Improving Performance of Position-Dependent Mechatronic Systems Using Gaussian Processes

van Haren, Max

Award date:
2021

Master Systems and Control
Department of Mechanical Engineering
Control Systems Technology Research Group

ASM Center of Competency
Beuningen

Improving Performance of Position-Dependent Mechatronic Systems Using Gaussian Processes

Max (M.J.) van Haren
0953564

CST2021.007

Supervisors	dr. ir. Tom (T.A.E.) Oomen	TU/e
	dr. Jim (J.W.) Portegies	TU/e
	ir. Maurice (M.M.) Poot	TU/e
	dr. ir. Dragan Kostić	ASMPT
	ir. Robin van Es	ASMPT
External committee member	prof. dr. ir. Marcel (M.F.) Heertjes	TU/e

05 Apr, 2021

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

05-04-2021

Name

Max van Haren

ID-number

0953564

Signature

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Improving Performance of Position-Dependent Mechatronic Systems Using Gaussian Processes

Max (M.J.) van Haren

***Abstract*—Due to ever increasing performance requirements for motion control, position-dependent dynamics cannot be neglected anymore and should be accounted for in feedforward control. The aim of this work is to create a framework which models position-dependent feedforward parameters using a Gaussian process. Mutual information is employed to optimize the training positions of the Gaussian process. The feedforward parameters are learned in a trial-to-trial fashion, which is either iterative learning control with basis functions or instrumental variable based feedforward control. The framework is both validated in a computer simulation and experimental setting on a commercial wirebonder, showing the advantage of the framework.**

I. Introduction

Motion controller performance demands are constantly increasing and therefore advanced FeedForward (FF) controllers are necessary. Positioning accuracy, throughput and reliability should be constantly increased and therefore many factors which were previously not taken into account, for instance position-dependency, are getting more relevant.

The motion control demands are especially apparent in the semiconductor back-end industry[1]. Firstly, semiconductor manufacturing equipment needs to have micrometer positioning accuracy due to the decreasing size of semiconductor devices. Secondly, high throughput is demanded such that as many devices can be handled in as little as time as possible. This results in high requirements for both the velocity and acceleration of the machine. Lastly, reliability is expected in all machines. In other words, the positioning performance of semiconductor back-end machines need to perform uniformly, regardless of industrial environment or machine-to-machine differences.

Learning control is a suitable choice to achieve high accuracy and throughput. Furthermore, it performs uniformly regardless of industrial environment or machine-to-machine differences and can keep costs down by eliminating manual tuning and complex system identification [1].

Iterative Learning Control (ILC) can significantly increase the control performance in a trial-to-trial fashion by learning. ILC utilizes the measured error and command signal from the previous trial to decrease the tracking error up to the reproducible part. ILC uses a constant reference trajectory [2]. Possible applications of ILC can range from printing systems [3] to semiconductor back-end machines [4]. Although ILC is proven to have excellent tracking performance, the FF signal is only suitable for one specific reference, hence extrapolation to other references deteriorates tracking performance. Basis Functions (BF) are adopted to the ILC scheme (ILCBF) in order to improve extrapolation capabilities to other references [5, 6]. Here, the FF signal is parameterized using BF, which are a function of the reference signal. Many BFs are possible, whereas two examples are polynomial [5–7] or rational [8–10]. However, in [11] there is shown that the FF parameter estimate from ILCBF can have a bias error.

Instrumental Variable (IV) based FF control [11, 12] learns the FF parameters in a trial-to-trial fashion and results in an unbiased FF parameter estimate. Moreover, it utilizes a similar FF parameterization as ILCBF and therefore has extrapolation capabilities. Both ILCBF and IV FF control however, do not take into account that the dynamics of the machine can be position-dependent.

Several options for handling systems with position-dependent dynamics are possible. First, the position-dependent dynamics can be ignored, by determining FF parameters in only one position. This set of FF parameters is then used throughout the entire operating range of the machine. This can show performance degradation when the machine moves away from the position where the parameters are determined [13]. Second, the FF parameters can be determined in a grid covering the operating range of the machine. The FF parameters can then be estimated using a nearest-neighbour search [14, 15] or interpolation [16, 17]. An accurate representation of the FF parameters can be achieved by choosing a fine grid, however, this would require many training positions and is therefore time consuming. A coarse grid would reduce the amount of training positions, but it is challenging to pick a coarse grid such that the position-dependency is captured accurately. Furthermore, the nearest-neighbour search will result in a piecewise constant representation of the FF parameters, which is unlikely to be the true set of FF parameters. Moreover, the interpolation requires a predetermined relation between the FF parameters and the position, which might be incorrect or unknown. Third, Linear Parameter Varying (LPV) ILC [18] can be applied to position-dependent systems, improving performance compared with Linear Time Invariant (LTI) ILC applied to position-dependent systems. On the other hand, LPV ILC requires modelling or identification of the LPV system, which is often not desirable.

The proposed approach of this work is to use Gaussian Processes (GPs) [19] to model position-dependent FF parameters. This both eliminates the need of a complex model of the system and does not assume a predetermined relation between the FF parameters and position. Note that a GP is

[1]This research has been conducted in partnership with ASM PT, Center of Competency in Beuningen, the Netherlands.

non-parametric and therefore capable of modelling black box or (highly) non-linear functions. Furthermore, it opens up to many optimization techniques such as Bayesian optimization [20], Mutual Information (MI) optimization [21] and active learning [22] to determine the training positions based on data. This removes the necessity to choose a grid for the training positions such that the position-dependency is modelled accurately. In summary, the main contributions of this work are formulated as:

- C1: A framework to model position-dependent FF parameters as a GP,
- C2: MI optimal training positions for the GP,
- C3: A combined framework of C1 and C2,
- C4: Validation of C1, C2 and C3 in a computer simulation and experiments on a commercial wirebonder.

The structure of this work is as follows. In Section II, the problem is defined. Next, GPs are introduced in Section III. In Section IV, MI optimization is introduced to determine near-optimal training positions for the GP. Both ILCBF and IV based FF control to determine the FF parameters are explained in respectively Section V and Section VI. In Section VII, a complete framework is presented combining ILCBF, GPs and MI optimization to identify and model the FF parameters as a function of position. In Section VIII and Section IX the framework applied to a computer simulation and an experimental setup of an ASM PT machine are presented and validated. The work is ended with some concluding remarks in Section X.

A. Preliminaries

The amount of samples in one trial is equal to $N \in \mathbb{N}^+$. For a vector x, the W norm, $\|x\|_W$, is equal to $\sqrt{x^T W x}$. Let A be a matrix, $\bar{\sigma}$(A) is the highest singular value of matrix A. $\mathbb{R}$ is used to denote all real numbers and $\mathbb{R}(q)$ expresses all real polynomials in q. The symbol q is used as the shift operator, i.e. $qu(t) = u(t+1)$. $\mathcal{A} \subseteq \mathcal{B}$ expresses that $\mathcal{A}$ is a subset of $\mathcal{B}$, which means that all elements of $\mathcal{A}$ are inside $\mathcal{B}$. $\mathcal{A} \setminus \mathcal{B}$ is used to represent the set difference between $\mathcal{A}$ and $\mathcal{B}$, meaning the elements in $\mathcal{A}$, but not in $\mathcal{B}$. The operator $\cup$ is used as set union, where $\mathcal{A} \cup \mathcal{B}$ means all elements in $\mathcal{A}$ and $\mathcal{B}$. The length of a set $\mathcal{A}$ is denoted as $|\mathcal{A}|$.

II. Problem Formulation

This section will formulate the addressed problem. In Section II-A the problem setup is discussed, in Section II-B a parametrized FF signal is introduced and in Section II-C several options for handling position-dependent FF parameters are considered. Finally, in Section II-D, the problem is defined.

A. Problem Setup

The closed-loop control structure considered can be seen in Fig. 1. A trial, iteration or task, is denoted with index j. The plant G is position-dependent and can be Multiple-Input Multiple-Output (MIMO), having n_i inputs and n_o outputs. The plant is controlled using a stabilizing feedback controller C and FF controller F. The reference, error and output are

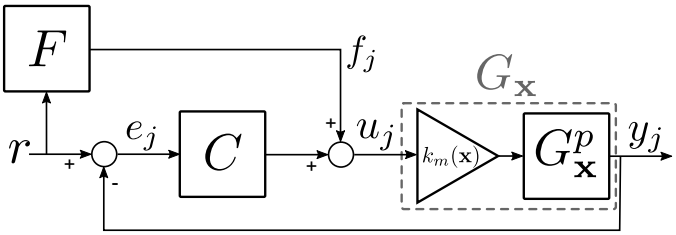

Fig. 1: Control structure with trial index j, for position-dependent plant $G_{\mathbf{x}}$, including the position-dependent motor force constant $k_m(\mathbf{x})$ and position-dependent system $G^p_{\mathbf{x}}$.

respectively defined as r, e_j, $y_j \in \mathbb{R}^{N \times n_o}$. The input to the plant is given by $u_j \in \mathbb{R}^{N \times n_i}$.

The position-dependent plant G is zero-order hold discretized and characterized as:

$$y_j(t) = \Big(G^p_{\mathbf{x}}(z) k_m(\mathbf{x})\Big) u_j(t) := G_{\mathbf{x}}(z) u_j(t), \tag{II.1}$$

with z a complex indeterminate and $t \in \mathbb{Z}$. The frozen (i.e. the LTI) dynamics [23] of position-dependent plant G at initial position $\mathbf{x} \in \mathbb{R}^{n_o}$, e.g. for an XY motion stage $\mathbf{x} = \begin{bmatrix} x_0 & y_0 \end{bmatrix}$, is denoted as $G_{\mathbf{x}}$. The following assumption ensures that the system $G_{\mathbf{x}}$ is LTI.

Assumption 1. The position-dependency due to the reference r is assumed to be negligible compared with the position-dependency due to the initial position $\mathbf{x}$, i.e. $r(t) \approx \mathbf{x}$. ■

Furthermore, (II.1) shows the plant is separated in a motor force constant $k_m(\mathbf{x}) \in \mathbb{R}^{n_i \times n_i}$ and a system $G^p_{\mathbf{x}} \in \mathbb{R}^{n_o \times n_i}$. This shows the position-dependency consists of both the physical properties of $G^p_{\mathbf{x}}$ and motor force constant $k_m(\mathbf{x})$, due to force ripple and changing magnetic flux densities [24], as illustrated in Example 1.

Example 1. Consider a mass-spring-damper system $G_{\mathbf{x}}$ with position-dependent flexible dynamics, e.g. due to flexible modes [13], and position-dependent motor force constant. The magnitude Frequency Response Function (FRF) of system $G^p_{\mathbf{x}}$ can be seen in the left of Fig. 2. The rigid-body and position-dependent flexible dynamics can be separated, resulting in the middle of Fig. 2. The combined effect of position-dependent flexible modes and motor force constant can be seen in the right of Fig. 2. This shows the position-dependent effects of the dynamics and the motor force constant are hard to isolate. ■

The error in trial j can be derived from Fig. 1 and is equal to:

$$\begin{aligned} e_j(t) =& \Big(I + G_{\mathbf{x}}(z)C(z)\Big)^{-1} r(t) \\ & - \Big(I + G_{\mathbf{x}}(z)C(z)\Big)^{-1} G_{\mathbf{x}}(z) f_j(t), \\ :=& S_o(z) r(t) - S_o(z) G_{\mathbf{x}}(z) f_j(t), \end{aligned} \tag{II.2}$$

where $S_o(z)$ is the output sensitivity of the frozen plant $G_{\mathbf{x}}$ and feedback controller C. The objective in learning control is

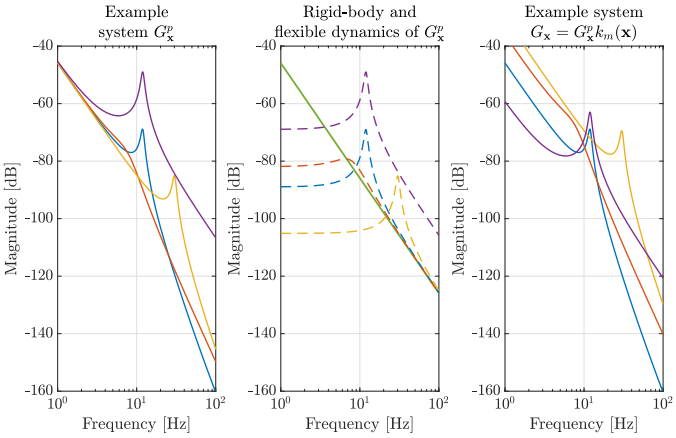

Fig. 2: Left: systems $G^p_{\mathbf{x_1}}$ (—), $G^p_{\mathbf{x_2}}$ (—), $G^p_{\mathbf{x_3}}$ (—) and $G^p_{\mathbf{x_4}}$ (—). Middle: separation of rigid-body (—) and position-dependent flexible dynamics of $G^p_{\mathbf{x_1}}$ (- -), $G^p_{\mathbf{x_2}}$ (- -), $G^p_{\mathbf{x_3}}$ (- -) and $G^p_{\mathbf{x_4}}$ (- -). Right: combined position-dependent dynamics and motor force constant, i.e. $G_{\mathbf{x}} = G^p_{\mathbf{x}} k_m(\mathbf{x})$, using $G_{\mathbf{x_1}}$ with $k_m = 1$ (—), $G_{\mathbf{x_2}}$ with $k_m = 3$ (—), $G_{\mathbf{x_3}}$ with $k_m = 6$ (—) and $G_{\mathbf{x_4}}$ with $k_m = 0.2$ (—).

to learn the FF force f_{j+1} in a trial-to-trial fashion to reduce the error e_{j+1}:

$$\begin{aligned} e_{j+1}(t) &= S_o(z)r(t) - S_o(z)G_{\mathbf{x}}(z)f_{j+1}(t) \\ &= e_j(t) - S_o(z)G_{\mathbf{x}}(z)\Big(f_{j+1}(t) - f_j(t)\Big). \end{aligned} \tag{II.3}$$

This shows the FF force that minimizes e_{j+1} becomes invariant under changes of the reference. As a result, extrapolation to other references deteriorates performance. The following section will introduce a FF parametrization in order to achieve extrapolation capabilities.

B. Parameterized FF signal

In [5–7] BF are added to learning control such that extrapolation to other references is achieved. The FF force f_j is parametrized using the BF Ψ, which is a function of the reference r, and the FF parameters $\vec{\theta}_j$:

$$f_j(t) = \Psi\big(r(t)\big)\vec{\theta}_j. \tag{II.4}$$

This is visually supported by Fig. 3.

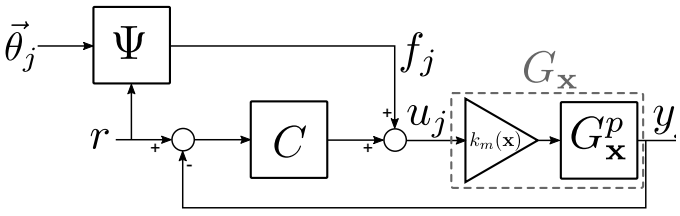

Fig. 3: FF force parametrization in the closed loop control structure seen in Fig. 1, using the BF Ψ and FF parameters $\vec{\theta}_j$.

