Position-Dependent Snap Feedforward

A Gaussian Process Framework

van Haren, Max; Poot, Maurice ; Portegies, Jim ; Oomen, T.A.E.

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Position-Dependent Snap Feedforward: A Gaussian Process Framework

Max van Haren[1], Maurice Poot[1], Jim Portegies[2] and Tom Oomen[1,3]

Abstract— Mechatronic systems have increasingly high performance requirements for motion control. The low-frequency contribution of the flexible dynamics, i.e., the compliance, should be compensated for by means of snap feedforward to achieve high accuracy. Position-dependent compliance, which often occurs in motion systems, requires the snap feedforward parameter to be modeled as a function of position. Position-dependent compliance is compensated for by using a Gaussian process to model the snap feedforward parameter as a continuous function of position. A simulation of a flexible beam shows that a significant performance increase is achieved when using the Gaussian process snap feedforward parameter to compensate for position-dependent compliance.

I. INTRODUCTION

Feedforward control is fundamental for the tracking performance of motion systems, including wafer stages [1] and printing systems [2]. Traditionally, feedforward control is based on manual tuning. Recently, the focus has shifted towards learning feedforward from data, due to the increase of computational power [3]. For example, Iterative Learning Control (ILC) achieves high tracking performance by learning feedforward [4]. The key idea behind ILC is to learn, from trial to trial, an input that compensates for a known trial-invariant disturbance, for example, a reference signal. In contrast, fast motion is realized by designing lightweight systems, introducing dominant flexible dynamics affecting the tracking performance [5]. In addition, the combination of moving bodies and flexible dynamics introduce position-dependent behavior [6], [7].

Classical Linear Time-Invariant (LTI) acceleration feedforward compensates for the rigid-body dynamics of a system. In this case, the feedforward signal is scaled to the acceleration of the reference trajectory [8]. However, a well-tuned acceleration feedforward does not compensate for flexible dynamics [9].

Flexible dynamics lead to a situation where the compliance, i.e., the low-frequency contribution of the flexible dynamics, is compensated for by means of snap feedforward

[1]Max van Haren, Maurice Poot, and Tom Oomen are with the Control Systems Technology Section, Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. `m.j.v.haren@tue.nl`

[2]Jim Portegies is with the Centre for Analysis, Scientific Computing and Applications, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

[3]Tom Oomen is with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.

[9]. Snap feedforward uses the scaled fourth derivative of the reference. The snap feedforward parameter can be tuned online and improves performance for both Single-Input, Single-Output (SISO) systems [9] and multiple-input, multiple-output systems [10].

Many systems contain position-dependent behavior that introduces position-dependent compliance [11]–[13], which necessitates the need for a position-dependent compensation thereof. For this purpose, the snap feedforward parameter can be determined in a grid and estimated with, e.g., spline or linear interpolation. However, these interpolations have approximation errors since the dependency between position and snap feedforward parameters is generally unknown.

High motion control performance for systems with position-dependent dynamics can furthermore be achieved through the use of Linear Parameter Varying (LPV) control of the system. First, LTI dynamics can be scheduled according to the current configuration of the LPV system, resulting in high control performance, e.g., wafer stages [1], [14], or xy-positioning tables [15]. Second, ILC can be extended for LPV systems, which results in high performance through learning [16]. LPV model-free approaches that directly learn LPV controllers from data are investigated in [17], but are at present not competitive with model-based designs. Therefore, LPV control requires LPV modeling, which is often very challenging and the high modeling cost and complexity are usually not justified for industrial control applications.

Although feedforward design has improved significantly compared to traditional acceleration feedforward, a snap feedforward with systematic tuning for position-dependent compliance, capable of estimation at any arbitrary position, is currently lacking. This paper models the snap feedforward parameter as a continuous function of position by means of a Gaussian Process (GP) [18], [19], which allows for the compensation of position-dependent compliance without an LPV model. In addition, a GP is non-parametric and therefore does not require an assumption on the parametric form between the position and the snap feedforward parameter. In this paper, the feedforward parameters of a system are learned in a trial to trial fashion using ILC with Basis Functions (ILCBF) [20]. The contributions include:

C1 a framework to model the snap feedforward parameter as a function of position with a GP (Section III),

C2 ILCBF to automatically learn the snap feedforward parameter, which is directly used in the GP (Section IV),

C3 application to a benchmark example, confirming the capabilities of the framework (Section V).

