TUHH
Hamburg
University of
Technology

ICS
Institute
of Control
Systems

Project work

Data Driven Model Predictive Control using Gaussian Processes

Author:
Harshith Gowda Shakaladevanapura
Maregowda

Supervisor: Jannis Lübsen, M.Sc.

Examiner: 1. Prof. Dr.-Ing Annika Eichler
2. Jannis Lübsen, M.Sc.

May 16, 2024

ICS Institute of Control Systems

Project Work **P — 2 — 2024**
for Mr. Harshith Gowda Shakaladevanapura Maregowda | Student ID: 565699 | Study Course: MEC

Title:
Data Driven Model Predictive Control using Gaussian Processes

Project Description:

Model predictive control (MPC) has gained more attention in the past years as it is a powerful modern control technique that relies on repeatedly solving an open-loop optimal control problem. In order to obtain high-quality optimal input signals, an accurate prediction model is required, which is often unavailable or difficult to identify. In recent years data-driven control approaches have become more interesting because they use only measured data to control unknown systems without prior model identification. A major issue in data-driven control is the treatment of measurement noise, which in recent publications is treated to be deterministically bounded [1]. However, this assumption does not generally hold for a real system. Alternatively, a stochastic model can be used to model the state transition function as in [2]. However, there is a lack of theoretical guarantees of probabilistic models, e.g., Gaussian processes.

The goal of this thesis is to implement a probabilistic MPC scheme relying on Gaussian process models to represent the transition function of a discrete-time system with noisy measurements and uncertainty models. The thesis proves that one probabilistic GP-MPC algorithm is enough for both linear and nonlinear models. A common challenge is that most data-driven approaches require numerous interactions with the environment. However, conducting a vast number of interactions can often be unfeasible in real-world scenarios, especially in fields like robotics[3]. The proposed GP-MPC model is capable of controlling any system with a small number of interactions, which increases the data efficiency even in an uncertainty model.

Tasks:

1. Literature review
2. Familiarization with Gaussian processes and MPC
3. To learn a probabilistic transition model using Gaussian Processes (GPs) to incorporate model uncertainty into long-term predictions
4. Implementation of the probabilistic MPC algorithm
5. Implementation of GP-MPC on linear and nonlinear dynamics.
6. Comparison to other MPC algorithms/implementations

7. Testing in simulation and, if successful, testing on a real plant(optional)
8. Evaluation and discussion

References:

[1] J. Berberich, J. Kohler, M. A. Muller, and F. Allgower, “Data-driven model predictive control with stability and robustness guarantees”, *IEEE Transactions on Automatic Control*, 2021.

[2] S. Kamthe and M. P. Deisenroth, *Data-efficient reinforcement learning with probabilistic model predictive control*, 2018.

[3] M. Deisenroth, D. Fox, and C. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 408–423, Feb. 2015. DOI: 10.1109/TPAMI.2013.218.

Supervisor: Jannis Lübsen, M.Sc.
Examiner: 1. Prof. Dr.-Ing Annika Eichler
2. Jannis Lübsen, M.Sc.

Deutscher Titel: Datengetriebene modelprädiktive Regelung mit Gaußschen Prozessen

Start Date: 15.01.2024
Due Date: 16.05.2024

15.05.2024, Prof. Dr.-Ing A. Eichler

Hereby I declare that I produced the present work myself only with the help of the indicated aids and sources.

Hamburg, May 16, 2024 Harshith Gowda Shakaladevanapura Maregowda

Abstract

Model Predictive Control (MPC) has emerged as a potent control technique in recent years, relying on solving open-loop optimal control problems iteratively. However, the necessity of accurate prediction models poses challenges, particularly when dealing with systems where such models are unavailable or difficult to identify. Data-driven control approaches offer an alternative, leveraging measured data to control unknown systems without prior model identification. Despite their promise, the treatment of measurement noise remains a significant challenge.

In this thesis, we propose an approach to address this challenge by implementing a probabilistic MPC scheme utilizing Gaussian process models to represent the transition function of a nonlinear time-discrete system with noisy measurements and uncertainties. We employ a multioutput Gaussian process model comprising independent subprocesses to capture the stochastic nature of the system dynamics. The GP-MPC model presented has the ability to effectively control various systems with minimal interactions, thereby enhancing data efficiency even within an uncertainty framework.

The thesis begins with a comprehensive literature review on Gaussian processes and probabilistic MPC, providing the necessary background for the proposed methodology. Subsequently, the methodology is put into practice and thoroughly evaluated through simulations encompassing various scenarios, including linear(DC motor) and nonlinear model(Van der Pol oscillator), uncertainties, and noisy measurements. Reference tracking for two distinct set points is implemented for the Van der Pol oscillator. All the GP-MPC results are compared to direct MPC results, where MPC is directly applied to the dynamics of the system. The obtained results are analyzed and discussed, shedding light on the effectiveness and applicability of the proposed probabilistic MPC approach.

The Git repository of our project work's code can be accessed via **this link**.

Contents

1 INTRODUCTION 1

1.1 Background . . . 1

1.2 Motivation . . . 1

1.3 Approach . . . 2

1.4 Outline . . . 3

2 Gaussian Processes 4

2.1 Linear Regression Model . . . 4

2.1.1 Posterior Distribution . . . 5

2.1.2 Prediction . . . 7

2.2 Extension to Non-Linear Functions . . . 8

2.3 Long-term Prediction . . . 9

2.4 Optimization of the Hyperparameters . . . 11

3 Model Predictive Control and Learning 14

3.1 Model Predicitve control . . . 14

3.1.1 MPC formulation . . . 15

3.1.2 Prediction Model . . . 16

3.1.3 Cost Function . . . 16

3.1.4 Optimal Control Problem . . . 17

3.1.5 Feasibility . . . 18

3.2 GP-MPC Framework . . . 22

3.2.1 GP Model Learning . . . 23

3.2.2 Controller Design . . . 25

4 Simulation Results 29

4.1 DC Motor . . . 29

4.1.1 Validation of DC Motor Learning . . . 31

4.1.2 Modelling Without Uncertainty . . . 35

4.1.3 Modelling With Uncertainty . . . 38

4.2 Van der Pol Oscillator . . . 41

4.2.1 Validation of Van der Pol Oscillator Learning . . . 43

4.2.2 Modelling Without Uncertainty 46
4.2.3 Modelling With Uncertainty 48
4.2.4 Reference Tracking of Van der Pol Oscillator 51

5 Conclusion and Outlook **56**

5.1 Conclusion 56
5.2 Future Research 57

1 INTRODUCTION

1.1 Background

In the model-based approach, having a mathematical representation of the plant is crucial for devising a specific controller. Modeling the plant is a pivotal yet challenging aspect of these methods. As outlined in [[1]], model identification serves as a means to obtain the plant model by utilizing data from experimental tests conducted on the actual system. Various identification techniques can be employed for this purpose: within the black-box framework, the model is directly derived from input-output data alone, while grey-box algorithms first establish a physically grounded model based on fundamental principles and then adjust the model parameters using experimental data. Nonetheless, regardless of the sophistication of the identification method employed, the model always serves as an approximation of the actual system, leading to inevitable errors. In addition, it's imperative to acknowledge the diverse landscape of machine learning models applied in control systems[2]. This research specifically delves into Gaussian Process (GP) model controls, amidst the plethora of techniques utilized within the field. By focusing on GP models, the study aims to explore their efficacy within the realm of black box modeling for nonlinear systems, particularly in the construction of Model Predictive Controllers (MPCs) to regulate system dynamics.

1.2 Motivation

Despite many recent advancements [3] [4], data-driven approaches face a significant drawback: their learning process is slow, requiring an impractical number of interactions with the environment to accurately model the system. This limitation poses a practical challenge in real-world systems like robots, where numerous interactions can be time-consuming and impractical. Such data inefficiency renders learning in control and robotic systems impractical and hinders the application of data-driven approaches in more complex scenarios. To enhance data efficiency, either task-specific prior knowledge or the extraction of more information from available data is required. However, obtaining task-specific prior knowledge is often impossible, especially in cases such as robotics. Instead, a viable alternative involves modeling observed dynamics using a flexible nonparametric approach. Generally, model-based methods, which involve learning an explicit dynamics model of the environment, hold more promise for efficiently extracting valuable information from available data compared to model-free methods. Despite their potential, model-based methods are not widely used due to their susceptibility to model errors. These methods inherently assume that the learned model accurately resembles the real environment, which can severely impact their effectiveness.

1.3 Approach

A promising approach to enhance the data efficiency of the Model-Based Learning Model without relying on task-specific prior knowledge is to learn models of the underlying system dynamics. When a high-quality model is available, it can effectively serve as a stand-in for the real environment. This means that beneficial policies can be derived from the model without the need for additional interactions with the actual system. However, accurately modeling the underlying transition dynamics presents a significant challenge and inevitably introduces model errors [5].

To address these model errors, probabilistic models have been proposed specifically the Probabilistic Gaussian process [6]. These models explicitly incorporate uncertainties about the system dynamics. By considering model uncertainty, Model-based learning algorithms can substantially reduce the number of interactions required with the real system. This approach acknowledges that while perfect models are often unattainable, probabilistic models offer a more robust framework for navigating the inherent uncertainties in the modeling process. Thus, leveraging probabilistic models can lead to more efficient learning and improved performance in learning tasks by better accommodating the inevitable inconsistencies between the learned model and the real system[7].

This probabilistic Gaussian process has a limitation in that it cannot accept Gaussian input and predict over the entire prediction horizon (detailed explanation in Section 2.3). To address this, a transition model is proposed for long-term prediction, which serves as input to the MPC controller. However, an open-loop controller cannot stabilize the system on its own. Therefore, obtaining a feedback controller is essential. Model Predictive Control (MPC) provides a practical framework for this purpose [8]. During interaction with the system, MPC determines an N-step open-loop control trajectory $u_0, \ldots, u_{N-1}$, starting from the current state x_t. Only the first control signal u_0 is applied to the system[9]. When the system transitions to x_{t+1}, the Gaussian Process (GP) model is updated with the newly available information, and MPC re-plans $u_0, \ldots, u_{N-1}$. This process effectively transforms an open-loop controller into an implicit closed-loop (feedback) controller by continuously re-planning N steps ahead from the current state, as illustrated in Figure 1.1.

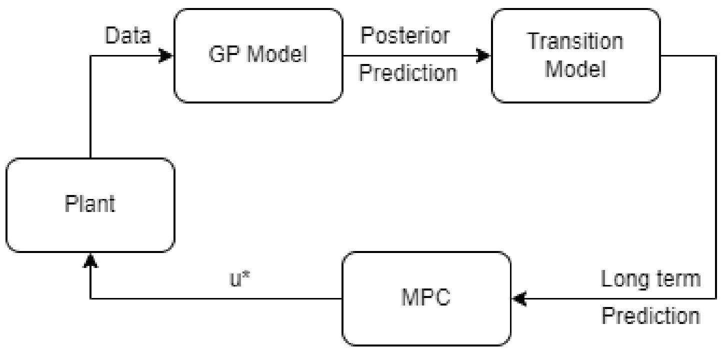


Figure 1.1: Block Diagram of GP-MPC Controller Framework

1.4 Outline

The thesis starts with the introduction of the utilized model, in Section 2, which mainly focuses on Gaussian process regression in supervised learning, and after that the literature concerning the application of Gaussian process regression in control and focus on the learning of dynamic models. Furthermore, necessary assumptions to guarantee the applicability of GPs are described. In addition, the GP transition model is introduced for long-term prediction with a detailed explanation. In Section 3, focuses on Model Predictive control and its basics. In addition, a detailed framework of the GP-MPC learning controller. Section 4 discusses the implementation and evaluation of the proposed approach through simulations on both linear and nonlinear systems. Finally, Section 5 provides a summary of the findings and results. It outlines directions for future research in the field of data-efficient control methodologies.

2 Gaussian Processes

This chapter presents Gaussian processes (GPs), which play a key role in this thesis. They can be used for classification and regression problems, Regression is the one that is more important in the context of control. For this reason, Gaussian process regression in the realm of control systems is presented in detail, where Section 2.1 describes the method based on linear functions and Section 2.2 extended to non-linear functions. In Section 2.3, the focus shifts to long-term prediction using Moment Matching Approximation. Here, the capability of prediction to provide predictions over extended time horizons is explored, which is crucial for effective control strategies that require anticipation of future system behavior. Finally, Section 2.4 delves into GP hyperparameter optimization, elucidating techniques for fine-tuning the parameters of Gaussian processes to enhance their predictive performance. This section highlights the importance of parameter optimization in maximizing the utility of GPs within control systems.

2.1 Linear Regression Model

In the realm of machine learning algorithms, Gaussian processes are often used for supervised learning, i.e., labeled input-output data is used to determine a mapping between input and output (see [10] for an exception). Probabilistic regression is usually formulated as follows: given a training set $D = \{(\mathbf{x_i}, y_i), i = 1, \ldots, n\}$ of n pairs of (vectorial) inputs $\mathbf{x_i}$ and noisy (real, scalar) outputs y_i, compute the predictive distribution of the function values f^* (or noisy y^*) at test locations $\mathbf{x}^*$. In the simplest case (which we deal with here), we assume that the noise is additive, independent, and Gaussian, such that the relationship between the (latent) function $f(x)$ and the observed noisy targets y is given by:

$$f(x_i) = \mathbf{x_i}^T\mathbf{w}, \qquad y_i \;= f(\mathbf{x_i}) + \epsilon_i \tag{2.1}$$

where $\mathbf{w} \in \mathbb{R}^D$ is the weight column vector of the linear model, $\mathbf{x}$ is the input vector, $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$, and σ^2_{noise} is the variance of the noise and y is the observed value. The noise ϵ is assumed to be zero-mean with variance σ_n^2, thus $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$

Definition 1: A Gaussian process (GP) is a collection of random variables, any finite number of which have consistent joint Gaussian distributions. It is a Bayesian approach that assumes a GP prior over functions [11], i.e., assumes a prior that function values behave according to

$$p(f|x_1, x_2, \ldots, x_n) = \mathcal{N}(0, K(\mathbf{x_1}, \mathbf{x_2})), \tag{2.2}$$

where $f = [f_1, f_2, \ldots, f_n]^\top$ is a vector of latent function values, $f_i = f(\mathbf{x_i})$, and $K(\mathbf{x_1}, \mathbf{x_2})$ is a covariance matrix, whose entries are given by the covariance function, $K(\mathbf{x_1}, \mathbf{x_2})_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$. The covariance function encodes our assumptions about the function we wish to learn by defining a notion of similarity between two function values, as a function of the corresponding two inputs. A very common covariance function is the Gaussian, or squared exponential[12],

$$K_{ij} = k(\mathbf{x_i}, \mathbf{x_j}) = \sigma_f^2 \exp\left(-\frac{(\mathbf{x_i} - \mathbf{x_j})^2}{2l^2}\right), \tag{2.3}$$

where σ_f^2 controls the prior variance, and l is an isotropic lengthscale parameter that controls the rate of decay of the covariance, i.e., determines how far away $\mathbf{x_i}$ must be from $\mathbf{x_j}$ for f_i to be unrelated to f_j. We term the parameters that define the covariance functions hyperparameters. Optimization of the hyperparameters based on data is discussed in Section 2.4.

2.1.1 Posterior Distribution

To predict function values for all inputs considering observations, the equation

$$f(X^*) = X_*^T \mathbf{w} \tag{2.4}$$

is used, where X^* denotes a matrix consisting of test points $\mathbf{x}^*$ such that $X^* \in \mathbb{R}^D$, $X^* = [\mathbf{x_1^*}, \ldots, \mathbf{x_s^*}]$. The posterior distribution of weights, denoted as w, is used to predict function values for all inputs considering observations. This distribution is obtained through Bayes' rule, where the posterior $(p(a|b))$ is proportional to the likelihood $(p(b|a))$ multiplied by the prior $(p(a))$ and normalized by the marginal likelihood $(p(b))$. In the context of the posterior distribution of weights, Bayes' rule can be rewritten as:

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w}) \cdot p(\mathbf{w})}{p(\mathbf{y}|X)} \tag{2.5}$$

The prior represents the model weights' beliefs without observations, often assumed as a Gaussian distribution with zero mean and covariance matrix as shown in Equation (2.2). The likelihood $p(y|X, w)$ is the probability density function of the observations given the inputs and model weights, which can be determined using Equation (2.1).

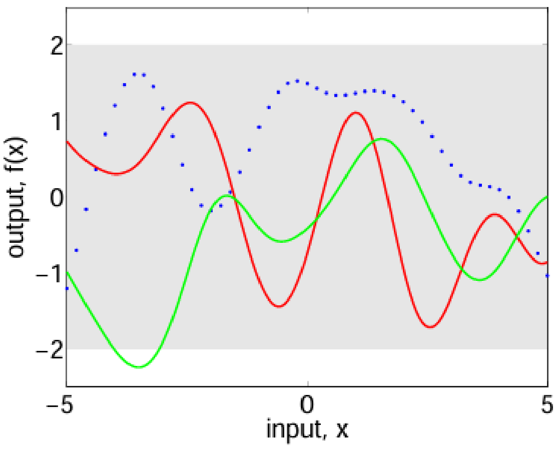
$$p(\mathbf{y}|X, \mathbf{w}) = \mathcal{N}(X^T\mathbf{w}, \sigma_n^2 I) \tag{2.6}$$

By combining the prior Equation (2.2) and likelihood Equation (2.6), the posterior distribution of weights given observations can be determined by their multiplication as shown in Equation 2.7

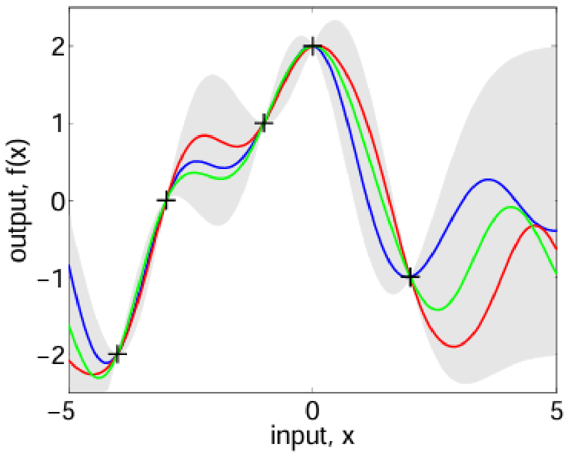
$$p(\mathbf{w}|\mathbf{y}, X) \propto p(\mathbf{y}|X, \mathbf{w}) \cdot p(\mathbf{w}) \tag{2.7}$$

The marginal likelihood is independent of w and just a normalization constant which does not influence the shape of the distribution. By substituting $A = \sigma_n^{-2} XX^T + \Sigma_p^{-1}$ and adding the terms $\pm\sigma_n^{-2} y^T X^T A^{-1} A^{-1} X y \sigma^{-2}$ to the exponent, the equation can be rewritten as:

$$p(\mathbf{w}|\mathbf{y}, X) \propto \exp\left(-\frac{1}{2}\mathbf{w}^T\left(\sigma_n^{-2} A^{-1} X\mathbf{y}\right) - \frac{1}{2}\sigma_n^{-2} y^T X^T A^{-1} A^{-1} X\mathbf{y}\right), \tag{2.8}$$



(a) prior



(b) posterior

Figure 2.1: Panel (a) displays three functions drawn at random from a GP prior. The dots represent values of y actually generated, while the other functions have been drawn as lines by connecting a large number of evaluated points.Panel (b) illustrates three random functions drawn from the posterior, i.e., the prior conditioned on the five noise-free observations indicated. In both plots, the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior, respectively [12].

where $\alpha = \sigma_n^{-2}(A^{-1}y^Ty - y^TXA^{-1}A^{-1}X^Ty)$ is independent of w and hence also a scaling factor. Finally, the posterior is distributed according to:

$$p(\mathbf{w}|\mathbf{y}, X) = \mathcal{N}\left(\frac{1}{\sigma_n^2}A^{-1}X\mathbf{y}, A^{-1}\right) \tag{2.9}$$

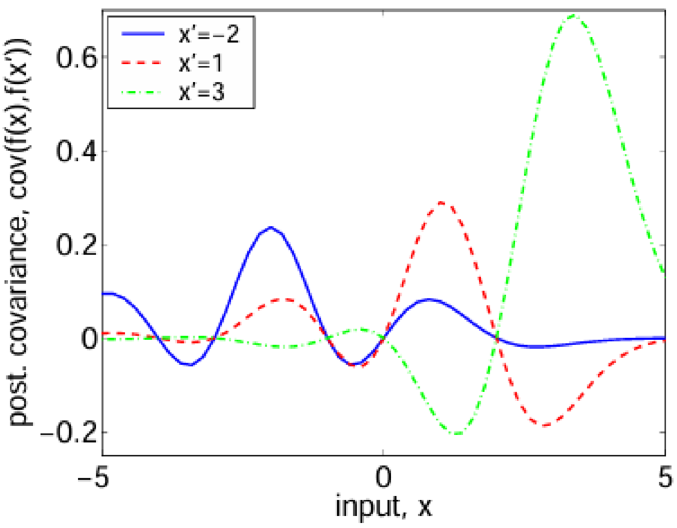


Figure 2.2: The posterior covariance illustrates the covariance between $f(x)$ and $f(x')$ for the same data, with three different values of x'. Notably, the covariance is high at points close to each other, diminishes to zero at the training points (where variance is absent due to the noise-free process), and subsequently becomes negative. This variation occurs because if the smooth function happens to be below the mean on one side of a data point, it is inclined to surpass the mean on the opposite side, resulting in a change in the sign of the covariance at the data points. In contrast, the prior covariance exhibits a Gaussian shape and remains non-negative. [12].

2.1.2 Prediction

When making predictions using Gaussian processes (GPs), incorporating observed data is crucial. For noise-free observations $\{(\mathbf{x_i}, f_i)|i = 1, ..., n\}$, the joint distribution of training outputs f and test outputs f^* under the prior is given by:

$$\left[f, f^*\right] \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right) \tag{2.10}$$

To obtain the posterior distribution over functions, we condition this joint prior distribution on the observed data points. The resulting posterior distribution is expressed as:

$$f^*|X^*, X, f \sim \mathcal{N}\left(K(X^*, X)K(X, X)^{-1}f, K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*)\right) \tag{2.11}$$

This enables sampling function values f^* from the joint posterior distribution by evaluating the mean and covariance matrix.

Prediction with noisy observations incorporates additive independent identically distributed Gaussian noise. The joint distribution of observed target values and function values at test locations under the prior is given by:

$$\left[\mathbf{y}, f^*\right] \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right) \tag{2.12}$$

The predictive equations for GP regression are:

$$\bar{f}^* = K(X^*,X)[K(X,X) + \sigma_n^2 I]^{-1} y \tag{2.13}$$
$$\mathrm{cov}(f^*) = K(X^*,X^*) - K(X^*,X)[K(X,X) + \sigma_n^2 I]^{-1} K(X,X^*) \tag{2.14}$$

These equations provide mean predictions and uncertainties for test targets.

