}^E \\ \boldsymbol{\eta}^E \end{bmatrix} \tag{6.1}$$

The linear and angular velocities of the quadrotor also are defined in the B-frame by $\boldsymbol{V}^B[m \cdot s^{-1}] = [u, v, w]^T$ and $\boldsymbol{\Omega}^B[rad \cdot s^{-1}] = [p, q, r]$.

In this section, the dynamical models of the quadrotor are constructed based on the Euler-Lagrange formalism from the energy perspective. In particular, the Lagrangian of



6.2.a: Hovering or Taking up/down

6.2.b: Roll

6.2.c: Pitch

6.2.d: Yaw

Figure 6.2: Schematic diagram of quadrotor movements. Where Ω denotes the speed of propellers, and $\Delta\Omega$ represents the increment on Ω.

the quadrotor is typically defined by,

$$\boldsymbol{\mathcal{L}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = E_T + E_R - E_P \tag{6.2}$$

where E_T and E_R denote the translational and rotational kinetic energies, and E_P is the total potential energy. The Euler-Lagrange equation is subsequently obtained [2, 140] as,

$$\begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\Gamma} \end{bmatrix} = \frac{d}{dt}\left(\frac{\partial \boldsymbol{\mathcal{L}}}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \boldsymbol{q}} \tag{6.3}$$

where $\boldsymbol{F}$ is the translational force applied to the quadrotor, and $\boldsymbol{\Gamma} = [\boldsymbol{\tau}_\phi, \boldsymbol{\tau}_\theta, \boldsymbol{\tau}_\psi]^T$ represents the moments in the roll, pitch and yaw directions.

As discussed in [2], (6.3) can be further separated into a translational motion equation given by,

$$\boldsymbol{F} = m\ddot{\boldsymbol{\xi}}^E + mg\boldsymbol{e_z} \tag{6.4}$$

and a rotational motion equation given by,

$$\boldsymbol{\Gamma} = \boldsymbol{\mathcal{J}}\ddot{\boldsymbol{\eta}}^E + \boldsymbol{\mathcal{C}}\dot{\boldsymbol{\eta}}^E \tag{6.5}$$

where m is the mass of the quadrotor and g is gravitational acceleration, $\boldsymbol{e_z} = [0, 0, 1]^T$ is a unit vector in the E-frame. In addition, $\boldsymbol{\mathcal{J}}$ and $\boldsymbol{\mathcal{C}}$ are the Jacobian and Coriolis matrices (see [2] for an elaboration of the two matrices).

The translational motion equation (6.4) can be further expressed by means of state vector $\boldsymbol{\xi}^E$,

$$\begin{cases} \ddot{x} = \dfrac{1}{m} u_x U_1 + \dfrac{A_x}{m} \\ \ddot{y} = \dfrac{1}{m} u_y U_1 + \dfrac{A_x}{m} \\ \ddot{z} = -g + \dfrac{1}{m}(\cos\phi\cos\theta) + \dfrac{A_z}{m} \end{cases} \tag{6.6}$$

with intermediate controls

$$\begin{aligned} u_x &= \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ u_y &= \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \end{aligned} \tag{6.7}$$

where A_x, A_y and A_z are aerodynamic forces that are independently applied to x, y and z axis in the E-frame.

Assuming the state vector $\mathbf{x}^\xi = [x, \dot{x}, y, \dot{y}, z, \dot{z}]^T$ and control input vector $\mathbf{u}^\xi = [U_1, u_x, u_y]^T$, the corresponding translational subsystem based on the (6.6) can be obtained in the state-space form,

$$\begin{aligned} \dot{\mathbf{x}}^\xi &= f\left(\mathbf{x}^\xi, \mathbf{u}^\xi\right) + \boldsymbol{\epsilon}^\xi \\ &= \begin{pmatrix} \dot{x} \\ u_x \dfrac{U_1}{m} + \dfrac{A_x}{m} \\ \dot{y} \\ u_y \dfrac{U_1}{m} + \dfrac{A_y}{m} \\ \dot{z} \\ -g + (\cos\phi\cos\theta)\dfrac{U_1}{m} + \dfrac{A_z}{m} \end{pmatrix} + \boldsymbol{\epsilon}^\xi, \end{aligned} \tag{6.8}$$

where $\boldsymbol{\epsilon}^\xi$ denotes the external disturbances in the translational subsystem.

Similarly, we can rewrite the (6.5) by means of $\boldsymbol{\eta}^E$,

$$\begin{cases} \ddot{\theta} = \dot{\theta}\dot{\psi}\left(\dfrac{I_{yy} - I_{zz}}{I_{xx}}\right) + \dfrac{J_R}{I_{xx}}\dot{\theta}\Omega_R + \dfrac{L}{I_{xx}}U_2 \\ \ddot{\theta} = \dot{\phi}\dot{\psi}\left(\dfrac{I_{zz} - I_{xx}}{I_{yy}}\right) - \dfrac{J_R}{I_{yy}}\dot{\phi}\Omega_R + \dfrac{L}{I_{yy}}U_3 \\ \ddot{\psi} = \dot{\theta}\dot{\phi}\left(\dfrac{I_{xx} - I_{yy}}{I_{zz}}\right) + \dfrac{L}{I_{zz}}U_4 \end{cases} \tag{6.9}$$

where L is the arm length of the quadrotor, J_R denotes the inertial moment of rotors, and Ω_R represents the overall residual angular speed of propellers. In addition, I_{xx},I_{yy}

and I_{zz} are the inertial moments w.r.t. the corresponding x, y and z axis.

The rotational subsystem can be obtained as well with the state vector $\mathbf{x}^\eta = [\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}]^T$ and control input vector $\mathbf{u}^\eta = [U_2, U_3, U_4]^T$,

$$\begin{aligned}\dot{\mathbf{x}}^\eta &= g\left(\mathbf{x}^\eta, \mathbf{u}^\eta\right) + \boldsymbol{\epsilon}^\eta \\ &= \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right) + \frac{J_R}{I_{xx}}\dot{\theta}\Omega_R + \frac{L}{I_{xx}}U_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right) - \frac{J_R}{I_{yy}}\dot{\phi}\Omega_R + \frac{L}{I_{yy}}U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right) + \frac{L}{I_{zz}}U_4 \end{pmatrix} + \boldsymbol{\epsilon}^\eta,\end{aligned} \tag{6.10}$$

where $\boldsymbol{\epsilon}^\eta$ is the external disturbances in the rotational subsystem.

6.2 Quadrotor Control using GPMPC

6.2.1 Overall Control Scheme

The trajectory tracking problem of the quadrotor involves tracking the desired positions of the translational subsystem, as well as the desired attitudes of the rotational subsystem. A hierarchical control structure used in [2, 141] is employed in this section to handle these two tracking problems sequentially. The block diagram of this structure is shown in Figure 6.3. In the outer loop, the translational subsystem is controlled to follow the sequence of desired positions generated by the "Trajectory Generator". The optimal controls U_1 are obtained by minimizing the tracking errors in the "Translation Controller" that also produces desired attitudes θ_d and ϕ_d from intermediate controls u_x and u_y given by (6.7). Then, the attitudes of the rotational subsystem are tuned to achieve the target values in the inner loop where the desired ψ_d is always set to zero. By minimizing the attitude errors again, the optimal controls U_2, U_3 and U_4 can be obtained from the "Rotation Controller". Finally, the optimized control inputs U_1, U_2, U_3 and U_4 are applied to the quadrotor.

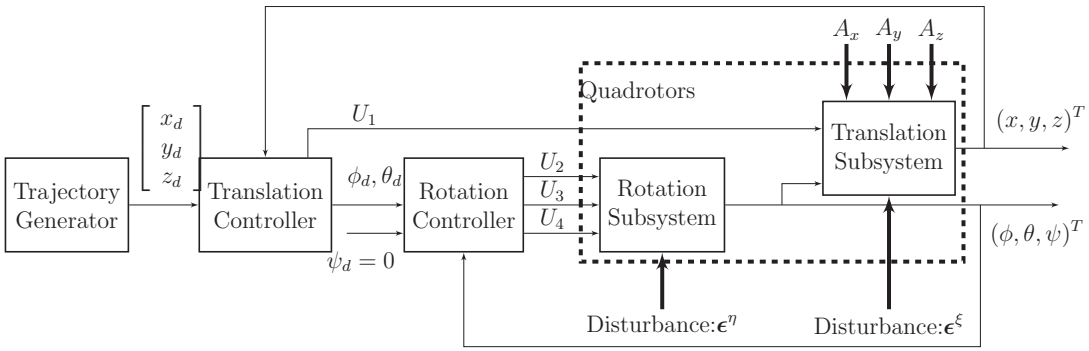


Figure 6.3: The Overall Control Scheme for Quadrotors

6.2.2 GP Learning of Quadrotor Dynamic Equations

The accurate translational and rotational equations are required when using the proposed MPC based control approaches. In this section, they are assumed unknown and are learnt by using the data-driven GP modelling technique, while the deterministic models are required in [2, 141]. As shown in Section 6.1, they are basically nonlinear time-varying systems and can be uniformly expressed by the general form in (4.1). Then through using GP models with the consideration of the uncertainty propagation, the corresponding GP models for the translational and rotational subsystems can be obtained in the same way presented in Section 4.1.1.

6.2.3 GPMPC2 for Quadrotor Trajectory Tracking Control

The discrete MPC trajectory tracking problem of the translational subsystem can be defined by,

$$\min_{\mathbf{u}^{\xi}(\cdot)} \sum_{i=1}^{H} \left\{ \left\| \mathbf{x}^{\xi}_{k+i} - \mathbf{r}^{\xi}_{k+i} \right\|^2_{\mathbf{Q}} + \left\| \mathbf{u}^{\xi}_{k+i-1} \right\|^2_{\mathbf{R}} \right\} \tag{6.11a}$$

$$\text{s.t.} \quad \mathbf{x}^{\xi}_{k+i+1} = \hat{f}(\mathbf{x}^{\xi}_{k+i}, \mathbf{u}^{\xi}_{k+i-1}) + \boldsymbol{\epsilon}^{\xi}_{k+i} \tag{6.11b}$$

$$\mathbf{x}^{\xi}_{\min} \leq \mathbf{x}^{\xi}_{k+i} \leq \mathbf{x}^{\xi}_{\max} \tag{6.11c}$$

$$\mathbf{u}^{\xi}_{\min} \leq \mathbf{u}^{\xi}_{k+i-1} \leq \mathbf{u}^{\xi}_{\max} \tag{6.11d}$$

where $\hat{f}(\cdot)$ is the discretized function of $f(\cdot)$ in (6.8), $\mathbf{x}^{\xi}_{k+i}$, $\mathbf{u}^{\xi}_{k+i-1}$ and $\boldsymbol{\epsilon}^{\xi}_{k+i}$ are the corresponding discretized states, control inputs and external noises. In addition, $\mathbf{r}^{\xi}_{k+i}$ denotes the desired positions at sampling time $k+i$ with the entries $x_{d,k+i}$, $y_{d,k+i}$ and $z_{d,k+i}$.

Similarly, we can define the MPC trajectory tracking problem of the rotational subsystem,

$$\min_{\mathbf{u}^{\eta}(\cdot)} \sum_{i=1}^{H} \left\{ \left\| \mathbf{x}_{k+i}^{\eta} - \mathbf{r}_{k+i}^{\eta} \right\|_{\mathbf{Q}'}^{2} + \left\| \mathbf{u}_{k+i-1}^{\eta} \right\|_{\mathbf{R}'}^{2} \right\} \tag{6.12a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+i+1}^{\eta} = \hat{g}(\mathbf{x}_{k+i}^{\eta}, \mathbf{u}_{k+i-1}^{\eta}) + \boldsymbol{\epsilon}_{k+i}^{\eta} \tag{6.12b}$$

$$\mathbf{x}_{\min}^{\eta} \leq \mathbf{x}_{k+i}^{\eta} \leq \mathbf{x}_{\max}^{\eta} \tag{6.12c}$$

$$\mathbf{u}_{\min}^{\eta} \leq \mathbf{u}_{k+i-1}^{\eta} \leq \mathbf{u}_{\max}^{\eta} \tag{6.12d}$$

where $\hat{g}(\cdot)$ is the discretized function of $g(\cdot)$ in (6.10), $\mathbf{x}_{k+i}^{\eta}$, $\mathbf{u}_{k+i-1}^{\eta}$ and $\boldsymbol{\epsilon}_{k+i}^{\eta}$ are the corresponding discretized states, control inputs and external noises, $\mathbf{r}_{k+i}^{\eta}$ denotes the desired attitudes at sampling time $k+i$ with the entries $\phi_{d,k+i}$, $\theta_{d,k+i}$ and $\psi_{d,k+i}$.

In [2, 141], the deterministic translational and rotational subsystem functions are known. Therefore, the "Translation Controller" and "Rotation Controller" are directly designed based on the deterministic nonlinear MPC technique. In this section, the MPC problem (6.11) and (6.12) are stochastic due to the subsystems are learnt by GP models.

The stochastic problems can be uniformly expressed by the general MPC problem (5.9) with the cost function (5.10). As given in Section 5.2, they are subsequently reformulated to the deterministic ones by specifying the confidence level in the chance constraints. The resultant MPC problems can be effectively solved by using the proposed GPMPC1 and GPMPC2 algorithms. In this section, the GPMPC2 algorithm is used due to its superior efficiency shown in the Section 5.5.

