Gaussian Process Model Predictive Control of Unknown Nonlinear Systems

Gang Cao[1,*], Edmund M-K Lai[2], Fakhrul Alam[1]

[1]School of Engineering and Advanced Technology, Massey University, Auckland, New Zealand.
[2]Department of Information Technology and Software Engineering, Auckland University of Technology, Auckland, New Zealand.
[*]g.cao@massey.ac.nz

Abstract: Model Predictive Control (MPC) of an unknown system that is modelled by Gaussian Process (GP) techniques is studied in this paper. Using GP, the variances computed during the modelling and inference processes allow us to take model uncertainty into account. The main issue in using MPC to control systems modelled by GP is the propagation of such uncertainties within the control horizon. In this paper, two approaches to solve this problem, called GPMPC1 and GPMPC2, are proposed. With GPMPC1, the original Stochastic Model Predictive Control (SMPC) problem is relaxed to a deterministic nonlinear MPC based on a basic linearized GP local model. The resulting optimization problem, though non-convex, can be solved by the Sequential Quadratic Programming (SQP). By incorporating the model variance into the state vector, an extended local model is derived. This model allows us to relax the non-convex MPC problem to a convex one which can be solved by an active-set method efficiently. The performance of both approaches is demonstrated by applying them to two trajectory tracking problems. Results show that both GPMPC1 and GPMPC2 produce effective controls but GPMPC2 is much more efficient computationally.

1. Introduction

Model Predictive Control (MPC), also known as receding horizon control, is a class of computer control algorithms that predicts future responses of a plant based on its system model, and computes optimized control inputs by repeatedly solving a finite horizon optimization problem [1]. 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 by other control methods. It has been applied to many different types of control problems [2, 3].

The performance of MPC is highly dependent on the accuracy of the system model that describes its dynamics. Traditionally, these models are derived mathematically. More recently, data-driven modelling approaches based on computational intelligence and machine learning techniques are becoming popular [4, 5]. This approach is especially suitable for complex and highly nonlinear systems where complete knowledge of the system dynamics is seldom available, giving rise to unmodelled dynamics or model uncertainty. From the MPC perspective, attempts to address the issue of model uncertainty has been made through Robust Model Predictive Control (RMPC) schemes such as open-loop "min-max" MPC [6], closed-loop "min-max" MPC [7] and tube-based MPC [8].

"Min-max" MPC is conceptually simple. However, its control laws are computed based on worst-case scenarios and are therefore considered too conservative. Tube-based MPC overcomes this problem by combining a conventional MPC for the nominal system and a local feedback control law that steers the states of the unknown system to the inside of a "tube" centered on the nominal trajectory [9]. This "tube", which relates to the uncertainty bounds, must be carefully defined. Otherwise, there may not be a feasible solution. The major problem with RMPC is that model uncertainties are assumed to be deterministic even though they are typically stochastic.

Stochastic Model Predictive Control (SMPC) is an alternative where model uncertainties are assumed to be stochastic with an underlying probability distribution [10–13]. Control laws are computed by solving a stochastic optimization problem. Furthermore, since the state or output constraints are also probabilistic, they can be satisfied with a predefined level of confidence. This effectively alleviates the conservatism of "min-max" MPC. Furthermore, it is possible to trade-off control performance with robustness against model uncertainties by adjusting these probabilistic constraints. A key problem with SMPC is the propagation of uncertainties over a finite prediction horizon. The most common solution is to use sampling-based Monte Carlo (MC) simulation techniques. However, they are computationally demanding. More recently, a technique known as polynomial chaos expansions has been proposed to lighten the computation burden [13, 14].

A model known as Gaussian Process (GP) has become very useful in statistical modelling [15]. The GP variances which are computed as part of the modelling process provide a useful indication of the accuracy of the model. These variances can also be propagated in multiple-step ahead predictions. The hyperparameters of these models are learnt from data by maximizing the log-likelihood function. This optimization problem is unconstrained, nonlinear and non-convex optimization. It is typically solved by Conjugate Gradient (CG) [15] or by Particle Swarm Optimization (PSO) techniques [16–18].

A GP based MPC scheme was first introduced in [19]. Subsequently, an SMPC scheme using GP was proposed in [20]. Even though GP is a probabilistic model, the cost functions used in these papers are deterministic. Consequently, the variances could only be treated as slack variables of the state constraints. This indirect way of handling GP variances leads to a nonlinear optimization problem that is very computationally demanding to solve. More recently, in [21–23], variances are included in the cost function and can be directly handled in the optimization process. However, only unconstrained MPC have been considered.

In this paper, two new GP based MPC approaches, referred to as GPMPC1 and GPMPC2, are proposed for the control of unknown nonlinear dynamical systems with input and state constraints. The GPMPC1 approach is similar to those in [19, 20] in the sense that the GP variances are considered as a slack variable in the state constraints. The main difference is that the resulting non-convex optimization problem is solved by using a Sequential Quadratic Programming (SQP) based method together with a linearized GP model which is called the basic GP based local model in this paper. The constrained stochastic problem is then relaxed to a deterministic one by specifying the confidence level. With GPMPC2, the nonlinear MPC problem is reformulated to a convex optimization problem. In contrast with earlier methods, GP variances are directly included in the cost function of the optimization. The solution method makes use of a modified version of the basic local model which includes the variance in the modified state variable of the system. The resulting MPC problem is efficiently solved by using an active-set method. The effectiveness of these approaches are demonstrated by applying them to two trajectory tracking problems.

The rest of this paper is organized as follows. Section 2 introduces the modelling of the unknown nonlinear system by using GP models. The basic and extended GP based local dynamical

models are presented in the Section 3. In Section 4, the proposed GPMPC1 and GPMPC2 are presented for the general trajectory tracking problem of the unknown nonlinear system. In addition, the feasibility and stability of the proposed algorithms are also discussed. The simulation results are next reported to demonstrate the performance of the proposed algorithms in Section 5. Finally, Section 6 draws the conclusions.

2. Unknown system modelling using GP

Consider a discrete-time nonlinear dynamical system described by the following general form:

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

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear function, $\mathbf{w} \in \mathbb{R}^n$ represents additive external disturbances, $\mathbf{x} \in \mathbb{R}^n$ denotes the state vector, and $\mathbf{u} \in \mathbb{R}^m$ are control signals. In this paper, we assume that f is totally unknown but can be represented by a GP model. The uncertainty of a GP model can be measured by the GP variances. Therefore, a disturbance observer will not be required. The hyperparameters of a GP model is learnt from a set of training data consisting of inputs to the system and the system's response as target.

To model a system given by (1), a natural choice of the model inputs and their targets are the state-control tuple $\tilde{\mathbf{x}}_k = (\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}^{n+m}$ and the next state $\mathbf{x}_{k+1}$ respectively. Let $\Delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k \in \mathbb{R}^n$. In practice, the variation between $\Delta\mathbf{x}_{k+1}$ and $\Delta\mathbf{x}_k$ is much less the variation between $\mathbf{x}_{k+1}$ and $\mathbf{x}_k$, for all k. Therefore it is more advantageous to use $\Delta\mathbf{x}_k$ as the model target instead [24]. This will be assumed in the rest of this paper.

