Gang Cao\textsuperscript{1,3}, Edmund M.-K. Lai\textsuperscript{2}, Fakhrul Alam\textsuperscript{1}
\textsuperscript{1}School of Engineering and Advanced Technology, Massey University, Auckland, New Zealand
\textsuperscript{2}Department of Information Technology and Software Engineering, Auckland University of Technology, Auckland, New Zealand
\textsuperscript{*E-mail: g.cao@massey.ac.nz}

Abstract: Model predictive control (MPC) of an unknown system that is modelled by Gaussian process (GP) techniques is studied. 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 study, 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 non-linear MPC based on a basic linearised GP local model. The resulting optimisation problem, though non-convex, can be solved by the sequential quadratic programming. 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 optimised control inputs by repeatedly solving a finite horizon optimisation 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 non-linear 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.

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 optimisation 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 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 maximising the log-likelihood function. This optimisation problem is unconstrained, non-linear and non-convex optimisation. It is typically solved by conjugate gradient (CG) [15] or by particle swarm optimisation (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 non-linear optimisation 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 optimisation 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 non-linear 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 optimisation problem is solved by using a sequential quadratic programming (SQP) based method together with a linearised 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 non-linear MPC problem is reformulated to a convex optimisation problem. In contrast with earlier methods, GP
variances are directly included in the cost function of the optimisation. 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 organised as follows. Section 2 introduces the modelling of the unknown non-linear 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 GP-MPC1 and GP-MPC2 are presented for the general trajectory tracking problem of the unknown non-linear 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 non-linear dynamical system described by the following general form:

\[ x_{k+1} = f(x_k, u_k) + w_k \]

(1)

where \( f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n \) is a non-linear function, \( w \in \mathbb{R}^n \) represents additive external disturbances, \( x \in \mathbb{R}^n \) denotes the state vector, and \( u \in \mathbb{R}^m \) 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 are 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 \( x_k = (x_k, u_k) \in \mathbb{R}^{n+m} \) and the next state \( x_{k+1} \), respectively. Let \( \Delta x_k = x_{k+1} - x_k \in \mathbb{R}^n \). In practice, the variation between \( \Delta x_{k+1} \) and \( \Delta x_k \) is much less the variation between \( x_{k+1} \) and \( x_k \) for all \( k \). Therefore it is more advantageous to use \( \Delta x_k \) 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 \( \bar{x}_k \) is zero, the squared exponential covariance is given by

\[ K(x_k, \bar{x}_k) = \sigma^n_x^2 \exp(-\frac{1}{2}(x_k - \bar{x}_k)^T \Lambda (x_k - \bar{x}_k)) + \sigma^n_n \]

where \( \sigma^n_x, \sigma^n_n \) and the entries of matrix \( \Lambda \) are referred to as the hyperparameters \( \theta \) of a GP model. Given \( D \) training inputs \( \bar{X} = [\bar{x}_1, \ldots, \bar{x}_D] \), and their corresponding training targets \( \bar{y} = [\Delta y_1, \ldots, \Delta y_D]^T \), the joint distribution between \( \bar{y} \) and a test target \( \Delta x_k^{\ast} \) for training input \( \bar{x}_k^{\ast} \) is assumed to follow a Gaussian distribution. That is,

\[ p(\bar{y} | \Delta x_k^{\ast}) \sim \mathcal{N}(0, K(\bar{x}_k^{\ast}, \bar{X}) + \sigma^n_n \mathbf{I}) \]

(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:

\[ m(\Delta x_k^{\ast}) = E_{\bar{y} \mid \Delta x_k^{\ast}} = K(\Delta x_k^{\ast}, \bar{X})K^{-1}(\bar{X}, \bar{X})\bar{y} \]

(3a)

\[ \sigma^2(\Delta x_k^{\ast}) = \text{Var}_{\bar{y} \mid \Delta x_k^{\ast}} = K(\Delta x_k^{\ast}, \Delta x_k^{\ast}) - K(\Delta x_k^{\ast}, \bar{X})K^{-1}(\bar{X}, \bar{X})K(\Delta x_k^{\ast}, \bar{X}) \]

(3b)

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

\[ p(x_{k+1} \mid \Delta x_k^{\ast}) \sim \mathcal{N}(\mu_{k+1}^{\ast}, \Sigma_{k+1}^{\ast}) \]

(4)

where

\[ \mu_{k+1}^{\ast} = x_k + m(\Delta x_k^{\ast}) \]

(5a)

\[ \Sigma_{k+1}^{\ast} = \sigma^2(\Delta x_k^{\ast}) \]

(5b)

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

\[ \log p(y | x, \theta) = -\frac{1}{2} y^T K^{-1} y - \frac{1}{2} \log |K| - D \log(2\pi) \]

(6)

This results in a non-linear non-convex optimisation problem that is traditionally solved by using CG or Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms. Recently, PSO based algorithms that minimises 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,

\[ p(\Delta x_k) = p(x_k, u_k) \sim \mathcal{N}(\bar{\mu}_k, \bar{\Sigma}_k) \]

(7)

with mean and variance given by

\[ \bar{\mu}_k = [\mu_k, E[u_k]]^T \]

(8a)

\[ \bar{\Sigma}_k = \begin{bmatrix} \Sigma & \text{Cov}[x_k, u_k] \\ \text{Cov}[u_k, x_k] & \text{Var}[u_k] \end{bmatrix} \]

(8b)

where \( \text{Cov}[x_k, u_k] = E[x_k u_k] - \mu_k E[u_k] \). Here, \( E[u_k] \) and \( \text{Var}[u_k] \) 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:

\[ p(\Delta x_k) \sim \int p(f(\Delta x_k) | \bar{x}_k) p(\bar{x}_k) d\bar{x}_k \]

(9)