Notation: Systems are SISO and discrete-time, unless stated otherwise. Continuous-time systems are transformed

in their discrete-time counterpart using finite-difference approximation. The trial number is indicated with the index j. Signals are assumed to be of length N. The weighted 2-norm of a vector $x \in \mathbb{R}^N$ is denoted as $\|x\|_W := \sqrt{(x^\top W x)}$, where $W \in \mathbb{R}^{N \times N}$ is a weighting matrix. Matrix $A \in \mathbb{R}^{N \times N}$ is positive (semi-)definite if and only if $x^\top A x \geq 0$, $\forall x \neq 0 \in \mathbb{R}^N$ and is denoted as $A \succeq 0$.

II. PROBLEM FORMULATION

In this section, the problem for determining a feedforward controller for position-dependent flexible dynamics is formulated. First, a general description for system with position-dependent flexible dynamics is given. Second, feedforward design, including acceleration and snap feedforward, is investigated. Finally, the problem addressed in this paper is defined.

A. Considered Class of Position-Dependent Systems

The considered class of systems with position-dependent flexible modes are encompassed using spatially distributed LTI systems [21], [22, Section 3.2]

$$y(k) = G(\rho, q^{-1})u(k),$$

$$G(\rho, q^{-1}) = \underbrace{\sum_{l=1}^{n_{RB}} G_{l,RB}(q^{-1})}_{\text{rigid-body modes}} + \underbrace{\sum_{i=1}^{n_f} D_i(\rho)G_{i,f}(q^{-1})}_{\text{flexible modes}}, \quad (1)$$

with scheduling variable $\rho \in \mathbb{R}^{\mathcal{D}}$, where $\mathcal{D}$ is the parameter space and q the shift operator, i.e., $q^{-\tau}s(k) = s(k - \tau)$. Note that ρ is not limited to the position, but can be any scheduling variable, see e.g. [23, Fig. 7]. The flexible mode shapes are represented by $D_i(\rho) = c_i(\rho)b_i^\top(\rho) \in \mathbb{R}$ and $n_{RB}, n_f \in \mathbb{N}_+$ are the amount of rigid-body and flexible modes. The systems $G_{l,RB}$ and $G_{i,f}$ are described as [21]

$$G_{l,RB}(q^{-1}) = \frac{c_l b_l^\top T_s^2}{(1 - q^{-1})^2},$$

$$G_{i,f}(q^{-1}) = \frac{1}{\frac{(1-q^{-1})^2}{T_s^2} + 2\zeta_i\omega_i\frac{1-q^{-1}}{T_s} + \omega_i^2}, \quad (2)$$

with T_s the sampling time, $c_l b_l^\top$ a representation of the rigid-body mode shape and $\zeta_i, \omega_i \in \mathbb{R}_+$ the flexible mode damping and frequency, respectively. A benchmark example is the flexible beam with varying performance location, as seen in Fig. 1.

For a fixed value of ρ, the system $G(\rho, q^{-1})$ is an LTI system, which is called the frozen dynamics of the system. Several bode magnitude diagrams evaluated at different values of ρ can be seen in Fig. 2.

B. Feedforward Design and the Limitation of Acceleration Feedforward

The main goal of feedforward is to minimize the error e, given a reference signal r, see Fig. 3. Consider the continuous-time LTI position-independent system equivalent to (1),

$$G_0(s) = \sum_{l=1}^{n_{RB}} \frac{c_l b_l^\top}{s^2} + \sum_{i=1}^{n_f} \frac{c_i b_i^\top}{s^2 + 2\zeta_i\omega_i s + \omega_i^2}, \quad (3)$$
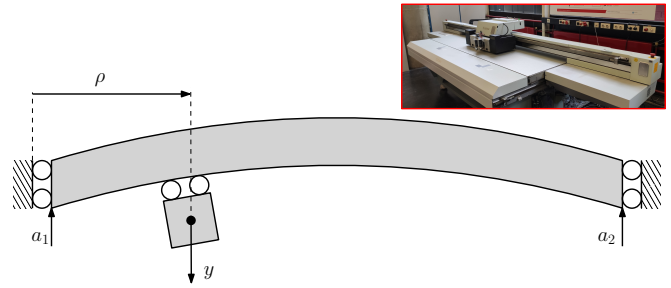