The selection of Ψ is important to achieve good tracking performance. Zero reference induced error can be achieved by designing:

$$\Psi\big(r(t)\big)\vec{\theta}_j \approx G^{-1}_{\mathbf{x}}(z)r(t). \tag{II.5}$$

This can be seen by substituting (II.4) and (II.5) into (II.2):

$$\begin{aligned} e_j(t) &= S_o(z)\Big(r(t) - G_{\mathbf{x}}(z)\Psi\big(r(t)\big)\vec{\theta}_j\Big), \\ &\rightarrow \Psi\big(r(t)\big)\vec{\theta}_j = G^{-1}_{\mathbf{x}}(z)r(t) \rightarrow e_j(t) = 0. \end{aligned} \tag{II.6}$$

The matrix Ψ can be either a full or block diagonal matrix:

$$\Psi\big(r(t)\big) = \begin{bmatrix} \Psi_{1,1}\big(r(t)\big) & \cdots & \Psi_{1,n_\theta}\big(r(t)\big) \\ \vdots & \ddots & \vdots \\ \Psi_{n_i,1}\big(r(t)\big) & \cdots & \Psi_{n_i,n_\theta}\big(r(t)\big) \end{bmatrix} \tag{II.7}$$

Typically, the BF $\Psi_{k,l}$ are chosen to be differentiators or polynomials of the reference [6, 9, 10]. Note that selecting Ψ as a function of r, the FF force f_j becomes a function of the reference as well, thus achieving extrapolation capabilities. In addition, the FF parameters $\vec{\theta}$ should be modelled as a function of position, in order to fulfill the condition seen in (II.5) for any $\mathbf{x}$. Next, an example position-dependent system is analyzed and the corresponding FF parametrization is determined such that (II.5) holds.

Example 2. Suppose a SISO mass-damper system G is discretized using a backward Euler method and has transfer function:

$$G_{\mathbf{x}} \begin{cases} G^p_{\mathbf{x}} & = \frac{1}{m(\mathbf{x})\left(\frac{1-z^{-1}}{T_s}\right)^2 + c(\mathbf{x})\left(\frac{1-z^{-1}}{T_s}\right)}, \\ k_m & = 1, \end{cases} \tag{II.8}$$

where T_s is the sampling time, m is the mass and c a damping coefficient. To satisfy (II.5), the FF force f_j could be parametrized as:

$$f_j(t) = \theta_{j,1}\ddot{r}(t) + \theta_{j,2}\dot{r}(t), \tag{II.9}$$

which shows that the FF force is linear in the acceleration and velocity of the reference. The values of the FF parameters which achieve minimal tracking error can directly be seen by evaluating $G^{-1}_{\mathbf{x}}(z)r(t)$, which is equal to $m(\mathbf{x})\ddot{r}(t) + c(\mathbf{x})\dot{r}(t)$. Specifically, $\theta_{j,1}$ and $\theta_{j,2}$ should be equal to the mass $m(\mathbf{x})$ and damping coefficient $c(\mathbf{x})$, showing the necessity to model the FF parameters as a function of position.

Moreover, suppose the motor force constant is position-dependent, i.e.:

$$k_m := k_m(\mathbf{x}). \tag{II.10}$$

The plant $G_{\mathbf{x}}$ is then described as:

$$G_{\mathbf{x}} = \frac{k_m(\mathbf{x})}{m(\mathbf{x})\left(\frac{1-z^{-1}}{T_s}\right)^2 + c(\mathbf{x})\left(\frac{1-z^{-1}}{T_s}\right)}. \tag{II.11}$$

Without knowledge of $k_m(\mathbf{x})$, both $m(\mathbf{x})$ and $c(\mathbf{x})$ cannot be identified. However, the parameters $\theta_{j,1} := k_m(\mathbf{x})/m(\mathbf{x})$ and $\theta_{j,2} := k_m(\mathbf{x})/c(\mathbf{x})$ can still be identified and are able to achieve zero reference induced tracking error. The parameters $\theta_{j,1}$ and $\theta_{j,2}$ combine the position-dependent effect of both the physical properties and the motor force constant. This shows the necessity to model the FF parameters as a function of position, without limiting to the position-dependency of either $G^p_{\mathbf{x}}$ or $k_m(\mathbf{x})$. ■

C. Methods for Position-Dependent FF Parameters

Several options for handling position-dependent FF parameters are possible. For example, three methods which are typically applied to position-dependent systems are:

- ignore the position-dependent characteristics of $G_{\mathbf{x}}$ and therefore using constant FF parameters,
- nearest-neighbour search on finite set of FF parameters at different training positions [14, 15] and
- interpolation on finite set of FF parameters at different training positions [16, 17].

Apart from being simple to implement, these three methods all have their disadvantages. First, when the position-dependency of the FF parameters is ignored, the position-dependent dynamics are not taken into account, hence the FF controller cannot be equal to the inverse plant. Secondly, the nearest-neighbour method is limited to a piecewise constant dependency on position, whereas the plant can have any position-dependency. Thirdly, the interpolation requires a predetermined relation between the FF parameters and the position, which might be unknown and is therefore likely to be incorrect. As a result, the three methods do not achieve minimal reference induced tracking error. Additionally, for both the nearest-neighbour and interpolation methods, it is unclear how the training positions for the FF parameters should be chosen. The training positions could be picked based on prior knowledge of the position-dependent FF parameters, which is often not available. The positions can also be determined arbitrarily, such as a grid or randomly. However this results in sub-optimal accuracy for the estimation of FF parameters in the operating range of the machine [25]. Due to the disadvantages of the state-of-the-art approaches, an improved method is required.

D. Problem Definition

The three identified problems that are considered in this work are:

- P1: How to model position-dependent FF parameters accurately?
- P2: How to learn FF parameters on a single position, which can be used to model position-dependent FF parameters?
- P3: How to pick the training positions when identifying the FF parameters, such that the position-dependent FF parameters are modelled as accurately as possible?

The following sections describe that a GP, ILCBF or IV FF control and MI optimization can solve these problems.

III. Gaussian Process Regression

This section presents GPs, which are used to model the position-dependent FF parameters accurately and thus tackles P1 of the problem formulation. First, some general definitions of a GP are given. Second, covariance functions, the prior and the posterior are discussed. Finally, some practical aspects in terms of the hyperparameter optimization and mean functions are presented.

A. Gaussian Processes

A GP is defined as a collection of random variables $f(\mathbf{x})$, indexed by $\mathbf{x} \in \mathbb{R}^D$, such that the joint distribution of any finite subset of random variables is multivariate Gaussian. The GP is then written as

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right). \tag{III.1}$$

This shows a GP is fully defined by its covariance function (or kernel) $k(\mathbf{x}, \mathbf{x}')$ and the mean function $m(\mathbf{x})$:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\Big(f(\mathbf{x}) - m(\mathbf{x})\Big)\Big(f(\mathbf{x}') - m(\mathbf{x}')\Big)\right], \\ m(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right]. \end{aligned} \tag{III.2}$$

The mean function m can be interpreted as the mean at any input point and the covariance function k as the *similarity* between values of $f(\mathbf{x})$ trained on different positions. The mean function m is often taken equal to zero, but can also be non-zero, as will be seen in Section III-F. The training data is defined by sampling the function on inputs and measuring the output $\mathbf{y}$:

$$\begin{aligned} \mathbf{y} &= f(\mathbf{x}) + \epsilon, \\ \text{where} : \epsilon &\sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I}), \end{aligned} \tag{III.3}$$

where σ_ϵ^2 is the variance of the noise acting on output. This shows direct access to the function is not available, but a noisy measurement thereof.

Definition 1. Each FF parameter is modelled using a separate GP, i.e.:

$$\vec{\theta}(\mathbf{x}) := \vec{f}(\mathbf{x}), \tag{III.4}$$

as a result, the FF parameters are modelled as a function of position, e.g. $\mathbf{x} = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}$, using a GP. The training data, seen in (III.3), for FF parameters is defined as:

$$\mathbf{y} = \begin{bmatrix} \boldsymbol{\theta}_1 & \cdots & \boldsymbol{\theta}_k \end{bmatrix}^\top \in \mathbb{R}^{l \times 1}, \tag{III.5}$$

where k is the amount of unique training positions and $\boldsymbol{\theta}$ is:

$$\boldsymbol{\theta}_i = \begin{bmatrix} \theta_{i,1} & \cdots & \theta_{i,p} \end{bmatrix}^\top \in \mathbb{R}^{p \times 1}, \tag{III.6}$$

where p is the amount of FF parameters per training position and $i \in \{1, \ldots, k\}$. ■

B. Covariance Function

The covariance function or *kernel* $k(\mathbf{x}, \mathbf{x}')$ specifies the covariance between the inputs $\mathbf{x}$ and $\mathbf{x}'$. Covariance functions are often *stationary*, meaning it only depends on the difference between the inputs. An example of a stationary covariance function is the squared exponential or Radial Basis Function (RBF) covariance function, which is used throughout this work:

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\top \mathbf{L}(\mathbf{x}-\mathbf{x}')}, \tag{III.7}$$

which shows that the entries for the covariance function are low when the inputs are far away from each other and close to σ_f^2 when they are close to each other. In addition, the σ_f^2 and

the $\mathbf{L}$ are the so-called hyperparameters of respectively size 1 and $D \times D$. The hyperparameters are tuned in Section III-E. Typically, the matrix $\mathbf{L}$ is chosen as:

$$\mathbf{L} = \text{diag}\left(\vec{\ell}\right)^{-2}, \tag{III.8}$$

where $\vec{\ell}$ is a vector of positive length scales of size D. The use of multiple length scales enables the user to specify how relevant each input in $\mathbf{x}$ is. Another example of a stationary kernel is the periodic kernel, that models functions which repeat themselves:

$$k_{per}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 e^{\frac{-2\sin^2\left(\pi\frac{|\mathbf{x}-\mathbf{x}'|}{p}\right)}{\ell^2}}, \tag{III.9}$$

where now p is equal to the period of the kernel, which determines the distance between repetitions of the function. Note that this kernel only uses one length scale and period, but can be extended to contain multiple length scales and periods [19, Section 5.1]. Many other (stationary) covariance functions are possible, such as the Matérn covariance function [26, Chapter 2].

Interpretation 1. Both the RBF and periodic kernel have an easily interpretable application for FF parameters. Firstly, when using an RBF kernel, the similarity of the FF parameters will only depend on the distance between the positions of the FF parameters. Secondly, a periodic kernel models the spatial-periodic effects of the FF parameters, for instance due to force ripple. ■

C. Gaussian Process Prior

The definition of a covariance function seen in (III.2) suggests there exists a distribution over functions. This is illustrated by using the covariance function from (III.7) in a random number generator, assuming zero mean function and drawing any (finite) amount of samples. Specifically, samples are drawn at indices X_* called *test inputs*, which can be any single input or vector of inputs, e.g.:

$$X_* = \begin{bmatrix} X_{*,1} & X_{*,2} & \cdots & X_{*,l_*} \end{bmatrix}^\top \in \mathbb{R}^{l_* \times D}, \tag{III.10}$$

where l_* is the amount of test inputs and D the input dimension. A random Gaussian vector, assuming the prior mean function $m(\mathbf{x})$ to be zero, can be calculated using the covariance function and the distribution:

$$f(X_*) \sim \mathcal{N}(\mathbf{0}, K(X_*, X_*)). \tag{III.11}$$

This is the so-called *prior* distribution. The matrix $K(X_*, X_*) \in \mathbb{R}^{l_* \times l_*}$ is equal to the covariance function $k(\mathbf{x}, \mathbf{x}')$, evaluated at the test inputs X_*. Some examples of samples drawn from the prior distribution can be seen in Fig. 4.

Interpretation 2. The prior is as a collection of information that is known about the function before sampling the function. This can therefore be used to supply the GP with prior knowledge about the FF parameters. ■

D. Gaussian Process Posterior

Usually, generating random Gaussian vectors using the prior is not helpful, but knowledge of the function provided by the *training data* can be integrated in this distribution. This means that function samples can be incorporated into the prior distribution, resulting in a new distribution which can be used to make estimations, called regression.

The observations are done at inputs X, called the *training inputs*, which in this work are equal to the training positions:

$$X = \begin{bmatrix} X_1 & X_2 & \cdots & X_l \end{bmatrix}^\top \in \mathbb{R}^{l \times D}, \tag{III.12}$$

where l is the amount of training inputs. Recall from (III.3) that the observations of the functions or *training outputs* are defined as:

$$\begin{aligned} \mathbf{y} &= f(X) + \epsilon, \\ \text{where : } \epsilon &\sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I}). \end{aligned} \tag{III.13}$$

The parameter σ_ϵ^2 is approximated using the parameter σ_n^2. The σ_n^2 is considered an additional hyperparameter, called the noise variance hyperparameter and is tuned in Section III-E.

Estimations of the unknown function $f(X_*)$ have to be made, called the *test outputs*. The joint distribution in a GP between the training and test outputs, again assuming $m(\mathbf{x}) = 0$, is described as [19]:

$$\begin{bmatrix} \mathbf{y} \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 \mathbf{I} & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right), \tag{III.14}$$

where the kernels $K_y := K(X,X) + \sigma_n^2 \mathbf{I}$, $K_* := K(X, X_*) = K^\top(X_*, X)$ and $K_{**} := K(X_*, X_*)$ are respectively of size $l \times l$, $l \times l_*$ and $l_* \times l_*$. The joint Gaussian distribution in (III.14), the posterior, can be *conditioned* on the function observations $\mathbf{y}$ using Bayesian inference [19]:

$$f(X_*) \Big| \left[X_*, X, \mathbf{y}\right] \sim \mathcal{N}\Big(\bar{f}(X_*), \mathbb{V}\big[f(X_*)\big]\Big), \tag{III.15}$$

where:

$$\bar{f}(X_*) := \mathbb{E}\big[f(X_*)\big] = K_*^\top K_y^{-1} \mathbf{y}, \tag{III.16}$$

$$\mathbb{V}\big[f(X_*)\big] = K_{**} - K_*^\top K_y^{-1} K_*. \tag{III.17}$$

This is called the *posterior distribution*. This shows that the posterior mean, $\bar{f}(X_*)$, is unequal to zero, regardless of the prior mean function $m(x)$.

The prior and posterior create a framework to train a GP with function observations, enabling the user to estimate function values. The training of a GP is visually supported, with input dimension one, in Fig. 4 and Fig. 5.

Interpretation 3. This framework can be used to train a GP based on finite measurements of FF parameters. The assumption of noisy readings also holds true for measurements of FF parameters, e.g. FF parameters determined with ILCBF, show variations due to measurement noise [5]. As a result, the framework is robust to FF parameters determined with variance. ■

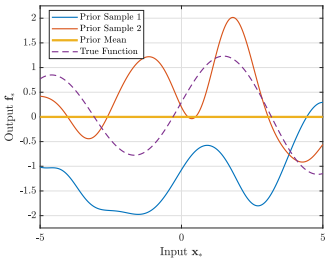

Fig. 4: Two samples drawn from the prior, the prior mean and an example unknown function to regress.