However, this integral is analytically intractable. Numerical solutions can be obtained using MC 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

\[ m(\Delta x_k) = E(\Delta x_k \mid E[\Delta x_k]) \]

(10a)

\[ \sigma^2(\Delta x_k) = E[\text{Var}(\Delta x_k \mid E[\Delta x_k])] + \text{Var}[E(\Delta x_k)] \]

(10b)
Equation (5) then becomes
\[ \mu_{k+1} = \mu_k + m(x_k) \tag{11a} \]
\[ \Sigma_{k+1} = \Sigma_k + \sigma^2(x_k) + \text{Cov}[x_k, \Delta x_k] + \text{Cov}[\Delta x_k, x_k] \tag{11b} \]

The computational complexity of GP inference using (10) is \( O(n^3(n + m)) \) 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 non-linear systems, it is common practice to obtain local linearised models of the system around operating points. The main purpose is to reduce the computation involved in the non-linear control problem. The same technique is used here for the GP based MPC optimisation 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 linearised.

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 non-linear optimisation problems considered are unconstrained.

In this section, we shall present two different GP based local models. They will be applied to the constrained non-linear problems presented in Section 4.

#### 3.1 Basic GP-based local model

Linearisation can be done based on the mean values in the GP model. In this case we replace the state vector \( x_k \) by its mean \( \mu_k \). Then (1) becomes

\[ \mu_{k+1} = \mathcal{F}(\mu_k, u_k) \tag{12} \]

Let \( (\mu_k^*, u_k^*) \) be the operating point at which the linearised model is to be obtained. Given that \( \Delta \mu_k = \mu_k - \mu_k^* \) and \( \Delta u_k = u_k - u_k^* \) are small, from (12), we have

\[ \Delta \mu_{k+1} = \frac{\partial \mathcal{F}}{\partial \mu_k} \Delta \mu_k + \frac{\partial \mathcal{F}}{\partial u_k} \Delta u_k \tag{13a} \]

\[ = \frac{\partial \mu_{k+1}}{\partial \mu_k} \Delta \mu_k + \frac{\partial \mu_{k+1}}{\partial u_k} \Delta u_k \tag{13b} \]

where \( \frac{\partial \mu_{k+1}}{\partial \mu_k} \) and \( \frac{\partial \mu_{k+1}}{\partial u_k} \) are the Jacobian state and input matrices, respectively. Using the chain rule, we get

\[ \frac{\partial \mu_{k+1}}{\partial \mu_k} = \frac{\partial \mu_{k+1}}{\partial \hat{\mu}_k} \frac{\partial \hat{\mu}_k}{\partial \mu_k} + \frac{\partial \mu_{k+1}}{\partial \hat{\mu}_k} \frac{\partial \hat{\mu}_k}{\partial \mu_k} + \frac{\partial \mu_{k+1}}{\partial \hat{\mu}_k} \frac{\partial \hat{\mu}_k}{\partial \mu_k} \frac{\partial \hat{\mu}_k}{\partial \mu_k} \tag{14a} \]

\[ \frac{\partial \mu_{k+1}}{\partial u_k} = \frac{\partial \mu_{k+1}}{\partial \hat{\mu}_k} \frac{\partial \hat{\mu}_k}{\partial u_k} + \frac{\partial \mu_{k+1}}{\partial \hat{\mu}_k} \frac{\partial \hat{\mu}_k}{\partial u_k} + \frac{\partial \mu_{k+1}}{\partial \hat{\mu}_k} \frac{\partial \hat{\mu}_k}{\partial u_k} \frac{\partial \hat{\mu}_k}{\partial u_k} \tag{14b} \]

where \( \frac{\partial \hat{\mu}_k}{\partial \mu_k}, \frac{\partial \hat{\mu}_k}{\partial u_k}, \frac{\partial \hat{\mu}_k}{\partial u_k}, \frac{\partial \hat{\mu}_k}{\partial u_k} \) can be easily obtained based on (8). Elaborations of \( \frac{\partial \mu_{k+1}}{\partial \mu_k} \) and \( \frac{\partial \mu_{k+1}}{\partial u_k} \) can be found in [24].

### 3.2 Extended GP-based local model

Model uncertainties are characterised 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 \( x_k \) in (1) by \( s_k = [\mu_k, \text{vec}(\Sigma_k)]^T \in \mathbb{R}^{1 + n^2} \) which shall be known as the ‘extended state’. Here, \( \text{vec}(\cdot) \) denotes the vectorisation of a matrix \( \Sigma_k \) is a real symmetric matrix therefore can be diagonalised. The square root of a diagonal matrix can simply be obtained by computing the square roots of diagonal entries. Hence (1) becomes

\[ s_{k+1} = \mathcal{F}(s_k, u_k) \tag{15} \]

Linearising at the operating point \( (s_k^*, u_k^*) \) where \( s_k^* = [\mu_k^*, \text{vec}(\Sigma_k^*)]^T \), we have

\[ \Delta s_{k+1} = \frac{\partial \mathcal{F}}{\partial s_k} \Delta s_k + \frac{\partial \mathcal{F}}{\partial u_k} \Delta u_k \tag{16} \]