Fig. 1. Benchmark system for position-dependent snap feedforward, representing an H-type gantry commonly used in ,e.g., semiconductor back-end machines or large-format printing systems seen in the top right. The first mode shape of the flexible beam is displayed, with two actuators a_1 and a_2 and the output y. The performance location can vary, indicated by the scheduling variable ρ.
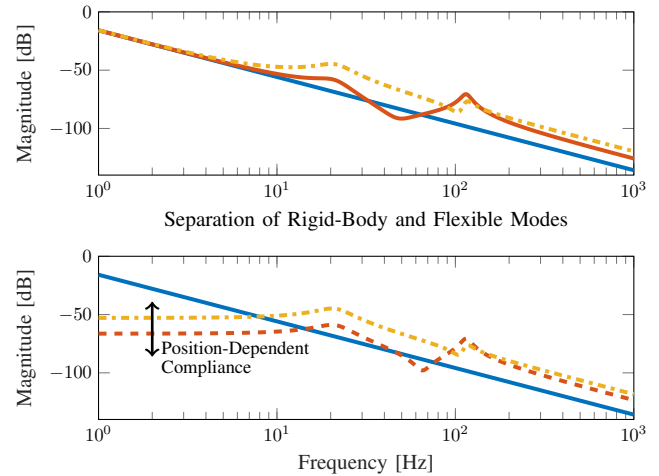

Fig. 2. Top: bode magnitude diagram of the frozen transfer function $G(\rho, q^{-1})$ evaluated at $\rho = 0$ mm (——), $\rho = 35$ mm (– –) and $\rho = 250$ mm (–··). Bottom: the separation of rigid-body (——) and flexible modes at $\rho = 35$ mm (– –) and $\rho = 250$ mm (–··) for the flexible beam in Fig. 1.

where s is the continuous-time indeterminate. The objective of feedforward, for the plant G_0, is to minimize

$$e(s) = S(s)r(s) - S(s)G_0(s)F(s)r(s),$$

with feedforward controller $F(s)$, feedback controller $C(s)$ and

$$S(s) = (I + G_0(s)C(s))^{-1}.$$

For systems with rigid-body modes which are not suspended, typical feedforward consists of acceleration feedforward,

$$F_a(s) = \hat{m}s^2, \quad (4)$$

with $\hat{m}$ an estimate of the mass of the system. Consider the open-loop tracking error e_o when assuming the system G_0 has only one translational rigid-body mode and well-tuned acceleration feedforward, i.e., $\sum_{i=l}^{n_{RB}} c_l b_l^\top = \frac{1}{m}$ in (3) and $\hat{m} = m$,

$$e_o(s) = r(s) - G_0(s)F_a(s)r(s)$$

$$= -\sum_{i=1}^{n_f} \frac{c_i b_i^\top}{s^2 - 2\omega_i\zeta_i s + \omega_i^2} ms^2 r(s). \quad (5)$$
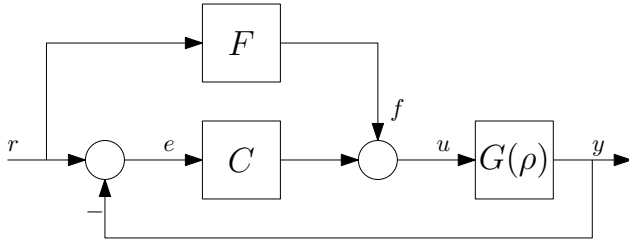
4779

Fig. 3. Control structure considered in this paper, consisting of feedforward controller F, feedback controller C and the system $G(\rho)$.

For low frequencies, the relation between the error in (5) and the acceleration of the reference is

$$\lim_{s \to 0} \left(\frac{e_o(s)}{s^2 r(s)} \right) = -m \sum_{i=1}^{n_f} \frac{c_i b_i^\top}{\omega_i^2}, \tag{6}$$

which shows a remaining contribution of the compliance term to the open-loop servo error for low frequencies when using acceleration feedforward.