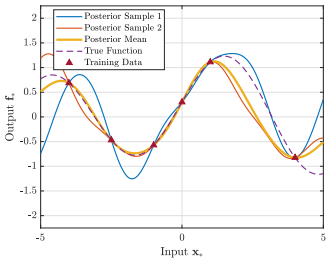

Fig. 5: Two samples drawn from the posterior, the posterior mean, the unknown function to regress and the training data.

E. Hyperparameter Optimization

The hyperparameters seen in the previous sections still need to be chosen. These can be arbitrarily picked or manually tuned. However, this is not desirable since there are often many hyperparameters, making arbitrary picking or manual tuning inaccurate and tedious. An automatic way of tuning the hyperparameters is therefore preferred. Typically, the marginal likelihood optimization is used to automatically tune the hyperparameters.

The log marginal likelihood, can be described as [19]:

$$\log\left(p\left(\mathbf{y} \mid X, \vec{\beta}\right)\right) = -\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - c_1. \quad \text{(III.18)}$$

This shows the log marginal likelihood needs to invert matrix K_y and therefore has computational complexity of $\mathcal{O}(l^3)$ per evaluation. The log marginal likelihood has three readily interpretable terms. The last term, $c_1 = \frac{l}{2}\ln 2\pi$, is a normalization constant. The second term is the complexity penalty term. The first term is the data fit term, the only term containing the observed function values. A good data fit might be achieved by giving the matrix K_y very high values, making the term $-\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y}$ very small. This translates into a model which assumes there is so much noise, that all data fits within the model. However, the second term, the complexity penalty $-\frac{1}{2}\log|K_y|$, prevents this. This shows that maximizing the marginal likelihood automatically makes a trade-off between model complexity and data fit (regularization) through its foundation in Bayesian probability theory [27].

F. Mean Functions

A mean function, which was previously taken equal to zero, can also be implemented in the GP. Taking a zero mean function implies there is no prior knowledge of the mean. This is often sufficient since the posterior mean is unequal to zero, regardless of the mean function. However, when using stationary covariance functions, the posterior mean in (III.16) will be close to zero when evaluated (far) outside the training inputs, i.e. extrapolating. A mean function can prevent this. Many functions often have an average value of higher than zero and therefore it might be useful to use a mean function, such that the posterior mean is not necessarily close to zero when extrapolating.

A common method of applying a mean function in a GP is to use explicit basis functions, the mean function is then defined as:

$$m(\mathbf{x}) = h(\mathbf{x})^\top \vec{\gamma}, \quad \text{(III.19)}$$

where $h \in \mathbb{R}^{l \times n_\gamma}$ is a matrix containing explicit basis functions and $\vec{\gamma} \in \mathbb{R}^{n_\gamma \times 1}$ a vector containing the parameters of the explicit basis functions. An example of an explicit basis function could be polynomial , e.g. $h(\mathbf{x}) = \begin{bmatrix}\mathbf{1} & \mathbf{x} & \mathbf{x}^2 & \cdots\end{bmatrix}^\top$. Tuning the parameters in $\vec{\gamma}$ can be done using a regression method, such as least squares. A different method is described in [19, Section 2.7], using a prior on $\vec{\gamma}$ and optimizing the hyperparameters in parallel with the covariance function hyperparameter optimization.

The posterior mean, using (III.19) as mean function, changes to:

$$\mathbb{E}\left[f(X_*)\right] = h(X_*)^\top\vec{\gamma} + K_*^\top K_y^{-1}\left(\mathbf{y} - h(X)^\top\vec{\gamma}\right), \quad \text{(III.20)}$$

where $h(X)$ and $h(X_*)$ are respectively equal to the explicit basis function evaluated at the training and test inputs. The posterior variance in (III.17), changes to:

$$\begin{aligned}\mathbb{V}\left[f(X_*)\right] =& K_{**} - K_*^\top K_y^{-1} K_* \\ &+ R^\top\left(h(X)K_y^{-1}h(X)^\top\right)R,\end{aligned} \quad \text{(III.21)}$$

where $R = h(X_*) - h(X)\ K_y^{-1}K_*$. Note that when using a constant mean function, i.e. $h(\mathbf{x}) = \mathbf{1}$, the posterior variance from (III.17) remains unchanged.

Interpretation 4. The mean function enables the user to supply additional prior information of the underlying function or FF parameters and will simultaneously result in better extrapolation on the data. For the FF parameters a constant mean function implies the prior information of the mean is a constant value, that is assuming no position-dependency on the FF parameters as a base value. ■

Using a GP, the FF parameters can now be modelled as a function of position using training data. Next, several options to determine the training positions are discussed.

IV. Mutual Information Optimization

This section will present MI optimization for choosing the near-optimal training positions for a GP, hence tackles P3 of the problem formulation.

The positions where the FF parameters are measured is an essential factor for the quality of a regression. This is considered a sensor placement problem, see e.g. [21]. For a GP, consider again the situation where estimations of the unknown function $f(\mathbf{x})$ have to be made at certain test positions. The test positions can be located anywhere, covering the operating range of the machine. The training positions X have to be determined and should be such that the FF parameters are modelled as accurately as possible.

Firstly, the MI optimization method is reviewed. Finally, a greedy approximation for the MI, such that the computations are feasible, is presented.

A. Mutual Information

MI optimization tries to find training positions for the GP which are most informative about the positions where no training positions are located. The possible training and test positions are considered as a set of discrete locations $\overline{X}$, generally a fine grid covering the domain.

The goal of MI is defined as the set of k training positions that give good predictions in the uninstrumented positions [28]:

$$X^{MI} = \underset{X \subseteq \overline{X}, |X|=k}{\arg\max} \; MI[X], \tag{IV.1}$$

where X^{MI} is the set of MI optimal training positions and of size $k \times D$. The MI is defined as:

$$MI[X] = H\left[f(\overline{X} \setminus X)\right] - H\left[f(\overline{X} \setminus X) \mid \mathbf{y}\right], \tag{IV.2}$$

where H is the entropy function. Combining (IV.1) and (IV.2) shows that MI maximizes the difference of the entropy for the unobserved space using no observations, compared with the entropy where observations are used. The optimization of the MI in (IV.1) is an NP-complete problem, due to the many decisions possible when choosing a set of k training positions. Therefore, a greedy approximation is applied, which is seen in the following section.

B. Greedy Mutual Information Approximation

The greedy algorithm [29, 30] consecutively adds a single training position x to the greedy MI set X^G, until k training positions have been chosen, introducing subscript i:

$$X^G_{i+1} = X^G_i \; \cup \underset{x \subseteq \left(\overline{X} \setminus X^G_i\right)}{\arg\max} \left[\delta_x\right], \tag{IV.3}$$

where:

$$\begin{aligned} \delta_x :&= MI[X^G_i \cup x] - MI[X^G_i] \\ &= H\Big[f(x) \mid \mathbf{y}\Big] - H\Big[f(x) \mid f(\hat{X})\Big], \end{aligned} \tag{IV.4}$$

and $\mathbf{y}$ are the function observations at X^G_i and $\hat{X}$ is $\overline{X} \setminus (X^G_i \cup x)$. An example of the term δ_x as a function of position, using the fine grid $\overline{X}$, can be seen in Fig. 20. In [21] the entropy functions H conditioned on the sets of variables are written as a function of their variances:

$$\begin{aligned} H\Big[f(x) \mid \mathbf{y}\Big] &= \frac{1}{2}\ln\left(\sigma^2_{f(x)|\mathbf{y}}\right) + c_2, \\ H\Big[f(x) \mid f(\hat{X})\Big] &= \frac{1}{2}\ln\left(\sigma^2_{f(x)|f(\hat{X})}\right) + c_2, \end{aligned} \tag{IV.5}$$

where c_2 is equal to $\frac{1}{2}\big(\ln(2\pi)+1\big)$. For a GP, assuming zero mean function, the posterior variance is given by:

$$\begin{aligned} \sigma^2_{f(x)|\mathbf{y}} &= k(x,x) - K(x,X)\Big(K(X^G_i, X^G_i) + \sigma_n^2 I\Big)^{-1} K(X,x), \\ \sigma^2_{f(x)|f(\hat{X})} &= k(x,x) - K(x,\hat{X})\Big(K(\hat{X},\hat{X}) + \sigma_n^2 I\Big)^{-1} K(\hat{X},x). \end{aligned} \tag{IV.6}$$

By substitution of (IV.6) in (IV.5) and using (IV.4), the term δ_x is expressed as:

$$\delta_x = \frac{k(x,x) - K(x,X^G_i)\Big(K(X^G_i,X^G_i) + \sigma_n^2 I\Big)^{-1} K(x,X^G_i)^\top}{k(x,x) - K(x,\hat{X})\left(K(\hat{X},\hat{X}) + \sigma_n^2 I\right)^{-1} K(x,\hat{X})^\top} \tag{IV.7}$$

Using (IV.3) and (IV.7), the MI optimal training positions are computed in a greedy manner. The MI optimal training positions can both be computed using an *a priori* or a *sequential* optimization. The a priori method optimizes the MI before sampling the function, whereas the sequential method selects new positions based on previous observations [25]. Specifically, the sequential method iteratively samples the function, optimizes the hyperparameters of the GP based on the new observations and follows with a (greedy) maximization of the MI. The a priori and sequential method are discussed below.

1) A Priori greedy MI optimization: Since the greedy optimization of the MI requires only the kernel hyperparameters to be known and not the training outputs $\mathbf{y}$, the algorithm can be executed a priori. Specifically, the MI optimal training positions X^G are chosen before any training outputs $\mathbf{y}$ are generated. However, since the hyperparameters used in the kernels seen in (IV.7) are often not (exactly) known and no data is available such that the marginal likelihood can be optimized, the a priori training positions will be sub-optimal. In Algorithm 1, an a priori greedy MI optimization algorithm can be seen.

Algorithm 1: Greedy MI algorithm to determine optimal training positions using an a priori manner.

Input: Hyperparameters for kernels, see e.g. (IV.7), discretized space $\overline{X}$
Output: Near-optimal greedy training positions X^G_k

$X^G_0 \leftarrow \emptyset$;
for $i = 1$ **to** k **do**
  for $x \in \overline{X} \setminus X^G_i$ **do**
    $\hat{X} \leftarrow \overline{X} \setminus \left(X^G_i \cup j\right)$;
    Calculate $k(x,x)$, $K(X^G_i, X^G_i)$, $K(x, X^G_i)$, $K(x,\hat{X})$ & $K(\hat{X},\hat{X})$;
    Calculate δ_x ;
  $X^G_i \leftarrow X^G_{i-1} \cup \arg\max_x \delta_x$
return X^G_k;

2) Sequential greedy MI optimization: A different approach is to tackle the problem in a sequential manner. In contrast to the a priori approach, the kernel hyperparameters do not need to be known beforehand, since they will be optimized using the training data. The hyperparameters are optimized according to Section III-E. Algorithm 3 in Appendix A describes the sequence which is performed, such that the MI near-optimal training positions are determined in a sequential manner.

In conclusion, using either an a priori or sequential MI optimization results in near-optimal training positions to determine the FF parameters for the GP. The advantage of this method

compared with choosing an arbitrary grid is seen by comparing the posterior variances of Fig. 11 and Fig. 12.

V. ILC with Basis Functions

The FF parameters which are used as training data for the GP, still need to be determined. This section presents ILCBF to determine the FF parameters on different positions, therefore solves P2 of the problem formulation. Firstly, ILCBF is introduced and is followed by the convergence analysis.

A. ILC with Basis Functions

ILCBF parametrizes the FF signal and learns the FF parameters in a trial-to-trial fashion. The optimization criterion in ILCBF is specified as [6, 10, 31]:

$$V\left(\vec{\theta}_{j+1}\right) := \|e_{j+1}\|^2_{W_e} + \|f_{j+1}\|^2_{W_f} + \|f_{j+1} - f_j\|^2_{W_{\Delta f}}, \quad \text{(V.1)}$$

with $W_e, W_f, W_{\Delta f}$ positive (semi-)definite weighting matrices and f is parametrized using the BF Ψ as seen in (II.4). Recall from Section II that the error in trial $j+1$ can be written as:

$$\begin{aligned} e_{j+1}(t) &= S_o(z)r(t) - S_o(z)G_{\mathbf{x}}(z)f_{j+1}(t) \\ &= e_j(t) - S_o(z)G_{\mathbf{x}}(z)\Big(f_{j+1}(t) - f_j(t)\Big). \end{aligned} \quad \text{(V.2)}$$

When the FF force is parametrized using the BF, as seen in (II.4), the error in trial $j+1$ can be expressed as:

$$e_{j+1}(t) = e_j(t) - S_o(z)G_{\mathbf{x}}(z)\ \Psi\big(r(t)\big)\Big(\vec{\theta}_{j+1} - \vec{\theta}_j\Big) \quad \text{(V.3)}$$

The optimal solution of (V.1), obtained by solving $\frac{\partial V\left(\vec{\theta}_{j+1}\right)}{\partial \vec{\theta}_{j+1}} = 0$ for (V.1), results in:

$$\vec{\theta}_{j+1} = Q\vec{\theta}_j + Le_j(t), \quad \text{(V.4)}$$

where:

$$\begin{aligned} Q &= \Big(\Psi^\top\Big(J^\top W_e J + W_f + W_{\Delta f}\Big)\Psi\Big)^{-1}\Psi^\top\Big(J^\top W_e J + W_{\Delta f}\Big)\Psi, \\ L &= \Big(\Psi^\top\Big(J^\top W_e J + W_f + W_{\Delta f}\Big)\Psi\Big)^{-1}\Psi^\top J^\top W_e, \end{aligned} \quad \text{(V.5)}$$

and J is the impulse response matrix of $S_o(z)G_{\mathbf{x}}(z)$. The values for Q and L in combination with (V.4) can be used to learn the FF parameters in a trial-to-trial fashion.

B. Convergence Analysis

The ILCBF algorithm can be analyzed to assess the convergent properties of the scheme. f_{j+1} can be rewritten in terms of f_j by rewriting (V.4):

$$f_{j+1}(t) = (Q - LJ)f_j(t) + LS_o(z)r(t). \quad \text{(V.6)}$$

Monotonic convergence, with respect to f_{j+1}, can be achieved by [17]:

$$\bar{\sigma}(Q - LJ) < 1, \quad \text{(V.7)}$$

and is guaranteed if:

$$\Psi^T\Big(J^T W_e J + W_f + W_{\Delta f}\Big)\Psi \succ 0. \quad \text{(V.8)}$$

The matrices W_e, W_f and $W_{\Delta f}$ can be chosen such that (V.8) is satisfied.

1) Convergence Analysis for Position-Dependent Systems: The convergence criteria specified in (V.7) and (V.8) should also hold true for position-dependent systems to achieve monotonic convergence. Since there is often no model of the position-dependency, it is more difficult to explicitly check (V.7). However, (V.8) shows when W_f or $W_{\Delta f}$ are sufficiently high, monotonic convergence is guaranteed. Here, W_f is used to create robustness for model uncertainties and $W_{\Delta f}$ for trial variant disturbances [6]. Robustness can be improved by increasing W_f or $W_{\Delta f}$, respectively at the cost of maximum attainable performance and convergence speed. Therefore, to satisfy (V.8), W_f can be increased to cope with robustness with respect to model mismatch because of position-dependent dynamics.