Here, \( \Delta s_k = s_k - s_k^* \) and \( \Delta u_k = u_k - u_k^* \). The Jacobian matrices are

\[ \frac{\partial \mathcal{F}}{\partial s_k} = \left[ \begin{array}{c} \frac{\partial \mu_{k+1}}{\partial s_k} \\ \frac{\partial \Sigma_{k+1}}{\partial s_k} \end{array} \right] \in \mathbb{R}^{1 + n^2} \times (1 + n^2) \tag{17a} \]

\[ \frac{\partial \mathcal{F}}{\partial u_k} = \left[ \begin{array}{c} \frac{\partial \mu_{k+1}}{\partial u_k} \\ \frac{\partial \Sigma_{k+1}}{\partial u_k} \end{array} \right] \in \mathbb{R}^{1 + n^2} \times m \tag{17b} \]

with the entries given by

\[ \frac{\partial \mu_{k+1}}{\partial s_k} = \frac{\partial \mu_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial s_k} + \frac{\partial \mu_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial s_k} \tag{18a} \]

\[ \frac{\partial \Sigma_{k+1}}{\partial s_k} = \frac{\partial \Sigma_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial s_k} + \frac{\partial \Sigma_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial s_k} \tag{18b} \]

\[ \frac{\partial \mu_{k+1}}{\partial u_k} = \frac{\partial \mu_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial u_k} + \frac{\partial \mu_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial u_k} \tag{18c} \]

\[ \frac{\partial \Sigma_{k+1}}{\partial u_k} = \frac{\partial \Sigma_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial u_k} + \frac{\partial \Sigma_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial u_k} \tag{18d} \]

Since

\[ \frac{\partial \Sigma_k}{\partial \Sigma_k} = \frac{1}{2 \sqrt{\Sigma_k}} \quad \text{and} \quad \frac{\partial \Sigma_{k+1}}{\partial \Sigma_{k+1}} = \frac{1}{2 \sqrt{\Sigma_{k+1}}} \]

they can be expressed as

\[ \frac{\partial \mu_{k+1}}{\partial \Sigma_k} = \frac{\partial \mu_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial \Sigma_k} + \frac{\partial \mu_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial \Sigma_k} \tag{19a} \]

\[ \frac{\partial \Sigma_{k+1}}{\partial \mu_k} = \frac{\partial \Sigma_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial \mu_k} + \frac{\partial \Sigma_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial \mu_k} \tag{19b} \]

\[ \frac{\partial \Sigma_{k+1}}{\partial \Sigma_k} = \frac{\partial \Sigma_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial \Sigma_k} + \frac{\partial \Sigma_{k+1}}{\partial u_k} \frac{\partial u_k}{\partial \Sigma_k} \tag{19c} \]
\[
\frac{\partial \Sigma_{k+1}}{\partial u_i} = \frac{\partial \Sigma_{k+1}}{\partial \mu_k} \frac{\partial \mu_k}{\partial u_i} + \frac{\partial \Sigma_{k+1}}{\partial \Sigma_k} \frac{\partial \Sigma_k}{\partial u_i}
\]  
(19d)

\(\mu_k/\partial \Sigma_k\) and \(\partial \Sigma_k/\partial \Sigma_k\) can be easily obtained based on (8). Elaborations of \(\partial \Sigma_{k+1}/\partial \mu_k\) and \(\partial \Sigma_{k+1}/\partial \Sigma_k\) can be found in [24].

### 4 Model predictive control based on GP

A discrete-time non-linear dynamical system defined by (1) is required to track a trajectory given by \(\{r_i\}\) for \(k = 1, 2, \ldots\). Using MPC with a prediction horizon \(H \geq 1\), the optimal control sequence can be obtained by solving the following problem:

\[
V^*_k = \min_{u_{k:i}} J(x_k, u_{k:i}, r_i) 
\]  
(20a)

subject to

\[
x_{k+i:k} = f(x_{k+i-1:k}, u_{k+i-1:k})
\]  
(20b)

\[
x_{\min} \leq x_{k+i:k} \leq x_{\max}
\]  
(20c)

\[
u_{\min} \leq u_{k+i-1:k} \leq u_{\max}
\]  
(20d)

where only the first control action \(u_1\) of the resulting control sequence \(u(\cdot) = [u_{k:i}, \ldots, u_{k+i-1:k}]^T\) is applied to the system at time \(k\). \(x_{\min} \leq x_{k} \leq x_{\max}\) and \(u_{\min} \leq u_{k} \leq u_{\max}\) are the upper and lower bounds of the system states and control inputs, respectively.

In the rest of this paper, the cost function \(J(x_k, u_{k:i}, r_i)\) shall be rewritten as \(J(x_k, u_{k:i})\) for brevity. The quadratic cost function given by

\[
J(x_k, u_{k:i}) = \sum_{i=1}^{H} \left\{ \left\| u_{k+i} - r_{k+i} \right\|_Q + \left\| u_{k+i} \right\|_R \right\} 
\]  
(21)

will be used. Here, \(\left\| \cdot \right\|_Q\) and \(\left\| \cdot \right\|_R\) denote the two 2-norms weighted by positive definite matrices \(Q \in \mathbb{R}^{m \times m}\) and \(R \in \mathbb{R}^{m \times m}\), 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]:

\[
V^*_k = \min \mathbb{E}[J(x_k, u_{k:i})] 
\]  
(22a)

subject to

\[
p(x_{k+i|k}, x_k) \sim GPMPC_1(\mu_{k:i}, \Sigma_{k+i})
\]  
(22b)

\[
u_{\min} \leq u_{k+i:i} \leq u_{\max}
\]  
(22c)

\[
p(x_{k+i:k} \geq x_{\min}) \geq \eta
\]  
(22d)

\[
p(x_{k+i:k} \leq x_{\max}) \geq \eta
\]  
(22e)

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

\[
\mu_{k+1} - 2\Sigma_{k+1} \geq x_{\min}
\]  
(23a)

\[
\mu_{k+1} + 2\Sigma_{k+1} \leq x_{\max}
\]  
(23b)

Using (21) as the cost function, we get

\[
\mathbb{E}[f(x_k, u_{k:i})] = \sum_{i=1}^{H} \left\{ \mathbb{E}[\left\| u_{k+i} - r_{k+i} \right\|_Q + \left\| u_{k+i} \right\|_R] \right\}
\]  
(25)

In practice, the controls are deterministic. Hence, \(\mathbb{E}[u_{k:i}] = u^*_{k:i}\) and (25) becomes (see (25))