6.3 Simulation Results

The performance of the GPMPC2 based hierarchical control approach to the trajectory tracking problem of the quadrotor is evaluated by the computer simulations. The parameters used in translational and rotational dynamical equation (6.4) and (6.5) are the same as those used in [141]. All simulations are independently repeated 50 times and the average results are shown here.

The modelling performance of proposed algorithm is evaluated by computing the consumed time and training MSE given as (5.52) in Section 5.5. The control performance is demonstrated in terms of computed inputs and outputs, as well as the IAE value given

as (5.52) in Section 5.5.

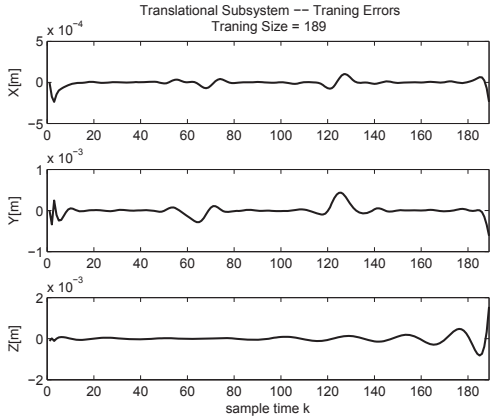
The so called "Elliptical" and "Lorenz" trajectories (shown as red dotted line in Figure 6.8.a and 6.8.b) are used in the presence of external Gaussian white noises with zero mean and unit variance. The translation subsystem inputs are subjected to $0 \leq U_1(k) \leq 100, -0.2 \leq u_x(k) \leq 0.2, -0.2 \leq u_y(k) \leq 0.2$ for the "Elliptical" trajectory and $-45 \leq u_1(k) \leq 0, -2 \leq u_x(k) \leq 2, -2 \leq u_y(k) \leq 2$ for the "Lorenz" trajectory. For the rotational subsystem, all observations are scaled to the range $[0.1, 0.9]$ therefore the inputs are scaled as well. This is necessary mainly due to the large numerical ranges in the original data. For example, the unscaled angle ϕ lies in the range $[-1.57, 1.57]$ while input U_4 lies in the range $[-3.2, 6.2] \times 10^{-8}$. The scaled data leads to much improved training results.

To generate the observations, each trajectory tracking problem is first solved by using the hierarchical scheme based on the NMPC strategy proposed in [126]. 189 observations including inputs, states and outputs are consequently collected and are used to learn the GP models for each trajectory tracking problem.

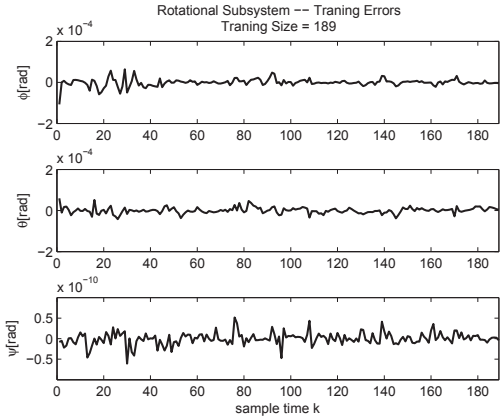
The MPC parameters used in the translational subsystem are given as: initial states $\mathbf{x}_0 = [0, 0, 0, 0, 0, 0]^T$ and initial control inputs $\mathbf{u}_0 = [0, 0, 0]^T$, weighting matrix $\mathbf{R} = \text{diag}\{[27000, 21000, 21000]\}$ and $\mathbf{Q} = \mathbf{I}_{6\times6}$, sampling frequency $f_s = 1$Hz. The same parameters used in the rotational subsystem. In addition, it is required to specify the horizon H. Theoretically, a long enough H is necessary to guarantee the stability of MPC controllers. However, the complexity of MPC problem increases exponentially with the horizon's growth. Meanwhile, when using the GP models, variances are also propagated over the horizon. Thus, it is usually proposed to chose the H to make a trade-off between the control performance and computational complexity. In this section, it is defined that $H = 10$ for both "GPMPC2" based controller.

6.3.1 Modelling Results

Table 6.1 describes the obtained MSE values of learning the unknown translational and rotational subsystems by using GP models in the trajectory tracking problems. The corresponding translational and rotational training errors over the 189 samples are given in Figure 6.4 and 6.5 respectively. These results show the obtained translational and rotational subsystems learnt by GP models produce very small MSE values and tracking errors in the both trajectory tracking simulations.



6.4.a: Translational Subsystem



6.4.b: Rotational Subsystem

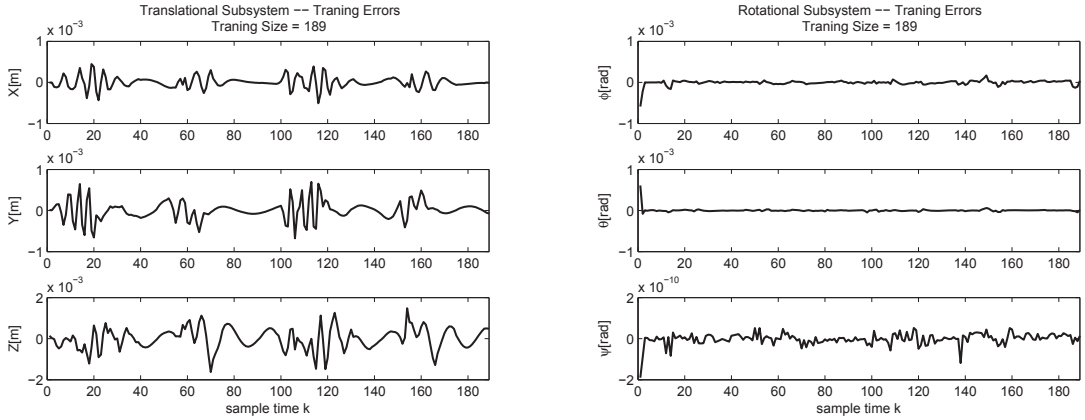
Figure 6.4: Modelling results of using GP modelling technique in the "Elliptical" trajectory tracking problem

Table 6.1: Modelling MSE values of the translational and rotational subsystems using the GP models in the trajectory tracking problems

	MSE Values	
	"Elliptical"	"Lorenz"
Position X	1.5603×10^{-9}	2.004×10^{-8}
Position Y	1.2642×10^{-9}	4.8964×10^{-8}
Position Z	3.7724×10^{-8}	2.6668×10^{-7}
Attitude ϕ	3.2111×10^{-10}	3.5194×10^{-9}
Attitude θ	2.1667×10^{-10}	3.5194×10^{-9}
Attitude ψ	2.2526×10^{-12}	7.1917×10^{-11}

6.3.2 Control Results

The results of using the GPMPC2 based hierarchical control approach to the trajectory tracking problems are given in Figure 6.6 and 6.7. The results of using the NMPC are used here as a reference. The effectivenesses of the proposed control scheme are obvious since the computed outputs of the both translational and rotational subsystems are closed to the desired ones. In addition as shown in the Figure 6.6.b, 6.6.d, 6.7.b and 6.7.d, the quadrotor is able to overall follow the desired trajectories with constrained control inputs by using the proposed approach. The overall trajectory tracking results are depicted in Figure 6.8.a and 6.8.b.



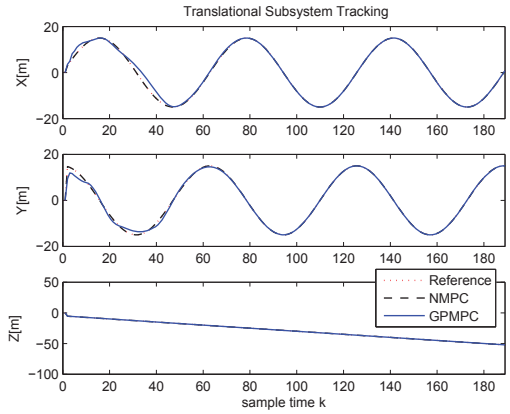
6.5.a: Translational Subsystem

6.5.b: Rotational Subsystem

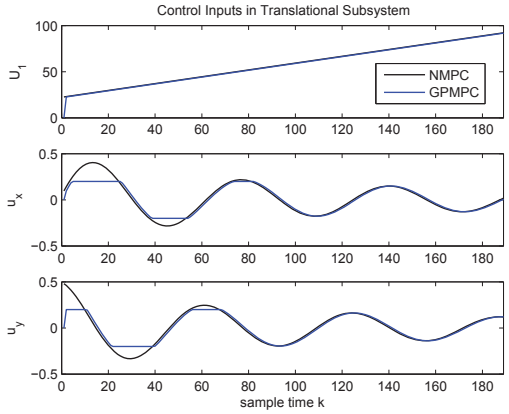
Figure 6.5: Modelling results of using GP modelling technique in the "Lorenz" trajectory tracking problem

6.4 Conclusion

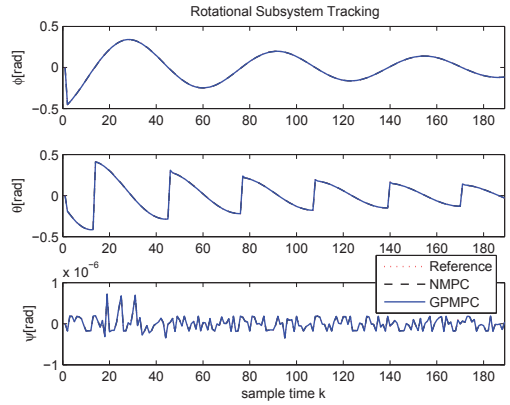
A hierarchical scheme based on the GPMPC2 algorithm is proposed in this chapter for the trajectory tracking problem of the quadrotor. The overall control structure consists of two separate MPC controllers for the translational and rotational subsystem respectively. The GP models are used to learn the unknown translational and rotational subsystem of the quadrotor due to the model uncertainties can be explicitly evaluated. The GPMPC2 algorithm is subsequently used to compute the optimized control inputs in the trajectory tracking problems of translational and rotational subsystem respectively. Simulation results on the quadrotor trajectory tracking problems demonstrate the effectiveness of using the proposed hierarchical scheme based on the GPMPC2 algorithm.



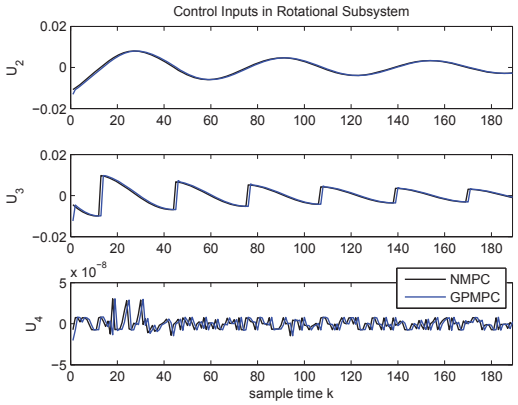
6.6.a: Translational Controlled Outputs



6.6.b: Translational Control Inputs

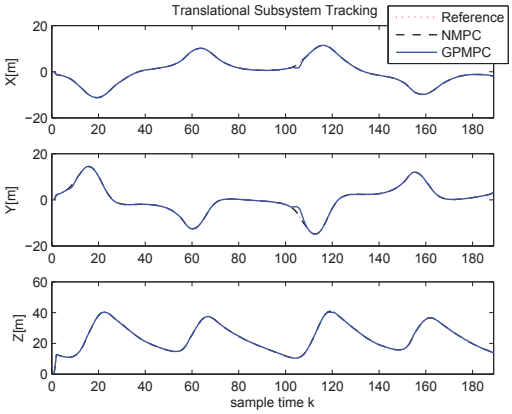


6.6.c: Rotational Controlled Outputs

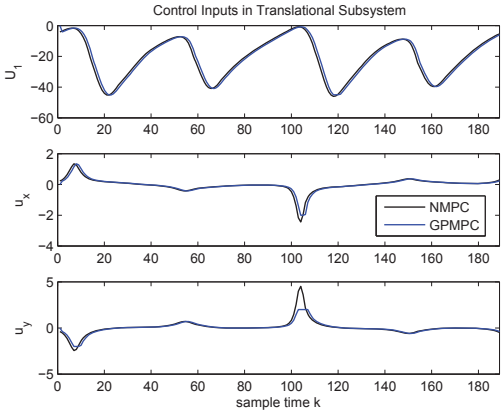


6.6.d: Rotational Control Inputs

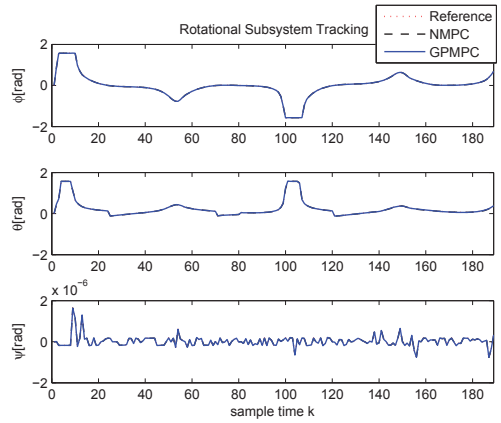
Figure 6.6: Simulation results of tracking the "Elliptical" trajectory using the GPMPC2 based approach