VI. Instrumental Variable FF Control

The FF parameters determined with ILCBF can be biased due to measurement noise [11]. This section shows a different procedure for determining the FF parameters, being an alternative solution to P2, that results in unbiased FF parameter estimates. This is done with the use of IVs in combination with iterative FF parameter learning. Additionally, IV based FF control can learn the FF parameters without the use of a model.

First, the setup for IV based FF control is described, followed by the addition of IVs to the control problem. Afterwards, computation and accuracy of the estimated optimal IVs are presented. Finally, an extension to MIMO systems for IV FF control is proposed.

A. Setup for IV FF Control

Consider the SISO controller structure seen in Fig. 6, where $C^{fb}(q) \in \mathbb{R}(q)^{1\times1}$ is the feedback controller, $G_{\mathbf{x}} \in \mathbb{R}(q)^{1\times1}$ the plant, $r(t) \in \mathbb{R}^{N\times1}$ the reference, $e_j(t) \in \mathbb{R}^{N\times1}$ the error, $f_j(t) \in \mathbb{R}^{N\times1}$ the FF force, $u_j(t) \in \mathbb{R}^{N\times1}$ the input, $y^m_j(t) \in \mathbb{R}^{N\times1}$ the measured output, $C^{ff}_j(q) \in \mathbb{R}(q)^{1\times1}$ the FF controller and $C^{ff}_\Delta(q) \in \mathbb{R}(q)^{1\times1}$ the update of the FF controller in each trial.

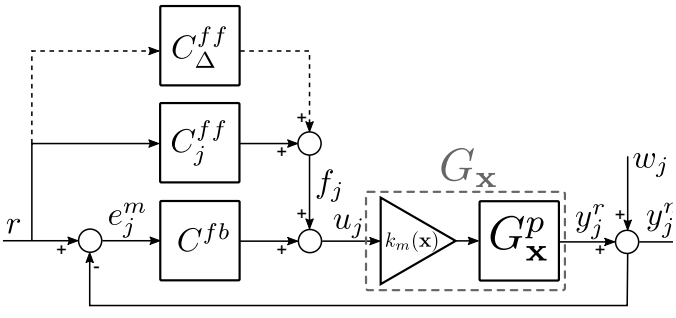

Fig. 6: SISO FB and FF control structure for IV based FF control.

This figure shows the iterative update scheme of the FF controller using C^{ff}_Δ. The update law is defined as follows:

$$C^{ff}_{j+1}(q) = C^{ff}_j(q) + C^{ff}_\Delta(q) = \Psi(q)\vec{\theta}_j + \Psi(q)\vec{\theta}_\Delta \quad \text{(VI.1)}$$

where $\Psi(q) \in \mathbb{R}(\mathrm{q})^{1\times n_\theta}$ is the BF matrix, i.e.:

$$\Psi(q) = \begin{bmatrix}\psi_1(q^{-1}) & \cdots & \psi_{n_\theta}(q^{-1})\end{bmatrix}. \tag{VI.2}$$

The FF parameter vectors $\vec{\theta}_j$ and $\vec{\theta}_\Delta$ are gathered in a similar manner:

$$\vec{\theta}_j = \begin{bmatrix}\theta_{j,1}\\ \vdots \\ \theta_{j,n_\theta}\end{bmatrix}, \quad \vec{\theta}_\Delta = \begin{bmatrix}\theta_{\Delta,1}\\ \vdots \\ \theta_{\Delta,n_\theta}\end{bmatrix}. \tag{VI.3}$$

The predicted error, for now assuming zero measurement noise, of the system in trial $j+1$ based on measurements of trial j can be expressed as:

$$\hat{\epsilon}_{j+1}(t,\vec{\theta}_\Delta) = e_j^m(t) - S_o(q)G_{\mathbf{x}}(q)C_\Delta^{ff}(q,\vec{\theta}_\Delta)r(t), \tag{VI.4}$$

where e_j^m is the measured error in trial j. Since the plant is considered to be unknown, the predicted error is estimated as:

$$\hat{e}_{j+1}(t,\vec{\theta}_\Delta) = e_j^m(t) - C_\Delta^{ff}(q,\vec{\theta}_\Delta)\left(C^{fb}(q)+C_j^{ff}(q)\right)^{-1} y_j^m(t), \tag{VI.5}$$

where now y_j^m is the measured output of the plant. The result of (VI.5) can be obtained by defining:

$$\hat{y}_j^m(t) = S_o(q)G(q)\left(C_j^{ff}(q)+C^{fb}(q)\right)r(t), \tag{VI.6}$$

and substituting it in (VI.5), resulting in (VI.4). The estimated error in trial $j+1$ can now be written as

$$\hat{e}_{j+1}(t,\vec{\theta}_\Delta) = e_j^m(t) - (\varphi_j(t))^\top \vec{\theta}_\Delta, \tag{VI.7}$$

where:

$$\varphi_j(t) = \Psi(q)\left(C^{fb}(q)+C_j^{ff}(q)\right)^{-1} y_j^m(t). \tag{VI.8}$$

The equations presented above will be used to learn the FF parameters in a trial-to-trial fashion using IVs without the use of a model.

B. Iterative FF control with Instrumental Variables

Iterative FF control based on IVs is proposed in [11], which in contrast to ILCBF results in unbiased estimates of the FF parameters without the need of a model for the measurement noise. Now, the measurement noise w_j is not assumed to be zero anymore. Herein, a performance criterion for iterative learning is defined as:

$$V\left(\vec{\theta}_\Delta\right) = \left\| \frac{1}{N}\sum_{t=1}^{N} z(t)L(q)\hat{e}_{j+1}\left(t,\vec{\theta}_\Delta\right)\right\|_W^2, \tag{VI.9}$$

where $z(t) \in \mathbb{R}^{N\times n_z}$ are the IVs that should be uncorrelated with the measurement noise w_j. W is a positive definite weighting matrix, $L(q)$ a prefilter and $\hat{e}_{j+1}$ as in (VI.7). Minimum variance is obtained when n_z is equal to n_θ and W and $L(q)$ are taken as unity matrix [12]. Therefore, from now on:

$$n_z = n_\theta, \qquad L(q) = \mathbf{I}, \qquad W = \mathbf{I}. \tag{VI.10}$$

The optimal solution to the performance criterion in (VI.9) is equal to:

$$\begin{aligned}\vec{\theta}_\Delta^{IV} &= \left(\frac{1}{N}z(t)^\top\varphi_j(t)\right)^{-1}\frac{1}{N}z(t)^\top e_j^m \\ &:= \left(R_{z\varphi_j}\right)^{-1} R_{ze_j^m}.\end{aligned} \tag{VI.11}$$

This makes it possible to learn $\vec{\theta}_j$ in a trial-to-trial fashion using the measured error, measured output and the IVs.

C. Estimation of Optimal IVs

The optimal IVs, z_{opt}, are defined as the IVs which result in minimal variance of the FF parameters. The optimal IVs are equal to φ_j^r, which is equal to the reference induced part of φ_j [12]. φ_j^r can be calculated as:

$$z_{opt} := \varphi_j^r(t) = \Psi(q)\left(C^{fb}(q)+C_j^{ff}(q)\right)^{-1} y_j^r(t), \tag{VI.12}$$

where now, y_j^r is the reference induced output of the system. Since there is no direct access to y_j^r and therefore φ_j^r, an estimate has to be made. An iterative scheme, introducing superscript $< i >$, has been created to compute an estimate of φ_j^r:

$$\begin{aligned}z^{<i>}(t) &= \hat{\varphi}_j^r(t) \\ &:= \Psi(q)\left(C^{fb}(q)+C^{ff,<i>}(q,\hat{\theta}_\Delta^{<i-1>})\right)^{-1} r(t).\end{aligned} \tag{VI.13}$$

Herein, the $\hat{\theta}_\Delta^{<i-1>}$ is updated as follows:

$$\begin{aligned}\hat{\theta}_\Delta^{<i>} &= \left(\frac{1}{N}z^{<i>}(t)^\top\varphi_j(t)\right)^{-1}\frac{1}{N}z^{<i>}(t)^\top e_j^m(t) \\ &:= \left(R_{z\varphi_j}^{<i>}\right)^{-1} R_{ze_j^m}^{<i>},\end{aligned} \tag{VI.14}$$

with φ_j from (VI.8).

All the equations above can be combined to update the FF signal per trial and computing an estimate of the IVs and $\hat{\theta}_\Delta$, which can be seen in Algorithm 2.

D. Accuracy Analysis of Optimal IVs

The covariance of the estimated FF parameters can be calculated, to evaluate the accuracy of the estimation. To assess the accuracy, the covariance matrix P_{IV} is defined as:

$$\sqrt{N}\left(\hat{\theta}_\Delta - \bar{\theta}_\Delta\right) \sim \mathcal{N}\left(0, P_{IV}\right), \tag{VI.15}$$

where $\bar{\theta}_\Delta$ is the asymptotic parameter estimate, specifically:

$$e_j^m(t) = \left(\varphi_j(t)\right)^\top \bar{\theta}_\Delta - e_j^w(t), \tag{VI.16}$$

where $e_j^w(t)$ is the error induced due to measurement noise in trial j. When taking n_z, $L(q)$ and W equal to (VI.10), the covariance matrix P_{IV} is equal to [12]:

$$P_{IV} = R_{z\varphi_j}^{-1}\lambda_w^2\mathbb{E}\Big[z(t)\,z^\top(t)\Big]R_{z\varphi^j}^{-\top}, \tag{VI.17}$$

with $R_{z\varphi^j}$ from (VI.14). The λ_w^2 is equal to the variance of the zero-mean white noise w acting on the output.

When using (VI.12) and (VI.13), the covariance matrix related to the FF parameters can be calculated. Furthermore,

Algorithm 2: IV FF control algorithm, with estimated optimal IVs.

Input: N_{trials}, N_{it}, $\Psi(q)$, $r(t)$, $C^{fb}(q)$
Output: Converged FF parameters $\vec{\theta}_{N_{trials}}$

Set $\vec{\theta}_1 \leftarrow \vec{0}$;
for $j = 1$ **to** $N_{trials} - 1$ **do**
  Set $f_j(t) \leftarrow \Psi(q)\vec{\theta}_j r(t)$;
  Run system with f_j, r and C^{fb}. Measure y_j^m ;
  $\varphi_j(t) \leftarrow \Psi(q)\left(C^{fb}(q) + C_j^{ff}(q)\right)^{-1} y_j^m(t)$;
  for $< i >= 1$ **to** N_{it} **do**
    Calculate $z^{<i>}(t)$, see (VI.13);
    Calculate $\hat{R}_{z\varphi_j}^{<i>}$ and $\hat{R}_{ze_j^m}^{<i>}$;
    Calculate $\hat{\theta}_{\Delta}^{<i>}$, using (VI.14);
  Update FF parameters with $\vec{\theta}_{j+1} \leftarrow \vec{\theta}_j + \hat{\theta}_{\Delta}^{<N_{it}>}$;
  Update FF controller with $C_{j+1}^{ff} \leftarrow \Psi(q)\vec{\theta}_{j+1}$;
return $\vec{\theta}_{N_{trials}}$

the covariance matrix P_{IV} can be used as initial guess for optimizing the σ_n^2 hyperparameter of the GP, seen in Section III-E, provided an estimation of the variance of the noise present in the output, λ_w^2, is known.

E. Extending IV to MIMO Systems

Since IV FF control requires the SISO commutative property [12], the algorithm is constructed using multiple independent SISO structures to accommodate MIMO systems. Algorithm 2 shows the algorithm for SISO IV but can easily be extended to MIMO systems, as can be seen in Algorithm 4 in Appendix A. For each direction, the FF force is calculated as:

$$f_{j,dir}(t) = \Psi_{dir}(q)\vec{\theta}_{j,dir} r_{dir}(t). \tag{VI.18}$$

Example 3. Consider a FF force for the x axis is parameterized as:

$$f_{j,x}(t) = \theta_{j,1}\ddot{r}_x(t) + \theta_{j,2}\ddot{r}_y(t). \tag{VI.19}$$

The term r_{dir} in (VI.18), using this example, is equal to:

$$r_{dir}(t) = \begin{bmatrix} r_x(t) & r_y(t) \end{bmatrix}.$$

To avoid singularity of the inversion in (VI.14), (VI.8) is changed to:

$$\varphi_{j,dir}(t) = \Psi_{dir}(q)\Big(C_{dir}^{fb}(q) + C_{j,dir}^{ff}(q)\Big)^{-1} y_{j,dir}^m(t), \tag{VI.20}$$

where $y_{j,dir}^m(t)$, for the example FF parameterization in (VI.19), is equal to:

$$y_{j,x}^m(t) = \begin{bmatrix} y_{j,x}^m(t) & y_{j,y}^m(t) \end{bmatrix}.$$

This will influence the value of $z(t)$ and will therefore affect the convergence speed and accuracy of the FF parameter estimates. However, using sub-optimal IVs will still result in unbiased FF parameter estimates, as long as $z(t)$ is uncorrelated with the error due to measurement noise, as is shown in [11]. ■

The matrices $y_{j,dir}^m$ and r_{dir} together with Algorithm 4 enable IV based FF control to be applied to MIMO systems using multiple SISO structures. However, this does not guarantee convergence in trial domain or in the estimation of the optimal IVs for MIMO systems. Furthermore, many non-linear BF cannot be incorporated in IV FF control since the BFs need to be translated into $\mathbb{R}(q)$.

VII. Integration

This section shows a framework for modelling position-dependent FF parameters, by combining FF parameter learning, GP modelling and MI optimization seen in respectively Sections V or VI, III and IV. This section thereby constitutes contribution C3 and will describe the method such that the individual solutions to P1, P2 and P3 can be combined.

First, some design choices are made. Second the training data set is defined. Third, the greedy sequential MI optimization design choices are elaborated upon. Finally, the complete framework is encapsulated.

A. Design Choices

Certain design choices have been made for the framework. The FF parameters are determined using ILCBF as specified in Section V, since ILCBF is able to cope with MIMO systems and can have non-linear BF, which IV is not capable of. The covariance function which is used is the RBF kernel, seen in (III.7). This covariance function is chosen since it is widely used and is a suitable choice for modelling position-dependent FF parameters. This can be explained since the RBF kernel assigns a high similarity to FF parameters close to each other and a low similarity to FF parameters far away from each other, which is expected. Each FF parameter is modelled using a separate GP, therefore the result of the framework will return n_θ GPs. These design choices enable the framework to be applied to MIMO or SISO systems with multiple (non-linear) basis functions.