2.1. GP Modelling

A GP model is completely specified by its mean and covariance function [15]. Assuming that the mean of the model input $\tilde{\mathbf{x}}_k$ is zero, the squared exponential covariance is given by $\mathbf{K}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \sigma_s^2 \exp(-\frac{1}{2}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)^T \boldsymbol{\Lambda}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)) + \sigma_n^2$. The parameters σ_s^2, σ_n^2 and the entries of matrix $\boldsymbol{\Lambda}$ are referred to as the hyperparameters $\boldsymbol{\theta}$ of a GP model. Given D training inputs $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_D]$ and their corresponding training targets $\mathbf{y} = [\Delta\mathbf{x}_1, \cdots, \Delta\mathbf{x}_D]^T$, the joint distribution between $\mathbf{y}$ and a test target $\Delta\mathbf{x}_k^*$ for training input $\tilde{\mathbf{x}}_k^*$ is assumed to follow a Gaussian distribution. That is,

$$p\begin{pmatrix} \mathbf{y} \\ \Delta\mathbf{x}_k^* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{matrix} \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I} & \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k^*) \\ \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}}) & \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{x}}_k^*) \end{matrix}\right) \tag{2}$$

In addition, the posterior distribution over the observations can be obtained by restricting the joint distribution to only contain those targets that agree with the observations [15]. This is achieved by conditioning the joint distribution on the observations, and results in the predictive mean and variance function as follows:

$$m(\tilde{\mathbf{x}}_k^*) = E_f[\Delta\mathbf{x}_k^*] = \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{y} \tag{3a}$$

$$\sigma^2(\tilde{\mathbf{x}}_k^*) = Var_f[\Delta\mathbf{x}_k^*] = \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{x}}_k^*) - \mathbf{K}(\tilde{\mathbf{x}}_k^*, \tilde{\mathbf{X}})\mathbf{K}_\sigma^{-1}\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_k^*) \tag{3b}$$

where $\mathbf{K}_\sigma = \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I}$. The state at the next sampling time also follows a Gaussian distribution. Thus,

$$p(\mathbf{x}_{k+1}) \sim \mathcal{N}(\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}) \tag{4}$$

where

$$\boldsymbol{\mu}_{k+1} = \mathbf{x}_k + m(\tilde{\mathbf{x}}_k^*) \tag{5a}$$

$$\boldsymbol{\Sigma}_{k+1} = \sigma^2(\tilde{\mathbf{x}}_k^*) \tag{5b}$$

Typically, the hyperparameters of the GP model are learned by maximizing the log-likelihood function given by

$$\log p(\mathbf{y}|\tilde{\mathbf{X}}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T \mathbf{K}_\sigma^{-1} \mathbf{y} - \frac{1}{2}\log\left|\mathbf{K}_\sigma^{-1}\right| - \frac{D}{2}\log(2\pi) \tag{6}$$

This results in a nonlinear non-convex optimization problem that is traditionally solved by using CG or Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms. Recently, PSO based algorithms that minimizes the model error instead of the log-likelihood function have been shown in [18] to be more efficient and effective.

2.2. Uncertainty propagation

With the GP model obtained, one-step-ahead predictions can be made by using (3) and (5). When multiple-step predictions are required, the conventional way is to iteratively perform multiple one-step-ahead predictions using the estimated mean values. However, this process does not take into account the uncertainties introduced by each successive prediction. This issue has been shown to be important in time-series predictions [25].

The uncertainty propagation problem can be dealt with by assuming that the joint distribution of the training inputs is uncertain and follows a Gaussian distribution. That is,

$$p(\tilde{\mathbf{x}}_k) = p\big(\mathbf{x}_k, \mathbf{u}_k\big) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k) \tag{7}$$

with mean and variance given by

$$\tilde{\boldsymbol{\mu}}_k = \left[\boldsymbol{\mu}_k, E\left[\mathbf{u}_k\right]\right]^T \tag{8a}$$

$$\tilde{\boldsymbol{\Sigma}}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k & Cov\left[\mathbf{x}_k, \mathbf{u}_k\right] \\ Cov\left[\mathbf{u}_k, \mathbf{x}_k\right] & Var\left[\mathbf{u}_k\right] \end{bmatrix} \tag{8b}$$

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

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

$$p(\Delta \mathbf{x}_k^*) = \int p(f(\tilde{\mathbf{x}}_k^*)|\tilde{\mathbf{x}}_k^*) p(\tilde{\mathbf{x}}_k^*) d\tilde{\mathbf{x}}_k^* \tag{9}$$

However, this integral is analytically intractable. Numerical solutions can be obtained using Monte-Carlo simulation techniques. In [26], a moment-matching based approach is proposed to obtain an analytical Gaussian approximation. The mean and variance at an uncertain input can

be obtained through the laws of iterated expectations and conditional variances respectively [24]. They are given by

$$m(\tilde{\mathbf{x}}_k^*) = E_{\tilde{\mathbf{x}}_k^*}\Big[E_f\big[\Delta\mathbf{x}_k^*\big]\Big] \tag{10a}$$

$$\sigma^2(\tilde{\mathbf{x}}_k^*) = E_{\tilde{\mathbf{x}}_k^*}\Big[Var_f\big[\Delta\mathbf{x}_k^*\big]\Big] + Var_{\tilde{\mathbf{x}}_k^*}\Big[E_f\big[\Delta\mathbf{x}_k^*\big]\Big] \tag{10b}$$

Equation (5) then becomes

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + m(\tilde{\mathbf{x}}_k^*) \tag{11a}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{k+1} =& \boldsymbol{\Sigma}_k + \sigma^2(\tilde{\mathbf{x}}_k^*) \\ &+ Cov\big[\mathbf{x}_k, \Delta\mathbf{x}_k\big] + Cov\big[\Delta\mathbf{x}_k, \mathbf{x}_k\big]\end{aligned} \tag{11b}$$

The computational complexity of GP inference using (10) is $\mathcal{O}(D^2n^2(n+m))$ which is quite high. Hence, 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, sparse GP approaches [27] are often used.

3. 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 involved in the nonlinear control problem. The same technique is used here for the GP based MPC optimization problem. The main difference here is that the model of the system is probabilistic rather than deterministic. Thus there is more than one way by which the GP model could be linearized.

In [28], 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 [29] 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 4.

3.1. 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 (1) becomes

$$\boldsymbol{\mu}_{k+1} = \mathcal{F}(\boldsymbol{\mu}_k, \mathbf{u}_k) \tag{12}$$

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 (12), we have

$$\Delta\boldsymbol{\mu}_{k+1} = \frac{\partial\mathcal{F}}{\partial\boldsymbol{\mu}_k}\Delta\boldsymbol{\mu}_k + \frac{\partial\mathcal{F}}{\partial\mathbf{u}_k}\Delta\mathbf{u}_k \tag{13a}$$

$$= \frac{\partial\boldsymbol{\mu}_{k+1}}{\partial\boldsymbol{\mu}_k}\Delta\boldsymbol{\mu}_k + \frac{\partial\boldsymbol{\mu}_{k+1}}{\partial\mathbf{u}_k}\Delta\mathbf{u}_k \tag{13b}$$

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

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k} \tag{14a}$$

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k}\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k}\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k} \tag{14b}$$

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 (8). 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 [24].

3.2. 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 (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 (1) becomes

$$\mathbf{s}_{k+1} = \mathcal{F}'(\mathbf{s}_k, \mathbf{u}_k) \tag{15}$$

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

$$\Delta \mathbf{s}_{k+1} = \frac{\partial \mathcal{F}'}{\partial \mathbf{s}_k}\Delta \mathbf{s}_k + \frac{\partial \mathcal{F}'}{\partial \mathbf{u}_k}\Delta \mathbf{u}_k \tag{16}$$

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

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{s}_k} = \begin{bmatrix} \dfrac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\mu}_k} & \dfrac{\partial \boldsymbol{\mu}_{k+1}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \\ \dfrac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\mu}_k} & \dfrac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \end{bmatrix} \in \mathbb{R}^{(n+n^2)\times(n+n^2)} \tag{17a}$$

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{u}_k} = \begin{bmatrix} \dfrac{\partial \boldsymbol{\mu}_{k+1}}{\partial \mathbf{u}_k} \\ \dfrac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \mathbf{u}_k} \end{bmatrix} \in \mathbb{R}^{(n+n^2)\times m} \tag{17b}$$

[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

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\Sigma}_k} \frac{\partial \boldsymbol{\Sigma}_k}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \tag{18a}$$

$$\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\mu}_k} \tag{18b}$$

$$\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \sqrt{\boldsymbol{\Sigma}_k}} = \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\Sigma}_k} \frac{\partial \boldsymbol{\Sigma}_k}{\partial \sqrt{\boldsymbol{\Sigma}_k}} \tag{18c}$$

$$\frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \mathbf{u}_k} = \frac{\partial \sqrt{\boldsymbol{\Sigma}_{k+1}}}{\partial \boldsymbol{\Sigma}_{k+1}} \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \mathbf{u}_k} \tag{18d}$$

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

$$\frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k} + \frac{\partial \boldsymbol{\mu}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k} \tag{19a}$$

$$\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\mu}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\mu}_k} \tag{19b}$$

$$\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k} \tag{19c}$$

$$\frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \mathbf{u}_k} = \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\mu}}_k} \frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \mathbf{u}_k} + \frac{\partial \boldsymbol{\Sigma}_{k+1}}{\partial \tilde{\boldsymbol{\Sigma}}_k} \frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \mathbf{u}_k} \tag{19d}$$

$\frac{\partial \tilde{\boldsymbol{\mu}}_k}{\partial \boldsymbol{\Sigma}_k}$ and $\frac{\partial \tilde{\boldsymbol{\Sigma}}_k}{\partial \boldsymbol{\Sigma}_k}$ can be easily obtained based on (8). 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 [24].