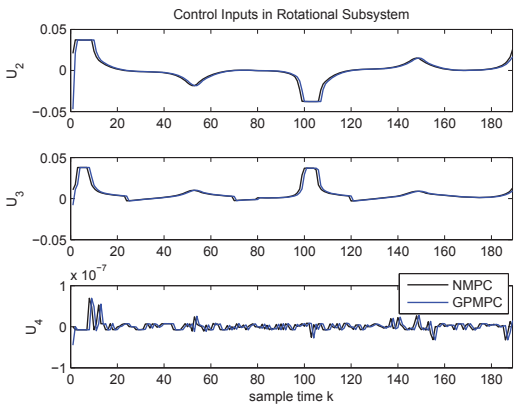
6.7.a: Translational Controlled Outputs



6.7.b: Translational Control Inputs

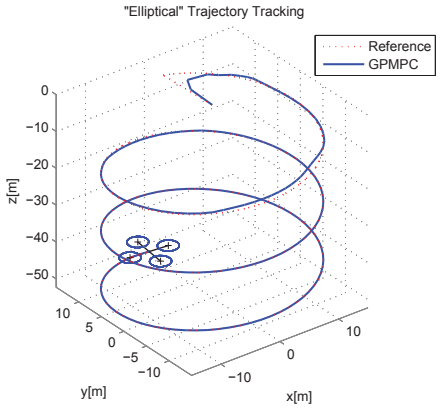


6.7.c: Rotational Controlled Outputs

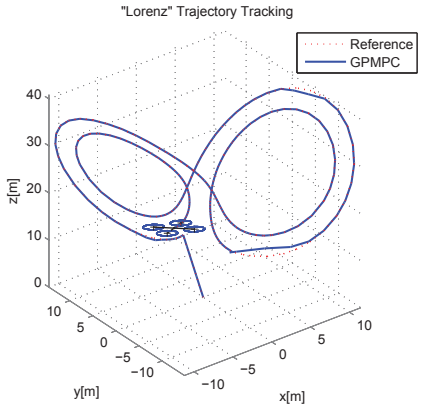


6.7.d: Rotational Control Inputs

Figure 6.7: Simulation results of tracking the "Lorenz" trajectory using the GPMPC2 based approach



6.8.a: "Elliptical"



6.8.b: "Lorenz"

Figure 6.8: "Elliptical" and "Lorenz" trajectory tracking results of using the GPMPC2 based approach

Chapter 7

Stability Guaranteed GPMPC

The MPC problem (5.9) based on the cost function (5.10) results in open-loop control with no guaranteed stability. This is mainly because the prediction and control horizon H is finite, and therefore the system behaviour after H is not taken into account in the optimization process. Several approaches have been used to address the stability issue of MPC [1, 142]. A simple and direct approach is to extend the finite horizon to an infinite one [143, 144]. However this requires solving an infinite MPC optimization problem that is computationally very demanding or even infeasible. Another commonly used approach is to add a terminal cost to the cost function. Usually, an appropriate terminal constraint set may be also need to be added in order to guarantee the feasibility of predicted system states beyond the horizon.

In this chapter, the stability issue with GPMPC is addressed through the introduction of terminal cost and terminal constraints. The GPMPC problem is reformulated and a computationally efficient algorithm based on the extended GP local model is proposed to solve it. Finally, the stability of the proposed GPMPC algorithm is proved.

7.1 Stability Guaranteed GPMPC Algorithm

7.1.1 Terminal Cost and Constraints

Assume that a terminal state feedback controller $\mathcal{K}(\cdot)$ exists which satisfies the following conditions:

$$\mathbf{\Omega}_F \subset \mathbb{X} \tag{7.1a}$$

$$\mathcal{K}(\mathbf{x}) \in \mathbb{U}, \forall \mathbf{x} \in \mathbf{\Omega}_F \tag{7.1b}$$

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathcal{K}(\mathbf{x}) \in \mathbf{\Omega}_F, \forall \mathbf{x} \in \mathbf{\Omega}_\Psi \tag{7.1c}$$

$$\mathbf{\Psi}(\mathbf{x}) - \mathbf{\Psi}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathcal{K}(\mathbf{x})) \geq \|\mathbf{x}\|^2_{\mathbf{Q}} + \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathcal{K}(\mathbf{x})\|^2_{\mathbf{R}} \tag{7.1d}$$

Here, $\mathbb{X}$ is the feasible domain of the states which is defined by the state constraints in (5.9), $\mathbb{U}$ is the feasible domain of the control inputs which is defined by the control constraints in (5.9), $\mathbf{\Omega}_\Psi$ denotes the feasible domain of the terminal states, $\mathbf{\Psi}(\cdot) : \mathbb{R}^n \to \mathbb{R}$ represents the terminal cost function, $\mathbf{A}$ and $\mathbf{B}$ are two Jacobian matrices of the linearised model of the system.

In order to guarantee asymptotic stability of the MPC controller, terminal cost is added to the original cost function (5.10) so that it becomes

$$\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) = \sum_{i=1}^{H} \left\{ \left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|^2_{\mathbf{Q}} + \left\| \mathbf{u}_{k+i-1} \right\|^2_{\mathbf{R}} \right\} + F(\mathbf{x}_{k+H+1|k} - \mathbf{r}_{k+H+1}) \tag{7.2}$$

In, There are a number of methods for choosing this terminal cost and the corresponding terminal constraints [1, 142]. The terminal cost is usually of an ellipsoid form:

$$F(\mathbf{x}_{k+H+1|k} - \mathbf{r}_{k+H+1}) = \left\| \mathbf{x}_{k+H+1|k} - \mathbf{r}_{k+H+1} \right\|^2_{\mathbf{P}} \tag{7.3a}$$

$$\mathbf{\Omega}_F = \left\{ \mathbf{x}_{k+H+1|k} \in \mathbb{R}^n, \left\| \mathbf{x}_{k+H+1|k} - \mathbf{r}_{k+H+1} \right\|^2_{\mathbf{P}} \leq \varepsilon \right\} \tag{7.3b}$$

where $\|\cdot\|^2_{\mathbf{P}}$ denote a 2-norm with weighting matrix $\mathbf{P}$, and $\varepsilon \geq 0$ is the upper bound of the terminal cost. The advantage of using (7.3) is that the MPC optimization problem is kept computationally simple. In addition, the $\mathbf{\Omega}_\Psi$ with a large enough volume of $\mathbf{vol}(\mathbf{\Omega}_\Psi) = 4/3\pi \det(\mathbf{P}/\varepsilon)$ is required to guaranteed the feasibility of the MPC problem [145]. This is easy to implement by tuning the value of ε using a trial-and-error approach.

A simple proportional controller $\mathcal{K}(\cdot) = \mathbf{K}\mathbf{x}$ can be used as the terminal state feedback controller, where $\mathbf{K}$ denotes the feedback gain matrix. $\mathbf{K}$ and $\mathbf{P}$ are usually obtained

by using the LMI [145] techniques. They can also be computed by solving an infinite horizon unconstrained Linear-Quadratic Regulator (LQR) problem [136] with the same weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ as those used for (5.10). This results in

$$\mathbf{K} = -(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A} \tag{7.4a}$$

$$\mathbf{P} = (\mathbf{A} + \mathbf{B}\mathbf{K})^T\mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{K}) + \mathbf{K}^T\mathbf{R}\mathbf{K} + \mathbf{Q} \tag{7.4b}$$

This implies that MPC control will switch to the unconstrained LQR control beyond the control horizon H.

7.1.2 Problem Formulation

With the unknown nonlinear system is represented by a GP model, a new stochastic MPC problem can now be stated using the modified cost function (7.2) as follows.

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \tag{7.5a}$$

$$\text{s.t. } p(\mathbf{x}_{k+1}|\mathbf{x}_k) \sim \mathcal{N}(\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}) \tag{7.5b}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i-1} \leq \mathbf{u}_{\max} \tag{7.5c}$$

$$p\big\{\mathbf{x}_{k+i|k} \geq \mathbf{x}_{\min}\big\} = \eta \tag{7.5d}$$

$$p\big\{\mathbf{x}_{k+i|k} \leq \mathbf{x}_{\max}\big\} = \eta \tag{7.5e}$$

$$\big\|\mathbf{x}_{k+H+1|k} - \mathbf{r}_{k+H+1}\big\|_{\mathbf{P}}^2 \leq \varepsilon \tag{7.5f}$$

where the η denotes the confidence level. For $\eta = 0.95$, the chance constraints (7.5d) and (7.5e) are equivalent to,

$$\boldsymbol{\mu}_{k+i} - 2\boldsymbol{\Sigma}_{k+i} \geq \mathbf{x}_{\min} \tag{7.6a}$$

$$\boldsymbol{\mu}_{k+i} + 2\boldsymbol{\Sigma}_{k+i} \leq \mathbf{x}_{\max} \tag{7.6b}$$

The expectation of cost function (7.2) is given by

$$\begin{aligned}
&E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \\
&= E\Big[\sum_{i=1}^{H}\Big\{\big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big\} + \|\mathbf{x}_{k+H+1} - \mathbf{r}_{k+H+1}\|_{\mathbf{P}}^2\Big] \\
&= \sum_{i=1}^{H} E\Big[\big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big] + E\Big[\|\mathbf{x}_{k+H+1} - \mathbf{r}_{k+H+1}\|_{\mathbf{P}}^2\Big] \\
&= \sum_{i=1}^{H}\Big\{E\Big[\big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2\Big] + E\Big[\big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big]\Big\} + E\Big[\|\mathbf{x}_{k+H+1} - \mathbf{r}_{k+H+1}\|_{\mathbf{P}}^2\Big]
\end{aligned} \tag{7.7}$$

where $E\big[\mathbf{u}_k^2\big] = \mathbf{u}_k^2$ because the control inputs are assumed to be deterministic. (7.7) can be further simplied to

$$\begin{aligned}
&E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \\
=&\sum_{i=1}^{H}\Big\{\big\|\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \mathrm{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}) + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big\} \\
&+ \|\boldsymbol{\mu}_{k+H+1} - \mathbf{r}_{k+H+1}\|_{\mathbf{P}}^2 + \mathrm{trace}(\mathbf{P}\boldsymbol{\Sigma}_{k+H+1})
\end{aligned} \tag{7.8}$$

This essentially relaxes the original stochastic optimization problem (7.5) to a deterministic one similar to the method used in Chapter 5.

7.1.3 Solution

Similar to GPMPC2 in Section 5.3, the extended GP local model can be used to relax the above non-convex and nonlinear optimization problem to a convex one.

Using the state and control variables defined by (5.25)-(5.27), problem (7.5) can be rewritten as

$$\min_{\mathbf{U}} \underbrace{\big\|\mathbf{Z}_{k+1} - \mathbf{r}_{k+1}^*\big\|_{\tilde{\mathbf{Q}}}^2 + \|\mathbf{U}_{k+1}\|_{\tilde{\mathbf{R}}}^2}_{\text{Stage Cost} \quad \boldsymbol{\Phi}_{0\to H}} + \underbrace{\|\mathbf{s}_{k+H+1} - \mathbf{r}_{k+H+1}\|_{\mathbf{P}}^2}_{\text{Terminal Cost} \quad \boldsymbol{\Phi}_{H\to\infty}} \tag{7.9a}$$

$$\text{s.t.} \quad \mathbf{I}_{Hn}\mathbf{x}_{\min} \le \mathbf{M}_z\mathbf{Z}_{k+1} \le \mathbf{I}_{Hn}\mathbf{x}_{\max} \tag{7.9b}$$

$$\mathbf{I}_{Hm}\mathbf{u}_{\min} \le \mathbf{U}_{k+1} \le \mathbf{I}_{Hm}\mathbf{u}_{\max} \tag{7.9c}$$

$$\|\mathbf{s}_{k+H+1} - \mathbf{r}_{k+H+1}\|_{\mathbf{P}}^2 \le \varepsilon \tag{7.9d}$$

where the parameter matrices $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{R}}$, $\mathbf{I}_{Hm}$, $\mathbf{I}_{Hn}$ and $\mathbf{M}_z$ are exactly same to those in

Initialization
the feasible point $\Delta\mathbf{U}_k^0 \in \mathbf{\Pi}_{\Delta\mathbf{U}}$;
the working set $\mathcal{W}^0 = \mathcal{A}(\Delta\mathbf{U}_k^0)$;
for $j = 0, 1, 2, \cdots$ **do**
  Replacing the $\mathbf{\Phi}$ and $\boldsymbol{\psi}$ in (5.42) with $\tilde{\mathbf{\Phi}}$ and $\tilde{\boldsymbol{\psi}}$ in (7.16);
  Compute the $\boldsymbol{\delta}^j$ and $\boldsymbol{\lambda}_k^*$ by solving the resultant linear equations;
  if $\boldsymbol{\delta}^j = 0$ **then**
    $\lambda_k^* = \min_{i \in \mathcal{W}_k^j \cap \mathcal{I}} \boldsymbol{\lambda}_{k,i}^*$,
    $p = \operatorname{argmin}_{i \in \mathcal{W}_k^j \cap \mathcal{I}} \boldsymbol{\lambda}_{k,i}^*$
    if $\lambda_k^* \geq 0$ *and (7.15c) is satisfied* **then**
      $\Delta\mathbf{U}_k^* = \Delta\mathbf{U}_k^j$;
      Stop.
    else
      $\mathcal{W}_k^{j+1} = \mathcal{W}_k^j \setminus p$;
      $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j$;
    end
  else
    Compute the step length κ^j by (5.46),
    $q = \operatorname{argmin}_{i \in \mathcal{B}(\Delta\mathbf{U}_k^j)} \dfrac{\tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^j}{\tilde{\mathbf{G}}_i \boldsymbol{\delta}^j}$
    if $\kappa^j < 1$ **then**
      $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j + \kappa^j \boldsymbol{\delta}^j$;
      $\mathcal{A}(\Delta\mathbf{U}_k^{j+1}) = \mathcal{A}(\Delta\mathbf{U}_k^j) \cup q$;
    else
      $\Delta\mathbf{U}_k^{j+1} = \Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j$;
      $\mathcal{A}(\Delta\mathbf{U}_k^{j+1}) = \mathcal{A}(\Delta\mathbf{U}_k^j)$;
    end
  end
end

