Gaussian Process based Model Predictive Control for Linear Time Varying Systems

Gang Cao, Edmund M-K Lai, Fakhrul Alam
School of Engineering and Advanced Technology, Massey University
Albany, Auckland, New Zealand
Email: {g.cao,e.lai,f.alam}@massey.ac.nz

Abstract Two main issues associated with Model Predictive Control (MPC) are learning the unknown dynamics of the system and handling model uncertainties. In this paper, unknown Linear Time-Varying (LTV) system with external noise is represented by using probabilistic Gaussian Process (GP) models. In this way, we can explicitly evaluate model uncertainties as variances. As a result, it is possible to directly take obtained variances into account when planing the policy. In addition, through using analytical gradients that are available during the GP modelling process, the optimization problem in GP based MPC can be solved faster. The performance of proposed approach is demonstrated by simulations on trajectory tracking problem of a LTV system.

I. Introduction

Model Predictive Control (MPC) [1, 2] refers to a class of computer control algorithms that predict future responses of a plant based on its system model. Control actions are obtained by repeatedly solving a finite horizon optimal control problem. The advantage of MPC mainly lies in its ability to handle multiple variable control problems. In addition, it can naturally incorporate input and output constraints that are commonly encountered in practice but are not well addressed by other control methods. Consequently, it has been widely used to control linear and nonlinear systems in research and in practice [3].

MPC requires an accurate, explicit model of the system dynamics of the plant. Where an explicit model is not available, data-driven models such as Artificial Neural Networks (ANN) [4, 5] and Fuzzy Models (FMs) [6, 7] have been used in the past. The training of ANN models requires a large number of training data while sufficient prior knowledge of the plant is need for the construction of FMs. Hence these methods are not applicable when the number of training observations are small and minimal prior knowledge of the plant is available. Another issue is concerned with the modelling of uncertainties arising from time-varying model parameter changes and external disturbances. This problem can be addressed by using deterministic robust MPC [1, 8] where the uncertainties are assumed to be bounded and robustness is guaranteed by using the "min-max" method [9]. However, controllers designed using this method are too conservative since the design is based on worst-case perturbations. In addition, uncertainty bounds are not easy to define in practice. An alternative is to use stochastic MPC where uncertainties are taken into account explicitly by probabilistic constraints [10]–[13].

In [14, 15], a Gaussian Process (GP) based stochastic MPC is proposed. The advantage of using GP models is that prediction accuracy is explicitly expressed through the computation of variances which is part of the modelling process. When performing multiple step predictions, GP models allow uncertainties to propagate, making it natural to be incorporated into stochastic MPC to address the model robustness issue. While model uncertainties are expressed as probabilistic "hard-constraints" in [14, 15], the use of "soft-constraints" to incorporate model uncertainty into policy planning and evaluation in a straightforward manner has also been proposed [14]. This latter scheme has recently been used to control a linear system with periodic errors [16]. But this method is computationally demanding.

In this paper, we proposed a computationally efficient way to solve the optimization problem in [16] by making use of gradients that can be obtained analytically from the GP model. We show through a trajectory tracking problem of an Linear Time-Varying (LTV) system, our proposed method is able to provide good modelling and control performances while substantially reducing the computation time.

The rest of this paper is organized as follows. Section II provides a brief overview of the GP modelling technique. In Section III, GP based MPC algorithm is described and the gradient expressions are derived. Simulation of trajectory tracking of an LTV system in Section IV demonstrate the effectiveness of the proposed method. Finally, Section V draws the conclusions.