4. Model Predictive Control based on GP

A discrete-time nonlinear dynamical system defined by (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:

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}, \mathbf{r}_k) \tag{20a}$$

$$\text{s.t. } \mathbf{x}_{k+i|k} = f(\mathbf{x}_{k+i-1|k}, \mathbf{u}_{k+i-1}) \tag{20b}$$

$$\mathbf{x}_{\min} \leq \mathbf{x}_{k+i|k} \leq \mathbf{x}_{\max} \tag{20c}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i-1} \leq \mathbf{u}_{\max} \tag{20d}$$

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

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 paper, 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

$$\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) = \sum_{i=1}^{H} \left\{ \left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 \right\} \tag{21}$$

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.

4.1. GPMPC1

4.1.1. Problem Reformulation: We assume that the system function $f(\cdot)$ is unknown and it is replaced by a GP model. Consequently, problem (20) becomes a stochastic one [30]:

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \tag{22a}$$

$$\text{s.t.} \quad p(\mathbf{x}_{k+1} | \mathbf{x}_k) \sim \mathcal{N}(\boldsymbol{\mu}_{k+1}, \boldsymbol{\Sigma}_{k+1}) \tag{22b}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{k+i-1} \leq \mathbf{u}_{\max} \tag{22c}$$

$$p\big\{\mathbf{x}_{k+i|k} \geq \mathbf{x}_{\min}\big\} \geq \eta \tag{22d}$$

$$p\big\{\mathbf{x}_{k+i|k} \leq \mathbf{x}_{\max}\big\} \geq \eta \tag{22e}$$

where η denotes a confidence level. For $\eta = 0.95$, the chance constraints (22d) and (22e) are equivalent to

$$\boldsymbol{\mu}_{k+i} - 2\boldsymbol{\Sigma}_{k+i} \geq \mathbf{x}_{\min} \tag{23a}$$

$$\boldsymbol{\mu}_{k+i} + 2\boldsymbol{\Sigma}_{k+i} \leq \mathbf{x}_{\max} \tag{23b}$$

Using (21) as the cost function, we get

$$\begin{aligned} & E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \\ &= E\Big[\sum_{i=1}^{H} \Big\{ \left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 \Big\} \Big] \\ &= \sum_{i=1}^{H} E\Big[\left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 \Big] \\ &= \sum_{i=1}^{H} \Big\{ E\Big[\left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 \Big] + E\Big[\left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 \Big] \Big\} \end{aligned} \tag{24}$$

In practice, the controls are deterministic. Hence, $E\big[\mathbf{u}_k^2\big] = \mathbf{u}_k^2$ and (24) becomes

$$\begin{aligned} & E\big[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\big] \\ =& \sum_{i=1}^{H} \Big\{ E\Big[\left\| \mathbf{x}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 \Big] + \left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 \Big\} \\ =& \sum_{i=1}^{H} \Big\{ \left\| \boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_{k+i-1} \right\|_{\mathbf{R}}^2 + \mathit{trace}\big(\mathbf{Q}\boldsymbol{\Sigma}_{k+i}\big) \Big\} \\ =& h\left(\boldsymbol{\mu}_k, \mathbf{u}_{k-1}\right) \end{aligned} \tag{25}$$

Now we have a deterministic cost function which involve the model variance $\mathbf{\Sigma}$ that allows model uncertainties to be explicitly included in the computation of the optimized controls. Note that for multiple-step predictions with uncertainty propagation, the computational complexity of problem (22) will not increase even though the GP model becomes more complicated. This is because the modelling and the control processes are independent of each other.

4.1.2. Nonlinear Optimization Solution: With the cost function (25) and the state constraints (23), the original stochastic optimization problem (22) has been relaxed to a deterministic constrained nonlinear optimization problem. But it is typically non-convex. This is usually solved by derivative-based approaches, such as Lagrange multipliers [31] based on first-order derivatives (gradient), or by SQP and interior-point algorithms based on second-order derivatives (Hessians matrix) [32]. When the derivative of the cost function is unavailable or is too difficult to compute, it could be approximated iteratively by a sampling method [33, 34]. Alternatively, evolutionary algorithms, such as PSO [35] and Genetic Algorithm (GA) [36], 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 [32].

A suitable technique for solving our MPC problem is the Feasibility-Perturbed Sequential Quadratic Programming (FP-SQP) algorithm proposed in [37]. It can be explained using the following general form of a constrained nonlinear optimization problem:

$$\min_{\mathbf{z}} h(\mathbf{z}) \qquad \mathbf{z} \in \mathbb{R}^{n+m} \tag{26a}$$

$$\text{s.t.} \quad c(\mathbf{z}) = 0 \tag{26b}$$

$$d(\mathbf{z}) \leq 0 \tag{26c}$$

where $h : \mathbb{R}^{n+m} \rightarrow \mathbb{R}$ is the objective function, $c : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n}$ and $d : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n+m}$ represents the corresponding equality and inequality constraints, respectively. FP-SQP generates a sequence of feasible solutions $\{\mathbf{z}^j\}_{j=0,1,2,\dots}$ 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:

$$\min_{\Delta\mathbf{z}^j} \nabla h(\mathbf{z}^j)^T \Delta\mathbf{z}^j + \frac{1}{2} \Delta\mathbf{z}^{j^T} \mathbf{H}^j \Delta\mathbf{z}^j \tag{27a}$$

$$\text{s.t. } c(\mathbf{z}^j) + \nabla c(\mathbf{z}^j)^T \Delta\mathbf{z}^j = 0 \tag{27b}$$

$$d(\mathbf{z}^j) + \nabla d(\mathbf{z}^j)^T \Delta\mathbf{z}^j \leq 0 \tag{27c}$$

under the trust-region constraint

$$\left\| \Delta\mathbf{z}^j \right\| \leq \gamma^j \tag{28}$$

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:

$$\mathbf{z}^j + \Delta\tilde{\mathbf{z}}^j \in \mathbf{\Pi} \tag{29a}$$

$$\frac{1}{2} \left\| \Delta\mathbf{z} \right\|_2 \leq \left\| \tilde{\Delta\mathbf{z}} \right\|_2 \leq \frac{3}{2} \left\| \Delta\mathbf{z} \right\|_2 \tag{29b}$$

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 (27);
  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 (30);
    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 (35);
    $j = j + 1$;
  end
end

Algorithm 1: The Feasibility-Perturbed Sequential Quadratic Programming used in the GPMPC1 algorithm

where $\mathbf{\Pi}$ denotes the feasible points set of problem (26). A method to obtain such a perturbation is proposed in [38]. An acceptability value of $\Delta\mathbf{z}^j$ defined by:

$$\rho^j = \frac{h(\mathbf{z}^{j+1}) - h(\mathbf{z}^j)}{-\nabla h(\mathbf{z}^j)^T\Delta\mathbf{z}^j - \frac{1}{2}\Delta\mathbf{z}^{jT}\mathbf{H}^j\Delta\mathbf{z}^j} \tag{30}$$

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 [39]. The complete FP-SQP algorithm is described in Algorithm 1.