Now we have a deterministic cost function which involves the model variance \(\Sigma\), that allows model uncertainties to be explicitly included in the computation of the optimised 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 Non-linear optimisation solution

With the cost function (25) and the state constraints (23), the original stochastic optimisation problem (22) has been relaxed to a deterministic constrained non-linear optimisation problem. However, 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 optimisation problems. However, there is no guarantee that the solutions obtained are near optimum. A review of non-linear optimisation 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 (FPSQP) algorithm proposed in [37]. It can be explained using the following general form of a constrained non-linear optimisation problem:

\[
\min \quad h(z) 
\]  
(26a)

s.t.

\[
c(z) = 0
\]  
(26b)

\[
d(z) \leq 0
\]  
(26c)

where \(h: \mathbb{R}^{n+m} \rightarrow \mathbb{R}\) is the objective function, \(c: \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n\) and \(d: \mathbb{R}^{n+m} \rightarrow \mathbb{R}^m\) represents the corresponding equality and inequality constraints, respectively. FPSQP generates a sequence of feasible solutions \(z^*_{j+1}\) by splitting the original problem into several quadratic programming (QP) sub-problems. In particular, a step \(\Delta z\) from current iterate \(z^*_j\) to the next one \(z^*_{j+1}\) can be obtained by solving the following QP sub-problem:

\[
\begin{align*}
\min & \quad \frac{1}{2} \Delta z^T M \Delta z + n(z_j) + c(z_j)^T \Delta z \\
\text{s.t.} & \quad \Delta z + c(z_j) = 0 \\
& \quad d(z_j) + \Delta z \leq 0
\end{align*}
\]  
(24)
\begin{align*}
\min_{\Delta z'} & \nabla h(z')^T \Delta z' + \frac{1}{2} \Delta z'^T H \Delta z' \\
\text{s.t.} & \ c(z') + \nabla c(z')^T \Delta z' = 0 \\
\quad & d(z') + \nabla d(z')^T \Delta z' \leq 0
\end{align*}

(27a)

under the trust-region constraint

$$\| \Delta z' \| \leq \gamma \quad (28)$$

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

$$z' + \Delta \tilde{z}' \in \Pi \quad (29a)$$

$$\frac{1}{2} \| \Delta z' \| \leq \| \Delta \tilde{z}' \| \leq \frac{3}{2} \| \Delta z' \| \quad (29b)$$

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

$$\rho' = \frac{h(z'^{+}) - h(z')}{-\nabla h(z')^T \Delta z' - \frac{1}{2} \Delta z'^T H \Delta z'} \quad (30)$$

If this value is not acceptable, then the trust-region radius \( \gamma' \) will need to be adjusted. An adaptive method to adjust \( \gamma' \) can be found in [39]. The complete FPSQP algorithm is described in Algorithm 1.

Algorithm 1 Feasibility-perturbed sequential quadratic programming used in the GPMPC1 algorithm:

1. **Initialisation**
   - feasible point \( z' \in \Pi \),
   - Hessian matrix \( H' \),
   - trust region upper bound \( \gamma_{max} > 0 \),
   - initial trust region radius \( \gamma = \| \nabla h(z') \| \), \( \tau = 0, 0 < \tau_1 < \tau_2 < 1 \).
2. for \( j = 0, 1, 2, \ldots, J < \infty \)
3. Obtain step \( \Delta z' \) by solving the problem (27);
4. If \( \nabla h(z')^T \Delta z' + \frac{1}{2} \Delta z'^T H \Delta z' = 0 \) then
5. Stop;
6. else
7. Update \( \gamma' \) by using (30);
8. Update \( z'^+ \), \( \tau' \):

\begin{align*}
\Delta \tilde{z}' &= z' + \Delta \tilde{z}' \\
\tau' &= \frac{\| \Delta \tilde{z}' \|}{\| \nabla h(z'^{+}) - h(z') \|} \quad \rho' \geq \tau_1
\end{align*}

(31)

9. Update trust region radius:

$$H_{k+1} = H_k - \frac{\Delta z_k \Delta z_k^T H_k}{\Delta z_k^T H_k \Delta z_k} + \frac{\sum_{j=1}^{n} \beta_j \nabla \nabla \theta(z_k)}{\sum_{j=1}^{n} \beta_j \| \nabla \theta(z_k) \|^2} \quad (35)$$

10. Update Hessian matrix \( H'_{k+1} \) by using (35);
11. \( j = j + 1; \)
12. end
13. end
where $\Delta z_k = z_{k+1} - z_k$ and $y_k = \mu_{k+1} - \mu_k$.

### 4.2 GPMPC2

With GPMPC1, model uncertainty was introduced through the variance term into the objective function in (25). However, 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 optimisation process.

Another disadvantage of GPMPC1 is that the MPC optimisation problem (22) is non-convex. Due to the recursive nature of SQP optimisations, 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 optimisation problem can be solved offline by using multi-parametric quadratic programs (MPQP) [41] where the explicit solutions are computed as a lookup table of non-linear optimisations, the process could be time consuming. With GPMPC1, model uncertainty was introduced through the extended local model, with the state and control

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

$$Z_{k+1} = [s_{k+1}; \ldots; s_{k+H}]^T \in \mathbb{R}^{Hn + s_H^2}$$

Moreover, let

$$U_k = [u_k; \ldots; u_{k+H}]^T \in \mathbb{R}^{Hn^2}$$

$$r_{k+1}^T = [r_{k+1}; 0; \ldots; r_{k+H}; 0]^T \in \mathbb{R}^{Hn^2}$$

Problem (22) then becomes

$$\min \| Z_{k+1} - r_{k+1}^T \|_Q + \| U_k \|_R^2$$

s.t. $M_{Hn}x_{\text{max}} \leq M_{Hn}z_{\text{max}} \leq M_{Hn}x_{\text{min}}$

$$I_{Hn^2}u_{\text{max}} \leq U_k \leq I_{Hn^2}u_{\text{min}}$$

where (see (40a))

$$\tilde{R} = \text{diag}([R; \ldots; R]) \in \mathbb{R}^{Hn^2 \times Hn^2}$$

$$L_g \in \mathbb{R}^n$$ is the identity vector, and

$$M_z = \begin{bmatrix} 1^T & 21^T & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1^T & 21^T & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1^T \\ 21^T & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \in \mathbb{R}^{Hn \times (Hn + n^2)}$$