B. Defining the Training Data

The training data for the GP, can be defined in several different ways. The FF parameters should be modelled as a function of XY position, meaning the input dimension D is equal to 2. Therefore, the test and training positions, seen in (III.10) and (III.12), are defined as:

$$\begin{aligned} X &= \begin{bmatrix} x_1 & x_2 & \cdots & x_l \\ y_1 & y_2 & \cdots & y_l \end{bmatrix}^\top \in \mathbb{R}^{l\times 2}, \\ X_* &= \begin{bmatrix} x_{*,1} & x_{*,2} & \cdots & x_{*,l_*} \\ y_{*,1} & y_{*,2} & \cdots & y_{*,l_*} \end{bmatrix}^\top \in \mathbb{R}^{l_*\times 2}. \end{aligned} \tag{VII.1}$$

The training positions X are determined using the MI optimization seen in Section IV. For visualization purposes, the test positions X_* are chosen as a fine grid covering the operating range, but can be any single position or vector of positions.

All converged FF parameters are used as training data for the GP. Specifically, the training data $\boldsymbol{\theta}$ seen in (III.6) is now:

$$\boldsymbol{\theta}_i = \begin{bmatrix} \theta_{i,N_{conv}} & \cdots & \theta_{i,N_{trials}} \end{bmatrix}^\top, \tag{VII.2}$$

where N_{conv} is the trial where the FF parameters in ILCBF are converged. This shows there are multiple training outputs $\mathbf{y}$ for each training position X. This is done such that the marginal likelihood optimization can more easily capture the noise variance hyperparameter σ_n^2. Furthermore, this makes the framework more robust to variations of the FF parameters due to trial-varying disturbances and noise realizations.

C. Greedy Sequential MI Optimization

A greedy sequential MI optimization scheme is implemented to pick the training positions X near-optimal. A sequential method has been chosen because the hyperparameters of the GPs are not (exactly) known. In Section IV the procedure for applying a greedy sequential MI optimization can be seen. Since all GPs are trained on the same training positions (*isotopic* data), the MI can only be optimized for one GP. Hence, near-optimality for the training positions is only guaranteed for one GP. Here, the GP of the FF parameter which has the highest contribution to the error signal is chosen to optimize the MI for, which is determined by comparing the values of $S_o(z)G_{\mathbf{x}}(z)\big(\Psi_i\theta_i\big)$. However, because the data is isotopic, the near-optimal training positions for different GPs do not vary much. Using this, the greedy sequential MI optimization is used for choosing training positions when identifying multiple FF parameters as a function of position.

D. Complete Framework

The complete framework for position-dependent GP FF is a combination of ILCBF, MI optimization and a GP regression. Fig. 7 visually illustrates the sequence how these methods are combined. Fig. 7 first shows an initialization phase of the framework, since the MI algorithm requires an initial hyperparameter set. The initial hyperparameter set can be initialized random or based on prior knowledge. Subsequently, $k-1$ repetitions of ILCBF, optimization of hyperparameters and picking of a new training position using MI are done. The value for k is the total amount of training positions and can either be a set constant, or the framework can be stopped when the MI does not change significantly anymore.

Algorithm 5 seen in Appendix A shows the pseudocode of the entire learning algorithm over the operating range of a machine, including the identification of the FF parameters using ILCBF, hyperparameter optimization, greedy sequential MI optimization and GP regression. Using the obtained GPs, FF parameters can be determined at test positions X_*, using (III.15). Next, the framework will be applied to a computer simulation.

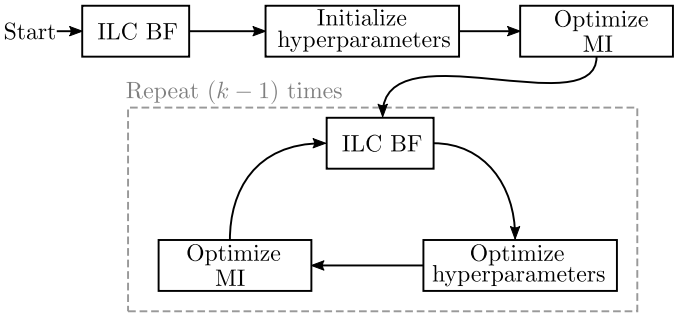

Fig. 7: Illustration of the complete framework, which is executed before normal operation, to identify FF parameters, choosing training positions with MI and make a GP regression of the identified FF parameters.

VIII. Simulation Results

The framework presented in Section VII is implemented in a computer simulation of the XYZ stage from the wirebonder seen in Fig. 16. The model of the XYZ stage is derived in [16] and only the X and Y stages are actuated. The initial position for the X and Y stages can be varied and the reference is designed according to Assumption 1. This section will describe the application and results of the framework described in Section VII for the computer simulation.

The identification of FF parameters with ILC is presented in Section VIII-A. In Section VIII-B the GP is trained and it is ended with a performance comparison with other position-dependent FF methods in Section VIII-C.

A. ILCBF

The ILCBF algorithm seen in Section V has been applied to the computer simulation. The BF and FF parameters are selected using velocity, acceleration and snap FF and are:

$$\Psi\big(r(t)\big) = \begin{bmatrix} \dot{r}_x(t) & \ddot{r}_x(t) & r_x^{(4)}(t) & 0 & 0 & 0 \\ 0 & 0 & 0 & \dot{r}_y(t) & \ddot{r}_y(t) & r_y^{(4)}(t) \end{bmatrix},$$
$$\vec{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}^\top. \tag{VIII.1}$$

Fig. 8 clearly shows the convergent properties of the ILCBF algorithm.

The convergent properties are further illustrated by looking at the time domain error signal for both the first and last trial, seen in Fig. 9, which demonstrates the substantial error reduction ILCBF achieves.

B. Training the GP

To create a GP regression of the FF parameters as a function of position, ILCBF has been carried out on several training positions. The training positions, both based on a grid and MI optimization, can be seen in Fig. 10.

Position-dependent GP modelled FF parameters can be made with the procedure in Section VII. The advantage of using MI optimal training positions compared with grid based training positions can be seen by comparing the posterior

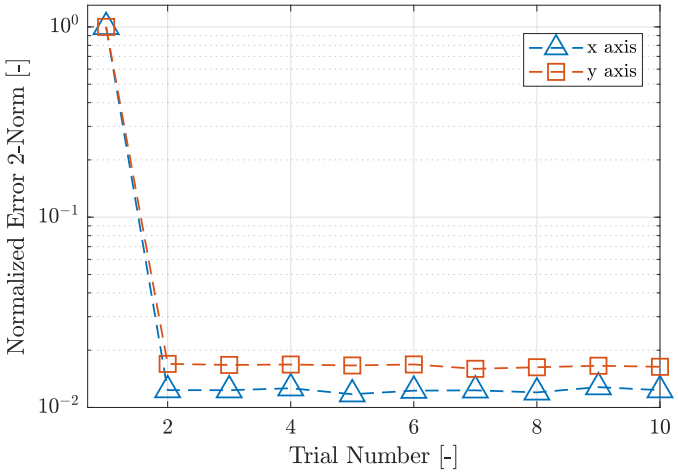

Fig. 8: Normalized convergence results in terms of error 2-norm for x and y axes at center position.

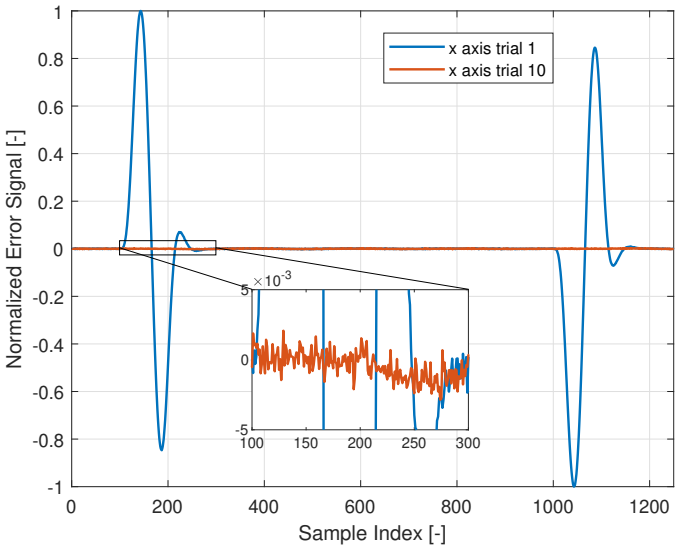

Fig. 9: Normalized time domain error for the first and the last trial of ILCBF at the center position.

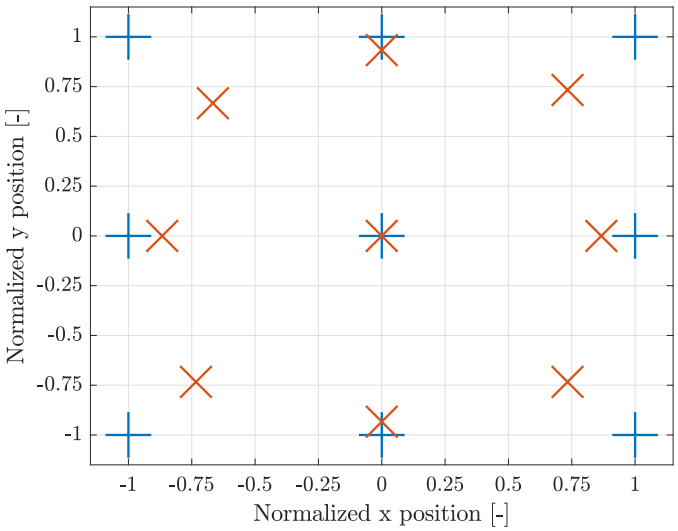

Fig. 10: Normalized training positions used to measure the FF parameters with ILC using BF for a computer simulation, both using a grid (+) and MI based positions (×).

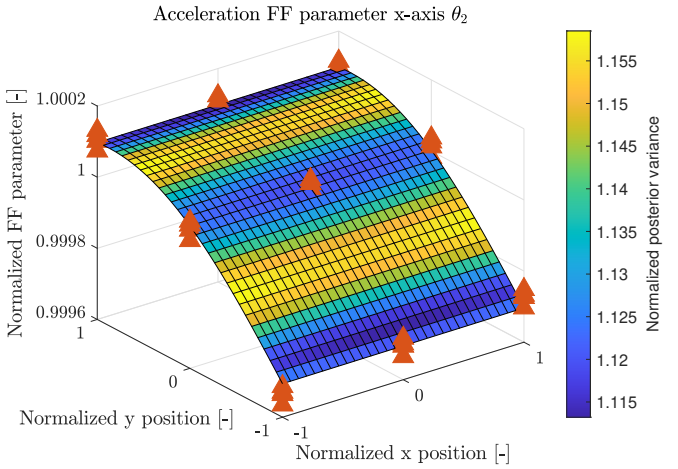

Fig. 11: Normalized GP regression of the acceleration FF parameter for the x axis. The color indicates the posterior variance of the GP, which is normalized using the same factor as in Fig. 12. The FF parameters used as training data (▲) are determined using ILCBF. The training positions are an equispaced grid.

variances of the acceleration FF parameter GP for the x axis, seen in Fig. 11 and Fig. 12.

When comparing the posterior variances, Fig. 12 shows a much lower posterior variance compared with Fig. 11, in both the measured and unmeasured space. This indicates the GP is more confident in estimating FF parameters. From now on, the MI optimal training positions are used. The GP regression of the acceleration FF parameter for the y axis can be seen in Fig. 13.

The regressions of the FF parameters both show how the GP models the FF parameters as a function of position and does this with a relatively small set of training data.

C. Performance Comparison

To validate the framework for the computer simulation, three methods for position-dependent FF have been performed on several test positions to compare the framework with. The test positions are chosen as an equispaced four by four grid covering the operating range. The three methods for FF on these positions which are executed are:

- M1: FF parameters determined in the **center** position of the machine, i.e. not taking position-dependent dynamics into account for FF control,
- M2: FF parameters determined using a **nearest-neighbour** search, meaning the FF parameters of a training position closest to the test position,
- M3: FF parameters determined from the **GP** regressions.

The error 2-norm for the different methods and test positions can be seen in Fig. 14.

Fig. 14 clearly shows applying either the nearest-neighbour or GP method can improve the error 2-norm in certain areas

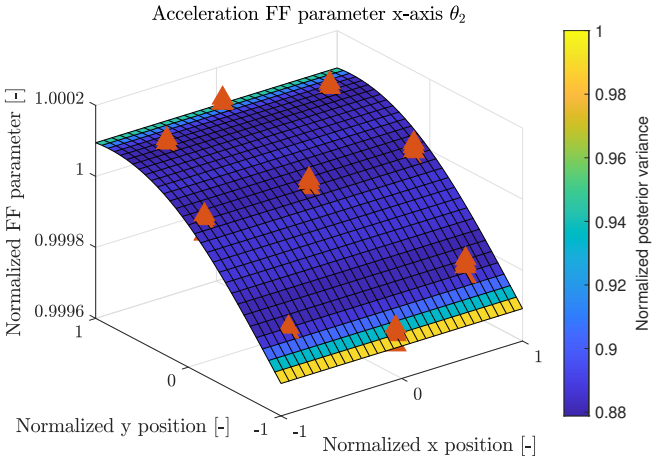

Fig. 12: Normalized GP regression of the acceleration FF parameter. The color indicates the posterior variance of the GP. The FF parameters used as training data (▲) are determined using ILCBF. The training positions X are determined near-optimal using MI. Note that the posterior variance is lower compared with the grid based training positions, as seen in Fig. 11.

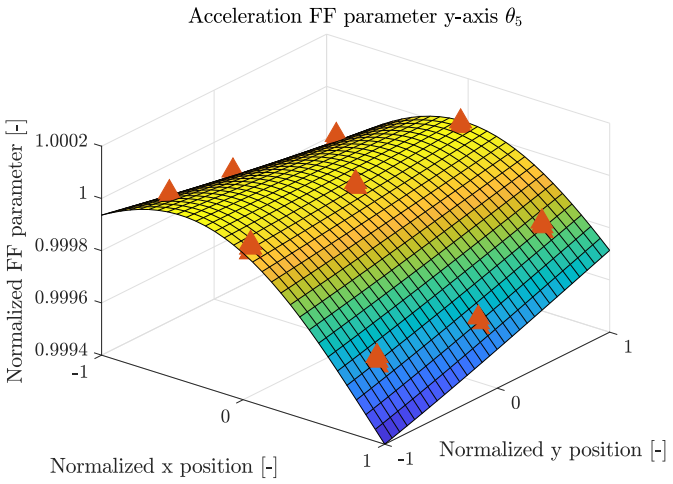

Fig. 13: Normalized GP regression of the acceleration FF parameter of the y axis. The x and y axes are the machine position. The FF parameters used as training data (▲) are determined using ILCBF.

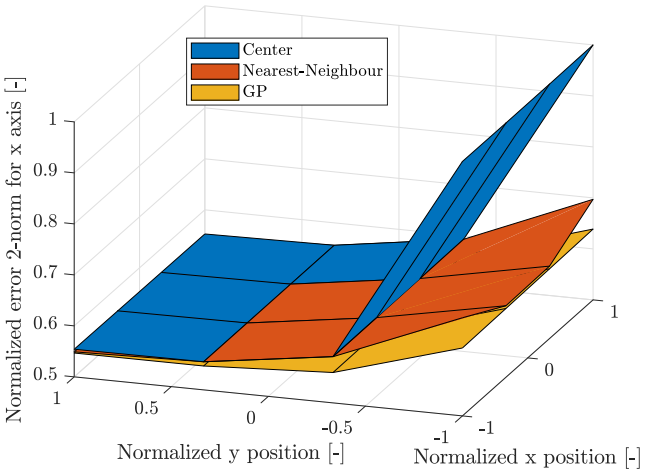

Fig. 14: Normalized error 2-norm for the x axis when using different methods for picking the FF parameters. This shows the GP FF method outperforms the other methods in all positions.