4.1.3. Application to GPMPC1: Applying FP-SQP to the GPMPC1 problem (22), it should be noted that the constraints (23) 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

$$\mathbf{z}^{j+1} = \mathbf{z}^{j} + \Delta\mathbf{z}^{j} \tag{31}$$

Expressing the cost function (25) as (26a), define $\mathbf{z}_k = [\boldsymbol{\mu}_k^T, \mathbf{u}_{k-1}^T]^T \in \mathbb{R}^{n+m}$. Hence,

$$h(\mathbf{z_k}) = \sum_{i=1}^{H} \Big\{ \mathbf{z}_{k+i}^T \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & \mathbf{R} \end{bmatrix} \mathbf{z}_{k+i} + \text{trace}\big\{\mathbf{Q}\boldsymbol{\Sigma}_{k+i}\big\} \Big\} \tag{32}$$

One key issue in using FP-SQP is the local linearisation at $\Delta\mathbf{z}^j$. The basic GP based local model derived Section 3.1 shall be used to derive the QP subproblem as:

$$\min_{\Delta\mathbf{z}_k, \Delta\boldsymbol{\Sigma}_k} \sum_{i=1}^{H} \Big\{ \frac{\partial h}{\partial \mathbf{z}_{k+i}} \Delta\mathbf{z}_{k+i} + \frac{1}{2} \Delta\mathbf{z}_{k+i}^T \mathbf{H}_k \Delta\mathbf{z}_{k+i} \tag{33a}$$

$$+ \text{trace}\big\{\mathbf{Q}(\boldsymbol{\Sigma}_{k+i} + \Delta\boldsymbol{\Sigma}_{k+i})\big\} \Big\} \tag{33b}$$

$$\text{s.t.} \quad \Delta\boldsymbol{\mu}_{k+i+1} = \mathbf{A}_{k+i}\Delta\boldsymbol{\mu}_{k+i} + \mathbf{B}_{k+i}\Delta\mathbf{u}_{k+i} \tag{33c}$$

$$\mathbf{u}_{\text{min}} \leq \mathbf{u}_{k+i} + \Delta\mathbf{u}_{k+i} \leq \mathbf{u}_{\text{max}} \tag{33d}$$

$$\boldsymbol{\mu}_{k+i} + \Delta\boldsymbol{\mu}_{k+i} - 2(\boldsymbol{\Sigma}_{k+i} + \Delta\boldsymbol{\Sigma}_{k+i}) \geq \mathbf{x}_{\text{min}} \tag{33e}$$

$$\boldsymbol{\mu}_{k+i} + \Delta\boldsymbol{\mu}_{k+i} + 2(\boldsymbol{\Sigma}_{k+i} + \Delta\boldsymbol{\Sigma}_{k+i}) \leq \mathbf{x}_{\text{max}} \tag{33f}$$

$$\|\Delta\mathbf{z}_{k+i}\| \leq \gamma \tag{33g}$$

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 (12).

The computation of the Hessian matrix $\mathbf{H}_k$ of the Lagrangian in (27) is another key issue when using the FP-SQP algorithm. The exact Hessian matrix is usually obtained by

$$\mathbf{H}_k = \nabla^2 h(\mathbf{z}_k) + \sum_{i=1}^{n} \alpha_i \nabla^2 c(\mathbf{z}_k) + \sum_{i=1}^{n+m} \beta_i \nabla^2 d(\mathbf{z}_k) \tag{34}$$

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. To address these issues, approximation approaches have been proposed in [40]. In our work, $\mathbf{H}_k$ is approximately updated by using a Quasi-Newton method based on the BFGS. The update equation is given by

$$\mathbf{H}_{k+1} = \mathbf{H}_k - \frac{\mathbf{H}_k \Delta\mathbf{z}_k \Delta\mathbf{z}_k^T \mathbf{H}_k}{\Delta\mathbf{z}_k^T \mathbf{H}_k \Delta\mathbf{z}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \Delta\mathbf{z}_k} \tag{35}$$

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

[2] As shown in [24], 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.

4.2. GPMPC2

With GPMPC1, model uncertainty was introduced through the variance term into the objective function in (25). 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 (16). In this way, the variances are directly handled in the optimization process.

Another disadvantage of GPMPC1 is that the MPC optimization problem (22) 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, making it much easier to solve. Sensitivity to initial conditions is reduced and in most cases exact solutions can be obtained [32]. This convex optimization problem can be solved offline by using Multi-Parametric Quadratic Programs (mp-QP) [41] where the explicit solutions are computed as a lookup table of nonlinear controllers. An example can be found in [20]. 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 [42]. Using the extended GP based local model (16), the problem can be solved efficiently by an online active-set algorithm.

4.2.1. Problem Reformulation: Based on the extended local model in Section 3.2, define the state variable as

$$\begin{aligned} \mathbf{Z}_{k+1} &= \left[\mathbf{s}_{k+1|k}, \cdots, \mathbf{s}_{k+H|k}\right]^T \in \mathbb{R}^{H(n+n^2)} \\ &= \left[\boldsymbol{\mu}_{k+1}, \sqrt{\boldsymbol{\Sigma}_{k+1}}, \cdots, \boldsymbol{\mu}_{k+H}, \sqrt{\boldsymbol{\Sigma}_{k+H}}\right]^T \end{aligned} \tag{36}$$

Also, let

$$\mathbf{U}_k = \left[\mathbf{u}_k, \cdots, \mathbf{u}_{k+H-1}\right]^T \in \mathbb{R}^{Hm} \tag{37}$$

$$\mathbf{r}^*_{k+1} = \left[\mathbf{r}_{k+1}, \mathbf{0}, \cdots, \mathbf{r}_{k+H}, \mathbf{0}\right]^T \in \mathbb{R}^{H(n+n^2)} \tag{38}$$

Problem (22) then becomes

$$\min_{\mathbf{U}} \left\{ \left\| \mathbf{Z}_{k+1} - \mathbf{r}^*_{k+1} \right\|^2_{\tilde{\mathbf{Q}}} + \left\| \mathbf{U}_{k+1} \right\|^2_{\tilde{\mathbf{R}}} \right\} \tag{39a}$$

$$\text{s.t.} \quad \mathbf{I}_{Hn}\mathbf{x}_{\min} \leq \mathbf{M}_z \mathbf{Z}_{k+1} \leq \mathbf{I}_{Hn}\mathbf{x}_{\max} \tag{39b}$$

$$\mathbf{I}_{Hm}\mathbf{u}_{\min} \leq \mathbf{U}_{k+1} \leq \mathbf{I}_{Hm}\mathbf{u}_{\max} \tag{39c}$$

where

$$\tilde{\mathbf{Q}} = \text{diag}\{[\mathbf{Q}, \text{diag}\{\mathbf{vec}(\mathbf{Q})\}, \cdots, \mathbf{Q}, \text{diag}\{\mathbf{vec}(\mathbf{Q})\}]\} \in \mathbb{R}^{H(n+n^2)\times H(n+n^2)}, \tag{40a}$$

$$\tilde{\mathbf{R}} = \text{diag}\{[\mathbf{R}, \cdots, \mathbf{R}]\} \in \mathbb{R}^{Hm\times Hm}, \tag{40b}$$

$\mathbf{I}_a \in \mathbb{R}^a$ is the identity vector, and

$$\mathbf{M}_z = \begin{bmatrix} \mathbf{I}_n^T & 2\mathbf{I}_{n^2}^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_n^T & 2\mathbf{I}_{n^2}^T & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_n^T & 2\mathbf{I}_{n^2}^T \end{bmatrix} \in \mathbb{R}^{H\times H(n+n^2)} \tag{41}$$

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

$$\mathbf{U}_k = \mathbf{I}_{Hm}\mathbf{u}_{k-1} + \mathbf{T}_u \Delta \mathbf{U}_k \tag{42}$$

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

$$\Delta \mathbf{Z}_{k+1} = \tilde{\mathbf{A}} \Delta \mathbf{s}_k + \tilde{\mathbf{B}} \Delta \mathbf{U}_k \tag{43}$$

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

$$\tilde{\mathbf{A}} = \left[\mathbf{A}, \mathbf{A}^2, \cdots, \mathbf{A}^H\right]^T \in \mathbb{R}^{H(n+n^2)} \tag{44a}$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{A}^{H-1}\mathbf{B} & \mathbf{A}^{H-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix} \in \mathbb{R}^{H(n+n^2) \times Hm} \tag{44b}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the two Jacobian matrices (17) and (18) respectively. The corresponding state variable $\mathbf{Z}_{k+1}$ is therefore given by

$$\mathbf{Z}_{k+1} = \mathbf{s}_k + \mathbf{T}_z \left(\tilde{\mathbf{A}} \Delta \mathbf{s}_k + \tilde{\mathbf{B}} \Delta \mathbf{U}_k \right) \tag{45}$$