2.2 Extension to Non-Linear Functions

Gaussian processes can be used for both linear regression, restricting them to linear regression means that it won't capture nonlinear relationships between variables effectively. If the relationship between the input and output variables is nonlinear, using a linear model might result in poor predictions and biased estimates. Since the order of the target function is generally unknown, direct application of raw inputs is not effective. To overcome this problem, the idea is to transform the inputs into a higher dimensional space via basis functions $\psi(x)$, $\psi : \mathbb{R}^D \to \mathbb{R}^N$, where $N \gg D$, and then apply the linear regression model. Thus the linear regression model Equation 2.1 is reformulated such that

$$f(x) = \psi(x) w, \tag{2.15}$$

Accordingly, the predictive distribution is given by:

$$p(f^*|X^*, X, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2}\Psi(X^*)^T A^{-1}\Psi(X)\mathbf{y}, \frac{1}{\sigma_n^2}\Psi(X^*)^T A^{-1}\Psi(X^*)\right) \tag{2.16}$$

With $A = \sigma^{-2} n \Psi(X)\Psi(X)^T + \Sigma_p$, can be rearranged as shown in [12], such that

$$p(f^*|X^*, X, y) = \mathcal{N}(\Psi_*^T \Sigma_p \Psi_K + \sigma_n^2 I)^{-1} y, \Psi_*^T \Sigma_p \Psi_* - \Psi_*^T \Sigma_p \Psi_K + \sigma_n^2 I)^{-1} \Psi_*^T \Sigma_p \Psi_*, \tag{2.17}$$

where $K = \Psi_*^T \Sigma_p \Psi$. The Equation 2.17 explicitly depends on basics functions ψ i.e the feature space appears only in the form of $\Psi^* \Sigma_p \Psi^*$, $\Psi^* \Sigma_p \Psi$, and $\Psi \Sigma_p \Psi$. the basis functions ψ explicitly is not feasible. Thus, defining $\varphi(x) = \Sigma_{1/2}^p \psi(x)$ (notice that Σ_p is positive definite), the expressions can be substituted by $\varphi(x)\varphi(x)^T$. So $\langle \varphi(x), \varphi(x)^T \rangle$ can be replaced by kernel function $k(x, x^T)$ using Mercer's theorem [13]. The kernel matrices $K(\cdot, \cdot)$ which are given by:

$$K(X,Y) = \begin{pmatrix} k(X_1,Y_1) & \cdots & k(X_1,Y_M) \\ \vdots & \ddots & \vdots \\ k(X_N,Y_1) & \cdots & k(X_N,Y_M) \end{pmatrix} \tag{2.18}$$

Substituting kernal function (2.18) into (2.17)

$$\begin{aligned}p(f^*|X^*, X, \mathbf{y}) = \mathcal{N}(&K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1} y,\\ &K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X^*)).\end{aligned}$$

This new formalism depends only implicitly on the basis functions via the kernel matrices and is preferred because determining the basis functions explicitly is quite hard Equation 2.19

Assumption 2.1: The objective function f is a member of a reproducing kernel Hilbert space (RKHS), with known kernel function $k(\cdot,\cdot) : X \times X \to \mathbb{R}$ and bounded norm $\|f\|_H^2 \leq B$.

Assumption 2.1 is necessary to ensure the applicability of Gaussian processes [12], which can be seen in the modified problem description (2.15) where $f(x)$ is defined as the linear combination of the basis functions $\psi(x)$. The bound on $\|f\|_H$ imposes a complexity and magnitude condition on the RKHS; if $\|f\|_H$ gets smaller, f gets smoother (less complex) [13]. One might imagine that a function assumed to have infinite complexity would not be feasible because the uncertainty for the predictive distribution would be infinite for all $X \setminus O$.

2.3 Long-term Prediction

When the input X^* (we employ lower-case x^* for a single test input and upper-case $X*$ for several test inputs) is a deterministic input, the mean and covariance of the prediction are calculated according to Equation (2.10). However, if the input itself is Gaussian, $X \sim \mathcal{N}(\bar{x^*}, \Sigma_x^*)$, the predictive distribution becomes:

$$p(f^*|\mu_x^*, \Sigma_x^*, \mathbf{D}) = \int p(f^*|X^*, \mathbf{D}) \cdot p(X^*|\Sigma_x^*, \mathbf{D})\, dx^* \tag{2.19}$$

The integral in Equation (2.19) is analytically intractable [14]. To address this, various methods have been developed to approximate the posterior distribution [15]. One approach is to approximate the integral numerically using Monte-Carlo methods [16]. Alternatively, the posterior distribution $p(f|x, \mathbf{X}, \mathbf{D})$ can be approximated as a Gaussian by calculating its mean and variance. Several methods exist for this Gaussian approximation, such as Moment Matching and linearization of the posterior GP mean function[17]. Moment Matching computes the first two moments of the predictive distribution exactly, whereas linearization provides a computationally efficient approximation by explicitly linearizing the posterior GP. In this thesis, we will focus on the Moment Matching Gaussian approximation.

As shown in [18], it is possible to compute the mean and variance analytically in the case of a Gaussian kernel function. The main results are presented here. Following the law of iterated expectations, for target dimensions $a = 1, ..., D$ we obtain the predictive mean:

$$\begin{aligned}
\mu_t^a &= \mathbb{E}_{\tilde{\boldsymbol{x}}_{t-1}}\left[\mathbb{E}_{f_a}\left[f_a\left(\tilde{\boldsymbol{x}}_{t-1}\right) \mid \tilde{\boldsymbol{x}}_{t-1}\right]\right] = \mathbb{E}_{\tilde{\boldsymbol{x}}_{t-1}}\left[m_{f_a}\left(\tilde{\boldsymbol{x}}_{t-1}\right)\right] \\
&= \int m_{f_a}\left(\tilde{\boldsymbol{x}}_{t-1}\right) \mathcal{N}\left(\tilde{\boldsymbol{x}}_{t-1} \mid \tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}\right) d\tilde{\boldsymbol{x}}_{t-1} \\
&= \boldsymbol{\beta}_a^T \boldsymbol{q}_a
\end{aligned} \tag{2.20}$$

where,

$$\boldsymbol{\beta}_a = \left(\boldsymbol{K}_a + \sigma_{w_a}^2\right)^{-1} \boldsymbol{y}_a \tag{2.21}$$

with $\boldsymbol{q}_a = \left[q_{a_1}, \ldots, q_{a_n}\right]^T$. The entries of $\boldsymbol{q}_a \in \mathbb{R}^n$ are computed using standard results from multiplying and integrating over Gaussians and are given by

$$\begin{aligned}
q_{a_i} &= \int k_a\left(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_{t-1}\right) \mathcal{N}\left(\tilde{\boldsymbol{x}}_{t-1} \mid \tilde{\boldsymbol{\mu}}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}\right) d\tilde{\boldsymbol{x}}_{t-1} \\
&= \sigma_{f_a}^2 \left|\tilde{\boldsymbol{\Sigma}}_{t-1} \boldsymbol{\Lambda}_a^{-1} + \boldsymbol{I}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \boldsymbol{\nu}_i^T \left(\tilde{\boldsymbol{\Sigma}}_{t-1} + \boldsymbol{\Lambda}_a\right)^{-1} \boldsymbol{\nu}_i\right),
\end{aligned} \tag{2.22}$$

where we define

$$\nu_i := \left(\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{\mu}}_{t-1}\right) \tag{2.23}$$

is the difference between the training input $\tilde{\boldsymbol{x}}_i$ and the mean of the test input distribution $p\left(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}\right)$.

Computing the predictive covariance matrix $\boldsymbol{\Sigma}_\Delta \in \mathbb{R}^{D \times D}$ requires us to distinguish between diagonal elements σ_{aa}^2 and off-diagonal elements $\sigma_{ab}^2, a \neq b$: Using the law of total (co-)variance, we obtain for target dimensions $a, b = 1, \ldots, D$

$$\sigma_{aa}^2 = \mathbb{E}_{\tilde{\boldsymbol{x}}_t}\left[\operatorname{var}_f\left[\Delta_a \mid \tilde{\boldsymbol{x}}_t\right]\right] + \mathbb{E}_{f, \tilde{\boldsymbol{x}}_t}\left[\Delta_a^2\right] - \left(\boldsymbol{\mu}_\Delta^a\right)^2, \tag{2.24}$$

$$\sigma_{ab}^2 = \mathbb{E}_{f, \tilde{\boldsymbol{x}}_t}\left[\Delta_a \Delta_b\right] - \boldsymbol{\mu}_\Delta^a \boldsymbol{\mu}_\Delta^b, \quad a \neq b, \tag{2.25}$$

respectively, where μ_Δ^a is known from (2.20). The off-diagonal terms σ_{ab}^2 do not contain the additional term $\mathbb{E}_{\tilde{\boldsymbol{x}}_t}\left[\operatorname{cov}_f\left[\Delta_a, \Delta_b \mid \tilde{\boldsymbol{x}}_t\right]\right]$ because of the conditional independence assumption of the GP models: Different target dimensions do not covary for given $\tilde{\boldsymbol{x}}_t$.

We start the computation of the covariance matrix with the terms that are common to both the diagonal and the off-diagonal entries: With $p\left(\tilde{\boldsymbol{x}}_t\right) = \mathcal{N}\left(\tilde{\boldsymbol{x}}_t \mid \tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t\right)$ and the law of iterated expectations, we obtain

$$\begin{aligned}
\mathbb{E}_{f, \tilde{\boldsymbol{x}}_t}\left[\Delta_a \Delta_b\right] &= \mathbb{E}_{\tilde{\boldsymbol{x}}_t}\left[\mathbb{E}_f\left[\Delta_a \mid \tilde{\boldsymbol{x}}_t\right] \mathbb{E}_f\left[\Delta_b \mid \tilde{\boldsymbol{x}}_t\right]\right] \\
&\overset{(6)}{=} \int m_f^a\left(\tilde{\boldsymbol{x}}_t\right) m_f^b\left(\tilde{\boldsymbol{x}}_t\right) p\left(\tilde{\boldsymbol{x}}_t\right) \mathrm{d}\tilde{\boldsymbol{x}}_t
\end{aligned} \tag{2.26}$$

because of the conditional independence of Δ_a and Δ_b given $\tilde{\boldsymbol{x}}_t$. Using the definition of the GP mean function, we obtain

$$\mathbb{E}_{f, \tilde{\boldsymbol{x}}_t}\left[\Delta_a \Delta_b\right] = \boldsymbol{\beta}_a^\top \boldsymbol{Q} \boldsymbol{\beta}_b, \tag{2.27}$$

$$\boldsymbol{Q} := \int k_a\left(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{X}}\right)^\top k_b\left(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{X}}\right) p\left(\tilde{\boldsymbol{x}}_t\right) \mathrm{d}\tilde{\boldsymbol{x}}_t. \tag{2.28}$$

Using standard results from Gaussian multiplications and integration, we obtain the entries Q_{ij} of $Q \in \mathbb{R}^{n \times n}$

$$Q_{ij} = |\boldsymbol{R}|^{-\frac{1}{2}} k_a\left(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{\mu}}_t\right) k_b\left(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{\mu}}_t\right) \exp\left(\frac{1}{2} \boldsymbol{z}_{ij}^{\top} \boldsymbol{T}^{-1} \boldsymbol{z}_{ij}\right) \tag{2.29}$$

where we define $\quad \boldsymbol{R} := \tilde{\boldsymbol{\Sigma}}_t\left(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}\right) + \boldsymbol{I}, \qquad \boldsymbol{T} := \boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1} + \tilde{\boldsymbol{\Sigma}}_t^{-1},$
$\boldsymbol{z}_{ij} := \boldsymbol{\Lambda}_a^{-1} \boldsymbol{\nu}_i + \boldsymbol{\Lambda}_b^{-1} \boldsymbol{\nu}_j,$

with ν_i defined in (2.23). Hence, the off-diagonal entries of Σ_Δ are fully determined by (2.20)-(2.23), (2.25), and (2.27)-(2.29).

From (2.24), we see that the diagonal entries contain the additional term

$$\mathbf{E}_{\tilde{\boldsymbol{x}}_t}\left[\operatorname{var}_f\left[\Delta_a \mid \tilde{\boldsymbol{x}}_t\right]\right] = \sigma_{f_a}^2 - \operatorname{tr}\left(\left(\boldsymbol{K}_a + \sigma_{w_a}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{Q}\right) + \sigma_{w_a}^2 \tag{2.30}$$

with Q given in (2.29) and $\sigma_{w_a}^2$ being the system noise variance of the a th target dimension. This term is the expected variance of the function, under the distribution $p\left(\tilde{\boldsymbol{x}}_t\right)$.

The final covariance matrix is given by

$$\sum(:) = \begin{bmatrix} \sigma_{aa}^2 & \sigma_{ab}^2 & \ldots\ldots \\ \sigma_{ab}^2 & \sigma_{bb}^2 & \ldots\ldots \\ . & . & \ldots\ldots \\ . & . & \ldots\ldots \end{bmatrix}_{DxD} \tag{2.31}$$

2.4 Optimization of the Hyperparameters

To ensure accurate predictions, selecting appropriate hyperparameters is crucial. However, in the case of a black-box system, precise information about these hyperparameters is often unavailable, necessitating the use of computationally efficient methods for optimization. In [12], two such methods are introduced: maximizing the marginal likelihood (Bayesian model selection) and cross-validation. Hyperparameters are typically determined using available training data $\mathcal{D}$, rather than relying solely on prior knowledge or physical insight, which can be challenging. Given a prior probabilistic belief of the hyperparameters' distribution $p(\theta)$ (often assumed to be uniform), the goal is to infer the hyperparameters' posterior distribution $p(\theta|W, z)$ using Bayes' theorem:

$$\begin{aligned} p(w|y, X, \theta) &= \frac{p(y|X, w, \theta) \cdot p(w|\theta)}{p(y|X, \theta)}, \\ p(y|X, \theta) &= \int p(y|X, w, \theta) \cdot p(w|\theta)\, dw. \end{aligned} \tag{2.32}$$

The optimal solution would be a predictive distribution without dependency on the hyperparameters. By marginalization of θ, a hyperparameter-independent formalism can be obtained:

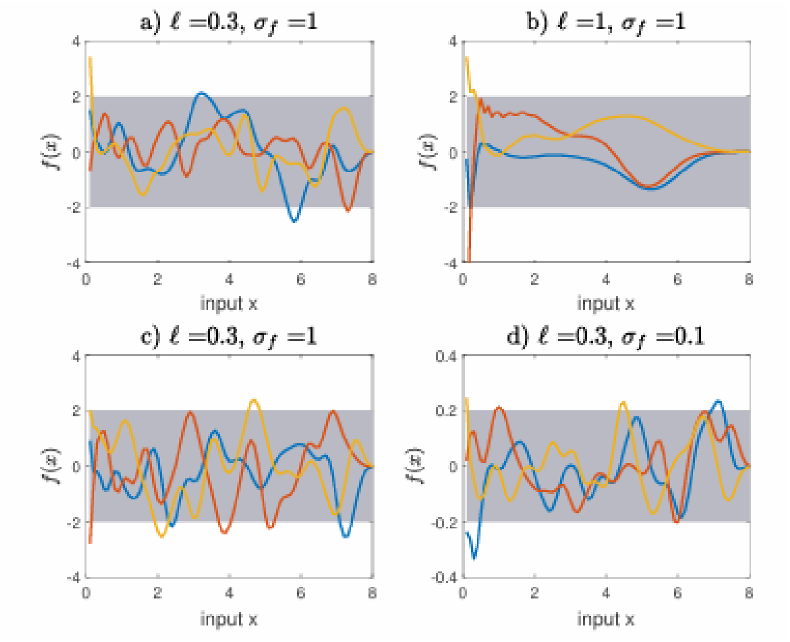


Figure 2.3: Three samples are drawn from the prior with different sets of hyperparameters. Figures (a) and (b) demonstrate the impact of changing the lengthscale, while Figures (c) and (d) illustrate the effect of varying the signal standard deviation f. The uncertainty is depicted by the 95% confidence interval, shaded in gray [19].

$$p(f^*|y, X, X^*) = \int p(f^*|y, \theta, X, X^*) \cdot p(y|\theta, X, X^*) \cdot p(\theta|X, X^*)\, d\theta. \tag{2.33}$$

Unfortunately, this integral is not analytically tractable, as $p(f^*|y, \theta, X, X^*)$ and a $p(y|\theta, X, X^*)$ have non-trivial dependencies on θ. Several methods exist to estimate the integral, such as Hamiltonian Monte Carlo (HMC), Bayesian Monte Carlo (BMC), and Sequential Monte Carlo (SMC).

An alternative approach to approximate (2.33) is by maximizing the second-level marginal likelihood (ML-II) given by (2.32). As shown in previous work, finding the hyperparameters $\theta_{\max}$ that maximize $p(y|X, \theta)$ leads to:

$$p(f^*|y, X, X^*) \approx p(f^*|\theta_{\max}, y, X, X^*).$$

This method offers the advantage of analytically tractable integration. Gradient-based algorithms are typically used for maximization due to the computational efficiency of determining the partial derivatives of the likelihood with respect to θ. However, the non-convex nature of the marginal likelihood surface can lead to overfitting or underfitting. Additionally, gradient-based optimization is highly sensitive to initial values.

The contour of the marginal likelihood as a function of the length scale and noise is illustrated in Figure 2.4, showing both local and global minima. Global optimization using the hyperparameters at the global optimum results in preferable predictions compared to local minima, where observations may be misinterpreted as noise.

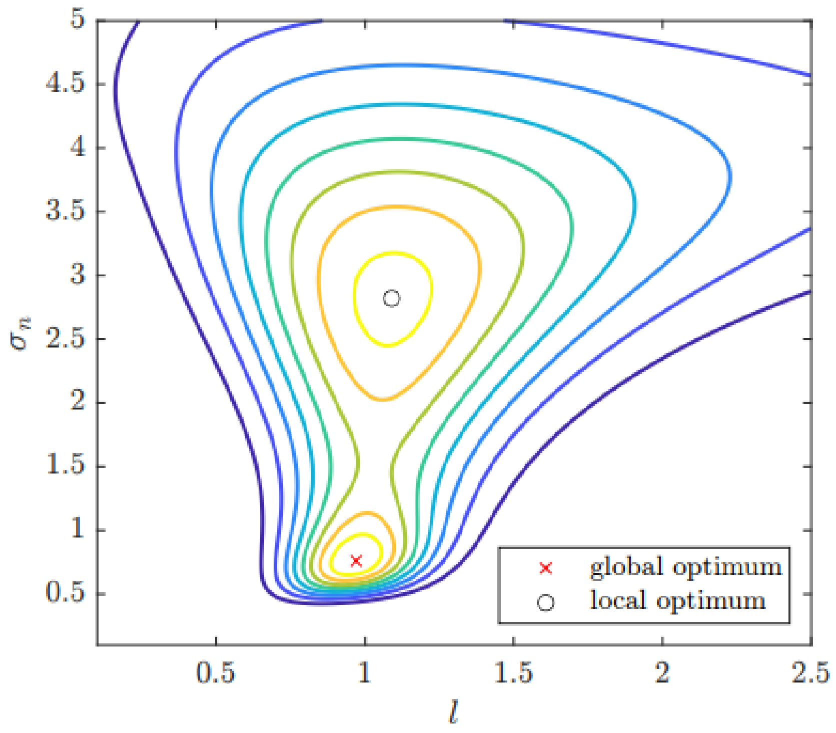


Figure 2.4: Contour of the marginal likelihood depending on the length scale l and noise σ_n [20].

Reasonable hyperparameter selection is crucial, and alternative methods such as global optimizers like Dividing Rectangles (DIRECT) can be utilized to mitigate sensitivity to initial values. If the noise level is known, fixing the respective hyperparameter during optimization can prevent misinterpretation of observations.

3 Model Predictive Control and Learning

We present a Model predictive control (MPC) scheme that uses Gaussian process transition models and is capable of online learning. To this end, we employ the GP model and GP Transition model for long-term prediction. we formulate the considered control problem and present the online learning scheme of the Gaussian process model predictive control and we denote it as GP-MPC.

The chapter starts with an introduction (Section 3.1) to the topic of Model Predictive Control and its challenges, along with the mathematical basics of MPC, together with a detailed discussion of feasibility and stability required later for the main theoretical results of this chapter. In 3.2, we formulate the considered control problem and present the online learning scheme of GP-MPC and also the resulting optimal control problem.

3.1 Model Predicitve control

Model Predictive Control (MPC, [21]) is a control methodology adept at managing multi-input multi-output systems, accommodating constraints from the outset of the design process. Essentially, MPC operates as iterative optimal control, leveraging a model of a real process $x_{k+1} = f(x_k, u_k)$, with state x_k and input u_k at time k, for prediction and to address a finite horizon optimal control problem (OCP). The initial computed optimal input sequence is applied to the plant, and the OCP is resolved at the subsequent time step. This approach considers potential disturbances and uncertainties occurring between consecutive time steps. The recurrent solving of the OCP at each time instant advances the prediction horizon by one step, hence the term receding horizon control[8].

In terms of performance, Model Predictive Control often outperforms other control methods due to its ability to predict future process behavior, enabling the computation of control actions based on anticipated outcomes. Additionally, MPC allows for the incorporation of preview information about references and disturbances, if available. Unlike many other control approaches, MPC facilitates the direct consideration of constraints on input, state, and output during the design phase. This feature has garnered significant scientific interest and found practical applications. Over the past decades, a robust theoretical framework has been developed to provide guarantees concerning aspects like recursive constraint satisfaction and stability.

However, despite the advantages it offers, MPC also presents challenges. For example, the standard MPC formulation necessitates full state information, meaning the entire state x_k must be measurable. If this requirement cannot be met, various state estimation methods, such as Luenberger observers, Kalman filters, or more sophisticated techniques like moving horizon estimation [22], can be employed. Alternatively, output feedback MPC schemes that solely utilize measured system output offer an alternative approach.

Another challenge is the potentially high computational costs associated with MPC. Solving the (possibly nonlinear and nonconvex) optimal control problem within the sampling time frame is crucial. Initially, this constraint limited MPC deployment to systems with large time constants, such as process industry plants. However, advancements in algorithms, such as acados [23], and the increasing computational power of digital processors now permit the use of MPC in more demanding applications, including embedded systems for mechatronics [24]. Moreover, MPC's scope extends beyond traditional engineering tasks, with applications in diverse fields such as HIV treatment strategies in medicine [25] and others [26]. The basic procedure applied by a model predictive control scheme can be illustrated in Figure 3.1 and summarized as follows:

1. Obtain the state x_k at the current time step k.
2. Formulate and solve a finite horizon optimal control problem. This involves predicting the future evolution of the system and determining an optimal input sequence that minimizes a given cost function over a finite prediction horizon.
3. Apply the first part of the optimal input sequence to the system.
4. Repeat, go back to Step 1.

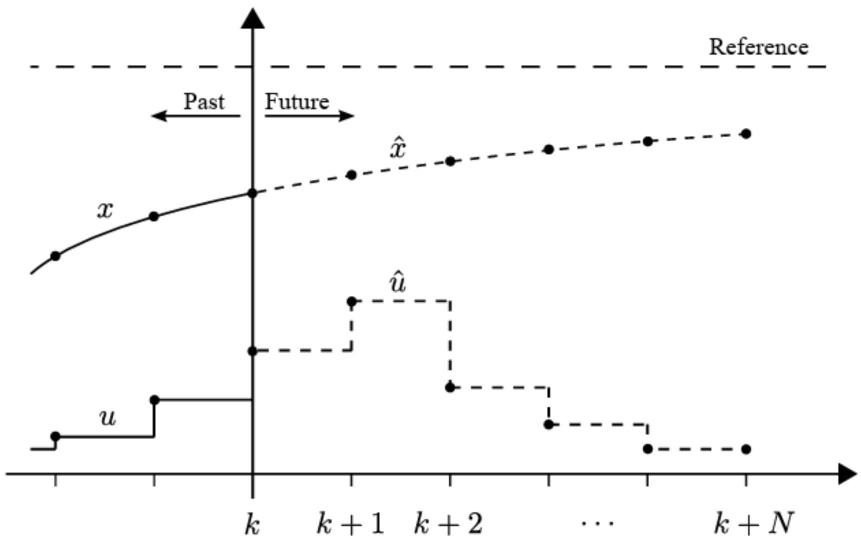


Figure 3.1: MPC Illustration: For a particular initial condition x_k at time instant k, a predicted open-loop sequence of inputs $\hat{u}$ is computed up to a prediction horizon N. The input sequence, together with the resulting predicted states $\hat{x}$, are computed in such a way that they minimize a given cost[27].

3.1.1 MPC formulation

We consider discrete-time-invariant and constrained nonlinear state-space systems of the form:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ u_k &\in \mathcal{U} \\ x_k &\in \mathcal{X} \end{aligned} \tag{3.1}$$

where $k \in \mathbb{N}_0$ denotes the discrete time, $x_k \in \mathbb{R}^{n_x}$ represents the state, and $u_k \in \mathbb{R}^{n_u}$ denotes the input.

The state and input constrained sets are subsets of the respective spaces, i.e., $X \subset \mathbb{R}^{n_x}$ and $U \subset \mathbb{R}^{n_u}$. Input constraints usually arise due to actuator saturation, for example, a valve cannot be opened more than 100% and not less than 0%. State constraints often arise as a consequence of process operation conditions, for example, if for safety reasons a temperature should not exceed a predefined limit.