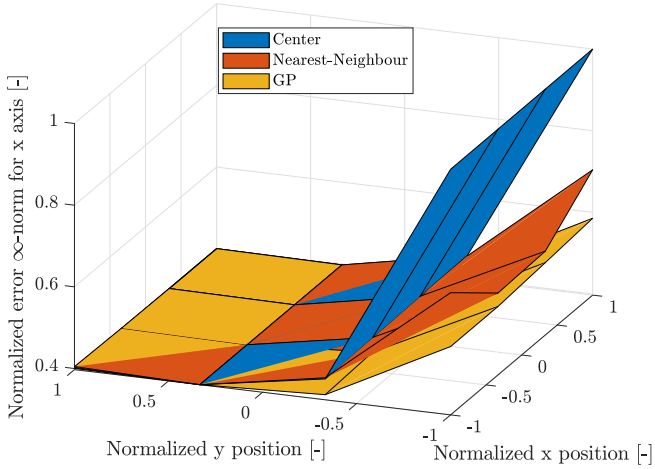

Fig. 15: Normalized error ∞-norm for the x axis when using different methods for picking the FF parameters. This shows the GP FF method outperforms the other methods in all positions.

of the machine drastically. For negative y positions, the error 2-norm can be reduced up to a factor 2 by utilizing a GP FF compared with the center approach. The GP method has similar performance compared to nearest-neighbour, but still has a lower error 2-norm for all test positions. A similar plot can be made for the y axis, however, since there is no significant difference for the different FF methods, it is not presented here.

The error ∞-norm, seen in Fig. 15, shows a similar situation, the GP method outperforms both the center and nearest-neighbour methods. Again, when observing the error ∞-norm of the y axis, there is no significant difference between the three methods.

To conclude, a significant performance gain can be achieved for the x axis in a computer simulation when using the GP FF method, compared with the center or nearest-neighbour method. This is caused by the position-dependent dynamics for this axis. In addition, the y axis does not contain significant position-dependent effects in the computer simulation, shown by the similar performance of the three methods.

IX. Experimental Results

The framework presented in Section VII is implemented on the XYZ stage from the commercial wirebonder seen in Fig. 16. This section will describe the application of the framework and validation thereof, by using FF parameters determined with the GP on several test positions of the machine.

Fig. 16: AB383 wirebonder designed and built by ASM PT. Photo courtesy of ASM PT.

Firstly, in Section IX-A, the experimental setup will be described. Secondly, Section IX-B shows the application and results of applying ILCBF on the setup. This is followed by the training positions determined with MI in Section IX-C. The GP is trained in Section IX-D. Finally, a performance comparison with other FF methods is done in Section IX-E

A. Experimental Setup

The experimental setup is a commercial wirebonder from ASM PT, model AB383, seen in Fig. 16. The wirebonder consists of an XYZ-stage, where the X and Y stages move perpendicular with respect to each other using linear motors and the Z stage moves vertically and is actuated with a rotating motor and a pivoting mechanism. On this setup, the initial position for both the X and Y stage can be varied and the reference is designed according to Assumption 1. The machine has interesting position-dependent effects, such as changing motor force constants and is therefore a good example for this work.

B. ILCBF

FF parameters on a single position $\vec{\theta}$ are learned on the wirebonder using the ILCBF procedure as described in Section V. The BF is based on the dynamic model and determined in [32, 33] and is equal to:

$$\Psi(r) = \begin{bmatrix} \ddot{r}_x & \dot{r}_x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddot{r}_y & \dot{r}_y & \psi_{c,y} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \psi_{c,z} & r_z^{(4)} & \ddot{r}_z & \dot{r}_z \end{bmatrix}, \quad \text{(IX.1)}$$

where (t) has been left out for brevity and:

$$\begin{aligned} \psi_{c,y} &= \begin{bmatrix} \ddot{r}_z \sin(r_z) + \dot{r}_z^2 \cos(r_z) & \ddot{r}_z \cos(r_z) - \dot{r}_z^2 \sin(r_z) \end{bmatrix}, \\ \psi_{c,z} &= \begin{bmatrix} \ddot{r}_y \sin(r_z) & \ddot{r}_y \cos(r_z) \end{bmatrix}. \end{aligned} \quad \text{(IX.2)}$$

The BF can be combined with the following FF parameters to parameterize the FF force:

$$\vec{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 & \theta_8 & \theta_9 \end{bmatrix}^\top. \quad \text{(IX.3)}$$

The reference used is a 7th order motion profile which can be seen in Fig. 17.

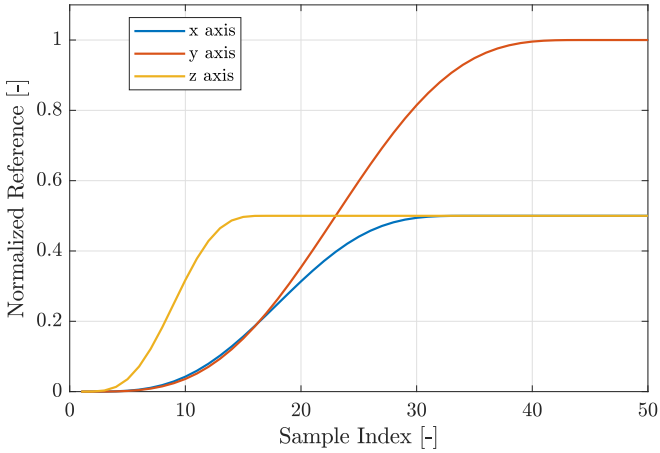

Fig. 17: Normalized reference profile for x, y and z axes used in the experiments.

The convergence result in terms of the error 2-norm can be seen in Fig. 18.

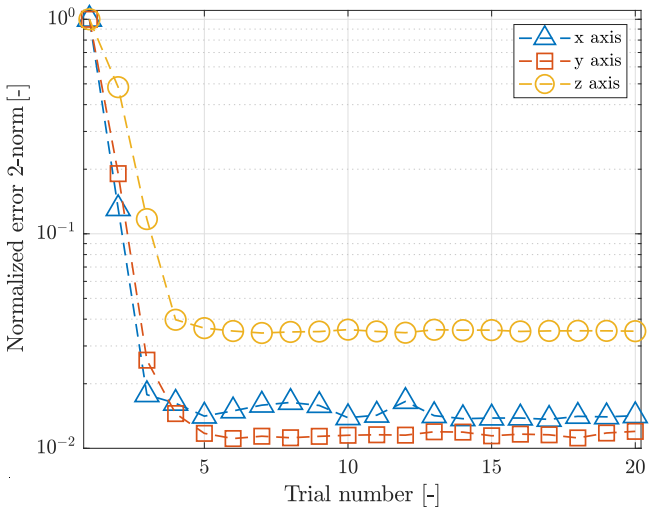

Fig. 18: Normalized error 2-norm over trials when performing ILCBF in the center position of the machine.

Fig. 18 shows the ILCBF scheme is convergent and significantly reduces the tracking error compared with feedback only.

C. Near-optimal Training Positions

The ILCBF is performed on several positions of the wirebonder, which are determined using MI optimization. A greedy sequential algorithm is used to determine the positions, presented in Section IV and seen in Algorithm 3. The initial hyperparameter set is determined using prior knowledge of the hyperparameters. The prior knowledge of the hyperparameters is acquired by performing the framework on arbitrary training positions and optimizing the marginal likelihood. The resulting training positions from the MI scheme can be seen in Fig. 19.

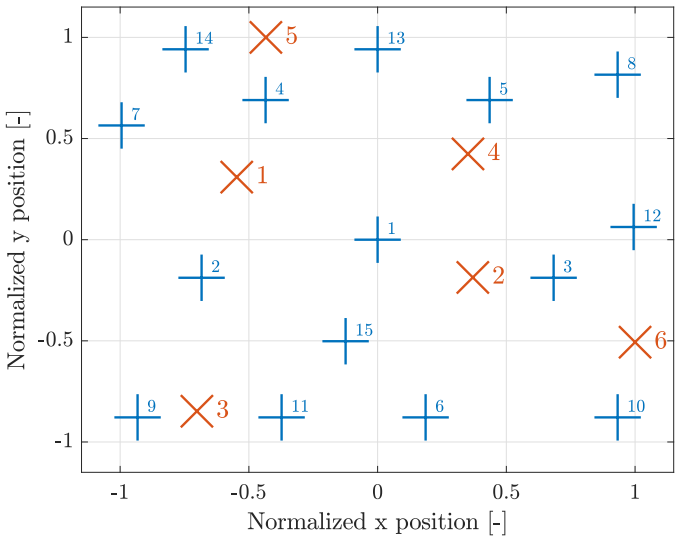

Fig. 19: Normalized training positions (+), determined with MI optimization, and test positions (×), arbitrarily chosen using interpolation and extrapolation with both high and low posterior variance of the GP, used in the experiments.

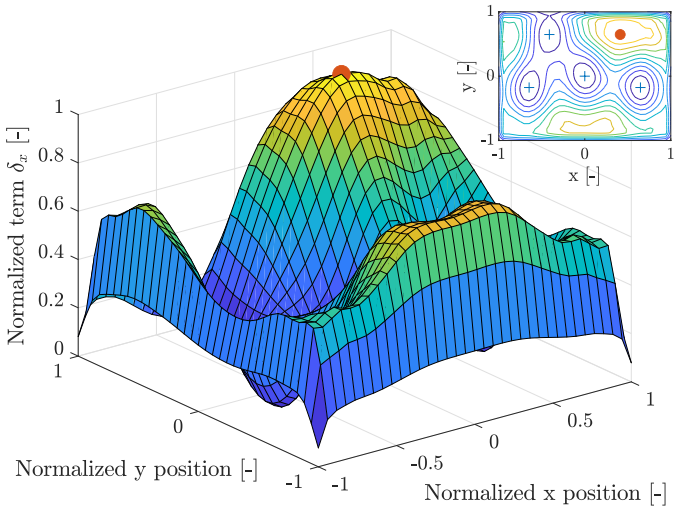

Fig. 20: Term δ_x seen in (IV.7), plotted as a function of position after training position 4. The maximum of the function (●) indicates the next training position. Training positions 1 to 4 (+) can be easily recognised in the upper right contour plot, as the local minima. An animation of an example (a priori) MI optimization can be seen at http://bit.ly/MIOptimizationGPFF.

An example of the term δ_x after training position 4, can be seen in Fig. 20. Fig. 20 shows that the next training position will be roughly equal to $(x, y) \approx (0.41, 0.65)$, which is equal to training position 5 seen in Fig. 19.

The maximization of the term δ_x is done after each training position and will result in the near-optimal training positions seen in Fig. 19.

D. Training the GP

On all training positions seen in Fig. 19, the FF parameters $\vec{\theta}$ are determined and together with the training positions, are combined to train a GP. The training data is specified as explained in Section VII and the hyperparameters are optimized using marginal likelihood optimization. A GP regression of the acceleration FF parameter for the x and y axes can be seen in respectively Fig. 21 and Fig. 22.

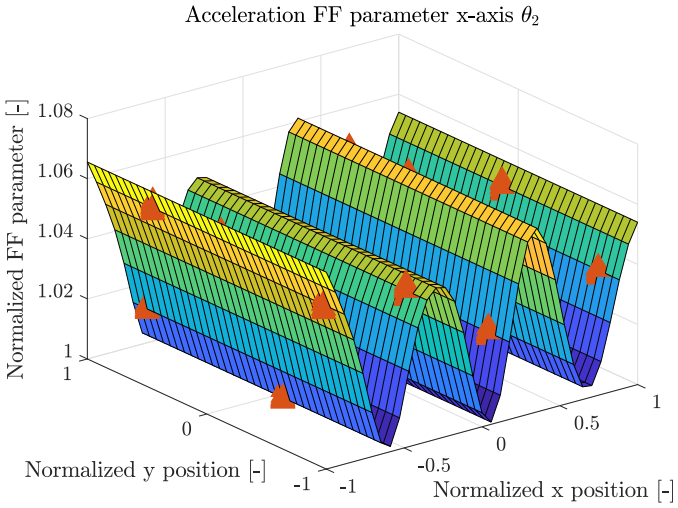

Fig. 21: Normalized GP regression of θ_2 for the experimental setup. The FF parameters (▲) are identified with ILC using BF on the training positions seen in Fig. 19.

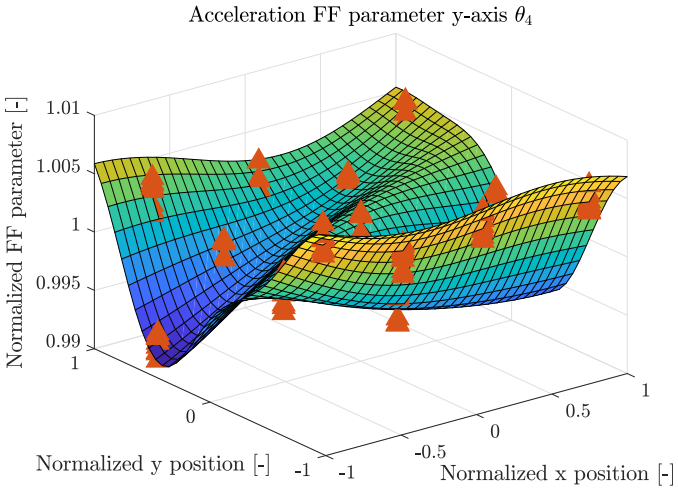

Fig. 22: Normalized GP regression of θ_4 for the experimental setup. The FF parameters (▲) are identified with ILC using BF on the training positions seen in Fig. 19.

This is done for all the other FF parameters as well and can be seen in Appendix E. The acceleration FF parameter for the x axis in Fig. 21 shows an interesting behaviour which is periodic in the x direction. The physical interpretation is straightforward, since the period distance in x direction of the GP regression is roughly equal to the magnet pitch of the linear motor. This suggests a periodic covariance function might be able to model this FF parameter accurately, which can be seen in Appendix D. Both Fig. 21 and Fig. 22 show that the wirebonder has significant position-dependent effects and should therefore be accounted for to achieve best performance.

E. Performance Comparison

To validate the framework for the experimental setup, three methods for position-dependent FF are performed on several test positions to compare the framework with. The test positions can be seen in Fig. 19. The three methods for FF on these positions which are executed are:

- M1: FF parameters determined in the **center** position of the machine, i.e. not taking position-dependent dynamics into account for FF control,
- M2: FF parameters determined using a **nearest-neighbour** search, meaning the FF parameters of a training position closest to the test position,
- M3: FF parameters determined from the **GP** regressions,
- R: **ILCBF** to serve as a reference frame to compare the three methods.