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

Based on (42) and (45), problem (39) can be expressed in a more compact form as

$$\min_{\Delta \mathbf{U}} \frac{1}{2} \left\| \Delta \mathbf{U}_k \right\|_{\mathbf{\Phi}}^2 + \boldsymbol{\psi}^T \Delta \mathbf{U}_k + \mathbf{C} \tag{46a}$$

$$\text{s.t.} \quad \Delta \mathbf{U}_{\min} \leq \begin{bmatrix} \mathbf{T}_u \\ \mathbf{T}_z \tilde{\mathbf{B}} \end{bmatrix} \Delta \mathbf{U}_k \leq \Delta \mathbf{U}_{\max} \tag{46b}$$

where

$$\mathbf{\Phi} = \tilde{\mathbf{B}}^T \mathbf{T}_z^T \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{B}} + \mathbf{T}_u^T \tilde{\mathbf{R}} \mathbf{T}_u \in \mathbb{R}^{Hm \times Hm} \tag{47a}$$

$$\boldsymbol{\psi} = 2(\mathbf{s}_k \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{B}} + \Delta \mathbf{s}_k \tilde{\mathbf{A}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{B}} \tag{47b}$$

$$- \mathbf{r}_{k+1}^* \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{B}} + \mathbf{u}_{k-1} \tilde{\mathbf{R}} \mathbf{T}_u) \in \mathbb{R}^{Hm} \tag{47c}$$

$$\mathbf{C} = (\mathbf{s}_k^2 + \mathbf{r}_{k+1}^*) \tilde{\mathbf{Q}} + 2\mathbf{s}_k \Delta \mathbf{s}_k \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{A}} \tag{47d}$$

$$+ \mathbf{u}_{k-1}^2 \tilde{\mathbf{R}} + \Delta \mathbf{s}_k^2 \tilde{\mathbf{A}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{A}} \tag{47e}$$

$$- 2\mathbf{r}_{k+1}^* (\mathbf{s}_k \tilde{\mathbf{Q}} - \Delta \mathbf{s}_k \tilde{\mathbf{Q}} \mathbf{T}_z \tilde{\mathbf{A}}) \tag{47f}$$

$$\Delta \mathbf{U}_{\min} = \begin{bmatrix} \mathbf{I}_{Hm}(\mathbf{u}_{\min} - \mathbf{u}_{k-1}) \\ \mathbf{I}_{H(n+n^2)}(\mathbf{x}_{\min} - \mathbf{s}_k - \mathbf{T}_z \tilde{\mathbf{A}} \Delta \mathbf{s}_k) \end{bmatrix} \tag{47g}$$

$$\Delta \mathbf{U}_{\max} = \begin{bmatrix} \mathbf{I}_{Hm}(\mathbf{u}_{\max} - \mathbf{u}_{k-1}) \\ \mathbf{I}_{H(n+n^2)}(\mathbf{x}_{\max} - \mathbf{s}_k - \mathbf{T}_z \tilde{\mathbf{A}} \Delta \mathbf{s}_k) \end{bmatrix} \tag{47h}$$

Since $\tilde{\mathbf{Q}}, \tilde{\mathbf{R}}, \mathbf{T}_z$ and $\mathbf{T}_u$ are positive definite, $\mathbf{\Phi}$ is also positive definite. Hence (46) is a constrained QP problem and is strictly convex. The solution will therefore be unique and satisfies the Karush-Kahn-Tucker (KKT) conditions.

4.2.2. Quadratic Programming Solution: The optimization problem (46) can be solved by an active-set method [43]. 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 are ill-conditioned or degenerated. In addition, it is conceptually simple and easy to implement. A warm-start technique could also be used to accelerate the optimization process substantially.

Let $\mathbf{G} = [\mathbf{T}_u, \mathbf{T}_z\tilde{\mathbf{B}}]^T$, the constraint (46b) can be written as

$$\begin{bmatrix} \mathbf{G} \\ -\mathbf{G} \end{bmatrix} \Delta\mathbf{U} \leq \begin{bmatrix} \Delta\mathbf{U}_{\max} \\ -\Delta\mathbf{U}_{\min} \end{bmatrix} \tag{48}$$

Ignoring the constant term $\mathbf{C}$, problem (46) becomes

$$\min_{\Delta\mathbf{U}} \frac{1}{2} \|\Delta\mathbf{U}_k\|_{\mathbf{\Phi}}^2 + \boldsymbol{\psi}^T \Delta\mathbf{U}_k \tag{49a}$$

$$\text{s.t.} \quad \tilde{\mathbf{G}}\Delta\mathbf{U}_k \leq \tilde{\Delta}_{\mathbf{U}} \tag{49b}$$

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 $\mathbf{\Pi}_{\Delta\mathbf{U}}$ 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 $\Delta\mathbf{U}_k^* \in \mathbf{\Pi}_{\Delta\mathbf{U}}$, the index set for the active set of constraints is defined as

$$\mathcal{A}(\Delta\mathbf{U}_k^*) = \{i \subseteq \mathcal{I} | \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^* = \tilde{\Delta}_{\mathbf{U},i}\} \tag{50}$$

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

$$\begin{aligned} \mathcal{B}(\Delta\mathbf{U}_k^*) &= \mathcal{I} \setminus \mathcal{A}(\Delta\mathbf{U}_k^*) \\ &= \{i \subseteq \mathcal{I} | \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^* < \tilde{\Delta}_{\mathbf{U},i}\} \end{aligned} \tag{51}$$

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 $\Delta\mathbf{U}_k^j \in \mathbf{\Pi}_{\Delta\mathbf{U}}$:

$$\min_{\boldsymbol{\delta}^j} \frac{1}{2} \left\|\Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j\right\|_{\mathbf{\Phi}}^2 + \boldsymbol{\psi}^T(\Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j) \tag{52a}$$

$$= \min_{\boldsymbol{\delta}^j} \frac{1}{2} \left\|\boldsymbol{\delta}^j\right\|_{\mathbf{\Phi}}^2 + (\boldsymbol{\psi} + \mathbf{\Phi}\Delta\mathbf{U}_k^j)^T\boldsymbol{\delta}^j \tag{52b}$$

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

$$\text{s.t.} \quad \tilde{\mathbf{G}}_i(\Delta\mathbf{U}_k^j + \boldsymbol{\delta}^j) = \tilde{\Delta}_{\mathbf{U},i}, i \in \mathcal{W}_k^j \tag{52c}$$

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

$$\min_{\boldsymbol{\delta}^j} \frac{1}{2} \left\|\boldsymbol{\delta}^j\right\|_{\mathbf{\Phi}}^2 + (\boldsymbol{\psi} + \mathbf{\Phi}\Delta\mathbf{U}_k^j)^T\boldsymbol{\delta}^j \tag{53a}$$

$$= \min_{\boldsymbol{\delta}^j} \frac{1}{2}\boldsymbol{\delta}^{jT}\mathbf{\Phi}\boldsymbol{\delta}^j + (\boldsymbol{\psi} + \mathbf{\Phi}\Delta\mathbf{U}_k^j)^T\boldsymbol{\delta}^j \tag{53b}$$

$$\text{s.t.} \quad \tilde{\mathbf{G}}_i\boldsymbol{\delta}^j = \tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i\Delta\mathbf{U}_k^j, i \in \mathcal{W}_k^j \tag{53c}$$

By applying the KKT conditions to problem (53), we can obtain the following linear equations:

$$\underbrace{\begin{bmatrix} \mathbf{\Phi} & \tilde{\mathbf{G}}_{\mathcal{A}}^T \\ \tilde{\mathbf{G}}_{\mathcal{A}} & \mathbf{0} \end{bmatrix}}_{\text{Lagrangian Matrix}} \begin{bmatrix} \boldsymbol{\delta}^j \\ \boldsymbol{\lambda}_k^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\psi} - \mathbf{\Phi}\Delta\mathbf{U}_k^j \\ \tilde{\Delta}_{\mathbf{U},\mathcal{A}} - \tilde{\mathbf{G}}_{\mathcal{A}}\Delta\mathbf{U}_k^j \end{bmatrix} \tag{54}$$

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

$$\begin{bmatrix} \mathbf{\Phi} & \tilde{\mathbf{G}}_{\mathcal{A}}^T \\ \tilde{\mathbf{G}}_{\mathcal{A}} & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{L}_2^T \\ \mathbf{L}_2 & \mathbf{L}_3 \end{bmatrix} \tag{55}$$