II. 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

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

where k is the integer index of time, $f(\cdot)$ is an unknown linear time-varying function, and $\mathbf{w} \in \mathbb{R}^n$ represents Gaussian noise with zero mean and variance $\mathbf{\Sigma}_w$. This system can be modelled by a GP model where 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 [17, 18]. 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 a mean and a covariance function [19]. If the mean $\boldsymbol{\mu}$ is zero and the squared exponential covariance, defined as $\mathbf{K}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \sigma_s^2 \exp(-\frac{1}{2}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)^T \boldsymbol{\Lambda}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)) + \sigma_n^2$, is used, then σ_s^2, σ_n^2 and matrix $\boldsymbol{\Lambda}$ are the hyperparameters of the GP model. Given D training inputs $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_D]$ and the corresponding training targets $\mathbf{y} = [\delta\mathbf{x}_1, \cdots, \delta\mathbf{x}_D]^T$, the joint distribution between training targets and test target $\delta\mathbf{x}^*$ at a given training input $\tilde{\mathbf{x}}^*$ follows the Gaussian distribution. That is,

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

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

$$\begin{aligned} \mathbb{E}_f[\delta\mathbf{x}_k] &= \mathbf{K}(\tilde{\mathbf{x}}^*, \tilde{\mathbf{X}})(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I})^{-1}\mathbf{y} \\ \mathbb{VAR}_f[\delta\mathbf{x}_k] &= \mathbf{K}(\tilde{\mathbf{x}}^*, \tilde{\mathbf{x}}^*) \\ &\quad - \mathbf{K}(\tilde{\mathbf{x}}^*, \tilde{\mathbf{X}})(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I})^{-1}\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}^*) \end{aligned} \quad (3)$$

Typically, hyperparameters $\boldsymbol{\theta} = [\sigma_s, \sigma_n, \text{vec}(\boldsymbol{\Lambda})]$ are learned by using the evidence maximization technique [20], where $\text{vec}(\cdot)$ denotes vectorization of given matrix. This requires $\mathcal{O}(nD^3)$ operations. Conventionally, Conjugate Gradient (CG) or BFGS approaches are used to solve this stochastic optimization problem. More recently, Particle Swarm Optimization (PSO) based algorithms [21] have been proposed to solve this problem.

III. Gaussian Process Based Model Predictive Control

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

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) \quad (4)$$

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

$$\begin{aligned} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1}) = \sum_{i=1}^{H} \big\{ &(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T Q (\mathbf{x}_{k+i} - \mathbf{r}_{k+i}) \\ &+ \mathbf{u}_{k+i-1}^T R \mathbf{u}_{k+i-1} \big\} \end{aligned} \quad (5)$$

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

If the dynamical system (1) is represented by GP models, the predictions of $\mathbf{x}_k$ are stochastic. Hence the MPC is a stochastic one and (4) becomes [15, 22]

$$\mathbf{V}_k^* = \min_{\mathbf{u}(\cdot)} \mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] \quad (6)$$

The expected value of the cost function can be derived as

$$\begin{aligned} \mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] &= \mathbb{E}\left[\sum_{i=1}^{H} \big\{ (\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T Q (\mathbf{x}_{k+i} - \mathbf{r}_{k+i}) \right. \\ &\quad \left. + \mathbf{u}_{k+i-1}^T R \mathbf{u}_{k+i-1} \big\}\right] \\ &= \sum_{i=1}^{H} \mathbb{E}\left[(\mathbf{x}_{k+i} - \mathbf{r}_{k+i})^T Q (\mathbf{x}_{k+i} - \mathbf{r}_{k+i}) \right. \\ &\quad \left. + \mathbf{u}_{k+i-1}^T R \mathbf{u}_{k+i-1}\right] \end{aligned} \quad (7)$$

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

$$\begin{aligned} p(\tilde{\mathbf{x}}_k) &= p\begin{pmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{pmatrix} \\ &\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_k \\ \mathbf{u}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_k & \mathbb{COV}[\mathbf{x}_k, \mathbf{u}_k] \\ \mathbb{COV}[\mathbf{u}_k, \mathbf{x}_k] & \mathbb{COV}[\mathbf{u}_k, \mathbf{u}_k] \end{bmatrix}\right) \end{aligned} \quad (8)$$

where $\mathbb{COV}[\mathbf{x}_k, \mathbf{u}_k]$, $\mathbb{COV}[\mathbf{u}_k, \mathbf{x}_k]$ and $\mathbb{COV}[\mathbf{u}_k, \mathbf{u}_k]$ are zero. The cost function (7) can be simplified to

$$\begin{aligned} \mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] = \sum_{i=1}^{H} \big\{ &(\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i})^T Q (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}) \\ &+ \text{trace}(Q\boldsymbol{\Sigma}_{k+i}) + \mathbf{u}_{k+i-1}^T R \mathbf{u}_{k+i-1} \big\} \end{aligned} \quad (9)$$

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.