In the test positions, the error signals and norms are compared to evaluate performance for each method. The error signals in test position 2 can be seen in Figs. 23, 24 and 25.

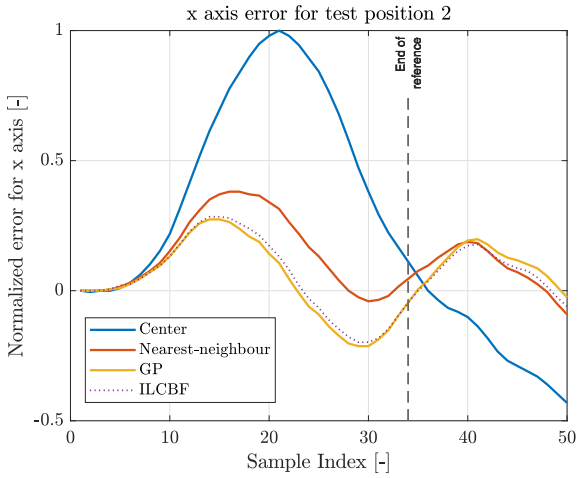

Fig. 23: Normalized error signals of the x axis for the three position-dependent FF methods and ILCBF on test position 2, as seen in Fig. 19.

Figs. 23 and 24 show the GP can reduce the error compared with the center and nearest-neighbour methods by using a GP to model the FF parameters for the x and y axes. For the z axis the error, seen in Fig. 25, remains relatively constant for all methods, even when using the validation data, where the FF parameters are determined at the test position using ILCBF.

The error 2-norm for all test positions can be seen in Fig. 26. This shows the GP method has a lower error 2-norm for most positions in the operating range of the machine, compared with the center or nearest-neighbour methods. For several test positions, such as test position 1, the center and nearest-neighbour method perform similarly compared to the GP method. This can be explained since these are located roughly one magnet pitch away from the center position, making the position-dependency negligible, as is seen in Fig. 21.

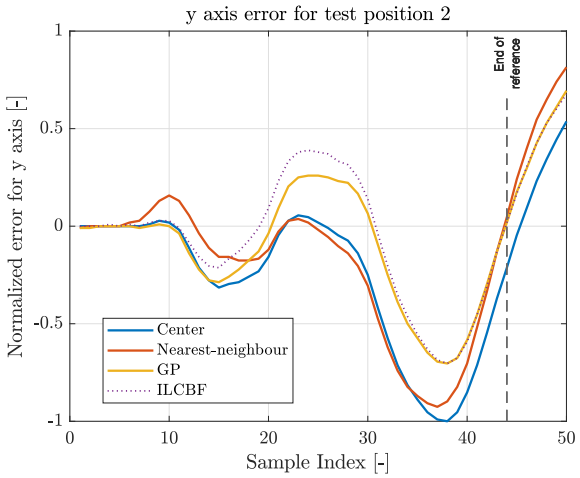

Fig. 24: Normalized error signals of the y axis for the three position-dependent FF methods and ILCBF on test position 2, as seen in Fig. 19.

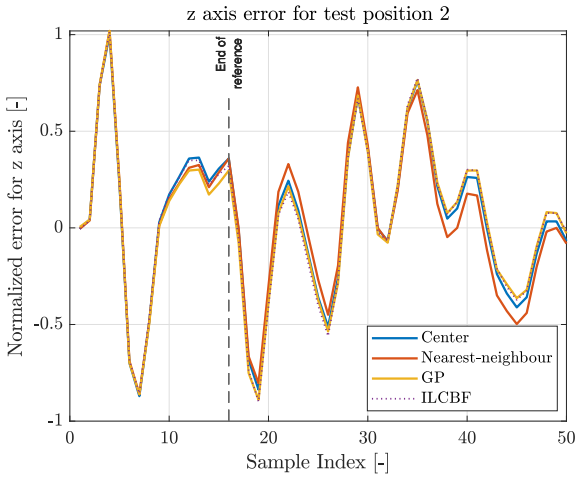

Fig. 25: Normalized error signals of the z axis for the three position-dependent FF methods and ILCBF on test position 2, as seen in Fig. 19.

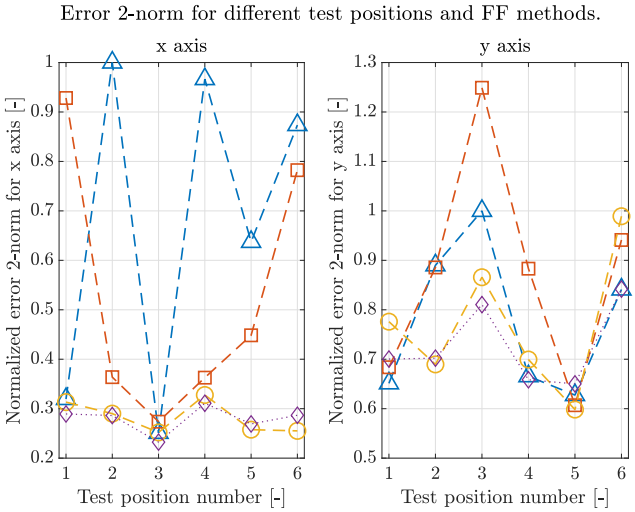

Fig. 26: Error 2-norm for FF methods: center ( ), nearest-neighbour ( ), GP ( ) and ILCBF ( ) per test position, as seen in Fig. 19. Note that the GP outperforms the center and nearest-neighbour methods for most test positions.

The error ∞-norm, measured only during the time where the reference velocity is not equal to zero, i.e. the dynamic part, can be seen for the x and y axes in Fig. 27. Similar conclusions as from the error 2-norm can be drawn with Fig. 27.

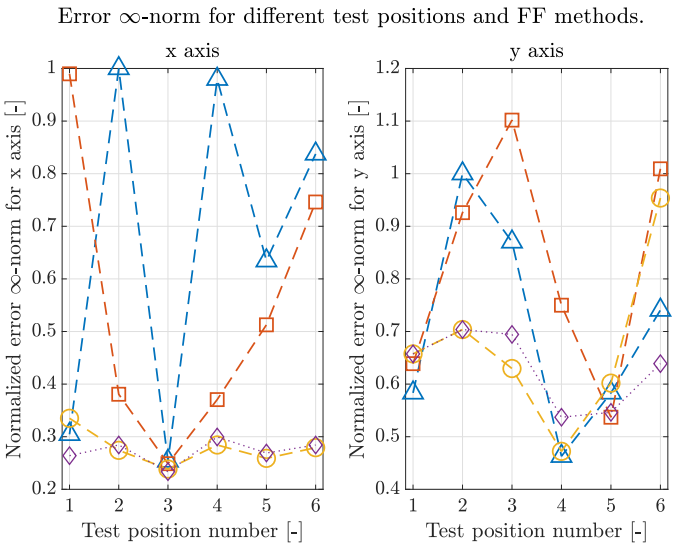

Fig. 27: Error ∞-norm for FF methods: center ( ), nearest-neighbour ( ), GP ( ) and ILCBF ( ) per test position, as seen in Fig. 19. Note that the GP outperforms the center and nearest-neighbour methods for most test positions.

Additionally, the sum of the error 2-norm for all directions is computed, seen in Fig. 28, which shows that the GP has improved performance compared with the center and nearest-neighbour method in all test positions, with the exception of test position 1. However, the small performance difference between the center and GP method can be explained due to trial varying disturbances.

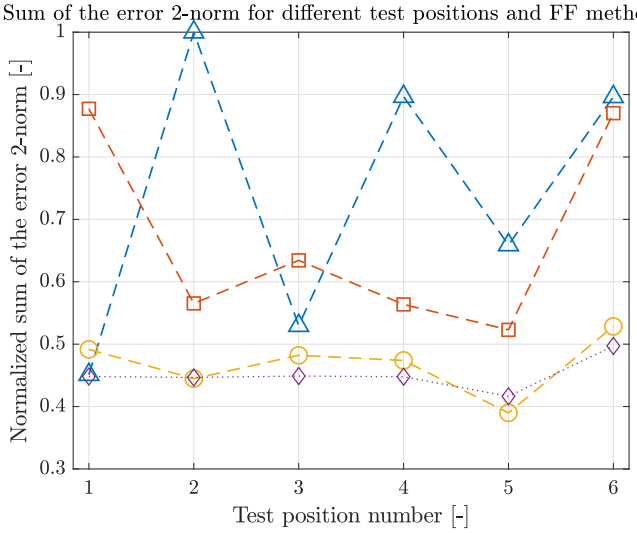

Fig. 28: Sum of the error 2-norm of all axes for FF methods: center ( ), nearest-neighbour ( ), GP ( ) and ILCBF ( ) per test position, as seen in Fig. 19. This shows, with the exception of test position 1, that the GP outperforms both the center and nearest-neighbour methods.

Both the error norms for x and y axes and the sum of the error 2-norm for all directions show that the GP method for determining the FF parameters achieves similar performance as ILCBF. For the center and nearest-neighbour methods, several positions have considerable higher error 2- and ∞-norms than the GP method, showing the superior performance achievable when using the GP position-dependent FF method.

X. Conclusions

In this work, position-dependent feedforward control is developed using Gaussian processes which model feedforward parameters as a function of position. Gaussian processes are used since they are non-parametric and therefore model non-linear and black box effects accurately. Additionally, the combined effect of position-dependent dynamics and motor force constant can be compensated for without prior knowledge of both, based on data. Furthermore, Gaussian processes enable the use of mutual information optimization to determine near-optimal training positions. The framework is tested in a computer simulation and experiments on a commercial wirebonder. Both show that a Gaussian process models the feedforward parameters such that it improves performance over the entire operating range.

Future research on this topic should be directed at investigating different kernel choices and the impact on the performance thereof. Secondly, prior knowledge of the FF parameters should be incorporated in the kernel choice, hyperparameters and mean function. This might make the algorithm more robust to less training data and could improve performance. Thirdly, the mutual information could be used to determine the amount of training positions necessary. Lastly, extrapolation capabilities to other references and to references that are not negligible with respect to the position dependency, should be investigated.

XI. Acknowledgements

I would like to thank my day-to-day supervisor Maurice Poot for his support during my MSc thesis. His ideas and guidance made this project enjoyable and achievable. Furthermore, I want to thank Kelvin Kai Wa Yan and Dragan Kostić for both their support during the experiments at ASM PT and the fruitful discussions we had about my project. Also, I would like to thank Robin van Es from ASM PT, who started the discussion about the results and interpretation of the experiments. Moreover, the facilitation of the project by Jim Portegies is greatly appreciated. Lastly, I want to thank Tom Oomen, who always looked critically at my work and therefore made the result of this thesis possible.

References

[1] Tom Oomen. "Learning in machines". In: *Mikroniek* 6 (2018), pp. 5–11.

[2] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. "Bettering operation of Robots by learning". In: *Journal of Robotic Systems* 1.2 (1984), pp. 123–140.

[3] Nard Strijbosch, Tom Oomen, and Lennart Blanken. "Frequency domain design of iterative learning control and repetitive control for complex motion systems". In: *Samcon* 2 (2018), pp. 1–2.

[4] Frank Boeren et al. "Frequency-Domain ILC Approach for Repeating and Varying Tasks: With Application to Semiconductor Bonding Equipment". In: *IEEE/ASME Transactions on Mechatronics* 21.6 (Dec. 2016), pp. 2716–2727.

[5] J. van de Wijdeven and O.H. H. Bosgra. "Using basis functions in iterative learning control: Analysis and design theory". In: *International Journal of Control* 83.4 (Apr. 2010), pp. 661–675.

[6] Joost Bolder et al. "Using iterative learning control with basis functions to compensate medium deformation in a wide-format inkjet printer". In: *Mechatronics* 24.8 (Dec. 2014), pp. 944–953.

[7] Xuemei Gao and Sandipan Mishra. "An iterative learning control algorithm for portability between trajectories". In: *Proceedings of the American Control Conference*. 2014.

[8] Jurgen Van Zundert, Joost Bolder, and Tom Oomen. "Iterative Learning Control for varying tasks: Achieving optimality for rational basis functions". In: *Proceedings of the American Control Conference* 2015-July.1 (July 2015), pp. 3570–3575.

[9] Lennart Blanken et al. "Flexible ILC: Towards a Convex Approach for Non-Causal Rational Basis Functions". In: *IFAC* (2017).

[10] Joost Bolder et al. "Enhancing flatbed printer accuracy and throughput: Optimal rational feedforward controller tuning via iterative learning control". In: *IEEE Transactions on Industrial Electronics* (2017).

[11] Frank Boeren, Tom Oomen, and Maarten Steinbuch. "Iterative motion feedforward tuning: A data-driven approach based on instrumental variable identification". In: *Control Engineering Practice* 37 (2015), pp. 11–19.

[12] Frank Boeren, Dennis Bruijnen, and Tom Oomen. "Enhancing feedforward controller tuning via instrumental variables: with application to nanopositioning". In: *International Journal of Control* 90.4 (2017), pp. 746–764.

[13] Robbert Voorhoeve et al. "Identifying Position-Dependent Mechanical Systems: A Modal Approach Applied to a Flexible Wafer Stage". In: *IEEE Transactions on Control Systems Technology* (2020), pp. 1–13.

[14] J M M Rovers, J W Jansen, and Prof E A Lomonova. "Performance measurements of the Double Layer Planar Motor". In: (), pp. 2–5.

[15] B. Denkena, J. Friederichs, and J. Fuchs. "Design and analysis of a 2-DOF synchronous planar drive for machine tools". In: *Production Engineering* 9.1 (2014), pp. 125–132.

[16] S Beer. *Data-driven Axes Decoupling and Motion Control of Bonding Machines (MSc Thesis)*. 2019.

[17] H.J.B. van Deursen. *Multivariable Iterative Learning Control with basis functions : performance enhancement of an industrial flatbed printer (MSc Thesis)*. 2015.

[18] Robin de Rozario, Tom Oomen, and Maarten Steinbuch. "Iterative Learning Control and feedforward for LPV systems: Applied to a position-dependent motion system". In: *2017 American Control Conference (ACC)*. 2017. IEEE, May 2017, pp. 3518–3523.

[19] Carl Edward Rasmussen. "Gaussian Processes in machine learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3176 (2004), pp. 63–71.

[20] Jasper Snoek, Hugo Larochelle, and Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems 25* (2012), pp. 2951–2959.

[21] Andreas Krause, Ajit Singh, and Carlos Guestrin. "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies". In: *Journal of Machine Learning Research* 9 (2008), pp. 235–284.

[22] Burr Settles. "Active Learning Literature Survey". In: *Machine Learning* (2010).

[23] Robin de Rozario and Tom Oomen. "Frequency response function identification of periodically scheduled linear parameter-varying systems". In: *Mechanical Systems and Signal Processing* 148 (2021), pp. 1–19.

[24] M.R. Hamers. *Actuation Principles of Permanent Magnet Synchronous Planar Motors A Literature Survey (DCT Report)*. December. Eindhoven University of Technology, 2005.

[25] Andreas Krause and Carlos Guestrin. "Nonmyopic active learning of Gaussian processes: An exploration- exploitation approach". In: *ACM International Conference Proceeding Series* 227.May (2007), pp. 449–456.