If this inverse exists, then the solution is given by

$$\boldsymbol{\delta}^j = -\mathbf{L}_1(\boldsymbol{\psi} + \mathbf{\Phi}\Delta\mathbf{U}_k^j) + \mathbf{L}_2^T(\tilde{\Delta}_{\mathbf{U},\mathcal{A}} - \tilde{\mathbf{G}}_{\mathcal{A}}\Delta\mathbf{U}_k^j) \tag{56a}$$

$$\boldsymbol{\lambda}_k^* = -\mathbf{L}_2(\boldsymbol{\psi} + \mathbf{\Phi}\Delta\mathbf{U}_k^j) + \mathbf{L}_3(\tilde{\Delta}_{\mathbf{U},\mathcal{A}} - \tilde{\mathbf{G}}_{\mathcal{A}}\Delta\mathbf{U}_k^j) \tag{56b}$$

where

$$\mathbf{L}_1 = \mathbf{\Phi}^{-1} - \mathbf{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T(\tilde{\mathbf{G}}_{\mathcal{A}}\mathbf{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T)^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}\mathbf{\Phi}^{-1} \tag{57a}$$

$$\mathbf{L}_2 = \mathbf{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T(\tilde{\mathbf{G}}_{\mathcal{A}}\mathbf{\Phi}^{-1} \tag{57b}$$

$$\mathbf{L}_3 = -(\tilde{\mathbf{G}}_{\mathcal{A}}\mathbf{\Phi}^{-1}\tilde{\mathbf{G}}_{\mathcal{A}}^T)^{-1} \tag{57c}$$

If $\boldsymbol{\delta}^j \neq 0$, then the set of feasible points $\Delta\mathbf{U}_k^j$ fails to minimize problem (49). 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

$$\kappa^j = \min\left\{1, \min_{i\in\mathcal{B}(\Delta\mathbf{U}_k^j)} \frac{\tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i\Delta\mathbf{U}_k^j}{\tilde{\mathbf{G}}_i\boldsymbol{\delta}^j}\right\} \tag{58}$$

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 (49) 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 2 summarizes the active set algorithm used in the GPMPC2.

4.2.3. Implementation Issues: The key to solving the linear equations (54) 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

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 (54);
  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 (58),
    $q = \operatorname{argmin}_{i \in \mathcal{B}(\Delta\mathbf{U}_k^j)} \dfrac{\tilde{\Delta}_{\mathbf{U},i} - \tilde{\mathbf{G}}_i \Delta\mathbf{U}_k^j}{\tilde{\mathbf{G}}_i \boldsymbol{\delta}^j}$
    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 2: Active set method for solving the GPMPC2 problem

$m_1 = \text{rank}(\tilde{\mathbf{G}}_{\mathcal{A}})$. $\mathcal{Q} \in \mathbb{R}^{Hm \times Hm}$ 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

$$\mathbf{L}_1 = \mathcal{Q}_2 (\mathcal{Q}_2^T \mathbf{\Phi} \mathcal{Q}_2)^{-1} \mathcal{Q}_2^T \tag{59a}$$

$$\mathbf{L}_2 = \mathcal{Q}_1 {\mathcal{R}^{-1}}^T - \mathbf{L}_1 \mathbf{\Phi} \mathcal{Q}_1 {\mathcal{R}^{-1}}^T \tag{59b}$$

$$\mathbf{L}_3 = \mathcal{R}^{-1} \mathcal{Q}_1^T \mathbf{\Phi} \mathbf{L}_2 \tag{59c}$$

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 [42].

4.3. Stability

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 (22) in the GPMPC1 algorithm, the objective (25) can be directly used as the Lyapunov function. Therefore, it can be known that

$$\mathcal{V}^*(k) = \sum_{i=1}^{H} \left\{ \left\| \Delta\boldsymbol{\mu}^*_{k+i} \right\|^2_{\mathbf{Q}} + \left\| \mathbf{u}^*_{k+i-1} \right\|^2_{\mathbf{R}} + \text{trace}(\mathbf{Q}\boldsymbol{\Sigma}^*_{k+i}) \right\} \tag{60}$$

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,

$$\begin{aligned}
&\mathcal{V}(k+1) && \text{(61a)}\\
&= \sum_{i=1}^{H} \left\{ \| \Delta\boldsymbol{\mu}_{k+1+i} \|^2_{\mathbf{Q}} + \| \mathbf{u}_{k+i} \|^2_{\mathbf{R}} + \text{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1+i}) \right\} && \text{(61b)}\\
&= \mathcal{V}^*(k) - \left\| \Delta\boldsymbol{\mu}^*_{k+1} \right\|^2_{\mathbf{Q}} - \| \mathbf{u}^*_k \|^2_{\mathbf{R}} - \text{trace}(\mathbf{Q}\boldsymbol{\Sigma}^*_{k+1}) && \text{(61c)}\\
&+ \| \Delta\boldsymbol{\mu}_{k+1+H} \|^2_{\mathbf{Q}} + \| \mathbf{u}_{k+H} \|^2_{\mathbf{R}} + \text{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1+H})
\end{aligned}$$

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,

$$\begin{aligned}
\mathcal{V}^*(k+1) \leq& \mathcal{V}(k+1) && \text{(62a)}\\
\leq& \mathcal{V}^*(k) + \| \Delta\boldsymbol{\mu}_{k+1+H} \|^2_{\mathbf{Q}} && \text{(62b)}\\
&+ \| \mathbf{u}_{k+H} \|^2_{\mathbf{R}} + \text{trace}(\mathbf{Q}\boldsymbol{\Sigma}_{k+1+H})
\end{aligned}$$

because of $\left\| \Delta\boldsymbol{\mu}^*_{k+1} \right\|^2_{\mathbf{Q}} \geq 0$, $\| \mathbf{u}^*_k \|^2_{\mathbf{R}} \geq 0$ and $\text{trace}(\mathbf{Q}\boldsymbol{\Sigma}^*_{k+1}) \geq 0$. The stability result of the problem (46) in the GPMPC2 algorithm can be obtained in the same way.

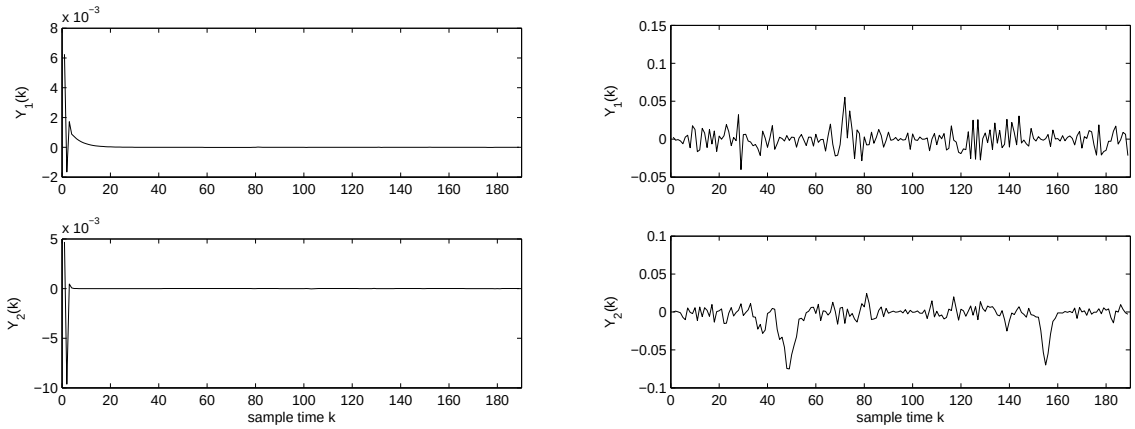
The result in (62) shows that, to guarantee the stability, additional terminal constraints on the means and variances of the GP model, as well as the control inputs are required such that,

$$\begin{aligned}
\boldsymbol{\mu}_{k+H+1|k} - \mathbf{r}_{k+H+1} &= 0 && \text{(63a)}\\
\boldsymbol{\Sigma}_{k+H+1|k} &= 0 && \text{(63b)}\\
\mathbf{u}_{k+H|k} &= 0 && \text{(63c)}
\end{aligned}$$

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].