Algorithm 8: Active set method for solving the Stability-Guaranteed GPMPC problem

(5.28). Based on (5.30) to (5.33), the "state cost" term $\boldsymbol{\Phi}_{0\to H}$ becomes

$$\boldsymbol{\Phi}_{0\to H} = \frac{1}{2} \|\Delta\mathbf{U}_k\|_{\mathbf{\Phi}}^2 + \boldsymbol{\psi}^T \Delta\mathbf{U}_k + \mathbf{C} \tag{7.10a}$$

$$\text{s.t.} \quad \Delta\mathbf{U}_{\min} \leq \begin{bmatrix} \mathbf{T}_u \\ \mathbf{T}_z \tilde{\mathbf{B}} \end{bmatrix} \Delta\mathbf{U}_k \leq \Delta\mathbf{U}_{\max} \tag{7.10b}$$

where $\mathbf{\Phi}$, $\boldsymbol{\psi}$, $\mathbf{C}$, $\Delta\mathbf{U}_{\max}$ and $\Delta\mathbf{U}_{\min}$ are exactly the same as in (5.35). In addition, based on the extended GP local model (5.5),

$$\begin{aligned}\mathbf{s}_{k+H+1} &= \mathbf{s}_{k+H} + \Delta\mathbf{s}_{k+H+1} \\ &= \tilde{\boldsymbol{\Gamma}}_1\mathbf{U}_k + \tilde{\boldsymbol{\Gamma}}_2\end{aligned} \tag{7.11}$$

with the parameters matrices $\tilde{\boldsymbol{\Gamma}}_1$ and $\tilde{\boldsymbol{\Gamma}}_2$ given by

$$\tilde{\boldsymbol{\Gamma}}_1 = \mathbf{\Gamma}_2 + \mathbf{\Gamma}_3\mathbf{A}\mathbf{B} + \mathbf{B}\mathbf{K}\mathbf{\Gamma}_3 \in \mathbb{R}^{(n+n^2)\times Hm} \tag{7.12a}$$

$$\tilde{\boldsymbol{\Gamma}}_2 = \mathbf{s}_k + \mathbf{\Gamma}_1\Delta\mathbf{s}_k + \mathbf{A}^{H+1}\Delta\mathbf{s}_k + \mathbf{B}\mathbf{K}\mathbf{A}^H\Delta\mathbf{s}_k \in \mathbb{R}^{n+n^2} \tag{7.12b}$$

where $\mathbf{A}$ and $\mathbf{B}$ are Jacobian matrices given in (5.6) and (5.7), and

$$\mathbf{\Gamma}_1 = \sum_{i=1}^{H}\mathbf{A}^i \in \mathbb{R}^{(n+n^2)\times(n+n^2)} \tag{7.13a}$$

$$\mathbf{\Gamma}_2 = \left[\sum_{i=0}^{H-1}\mathbf{A}^i\mathbf{B}, \sum_{i=0}^{H-2}\mathbf{A}^i\mathbf{B}, \cdots, \mathbf{A}\mathbf{B}, \mathbf{B}\right] \in \mathbb{R}^{(n+n^2)\times Hm} \tag{7.13b}$$

$$\mathbf{\Gamma}_3 = \left[\mathbf{A}^{H-1}, \cdots, \mathbf{A}, \mathbf{I}\right] \in \mathbb{R}^{(n+n^2)\times H(n+n^2)} \tag{7.13c}$$

Similarly, the "terminal cost" term $\boldsymbol{\Phi}_{H\to\infty}$ can be expressed as

$$\boldsymbol{\Phi}_{H\to\infty} = \tilde{\boldsymbol{\Gamma}}_1^2\mathbf{P}\mathbf{U}_k^2 + 2\mathbf{P}\tilde{\boldsymbol{\Gamma}}_1(\tilde{\boldsymbol{\Gamma}}_2 - \mathbf{r}_{k+H+1})\mathbf{U}_k + \mathbf{P}(\tilde{\boldsymbol{\Gamma}}_2 - \mathbf{r}_{k+H+1})^2 \tag{7.14}$$

Therefore, problem (7.9) can be expressed in the following condensed form:

$$\min_{\Delta\mathbf{U}} \frac{1}{2}\left\|\Delta\mathbf{U}_k\right\|_{\tilde{\mathbf{\Phi}}}^2 + \tilde{\boldsymbol{\psi}}^T\Delta\mathbf{U}_k + \tilde{\mathbf{C}} \tag{7.15a}$$

$$\text{s.t.} \quad \Delta\mathbf{U}_{\min} \le \begin{bmatrix}\mathbf{T}_u \\ \mathbf{T}_z\check{\mathbf{B}}\end{bmatrix}\Delta\mathbf{U}_k \le \Delta\mathbf{U}_{\max} \tag{7.15b}$$

$$\left\|\tilde{\boldsymbol{\Gamma}}_1\mathbf{T}_u\Delta\mathbf{U}_k + \mathbf{u}_{k-1}\tilde{\boldsymbol{\Gamma}}_1 + \tilde{\boldsymbol{\Gamma}}_2 - \mathbf{r}_{k+H+1}\right\|_{\mathbf{P}}^2 \le \varepsilon \tag{7.15c}$$

where

$$\tilde{\mathbf{\Phi}} = \mathbf{\Phi} + \tilde{\boldsymbol{\Gamma}}_1\mathbf{P} \in \mathbb{R}^{Hm\times Hm} \tag{7.16a}$$

$$\tilde{\boldsymbol{\psi}} = \boldsymbol{\psi} + 2\mathbf{P}\tilde{\boldsymbol{\Gamma}}_1(\tilde{\boldsymbol{\Gamma}}_2 - \mathbf{r}_{k+H+1}) \in \mathbb{R}^{Hm} \tag{7.16b}$$

$$\tilde{\mathbf{C}} = \mathbf{C} + \mathbf{P}(\tilde{\boldsymbol{\Gamma}}_2 - \mathbf{r}_{k+H+1})^2 \tag{7.16c}$$

Since $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{R}}$, $\tilde{\mathbf{T}}_z$ and $\tilde{\mathbf{T}}_u$ are positive definite, $\mathbf{\Phi}$ is also positive definite. In addition,

the Jacobian matrix $\mathbf{A}$ and terminal weighting matrix $\mathbf{P}$ are fully ranked. Hence $\tilde{\mathbf{\Phi}}$ is positive definite and the constrained QP problem (7.15) is strictly convex. The solution will therefore be unique and satisfies the KKT conditions.

Solving this QP problem could be computationally demanding if both constraints (7.15b) and (7.15c) are included in the optimization process. A more efficient way is to solve problem (7.15) without the terminal constraint (7.15c) using the active-set method described in Section 5.3. The validity of the optimal solution obtained can then be verified by checking the terminal constraint condition. Algorithm 8 describes this approach to solving (7.15).

7.1.4 Stability Analysis

The stability of the controller discussed above is guaranteed by the the following theorem.

Theorem 1. *Assuming that the condition (7.1) holds, and the feasibility at the initial state is guaranteed. The unknown nonlinear system (4.1) under the proposed GPMPC controller from (7.15) is asymptotically stable.*

Proof. The proof is based on the use of Lyapunov stability theory. The conditions (7.1a) to (7.1c) guarantee the satisfaction of terminal state constraints on the states beyond horizon H.

Let the cost function (7.2) with terminal cost (7.3) as a Lyapunov function given as follows,

$$\begin{aligned}\mathcal{V}^*(k) =& E\big[\mathcal{J}(\mathbf{x}_k^*, \mathbf{u}_{k-1}^*)\big] \\ =& \sum_{i=1}^{H} \left\{ \left\|\boldsymbol{\mu}_{k+i}^* - \mathbf{r}_{k+i}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}_{k+i-1}^*\right\|_{\mathbf{R}}^2 + \operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}^*) \right\} + \boldsymbol{\Psi}(\boldsymbol{\mu}_k^*, \boldsymbol{\Sigma}_k^*)\end{aligned} \tag{7.17}$$

where the terminal cost at sampling time k is given by,

$$\boldsymbol{\Psi}(\boldsymbol{\mu}_k^*, \boldsymbol{\Sigma}_k^*) = \left\|\boldsymbol{\mu}_{k+H+1}^* - \mathbf{r}_{k+H+1}\right\|_{\mathbf{P}}^2 + \operatorname{trace}(\mathbf{P}\boldsymbol{\Sigma}_{k+H+1}^*) \tag{7.18}$$

and the use of $*$ is a notation for the variables related to the corresponding optimal solutions. The Lyapunov function at sampling time $k+1$ can be subsequently obtained

by,

$$\begin{aligned}\mathcal{V}(k+1) =& \sum_{i=2}^{H}\{\left\|\boldsymbol{\mu}_{k+i}^{*}-\mathbf{r}_{k+i}\right\|_{\mathbf{Q}}^{2}+\left\|\mathbf{u}_{k+i-1}^{*}\right\|_{\mathbf{R}}^{2}+\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}^{*})\} \\ &+\left\|\boldsymbol{\mu}_{k+H+1}-\mathbf{r}_{k+H+1}\right\|_{\mathbf{Q}}^{2}+\left\|\mathbf{u}_{k+H}\right\|_{\mathbf{R}}^{2}+\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+H+1})+\boldsymbol{\Psi}(\boldsymbol{\mu}_{k+1},\boldsymbol{\Sigma}_{k+1}) \\ =& \underbrace{\sum_{i=1}^{H}\{\left\|\boldsymbol{\mu}_{k+i}^{*}-\mathbf{r}_{k+i}\right\|_{\mathbf{Q}}^{2}+\left\|\mathbf{u}_{k+i-1}^{*}\right\|_{\mathbf{R}}^{2}+\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}^{*})\}+\boldsymbol{\Psi}(\boldsymbol{\mu}_{k}^{*},\boldsymbol{\Sigma}_{k}^{*})}_{\mathcal{V}^{*}(k)} \\ &+\boldsymbol{\Psi}(\boldsymbol{\mu}_{k+1},\boldsymbol{\Sigma}_{k+1})-\boldsymbol{\Psi}(\boldsymbol{\mu}_{k}^{*},\boldsymbol{\Sigma}_{k}^{*}) \\ &+\left\|\boldsymbol{\mu}_{k+H+1}-\mathbf{r}_{k+H+1}\right\|_{\mathbf{Q}}^{2}+\left\|\mathbf{u}_{k+H}\right\|_{\mathbf{R}}^{2}+\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+H+1}) \\ &-\left\|\boldsymbol{\mu}_{k+1}^{*}-\mathbf{r}_{k+1}\right\|_{\mathbf{Q}}^{2}-\left\|\mathbf{u}_{k-1}^{*}\right\|_{\mathbf{R}}^{2}-\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1}^{*})\end{aligned} \tag{7.19}$$

Based on the condition (7.1d), it can be known that

$$\begin{aligned}&\boldsymbol{\Psi}(\boldsymbol{\mu}_{k+1},\boldsymbol{\Sigma}_{k+1})-\boldsymbol{\Psi}(\boldsymbol{\mu}_{k}^{*},\boldsymbol{\Sigma}_{k}^{*}) \\ &\qquad+\left\|\boldsymbol{\mu}_{k+H+1}-\mathbf{r}_{k+H+1}\right\|_{\mathbf{Q}}^{2}+\left\|\mathbf{u}_{k+H}\right\|_{\mathbf{R}}^{2}+\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+H+1})\leq 0\end{aligned} \tag{7.20}$$

Thus, the equality (7.19) becomes an inequality given by,

$$\mathcal{V}(k+1)\leq\mathcal{V}^{*}(k)-\left\|\boldsymbol{\mu}_{k+1}^{*}-\mathbf{r}_{k+1}\right\|_{\mathbf{Q}}^{2}-\left\|\mathbf{u}_{k-1}^{*}\right\|_{\mathbf{R}}^{2}-\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1}^{*}) \tag{7.21}$$

Due to the nature of optimization, it can be known that $\mathcal{V}^{*}(k+1)\leq\mathcal{V}(k+1)$. Thus,

$$\mathcal{V}^{*}(k+1)\leq\mathcal{V}^{*}(k)-\left\|\boldsymbol{\mu}_{k+1}^{*}-\mathbf{r}_{k+1}\right\|_{\mathbf{Q}}^{2}-\left\|\mathbf{u}_{k-1}^{*}\right\|_{\mathbf{R}}^{2}-\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1}^{*}) \tag{7.22}$$

where $\left\|\boldsymbol{\mu}_{k+1}^{*}-\mathbf{r}_{k+1}\right\|_{\mathbf{Q}}^{2}\geq 0$, $\left\|\mathbf{u}_{k-1}^{*}\right\|_{\mathbf{R}}^{2}\geq 0$ and $\operatorname{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1}^{*})\geq 0$. Therefore, the Lyapunov function is monotonically non-increasing. This implies that system states will asymptotically converge to the initial state and the unknown nonlinear system under proposed GPMPC controller is asymptotically stable. □

7.2 Conclusions

A stability guaranteed GPMPC algorithm is proposed in this chapter. The commonly used terminal cost and constraint are introduced into the MPC problem to stabilize the open-loop GPMPC1 or GPMPC2 controllers. This leads to a nonlinear, non-convex MPC optimization problem. We relaxed this problem to a convex one by using the extended GP

based local model. The resultant convex problem can be solved by using the active-set method, but is computationally demanding if the terminal constraint is always considered in the iterative optimization process. To address this issue, this optimization problem is solved without the consideration of terminal cost. The terminal cost is subsequently used to check the satisfaction of obtained solutions. In addition, based on the Lyapunov stability theory, we proved that the proposed GPMPC is guaranteed to be stable.