C. Snap Feedforward

Flexible systems necessitate a compensation of the flexible dynamics during constant acceleration, as seen in (6), which is explicitly compensated for by means of snap feedforward. For this purpose, the acceleration feedforward in (4) is extended with additional snap feedforward δs^4, leading to

$$F_s(s) = ms^2 + \delta s^4, \tag{7}$$

with δ the snap feedforward parameter. The open loop servo error in (5) is equal to zero by designing δ as,

$$G_0^{-1}(s) \frac{1}{s^4} - m \frac{1}{s^2}.$$

Again assuming one translational rigid body mode, the low-frequency contribution of this controller is equal to

$$\delta = \lim_{s \to 0} \left(G_0^{-1}(s) \frac{1}{s^4} - m \frac{1}{s^2} \right) = -m^2 \sum_{i=1}^{n_f} \frac{c_i b_i^\top}{\omega_i^2}.$$

Hence, the feedforward controller in (7) results in

$$F_s(s) = ms^2 - m^2 \sum_{i=1}^{n_f} \frac{c_i b_i^\top}{\omega_i^2} s^4. \tag{8}$$

Note that for position-independent systems, the value of δ is a real constant and can be learned or tuned online.

D. Problem Definition

Position-independent snap feedforward does not compensate for position-dependent compliance of the system $G(\rho, q^{-1})$. LPV modeling of the position-dependent compliance is often not feasible and a parametric form between the position and the snap feedforward parameter, that is used in interpolation, is generally unknown.

Hence, the aim of this paper is modeling the snap feedforward parameter δ in (7) as a continuous function of position, i.e. $\delta(\rho)$, such that it compensates for position-dependent compliance, without an LPV model or parametric form.

III. POSITION-DEPENDENT SNAP FEEDFORWARD USING GAUSSIAN PROCESSES

In this section, the snap feedforward parameter is modeled as a function of position, i.e., $\delta(\rho)$, such that it can compensate for position-dependent compliance, hence constituting contribution C1. First, GPs are investigated, including the covariance function, the prior distribution and the posterior distribution. Finally, a method to model the snap feedforward parameter as a function of position by means of a GP is shown.

A. Gaussian Processes

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

$$f(\rho) \sim \mathcal{GP}\big(m(\rho), k(\rho, \rho')\big), \tag{9}$$

and is completely determined by the covariance function $k(\rho, \rho')$ and mean function $m(\rho)$,

$$\begin{aligned} k(\rho, \rho') &= \mathbb{E}\left[(f(\rho) - m(\rho))(f(\rho') - m(\rho'))\right], \\ m(\rho) &= \mathbb{E}[f(\rho)]. \end{aligned} \tag{10}$$

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 $f(\rho)$ on different inputs ρ and ρ'. The covariance function is discussed in Section III-B and the mean function is assumed to be zero. Note that this is not necessary, see [18, Section 2.7]. Training data for a GP are defined by sampling the function f on inputs and measuring the (noisy) training outputs y

$$\begin{aligned} y &= f(\rho) + \epsilon, \\ \text{where } \epsilon &\sim \mathcal{N}\left(0, \sigma_\epsilon^2 I\right), \end{aligned} \tag{11}$$

and σ_ϵ^2 is the variance of the noise acting on the output.

B. Covariance Function

The covariance function, or kernel, specifies the covariance between the inputs ρ and ρ'. An example is the squared exponential or Radial Basis Function (RBF) covariance function,

$$k_{RBF}(\rho, \rho') = \sigma_f^2 e^{-\frac{1}{2}(\rho - \rho')^\top \ell(\rho - \rho')}, \tag{12}$$

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 ℓ are the so-called hyperparameters, respectively the signal variance and length scale, that are optimized based on data using marginal likelihood optimization [18, Chapter 5].