A particular initial condition x_k at time instant k and a sequence of inputs $u_k = \{u_k, u_{k+1}, \ldots\}$ applied to Equation (3.1) result in a state sequence $x_k = \{x_k, x_{k+1}, \ldots\}$. The state sequence as a whole depends on the initial condition and the applied input sequence, i.e., $x_k = x_k(x_k, u_k)$. However, for the sake of simpler notation, we do not emphasize this dependence.

For the standard MPC formulation, together with its theoretical results as presented in the following, the common objective is regulation to and stabilization of the origin. Thus, it is assumed that the system possesses an equilibrium point at the origin, i.e., $f(0, 0) = 0$.

3.1.2 Prediction Model

In order to compute the future behavior of the process Equation (3.1), a prediction model

$$\hat{x}_{k+i+1|k} = \hat{f}(\hat{x}_{k+i|k}, \hat{u}_{k+i|k}) \tag{3.2}$$

is used, which is evaluated over a finite prediction horizon $N \in \mathbb{N}$ where $i \in \{0, 1, ..., N-1\}$. The variable k denotes the global discrete time, whereas i denotes the internal controller time within the prediction horizon N. The hat notation $\hat{(\cdot)}$ denotes a predicted variable and the subset notation $(\cdot|k)$ emphasizes that the prediction is based on the information available at the specific time step k. The current measurement $x_k = \hat{x}_{k|k}$ serves as the initial condition for the prediction. The Long-term prediction model is explained in detail in Section 2.3. This GP transition model serves as a prediction model in our case.

Given a predicted input sequence $\hat{u}_{k|k} = \{\hat{u}_{k|k}, \hat{u}_{k+1|k}, ..., \hat{u}_{k+N-1|k}\}$ and initial condition x_k, the predicted state evolution is $\hat{x}_{k|k} = \{\hat{x}_{k|k}, \hat{x}_{k+1|k}, ..., \hat{x}_{k+N|k}\}$ with $\hat{x}_{k|k} = x_k$. Note that $\hat{x}_{k|k}$ has $N + 1$ elements, whereas $\hat{u}_{k|k}$ has N elements because no further input is needed in the last stage. The difference between the signal sequences u_k and x_k of the real process Equation (3.1) and the sequences $\hat{u}_{k|k}$ and $\hat{x}_{k|k}$ of the prediction model Equation (3.2) is that the predicted sequences are computed for each time instant k and used as internal variables of the controller, whereas u_k and x_k describe the applied control inputs and actual system evolution over time.

3.1.3 Cost Function

The model predictive controller operates by computing an open-loop input sequence $\hat{u}_{k|k}$ through the minimization of a cost function over a finite horizon N starting from the initial condition x_k.

$$V_N(x_k, \hat{u}_{k|k}) = \sum_{i=0}^{N-1} \ell(\hat{x}_{k+i|k}, \hat{u}_{k+i|k}) + V_f(\hat{x}_{k+N|k}) \tag{3.3}$$

This cost function, denoted as $V_N(x_k, \hat{u}_{k|k})$, is comprised of a stage cost $\ell(x, u)$ and a terminal cost $V_f(x)$, as shown in Equation (3.3).

The stage cost penalizes deviations of the predicted state and input trajectories from the origin/set point at each prediction step, while the terminal cost emphasizes penalizing the deviation of the last predicted state. To ensure the stability of the system, Assumption 3.1 needs to be met.

Assumption 3.1 *referred to as the Positive Definite Cost assumption, states that both the stage and terminal costs are positive definite and evaluated to zero at the origin:*

- $\ell(0, 0) = 0$ *and* $\ell(x, u) > 0$ *for all* $(x, u) \neq (0, 0)$.
- $V_f(0) = 0$ *and* $V_f(x) > 0$ *for all* $x \neq 0$.

It's important to note that the cost function Equation (3.3) does not explicitly rely on the state sequence $\hat{x}_{k|k}$, as it is determined by the given initial condition x_k and the input sequence $\hat{u}_{k|k}$.

A typical control objective involves driving the state to the origin rapidly while ensuring that control energy remains within specified limits. This presents a trade-off that must be considered when selecting a stage cost. Hence, determining the most suitable cost function according to the specific control objective is a significant task on its own.

The common stage cost function used is the quadratic cost function, where the function contains quadratic states and inputs.

$$\ell(\hat{x}_{k+i|k}, \hat{u}_{k+i|k}) = \hat{x}_{k+i|k}^T Q \hat{x}_{k+i|k} + \hat{u}_{k+i|k}^T R \hat{u}_{k+i|k} \tag{3.4}$$

where Q and R are matrices chosen to be positive definite, with all off-diagonal elements set to zero. Intuitively, the diagonal elements of Q represent the weighting assigned to individual error signals, while the elements of R represent the weighting assigned to each control action. In practice, as the controller designer, we manually select Q and R based on the desired performance of the system.

3.1.4 Optimal Control Problem

At each time step k, the model predictive controller minimizes the cost function Equation (3.3) concerning the open-loop input sequence $\hat{u}_{k|k}$, while adhering to state and input constraints and maintaining the modeled system dynamics Equation (3.2) and the initial condition x_k. Moreover, it ensures that the last predicted state $\hat{x}_{k+N|k}$ lies within a designated terminal region $X_f \subseteq X$, crucial for establishing recursive feasibility (refer to Section feasibility section).

The resulting optimal control problem used in model predictive control is

$$
\begin{aligned}
P_N(x_k) : \min_{\hat{u}_{k|k}} \quad & V_N(x_k, \hat{u}_{k|k}) \\
\text{subject to} \quad & \forall i \in I_{0:N-1} : \\
& \hat{x}_{k+i+1|k} = \hat{f}(\hat{x}_{k+i|k}, \hat{u}_{k+i|k}) \\
& \hat{x}_{k|k} = x_k \\
& \hat{u}_{k+i|k} \in \mathcal{U} \\
& \hat{x}_{k+i|k} \in \mathcal{X} \\
& \hat{x}_{k+N|k} \in \mathcal{X}_{\{}
\end{aligned}
\tag{3.5}
$$

The solution to $P_N(x_k)$ is denoted as $\hat{u}^*_{k|k} = \{\hat{u}^*_{k|k}, \hat{u}^*_{k+1|k}, ..., \hat{u}^*_{k+N-1|k}\}$, where the superscript $*$ signifies the optimal solution. Correspondingly, the resulting optimal state sequence is represented as $\hat{x}^*_{k|k} = \{\hat{x}^*_{k|k}, \hat{x}^*_{k+1|k}, ..., \hat{x}^*_{k+N|k}\}$. The optimal cost is $V^*_N(x_k) = V_N(x_k, \hat{u}^*_k|k)$, typically referred to as the value function. The initial element of the optimal input sequence $\hat{u}^*_{k|k}$, is implemented in the process, thereby defining the MPC control law $u_k = \kappa_{MPC}(x_k) = \hat{u}^*_k|k$. Subsequently, the successor state is updated as $x_{k+1} = f(x_k, \kappa_{MPC}(x_k))$, and the process repeats at the next time step by solving $P_N(x_k)$ again.

As widely recognized, the recursive solution to an optimal control problem doesn't inherently ensure stability. There's no guarantee of feasibility, meaning a solution to the optimal control problem may not exist at the subsequent time instant. However, a robust theory has been developed to ensure feasibility and stability, often employing concepts from Lyapunov stability and set theory, particularly for guaranteeing recursive feasibility. Subsequent subsections delve into these fundamental concepts and methodologies. Only the concept of feasibility is explained in detail while an assumption is made about stability.

3.1.5 Feasibility

Feasibility in the context of model predictive control encompasses the notions of initial and recursive feasibility. Initial feasibility pertains to the existence of a solution to the initial optimal control problem at $k = 0$, while recursive feasibility ensures that if a solution exists at $k = 0$, it remains attainable at all subsequent time steps $k > 0$.

We commence our exploration with the concept of initial feasibility, where we introduce the notion of admissible inputs, leading to the crucial concept of the feasible set. Additionally, we introduce two fundamental assumptions pivotal for establishing recursive feasibility and later, stability. Subsequently, we delve into recursive feasibility, incorporating further assumptions.

In the subsequent discussion, we consider the nominal scenario where the prediction model aligns with the real process, i.e., there is no error between the prediction model $\hat{f}(x, u)$ and the plant $f(x, u)$. For ease of narrative, we omit the estimation notation $\hat{(\cdot)}$ and the temporal dependency $(\cdot|k)$ throughout this section. Given the focus on the feasibility of optimal control problems, the differentiation between real and predicted variables becomes unnecessary. We occasionally employ $x^+ = f(x, u)$ as shorthand for $x_{k+1} = f(x_k, u_k)$. Much of the content presented here is informed by [21], [28], [29].

Initial Feasibility :

Initial feasibility in an MPC scheme pertains to whether the initial optimal control problem yields a solution. In static optimization, a problem of the form,

$$\underset{u \in U}{\text{minimize}}\, J(u)$$

admits a solution according to Weierstrass's theorem if J is continuous, and $\mathcal{U}$ is a compact and nonempty set [21],[30]. Here, $\mathcal{U}$ constitutes the feasible set.

In optimal control problems akin to $P_N(x_k)$, besides the constrained set $\mathcal{U}$, additional constraints arise in the form of a dynamical system $x^+ = f(x, u)$, along with constrained sets $\mathcal{X}$ and $\mathcal{X}_{\{}$. The objective is to ascertain a sequence of control inputs that guides the initial state x_k within a finite prediction horizon to the terminal region $\mathcal{X}_{\{}$ (and ultimately to the origin) at minimal cost while adhering to state and input constraints. Such an input sequence is termed admissible input(feasible input).

Definition 3.1 *(Admissible input) An input sequence $u = \{u_0, u_1, ..., u_{N-1}\}$ is admissible if, for the initial condition x_0, the resulting state sequence $x = \{x_0, x_1, ..., x_N\}$, and for all $i \in [0 : N-1]$, the conditions: (i) $u_i \in \mathcal{U}$ (ii) $x_i \in \mathcal{X}$ (iii) $x_N \in \mathcal{X}_{\{}$ hold, indicating satisfaction with input constraints and adherence to state and terminal constraints [27].*

It's worth noting that due to $\mathcal{X}_{\{} \subseteq \mathcal{X}$, $x_N \in \mathcal{X}$. Moreover, an admissible input need not necessarily minimize the cost function; it merely represents a candidate solution for the optimal control problem $P_N(x_k)$.

Admissibility of an input sequence is contingent upon a specific initial state x_0, with some initial states admitting admissible inputs while others do not. This distinction is particularly pertinent in MPC, where the initial value often represents the sole variable changing between the current and subsequent optimal control problems. This motivates the introduction of the feasible set.

Definition 3.2 *(Feasible set) The feasible set, denoted as $\mathcal{X}_{\mathcal{N}}$, encompasses all initial states x_0 for which at least one admissible input sequence u exists.*

$$\mathcal{X}_{\mathcal{N}} := \{x_0 \in \mathcal{X} : \exists u \text{ as per Definition 1}\}$$

By Definition 3.1, $\mathcal{X}_{\mathcal{N}} \subseteq \mathcal{X}$.

The optimal control problem $P_N(x_k)$ can only possess a solution if $x_k \in \mathcal{X}_{\mathcal{N}}$. However, this condition is necessary but not sufficient, as the admissible input sequences associated with points in $\mathcal{X}_{\mathcal{N}}$ may not necessarily minimize the cost function. Additional assumptions are thus required to ensure the existence of an optimal solution, i.e., an input sequence that satisfies constraints and minimizes the cost function [27].

Assumption 3.2 *(Continuity of system and cost)Functions $f(x, u)$, $\ell(x, u)$, and $V_f(x)$ are continuous.*

The initial value problem $x^+ = f(x, u)$ with initial condition x_0 admits a solution if $f(x, u)$ is continuous.

Assumption 3.3 *(Properties of constrained sets) Set $\mathcal{X}$ is closed, and sets $\mathcal{U}$ and $\mathcal{X}_{\{}$ are compact (closed and bounded), with each set containing the origin.*

With these assumptions, Proposition 3.1 can be asserted.

Proposition 3.1 *(Existence of solutions to OCPs) Suppose Assumptions 3.2 and 3.3 hold. Then, for each $x_k \in \mathcal{X_N}$, a solution to $P_N(x_k)$ exists.*

The complete proof is elaborated in [21]. In essence, Assumption 3.2 ensures continuity of the cost function $V_N(x_k, \hat{u}_{k|k})$, while Assumption 3 yields a compact set of admissible input sequences for every $x \in \mathcal{X_N}$. Application of Weierstrass's theorem consequently establishes the existence of a solution to $P_N(x_k)$ for every $x_k \in \mathcal{X_N}$.

Proposition 3.1 affirms the existence of solutions to optimal control problems for initial values x_k within $\mathcal{X_N}$. This implies that to ascertain whether a particular OCP admits a solution, only one must verify if the initial condition lies within $\mathcal{X_N}$, provided Assumptions 2 and 3 are satisfied. Furthermore, the guaranteed solution adheres to constraints and minimizes the cost function. If solely constraint satisfaction is considered, without minimizing the cost function, Assumptions 3.2 and 3.3 become unnecessary; $x_k \in \mathcal{X_N}$ suffices as a condition.

Remark 1: Some Remarks Regarding Proposition 3.1

- Proposition 1, adapted from [21], is a condensed version that also considers scenarios involving a closed but unbounded set U.
- Continuity of the cost function $V_N(x_k, \hat{u}_{k|k})$ doesn't automatically ensure continuity of the value function $V_N^*(x_k)$ (see [21]). Additionally, the conditions in Proposition 1 may be stringent depending on the specific application, and one might consider allowing for discontinuous value functions. However, this doesn't necessarily impede solutions to optimal control problems (see [21], [31], [32]).
- Proposition 3.1 doesn't specify how $\mathcal{X}_N$ can be computed. Its determination relies on various factors including constraints (e.g., $\mathcal{X}_N \subseteq \mathcal{X}$), prediction horizon length (longer horizon N implies larger $\mathcal{X}_N$), and system controllability. For instance, if the system is uncontrollable or the origin is an unstable equilibrium, $\mathcal{X}_N$ may be empty, rendering the optimal control problem unsolvable. Computing $\mathcal{X}_N$ is feasible primarily for simple cases like linear systems with polyhedral constraints. Further details on computing $\mathcal{X}_N$ are provided in [21], [33].

Recursive Feasibility :

In the preceding section, we discussed the conditions required to ensure the feasibility of an optimal control problem (OCP). Model Predictive Control (MPC) involves solving the same OCP at each time step with different initial conditions (repeated version with

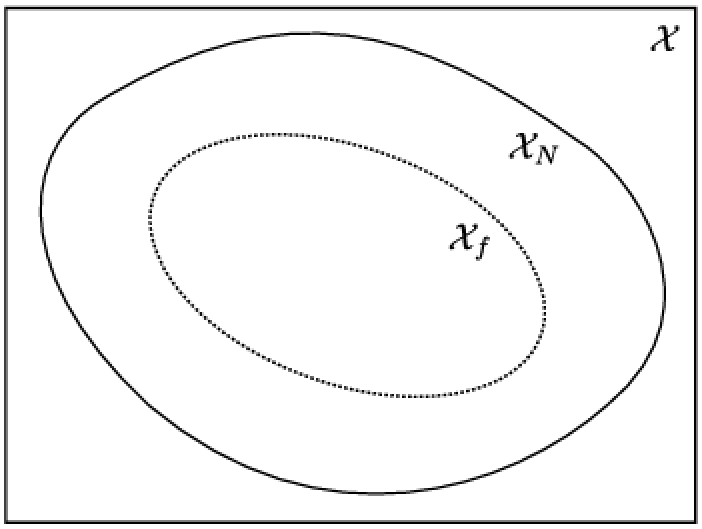


Figure 3.2: Illustration of the constrained set $\mathcal{X}$, the set of feasible initial conditions $\mathcal{X}_\mathcal{N}$, and the terminal region $\mathcal{X}_\{$ [27] .

different initial conditions). If the initial OCP at $k = 0$ has a solution, do all successive OCP also have a solution? This concept is termed recursive feasibility. It essentially asks whether the MPC controller can keep the state within the set $\mathcal{X}_\mathcal{N}$ over time.

However, achieving initial feasibility doesn't guarantee recursive feasibility. According to our earlier Definition 3.2, any point in $\mathcal{X}_\mathcal{N}$ can accommodate an admissible sequence pair (u, x) where x remains within $\mathcal{X}$ for the considered time interval. But since $\mathcal{X}_\mathcal{N}$ is typically a subset of $\mathcal{X}$, there's a possibility that x may exit $\mathcal{X}_\mathcal{N}$ at a certain instant, making the OCP infeasible for that specific time step.

Recursive feasibility is defined as the ability of the MPC controller to maintain feasibility for all time steps, given any initially feasible state $x_k \in X_N$ and any sequence of admissible control inputs u. To ensure this, we introduce the concept of positive and control invariant sets.

- A set $\mathcal{S}$ is positive invariant for $x^+ = f(x)$ if all x in $\mathcal{S}$ yield x^+ in $\mathcal{S}$.
- A set $\mathcal{S}$ is control invariant for $x^+ = f(x, u)$ if for all x in $\mathcal{S}$, there exists a u in U such that x^+ is in $\mathcal{S}$.

To achieve recursive feasibility, we need to ensure that the state x remains within $\mathcal{X}_\mathcal{N}$ for all time steps, i.e., $\mathcal{X}_\mathcal{N}$ must be control invariant for $x^+ = f(x, u)$ with the MPC control law $u = \kappa_{MPC}(x_k)$. The necessary assumption on this regard is the following

Assumption 3.4 *(Control Invariant Terminal Region:) The terminal region $\mathcal{X}_\{ \subseteq X$ is compact, contains the origin, and is control invariant for $x^+ = f(x, u)$, where $u \in \mathcal{U}$.*

Proposition 3.2 *It states that if Assumptions 3.2 to 3.4 hold, then X_N is positive invariant for $x_{k+1} = f(x_k, \kappa_{MPC}(x_k))$.*

For the complete proof of Proposition 3.2, refer to [21]. The underlying idea is to begin with $X_0 := X_f$ and define $X_1 = \{x \in X : \exists u \in U \text{ such that } f(x,u) \in X_0\}$.

The key approach to achieving recursive feasibility is to steer the state using MPC towards the terminal region X_f and then switch to a terminal controller $\kappa_f(x)$ that maintains X_f invariant. This terminal controller is not usually applied in practice; rather, it serves as a means to ensure recursive feasibility.

3.2 GP-MPC Framework

Gaussian process regression (GPR) was introduced as a powerful technique for estimating unknown output latent functions. The posterior mean function $m^{+}(w|D)$ serves as the desired estimator, providing a prediction of the output given the input data D. This posterior mean, along with the associated variance, is utilized in the Gaussian process transition model for long-term predictions, as outlined in equation (2.19). At each iteration of the algorithm, long-term predictions are computed using the Gaussian Process model trained on historical data. These predictions serve as the basis for formulating the optimal control problem[7]. The quadratic cost function is defined using these long-term predictions, capturing the trade-offs between control effort and performance objectives. Using the long-term predictions and the defined cost function, the optimal control problem is formulated. This problem incorporates state and control input constraints, ensuring that the resulting control inputs satisfy system limitations. The objective is to find control inputs that minimize the cost function while adhering to the constraints.

The formulated optimal control problem is solved using a nonlinear optimizer. This optimizer searches for optimal control inputs that minimize the cost function subject to the imposed constraints. Various optimization techniques, such as gradient-based methods or evolutionary algorithms, can be employed depending on the complexity of the problem. Once the optimal control inputs are obtained, the first element of the optimal control input sequence is applied to the plant's dynamics equation. The resulting states are measured, and the data set is updated with the new measured states. This updated data set is then used to train the Gaussian Process model for the next iteration of the algorithm. The entire GP-MPC framework, represented by Algorithm 1, is repeated over the prediction horizon N. This iterative process allows the control system to adapt to changing dynamics and uncertainties over time, resulting in effective and robust control performance.

In this section, we outline our approach for the recursive Gaussian Process-Model Predictive Control (GP-MPC) framework. In the recursive GP-MPC framework, firstly GP model learning, where we learn a prediction model only from measured input-output data using a Gaussian process, as explained in the section 2.1. Next, Controller design, where we design a Model Predictive Controller to tackle the control problem using the posterior distribution from the GP-trained model for long-term prediction. Additionally, we employ a nonlinear optimizer to solve the optimal control problem effectively.

Algorithm 1 Gaussian Process Model Predictive Control

```
Initialization:
  1. GP Parameters: Prior mean $m(\cdot)$, covariance function $K(\cdot,\cdot)$, likelihood function,
     inference function, initial hyperparameters $\theta$.
  2. MPC Parameters: Prediction horizon $N$, initial conditions, input constraints,
     state constraints, hard input constraint set $\mathcal{U}$, state constraint set $\mathcal{Y}$, Weighting
     matrices $Q$ and $R$.
  3. Data Set: Initialization required Data set $\mathcal{D}$.
Recursion: for $i \leftarrow 1$ to $N$ do
1: Initialization: initial input $u_0$ = first element $u_k = \kappa_{\text{MPC}}(\mathbf{x}_k|\mathcal{D}_k) = \hat{u}^*_{k|k}$.
2: Training the GP model: Repeat - for each state $x$ do
     • Optimize hyper parameters $\theta$ with initial data set $\mathcal{D}$.
     • Compute GP Posterior Eq. (2.19).
     • Predict mean $m(\cdot)$, covariance matrix $\Sigma(\cdot,\cdot)$ at time step $k+1$ using Posterior
       GP.
3: Model Predictive Controller Design
     • Compute long-term prediction over the control horizon.
     • Compute cost function see Section 3.1.3.
     • Formulate Optimal Control Problem see Section 3.1.4.
     • Solve OCP for initial conditions : state $x_k$, input $u_0$.
4: Update
     • Apply first element optimal control input $u_k = \kappa_{\text{MPC}}(\boldsymbol{x}_k \mid \mathcal{D}_k) = \hat{u}^*_{k|k}$ to plant.
     • Obtain new output $x_{k+1}$.
     • Construct new GP data point $(\tilde{\boldsymbol{x}}_k, x_{k+1})$ with $\boldsymbol{w}_k = (\tilde{\boldsymbol{x}}_k, u_k)$ to Data Set $\mathcal{D}$.
     • repeat.
```

3.2.1 GP Model Learning

In the recursive GP-MPC framework, the first step is GP model learning. Here, we aim to learn a prediction model solely from measured input-output data using a Gaussian process, as explained in Section 2.1. The Gaussian process regression is a powerful technique for estimating unknown output latent functions. Gaussian process regression allows us to model complex relationships between inputs and outputs without assuming a specific parametric form for the underlying function. This flexibility makes GP particularly suitable for modeling systems with nonlinear dynamics or uncertain behavior.

During GP model learning, we utilize the available input-output data to train the Gaussian

Algorithm 2 Training the Gaussian Process model

```
Input: $X_k$, $U_k$, $X_{k+1}$ Observation data
    $n_x, n_u$ number of state variables, inputs
    $cov \leftarrow \{@covSEard\}$
    $mean \leftarrow \{@meanLinear\}$
    $lik \leftarrow \{@likGauss\}$
    $inf \leftarrow \{@infGaussLik\}$
    $hyp_0$ initial hyperparameters
    number of conjugate gradient steps $N_{\text{cg}} = 100$
Output: $x_{k+1}(\mu, \sum)$
1: for $i \leftarrow 1$ to $n_x$ do
2:     if $N_{\text{cg}} == 0$ then
3:         $hyp \leftarrow hyp0$
4:     else
5:         $hyp$.state($i$) $\leftarrow$ fitGP(state($i$).$hyp0$, ..)    //hyperparameter optimization
6:     end if
7:     $x_{k+1}(i)[\mu, \sum] \leftarrow$ gp(state($i$).$hyp$, ...) // prediction of $x_{k+1}$ for each state
8: end for
9: return $x_{k+1}(\mu, \sum)$
```

process model. This involves estimating the hyperparameters of the kernel function and learning the posterior distribution over functions. The learned model provides predictions of the output given new input data, along with uncertainty estimates, which are crucial for robust control design. The posterior mean function $m^+(w|D)$ serves as the desired estimator, providing a prediction in the form of a gausiian distribution in terms of mean and variance. This posterior mean, along with the associated variance, is utilized in the Gaussian process (GP) transition model for long-term predictions, as outlined in equation (2.19).