Let $T_{\phi} \in \mathbb{R}^{Hn \times Hn^2}$ be a lower triangular matrices with unit entries. Then,

$$U_k = I_{Hn}u_{\text{max}} + T_{\phi} \Delta U_k$$

$$\Delta Z_{k+1} \text{ can be expressed as}$$

$$\Delta Z_{k+1} = \tilde{A} \Delta s_k + \tilde{B} \Delta U_k$$

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

$$\tilde{A} = [A, A^2, \ldots; A^H]^T \in \mathbb{R}^{Hn^2 \times Hn^2}$$

$$\tilde{B} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{H-2}B & A^{H-3}B & \cdots & B \end{bmatrix} \in \mathbb{R}^{Hn^2 \times Hn^2 \times Hn^2}$$

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

$$Z_{k+1} = s_k + T_{\phi}(\Delta s_k + \tilde{B} \Delta U_k)$$

where $T_{\phi} \in \mathbb{R}^{Hn^2 \times Hn^2 \times Hn^2}$ denotes a lower triangular matrix with unit entries.

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

$$\min \frac{1}{2\psi} \| \Delta U_k \|_Q^2 + \psi \| \Delta U_k \|_R^2 + C$$

s.t. $\Delta U_{\text{max}} \leq T_{\phi} \Delta U_k \leq \Delta U_{\text{max}}$

where

$$\Phi = \tilde{B}^T \tilde{Q} \tilde{T} \tilde{B} + T_{\phi}^T R_{\phi} \in \mathbb{R}^{Hn^2 \times Hn^2}$$

$$\psi = 2(s_k \tilde{Q} T \tilde{B} + \Delta s_k \tilde{A}^T \tilde{Q} \tilde{B})$$

$$\tilde{R} = (s_k \tilde{Q} + \tilde{B} \Delta s_k \tilde{T} \tilde{Q} \tilde{B})$$

$$C = (s_k + r_{k+1}^T) \tilde{Q} + 2(s_k \tilde{A} \Delta s_k \tilde{T} \tilde{Q} \tilde{B})$$

$$\Delta U_{\text{max}} = \begin{bmatrix} I_{Hn^2}u_{\text{max}} - u_{k+1} \\ I_{Hn^2} \Delta s_k - T_{\phi} \Delta s_k \\ \vdots \\ I_{Hn^2} \Delta s_k - T_{\phi} \Delta s_k \end{bmatrix}$$

$$\Delta U_{\text{min}} = \begin{bmatrix} I_{Hn^2}u_{\text{min}} - u_{k+1} \\ I_{Hn^2} \Delta s_k - T_{\phi} \Delta s_k \end{bmatrix}$$

Since $\tilde{Q}, \tilde{R}, T_{\phi}$, and $T_{\text{min}}$ are positive definite, $\Phi$ is also positive definite. Hence (46) is a constrained OP 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 optimisation 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 OP 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 optimisation process substantially.

Let \( G = [T_e, T_B]^T \), the constraint (46b) can be written as

\[
\begin{bmatrix}
G \\
-G
\end{bmatrix} \Delta U \leq \begin{bmatrix}
\Delta U_{\text{max}} \\
-\Delta U_{\text{min}}
\end{bmatrix}
\]

(48)

Ignoring the constant term C, problem (46) becomes

\[
\min_{\Delta U} \frac{1}{2} \| \Delta U \|_G^2 + \psi^T \Delta U
\]

s.t. \( \Delta U \leq \Delta U \)

(49a)

where \( G = [G_r - G_i]^T \in \mathbb{R}^{2H(m+n)\times Hn} \) and \( \Delta U = [\Delta U_{\text{max}} - \Delta U_{\text{min}}]^T \in \mathbb{R}^{2H(m+n)\times Hn} \).

Let \( \Pi_{\Delta U} \) be the set of feasible points, and \( \mathcal{J} = \{1, \ldots, 2H(m+n+n)\} \) be the constraint index set. For a feasible point \( \Delta U^i \in \Pi_{\Delta U} \), the index set for the active set of constraints is defined as

\[
\mathcal{A}(\Delta U^i) = \{ i \in \mathcal{J} | \tilde{G}_i \Delta U^i = \tilde{\Delta}_U \}
\]

(50)

where \( \tilde{G}_i \) is the ith row of \( G \) and \( \tilde{\Delta}_U \) is the ith row of \( \Delta U \). The inactive set is therefore given by

\[
\mathcal{B}(\Delta U^i) = \mathcal{J} \setminus \mathcal{A}(\Delta U^i) = \{ i \in \mathcal{J} | \tilde{G}_i \Delta U^i < \tilde{\Delta}_U \}
\]

(51)

Given any iteration \( j \), the working set \( \mathcal{W}_j \) 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. \( \mathcal{W}_j \) is considered given the feasible points \( \Delta U^i \in \Pi_{\Delta U} \):

\[
\min_{\delta^j} \frac{1}{2} \| \Delta U^j + \delta^j \|_G^2 + \psi^T (\Delta U^j + \delta^j)
\]

(52a)

\[
= \min_{\delta^j} \frac{1}{2} \| \delta^j \|_G^2 + (\psi + \Phi \Delta U^j)^T \delta^j
\]