C. Prior and Posterior Distribution

In a GP, function predictions are made using the posterior distribution, which is the prior distribution conditioned on function observations. Consider a finite set of test inputs $P_* \in \mathbb{R}^{n_* \times \mathcal{D}}$, with n_* the amount of test cases, the prior distribution of a GP is formulated as

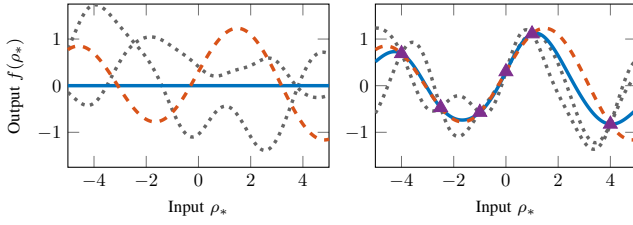$$f(P_*) \sim \mathcal{N}(0, K(P_*, P_*)), \tag{13}$$

4780

Fig. 4. Left: two samples (···) drawn from the prior, the prior mean (—) and a function to regress (- -). Right: two samples (···) drawn from the posterior, the posterior mean (—), a function to regress (- -) and the training data (▲).

that is, evaluate the covariance function at P_* and take the associated Gaussian distribution. Several samples drawn from this distribution can be seen in Fig. 4. The test outputs $f(P_*)$ are jointly distributed with the training outputs y as

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

with kernel matrices $K(P,P) \in \mathbb{R}^{n \times n}$, $K(P_*,P_*) \in \mathbb{R}^{n_* \times n_*}$ and $K(P,P_*) = K(P_*,P)^\top \in \mathbb{R}^{n \times n_*}$, with n the amount of training cases. The parameter σ_n^2 is an approximation of the noise acting on the output σ_ϵ^2, seen in (11), and an additional hyperparameter. The training inputs $P \in \mathbb{R}^{n \times \mathcal{D}}$ and test inputs $P_* \in \mathbb{R}^{n_* \times \mathcal{D}}$ can be any single point or vector of positions, i.e.,

$$
\begin{aligned}
P_* &= \begin{bmatrix} \rho_{*,1} & \rho_{*,2} & \cdots & \rho_{*,n_*} \end{bmatrix}^\top, \\
P &= \begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_n \end{bmatrix}^\top.
\end{aligned} \tag{15}
$$

Using the joint distribution in (14) and Bayes' rule, estimations are made using the posterior distribution

$$
f(P_*) \,\big|\, [P_*, P, y] \sim \mathcal{N}\left(\bar{f}(P_*), \mathrm{cov}(f(P_*))\right), \tag{16}
$$

where

$$
\begin{aligned}
\bar{f}(P_*) &\equiv \mathbb{E}[f(P_*)] = K_*^\top \left(K(P,P) + \sigma_n^2\right)^{-1} y, \\
\mathrm{cov}(f(P_*)) &= K_{**} - K_*^\top \left(K(P,P) + \sigma_n^2\right)^{-1} K_*.
\end{aligned} \tag{17}
$$

The prior and posterior enable the user to train a GP with function observations and estimate function values.

D. Gaussian Processes for Snap Feedforward

The snap feedforward parameter is modeled as a continuous function of position to compensate for position-dependent compliance by means of a GP, i.e.,

$$
\delta(\rho) := f(\rho) \sim \mathcal{GP}(m(\rho), k(\rho, \rho')). \tag{18}
$$

The training data for the GP, seen in (11), are defined as

$$
y = \begin{bmatrix} \delta_1 & \cdots & \delta_n \end{bmatrix}^\top \in \mathbb{R}^{n \times 1}, \tag{19}
$$

where δ_i is determined using the method in Section IV. The training inputs in (15), for systems with position-dependent flexible modes, are equal to the scheduling variables ρ where the snap feedforward parameters δ_i are determined.

Snap feedforward parameters can be estimated on positions ρ_* using

$$
\delta(\rho_*) = K(P, \rho_*)^\top \left(K(P,P) + \sigma_n^2\right)^{-1} y, \tag{20}
$$

with y from (19). Using (20), snap feedforward parameters are estimated on positions ρ_* using the training data, such that position-dependent compliance is compensated for.

IV. LEARNING FROZEN PARAMETERS $\delta(\rho)$ VIA BASIS FUNCTIONS

In this section, feedforward parameters are learned in a trial to trial fashion, which will serve as training data for the GP, leading to contribution C2. Here, ILCBF is used to learn the parameters, but the framework can directly be extended to other feedforward parameter tuning approaches.