To adapt to changing processes and environments, an online learning approach for the Gaussian process is introduced. This approach updates the training dataset during operation and optimizes the hyperparameters according to the dataset used to compute the posterior distribution. This ensures that the model remains accurate and effective in capturing the underlying dynamics of the system, even as they evolve over time.

Selecting appropriate hyperparameters is crucial for ensuring accurate predictions. The normal algorithm successfully identifies the minimum of the target function, demonstrating its efficacy in optimization tasks. However, it becomes evident that improper hyperparameter selection can hinder convergence, undermining the required assumptions for safe exploration. Interestingly, even with larger initial length scales, convergence is attainable across different starting points, provided the non-constant segments are adequately evaluated. Subsequently, during initial optimization, these length scales are reduced, gradually fine-tuning with an increasing number of data points to better approximate the target function.

Employing gradient-based optimizers alongside larger length scales and elevated noise standard deviation (σ_n) poses a risk: the maximum likelihood estimate may falter, converging to a local minimum instead of the desired global minimum, as depicted. This scenario necessitates careful consideration and the adoption of strategies outlined in Section 2.4 to mitigate such challenges. The `fitGP` function is employed for this purpose, which optimizes the hyperparameters using the gradient descent method. The literature ([34]) extensively discusses approaches to address converging to a local minimum issue, emphasizing the importance of tailored methodologies for robust optimization. These strategies encompass various techniques aimed at enhancing algorithmic resilience and fostering convergence towards global minima, thereby bolstering the reliability and efficacy of optimization processes.

In summary, the report discusses the utilization of Gaussian process regression for long-term prediction, introducing an online learning approach for adaptation to changing environments. The `fitGP` function is highlighted as a tool for efficiently optimizing hyperparameters, ensuring accurate predictions even as the system dynamics evolve.

In summary, the algorithm of Gaussian Process model is shown in detail in Algorithm 2 training the report delves into the application of Gaussian process regression (GP) for long-term prediction, offering insights into its effectiveness in capturing complex relationships between inputs and outputs without relying on specific parametric forms. The introduction of an online learning approach enhances the adaptability of GP models to evolving environments, allowing for continuous updates and refinement based on incoming data.

Furthermore, robust optimization strategies play a crucial role in model training, especially in mitigating the risk of converging to local minima during the optimization process. By carefully selecting hyperparameters and employing gradient-based optimizers alongside larger length scales and elevated noise standard deviation, the algorithm can navigate complex optimization landscapes more effectively, ultimately improving the overall performance and efficacy of the predictive modeling framework. To ensure the reliability and adaptability of long-term prediction models in dynamic environments with uncertainty, the better performance and efficacy of the GP model are crucial, which highlighting the importance of adaptive learning approaches and efficient hyperparameter optimization techniques.

3.2.2 Controller Design

Following GP model learning, the next step in the recursive GP-MPC framework is Controller design. Here, we leverage the learned Gaussian process model to design a Model Predictive Controller for tackling the control problem. MPC is a control strategy that repeatedly solves an optimization problem over a finite prediction horizon, taking into account the system dynamics and constraints, to generate optimal control inputs.

Algorithm 3 Function: long-term prediction

```
Input: augmented data matrix $\tilde{X}_k = [X_k, U_k]$,
       one-step ahead state data $Y = X_{k+1}$,
       augmented mean matrix $\tilde{\boldsymbol{\mu}}_{k+1} = \left[\mu_{x_{k+1}}, u^*\right]$,
       augmented covariance matrix $\tilde{\sum}_{k+1} = \left[\Sigma_{x_{k+1}} \quad 0\right]$,
       GP hyperparameters $\Lambda, \sigma_w, \sigma_f$,
       $n_x, n_u$ number of state variables, inputs
Output: $x(\mu^*, \sum^*)$
 1: for $k = 1$ to $n_x$ do
 2:     for $i = 1$ to $n_x$ do
 3:         $\boldsymbol{\beta}_i \leftarrow \left(\boldsymbol{K}_i + \sigma^2_{w_i}\right)^{-1} Y_i$
 4:         $\nu \leftarrow \left(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{\mu}}_{k+1}\right)$
 5:         Compute $q_i$            // equation (2.22)
 6:         $R_i \leftarrow \tilde{\Sigma}_t \left(\Lambda_a^{-1} + \Lambda_b^{-1}\right) + I$
 7:         $T_i \leftarrow \Lambda_a^{-1} + \Lambda_b^{-1} + \tilde{\Sigma}_t^{-1}$
 8:         $z_i \leftarrow \Lambda_a^{-1}. * \nu + \Lambda_b^{-1}. * \nu$
 9:         Compute $Q_i$            // equation (2.29)
10:     end for
11:     Calculate $\mu_k$            // equation (2.20)
12: end for
13: for $i = 1$ to $n - 1$ do
14:     for $j = 1$ to $n - 1$ do
15:         if $i == j$ then
16:             Calculate $\sum(i, i)$            // equation (2.24)
17:         else
18:             Calculate $\sum(i, j)$ and $\sum(j, i)$            // equation (2.25)
19:         end if
20:     end for
21: end for
```

$$\begin{aligned}
&\underset{u_k^*}{\text{minimize}}\ J(k) = \sum_{k=i}^{i+N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) \\
\text{subject to}& \\
&x_{k+1} = f(x_k, u_k) + \epsilon \quad \text{for } k = i \\
&u_{\min} \leq u_k \leq u_{\max} \quad \text{for } k = i, \ldots, i+N-1 \\
&A x_k \leq B \quad \text{for } k = i, \ldots, i+N-1
\end{aligned} \tag{3.6}$$

This enables us to incorporate uncertainty into the control decision-making process, leading to more robust and adaptive control strategies. The formulation of the optimal control problem holds significant importance in the design of a controller. At its core, the optimal control problem aims to determine the best sequence of control inputs over time to optimize a certain objective while adhering to system dynamics and constraints. This problem formulation is crucial for designing efficient and robust control strategies in complex systems where achieving desired performance while considering limitations and uncertainties is paramount. As discussed in Section 3.1.4, we need to ensure the feasibility and stability of the optimal control problem. The modified Model Predictive Control optimal control problem can be seen in Equation (3.6). We aim to make states zero with minimal control input, i.e., our set point is zero. If we want to track the reference signal or change the set point to another value, then we need to modify Equation (3.6) with $(x_k - x_{\text{ref}})$. The simple state and control input constraints are subjected to an optimal control problem (OCP). To solve the OCP, it requires state prediction over the prediction horizon from $k = i, \dots, i + N - 1$, these predictions should be accurate and efficient in defining the actual dynamics of the plant model with uncertainty.

In our framework, we use the posterior distribution obtained from the Gaussian Process -trained model for long-term prediction within the Model Predictive Control formulation. This enables us to incorporate uncertainty into the control decision-making process, leading to more robust and adaptive control strategies. For this, we introduce moment-matching approximation as a long-term prediction model which provides more accurate and efficient predictions considering uncertainty into account. The concept is very simple: predicting the control inputs using GP hyperparameters and posterior prediction, as outlined in Algorithm 3.

A key aspect of the GP-MPC formulation is the utilization of the predicted mean and variance from the GP model for online predictions. The moment-matching approximation facilitates efficient computation of these quantities, enabling real-time decision-making in control applications. All the predictions from moment-matching approximation are Gaussian distributed in terms of mean and variance. By focusing on the mean prediction and omitting the consideration of variance dynamics for simplicity, the GP-MPC framework strikes a balance between computational complexity and predictive accuracy.

In the Gaussian Process Model Predictive Control (GP-MPC) framework, leveraging Gaussian process regression for long-term prediction introduces a flexible and probabilistic approach to modeling system dynamics. By incorporating the GP model alongside the nominal model $f(x_k, u_k)$, the GP-MPC problem formulation extends beyond traditional deterministic models, offering enhanced adaptability to uncertain or nonlinear system behaviors.

Furthermore, constraint handling in the GP-MPC formulation is crucial for ensuring the feasibility and stability of the control solutions. Finally, a nonlinear optimal control problem is formulated, which can be solved to find optimal control inputs. Additionally, to solve the optimal control problem effectively, we employ a nonlinear optimizer, which efficiently handles the nonlinearities and constraints inherent in many real-world control problems. The nonlinear optimizer that we used is `fmincon`, and there are other nonlinear optimizers [23]. The solution, which is optimal control inputs, satisfies specified bounds while optimizing performance objectives, thereby enhancing the robustness and safety of the control system. Finally, the MPC controller is designed and formulated as outlined in Algorithm 4.

Overall, the integration of Gaussian process regression into model predictive control offers a promising avenue for addressing challenges associated with uncertain and nonlinear system dynamics. By leveraging probabilistic modeling techniques and advanced optimization methods, GP-MPC facilitates robust and adaptive control strategies capable of handling real-world complexities and uncertainties. Continued research in this area holds the potential to further advance the capabilities and applicability of GP-MPC in diverse engineering and autonomous systems domains.

Algorithm 4 MPC controller

```
Input: $u_0$ initial control inputs
       $X_k$, $U_k$, $X_{k+1}$ observation data
       $\mu_{x_{k+1}}, \Sigma_{x_{k+1}}$ posterior GP prediction
       $Q$: Array of weightings for the state variables
       $R$: Weighting for the control inputs
       GP hyperparameters
       $A, B$ state constraints matrices
       $U_a, U_b$ control input constraints
Output: $u^*$ optimal control inputs
  $fun \leftarrow @(u0) \quad COST(...)$          cost function
  $u^* \leftarrow fmincon(fun)$          // Nonlinear optimization

 1: function COST($u_0, \tilde{X}, \mu, \Sigma, hyp, X_{k+1}, Q, R$)
 2:     Prediction horizon $N$
 3:     $J \leftarrow 0$
 4:     for $i = 1$ to $N$ do
 5:         if $i == 1$ then
 6:             $\tilde{\mu} \leftarrow [\mu_{k-1}, u_0(i)]$
 7:             $\tilde{\Sigma} \leftarrow \text{blkdiag}(\Sigma, 0)$
 8:         else
 9:             $\mu \leftarrow [\mu^*(i-1), u_0(i)]$
10:             $\tilde{\Sigma} \leftarrow \text{blkdiag}(\Sigma^*(i-1), 0)$
11:         end if
12:         $[\mu^*(i), \sum^*(i)] \leftarrow$ long term prediction function          //algorithm 3
13:         $J \leftarrow J + \mu^*(i) \cdot Q \cdot \mu^*(i) + u_0(i) \cdot R \cdot u_0(i)$          // equation (3.3)
14:     end for
15:     return $J$
16: end function
```

4 Simulation Results

This section primarily focuses on the application of the GP-MPC framework to two well-known examples: the DC motor and the Van der Pol oscillator. The main advantage of the GP-MPC framework lies in its versatility, as it can be applied to a wide range of systems, whether linear or nonlinear, with or without uncertainty, and with or without noisy measurements. Model uncertainty refers to parameter variations and external disturbances that affect the system's behavior. These mathematical uncertainty models represent all aspects of real-world systems, leading to uncertainty in predicting their behavior. To demonstrate its effectiveness, we have chosen to implement the GP-MPC framework on both a linear system, the DC motor, and a nonlinear system, the Van der Pol oscillator.

For each system, the following results are plotted and compared:

- Validation of the GP model and the long-term prediction model, ensuring their accuracy in predicting system behavior.
- Modeling without uncertainty, where the system dynamics are described by deterministic equations.
- Modeling with uncertainty, where stochastic elements and uncertainty parameters are introduced into the system equations to simulate real-world scenarios.

In validation of the Gaussian Process (GP) model, the posterior mean is used to predict the states. while in the validation of the long-term prediction model, Algorithm 3 is used to predict states and each time step. To provide a comprehensive comparison, we also directly apply Model Predictive Control (MPC) to the dynamics of each system. By comparing the results of GP-MPC with direct MPC, we can evaluate the performance of the learning process within the GP-MPC framework. By examining these results, we can gain insights into the performance of the GP-MPC framework and its ability to adapt to different system characteristics and conditions.

4.1 DC Motor

DC motors offer an appealing option over AC servo motors for demanding motion control tasks, particularly in low-power, high-precision applications due to their cost-effectiveness and simplicity in management. Conventionally, industrial motor controls employ a cascade control setup, where outer speed and inner current control loops are typically designed using PD or PI controllers. The cascade control setup consists of a parallel controller with an inner current loop and an outer speed loop. However, the authors assume that the inner current loop controller is sufficiently faster than the outer speed loop controller [35].

Recent literature suggests alternative strategies for identifying and controlling DC motors. Umeno and Hori [36] introduce a generalized speed control design approach for DC servomotors, utilizing the parametrization of two-degree-of-freedom controllers. They apply this method, incorporating a Butterworth filter, to ascertain controller parameters.

The angular velocity, $\omega = \dot{\theta}$, is regulated by the input voltage, v, with a consistent voltage drop attributed to brush and rotor resistance, along with a back-electromotive force (EMF) stemming from the rotary armature. The motor inductance contributes proportionally to the change in motor current, i. Motor current links the electrical and mechanical components, generating driving torque. This torque is counteracted by motor inertia, structural damping, friction, and external loads[37].

There are nonlinear effects, which can significantly impact the dynamic behavior of the modeled system. Hence two major assumptions are made, firstly the magnetic circuit is linear and another assumption is that the mechanical friction is only linear in the motor speed. These assumptions help to simplify the DC motor model and make it more amenable to Gp-MPC controller design [38].

The motor dynamics are defined by:

$$V(t) = L\frac{di}{dt} + R_m i(t) + K_e \omega(t) \tag{4.1}$$

$$\frac{d\dot{\theta}}{dt} = -\frac{b_v}{J}\dot{\theta} + \frac{K_m}{J} i \tag{4.2}$$

Where K_m, K_e, and b_v represent the motor torque, back-EMF, and damping constants respectively. J denotes mechanical inertia including the motor armature and shaft. L and R_m represent motor inductance and total connection resistance.

To find the physical parameters such as K_m, K_e, b_v,, R_m and J, the experiment is conducted in this [39], where the friction and induction are neglected ($b_v = 0$,$L = 0$). This experiment was conducted on one specific DC motor, where the transfer function for that specific device is formed by applying a voltage to the DC motor and measuring angular velocity. Finding the motor physical parameters from known measured data. The final transfer function of the DC motor found using the experiment [39] is

$$G(s) = \frac{21}{s(1.1s + 1)} \tag{4.3}$$

From the DC motor transfer function Equation (4.3), state-space equations are modeled in continuous time Equation (4.4).

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -0.9091 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 4 \\ 0 \end{bmatrix} u \\ y &= \begin{bmatrix} 0 & 4.7727 \end{bmatrix} x \end{aligned} \tag{4.4}$$

As discussed before, the Model Predictive Control (MPC) can primarily be applied to discretized systems, it's worth noting that discrete-time models offer certain advantages. Some approaches implement MPC in continuous time systems [40]. The discrete systems are straightforward to implement for digital controllers and are essential for stable control system design. Moreover, they find extensive use in digital communication systems, contributing significantly to various engineering applications.

As discussed before we consider discrete-time model due to their alignment with sampled data, compatibility with digital systems, computational efficiency, and availability of analysis tools. They offer straightforward implementation for digital controllers and are essential for stable control system design. Additionally, they are well-suited for digital communication systems, making them indispensable in various engineering applications. The continuous time model Equation (4.4) is converted to a discrete-time model with a sampling time of $T_s = 0.1s$ with exact discretization method(zoh), as outlined in the equation (4.5).

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} 0.9131 & 0 \\ 0.0956 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.3824 \\ 0.0194 \end{bmatrix} u_k \\ yk &= \begin{bmatrix} 0 & 4.7727 \end{bmatrix} x_k \end{aligned} \tag{4.5}$$

The discrete-time equation represents a system used for generating data. It consists of state equations and output equations. The output equations are ignored and considered the state equation for modeling the GP-MPC controller. Now, the state update (X_{k+1}) is the output data, whereas the augmented matrix $\tilde{X_k} = [X_k, U_k]$ is the input data. It has two states, the first state (x_1) is the angular velocity ($\dot{\theta}$), and the second state (x_2) is the angle (θ) whereas the control input is voltage (v). we assumed initial states are $x_1 = 0$ and $x_2 = \pi$ and the random control input is applied to the plant Equation (4.5) and states are measured. The size of the generated data is very small with 20 interactions. A small data set is used to validate our model data efficiency. Taking a small set of data can be advantageous for resource efficiency, faster learning, and focus on important features, while still gaining valuable insights and informing decision-making processes.

4.1.1 Validation of DC Motor Learning

Validation of Gaussian Process (GP) and Long-term prediction models is crucial to assess their reliability and accuracy. The accuracy of the long-term prediction model ultimately relies on the reliability of the GP model. GP models serve as valuable tools for regression, particularly when dealing with limited data, robust models, or noisy data. Therefore, we will focus on a plant with noisy measurements. Additionally, we will consider parameter uncertainty[41], resulting in modified plant state equations as follows:

$$\begin{aligned} & x_{k+1} = A_d x_k + B_d u_k + \epsilon_d \\ & \text{where,} \\ & A_d = A + \lambda_a A \\ & B_d = B + \lambda_b B \end{aligned} \tag{4.6}$$

Here, A_d and B_d matrices represent disturbed matrices affected by uncertainty variables λ. The λ_a and λ_b are small parameters representing model uncertainty. To validate the Gaussian model under realistic conditions, we set λ_a and λ_b as -0.1*random distribution in the interval (0,1), introducing maximum uncertainty to the plant dynamics. The ϵ_d is Gaussian noise, which is a small number in the order of 10^{-3}. Thus, the new

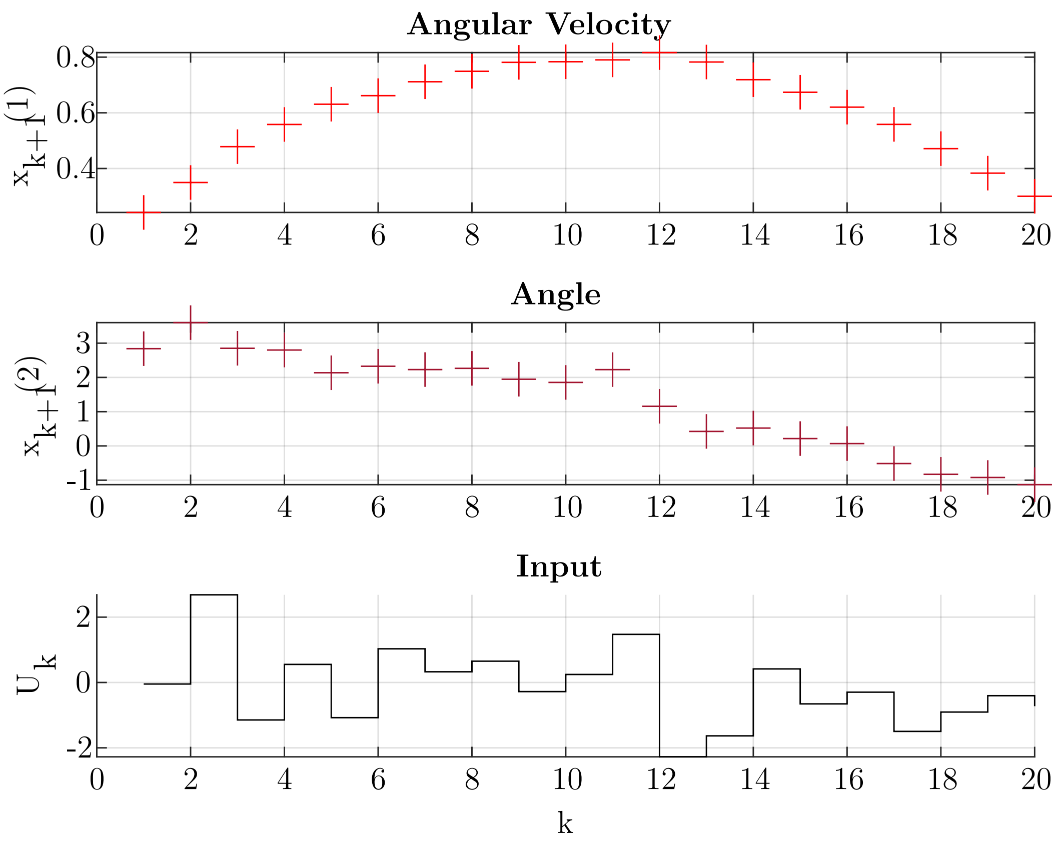


Figure 4.1: Training set,X_{k+1} and U_k for model learning validation generated using DC motor uncertainty model

model incorporates model uncertainty and measurement noise, resembling a realistic plant scenario.

The data was generated using the uncertain plant model in Equation 4.6 for a random set of input u_k. The data X_k and U_k are augmented and used as input for the GP training model, while X_{k+1} acts as output. The generated data is plotted in Figure 4.1. With a dataset size of 20, which is relatively small for model training, it's suitable for validation purposes.

The trained GP model is tested with 60 random inputs $\hat{u_k}$. At each time step k, x_k, u_k is used to predict x_{k+1} and the only predicted mean is considered for the next prediction, the variance is neglected. The predicted mean and variances are plotted alongside the control input, as shown in Figure 4.2. The predicted mean is compared with actual values from the uncertainty model (Equation 4.6). The root mean square error (RMSE) is calculated using Equation (4.7) and RMSE values of state x_1 is 0.96%, and for state x_2 is 2.11%. The average variances of predicted x_1 and x_2 are 2.0262×10^{-5} and 1.1588×10^{-5} respectively. Both RMSE values and average variances are very small, indicating high accuracy. Overall,

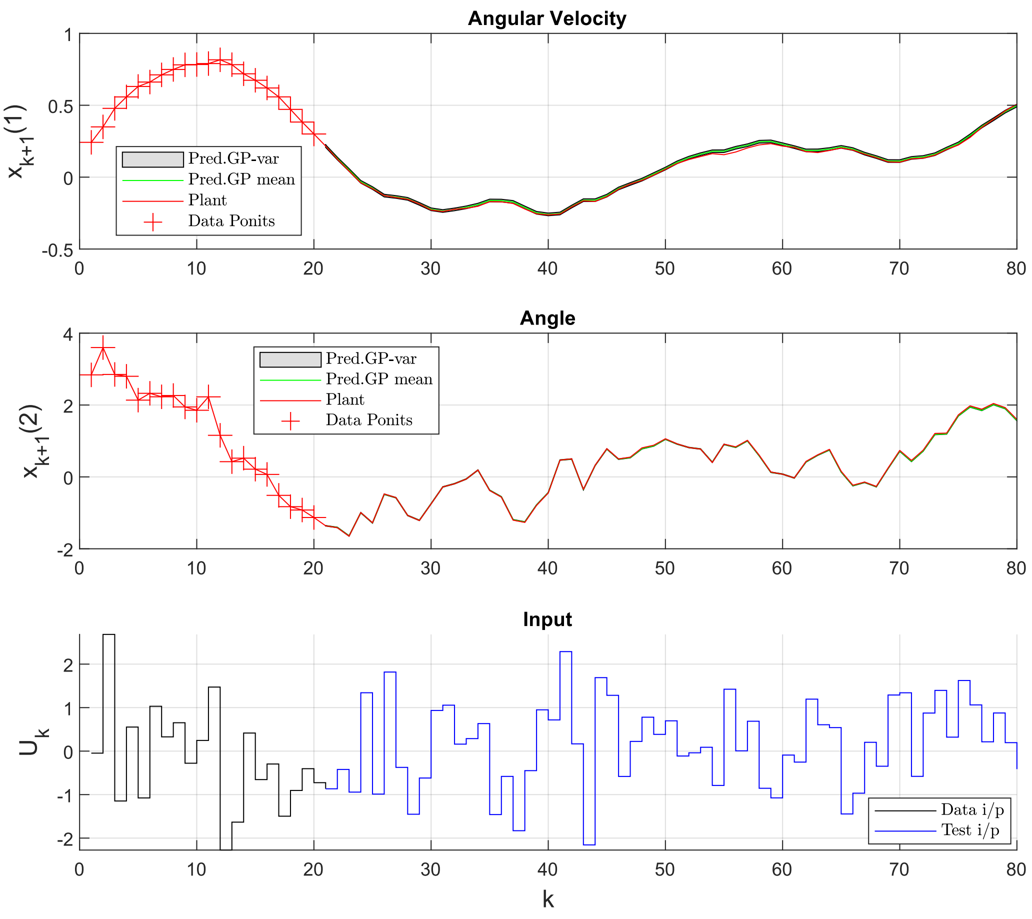


Figure 4.2: Gaussian Process model testing of DC motor

validation confirms that the GP model is reliable and accurate, and capable of making trustworthy predictions on unseen data in real-world scenarios.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}} \tag{4.7}$$

The model is tested with the same 60 random inputs $\hat{u_k}$. The predicted mean and variance are considered for the next prediction, where variance accounts for the reliability of the prediction. The predicted mean and control inputs are plotted in Figure 4.3. These predictions are compared with actual values from the uncertainty model Equation (4.6). The RMSE of state x_1 and state x_2 are 11.74% and 57% respectively.