A. Uncertainty Propagation

With the Stochastic Model Predictive Control (SMPC) given by (6), one-step ahead predictions can be computed by using (3). For multiple-step predictions, the conventional way is to perform multiple one-step ahead predictions using only the estimates of the mean. However, the uncertainties induced by each successive prediction are not taken into account. This issue has been shown to be important in [23] with a time-series prediction task.

This 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 $p(\tilde{\mathbf{x}}_k) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)$. The exact predictive distribution of the training target can then be defined as $p(\delta\mathbf{x}_k) = \int p(f(\tilde{\mathbf{x}}_k)|\tilde{\mathbf{x}}_k) p(\tilde{\mathbf{x}}_k) d\tilde{\mathbf{x}}_k$. Although this equation is analytically intractable, it can be approximated as a Gaussian with mean $\boldsymbol{\mu}_k^\delta$ and variance $\boldsymbol{\Sigma}_k^\delta$ by using the moment matching techniques [17, 24]. This results in

$$\begin{aligned} \boldsymbol{\mu}_k^\delta &= \mathbb{E}_{\tilde{\mathbf{x}}_k}[\mathbb{E}_f[\delta\mathbf{x}_k]] \\ \boldsymbol{\Sigma}_k^\delta &= \begin{bmatrix} \mathbb{VAR}_{f,\tilde{\mathbf{x}}_k}[\delta\mathbf{x}_{k_1}] & \cdots & \mathbb{COV}_{f,\tilde{\mathbf{x}}_k}[\delta\mathbf{x}_{k_n}, \delta\mathbf{x}_{k_1}] \\ \vdots & \ddots & \vdots \\ \mathbb{COV}_{f,\tilde{\mathbf{x}}_k}[\delta\mathbf{x}_{k_1}, \delta\mathbf{x}_{k_n}] & \cdots & \mathbb{VAR}_{f,\tilde{\mathbf{x}}_k}[\delta\mathbf{x}_{k_n}] \end{bmatrix} \end{aligned} \quad (10)$$

The distribution at time $k+1$ can be further approximated by a Gaussian with mean and variance given by

$$\begin{aligned}\boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \boldsymbol{\mu}_k^{\delta} \\ \boldsymbol{\Sigma}_{k+1} &= \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}_k^{\delta} \\ &\quad + \mathbb{COV}_{f,\tilde{\mathbf{x}}_k}[\mathbf{x}_k, \delta\mathbf{x}_k] + \mathbb{COV}_{f,\tilde{\mathbf{x}}_k}[\delta\mathbf{x}_k, \mathbf{x}_k]\end{aligned} \tag{11}$$

In this way, the uncertainties of all previous predictions can be iteratively propagated to the current one. More details on the computation of the means and variances for uncertain inputs can be found in [17, 25].

Note that the propagated variances have been included in the cost function (9). This allows model uncertainties to be included directly when solving (6).

For problems with higher dimensions, sparse GP approaches [26] are often used.

B. Gradient Based Optimization

Solving (6) is computationally demanding. The computational complexity of the one-step moment matching in (10) alone requires $\mathcal{O}(D^2n^2(n+m))$ 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 (6) can be described in a condensed form as

$$z^* = \arg\min_{z\in\mathbb{Z}} h(z) \tag{12}$$

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 [17].

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

$$z(i+1) = z(i) + \alpha_s \nabla_z h(z(i)) \tag{13}$$

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 [27]. Using this method, suboptimal solutions to (6) can still be found even if it is non-convex.

The key issue in implementing this gradient-based method on problem (6) is computing the gradients that are derivatives of the value function w.r.t. controls. Numerical methods such as finite difference [28] 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 [29].