(52b)

\[
= \min_{\delta^j} \frac{1}{2} \| \delta^j \|_G^2 + (\psi + \Phi \Delta U^j)^T \delta^j
\]

(52c)

s.t. \( \tilde{G}_i (\Delta U^j + \delta^j) = \tilde{\Delta}_U, i \in \mathcal{W}_j \)

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

\[
\min_{\delta^j} \frac{1}{2} \| \delta^j \|_G^2 + (\psi + \Phi \Delta U^j)^T \delta^j
\]

(53a)

\[
= \min_{\delta^j} \frac{1}{2} \delta^T \Phi \delta^j + (\psi + \Phi \Delta U^j)^T \delta^j
\]

(53b)

s.t. \( \tilde{G}_i \delta^j = \tilde{\Delta}_U, i \in \mathcal{W}_j \)

(53c)

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

\[
\begin{bmatrix}
\Phi & \tilde{G}_i^T \\
\tilde{G}_i & 0
\end{bmatrix}
\begin{bmatrix}
\delta^j \\
\lambda^j_i
\end{bmatrix}
= \begin{bmatrix}
-\psi - \Phi \Delta U^j \\
\tilde{\Delta}_U - \tilde{G}_i \Delta U^j
\end{bmatrix}
\]

(54)

where \( \lambda^j_i \in \mathbb{R}^{2H(m+n+n)} \) denotes the vector of Lagrangian multipliers, \( \tilde{G}_i \leq \tilde{G} \) and \( \tilde{\Delta}_U \leq \tilde{\Delta}_U \) are the weighting matrix and the upper bounds of the constraints w.r.t. \( \mathcal{W}_j \). Let the inverse of the Lagrangian matrix be denoted by

\[
\begin{bmatrix}
\Phi & \tilde{G}_i \lambda^j_i \\
\tilde{G}_i & 0
\end{bmatrix}
= \begin{bmatrix}
L_1 & L_1 \\
L_2 & L_2
\end{bmatrix}
\]

(55)

If this inverse exists, then the solution is given by

\[
\delta^j = -L_1 (\psi + \Phi \Delta U^j) + L_1^T (\tilde{\Delta}_U - \tilde{G}_i \Delta U^j)
\]

(56a)

\[
\lambda^j_i = -L_2 (\psi + \Phi \Delta U^j) + L_2 (\tilde{\Delta}_U - \tilde{G}_i \Delta U^j)
\]

(56b)

where

\[
L_1 = \Phi^T - \Phi^T \tilde{G}_i (\tilde{G}_i \Phi^T \tilde{G}_i)^{-1} \tilde{G}_i \Phi^T
\]

(57a)

\[
L_2 = \Phi^T \tilde{G}_i (\tilde{G}_i \Phi^T \tilde{G}_i)^{-1}
\]

(57b)

\[
L_1 = - (\tilde{G}_i \Phi^T \tilde{G}_i)^{-1}
\]

(57c)

If \( \delta^j \neq 0 \), then the set of feasible points \( \Delta U^j \) fails to minimise problem (49). In this case, the next set of feasible point is computed for the next iteration by \( \Delta U^{j+1} = \Delta U^j + \kappa \delta^j \) with step size

\[
\kappa^j = \min \left\{ 1, \min_{i \in \mathcal{W}(\Delta U^j)} \frac{\tilde{\Delta}_U^j - \tilde{G}_i \Delta U^j}{G \delta^j} \right\}
\]

(58)

If \( \kappa^j < 1 \), the inequality constraint with index

\[
q = \arg \min_{i \in \mathcal{W}(\Delta U^j)} \frac{\tilde{\Delta}_U^j - \tilde{G}_i \Delta U^j}{G \delta^j}
\]

should be ‘activated’, giving the working set \( \mathcal{W}^{j+1} = \mathcal{W}_j \cup q \). Otherwise, we have \( \mathcal{W}^{j+1} = \mathcal{W}_j \).

Alternatively, if the solution gives \( \delta^j = 0 \), then the current feasible points \( \Delta U^j \) could be the optimal solution. This can be verified by checking the Lagrangian multiplier \( \lambda^j_i = \min_{i \in \mathcal{W}_j} \lambda^j_i \). If \( \lambda^j_i \geq 0 \), the optimal solution of the (49) at sampling time \( k \) is found. Otherwise, this inequality constraint indexed by \( p = \arg \min_{i \in \mathcal{W}(\Delta U^j)} \lambda^j_i \) should be removed from the current working set, giving us \( \mathcal{W}^{j+1} = \mathcal{W}_j \setminus p \). Algorithm 2 summarises the active set algorithm used in the GPMPC2.

Algorithm 2 Active set method for solving the GPMPC2 problem:

1. **Initialisation**
   - The feasible point \( \Delta U^0 \in \Pi_{\Delta U} \)
   - The working set \( \mathcal{W}^0 = \mathcal{A}(\Delta U^0) \)
2. for \( j = 0, 1, 2, \ldots \) do
3.   Compute the \( \delta^j \) and \( \lambda^j_i \) by solving the linear equations (54); 4.   If \( \delta^j = 0 \) then
5.     \( \lambda^j_i = \min_{i \in \mathcal{W}_j} \lambda^j_i \)
6.     Set \( p = \arg \min_{i \in \mathcal{W}_j} \lambda^j_i \)
7.     If \( \lambda^j_i \geq 0 \) then
8. \( \Delta U_k = \Delta U_k^* \)
9. Stop.
10. else
11. \( \mathcal{W}_{k+1} = \mathcal{W}_k \cup p \);
12. \( \Delta U_k = \Delta U_k^* \)
13. end
14. else
15. Compute the step length \( \kappa' \) by (58),
16. \( q = \arg\min_{i \in \mathcal{S}} \frac{\Delta e_{i+1} - G \Delta u_{ij}}{G b} \)
17. If \( \kappa' < 1 \) then
18. \( \Delta U_{k+1} = \Delta U_k + \kappa' \delta \)
19. \( \delta = C_1(\Delta U_k) \cup q \);
20. else
21. \( \Delta U_{k+1} = \Delta U_k + \delta \);
22. \( \delta = C_1(\Delta U_k) \);
23. end
24. end
25. end