The validation process conducted on the Gaussian Process (GP) model and long-term prediction model has provided valuable insights into their reliability and accuracy, particularly in the context of noisy and uncertain plant dynamics. The GP model, utilized for regression tasks, has demonstrated high reliability and accuracy in capturing the underlying dynamics of the plant. By incorporating parameter uncertainty and measurement noise,

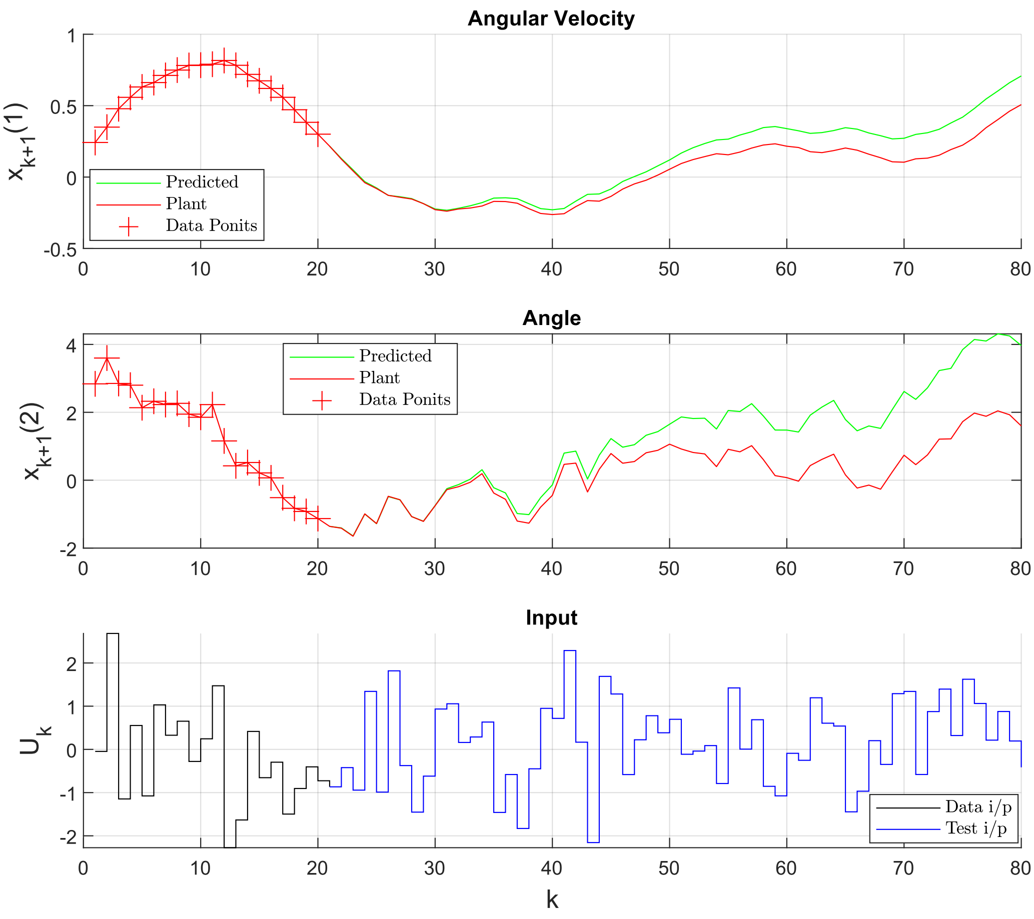


Figure 4.3: Longterm prediction model testing

the GP model effectively adapts to realistic scenarios, as evidenced by the small root mean square error (RMSE) values and average variances for both states. The main difference between the Gaussian Process (GP) model and the long-term prediction model, is that the long-term prediction model takes input as gaussian $\mathcal{N}(x^*, \Sigma_x^*)$ and gives output as gaussian $\mathcal{N}(x_{k+1}^*, \Sigma_{k+1}^*)$, whereas GP prediction model gives gaussian output but it cant take gaussian as input, so only mean is given as input, neglecting the variance. Even though the GP model demonstrates higher predictive accuracy, it neglects variance, which provides critical insight into the reliability and confidence levels of the model's estimations.

Moreover, the validation of the long-term prediction model is not perfect. In the early stages of the prediction horizon, the predictions are accurate, but as time progresses, the deviation becomes more pronounced due to the increase in the variance at each time step. Despite deviations from actual measurements, the Gaussian distributed predictions exhibit sufficient accuracy for modeling the GP-MPC controller because the considered prediction horizon is small. However, it's essential to acknowledge that as the prediction horizon increases, the effects of uncertainty and noisy measurements may lead to further

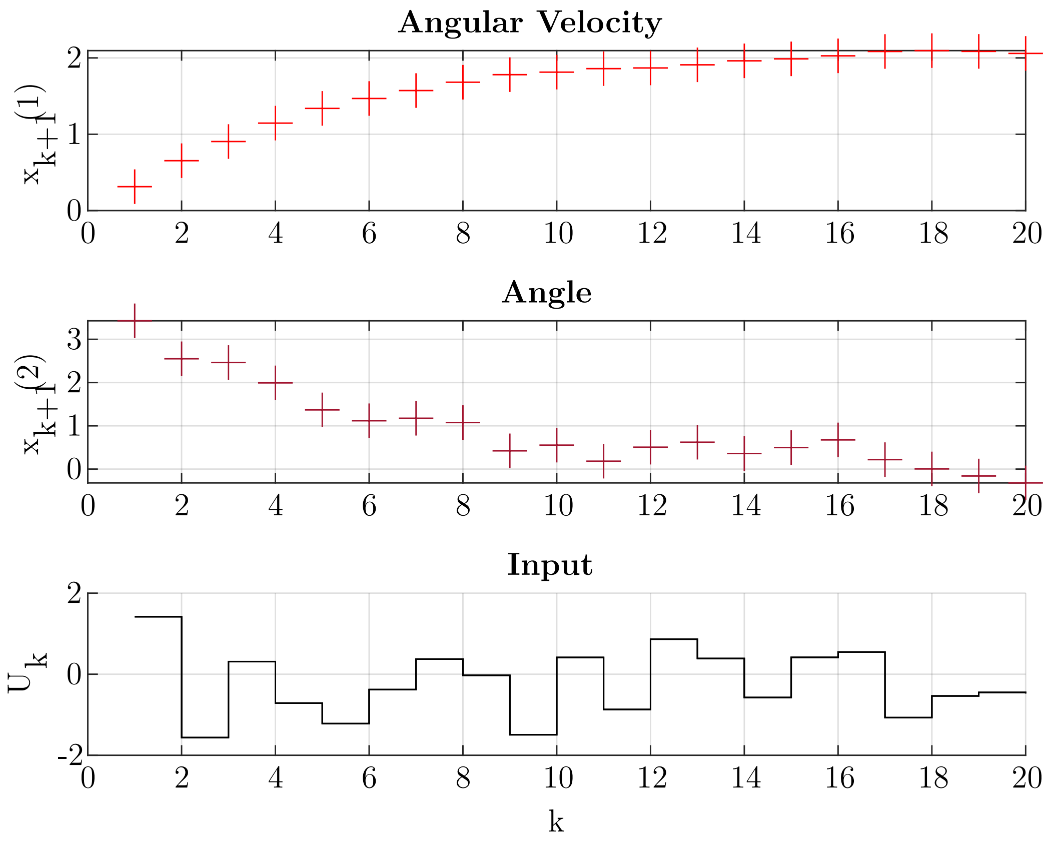


Figure 4.4: Data points from a noise-free DC motor plant

discrepancies between predicted and actual values. Overall, the validation process confirms the reliability and accuracy of the GP model and long-term prediction model for making trustworthy predictions in real-world scenarios with noisy and uncertain data for linear models.

4.1.2 Modelling Without Uncertainty

Modeling a system without uncertainty typically involves focusing solely on its deterministic aspects, disregarding any stochastic or random disturbances. Such a deterministic model accurately captures the system's behavior without considering noise or disturbances, facilitating analysis and simulation under ideal conditions.

In contrast to systems affected by noise, noise-free systems allow for clearer insights into the underlying dynamics and behavior. By eliminating stochastic influences, deterministic models provide a precise framework for understanding the system's response to inputs and its evolution over time. This clarity is particularly advantageous in scenarios where noise is either negligible or can be effectively accounted for through other means, such as through robust filtering techniques or noise compensation strategies.

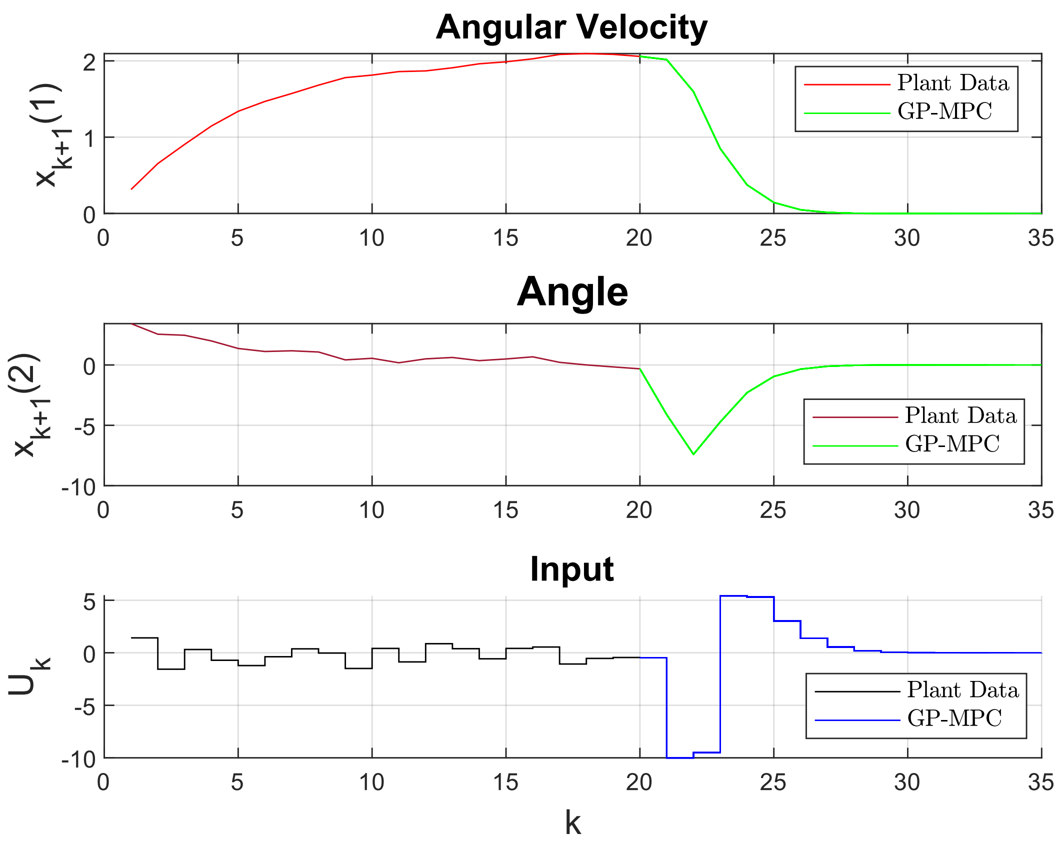


Figure 4.5: GP-MPC controller- noise free DC motor plant without parameter uncertainty

The dataset was generated using the standard plant model without any additional noise, as described in Equation (4.5), with a randomly selected set of input values u_k. The dataset, consisting of X_k and U_k, was then combined and utilized as input for training the Gaussian Process (GP) model, while X_{k+1}(generated data) served as the output. The resulting dataset, comprising 20 data points, is visually represented in Figure 4.4.

As discussed in Section 3.2.2, the optimal control problem for the DC motor can be represented by Equation (4.8).

$$
\begin{aligned}
&\underset{u_k^*}{\text{minimize}}\; J(k) = \sum_{k=i}^{i+N-1} (x_{k+1}^T Q x_{k+1} + u_k^T R u_k) \\
&\text{subject to} \\
&\qquad x_{k+1} = A x_k + B u_k \quad \text{for } k = i \\
&\qquad x_0 = X_{k+1}(end) \quad //\text{end point of dataset} \\
&\qquad -10 \leq u_k \leq 10 \quad \text{for } k = i, \ldots, i+N-1 \\
&\qquad u_0 = U_k(end)
\end{aligned}
\tag{4.8}
$$

Here, the objective is to control the states angle and angular velocity, aiming to minimize

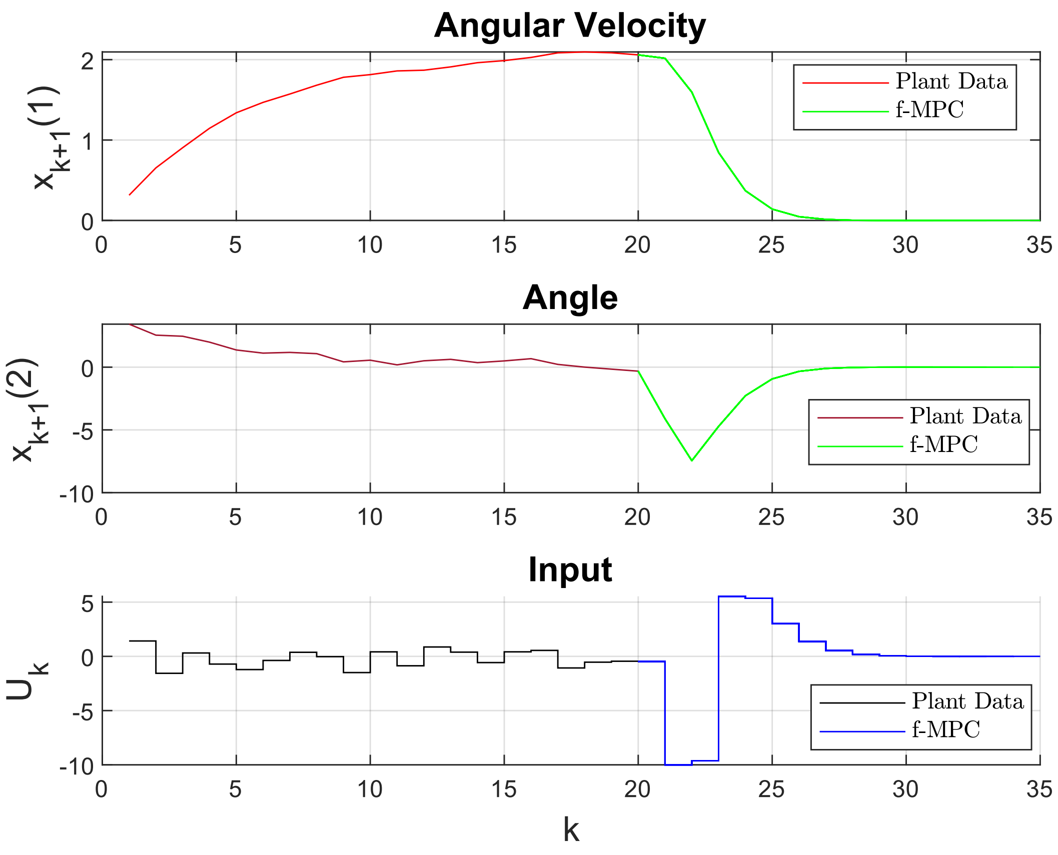


Figure 4.6: f-MPC controller- DC motor plant without uncertainty

the states to zero with minimal control input. The optimal control problem is defined over a prediction horizon of 15 timesteps. The initial states and control input are given as the end point of the dataset, and the initial control input for GP-MPC controller constraint is defined as $-10 \leq u_k \leq 10$. The weighting matrices Q and R are adjusted to ensure that the controller operates quickly with minimal input.

The application of GP-MPC to the noise-free(without measurement noise) DC motor system, as illustrated in Figure 4.5, further demonstrates the efficacy of deterministic modeling in control design. Through iterative optimization, the GP-MPC controller leverages the noise-free system dynamics to achieve precise control performance, even in the absence of stochastic disturbances.

The same Model Predictive Control (MPC) framework is applied to the dynamics function of the DC motor described by Equation (4.5), with identical optimal control problem formulations and constraints as represented in Equation (4.8). These results are depicted in Figure 4.6. Direct MPC results are compared against GP-MPC results to ensure the accuracy of the learning process of GP-MPC framework. The comparison between the plots in Figure 4.5 and Figure 4.6 reveals that the behavior of the GP-MPC closely resembles that of direct MPC, validating the learning process of GP-MPC.

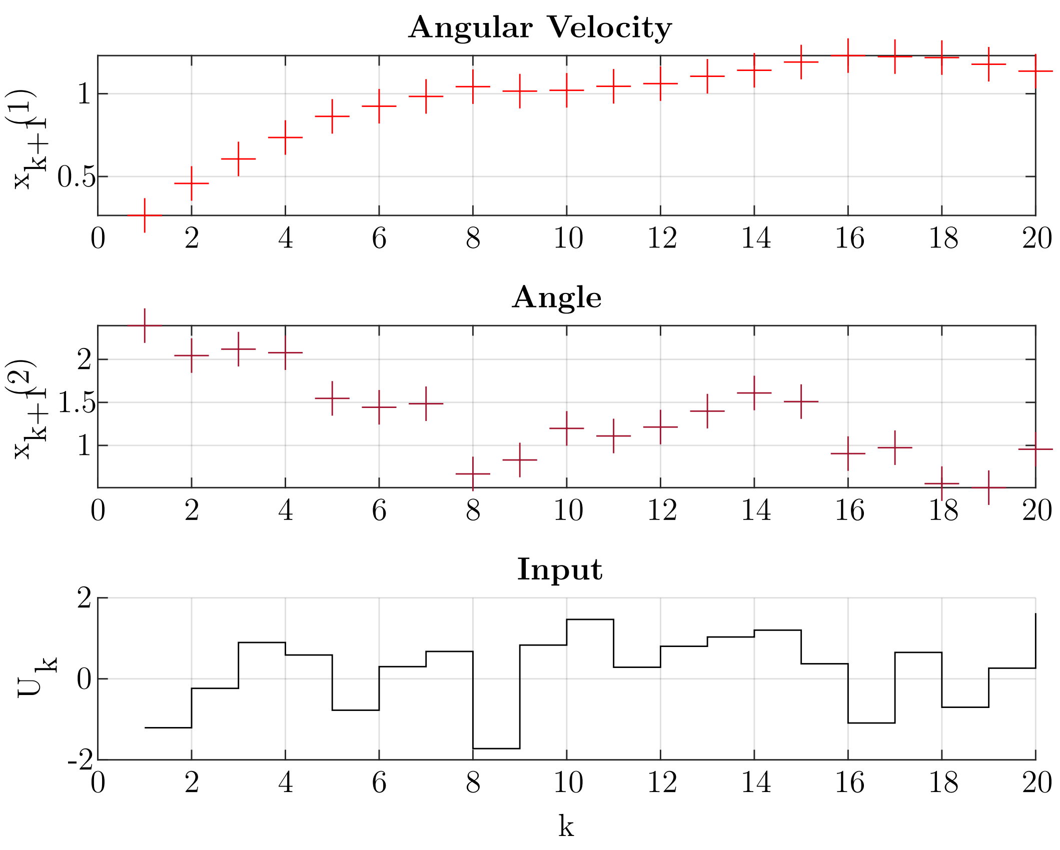


Figure 4.7: Data points from a noisy and uncertain DC motor plant

The similarity between the GP-MPC and direct MPC results underscores the effectiveness of Gaussian Process-based modeling in capturing the underlying dynamics of the DC motor system. By leveraging machine learning techniques, GP-MPC can accurately predict system behavior and generate control actions, offering a viable alternative to traditional control methods.

4.1.3 Modelling With Uncertainty

Modeling dynamic systems with noise and uncertainty is crucial for capturing real-world complexities and enhancing predictive accuracy. Noise represents random disturbances in the system or measurements, while uncertainty includes unknown or variable factors affecting system dynamics. Incorporating these elements into models provides a more realistic representation of system behavior and supports better-informed decision-making. Therefore, we will focus on a plant with noisy measurements. Additionally, we will consider parameter uncertainty, as shown in Equation 4.6. Modeling with and without uncertainty should be the same for the GP-MPC framework because the Gaussian Process (GP) always learns the model directly. This continuous, direct learning process allows the GP to adapt

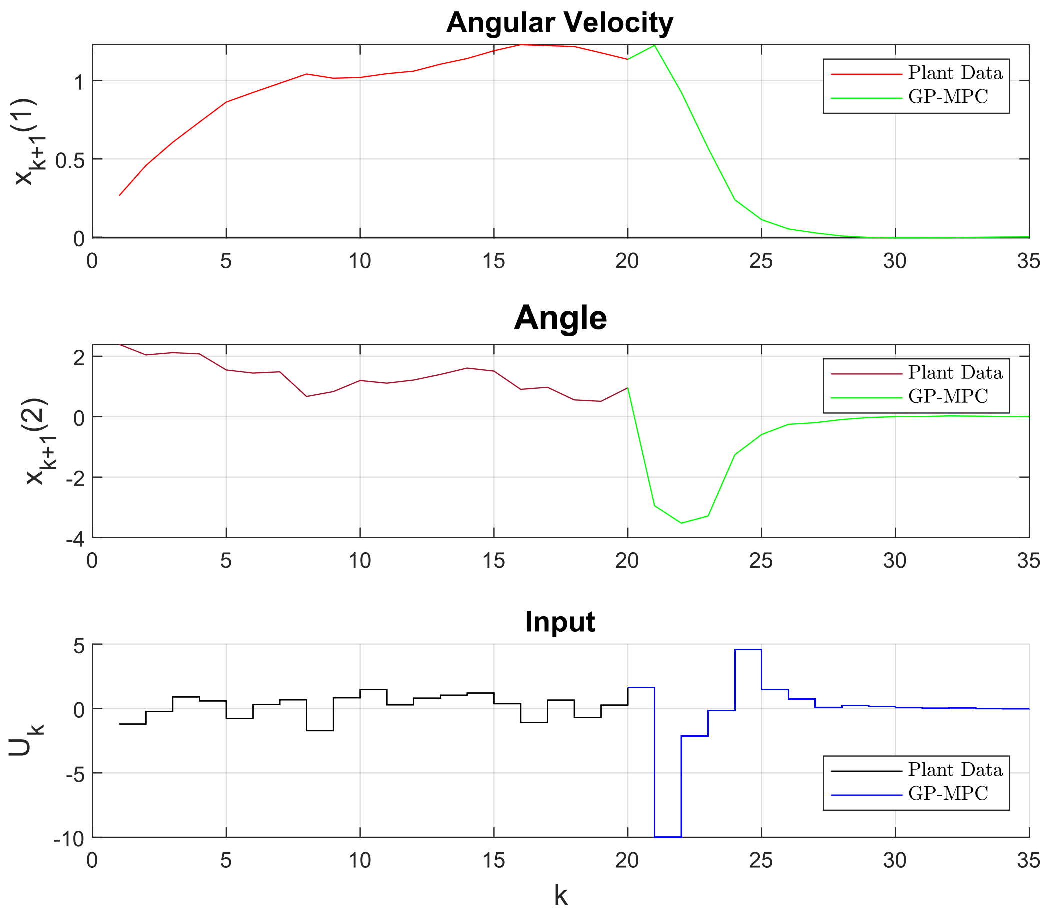


Figure 4.8: GP-MPC controller- DC motor plant with uncertainty and noisy meaurements

to and incorporate new data in real time, irrespective of whether uncertainty is explicitly modeled.