Input: Learning GP Models, H, $\mathbf{r}_k$, Q, 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
    Caculate graidents $\frac{d\mathbb{E}[\mathcal{J}(\mathbf{u}_i)]}{d\mathbf{u}_{i-1}}$ using (16);
    Update step length α_s accroding to [27];
    Update controls $\mathbf{u}_{i+1} = \mathbf{u}_i + \alpha_s \frac{d\mathbb{E}[\mathcal{J}(\mathbf{u}_i)]}{d\mathbf{u}_{i-1}}$;
    i=i+1;
  end
end
Output: Optimal controls $\mathbf{u}^*$.

Algorithm 1: Analytical gradient based optimization method

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

$$\begin{aligned}\mathcal{H}_i =& (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i})^T Q (\boldsymbol{\mu}_{k+i} - \mathbf{r}_{k+i}) \\ &+ \text{trace}(Q\boldsymbol{\Sigma}_k) + \mathbf{u}_{k+i-1}^T R \mathbf{u}_{k+i-1}\end{aligned} \tag{14}$$

Then from (9), $\mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] = \sum_{i=1}^{H} \mathcal{H}_i$. The gradients $d\mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right]/d\mathbf{u}_{k-1}$ can be expressed, using the chain-rule, as

$$\frac{d}{d\mathbf{u}_{k-1}} \mathbb{E}\left[\mathcal{J}(\mathbf{x}_k, \mathbf{u}_{k-1})\right] = \sum_{i=1}^{H} \frac{d\mathcal{H}_i}{d\mathbf{u}_{k+i-1}} \tag{15}$$

and

$$\begin{aligned}\frac{d\mathcal{H}_i}{d\mathbf{u}_{k+i-1}} =& \frac{\partial \mathcal{H}_i}{\partial \boldsymbol{\mu}_{k+i}} \frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \mathbf{u}_{k+i-1}} \\ &+ \frac{\partial \mathcal{H}_i}{\partial \boldsymbol{\Sigma}_{k+i}} \frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \mathbf{u}_{k+i-1}} + \frac{\partial \mathcal{H}_i}{\partial \mathbf{u}_{k+i-1}}\end{aligned} \tag{16}$$

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

$$\begin{aligned}\frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \mathbf{u}_{k+i-1}} &= \frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}} \frac{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}} \\ \frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \mathbf{u}_{k+i-1}} &= \frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}} \frac{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}}\end{aligned} \tag{17}$$

where $\frac{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}}$ and $\frac{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}}{\partial \mathbf{u}_{k+i-1}}$ can be easily obtained as well. More details about computations of $\frac{\partial \boldsymbol{\mu}_{k+i}}{\partial \tilde{\boldsymbol{\mu}}_{k+i-1}}$ and $\frac{\partial \boldsymbol{\Sigma}_{k+i}}{\partial \tilde{\boldsymbol{\Sigma}}_{k+i-1}}$ can be found in [17]. Algorithm 1 summarizes our proposed optimization method using analytical gradients at each iteration of MPC optimization.

IV. Numerical Simulations

The performance of proposed approach is verified by simulations on two trajectory tracking problems of a Multiple-Input Multiple-Output (MIMO) LTV system. All simulations

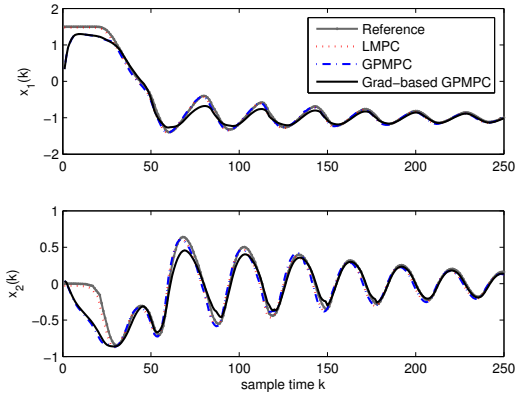

Fig. 1: Controlled states of using GP model based MPC in "Duffing" trajectory tracking problem