4.2.3 Implementation issues: The key to solving the linear equations (54) is the inverse of the Lagrangian matrix. However, \( \mathcal{G}_k \) is not always full ranked. Thus the Lagrangian matrix is not always invertible. This problem can be solved by decomposing \( \mathcal{G}_k \) using QR factorisation technique, giving us \( \mathcal{G}_k^T = C_1[\mathcal{R}, 0]^T \) where \( \mathcal{R} \in \mathbb{R}^{m \times m} \) is an upper triangular matrix with \( m = \text{rank}(\mathcal{G}_k) \).

\( \mathcal{A} \in \mathbb{R}^{m \times H} \) is an orthogonal matrix that can be further decomposed to \( \mathcal{A} = [\mathcal{G}_1, \mathcal{G}_2] \) where \( \mathcal{G}_1 \in \mathbb{R}^{H \times m} \) and \( \mathcal{G}_2 \in \mathbb{R}^{H \times m - H} \). Thus, \( \mathcal{G}_k^T = \mathcal{G} = \mathcal{G}_1 \mathcal{R} \) and

\[
\begin{align*}
L_1 &= \mathcal{G}_1 \mathcal{R}_1^T \mathcal{R}_2^T + \mathcal{G}_2 \mathcal{R}_2^T \\
L_2 &= \mathcal{G}_1 \mathcal{R}_1^T - \mathcal{G}_2 \mathcal{R}_2^T \\
L_3 &= \mathcal{R}_2^T \mathcal{G}_1 \mathcal{R}_2^T 
\end{align*}
\]

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 \( \Delta U_k \) as the starting point \( \Delta U_{k+1} \) for the next sampling time \( k+1 \). This warm-start technique is usually employed in MPC optimisations 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

\[
\mathcal{V}(k) = \sum_{i=1}^{H} \left( \| \Delta u_{i+1}^k \|_Q^2 + \| u_{i+1}^k \|_R^2 + \text{trace}(Q \Sigma_{i+1}^k) \right)
\]

where \( \Delta u_{i+1}^k = u_{i+1}^k - r_{i+1} \), \( u^* \) is the optimal control inputs, and \( \mu_{i+1}^k \) and \( \Sigma_{i+1}^k \) 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.

\[
\mathcal{V}(k+1) = \mathcal{V}(k) + \sum_{i=1}^{H} \left( \| \Delta u_{i+1}^k \|_Q^2 + \| u_{i+1}^k \|_R^2 + \text{trace}(Q \Sigma_{i+1}^k) \right)
\]

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

\[
\begin{align*}
\mathcal{V}(k+1) &\leq \mathcal{V}(k+1) \\
&\leq \mathcal{V}(k) + \| \Delta u_{i+1}^k \|_Q^2 \\
&+ \| u_{i+1}^k \|_R^2 + \text{trace}(Q \Sigma_{i+1}^k)
\end{align*}
\]

because of \( \| \Delta u_{i+1}^k \|_Q^2 \geq 0 \), \( \| u_{i+1}^k \|_R^2 \geq 0 \) and \( \text{trace}(Q \Sigma_{i+1}^k) \geq 0 \).

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,

\[
\begin{align*}
\mu_{k+H+\omega} - r_{k+H+\omega} &= 0 \\
\Sigma_{k+H+\omega} &= 0 \\
u_{k+H+\omega} &= 0
\end{align*}
\]

However, it should be noted that, these newly added constraints altered the optimisation 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].
They show that both algorithms exhibit equally good control stability of MPC controllers. However, the complexity of MPC (NMPC) strategy proposed in [45]. 189 observations are collected more than 8 times more efficient than GPMPC1. This shows the target. The integral absolute error values can be found in Fig 9.9114 × 10−5. Theoretically, a long enough computational complexity.

5.1 ‘Step’ trajectory tracking

The objective of the first experiment is to steer the non-linear system to follow a step trajectory shown as the reference in Fig 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 non-linear 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 s, with a training mean squared error (MSE) of 9.9114 × 10−5. Fig. 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 \( x_c = [0, 0, 0, 0] \)T and initial control inputs \( u_c = [0, 0] \)T, weighting matrix \( Q = I_{4 \times 4} \) and \( R = I_{2 \times 2} \). 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 Figs. 2a and b, 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 values can be found in Fig 2c.

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

5.2 ‘Lorenz’ trajectory tracking

The second problem is to track a ‘Lorenz’ trajectory as shown in Fig. 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 s with a training MSE of 0.0196. Fig. 1b shows the training error.

The MPC parameters are: initial states \( x_c = [0, 0, 0, 0] \)T, initial control inputs \( u_c = [0, 0] \)T, prediction horizon \( H = 10 \), weighting matrix \( Q = I_{4 \times 4} \) and \( R = \text{diag}([21000, 27000]) \).

The tracking results can be found in Figs. 2d-f. 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 s versus 24.72 s).