Figure 4.7 illustrates data points collected from a dynamic system affected by noise and parameter uncertainty. The presence of noise and uncertainty introduces variability and unpredictability into the system's behavior, leading to deviations from idealized models. Consequently, accurate modeling of such systems requires accounting for stochastic processes and uncertain parameters.

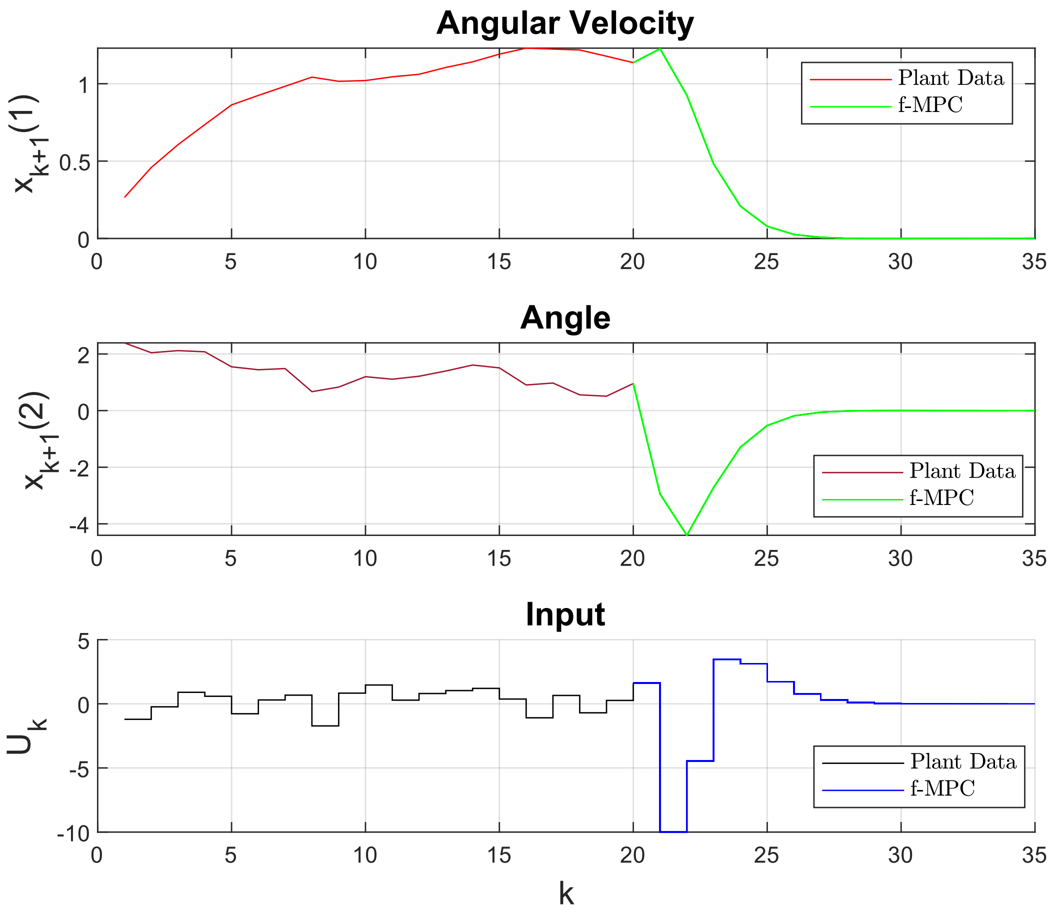


Figure 4.9: f-MPC controller- DC motor plant with uncertainty and noisy measurements

$$
\begin{aligned}
&\underset{u_k^*}{\text{minimize}}\ J(k) = \sum_{k=i}^{i+N-1} (x_{k+1}^T Q x_{k+1} + u_k^T R u_k) \\
&\text{subject to} \\
&\quad x_{k+1} = A_d x_k + B_d u_k + \epsilon \quad \text{for } k = i \\
&\quad x_0 = X_{k+1}(end) \qquad //\text{end point of dataset} \\
&\quad -10 \le u_k \le 10 \quad \text{for } k = i, \dots, i+N-1 \\
&\quad u_0 = U_k(end)
\end{aligned} \tag{4.9}
$$

In Section 3.2.2, the optimal control problem for the DC motor is described by Equation (4.9). This problem aims to regulate both the angle and angular velocity states of the motor. It is formulated over a prediction horizon spanning 15 steps. The initial states and control input are specified as the endpoint of the dataset, while the control input is bounded within the range of $-10 \le u_k \le 10$. To achieve a responsive and efficient controller, the weighting matrices Q and R are carefully adjusted, emphasizing quick control response with minimal input.

Figure 4.8 illustrates the learning process of the GP-MPC controller applied to the uncertainty model of the DC motor. Despite considering maximum uncertainty and noisy measurements, the performance of the MPC controller remains robust. Remarkably, it is comparable to the performance of the MPC controller without noise, as shown in Figure 4.5. Even though there is model uncertainty, the GP-MPC model adapts to the uncertainty quickly fast and controls over a few timesteps. The settling time of angular velocity is $t_s = 9$ and the settling time of angle is $t_s = 11$. Both states are faster reaching the set point within 11 timesteps. One key factor contributing to the better performance of the uncertainty model is the immediate inclusion of every observed state transition in the GP dynamics model.

The same Model Predictive Control (MPC) framework is applied to the dynamics of the DC motor is described by Equation (4.6), with an optimal control problem formulation and constraints identical to those represented in Equation (4.9). The outcomes of this application are presented in Figure 4.9. The settling time of angular velocity is $t_s = 8$ and the settling time of angle is $t_s = 9$. Both states are faster reaching the set point within 9 timesteps. Comparing the plots in Figure 4.5 and Figure 4.9, it becomes evident that the behavior of the GP-MPC closely mirrors that of direct MPC. This alignment validates the effectiveness of the learning process within the GP-MPC framework.

4.2 Van der Pol Oscillator

The Van der Pol oscillator is a non-linear second-order differential equation that describes self-sustained oscillations. It was introduced by Dutch physicist Balthasar van der Pol in 1920 while studying electronic circuits. The equation is commonly used to model various systems exhibiting oscillatory behavior, such as electrical circuits, biological systems, and mechanical systems [42].

The Van der Pol oscillator equation is typically written as:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0 \tag{4.10}$$

Here, x is the displacement of the oscillator from its equilibrium position, t is time, and μ (mu) is a parameter that represents the non-linearity and damping strength of the oscillator. When μ is small, the system behaves like a linear oscillator, but as μ increases, the non-linear effects become more prominent, leading to interesting dynamics such as limit cycles and chaos.

The Van der Pol oscillator exhibits a limit cycle, which means its solutions repeat periodically in phase space. This makes it useful for modeling systems with periodic behavior, such as electronic circuits, where it can represent relaxation oscillators and other types of oscillatory behavior.

The classical Van der Pol oscillator with control input can be described by the following dynamic equations [43]:

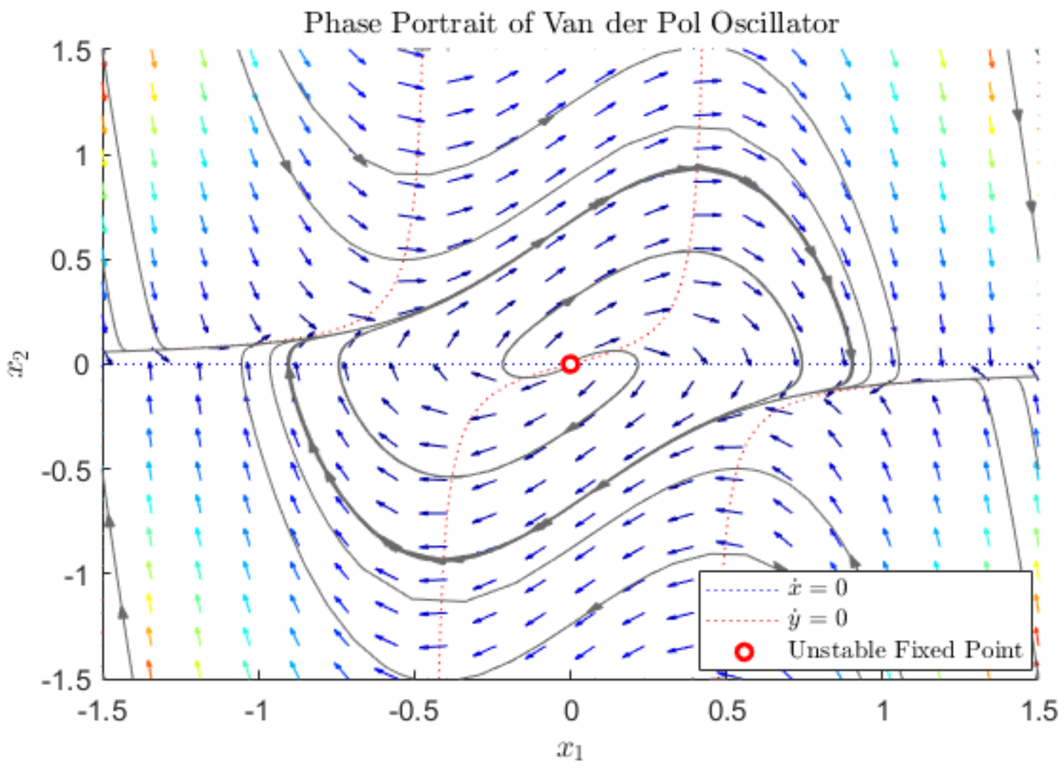


Figure 4.10: Phase portrait of the unforced Van der Pol oscillator

$$
\begin{aligned}
\dot{x}_1 &= 2x_2 \\
\dot{x}_2 &= -0.8x_1 + 2x_2 - 10x_1^2 x_2 + u
\end{aligned}
\tag{4.11}
$$

where x_1 and x_2 represent the state variables of the system, and u is the control input. The parameter μ in the original Van der Pol equation is represented in the damping term $-10x_2(1 - x_2^2)$, indicating the nonlinearity and damping strength. In this system, the sign of the damping term $-10x_2(1 - x_2^2)$ changes based on whether $|x_2|$ is greater or less than unity. This term introduces nonlinearity into the system dynamics[44]. The uncontrolled system, where $u = 0$, has an unstable fixed point at the origin and a stable limit cycle around the origin. This behavior is depicted in Figure 4.10. Nonlinear systems cannot be discretized conventionally such as linear systems. Therefore, the fourth-order Runge-Kutta (RK4) method with a fixed sampling time $T_s = 0.2$ s is utilized for discretization.

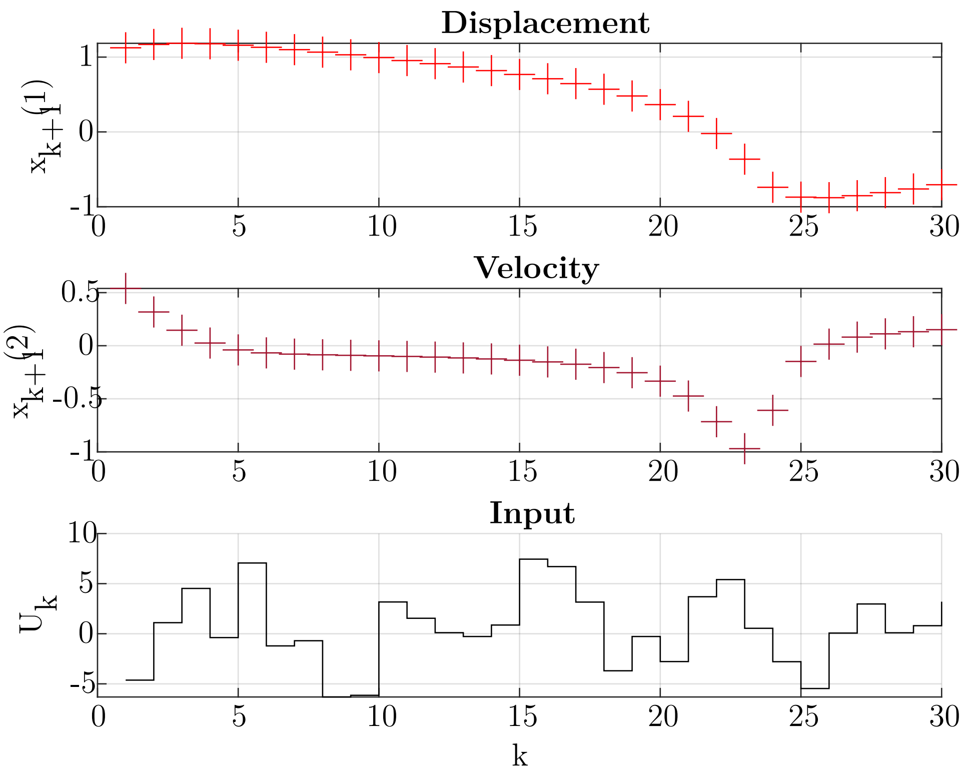


Figure 4.11: Data points,xk+1 and uk for model learning validation for the Van der Pol oscillator

4.2.1 Validation of Van der Pol Oscillator Learning

As explained, the validation of Gaussian Process (GP) and long-term prediction models is crucial for the design of the Model Predictive Controller. The effectiveness of the long-term prediction model hinges on the dependability of the Gaussian Process (GP) model. GP models are instrumental in regression tasks, especially in scenarios involving sparse data, resilient models, or noisy datasets that direct to uncertainty models. Therefore, we will focus on a plant with an uncertainty model of the Van der Pol oscillator.

Hence, uncertainty is introduced to the state equation of the Van der Pol oscillator Equation 4.11 and the modified state equation is,

$$\begin{aligned} \dot{x}_1 &= (2+2\lambda)x_2 \\ \dot{x}_2 &= -(0.8+0.8\alpha)x_1 + (2+2\lambda)x_2 - (10+10\gamma)x_1^2x_2 + u \end{aligned} \tag{4.12}$$

where λ, α and γ, are random distribution in the interval (0,1) that won't change Van der Pol oscillator system properties. The dataset was created by applying a random selection of input u_k to the Van der Pol Oscillator plant state equation described in Equation (4.12). The augmented data X_k and U_k are utilized as input for training the Gaussian Process (GP) model, while X_{k+1} serves as the output. The resulting dataset

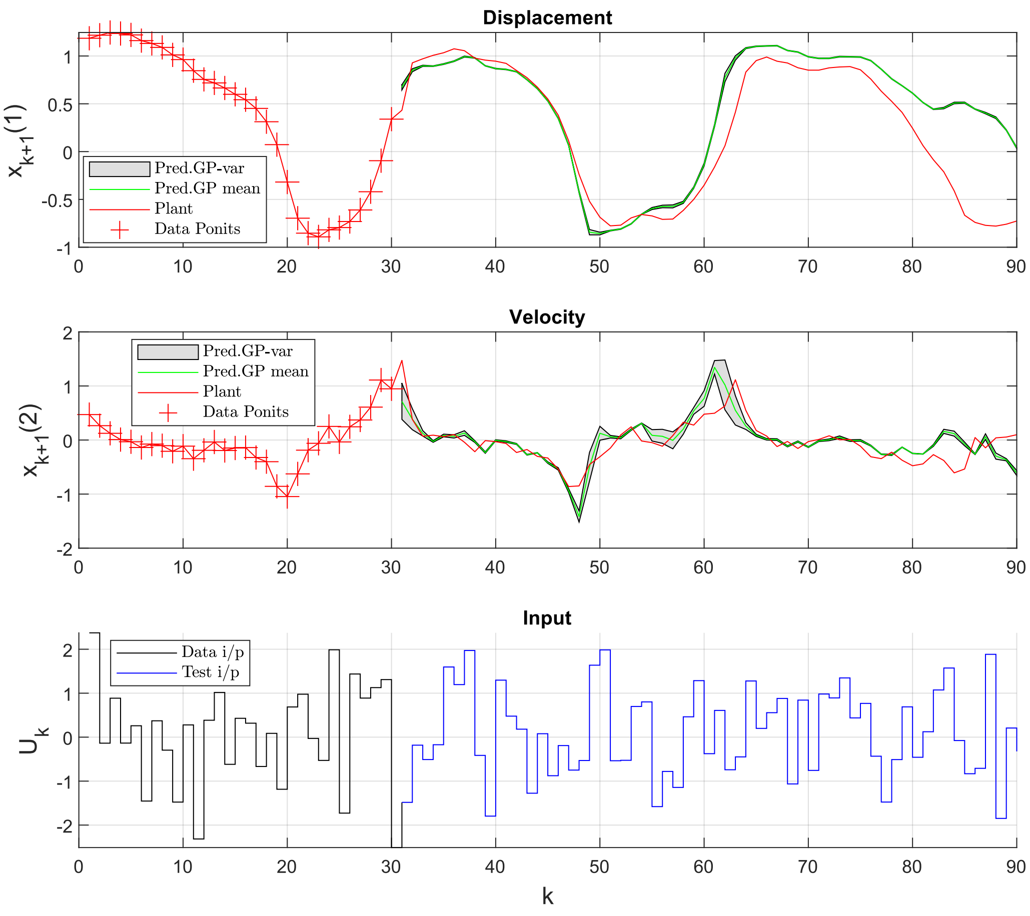


Figure 4.12: Gaussian Process model testing for the Van der Pol oscillator

is visualized in Figure 4.11. With a size of 30, the dataset is relatively small for model training, but it is adequate for validation purposes.

The trained Gaussian Process (GP) model undergoes testing with 60 random inputs $\hat{u_k}$. At every step, the resulting mean is used to predict the next state, and variance is neglected. The resulting predicted mean alongside the control input in Figure 4.12. A comparison is made between the predicted mean and the actual values derived from the Van der Pol Oscillator model described in Equation 4.11. The RMSE for x_1 is found to be 25%, and for x_2 it is 29%. These RMSE values are high compared to the linear model, this is because a number of interactions (training dataset size) is not enough to make accurate predictions for nonlinearity and noisy measurements of the state equation. Because of this, the variance is higher at some point, where the predictions are slightly deviated from actual values. This can be reduced by adding a few more data points to the learning. In summary, the validation process confirms the reliability and accuracy of the GP model, demonstrating its capability to provide trustworthy predictions.

The long-term prediction model undergoes testing with the same set of 60 random inputs

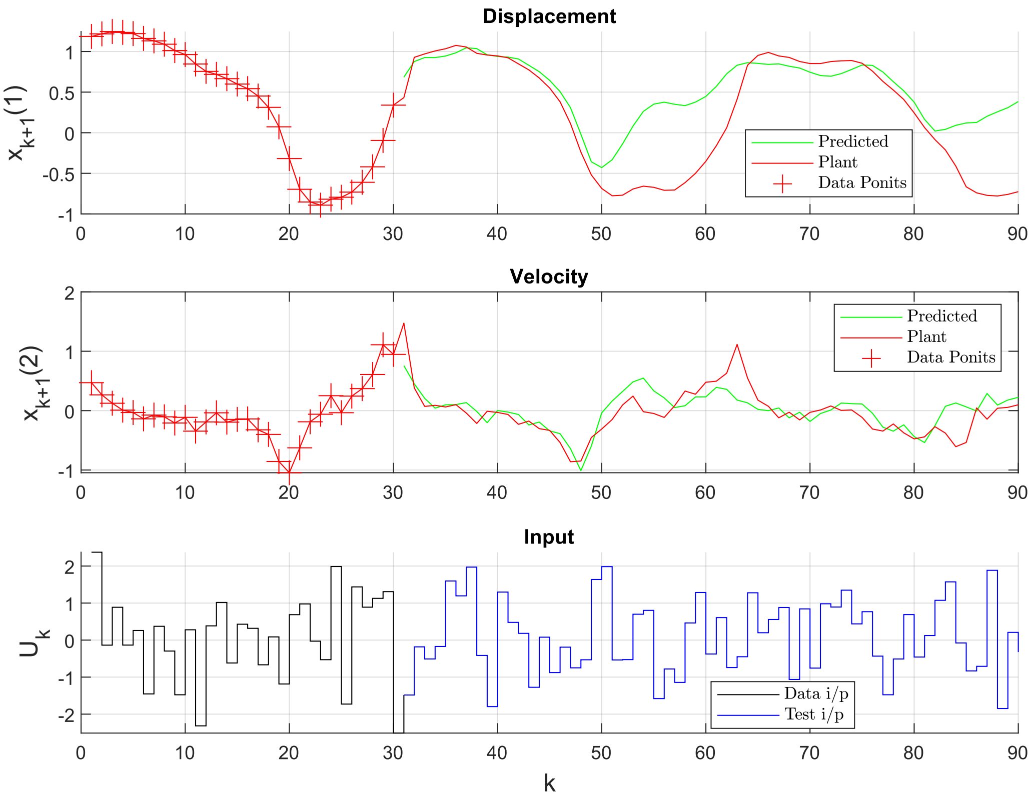


Figure 4.13: Long-term prediction model testing for the Van der Pol oscillator

$\hat{u_k}$. At each time step, we will use both mean and variance for the next prediction. The resulting predicted mean and control inputs are visualized in Figure 4.13. These predictions are then compared with the actual values derived from the uncertainty model described in Equation 4.12. The root mean square error (RMSE) for x_1 and x_2 are found to be 55% and 42% respectively.

The comparison between the Gaussian Process (GP) model and the long-term prediction model highlights notable differences in their performance. While the GP model demonstrates high accuracy with small root mean square errors (RMSE) of 25% for x_1 and 29% for x_2, the long-term prediction model yields higher RMSE values of 55% for x_1 and 42% for x_2. Despite the higher predictive accuracy often demonstrated by GP models, the neglect of previously predicted variance can be a drawback, particularly in scenarios where understanding the reliability and confidence levels of predictions is crucial. Therefore, depending on the specific application and the importance of capturing input uncertainty, long-term prediction models might be preferred over GP models despite their potentially lower predictive accuracy.

Furthermore, ensuring the validation of the long-term prediction model is vital for its integration into Model Predictive Control (MPC) systems. There are major disparities from actual measurements at the end of the prediction horizon, it is due to high variances in the GP model that occurred due to the small number of data points. However, it's

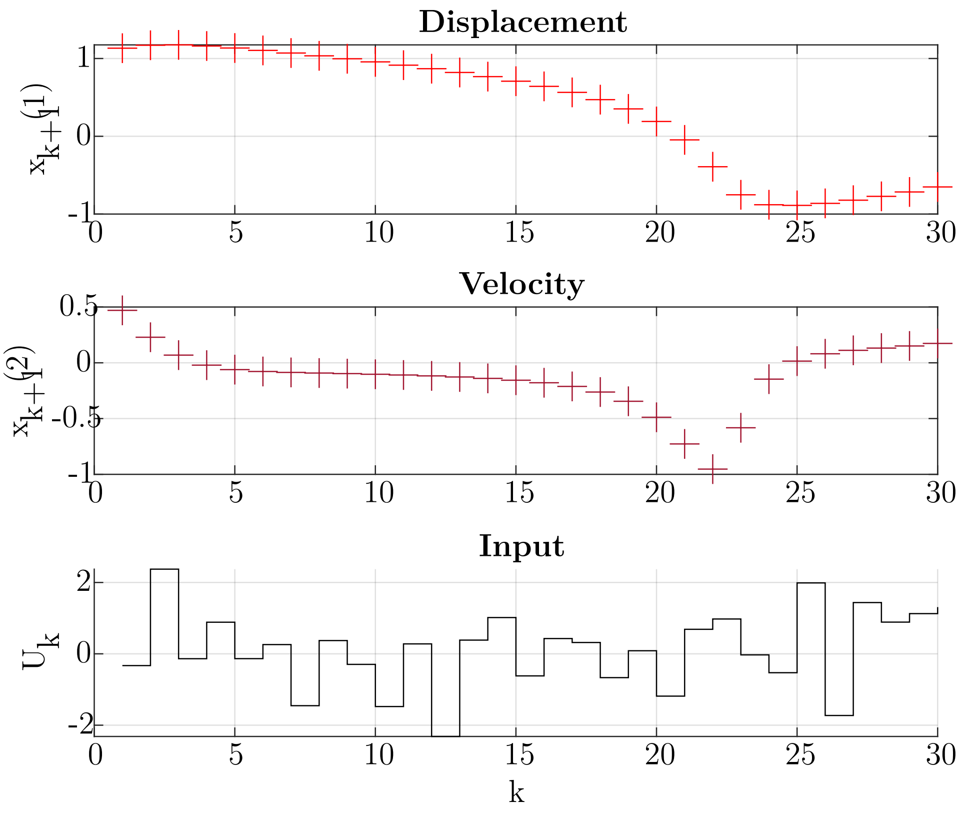


Figure 4.14: Data points of a Van der pol oscillator deterministic plant

important to recognize that as the prediction horizon extends, the impact of small-size data may result in additional disparities between predicted and actual values. With the smaller prediction horizon of up to 20 timesteps, the prediction is accurate with small prediction errors and the Gaussian distributed predictions demonstrate satisfactory accuracy for modeling the MPC controller.