Chapter 8

Conclusions and Future Works

8.1 Conclusions

The research into MPC control of unknown systems that are modelled by GP is reported in this thesis. Its contributions include efficient GP modelling techniques, both unconstrained and constrained MPC methods, and the use of terminal cost and constraints to guarantee GPMPC stability. They are summarized in more detail below.

Chapter 3 addressed a key issue in GP modelling which is the learning of the model hyperparameters. An efficient hybrid PSO algorithm has been proposed. It makes use of multi-start techniques to improve global search ability. Unlike other typical PSO algorithms, gradients of the fitness function are computed and used to enhance local search ability. Experimental results in modelling both linear and nonlinear time-varying systems show that this hybrid algorithm is more able to avoid getting stuck in local optima compared with existing algorithms. It also exhibits faster convergence and produces more accurate models. In addition, the use of MSE of the outputs as the fitness function is explored. Results showed that the quality of the models produced is essentially the same as that obtained by using the traditional LL function as fitness function. The advantage of using MSE is that the quality of the intermediate solutions is directly observable during the optimization process. Thus a physically meaningful termination condition could more easily be set through MSE values.

Chapter 4 is concerned with the unconstrained MPC problem. The dynamics of the systems being controlled are assumed to be unknown and thus need to be learnt from empirical data and modelled using GP. A GP based MPC algorithm is proposed. The

novelty of the proposed method is the inclusion of GP variances in the cost function. This allows model uncertainties, which are reflected by the GP variances, to be directly accounted for in the optimization process. The resulting MPC problem is stochastic. By specifying a confidence level, it has been shown that the problem could be relaxed to a corresponding deterministic problem. The resultant optimization problem can be solved efficiently by the use of analytic gradients of the GP models w.r.t. the inputs and outputs. The effectiveness of this method is demonstrated through some trajectory tracking problems.

MPC of GP modelled systems with input and/or state constraints are tackled in Chapter 5. One of the main issues with existing algorithms for this problem is that GP variances are not included in the cost function. They are usually included as chance (or probabilistic) instead. Furthermore, the solutions to the resultant constrained MPC problem are computationally expensive. Two alternative formulations, called GPMPC1 and GPMPC2, are proposed. With GPMPC1, GP variances are treated as slack variables of the constraints on the GP mean and are directly included in the objective function as penalty terms. The resulting MPC optimization problem is nonlinear and non-convex. With the derivation of a GP based local dynamical model, an effective solution is proposed that makes use of the FP-SQP optimization. With GPMPC2, the original non-convex problem is relaxed to a convex one by using a proposed extended GP based local dynamical model. The resultant convex problem is effectively solved by using the active-set method. Simulation results showed that both GPMPC1 and GPMPC2 are equally good for solving the trajectory tracking problem for an unknown nonlinear system. However, GPMPC2 is superior in terms of computational efficiency.

The effectiveness of GPMPC2 is further tested in Chapter 6 by applying it to the trajectory tracking problem of a quadrotor. The dynamic system is divided into two subsystems, one for translational and the other for rotational motion. The trajectory tracking problem therefore becomes a position tracking problem for the translational subsystem and an attitude tracking problem for the rotational subsystem. Both subsystems are assumed to be unknown and are represented by GP models which are learnt from empirical data. The two tracking problems are solved sequentially using GPMPC2 in a hierarchical scheme. Results showed that GPMPC2 is effective in tracking two different non-trivial trajectories.

The main issue with GPMPC1 and GPMPC2 is that they are developed for open-loop control. Thus there is no guarantee for their stability. In Chapter 7, a closed-loop

stability guaranteed GPMPC is proposed to address this issue. The original GPMPC is reformulated with terminal cost and terminal constraints. The non-convex optimization problem is relaxed to a convex one using the extended GP based local dynamical model in a similar way to GPMPC2. The resultant convex problem can first be solved without the terminal constraints. The solutions obtained are then check to see if they satisfy these constraints. This provides an efficient way to solve this MPC problem. The obtained GPMPC controller is mathematically proven to be asymptotically stable.

8.2 Future Works

The methods and results presented in this thesis help to advance GP based MPC, making it more computationally attractive as a control method for complex unknown systems. However, the GP models are learnt offline and are not online updated. As more data are collected during the operation of the system, the system model and hence the GPMPC control could be improved if real-time online updating is incorporated. Techniques to reduce the computational time required by online GP model updating will need to be developed.

Another important issue in the proposed approaches is that the performance could not guaranteed in the control tasks that are not included in the training process. A GPMPC trained for the "Elliptical" trajectory in Section 6.3, for instance, may not perform well for a "Lorenz" trajectory. This requires better generalization of the GP model which is embedded in deep latent features of the system. Some form of deep learning techniques may be used to achieve this goal, such as deep Gaussian processes proposed in [146] and recurrent Gaussian processes proposed in [147].

Appendix A

Mean and Variance at uncertain inputs

Through using the law of iterated expectations, the predictive mean $m(\tilde{\mathbf{x}}_k^*)$ is obtained as

$$\begin{aligned} m(\tilde{\mathbf{x}}_k^*) &= E_{\tilde{\mathbf{x}}_k^*}\Big[E_f\big[\Delta \mathbf{x}_k^*\big]\Big] \\ &= \int \big(\mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{y}\big)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\ &= \big(\mathbf{K}_\sigma^{-1}\mathbf{y}\big)^T \int \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}})\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\ &= \boldsymbol{\Omega}^T \mathbf{w}_k \end{aligned} \tag{A.1}$$

where $\boldsymbol{\Omega} = \mathbf{K}_\sigma^{-1}\mathbf{y} \in \mathbb{R}^{D\times n}$, and $\mathbf{w}_k = [w_k^1, \cdots, w_k^D]^T \in \mathbb{R}^{D\times 1}$ with the entries computed by

$$\begin{aligned} w_k^d &= \int \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}}_d)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\ &= \sigma_s^2\big|\tilde{\boldsymbol{\Sigma}}_k\boldsymbol{\Lambda}^{-1} + \mathbf{I}\big|^{-\frac{1}{2}} \exp\big(-\frac{1}{2}\boldsymbol{\chi}_d^T(\tilde{\boldsymbol{\Sigma}}_k + \boldsymbol{\Lambda})^{-1}\boldsymbol{\chi}_d\big) \end{aligned} \tag{A.2}$$

where $d = 1, \cdots, D$, and $\boldsymbol{\chi}_d = \tilde{\mathbf{X}}_d - \tilde{\boldsymbol{\mu}}_k$.

$$
\begin{aligned}
\left[\sigma^2(\tilde{\mathbf{x}}_k^*)\right]_{ab} =& E_{\tilde{\mathbf{x}}_k^*}\Big[\, Var_f\left[\Delta \mathbf{x}_k^*\right]\Big] + E_{\tilde{\mathbf{x}}_k^*}\Big[(\Delta \mathbf{x}_k^*)^2\Big] - E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_k^*\Big]^2 \\
=& \int \Big(\mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}_k) - \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k)\Big)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\
&+ \int \Big(\mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{y}\Big)^2 \mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\
&- \Big((\mathbf{K}_\sigma^{-1}\mathbf{y})^T \int \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k\Big)^2 \\
=& \int \Big(\mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}_k) - \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k)\Big)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\
&+ \int \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\boldsymbol{\Omega}\boldsymbol{\Omega}^T\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k - \Big(\boldsymbol{\Omega}^T\mathbf{w}_k\Big)^2 \\
=& \sigma_s^2 - \mathrm{trace}\Big(\mathbf{K}_\sigma^{-1}\int \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k\Big) + \sigma_n^2 \\
&+ \boldsymbol{\Omega}^T\Big(\int \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k\Big)\boldsymbol{\Omega} - \Big(\boldsymbol{\Omega}^T\mathbf{w}_k\Big)^2 \\
=& \sigma_s^2 - \mathrm{trace}\Big(\mathbf{K}_\sigma^{-1}\boldsymbol{\Xi}_k\Big) + \sigma_n^2 + \boldsymbol{\Omega}^T\boldsymbol{\Xi}_k\boldsymbol{\Omega} - \Big(\boldsymbol{\Omega}^T\mathbf{w}_k\Big)^2
\end{aligned}
\tag{A.3}
$$

In addition, the predictive variance $\sigma^2(\tilde{\mathbf{x}}_k^*)$ in (4.9) leads to a matrix with the entries specified by (A.3) when $a = b$, where the $\boldsymbol{\Xi}_k = \int \mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{X}})\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k$ has the entries

$$
\boldsymbol{\Xi}_k^{ij} = |\mathbf{M}_\Lambda|^{-\frac{1}{2}}\mathbf{K}(\tilde{\mathbf{x}}_i, \tilde{\boldsymbol{\mu}}_k)\mathbf{K}(\tilde{\mathbf{x}}_j, \tilde{\boldsymbol{\mu}}_k)\mathbf{exp}\Big(\frac{1}{2}\mathbf{M}_\chi^T\mathbf{P}^T\mathbf{M}_\chi\Big) \tag{A.4}
$$

where $\mathbf{M}_\Lambda = \tilde{\boldsymbol{\Sigma}}_k(\boldsymbol{\Lambda}_i^{-1} + \boldsymbol{\Lambda}_j^{-1}) + \boldsymbol{I}$, $\mathbf{M}_\chi = \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\chi}_i + \boldsymbol{\Lambda}_j^{-1}\boldsymbol{\chi}_j$ and $\mathbf{P} = \boldsymbol{\Lambda}_i^{-1} + \boldsymbol{\Lambda}_j^{-1} + \tilde{\boldsymbol{\Sigma}}^{-1}$.

When $a \neq b$, the entries of $\sigma^2(\tilde{\mathbf{x}}_k^*)$ is obtained by

$$
\left[\sigma^2(\tilde{\mathbf{x}}_k^*)\right]_{ab} = E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_{k,a}^* \Delta \mathbf{x}_{k,b}^*\Big] - E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_{k,a}^*\Big] E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_{k,b}^*\Big] \tag{A.5}
$$

where $E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_{k,a}^*\Big] = m(\tilde{\mathbf{x}}_{k,a}^*)$ and $E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_{k,b}^*\Big] = m(\tilde{\mathbf{x}}_{k,b}^*)$ computed by (A.1). And,

$$
\begin{aligned}
E_{\tilde{\mathbf{x}}_k^*}\Big[\Delta \mathbf{x}_{k,a}^* \Delta \mathbf{x}_{k,b}^*\Big] &= E_{\tilde{\mathbf{x}}_k^*}\Big[E_f\left[\Delta \mathbf{x}_{k,a}^*\right] E_f\left[\Delta \mathbf{x}_{k,b}^*\right]\Big] \\
&= \int \mathbf{K}(\tilde{\mathbf{x}}_{k,a}, \tilde{\mathbf{X}})\boldsymbol{\Omega}_a, \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_{k,b})\boldsymbol{\Omega}_b\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\
&= \boldsymbol{\Omega}_a^T\Big(\int \mathbf{K}(\tilde{\mathbf{x}}_{k,a}, \tilde{\mathbf{X}})\mathbf{K}(\tilde{\mathbf{x}}_{k,b}, \tilde{\mathbf{X}})\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k\Big)\boldsymbol{\Omega}_b \\
&= \boldsymbol{\Omega}_a^T\boldsymbol{\Xi}_k\boldsymbol{\Omega}_b
\end{aligned}
\tag{A.6}
$$

Appendix B

Cross-covariance between GP States and Outputs

It is not easy to compute the cross-covariance $Cov[\mathbf{x}_k, \Delta\mathbf{x}_k]$ between GP state and output. However, it is known that the cross-covariance between state-control and output is defined as

$$Cov\big[\tilde{\mathbf{x}}_k, \Delta\mathbf{x}_k\big] = \begin{bmatrix} Cov[\mathbf{x}_k, \Delta\mathbf{x}_k] \\ \\ Cov[\mathbf{u}_k, \Delta\mathbf{x}_k] \end{bmatrix} \in \mathbb{R}^{(m+n)\times n} \tag{B.1}$$