5. Numerical Simulations

GPMPC1 and GPMPC2 are applied to two trajectory tracking problems of a Multiple-Input Multiple-Output (MIMO) nonlinear system with time-varying parameters. For each problem, 50 independent simulations are performed on a computer with a 3.40GHz Intel® Core™ 2 Duo CPU with



(a) "Step" trajectory

(b) "Lorenz" trajectory

Fig. 1. *Training errors of the system outputs for the two trajectory tracking problems.*

16 GB RAM, using Matlab® version 8.1. The average simulation results of these 50 trials are presented here.

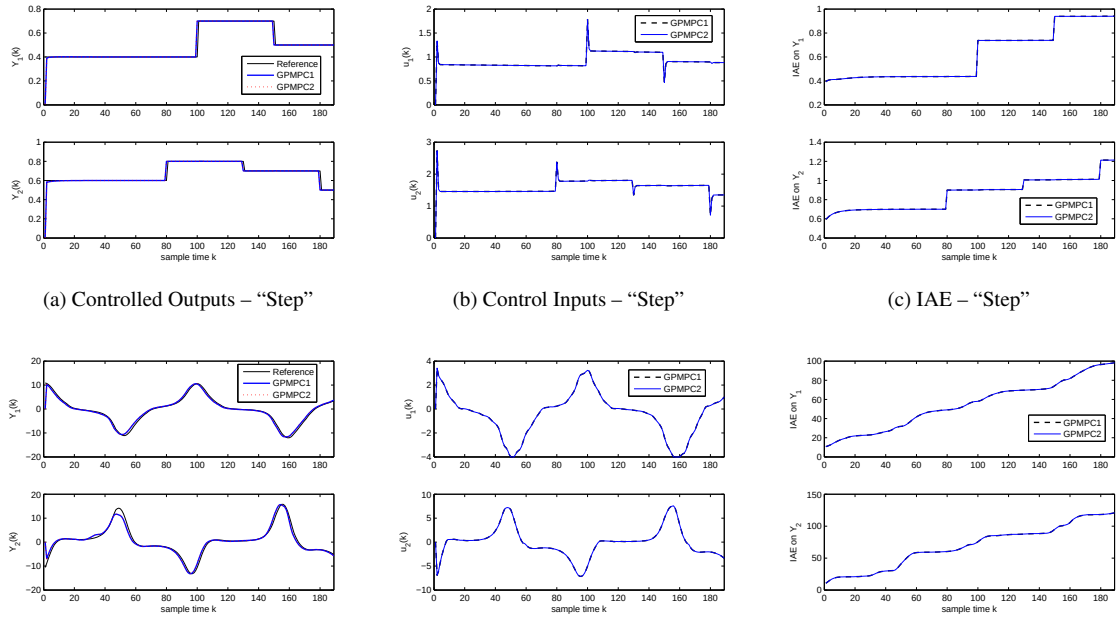
The MIMO nonlinear system in [44] is used for our simulations. It is described by:

$$\begin{aligned}
x_1(k+1) &= \frac{x_1(k)^2}{1+x_1(k)^2} + 0.3x_2(k) \\
x_2(k+1) &= \frac{x_1(k)^2}{1+x_2(k)^2+x_3(k)^2+x_4(k)^2} \\
&\quad + a(k)u_1(k) \\
x_3(k+1) &= \frac{x_3(k)^2}{1+x_3(k)^2} + 0.2x_4(k) \\
x_4(k+1) &= \frac{x_3(k)^2}{1+x_1(k)^2+x_2(k)^2+x_4(k)^2} \\
&\quad + b(k)u_2(k) \\
y_1(k+1) &= x_1(k+1) + \omega_1 \\
y_2(k+1) &= x_3(k+1) + \omega_2
\end{aligned} \tag{64}$$

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. $\omega_1, \omega_2 \sim \mathcal{N}(0, 0.01)$ are independent Gaussian white noise. In addition, the time-varying parameters $a(k)$ and $b(k)$ are given by

$$a(k) = 10 + 0.5\sin(k) \tag{65a}$$

$$b(k) = \frac{10}{1+\exp(-0.05k)} \tag{65b}$$



(a) Controlled Outputs – "Step" (b) Control Inputs – "Step" (c) IAE – "Step"

(d) Controlled Outputs – "Lorenz" (e) Control Inputs – "Lorenz" (f) IAE – "Lorenz"

Fig. 2. *Simulation results of the two trajectory tracking problems.*

5.1. "Step" Trajectory Tracking

The objective of the first experiment is to steer the nonlinear system to follow a step trajectory shown as the reference in Figure 2a. The system inputs are subjected to the following constraints:

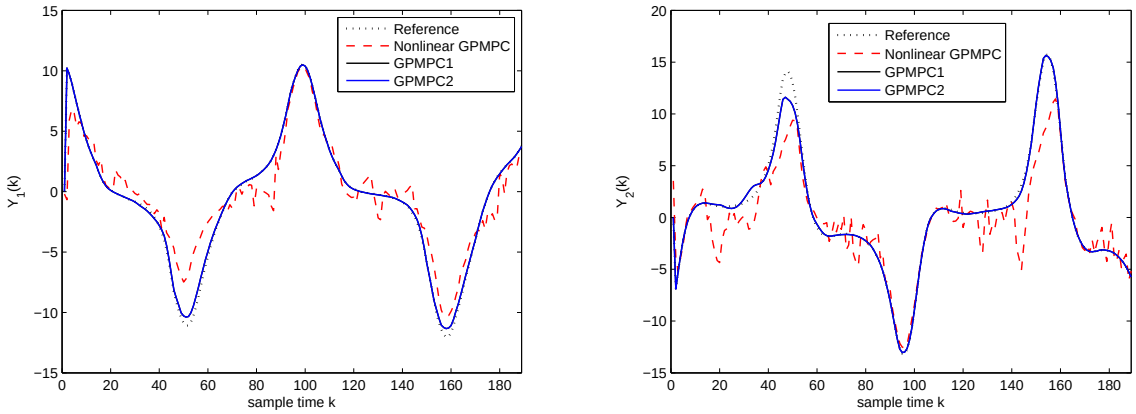
$$0 \leq u_1(k) \leq 5, \quad 0 \leq u_2(k) \leq 5$$

To generate the observations for GP modelling, this problem is first solved by using the Nonlinear Model Predictive Control (NMPC) strategy proposed in [45]. 189 observations are collected and are used to train the GP models. The learning process took approximately 2.1 seconds, with a training Mean Squared Error (MSE) of 9.9114×10^{-5}. Figure 1a shows the training errors for the 189 samples. These results show that the system is accurately learnt by using the GP models.

The MPC parameters in this simulation are: initial states $\mathbf{x}_0 = [0, 0, 0, 0]^T$ and initial control inputs $\mathbf{u}_0 = [0, 0]^T$, weighting matrix $\mathbf{Q} = \mathbf{I}_{4\times4}$ and $\mathbf{R} = \mathbf{I}_{2\times2}$. In addition, the prediction horizon H is 10. Theoretically, a long enough H is necessary to guarantee the stability of MPC controllers. However, the complexity of MPC problem increases exponentially with increasing H. This value of H is chosen as a trade-off between the control performance and computational complexity.

The resultant controlled outputs and control inputs by using GPMPC1 and GPMPC2 are shown in Figures 2a and 2b, respectively. They show that both algorithms exhibit equally good control performances in this task since they both produced outputs close to the target. The Integral Absolute Error (IAE) values can be found in Figure 2c.

GPMPC1 takes on average 34.1 seconds to compute the 189 optimized control inputs. However, GPMPC2 only requires 4.51 seconds which is more than 8 times more efficient than GPMPC1. This shows the advantage in our formulation of the problem as convex optimization.



(a) Y_1 – "Lorenz"

(b) Y_2 – "Lorenz"

Fig. 3. *Comparison of tracking performance of GPMPC1, GPMPC2 and nonlinear GPMPC in [46] for the Lorenz trajectory.*

5.2. "Lorenz" Trajectory Tracking

The second problem is to track a "Lorenz" trajectory as shown in Figure 2d. In this case, the constraints on the control inputs are:

$$-4 \leq u_1(k) \leq 4, \quad -7 \leq u_2(k) \leq 7$$

Similar to the previous experiment, the NMPC method is used to generate 189 observations for training the GP model. Training time is approximately 2.4 seconds with a training MSE of 0.0196. Figure 1b shows the training error.