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 Fig. 3. show that the both two proposed algorithms outperform than the non-linear GPMPC in the ‘Lorenz’ trajectory tracking problem. In addition, the GPMPC1 and GPMPC2 only require approximately 5 and 7 s to compute all 189 control actions, compared to 150 s used in non-linear 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. Fig. 4 shows how well each model track the reference outputs. Table 1 shows the 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 non-linear 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 optimisation 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 non-linear optimisation problem. The solution of the resultant problem is obtained using the FPSQP method based on a linearised 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 optimised controls. By using the extended linearised GP local model, the non-convex optimisation 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 optimisation problem will be different from the one considered in this paper. This issue will be addressed in future work.
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 |

7 References

[1] Mayne, D.Q., Rawlings, J.B., Rao, C.V. et al.: ‘Constrained model predictive control: stability and optimality’, Automatica, 2000, 36, (6), pp. 789–814

[2] Qin, S.J., Badgwell, T.A.: ‘A survey of industrial model predictive control technology’, Control Eng. Pract., 2003, 11, (7), pp. 733–746

[3] Mayne, D.Q.: ‘Model predictive control: recent developments and future promise’, Automatica, 2014, 50, (12), pp. 2976–2986

[4] Solomatine, D.P., Ostfeld, A.: ‘Data-driven modelling: some past experiences and new approaches’, J. Hydroinformatics, 2008, 10, (11), pp. 3–22

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

[6] Alamo, T., De La Peña, D.M., Limón, D.: ‘Constrained min-max predictive control: modifications of the objective function leading to polynomial complexity’, IEEE Trans. Autom. Control, 2005, 50, (3), pp. 710–714

[7] Limón, D., Alamo, T., Salas, F. et al.: ‘Input to state stability of min-max MPC controllers for nonlinear systems with bounded uncertainties’, Automatica, 2006, 42, (5), pp. 797–803

[8] Langson, W., Chryssochoos, I., Raković, S.: ‘Robust model predictive control based on Gaussian process models’. Int. Joint Conf. on Neural Networks (IJCNN), 2011, pp. 178–184

[9] Zhang, L., Zhuang, S., Braatz, R.D.: ‘A unifying view of sparse approximate Gaussian process regression’, J. Mach. Learn. Res., 2005, 6, pp. 1939–1959

[10] Schwarm, A.T., Nikolaou, M.: ‘Chance-constrained model predictive control: stability while improving performance’, IEEE/RSJ Proc. of Int. Conf. on Intelligent Robots and Systems (IROS), 2014

[11] Pan, Y., Theodorou, E.: ‘Probabilistic dynamic differential equations’, Adv. Neural Inf. Process. Syst., 2014, pp. 1907–1915

[12] Grancharova, A., Kocijan, J., Johansen, T.A.: ‘Explicit stochastic predictive control of combustion plants based on Gaussian process models’, Automatica, 2008, 44, (6), pp. 1621–1631

[13] Trottorsch, F.: ‘Regular Lagrange multipliers for control problems with mixed pointwise control-state constraints’, SIAM J. Optim., 2005, 15, (2), pp. 616–634

[14] Diehl, M., Ferreau, H.J., Haverbeke, N.: ‘Efficient numerical methods for nonlinear MPC and moving horizon estimation’. Int. Workshop on assessment and future directions on Nonlinear Model Predictive Control, 2008, pp. 391–417

[15] Lucidi, S., Scandicci, M., Tseng, P.: ‘Objective-derivative-free methods for constrained optimization’, Math. Program., 2002, 92, (1), pp. 37–59

[16] Liu, G., Lucidi, S., Sciandrone, M.: ‘Efficient nonlinear MPC and moving horizon estimation’. Int. Workshop on assessment and future directions on Nonlinear Model Predictive Control, 2008, pp. 391–417

[17] Grancharova, A., Kocijan, J., Johansen, T.A.: ‘Explicit stochastic predictive control of combustion plants based on Gaussian process models’, Automatica, 2008, 44, (6), pp. 1621–1631

[18] Trottorsch, F.: ‘Regular Lagrange multipliers for control problems with mixed pointwise control-state constraints’, SIAM J. Optim., 2005, 15, (2), pp. 616–634

[19] Yeniyi, L., Yeniyi, J., Yengi, J.: ‘An improved PSO algorithm for solving non-linear NLP/MINLP problems with equality constraints’, Comput. Eng., 2007, 31, (3), pp. 153–162

[20] Yeniyi, L.: ‘Penalty function methods for constrained optimization with genetic algorithms’, Math. Comput. Appl., 2005, 10, (1), pp. 45–56

[21] Wright, S.J., Tenny, M.J.: ‘A feasible trust-region sequential quadratic programming algorithm’, SIAM J. Optim., 2004, 14, (4), pp. 1074–1105

[22] Peng, Y.-h., Yao, S.: ‘A feasible trust-region algorithm for inequality constrained optimization’, Appl. Math. Comput., 2006, 173, (1), pp. 513–522

[23] Zhang, X., Zhang, J., Liao, L.: ‘An adaptive trust region method and its convergence’, Sci. China A Math., 2002, 45, (5), pp. 620–631

[24] Tenny, M.J., Wright, S.J., Rawlings, J.B.: ‘Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming’, Comput. Optim. Appl., 2004, 28, (1), pp. 87–121

[25] Alamo, T., Morari, M., Dua, V. et al.: ‘The explicit linear quadratic regulator for constrained systems’, Automatica, 2002, 38, (1), pp. 3–20

[26] Wang, Y., Boyd, S.: ‘Fast model predictive control using online optimization’, IEEE Trans. Control Syst. Technol., 2010, 18, (2), pp. 267–278

[27] Fletcher, R.: ‘Practical methods of optimization’ (Wiley-Interscience Publication, 1987, 2nd edn.)

[28] Pan, Y., Wang, J.: ‘Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks’, IEEE Trans. Ind. Electron., 2012, 59, (8), pp. 3089–3101

[29] Grüne, L., Pannek, J.: ‘Nonlinear model predictive control-theory and algorithms’ (Springer-Verlag, London, U.K., 2011)

[30] Kocijan, J., Murray-Smith, R.: ‘Nonlinear predictive control with a Gaussian process model’, Shorten, R., Murray-Smith, R. (eds.) ‘Switching and learning in feedback systems’ (Springer, Heidelberg, Berlin, Germany, 2005), pp. 185–200