Thus, $Cov[\mathbf{x}_k, \Delta\mathbf{x}_k]$ can be obtained by partitioning a $n\times n$ sub-matrix from $Cov\big[\tilde{\mathbf{x}}_k, \Delta\mathbf{x}_k\big]$. In particularly,

$$\begin{aligned} Cov\big[\tilde{\mathbf{x}}_k, \Delta\mathbf{x}_k\big] &= E_{\tilde{\mathbf{x}}_k}\big[\Delta\mathbf{x}_k\tilde{\mathbf{x}}_k\big] - m(\tilde{\mathbf{x}}_k^*)E_f\big[\tilde{\mathbf{x}}_k\big] \\ &= E_{\tilde{\mathbf{x}}_k}\big[\Delta\mathbf{x}_k\tilde{\mathbf{x}}_k\big] - \boldsymbol{\Omega}^T\mathbf{w}_k\boldsymbol{\mu}_k \end{aligned} \tag{B.2}$$

where for each state dimension $a = 1, \cdots, n$, $E_{\tilde{\mathbf{x}}_k}\big[\Delta\mathbf{x}_k^a\tilde{\mathbf{x}}_k\big]$ can be computed as

$$\begin{aligned} E_{\tilde{\mathbf{x}}_k}\big[\Delta\mathbf{x}_k^a\mathbf{x}_k\big] &= E_{\tilde{\mathbf{x}}_k}\big[\mathbf{x}_k E_f[\Delta\mathbf{x}_k^a]\big] \\ &= \int \tilde{\mathbf{x}}_k\Big(\sum_{d=1}^{D}\boldsymbol{\Omega}_{a,d}\mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}_d^a)\Big)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\ &= \sum_{d=1}^{D}\boldsymbol{\Omega}_{a,d}\int \tilde{\mathbf{x}}_k\mathbf{K}(\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}_d^a)\mathcal{N}(\tilde{\mathbf{x}}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)d\tilde{\mathbf{x}}_k \\ &= \sum_{d=1}^{D}\boldsymbol{\Omega}_{a,d}\mathbf{w}_{k,a}^d\tilde{\boldsymbol{\Sigma}}(\tilde{\boldsymbol{\Sigma}} + \boldsymbol{\Lambda}_a)^{-1}\boldsymbol{\chi}_d \end{aligned} \tag{B.3}$$

Appendix C

GP Derivatives

The $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k}$, $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\Sigma}_k}$, $\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\mu}_k}$, $\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\Sigma}_k}$, $\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k}$ and $\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \mathbf{u}_k}$ can be obtained analytically by using (4.7)-(4.10) and (A.1)-(A.5).

$$
\begin{aligned}
\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k} &= \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k} \\
\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\Sigma}_k} &= \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k} \\
\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\mu}_k} &= \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k} \\
\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\Sigma}_k} &= \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k} \\
\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k} &= \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k} \\
\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \mathbf{u}_k} &= \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k}
\end{aligned}
\tag{C.1}
$$

For each state dimension $a = 1, \cdots, n$, $\frac{\partial \boldsymbol{\mu}_{k+1}^a}{\partial \tilde{\boldsymbol{\mu}}_k}$, $\frac{\partial \boldsymbol{\mu}_{k+1}^a}{\partial \tilde{\boldsymbol{\Sigma}}_k}$, $\frac{\partial \boldsymbol{\Sigma}_{k+1}^{ab}}{\partial \tilde{\boldsymbol{\mu}}_k}$ and $\frac{\partial \boldsymbol{\Sigma}_{k+1}^{ab}}{\partial \tilde{\boldsymbol{\Sigma}}_k}$ are obtained as

$$
\begin{aligned}
\frac{\partial \boldsymbol{\mu}_{k+1}^a}{\partial \tilde{\boldsymbol{\mu}}_k} =& \sum_{d=1}^{D} \boldsymbol{\Omega}_{a,d} \mathbf{w}_{k,a}^d \boldsymbol{\chi}_d^T \tilde{\boldsymbol{\Sigma}} (\tilde{\boldsymbol{\Sigma}} + \boldsymbol{\Lambda}_a)^{-1} \\
\frac{\partial \boldsymbol{\mu}_{k+1}^a}{\partial \tilde{\boldsymbol{\Sigma}}_k} =& \sum_{d=1}^{D} \boldsymbol{\Omega}_{a,d} \mathbf{w}_{k,a}^d \Big(-\frac{1}{2} ((\boldsymbol{\Lambda}_a^{-1} \tilde{\boldsymbol{\Sigma}}_k + \mathbf{I})^{-1} \boldsymbol{\Lambda}_a^T)^{-1} - \frac{1}{2} \boldsymbol{\chi}_d^T (-1) (\boldsymbol{\Lambda}_a + \tilde{\boldsymbol{\Sigma}}_k)^{-2} \boldsymbol{\chi}_d \Big) \\
\frac{\partial \boldsymbol{\Sigma}_{k+1}^{ab}}{\partial \tilde{\boldsymbol{\mu}}_k} =& \frac{\partial \sigma^2(\tilde{\mathbf{x}}_k)}{\partial \tilde{\boldsymbol{\mu}}_k} + \frac{\partial Cov[\mathbf{x}_{k,a}, \Delta \mathbf{x}_{k,b}]}{\partial \tilde{\boldsymbol{\mu}}_k} + \frac{\partial Cov[\mathbf{x}_{k,b}, \Delta \mathbf{x}_{k,a}]}{\partial \tilde{\boldsymbol{\mu}}_k} \\
\frac{\partial \boldsymbol{\Sigma}_{k+1}^{ab}}{\partial \tilde{\boldsymbol{\Sigma}}_k} =& \frac{\partial \sigma^2(\tilde{\mathbf{x}}_k)}{\partial \tilde{\boldsymbol{\Sigma}}_k} + \frac{\partial Cov[\mathbf{x}_{k,a}, \Delta \mathbf{x}_{k,b}]}{\partial \tilde{\boldsymbol{\Sigma}}_k} + \frac{\partial Cov[\mathbf{x}_{k,b}, \Delta \mathbf{x}_{k,a}]}{\partial \tilde{\boldsymbol{\Sigma}}_k}
\end{aligned}
\tag{C.2}
$$

where

$$
\begin{aligned}
\frac{\partial \sigma^2(\tilde{\mathbf{x}}_k)}{\partial \tilde{\boldsymbol{\mu}}_k} =& \boldsymbol{\Omega}_a^T \Big(\frac{\partial \boldsymbol{\Xi}_k}{\partial \tilde{\boldsymbol{\mu}}}_k - \mathbf{w}_{k,b}^T \frac{\partial \mathbf{w}_{k,a}}{\partial \tilde{\boldsymbol{\mu}}_k} - \mathbf{w}_{k,a}^T \frac{\partial \mathbf{w}_{k,b}}{\partial \tilde{\boldsymbol{\mu}}_k} \Big) \boldsymbol{\Omega}_b - \mathbf{K}_\sigma^{-1} \frac{\partial \boldsymbol{\Xi}}{\partial \tilde{\boldsymbol{\mu}}}_k \\
\frac{\partial \sigma^2(\tilde{\mathbf{x}}_k)}{\partial \tilde{\boldsymbol{\Sigma}}_k} =& \boldsymbol{\Omega}_a^T \Big(\frac{\partial \boldsymbol{\Xi}_k}{\partial \tilde{\boldsymbol{\Sigma}}}_k - \mathbf{w}_{k,b}^T \frac{\partial \mathbf{w}_{k,a}}{\partial \tilde{\boldsymbol{\Sigma}}_k} - \mathbf{w}_{k,a}^T \frac{\partial \mathbf{w}_{k,b}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \Big) \boldsymbol{\Omega}_b - \mathbf{K}_\sigma^{-1} \frac{\partial \boldsymbol{\Xi}}{\partial \tilde{\boldsymbol{\Sigma}}}_k
\end{aligned}
\tag{C.3}
$$

where the entries for $\frac{\partial \boldsymbol{\Xi}_k}{\partial \tilde{\boldsymbol{\mu}}}$ and $\frac{\partial \boldsymbol{\Xi}_k}{\partial \tilde{\boldsymbol{\Sigma}}}$ are specified by

$$
\begin{aligned}
\frac{\partial \boldsymbol{\Xi}_k^{ab}}{\partial \tilde{\boldsymbol{\mu}}} =& \boldsymbol{\Xi}_k^{ab} \Big(\frac{\boldsymbol{\Lambda}_b}{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b} \tilde{\mathbf{x}}_{k,a} + \frac{\boldsymbol{\Lambda}_a}{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b} \tilde{\mathbf{x}}_{k,b} - \tilde{\boldsymbol{\mu}}_k \Big)^T \Big(\frac{1}{\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}} + \tilde{\boldsymbol{\Sigma}}_k \Big)^{-1} \\
\frac{\partial \boldsymbol{\Xi}_k^{ab}}{\partial \tilde{\boldsymbol{\Sigma}}} =& -\frac{1}{2} \boldsymbol{\Xi}_k^{ab} \Big(\frac{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b}{\boldsymbol{\Lambda}_a \boldsymbol{\Lambda}_b} \tilde{\boldsymbol{\Sigma}}_k + \mathbf{I} \Big)^{-1} \Big(\frac{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b}{\boldsymbol{\Lambda}_a \boldsymbol{\Lambda}_b} \Big) \\
& - \Big(\frac{\boldsymbol{\Lambda}_b}{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b} \tilde{\mathbf{x}}_{k,a} + \frac{\boldsymbol{\Lambda}_a}{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b} \tilde{\mathbf{x}}_{k,b} - \tilde{\boldsymbol{\mu}}_k \Big)^T \\
& \Big(\frac{1}{\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}} + \tilde{\boldsymbol{\Sigma}}_k \Big)^{-1} \Big(\frac{\boldsymbol{\Lambda}_b}{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b} \tilde{\mathbf{x}}_{k,a} + \frac{\boldsymbol{\Lambda}_a}{\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b} \tilde{\mathbf{x}}_{k,b} - \tilde{\boldsymbol{\mu}}_k \Big)
\end{aligned}
\tag{C.4}
$$

Also,

$$
\begin{aligned}
\frac{\partial Cov[\mathbf{x}_{k,a}, \Delta \mathbf{x}_{k,b}]}{\partial \tilde{\boldsymbol{\mu}}_k} =& \frac{\tilde{\boldsymbol{\Sigma}}_k}{\tilde{\boldsymbol{\Sigma}}_k + \boldsymbol{\Lambda}} \sum_{d=1}^{n+m} \boldsymbol{\Omega}_d \Big(\boldsymbol{\chi}_d \frac{\partial \mathbf{w}_{k,d}}{\partial \tilde{\boldsymbol{\mu}}_k} + \mathbf{w}_{k,d}^T \Big) \\
\frac{\partial Cov[\mathbf{x}_{k,a}, \Delta \mathbf{x}_{k,b}]}{\partial \tilde{\boldsymbol{\Sigma}}_k} =& \frac{\tilde{\boldsymbol{\Sigma}}_k + \boldsymbol{\Lambda} + \mathbf{I}}{(\tilde{\boldsymbol{\Sigma}}_k + \boldsymbol{\Lambda})^2} \sum_{d=1}^{n+m} \boldsymbol{\Omega}_d \mathbf{w}_{k,d}^T + \frac{\tilde{\boldsymbol{\Sigma}}_k}{\tilde{\boldsymbol{\Sigma}}_k + \boldsymbol{\Lambda}} \sum_{d=1}^{n+m} \boldsymbol{\Omega}_d \boldsymbol{\chi}_d \frac{\partial \mathbf{w}_{k,d}}{\partial \tilde{\boldsymbol{\Sigma}}_k}
\end{aligned}
\tag{C.5}
$$

In addition, the rests $\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k}$, $\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k}$, $\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k}$, $\frac{\partial \tilde{\boldsymbol{\Sigma}}_{kk}}{\partial \boldsymbol{\mu}_k}$, $\frac{\partial \tilde{\boldsymbol{\Sigma}}_{kk}}{\partial \mathbf{u}_k}$ and $\frac{\partial \tilde{\boldsymbol{\Sigma}}_{kk}}{\partial \boldsymbol{\Sigma}_k}$ are easily obtained based on (4.7).

Appendix D

Cost Function using GP

We first rewrite the (4.14) as follow

$$
\begin{aligned}
E[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})] &= E\Big[\sum_{i=1}^{H}\Big\{\big\|\mathbf{x}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2\Big\}\Big] \\
&= \sum_{i=1}^{H}\Bigg\{\underbrace{E\Big[(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T\mathbf{Q}(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})\Big]}_{\text{probabilistic term}} + \underbrace{\mathbf{u}_{k+i-1}^T\mathbf{R}\mathbf{u}_{k+i-1}}_{\text{determinisitc term}}\Bigg\}
\end{aligned}
\tag{D.1}
$$

Let Q_{ab} as the entries of $\mathbf{Q}$ thus $Q_{ab} = [\mathbf{Q}]_{ab}$ and ε_{ab} as the entries of $\mathbf{\Sigma}_k$ thus $\varepsilon_{ab} = [\mathbf{\Sigma}_k]_{ab}$, the "probabilistic term" can be further derived to

$$
\begin{aligned}
&E\Big[(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T\mathbf{Q}(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})\Big] \\
&= E\Big[\sum_{a=1}^{N}\sum_{b=1}^{N} Q_{ab}(\mathbf{x}_{k+i,a} - \mathbf{r}_{k+i,a})(\mathbf{x}_{k+i,b} - \mathbf{r}_{k+i,b})\Big] \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N} Q_{ab}E\Big[(\mathbf{x}_{k+i,a} - \mathbf{r}_{k+i,a})(\mathbf{x}_{k+i,b} - \mathbf{r}_{k+i,b})\Big] \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N} Q_{ab}\Bigg\{E\big[\mathbf{x}_{k+i,a} - \mathbf{r}_{k+i,a}\big]E\big[\mathbf{x}_{k+i,b} - \mathbf{r}_{k+i,b}\big] \\
&\quad + \underbrace{\mathrm{Cov}\Big((\mathbf{x}_{k+i,a} - \mathbf{r}_{k+i,a}), (\mathbf{x}_{k+i,b} - \mathbf{r}_{k+i,b})\Big)}_{\varepsilon_{ab}}\Bigg\} \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N} Q_{ab}\Big\{(\boldsymbol{\mu}_{k+i,a} - \mathbf{r}_{k+i,a})(\boldsymbol{\mu}_{k+i,a} - \mathbf{r}_{k+i,a}) + \varepsilon_{ab}\Big\}
\end{aligned}
\tag{D.2}
$$

where

$$\sum_{a=1}^{N}\sum_{b=1}^{N} Q_{ab}(\boldsymbol{\mu}_{k+i,a} - \mathbf{r}_{k+i,a})(\boldsymbol{\mu}_{k+i,a} - \mathbf{r}_{k+i,a}) = (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q}(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}) \tag{D.3}$$

and

$$\sum_{a=1}^{N}\sum_{b=1}^{N} Q_{ab}\varepsilon_{ab} = trace(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}) \tag{D.4}$$