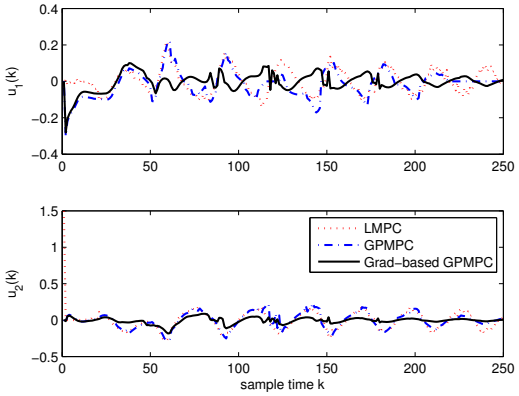

Fig. 2: Controls of using GP model based MPC in "Duffing" trajectory tracking problem

are conducted 50 times on a computer with a 3.40GHz Intel® Core™ 2 Duo CPU with 16 GB RAM, using Matlab® version 8.1.

The LTV numerical example [30] used in this paper is given as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} a(t) & 1 \\ b(t) & 0 \end{bmatrix} \mathbf{u} + \mathbf{w}_k \tag{18}$$

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 Gaussian noises $\mathbf{w} \sim \mathcal{N}(0, 0.01)$ in all simulations.

A. "Duffing" Trajectory Tracking

In the first simulation, the time-varying parameters are defined as $a(t) = 1 + \sin(2\pi t/1500)$ and $b(t) = \cos(2\pi t/1500)$. To collect observations, the system firstly is controlled to follow the "Duffing" trajectory (shown as grey dotted line in Figure 1) through using linear MPC approach proposed in [31]. 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 Mean Squared Error (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 learned GP models are then used to predict future system responses in the tracking problem. Theoretically, a "sufficiently long" prediction horizon H is required to guarantee stability and feasibility of using MPC scheme [32]. The approach to estimate required H is discussed in [33]. 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 issue discussed in Section II and III. 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) are solved by using derivative-free approach in [34], and proposed gradient-based algorithm.

The controlled states and controls in the first trajectory tracking problem are given in Figure 1 and 2 respectively. Where "Grad-based GPMPC" denotes 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 1, "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 controls shown in Figure 2 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 3. 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.

B. "Lorenz" Trajectory Tracking

In the second simulation, the control task is tracking a 2D "Lorenz" trajectory, shown in Figure 4. The time-varying parameters are now defined as $a(t) = 1/t - \exp(-t^2)/t$ and $b(t) = 1$. Similarly, we use a linear MPC strategy firstly to conduct a "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

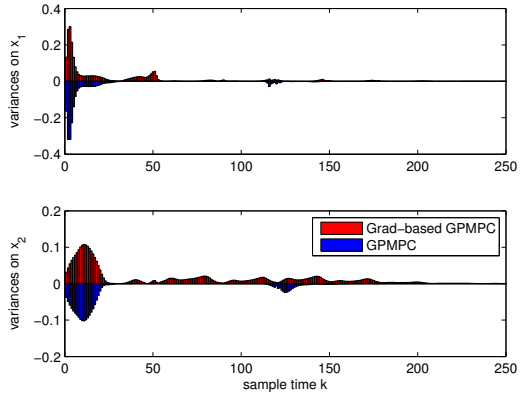

Fig. 3: Uncertainty propagation over whole trajectory in "Duffing" trajectory tracking problem

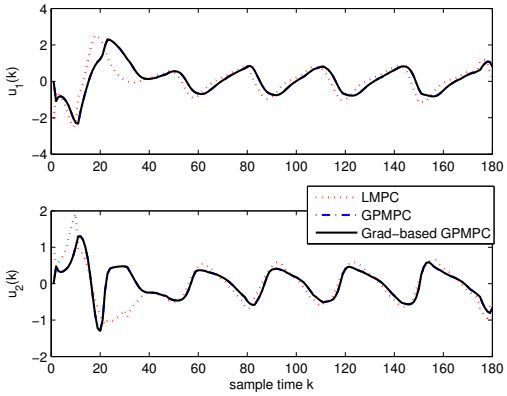

Fig. 5: Controls of using GP model based MPC in the "Lorenz" trajectory tracking problem