ILCBF parametrizes the feedforward signal and learns the feedforward parameters in a trial to trial fashion. The optimization criterion in ILCBF is specified as [2]

$$
V(\theta_{j+1}) = \|e_{j+1}(k)\|_{W_e}^2 + \|f_{j+1}(k)\|_{W_f}^2 + \|f_{j+1}(k) - f_j(k)\|_{W_{\Delta f}}^2, \tag{21}
$$

with weighting matrices $W_e \succ 0$ and W_f, $W_{\Delta f} \succeq 0$ and $\theta_j \in \mathbb{R}^{n_\theta}$ the feedforward parameters. Here, W_e enforces optimal performance, W_f robustness for model-mismatch and $W_{\Delta f}$ robustness against trial-varying disturbances. The error in trial $j+1$ can be written as

$$
\begin{aligned}
e_{j+1}(k) &= S(q^{-1})r(k) - S(q^{-1})G_0(q^{-1})f_{j+1}(k) \\
&= e_j(k) - S(q^{-1})G_0(q^{-1})\Big(f_{j+1}(k) - f_j(k)\Big),
\end{aligned} \tag{22}
$$

where now, G_0 is for instance a nominal model of a position-dependent system. The feedforward force is parameterized in terms of the feedforward parameters θ_j, i.e., $f_j(k) = F(\theta_j)r(k)$, with $F(\theta_j) \in \mathbb{R}^{N \times N}$ the convolution matrix of a linear system with parameters θ_j. ILCBF updates the feedforward parameters in a trial to trial fashion using

$$
\theta_{j+1}^* = \arg\min_{\theta_{j+1}} V(\theta_{j+1}). \tag{23}
$$

When choosing $F(\theta_j)$ linearly in θ_j, the optimization criterion in (21) becomes quadratic in θ_{j+1}. Hence, an analytic solution to (23) exists [24].

Given the basis function matrix $\Psi(k) = \partial/\partial\theta_j F(\theta_j)r(k) \in \mathbb{R}^{N \times n_\theta}$ and the weighting matrices W_e, W_f and $W_{\Delta f}$, the analytic solution to (23) is

$$
\begin{aligned}
\theta_{j+1} &= Le_j + Q\theta_j, \\
L &= R^{-1}\left(\Psi^\top G_0^\top S^\top W_e\right), \\
Q &= R^{-1}\Psi^\top \left(G_0^\top S^\top W_e G_0 S + W_{\Delta f}\right)\Psi, \\
R &= \left(\Psi^\top \left(G_0^\top S^\top W_e G_0 S + W_f + W_{\Delta f}\right)\Psi\right),
\end{aligned} \tag{24}
$$

where (q^{-1}) and (k) have been left out for brevity. The parameter update in (24) leads to monotonic convergence of $\|f_j(k)\|$, provided matrices W_e, W_f and $W_{\Delta f}$ are selected properly [2]. In addition, the definition of monotonic convergence in the iteration domain of ILCBF is given in [20]. Robustness, with respect to model mismatch due to the position-dependent dynamics, can be enforced by increasing
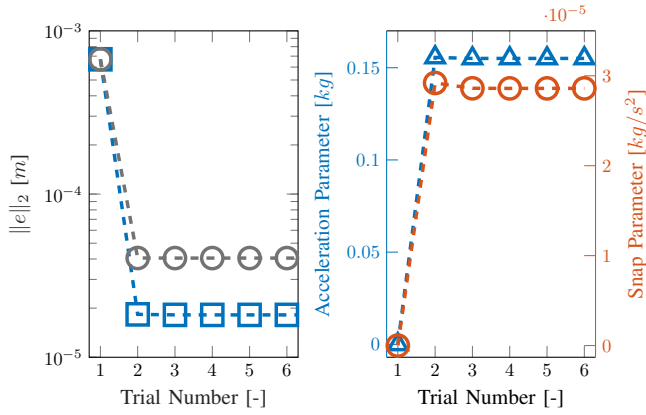
4781

Fig. 5. Left: error 2-norm for acceleration feedforward (- ⊖ -) and acceleration with snap feedforward (- ⊟ -) when applying ILCBF. Right: acceleration (- △ -) and snap (- ⊖ -) feedforward parameters when applying ILCBF. The scheduling variable ρ is equal to 250 mm.

Fig. 6. GP regression of the snap feedforward parameter of the flexible beam with the posterior mean (—), the posterior mean $\pm 2\sigma$ (▫) and the training data (+).