Therefore, (D.1) can be obtained by

$$\begin{aligned} E[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})] &= \sum_{i=1}^{H} \Big\{ (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i})^T \mathbf{Q}(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}) + \mathbf{u}_{k+i-1}^T \mathbf{R}\mathbf{u}_{k+i-1} \\ &\quad + trace(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}) \Big\} \\ &= \sum_{i=1}^{H} \Big\{ \big\|\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}\big\|_{\mathbf{Q}}^2 + \big\|\mathbf{u}_{k+i-1}\big\|_{\mathbf{R}}^2 + trace\big(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}\big) \Big\} \end{aligned} \tag{D.5}$$

Appendix E

Karush-Kuhn-Tucker (KKT) Conditions for Convex Optimization Problem

Considering the following constrained optimization problem with n-dimensional variables, m_1-dimensional equality and m_2-dimensional inequality constraints,

$$\min_{x\in\mathbb{R}^n} f(x) = e^T x + \frac{1}{2}x^T H x \tag{E.1a}$$

$$\text{s.t.} Ax = b \tag{E.1b}$$

$$Cx \geq d \tag{E.1c}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$ is the quadratic objective function, $H \in \mathbb{R}^{n\times n}$ denotes a symmetric positive definite Hessian matrix, $e \in \mathbb{R}^n$ is a constant vector. In addition, $A \in \mathbb{R}^{m_1\times n}$ and $b \in \mathbb{R}^{m_1}$ represent constraint matrix and constant vector in the equality constraints (E.1b), while $C \in \mathbb{R}^{m_2\times n}$ and $d \in \mathbb{R}^{m_2}$ are constraint matrix and constant vector in the inequality constraints (E.1c).

Let $x^* = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} f(x)$ be a local optimum of problem (E.1), then there exist a unique set of constants $\boldsymbol{\mu} = \{\mu_1, \cdots, \mu_{m_1}\}$ and $\boldsymbol{\lambda} = \{\lambda_1, \cdots, \lambda_{m_2}\}$ such that the following Karush-Kahn-Tucker (KKT) conditions are satisfied,

$$x^{*T} H^T + e^T - \boldsymbol{\mu} A - \boldsymbol{\lambda} C = \boldsymbol{\mu} b + \boldsymbol{\lambda} d \tag{E.2a}$$

$$Ax^* = b \tag{E.2b}$$

$$Ax^* = b \tag{E.2c}$$

$$Cx^* = d \tag{E.2d}$$

$$\forall \mu_i \geq 0, i = 1, \cdots, m_1 \tag{E.2e}$$

In the constrained optimization problem, (E.2) are necessary conditions for a local optimum.

Appendix F

List of Publications

- G. Cao, E. M.-K. Lai, and F. Alam, "Particle swarm optimization for convolved Gaussian process models," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 6-11 July 2014, pp.1573-1578.
- G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of unmanned quadrotors," in *International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 28-30 April, 2016.
- G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process based model predictive control for linear time varying systems," in *International Workshop on Advanced Motion Control (AMC Workshop)*. IEEE, 22-24 April, 2016.
- G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of unknown nonlinear systems," *IET Control Theory & Applications*. Accepted for publication, 2017.
- G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of an unmanned quadrotor helicopter," *Journal of Intelligent and Robotic Systems*. Under review, 2016.
- G. Cao, E. M.-K. Lai, and F. Alam, "Enhanced particle swarm optimization algorithms for multiple-input multiple-output system modelling using convolved Gaussian process models," *International Journal of Intelligent Systems Technologies and Applications*. Under review, 2016.

References

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[2] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear H∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[3] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances," *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[4] C.-S. Chiu, "TS fuzzy maximum power point tracking control of solar power generation systems," *IEEE Transactions on Energy Conversion*, vol. 25, no. 4, pp. 1123–1132, 2010.

[5] C. Quek, M. Pasquier, and B. Lim, "A novel self-organizing fuzzy rule-based system for modelling traffic flow behaviour," *Expert Systems with applications*, vol. 36, no. 10, pp. 12 167–12 178, 2009.

[6] F. Han, G. Feng, Y. Wang, and F. Zhou, "Fuzzy modeling and control for a nonlinear quadrotor under network environment," in *IEEE 4th Annual International Conference on Cyber Technology in Automation Control, and Intelligent Systems (CYBER)*. IEEE, 2014, pp. 395–400.

[7] J. Dunfied, M. Tarbouchi, and G. Labonte, "Neural network based control of a four rotor helicopter," in *IEEE Proceedings of International Conference on Industrial Technology (ICIT)*, vol. 3. IEEE, 2004, pp. 1543–1548.

[8] H. Voos, "Nonlinear and neural network-based control of a small four-rotor aerial robot," in *2007 IEEE/ASME international conference on Advanced intelligent mechatronics*. IEEE, 2007, pp. 1–6.

[9] C. Nicol, C. Macnab, and A. Ramirez-Serrano, "Robust neural network control of a quadrotor helicopter," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2008, pp. 1233–1237.

[10] A. Das, F. Lewis, and K. Subbarao, "Neural network based robust backstepping control approach for quadrotors," in *Porceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008, pp. 1–17.

[11] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor UAV using neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 50–66, 2010.

[12] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *IEEE Proceedings of International Joint Conference on Neural Networks (IJCNN)*, vol. 1. IEEE, 1994, pp. 55–60.

[13] G. Paass, "Assessing and improving neural network predictions by the bootstrap algorithm," in *Advances in Neural Information Processing Systems (NIPS)*. Morgan Kaufmann Publishers, 1993, pp. 196–196.

[14] J. Franke and M. H. Neumann, "Bootstrapping neural networks," *Neural computation*, vol. 12, no. 8, pp. 1929–1949, 2000.

[15] F. Aires, C. Prigent, and W. B. Rossow, "Neural network uncertainty assessment using Bayesian statistics: A remote sensing application," *Neural Computation*, vol. 16, no. 11, pp. 2415–2458, 2004.

[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 1613–1622.

[17] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 1 2006.

[18] P. Boyle and M. Frean, "Dependent gaussian processes," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2005, pp. 217–224.

[19] M. Alvarez and N. D. Lawrence, "Sparse convolved Gaussian processes for multi-output regression," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 57–64.

[20] G. Cao, E. M.-K. Lai, and F. Alam, "Particle swarm optimization for convolved Gaussian process models," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 6-11 July 2014, pp. 1573–1578.

[21] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *American Control Conference*, vol. 3. IEEE, 2004, pp. 2214–2219.

[22] B. Likar and J. Kocijan, "Predictive control of a gas–liquid separation plant based on a Gaussian process model," *Computers & Chemical Engineering*, vol. 31, no. 3, pp. 142–152, 2007.

[23] A. Grancharova, T. A. Johansen, and P. Tøndel, "Computational aspects of approximate explicit nonlinear model predictive control," in *Proceedings of the International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control*. Springer, 2007, pp. 181–192.

[24] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine, Accepted*, 2016.

[25] E. D. Klenske, M. N. Zeilinger, B. Scholkopf, and P. Hennig, "Gaussian process-based predictive control for periodic error correction," *IEEE Transactions on Control Systems Technology*, 2015.

[26] Z. Yan and J. Wang, "Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 457–469, 2014.

[27] A. Grancharova and T. A. Johansen, "Explicit approximate model predictive control of constrained nonlinear systems with quantized input," in *Proceedings of the International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Contro*. Springer, 2008, pp. 371–380.

[28] F. Zhu, C. Xu, and G. Dui, "Particle swarm hybridize with Gaussian process regression for displacement prediction," in *IEEE Proceedings of International Conference on Bio-Inspired Computing: Theories and Applications*. IEEE, 2010, pp. 522–525.

[29] D. Petelin and J. Kocijan, "Control system with evolving Gaussian process models," in *IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 2011, pp. 178–184.

[30] P. Boyle, "Gaussian processes for regression and optimisation," Ph.D. dissertation, Victoria University of Wellington, 2007.

[31] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 1.

[32] M. N. Gibbs, "Bayesian Gaussian processes for regression and classification," Ph.D. dissertation, University of Cambridge, 1997.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 1, p. 213, 2002.

[34] A. E. Ruano, *Intelligent Control Systems using Computational Intelligence Techniques*. IET, 2005, vol. 70.

[35] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*. CRC press, 2003.

[36] R. A. Choudrey, "Variational methods for Bayesian independent component analysis," Ph.D. dissertation, University of Oxford, 2002.

[37] C. E. Rasmussen and Z. Ghahramani, "Occam's Razor," in *Advances in Neural Information Processing Systems (NIPS)*. MIT, 2001, pp. 294–300.

[38] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT press, 1949.

[39] M. Hitsuda, "Representation of Gaussian processes equivalent to Wiener process," *Osaka Journal of Mathematics*, vol. 5, no. 2, pp. 299–312, 1968.

[40] R. M. Neal, "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification," Department of Computer Science, University of Toronto Toronto, Ontario, Canada, Tech. Rep., 1997.

[41] A. O'Hagan and J. Kingman, "Curve fitting and optimal design for prediction," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–42, 1978.

[42] R. M. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, University of Toronto, 1995.

[43] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 1996.

[44] C. E. Rasmussen, "Evaluation of Gaussian processes and other methods for non-linear regression," Ph.D. dissertation, University of Toronto, 1996.

[45] A. G. Wilson, D. A. Knowles, and Z. Ghahramani, "Gaussian process regression networks," in *IEEE Proceedings of International Conference on Machine Learning*, 2011.

[46] T. V. Nguyen and E. V. Bonilla, "Efficient variational inference for Gaussian process regression networks," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 2013, pp. 472–480.

[47] T. Chen, J. Morris, and E. Martin, "Gaussian process regression for multivariate spectroscopic calibration," *Chemometrics and Intelligent Laboratory Systems*, vol. 87, no. 1, pp. 59–71, 2007.

[48] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task Gaussian process prediction," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2007, pp. 153–160.

[49] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings, "Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes," in *IEEE Proceedings of International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 2008, pp. 109–120.

[50] P. Goovaerts, *Geostatistics for Natural Resources Evaluation*. Oxford university press, 1997.

[51] P. Smaragdis, "Blind separation of convolved mixtures in the frequency domain," *Neurocomputing*, vol. 22, no. 1, pp. 21–34, 1998.

[52] A. Liutkus, R. Badeau, and G. Richard, "Gaussian processes for underdetermined source separation," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3155–3167, 2011.

[53] M. Goulard and M. Voltz, "Linear coregionalization model: Tools for estimation and choice of cross-variogram matrix," *Mathematical Geology*, vol. 24, no. 3, pp. 269–286, 1992.

[54] J. M. Ver Hoef and R. P. Barry, "Constructing and fitting models for cokriging and multivariable spatial prediction," *Journal of Statistical Planning and Inference*, vol. 69, no. 2, pp. 275–294, 1998.

[55] D. Higdon, "Space and space-time modeling using process convolutions," in *Quantitative methods for current environmental issues*. Springer, 2002, pp. 37–56.

[56] S. Haykin, *Communication Systems*. Wiley Publishing, 2009.

[57] P. Hemakumara and S. Sukkarieh, "Non-parametric UAV system identification with dependent Gaussian processes," in *IEEE Proceedings of International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 4435–4441.

[58] ——, "UAV parameter estimation with multi-output local and global Gaussian process approximations," in *IEEE Proceedings of International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 5402–5408.

[59] M. A. Alvarez, "Convolved Gaussian process priors for multivariate regression with applications to dynamical systems," Ph.D. dissertation, University of Manchester, 2011.

[60] L. Paninski, "Log-concavity results on Gaussian process methods for supervised and unsupervised learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2004, pp. 1025–1032.

[61] A. v. d. Vaart and H. v. Zanten, "Information rates of nonparametric Gaussian process methods," *Journal of Machine Learning Research*, vol. 12, no. Jun, pp. 2095–2119, 2011.

[62] Y. Wang and B. Chaib-Draa, "A KNN based kalman filter Gaussian process regression," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, (IJCAI)*. AAAI Press, 2013, pp. 1771–1777.

[63] G. Gregorcic and G. Lightbody, "Gaussian processes for modelling of dynamic nonlinear systems," in *Proceedings of the Irish Signals and Systems Conference*, 2002, pp. 141–147.

[64] G. Gregorčič and G. Lightbody, "Gaussian process approach for modelling of nonlinear systems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 4, pp. 522–533, 2009.

[65] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 16, no. 3, 2004, pp. 329–336.

[66] J. Wang, A. Hertzmann, and D. M. Blei, "Gaussian process dynamical models," in *Advances in Neural Information Processing Systems (NIPS)*, 2005, pp. 1441–1448.