4.2.2 Modelling Without Uncertainty

The Van der Pol oscillator is a classic example of a deterministic nonlinear oscillator, often used to describe various phenomena in physics and engineering. By eliminating stochastic influences, deterministic models provide a precise framework for understanding how the system responds to inputs and evolves over time. This clarity is particularly advantageous in situations where noise is either minimal or can be effectively managed through other means. The deterministic Van der Pol oscillator is discussed in detail in Section 4.2.

The dataset was generated using the standard plant model without any additional noise, as described in Equation (4.11), with a randomly selected set of input values u_k. Such a deterministic model accurately captures the system's behavior under ideal conditions,

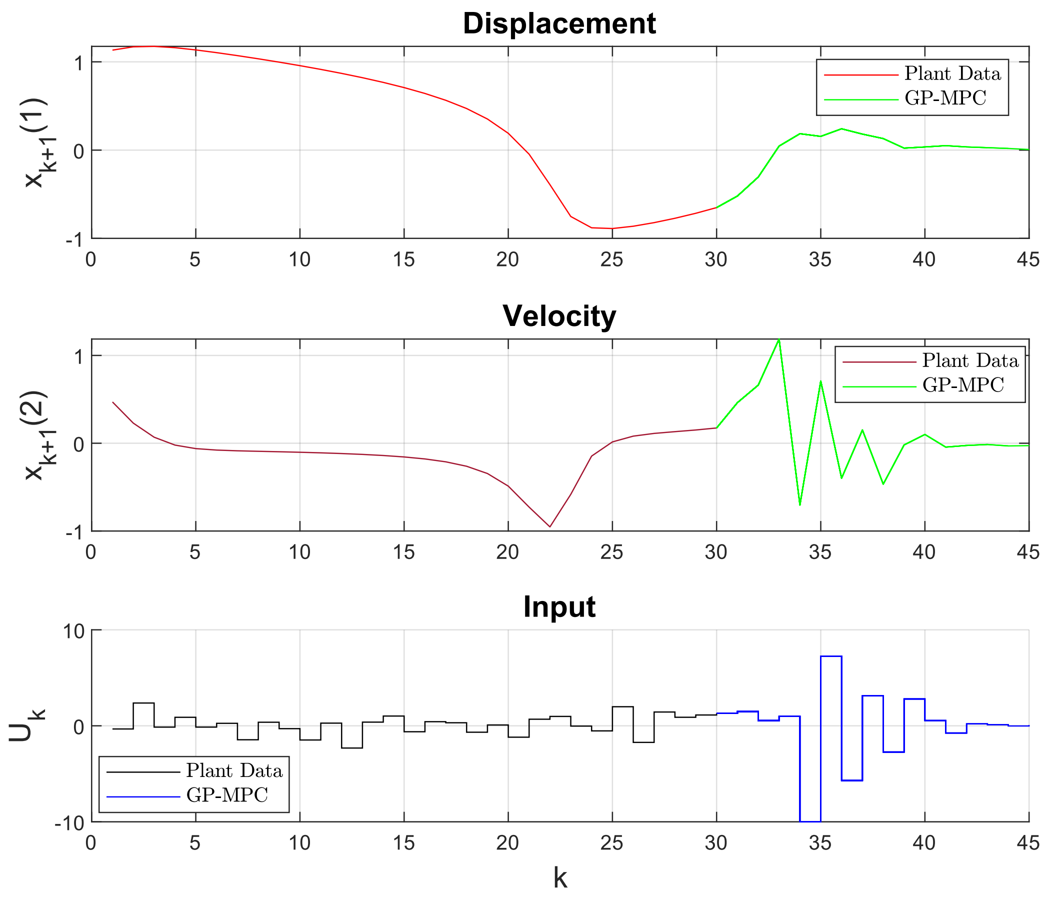


Figure 4.15: GP-MPC controller- Van der Pol oscillator deterministic plant

facilitating analysis and simulation. The resulting dataset, comprising 30 data points, is visually represented in Figure 4.14.

As discussed in Section 3.2.2, in the optimal control problem our objective is to control the states displacement and velocity, aiming to take the states to zero as fast as possible with minimal control input. The optimal control problem is defined over a prediction horizon of 15 steps. The initial states and control input are given as the end point of the dataset, and the control input constraint is defined as $-10 \leq u_k \leq 10$. The weighting matrices Q and R are adjusted to ensure that the controller operates quickly with minimal input.

The implementation of Gaussian Process Model Predictive Control (GP-MPC) to the noise-free Van der Pol oscillator, as illustrated in Figure 4.15. The settling time of displacement is $t_s = 13$ and the settling time of velocity is $t_s = 12$. Both states are faster reaching the set point within 13 timesteps. Through iterative optimization, the GP-MPC controller leverages the noise-free system dynamics to achieve control performance, even in the absence of stochastic disturbances.

The same MPC framework is applied to the dynamics of the Van der pol oscillator described

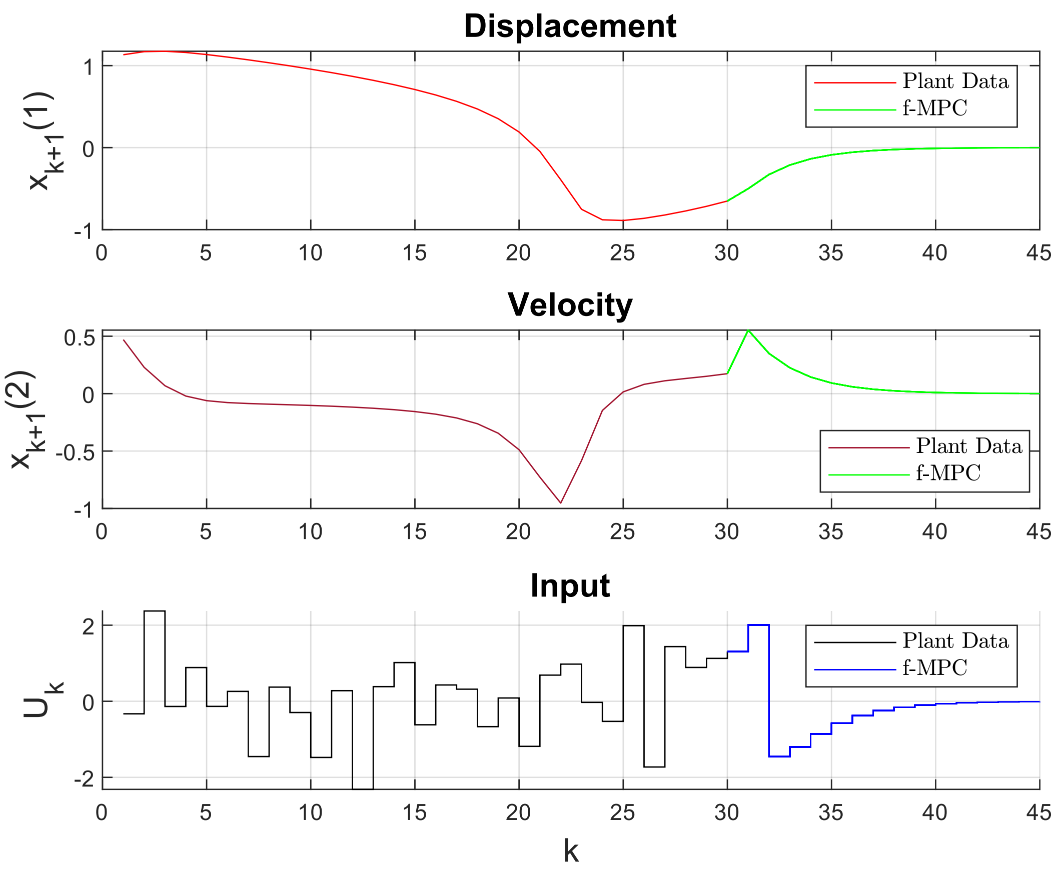


Figure 4.16: f-MPC controller- Van der Pol oscillator deterministic plant

by Equation (4.11), with identical optimal control problem formulations and constraints. These results are depicted in Figure 4.16. The settling time of displacement is $t_s = 9$ and the settling time of velocity is $t_s = 10$. Both states are faster reaching the set point within 10 timesteps, which is faster compared to the GP-MPC controller this is due to modeling errors that occurred during GP training and long-term prediction model learning.

Direct MPC results are compared against GP-MPC results to ensure the accuracy of the learning process of GP-MPC framework. The comparison between the plots in Figure 4.15 and Figure 4.16 reveals that the behavior of the GP-MPC struggles to control in the beginning stage of the control horizon which takes more control effort to control the states, this is due to prediction errors in the long-term prediction model.

4.2.3 Modelling With Uncertainty

In this section, we explore the application of the Gaussian Process Model Predictive Control (GP-MPC) to the Van der Pol oscillator in the presence of uncertainty. In the context of the Van der Pol oscillator, uncertainties can manifest as variations in system

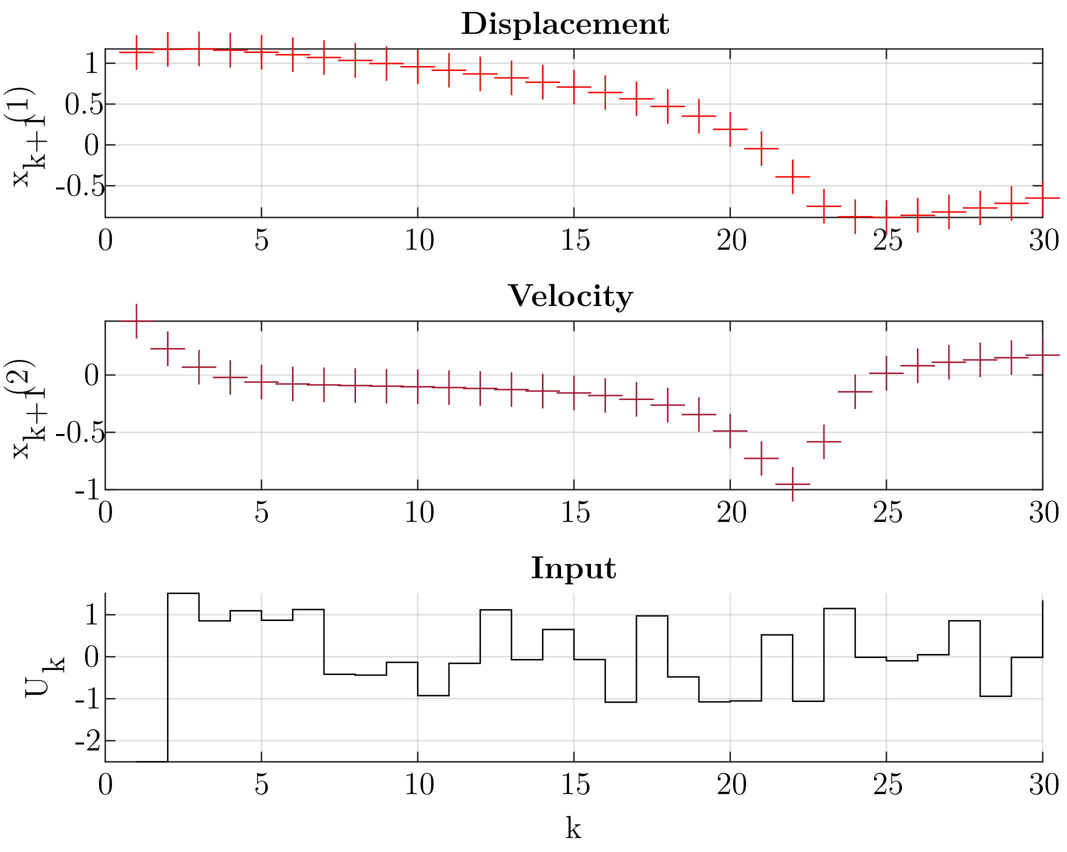


Figure 4.17: Data points of an uncertainty Van der pol oscillator plant

parameters or disturbances affecting the dynamics. To capture these uncertainties, we augment the system equations with stochastic terms, as shown in Equation (4.12).

To train the GP model under uncertainty, we generate a dataset by applying a random selection of control inputs (u_k) to the uncertain Van der Pol oscillator described by the modified state equation (4.12). The resulting state transitions (x_{k+1}), current states (x_k), and control inputs (u_k) constitute the training data for the GP model. Figure 4.17 illustrates the data points used for model learning validation.

Here, the optimal control problem is similar to the OCP of modeling without uncertainty, but here we take the uncertainty-trained model, aiming to take the states to zero as fast as possible with a minimal control input. Figure 4.18 illustrates the learning process of the GP-MPC controller applied to the uncertainty model of the Van der Pol Oscillator. The settling time of displacement is $t_s = 13$ and the settling time of velocity is $t_s = 14$. Both states are faster reaching the set point within 14 timesteps.

Through iterative optimization, the GP-MPC controller leverages the uncertainty system dynamics to achieve control performance, even in the presence of stochastic disturbances. One big reason why the uncertainty model works better is because we quickly include every change in the system's state into our model. This helps the model adapt and learn

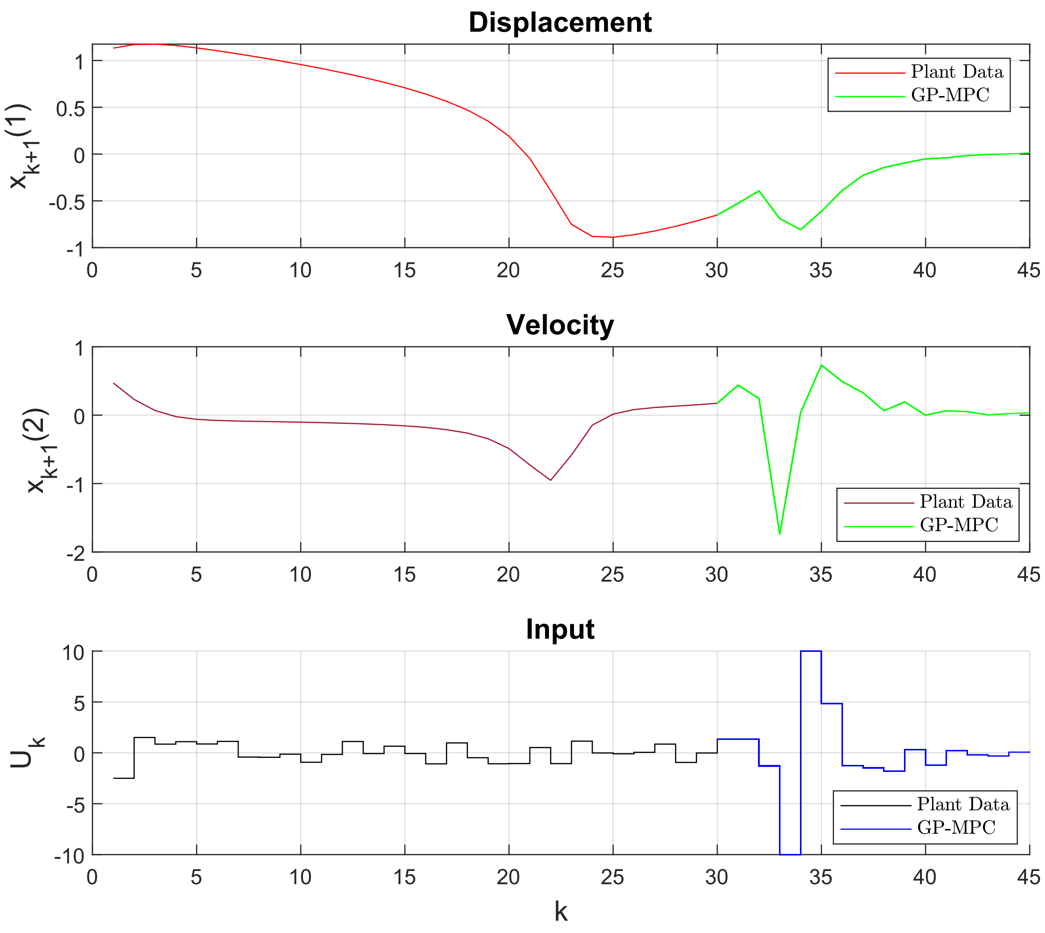


Figure 4.18: GP-MPC controller- Van der Pol oscillator plant with parameter uncertainty

from uncertainty as things change, making the controller better at handling mistakes in the model.

We use the same MPC directly on the Van der pol oscillator dynamics described by Equation (4.12). The results are shown in Figure 4.19. The time it takes for the displacement and velocity to settle is 9 timesteps $t_s = 9$. Both states reach their target faster, within 9 timesteps, better compared to the GP-MPC controller.

We compare the results of Direct MPC with GP-MPC to make sure GP-MPC is learning correctly. When we look at the plots in Figure 4.18 and Figure 4.19, we notice that GP-MPC struggles to control the system at the beginning of the control horizon. This increased control effort to regulate the states is primarily due to prediction errors in the long-term prediction model.

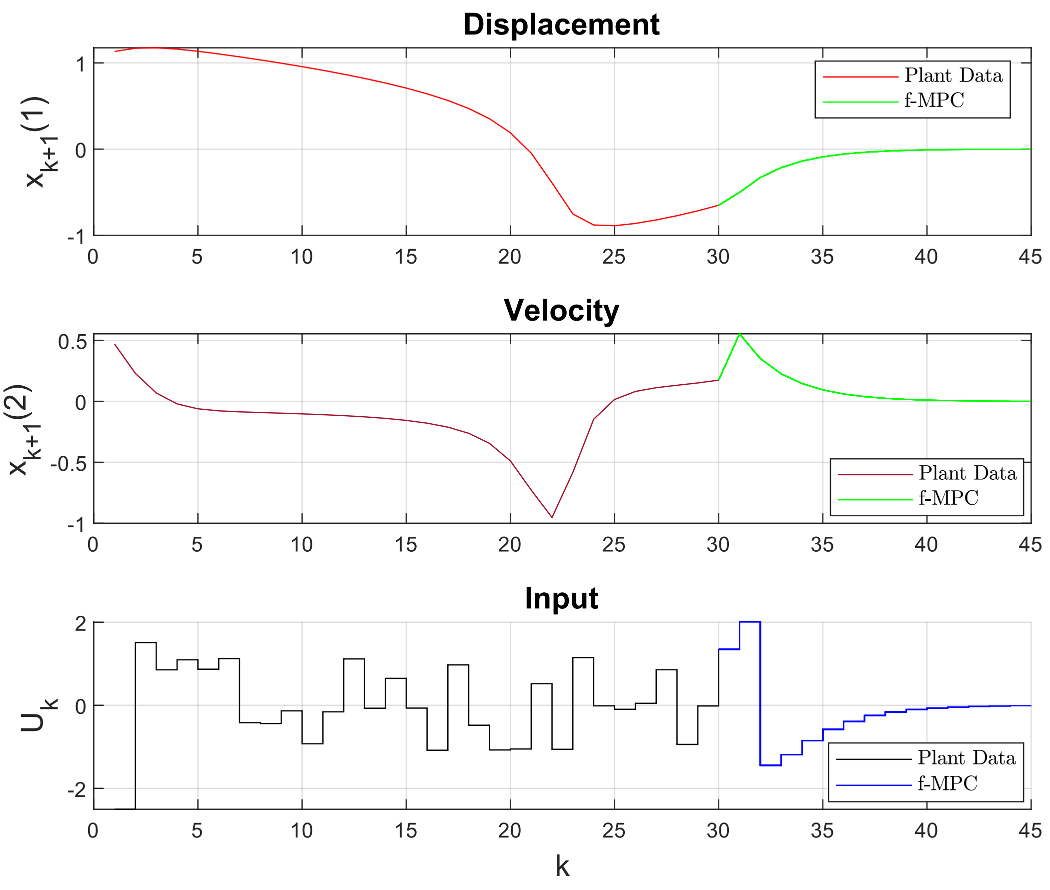


Figure 4.19: f-MPC controller- Van der Pol oscillator plant with parameter uncertainty

4.2.4 Reference Tracking of Van der Pol Oscillator

Reference tracking in control systems refers to the ability of the system to accurately reach the target setpoint over time. By adjusting the control inputs, the system's behavior is regulated to closely match the specified reference target point, enabling it to achieve desired performance objectives. In the context of the Van der pol oscillator uncertainty dynamics discussed earlier, implementing reference tracking involves setting new target values for displacement and velocity, and then utilizing Model Predictive Control (MPC) to ensure that the states closely follow these targets. The Reference tracking capability is crucial for applications where precise control and adherence to specified trajectories are essential, such as robotics, autonomous vehicles, and industrial automation.

In the preceding section, we observed the stabilization of the system from a point outside the stable limit cycle to the unstable equilibrium at the origin, using an initial point $x_0 = [1.0, 1.0]$ and a reference point $x_f = [0, 0]$.
In this section, we examine two kinds of reference tracking :

- The transition from a point outside the stable limit cycle to a stabilizable point

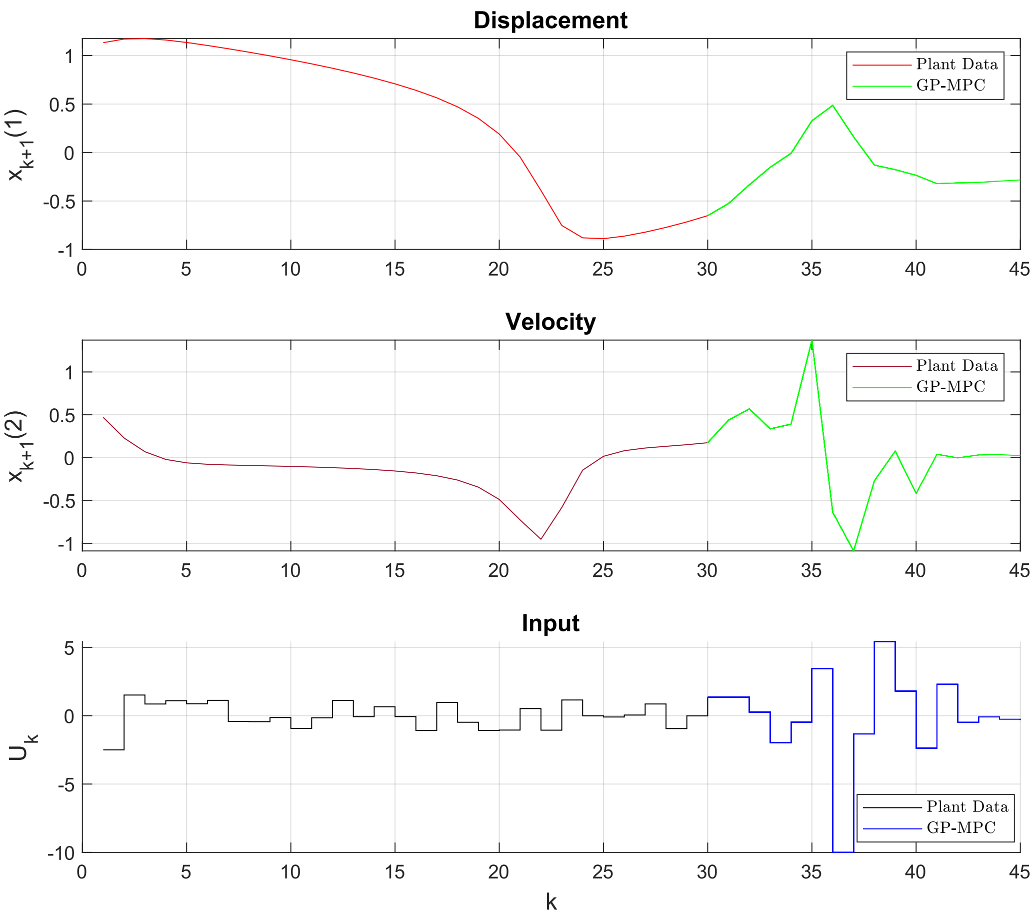


Figure 4.20: GP-MPC controller- Van der Pol oscillator plant with parameter uncertainty, set point $x_f = [-0.25, 0]$

inside the stable limit cycle, where the initial point remains $x_0 = [1.0, 1.0]$ but the reference point is adjusted to $x_f = [-0.25, 0]$.