The MPC parameters are: initial states $\mathbf{x}_0 = [0, 0, 0, 0]^T$, initial control inputs $\mathbf{u}_0 = [0, 0]^T$, prediction horizon $H = 10$, weighting matrix $\mathbf{Q} = \mathbf{I}_{4\times4}$ and $\mathbf{R} = \text{diag}\{[21000, 27000]\}$.

The tracking results can be found in Figures 2d, 2e and 2f. They demonstrate again that the control performance GPMPC1 and GPMPC2 are virtually the same. In this case, on average GPMPC2 is about 5 times more efficient than GPMPC1 (5.38 seconds versus 24.72 seconds).

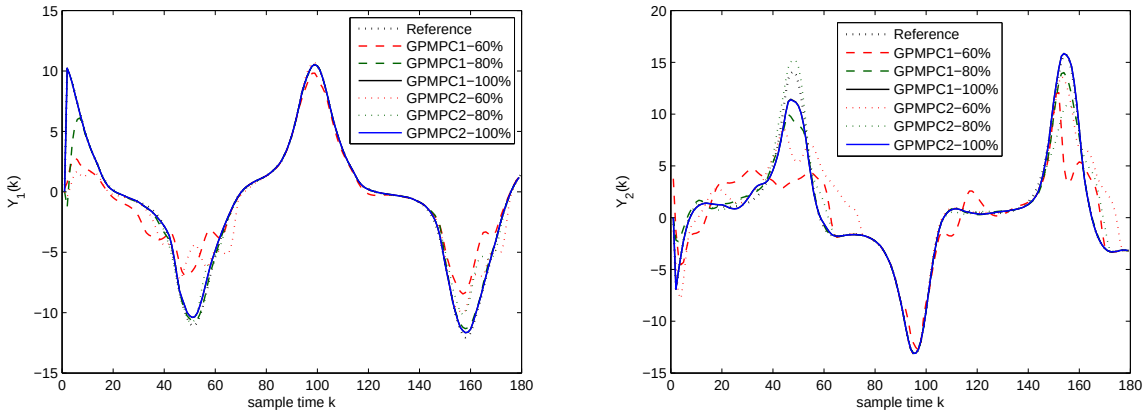
The performance of the two proposed algorithms is compared with the nonlinear GPMPC proposed in [46]. Even though problem (22) with cost function (25) is more complicated than the one considered in [46], they are essentially similar. Tracking results for $H = 1$ are shown in Figure 5.2. show that the both two proposed algorithms outperform than the nonlinear GPMPC in the "Lorenz" trajectory tracking problem. In addition, the GPMPC1 and GPMPC2 only require approximately 5 and 7 seconds to compute all 189 control actions, compared to 150 seconds used in nonlinear GPMPC.

5.3. Sensitivity to Training Data

Since the closed-loop stability of proposed GPMPC1 and GPMPC2 are not guaranteed as discussed in Section 4.3, it is necessary to test them with different models. Here, both GPMPC1 and GPMPC2 are each tested with three separate GP models for the Lorenz trajectory tracking problem. These models are trained by using 60%, 80% and 100% of all of 179 observations respectively. Figure 4 shows how well each model track the reference outputs. Table 1 shows the

Table 1 MSE values for Lorenz trajectory tracking problem with GPMPC1 and GPMPC2 using GP models with different amount of training data.

	Model for GPMPC1			Model for GPMPC2		
Training Data	60%	80%	100%	60%	80%	100%
Y_1	4.7831	1.36	0.0528	4.6493	0.6879	0.0539
Y_2	10.7518	1.0960	0.2995	6.6748	2.1522	0.3085



(a) Y_1 – "Lorenz"

(b) Y_2 – "Lorenz"

Fig. 4. *Comparisons between proposed approaches with different learnt GP models in the "Lorenz" trajectory tracking problem.*

tracking MSE values. These results indicate that while the models trained with 100% and 80% observations perform quite well, the ones trained with 60% data are inadequate.

6. Conclusions

Two GP based MPC approaches (GPMPC1 and GPMPC2) have been presented for the trajectory tracking problem of an unknown nonlinear dynamical system. The system is modelled using GP techniques offline. These two approaches handle the model uncertainties in the form of GP variances in different ways. GPMPC1 formulated the MPC optimization problem in such a way that model uncertainties are treated as the slack variables of GP mean constraints and are included in the objective function as the penalty term. The resulting SMPC problem is relaxed to a deterministic non-convex nonlinear optimization problem. The solution of the resultant problem is obtained using the FP-SQP method based on a linearized GP local model. With GPMPC2, the variance forms part of the state vector. This allows model uncertainties to be directly included in the computation of the optimized controls. By using the extended linearized GP local model, the non-convex optimization problem is relaxed to a convex one which is solved using an active-set method. Simulation results on two different trajectories show that both approaches perform equally well. However, GPMPC2 is several times more efficient computationally compared with GPMPC1, especially for a longer horizon. A brief discussion on how closed-loop stability could be guaranteed reveals that the resulting optimization problem will be different from the one considered in this paper. This issue will be addressed in future work.

7. 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] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[3] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[4] D. P. Solomatine and A. Ostfeld, "Data-driven modelling: some past experiences and new approaches," *Journal of hydroinformatics*, vol. 10, no. 1, pp. 3–22, 2008.

[5] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

[6] T. Alamo, D. M. de La Peña, D. Limón, and E. F. Camacho, "Constrained min-max predictive control: Modifications of the objective function leading to polynomial complexity," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 710–714, 2005.

[7] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, "Input to state stability of min–max MPC controllers for nonlinear systems with bounded uncertainties," *Automatica*, vol. 42, no. 5, pp. 797–803, 2006.

[8] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[9] L. Zhang, S. Zhuang, and R. D. Braatz, "Switched model predictive control of switched linear systems: Feasibility, stability and robustness," *Automatica*, vol. 67, pp. 8–21, 2016.

[10] A. T. Schwarm and M. Nikolaou, "Chance-constrained model predictive control," *American Institute of Chemical Engineers*, vol. 45, no. 8, pp. 1743–1752, 1999.

[11] D. Bernardini and A. Bemporad, "Scenario-based model predictive control of stochastic constrained linear systems," in *IEEE Proceedings of International Conference on Decision and Control*. IEEE, 2009, pp. 6333–6338.

[12] M. Cannon, B. Kouvaritakis, S. V. Raković, and Q. Cheng, "Stochastic tubes in model predictive control with probabilistic constraints," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 194–200, 2011.

[13] A. Mesbah, S. Streif, R. Findeisen, and R. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," in *American Control Conference*. IEEE, 2014, pp. 2413–2419.

[14] L. Fagiano and M. Khammash, "Nonlinear stochastic model predictive control via regularized polynomial chaos expansions," in *IEEE Proceedings of International Conference on Decision and Control*. IEEE, 2012, pp. 142–147.

[15] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 1 2006.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] ——, "Gaussian process model predictive control of Unmanned Quadrotors," in *International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 28-30 April 2016.

[24] M. P. Deisenroth, "Efficient reinforcement learning using Gaussian processes," Ph.D. dissertation, Karlsruhe Institute of Technology, 2010.

[25] 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*. MIT, 2003, pp. 545–552.

[26] 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.

[27] 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.

[28] 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.

[29] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Advances in Neural Information Processing Systems*, 2014, pp. 1907–1915.

[30] 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.

[31] 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.

[32] 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.

[33] S. Lucidi, M. Sciandrone, and P. Tseng, "Objective-derivative-free methods for constrained optimization," *Mathematical Programming*, vol. 92, no. 1, pp. 37–59, 2002.

[34] 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.

[35] 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.

[36] O. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, 2005.

[37] 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.

[38] 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.

[39] 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.

[40] M. J. Tenny, S. J. Wright, and J. B. Rawlings, "Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming," *Computational Optimization and Applications*, vol. 28, no. 1, pp. 87–121, 2004.

[41] 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.

[42] 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.

[43] R. Fletcher, *Practical methods of optimization*, 2nd ed. Wiley-Interscience Publication, 1987.

[44] 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.

[45] L. Grüne and J. Pannek, *Nonlinear model predictive control–Theory and Algorithms.* London, U.K: Springer-Verlag, 2011.

[46] J. Kocijan and R. Murray-Smith, "Nonlinear predictive control with a Gaussian process model," in *In R. Murray-Smith and R. Shorten (eds.), Switching and Learning in Feedback Systems*. Springer, 2005, pp. 185–200.