[67] W. Ni, S. K. Tan, W. J. Ng, and S. D. Brown, "Moving-window GPR for nonlinear dynamic system modeling with dual updating and dual preprocessing," *Industrial & Engineering Chemistry Research*, vol. 51, no. 18, pp. 6416–6428, 2012.

[68] R. Murray-Smith and A. Girard, "Gaussian process priors with ARMA noise models," in *Proceedings of the Irish Signals and Systems Conference*, 2001, pp. 147–152.

[69] J. Kocijan, B. Likar, B. Banko, A. Girard, R. Murray-Smith, and C. E. Rasmussen, "A case based comparison of identification with neural network and Gaussian process models," in *IEEE Proceedings of IFAC International Conference on Intelligent Control Systems and Signal Processing*, 2003, pp. 137–142.

[70] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with Gaussian processes," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 4, pp. 411–424, 2005.

[71] T. Hachino and S. Yamakawa, "Non-parametric identification of continuous-time Hammerstein systems using Gaussian process model and particle swarm optimization," *Artificial Life and Robotics*, vol. 17, no. 1, pp. 35–40, 2012.

[72] C. Bailer-Jones, T. Sabin, D. MacKay, and P. Withers, "Prediction of deformed and annealed microstructures using Bayesian neural networks and Gaussian processes," in *Proceedings of the Australasia Pacific Forum on intelligent processing and manufacturing of materials*, vol. 2, 1997, pp. 913–919.

[73] C. Bailer-Jones, H. Bhadeshia, and D. J. C. Mackay, "Gaussian process modelling of austenite formation in steel," *Materials Science and Technology*, vol. 15, no. 3, pp. 287–294, 1999.

[74] K. Ažman and J. Kocijan, "Application of Gaussian processes for black-box modelling of biosystems," *ISA Transactions*, vol. 46, no. 4, pp. 443–457, 2007.

[75] M. Słoński, "Bayesian neural networks and Gaussian processes in identification of concrete properties," *Computer Assisted Mechanics and Engineering Sciences*, vol. 18, no. 4, pp. 291–302, 2011.

[76] K. K. S. Neo, "Non-linear dynamics identification using Gaussian process prior models within a Bayesian context," Ph.D. dissertation, National University of Ireland Maynooth, 2008.

[77] R. Urtasun, D. J. Fleet, and P. Fua, "3D people tracking with Gaussian process dynamical models," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2006, pp. 238–245.

[78] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[79] T. Wu and J. Movellan, "Semi-parametric Gaussian process for robot system identification," in *IEEE/RSJ Proceedings of International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 725–731.

[80] R. Grbic, D. Slikovic, and P. Kadlec, "Adaptive soft sensor for online prediction based on moving window Gaussian process regression," in *IEEE Proceedings of International Conference on Machine Learning and Applications (ICMLA)*, vol. 2. IEEE, 2012, pp. 428–433.

[81] A. Abusnina and D. Kudenko, "Adaptive soft sensor based on moving Gaussian process window," in *IEEE Proceedings of International Conference on Industrial Technology (ICIT)*. IEEE, 2013, pp. 1051–1056.

[82] J. Yu and S. J. Qin, "Multimode process monitoring with Bayesian inference-based finite Gaussian mixture models," *American Institute of Chemical Engineers*, vol. 54, no. 7, pp. 1811–1829, 2008.

[83] J. Yu, "A nonlinear kernel Gaussian mixture model based inferential monitoring approach for fault detection and diagnosis of chemical processes," *Chemical Engineering Science*, vol. 68, no. 1, pp. 506–519, 2012.

[84] ——, "Online quality prediction of nonlinear and non-Gaussian chemical processes with shifting dynamics using finite mixture model based Gaussian process regression approach," *Chemical Engineering Science*, vol. 82, pp. 22–30, 2012.

[85] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf, "Learning inverse dynamics: A comparison," in *Advances in Computational Intelligence and Learning: Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*. Evere, Belgium: d-side publications, 2008, pp. 13–18.

[86] D. Nguyen-Tuong and J. Peters, "Learning robot dynamics for computed torque control using local Gaussian processes regression," in *IEEE ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*. IEEE, 2008, pp. 59–64.

[87] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *American Control Conference*. IEEE, 2008, pp. 212–217.

[88] ——, "Model learning with local Gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[89] R. Murray-Smith and D. Sbarbaro, "Nonlinear adaptive control using nonparametric Gaussian process prior models," in *IEEE International Federation of Automatic Control (IFAC) World Congress on Automatic Control*, 2002.

[90] R. Murray-Smith, D. Sbarbaro, C. E. Rasmussen, and A. Girard, "Adaptive, cautious, predictive control with Gaussian process priors," in *IEEE International Federation of Automatic Control (IFAC) Symposium on System Identification*. Elsevier Science, 2003.

[91] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain input – Application to multiple-step ahead time series forecasting," in *Advances in Neural Information Processing Systems (NIPS)*. MIT, 2003, pp. 545–552.

[92] D. Sbarbaro and R. Murray-Smith, "An adaptive nonparametric controller for a class of nonminimum phase non-linear system," in *IEEE International Federation of Automatic Control (IFAC) World Congress*. Citeseer, 2005, pp. 729–729.

[93] D. Petelin, A. Grancharova, and J. Kocijan, "Evolving Gaussian process models for prediction of ozone concentration in the air," *Simulation Modelling Practice and Theory*, vol. 33, pp. 68–80, 2013.

[94] G. Chowdhary, M. Mühlegg, J. P. How, and F. Holzapfel, "Concurrent learning adaptive model predictive control," in *Advances in Aerospace Guidance, Navigation and Control*. Springer, 2013, pp. 29–47.

[95] Z. Zhao, X. Xia, J. Wang, J. Gu, and Y. Jin, "Nonlinear dynamic matrix control based on multiple operating models," *Journal of Process Control*, vol. 13, no. 1, pp. 41–56, 2003.

[96] H. Bouhenchir, M. Cabassud, and M.-V. Le Lann, "Predictive functional control for the temperature control of a chemical batch reactor," *Computers & Chemical Engineering*, vol. 30, no. 6, pp. 1141–1154, 2006.

[97] P. H. Sørensen, M. Nørgaard, O. Ravn, and N. K. Poulsen, "Implementation of neural network based non-linear predictive control," *Neurocomputing*, vol. 28, no. 1, pp. 37–51, 1999.

[98] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar, "Predictive control with Gaussian process models," in *Proceedings of IEEE Region 8 EUROCON 2003:Computer As A Tool*, vol. A. Ljubljana: IEEE, 2003, pp. 352–356.

[99] A. Grancharova, J. Kocijan, and T. A. Johansen, "Explicit stochastic nonlinear predictive control based on Gaussian process models," in *European Control Conference*, 2007, pp. 2340–2347.

[100] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear model predictive control*, ser. Lecture Notes in Control and Information Sciences. Springer, 2009, vol. 384, pp. 345–369.

[101] F. Berkenkamp and A. P. Schoellig, "Learning-based robust control: Guaranteeing stability while improving performance," in *IEEE/RSJ Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[102] Y. Wang, C. Ocampo-Martinez, and V. Puig, "Robust model predictive control based on gaussian processes: Application to drinking water networks," in *European Control Conference*. IEEE, 2015, pp. 3292–3297.

[103] Y. Wang, C. Ocampo-Martinez, V. Puig, and J. Quevedo, "Gaussian process based demand forecasting for predictive control of drinking water networks," in *Proceedings of the 9th International Conference on Critical Information Infrastructures Security (CRITIS)*, 2014.

[104] M. A. Alvarez and N. D. Lawrence, "Computationally efficient convolved multiple output Gaussian processes," *Journal of Machine Learning Research*, vol. 12, no. May, pp. 1459–1500, 2011.

[105] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[106] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE Proceedings of International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942–1948.

[107] R. An, P. Gong, H. Wang, X. Feng, P. Xiao, Q. Chen, Q. Zhang, C. Chen, and P. Yan, "A modified PSO algorithm for remote sensing image template matching," *Photogrammetric engineering and remote sensing*, vol. 76, no. 4, pp. 379–389, 2010.

[108] M. M. Noel, "A new gradient based particle swarm optimization algorithm for accurate computation of global minimum," *Applied Soft Computing*, vol. 12, no. 1, pp. 353–359, 2012.

[109] S. Gaffour, M. Mahfouf, and Y. Y. Yang, "'Symbiotic' data-driven modelling for the accurate prediction of mechanical properties of alloy steels," in *IEEE Proceedings of International Conference of Intelligent Systems*. IEEE, 2010, pp. 31–36.

[110] M. Majji, "System identification: time varying and nonlinear methods," Ph.D. dissertation, Texas A&M University, 2009.

[111] Z. Hou and S. Jin, "Data-driven model-free adaptive control for a class of MIMO nonlinear discrete-time systems," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2173–2188, 2011.

[112] J. Zhang, S. S. Ge, and T. H. Lee, "Output feedback control of a class of discrete MIMO nonlinear systems with triangular form inputs," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1491–1503, 2005.

[113] M. P. Deisenroth, "Efficient reinforcement learning using Gaussian processes," Ph.D. dissertation, Karlsruhe Institute of Technology, 2010.

[114] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 1907–1915.

[115] J. Q. Candela, A. Girard, J. Larsen, and C. E. Rasmussen, "Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting," in *IEEE Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2. IEEE, 2003, pp. II–701.

[116] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[117] A. Grancharova, J. Kocijan, and T. A. Johansen, "Explicit stochastic predictive control of combustion plants based on Gaussian process models," *Automatica*, vol. 44, no. 6, pp. 1621–1631, 2008.

[118] B. Kouvaritakis and M. Cannon, "Stochastic model predictive control," *Encyclopedia of Systems and Control*, pp. 1–9, 2014.

[119] Y. Yuan, "Step-sizes for the gradient method," *AMS IP Studies in Advanced Mathematics*, vol. 42, no. 2, p. 785, 2008.

[120] C. Kirches, *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Springer, 2011.

[121] L. Imsland, P. Kittilsen, and T. S. Schei, "Model-based optimizing control and estimation using Modelica model," *Modeling, Identification and Control*, vol. 31, no. 3, pp. 107–121, 2010.

[122] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[123] A. Jadbabai and J. Hauser, "On the stability of receding horizon control with a general terminal cost," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 674–678, 2005.

[124] K. Worthmann, "Estimates on the prediction horizon length in model predictive control," in *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS2012)*, 2012.

[125] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[126] L. Grüne and J. Pannek, *Nonlinear model predictive control–Theory and Algorithms*. London, U.K: Springer-Verlag, 2011.

[127] F. Tröltzsch, "Regular Lagrange multipliers for control problems with mixed pointwise control-state constraints," *SIAM Journal on Optimization*, vol. 15, no. 2, pp. 616–634, 2005.

[128] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *International Workshop on assessment and future directions on Nonlinear Model Predictive Control*. Pavia, Italy: Springer, 2008, pp. 391–417.

[129] S. Lucidi, M. Sciandrone, and P. Tseng, "Objective-derivative-free methods for constrained optimization," *Mathematical Programming*, vol. 92, no. 1, pp. 37–59, 2002.

[130] G. Liuzzi, S. Lucidi, and M. Sciandrone, "Sequential penalty derivative-free methods for nonlinear constrained optimization," *SIAM Journal on Optimization*, vol. 20, no. 5, pp. 2614–2635, 2010.

[131] L. Yiqing, Y. Xigang, and L. Yongjian, "An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints," *Computers & chemical engineering*, vol. 31, no. 3, pp. 153–162, 2007.

[132] O. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, 2005.

[133] S. J. Wright and M. J. Tenny, "A feasible trust-region sequential quadratic programming algorithm," *SIAM journal on optimization*, vol. 14, no. 4, pp. 1074–1105, 2004.

[134] Y.-h. Peng and S. Yao, "A feasible trust-region algorithm for inequality constrained optimization," *Applied mathematics and computation*, vol. 173, no. 1, pp. 513–522, 2006.

[135] X. Zhang, J. Zhang, and L. Liao, "An adaptive trust region method and its convergence," *Science in China Series A: Mathematics*, vol. 45, no. 5, pp. 620–631, 2002.

[136] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[137] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[138] R. Fletcher, *Practical methods of optimization*, 2nd ed. Wiley-Interscience Publication, 1987.

[139] Y. Pan and J. Wang, "Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3089–3101, 2012.

[140] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *IEEE/RSJ Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2004, pp. 2451–2456.

[141] K. Alexis, G. Nikolakopoulos, and A. Tzes, "On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances," *Asian Journal of Control*, vol. 16, no. 1, pp. 209–224, 2014.

[142] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[143] M. A. Rodrigues and D. Odloak, "An infinite horizon model predictive control for stable and integrating processes," *Computers & Chemical Engineering*, vol. 27, no. 8, pp. 1113–1128, 2003.

[144] T. Erez, Y. Tassa, and E. Todorov, "Infinite-horizon model predictive control for periodic tasks with contacts," *Robotics: Science and systems VII*, p. 73, 2012.

[145] W.-H. Chen, D. J. Ballance, and J. O'Reilly, "Optimisation of attraction domains of nonlinear MPC via LMI methods," in *American Control Conference*, vol. 4. IEEE, 2001, pp. 3067–3072.

[146] A. C. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013, pp. 207–215.

[147] C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto, and N. D. Lawrence, "Recurrent Gaussian processes," in *International Conference on Learning Representations (ICLR)*, 2016.