- The transition from a point outside the stable limit cycle to a stabilizable point near the stable limit cycle, where the reference point is adjusted to $x_f = [1, 0]$.

The cost function is modified as suitable to reference tracking as follows,

$$J(k) = \sum_{k=i}^{i+N-1} ((x_{k+1} - x_f)^T Q(x_{k+1} - x_f) + u_k^T R u_k) \tag{4.13}$$

-Setpoint $x_f = [-0.25, 0]$

The GP-MPC framework is applied to the Van der Pol oscillator for a new setpoint $x_f = [-0.25, 0]$, utilizing a modified cost function. The results are shown in Figure 4.20. The settling time for displacement is $t_s = 12$, and for velocity, it's $t_s = 13$. Both states

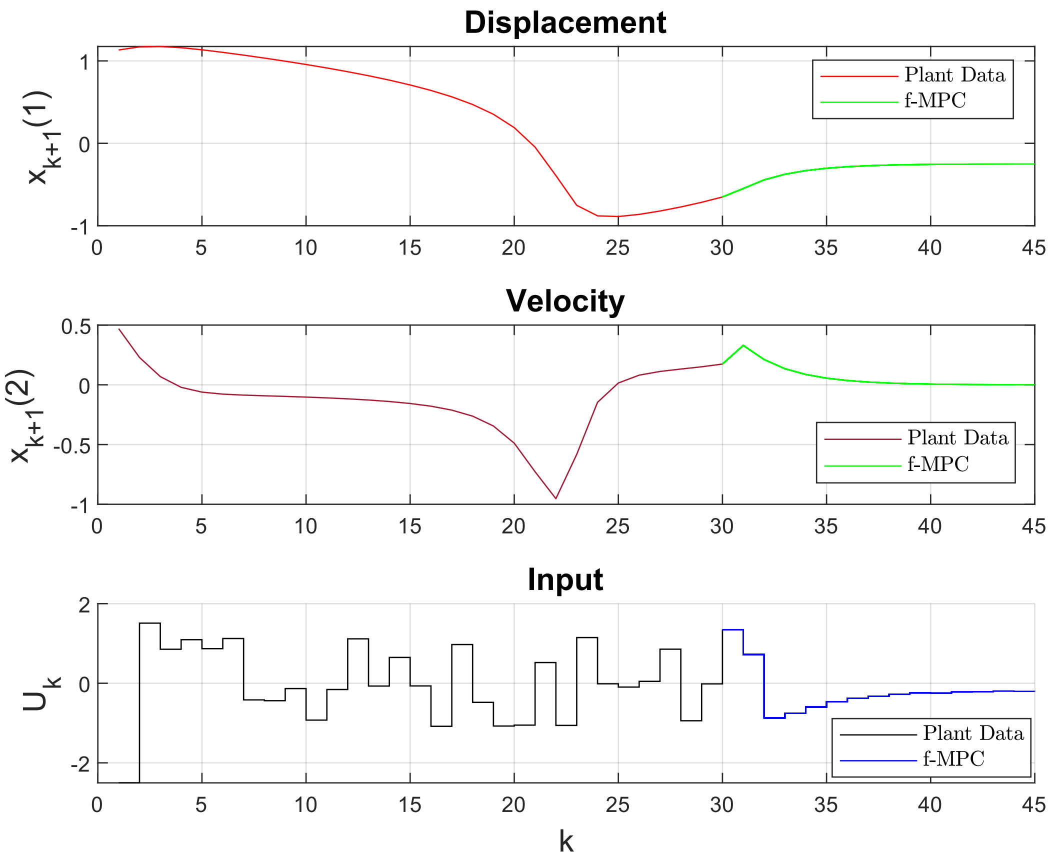


Figure 4.21: f-MPC controller- Van der Pol oscillator plant with parameter uncertainty, set point $x_f = [-0.25, 0]$

reach the setpoint faster within 13 timesteps. However, GP-MPC encounters challenges in controlling the system during the initial stages of the control horizon, requiring more control effort to regulate the states.

Similarly, we employ the Model Predictive Control (MPC) framework with a modified cost function on the dynamics described by Equation (4.12), maintaining identical optimal control problem formulations and constraints. These results are depicted in Figure 4.21. Direct comparison between f-MPC and GP-MPC results ensures the accuracy of the learning process within the GP-MPC framework.

-Setpoint $x_f = [1, 0]$

Similarly to a new setpoint $x_f = [1, 0]$, the GP-MPC framework is applied utilizing a modified cost function over 20 timesteps prediction horizon. The results are shown in Figure 4.22. The settling time for displacement is $t_s = 7$, and for velocity, it's $t_s = 14$. Both states reach the setpoint faster within 14 timesteps. However, GP-MPC encounters challenges in controlling the velocity during the initial stages of the control horizon, requiring more control effort to regulate the states. We employ the Model Predictive Control (MPC)

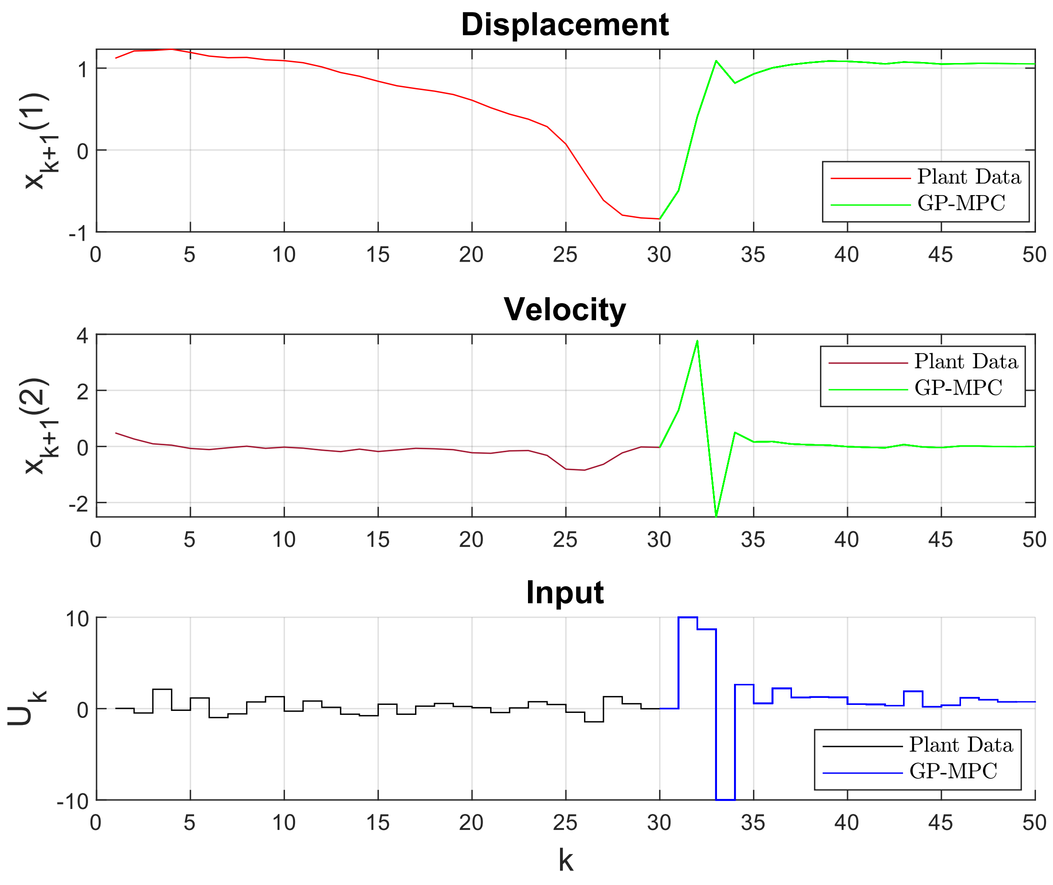


Figure 4.22: GP-MPC controller- Van der Pol oscillator plant with parameter uncertainty, set point $x_f = [1, 0]$

framework to a new setpoint $x_f = [1, 0]$ on the dynamics described by Equation (4.12), maintaining identical optimal control problem formulations and constraints. The settling time for displacement is $t_s = 12$, and for velocity, it's $t_s = 12$. These results are depicted in Figure 4.23. Direct comparison between f-MPC and GP-MPC results ensures the accuracy of the learning process within the GP-MPC framework. The main difference between the GP-MPC and direct MPC is that GP-MPC requires more control effort compared to direct MPC and hence it requires more time to reach the reference point.

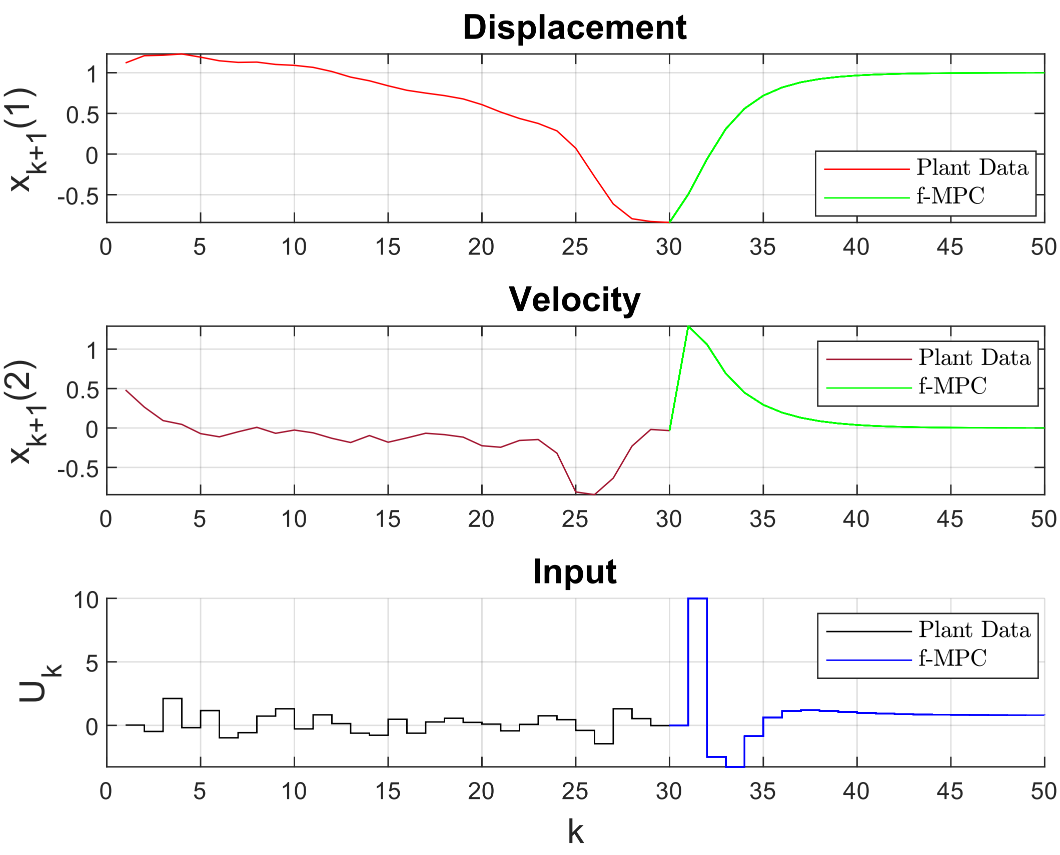


Figure 4.23: f-MPC controller- Van der Pol oscillator plant with parameter uncertainty, set point $x_f = [1, 0]$

5 Conclusion and Outlook

5.1 Conclusion

In this thesis, we introduced a probabilistic Model Predictive Control (MPC) approach utilizing Gaussian process models to handle the challenges posed by noisy and uncertain system dynamics. Through comprehensive simulations on both linear (DC motor) and nonlinear (Van der Pol oscillator) systems, we validated the efficacy of the proposed GP-MPC framework under various conditions.

In the case of the DC motor, the validation process of the Gaussian process (GP) model and long-term prediction model showcased their reliability and accuracy in capturing system dynamics, even in the presence of noise and uncertainty. By achieving small root mean square error (RMSE) values and average variances, these models demonstrated their capability to make trustworthy predictions in real-world scenarios. When applied to the noise-free DC motor system, GP-MPC exhibited precise control performance akin to direct MPC, highlighting the effectiveness of deterministic modeling in control design. Furthermore, when subjected to uncertainty and noisy measurements, the GP-MPC controller adapted quickly and showcased robust performance comparable to the noise-free scenario. This adaptability was attributed to the GP model's ability to incorporate observed state transitions, enabling the controller to learn and adapt to uncertainty over time.

Similarly, the comprehensive examination of the Van der Pol oscillator within the context of Gaussian Process Model Predictive Control (GP-MPC) has provided valuable insights into its efficacy across various scenarios, including both deterministic and uncertain environments. The validation of the GP model and its derivative long-term prediction model underscores their reliability and accuracy in capturing system dynamics, even in the presence of nonlinearity and noisy measurements. While deviations between predicted and actual values may occur, particularly in uncertain scenarios, the overall performance of the GP-MPC framework remains commendable, because of the small prediction horizon which has fewer errors and satisfactory accuracy for modeling the system dynamics.

In both scenarios without uncertainty and with uncertainty, the GP-MPC framework struggles to control at the beginning of the horizon but overall it demonstrates faster control performance and achieves desired setpoints within reasonable settling times(8-14 timesteps). However, in uncertain environments, where model errors and stochastic disturbances are present, the GP-MPC framework exhibits resilience, adapting to uncertainty and achieving satisfactory control performance despite initial challenges in the control horizon. The exploration of reference tracking capabilities within the GP-MPC framework for the Van der Pol oscillator emphasizes its importance in attaining accurate control and ensuring compliance with predefined target setpoints. By adjusting control inputs to regulate system behavior towards target setpoints, GP-MPC demonstrates the ability to facilitate transitions from points outside stable limit cycles to stabilizable points, both inside and near the stable limit cycle. Through simulation results, we observe that while GP-MPC achieves faster settling times for displacement and velocity towards the reference points, it encounters initial challenges in controlling the system during the early stages of the control horizon, necessitating more control effort compared to direct MPC.

In our approach, the incorporation of model uncertainty into both modeling and planning stands out as a pivotal aspect, particularly in navigating complex environments. By acknowledging and addressing the inherent uncertainty in system dynamics, our method facilitates targeted exploration, allowing the system to gather essential information and refine its understanding over time. Moreover, this consideration enables us to approach constraints in a risk-averse manner, crucially important, especially in the early stages of learning, where overly optimistic actions could lead to undesired outcomes. By embracing model uncertainty, our approach not only enhances adaptability and robustness but also fosters safer and more informed decision-making in practical applications.

The key to our GP-MPC algorithm lies in redefining the optimal control problem to incorporate uncertainty propagation through moment matching, thereby transforming it into a deterministic optimal control problem. By integrating Model Predictive Control (MPC), our approach enables immediate updates to the learned model, enhancing robustness against model inaccuracies. Through empirical validation via simulation results, we have demonstrated that our framework not only excels as an efficient controller but also exhibits remarkable data efficiency, particularly in learning environments characterized by uncertainties. This achievement places our approach at the forefront of data-efficient control methodologies.

5.2 Future Research

Building upon the insights gained from this project work, future research endeavors can explore several promising avenues to further enhance the capabilities and applicability of the proposed GP-MPC framework. One potential direction for advancement lies in optimizing the long-term prediction model to enhance its speed and accuracy. By leveraging advanced modeling techniques and algorithmic improvements, researchers can work towards developing a more efficient and reliable long-term prediction model, capable of providing faster and more accurate predictions in real-time scenarios.

Additionally, exploring the application of GP-MPC in Multi-Input Multi-Output (MIMO) systems opens up new avenues for research. By adapting the framework to handle multiple inputs and outputs simultaneously, researchers can address complex control challenges in interconnected systems, paving the way for advanced control strategies in diverse domains ranging from robotics to industrial automation.

Furthermore, extending the application of the GP-MPC framework to real-world systems both linear and nonlinear systems, presents an exciting opportunity for future investigation. Conducting experiments and validations on physical systems can provide invaluable insights into the practical feasibility and performance of the proposed approach in real-time control applications.

References

[1] P. Panizza, F. Riccardi, and M. Lovera, “Black-box and grey-box identification of the attitude dynamics for a variable-pitch quadrotor”, in *1st IFAC Workshop on Advanced Control and Navigation for Autonomous Aerospace Vehicles (ACNAAV)*, 2015.

[2] P. Dev, S. Jain, P. Kumar Arora, and H. Kumar, “Machine learning and its impact on control systems: A review”, *Materials Today: Proceedings*, vol. 47, pp. 3744–3749, 2021, 3rd International Conference on Computational and Experimental Methods in Mechanical Engineering, ISSN: 2214-7853. DOI: `https://doi.org/10.1016/j.matpr.2021.02.281`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2214785321013808`.

[3] D. Silver, A. Huang, C. J. Maddison, *et al.*, “Mastering the game of go with deep neural networks and tree search”, *Nature*, vol. 529, no. 7587, 2016.

[4] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, “Collective robot reinforcement learning with distributed asynchronous guided policy search”, *arXiv preprint arXiv:1610.00673*, 2016.

[5] J. G. Schneider, “Exploiting model uncertainty estimates for safe dynamic control learning”, in *Advances in Neural Information Processing Systems*, 1997.

[6] M. P. Deisenroth and C. E. Rasmussen, “Pilco: A model-based and data-efficient approach to policy search”, in *Proceedings of the International Conference on Machine Learning*, 2011.

[7] S. Kamthe and M. P. Deisenroth, *Data-efficient reinforcement learning with probabilistic model predictive control*, 2018.

[8] 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.

[9] J. Berberich, J. Kohler, M. A. Muller, and F. Allgower, “Data-driven model predictive control with stability and robustness guarantees”, *IEEE Transactions on Automatic Control*, 2021.

[10] N. Lawrence, “Probabilistic non-linear principal component analysis with gaussian process latent variable models”, *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[11] J. Quiñonero-Candela, C. E. Rasmussen, and C. K. I. Williams, “Approximation methods for gaussian process regression”, *Applied Games, Microsoft Research Ltd.*, May 2007.

[12] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning.* Cambridge, MA: MIT Press, 2006, vol. 2.

[13] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, "Gaussian processes and kernel methods: A review on connections and equivalences", 2018. DOI: 10.48550/ARXIV.1807.02582. [Online]. Available: https://arxiv.org/abs/1807.02582.

[14] L. Hewing and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression", *CoRR*, vol. abs/1705.10702, 2017. [Online]. Available: http://arxiv.org/abs/1705.10702.

[15] A. Girard, C. E. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs: Application to multiple-step ahead time series forecasting", in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds., MIT Press, 2003, pp. 545–552.

[16] A. Girard, C. E. Rasmussen, and R. Murray-Smith, "Gaussian process priors with uncertain inputs: Multiple-step-ahead prediction", Department of Computing Science, University of Glasgow, Tech. Rep., 2002.

[17] M. Deisenroth, D. Fox, and C. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 408–423, Feb. 2015. DOI: 10.1109/TPAMI.2013.218.

[18] J. Quiñonero-Candela, A. Girard, J. Larsen, and C. E. Rasmussen, "Propagation of uncertainty in bayesian kernel models—application to multiple-step ahead forecasting", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003.

[19] R. Rezvani Arany, "Gaussian process model predictive control for autonomous driving in safety-critical scenarios", PhD Thesis, Dissertation, 2019.

[20] J. O. Lübsen, "Bayesian optimization for the control parameters of the optical synchronization system at european xfel", M.S. thesis, Hamburg University of Technology, Jul. 2022.

[21] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design.* Nob Hill Publishing, Madison, 2009.

[22] J. B. Rawlings and B. R. Bakshi, "Particle filtering and moving horizon estimation", *Computers & Chemical Engineering*, vol. 30, no. 10-12, pp. 1529–1541, 2006.

[23] R. Verschueren, G. Frison, D. Kouzoupis, *et al.*, "Acados: A modular open-source framework for fast embedded optimal control", *arXiv preprint arXiv:1910.13753*, 2019.

[24] P. Zometa, M. Kogel, T. Faulwasser, and R. Findeisen, "Implementation aspects of model predictive control for embedded systems", in *American Control Conference (ACC)*, IEEE, 2012, pp. 1205–1210.

[25] E. Hernandez Vargas, P. Colaneri, and R. Middleton, "Switching strategies to mitigate hiv mutation", *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1623–1628, 2014.

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

[27] M. Maiworm, “Gaussian processes in control: Model predictive control with guarantees and control of scanning quantum dot microscopy”, Ph.D. Dissertation, Otto von Guericke University Magdeburg, Faculty of Electrical Engineering and Information Technology, Magdeburg, Germany, Jun. 2021. [Online]. Available: `https://doi.org/10.25673/38665`.

[28] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, “On the stability of constrained mpc without terminal constraint”, *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 832–836, 2006.

[29] D. Limón, T. Alamo, D. Raimondo, *et al.*, “Input-to-state stability: A unifying framework for robust model predictive control”, in *Nonlinear Model Predictive Control*, Springer, 2009, pp. 1–26.

[30] D. G. Luenberger, *Optimization by Vector Space Methods.* Wiley, New York, 1969.

[31] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings, “On the inherent robustness of optimal and suboptimal nonlinear mpc”, *Systems & Control Letters*, vol. 106, pp. 68–78, 2017.

[32] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel, “Nominally robust model predictive control with state constraints”, *IEEE Transactions on Automatic Control*, vol. 52, no. 10, pp. 1856–1870, 2007.

[33] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control.* Boston: Birkhäuser, 2008.

[34] J. O. Lübsen, M. Schütte, S. Schulz, and A. Eichler, “A safe bayesian optimization algorithm for tuning the optical synchronization system at european xfel”, *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3079–3085, 2023, 22nd IFAC World Congress, ISSN: 2405-8963. DOI: `https://doi.org/10.1016/j.ifacol.2023.10.1438`.

[35] L. Chevrel, L. Sicot, and S. Siala, “Switched lq controllers for dc motor speed and current control: A comparison with cascade control”, in *Proc. Power Electronics Specialists Conference PESC’96 Record.*, Baveno, Italy, Jun. 1996, pp. 906–912.

[36] H. Umeno and Y. Hori, “Robust speed control of dc servomotors using modern two degrees-of-freedom controller design”, *IEEE Transactions on Industrial Electronics*, vol. 38, pp. 363–368, 1991.

[37] S. K. Suman and V. K. Giri, “Speed control of dc motor using optimization techniques based pid controller”, in *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, IEEE, Coimbatore, India, 2016, pp. 581–587. DOI: `10.1109/ICETECH.2016.7569318`.

[38] V. Puig Cayuela and D. Ramasubramanian, “Identification and control of DC motors”, M.S. thesis, Universitat Politècnica de Catalunya, Sep. 2016. [Online]. Available: `http://hdl.handle.net/2117/98735`.

[39] H. Werner, *Introduction to control systems: Lecture notes*, Lecture notes, Hamburg, Germany, Jan. 2023.

[40] Q. Truong, “Continuous-time model predictive control”, M.S. thesis, School of Electrical and Computer Engineering, RMIT University, Melbourne, Australia, Mar. 2007.

[41] M. Dulău, M. Abrudean, A.-V. Duka, and S.-E. Oltean, “The DC motor as a system affected by parameter uncertainties”, *Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.*, vol. 61, no. 2, pp. 131–136, 2016.

[42] J. Guckenheimer, “Dynamics of the van der pol equation”, *Circuits and Systems, IEEE Transactions on*, vol. 27, pp. 983–989, Dec. 1980. DOI: `10.1109/TCS.1980.1084738`.

[43] M. Korda and I. Mezić, *Optimal construction of koopman eigenfunctions for prediction and control*, 2020. arXiv: `1810.08733 [math.OC]`.

[44] M. Girotti, *The van der pol oscillator*, Lecture Notes. [Online]. Available: `https://mathemanu.github.io/VanderPol.pdf`.