[26] David Kristjanson Duvenaud. "Automatic Model Construction with Gaussian Processes". PhD thesis. University of Cambridge, 2014.

[27] Hildo Bijl. "Gaussian Process Regression Techniques". PhD thesis. Delft University of Technology, 2018.

[28] W.F. Caselton and J.V. Zidek. "Optimal monitoring network designs". In: *Statistics & Probability Letters* 2.4 (Aug. 1984), pp. 223–227.

[29] N. A. C. Cressie. "Statistics for Spatial Data, Revised Edition." In: *Biometrics* (1994).

[30] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* (1979).

[31] Jurgen Van Zundert, Joost Bolder, and Tom Oomen. "Optimality and flexibility in Iterative Learning Control for varying tasks". In: *Automatica* 67 (2016), pp. 295–302.

[32] R.J.P.M Vromans. *Basis Function Iterative Learning Control of Bonding Machines : Bridging Data Driven Learning and Physical Modeling (MSc Thesis)*. 2018, p. 18.

[33] Georgios Maleas. *Multivariable Iterative Learning Control for Flexible Tasks with Application to Semiconductor Bonding Machines (MSc Thesis)*. 2020.

Appendix A
Sequential MI algorithm

An algorithm showing the sequential version of the MI optimization scheme can be found in Algorithm 3.

Algorithm 3: Greedy MI algorithm to determine optimal training positions in a sequential manner.

Input: Initial training data X_1^G and $\mathbf{y}_1$, discretized space $\overline{X}$
Output: Near-optimal training positions X^{MI} and training outputs $\mathbf{y}^{MI}$.

for $i = 1$ **to** $k-1$ **do**
  Optimize hyperparameters with X_i^G and $\mathbf{y}_i$;
  for $x \in \overline{X} \setminus X_i^G$ **do**
    Set $\hat{X} \leftarrow \overline{X} \setminus \left(X_i^G \cup x\right)$;
    Calculate $k(x,x)$, K_y, $K(x, X_i^G)$, $K(x, \hat{X})$ & $K(\hat{X}, \hat{X})$;
    Calculate δ_x with (IV.7);
  Set $x_{seq} \leftarrow \arg\max_x \delta_x$;
  Update $X_{i+1}^G \leftarrow X_i^G \cup x_{seq}$;
  Sample function with training position x_{seq};
  Update $\mathbf{y}_{i+1} \leftarrow \begin{bmatrix}\mathbf{y}_i & f(x_{seq}) + \epsilon\end{bmatrix}$;
return X_k^G and $\mathbf{y}_k$;

APPENDIX B
MIMO IV BASED FF CONTROL ALGORITHM

Algorithm 4 shows an extension of IV to MIMO systems, where the subscript dir is used to denote the input direction. This uses estimated optimal IVs and matrices $y_{j,dir}^m$ and r_{dir} that indicate the direction the basis functions in Ψ are acting.

Algorithm 4: MIMO IV FF control algorithm, based on multiple SISO loops with estimated optimal IVs.

Input: N_{trials}, iterations for estimation of optimal IVs N_{it}, BF Ψ, reference r, FB controller C^{fb}
Output: Converged FF parameters

Set $\vec{\theta}_j \leftarrow \vec{0}$;
for $j = 1$ **to** N_{trials} **do**
  for $dir = 1$ **to** n_i **do**
    Set $f_{j,dir} \leftarrow \Psi_{dir}\vec{\theta}_{j,dir} r_{dir}$;
  Run system with f_j and C^{fb}, measure y_j^m;
  for $dir = 1$ **to** n_i **do**
    Set $C_{dir} \leftarrow \left(C_{dir}^{fb} + C_{j,dir}^{ff}\right)$;
    Set $\varphi_{j,dir} \leftarrow \Psi_{dir} C_{dir}^{-1} y_{j,dir}^m$;
    for $< i >= 1$ **to** N_{it} **do**
      Calculate $z_{dir}^{<i>}$, see (VI.13);
      Calculate $\hat{R}_{z\varphi_j,dir}^{<i>}$ and $\hat{R}_{ze_j^m,dir}^{<i>}$;
      Calculate $\hat{\theta}_{\Delta,dir}^{<i>}$, as (VI.14);
    Update FF parameters and controllers with $\vec{\theta}_{j+1,dir} \leftarrow \vec{\theta}_{j,dir} + \hat{\theta}_{\Delta,dir}^{<N_{it}>}$ and $C_{j+1,dir}^{ff} \leftarrow \Psi_{dir}\vec{\theta}_{j+1,dir}$;
return Converged FF parameters

APPENDIX C
FULL FRAMEWORK ALGORITHM

Pseudocode for applying the full framework, where FF parameters are determined with ILCBF, training positions are determined with MI and the FF parameters are regressed using a GP can be seen in Algorithm 5.

Algorithm 5: Algorithm for GP regression of position dependent FF parameters in x and y direction.

Input: Trials for ILC N_{trials}, maximum measuring positions k, BF Ψ, arbitrary initial position x_1 and reference r for ILC
Output: Position-dependent FF parameter GP regression for the BFs used as input

for $i = 1$ **to** k **do**
  Procedure *ILCBF on position* x_i*:*
    Calculate $\mathbf{Q}$, $\mathbf{L}$;
    $\vec{\theta}_1 \leftarrow \vec{0}$;
    for $j = 1$ **to** N_{trials} **do**
      $f_j \leftarrow \Psi\vec{\theta}_j$;
      Run system with f_j and r ;
      $\vec{\theta}_{j+1} \leftarrow \mathbf{Q}\vec{\theta}_j + \mathbf{L}e_j$;
  Set $\mathbf{y}_i \leftarrow \begin{bmatrix}\mathbf{y}_{i-1} & \boldsymbol{\theta}_i\end{bmatrix}$ with $\boldsymbol{\theta}_i$ from (VII.2);
  Set $X_i \leftarrow \begin{bmatrix}X_{i-1} & x_i\end{bmatrix}$;
  Optimize GP kernel hyperparameters;
  Do GP regression;
  Determine measurement position x_{i+1} using MI;
return GP Regressions

APPENDIX D
INVESTIGATING A PERIODIC KERNEL

Different kernels can be used that the squared exponential kernel to model position-dependent FF parameters. For example, the periodic kernel seen in (III.9) can be used to model the acceleration FF parameter for the x axis, which is previously seen in Fig. 21. It makes sense to use a periodic kernel here, since the FF parameter seems to repeat itself in x direction. A GP regression when using a periodic kernel of the acceleration FF parameter can be seen in Fig. 29.

Interestingly, when using this periodic kernel, the period hyperparameter p optimized using the marginal likelihood is only 1.55% different than the designed magnet pitch. This shows when using the correct kernel function, physical properties can be estimated accurately using the GP modelled FF parameters.

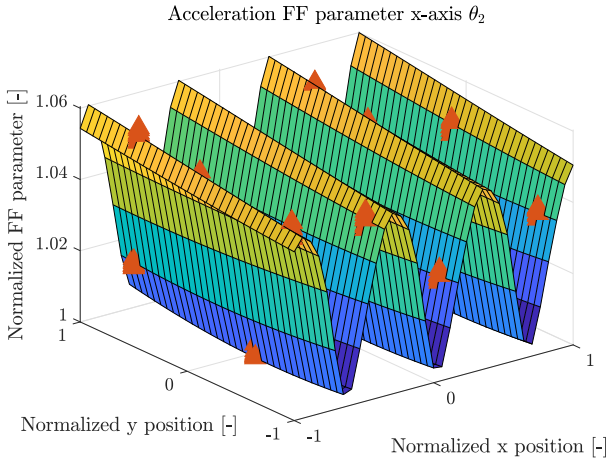

Fig. 29: Normalized GP regression of θ_2 for the experimental setup. The FF parameters (▲) are identified with ILC using BF on the training positions seen in Fig. 19. The kernel used is the periodic kernel, seen in (III.9)

Appendix E
GP Regressions of Experimental Setup

Next to the GP regressions of the acceleration parameters for x and y axes seen in Section IX, the other GP regressions for the FF parameters in (IX.3) can be seen below.

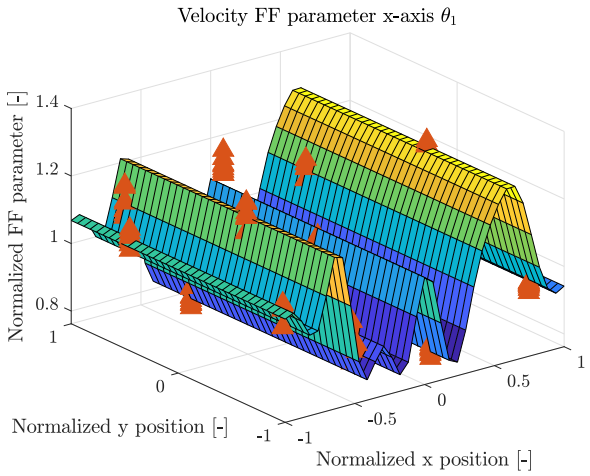

Fig. 30: Normalized GP regression using experimental data of FF parameter θ_1, seen in (IX.3)

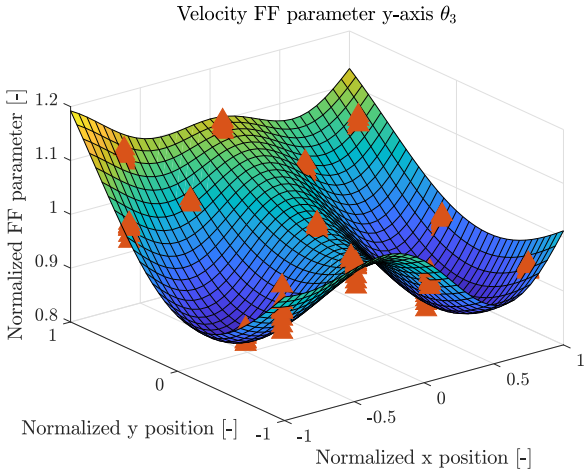

Fig. 31: Normalized GP regression using experimental data of FF parameter θ_3, seen in (IX.3)

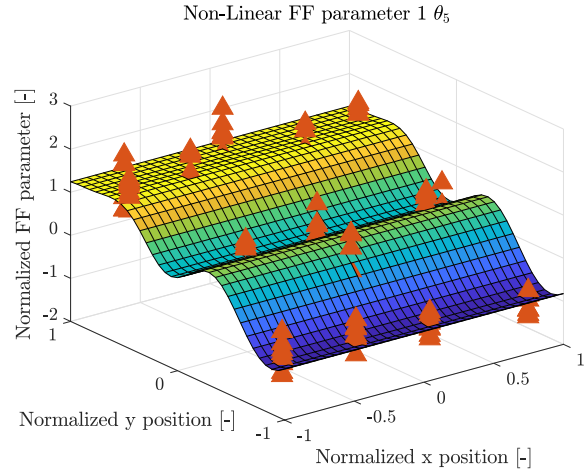

Fig. 32: Normalized GP regression using experimental data of FF parameter θ_5, seen in (IX.3)

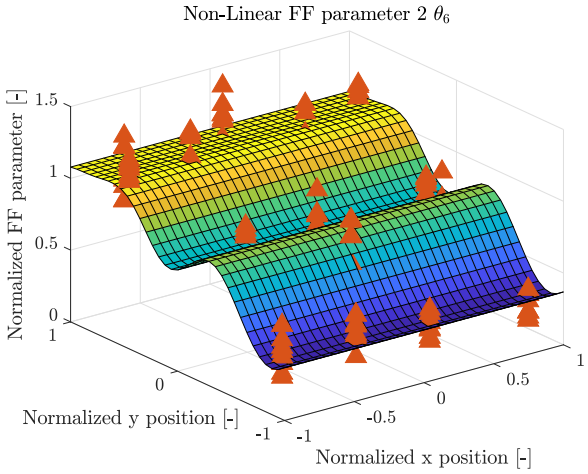

Fig. 33: Normalized GP regression using experimental data of FF parameter θ_6, seen in (IX.3)

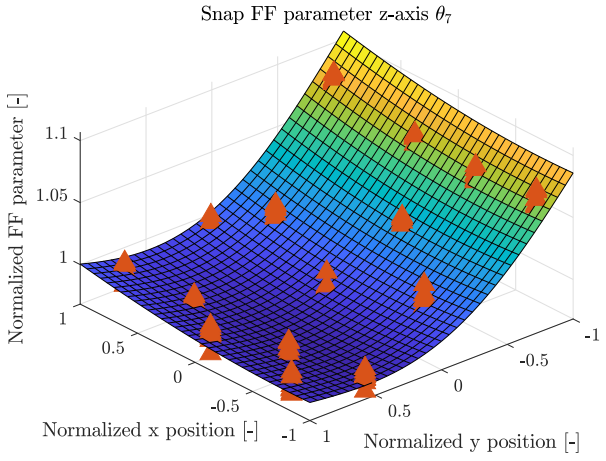

Fig. 34: Normalized GP regression using experimental data of FF parameter θ_7, seen in (IX.3)

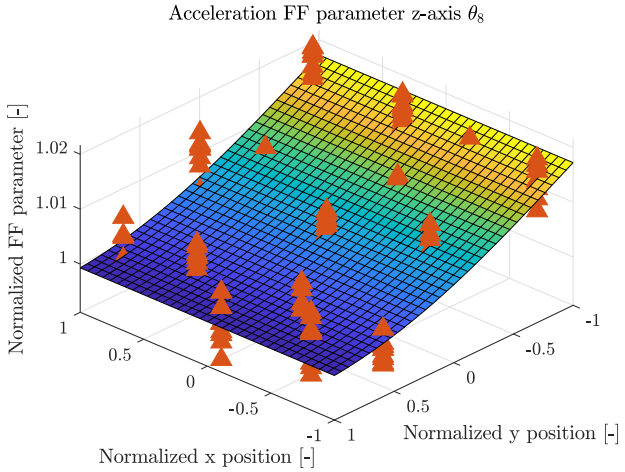

Fig. 35: Normalized GP regression using experimental data of FF parameter θ_8, seen in (IX.3)

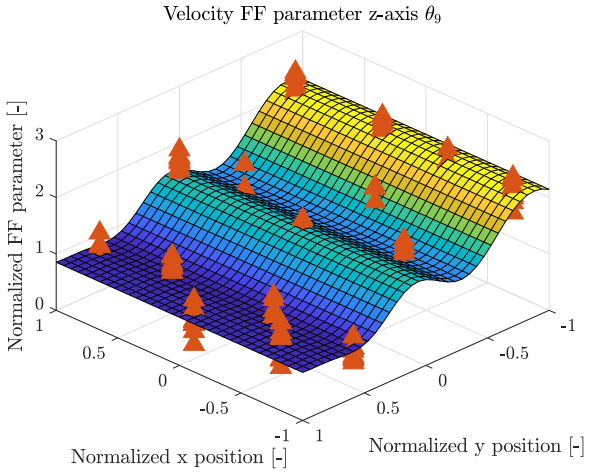

Fig. 36: Normalized GP regression using experimental data of FF parameter θ_9, seen in (IX.3)