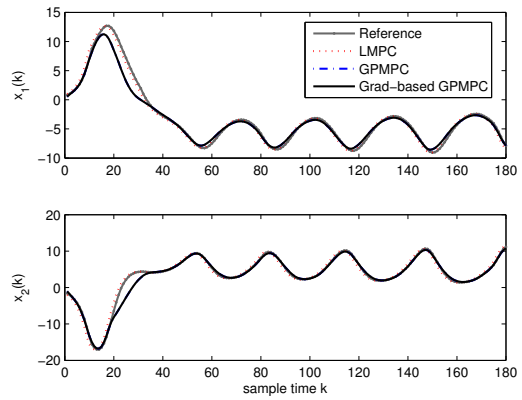

Fig. 4: Controlled states of using GP model based MPC in the "Lorenz" trajectory tracking problem

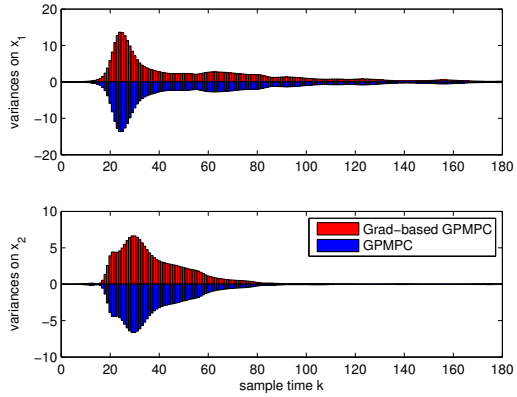

Fig. 6: Uncertainty propagation over whole trajectory in the "Lorenz" trajectory tracking problem

0.57s. In addition, we obtain 8.7979×10^{-5} training and 4.0674×10^{-4} testing MSE values.

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

As shown in Figure 4, 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. According to simulation results, two proposed approaches equally are able to produce controlled states that are overall close to target trajectory, even though bigger tracking MSE values they produced. 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 GPMPC" approach. In addition, as shown in Figure 5, obtained controls of using two approaches are approximately equal as well. The same situation again happens when we compute average predicted variances given in Figure 6. 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 "Grad-based GPMPC" outperforms than "GPMPC" with respect to computational efficiency.

V. Conclusion

A MPC strategy based on the probabilistic GP models is proposed. The proposed algorithm directly takes model uncertainties obtained during GP predictions into account when calculating MPC controls. This is a direct way and reduces computational burdens compared to most existing GP based stochastic MPC approaches. In addition, through using analytical gradients that are available when using GP models, the optimization problem is solved more efficiently. The simulation results on the trajectory tracking problem of a LTV system demonstrate well modelling and control performances of proposed MPC method, as well as the efficiency of proposed gradient based optimization algorithm.

REFERENCES

[1] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Y. Huang, H. H. Lou, J. Gong, and T. F. Edgar, "Fuzzy model predictive control," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 665–678, 2000.

[7] S. Mollov, R. Babuška, J. Abonyi, and H. B. Verbruggen, "Effective optimization for fuzzy model predictive control," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 5, pp. 661–675, 2004.

[8] D. Limon, T. Alamo, D. Raimondo, D. M. de la Pena, J. Bravo, A. Ferramosca, and E. Camacho, "Input-to-state stability: a unifying framework for robust model predictive control," in *Nonlinear model predictive control*. Springer, 2009, pp. 1–26.

[9] P. Scokaert and D. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[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] 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.

[15] 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.

[16] 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.

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

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

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

[20] D. J. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[21] G. Cao, E. M. 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.

[22] B. Kouvaritakis and M. Cannon, "Stochastic model predictive control," *Encyclopedia of Systems and Control*, pp. 1–9, 2014.

[23] 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.

[24] 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.

[25] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.

[26] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[27] Y. Yuan, "Step-sizes for the gradient method," *AMS IP Studies in Advanced Mathematics*, vol. 42, no. 2, p. 785, 2008.

[28] C. Kirches, *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Springer, 2011.

[29] 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.

[30] 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.

[31] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[32] 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.

[33] 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.

[34] 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.