W_f. Now, (24) can be used in combination with the error in trial j to compute a new set of feedforward parameters θ_{j+1} for a fixed value of ρ.

V. FLEXIBLE BEAM EXAMPLE

In this section, the GP snap framework is applied to a simulation of a flexible beam, hence constituting contribution C3. First, the example setup is discussed, followed by the application of both ILCBF and GP snap feedforward. Finally, the results and comparison to position-independent snap feedforward are given.

A. Example System

The unsupported (free-free) flexible beam in Fig. 1 is considered, having dominant flexible dynamics. The beam has a total length of 500 mm. The flexible beam consists of two actuators and a sensor with variable position. To assure the system is SISO, the two actuators have the same input, i.e., $a_1 = a_2 = u/2$. Varying performance location, e.g., as is occurring in wafer exposure [7] or flat-bed printing [16], can be emulated by changing the sensor position ρ. The translational rigid body and two flexible modes are considered, i.e., $n_{RB} = 1$ and $n_f = 2$. The feedback controller C is taken fixed and is a lead filter and a gain with a bandwidth of 4 Hz. Due to the inherent dynamics of the flexible beam and a changing performance location, it is a suitable example for position-dependent snap feedforward.

B. Learning Frozen Parameters on the Flexible Beam

ILCBF as specified in Section IV has been implemented on the flexible beam to iteratively learn the feedforward parameters, including the snap feedforward parameter. A fourth-order reference as designed in [25] has been used. The model G_0 for ILCBF for all positions, see e.g. (24), is a model of the flexible beam with sensor position $\rho = 250$ mm, which is seen in Fig. 2. Fig. 5 shows that the error 2-norm can be significantly reduced by learning the feedforward parameters in a trial to trial fashion. It furthermore shows
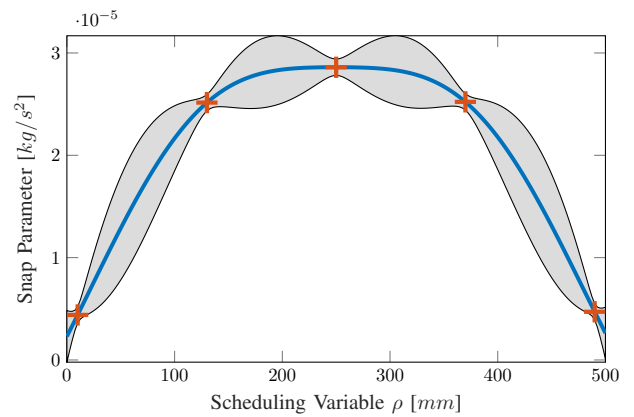
how additional snap feedforward can improve performance compared with traditional acceleration feedforward.

C. GP Snap Feedforward Parameter

The snap feedforward parameter is modeled as a function of position using a GP. First, the training inputs in (15) are defined as five equispaced positions,

$$P = \begin{bmatrix} 10 & 130 & 250 & 370 & 490 \end{bmatrix}^\top. \tag{25}$$

On the positions P, ILCBF has been performed and the resulting snap feedforward parameter is the training data as in (19). The RBF covariance function in (12) is used, with hyperparameters optimized based on marginal likelihood. For visualization purposes, the test positions P_* are defined as an equispaced dense grid covering the beam. The GP regression in Fig. 6 shows the snap feedforward parameter varies when changing the scheduling variable ρ, which is further supported by looking at Fig. 2.

D. Results

To evaluate the performance of the framework, GP snap feedforward is compared with position-independent snap feedforward. Position-independent snap feedforward uses the same snap feedforward parameters for all values of ρ, whereas the GP snap feedforward estimates the snap feedforward parameter using (20). The error 2-norm for both GP snap and position-independent snap feedforward for several test positions can be seen in Fig. 7. Fig. 7 shows that GP snap feedforward outperforms the position-independent feedforward significantly when the sensor position moves outside the center position. At the edges of the flexible beam, roughly a performance increase of factor two in terms of the error 2-norm is observed. Near the center of the beam, performance is equal, which is expected since the position-independent snap feedforward uses the snap feedforward parameter determined at the center position. The achievable performance gain is further supported by looking at the time domain error for $\rho = 30$ mm in Fig. 8.
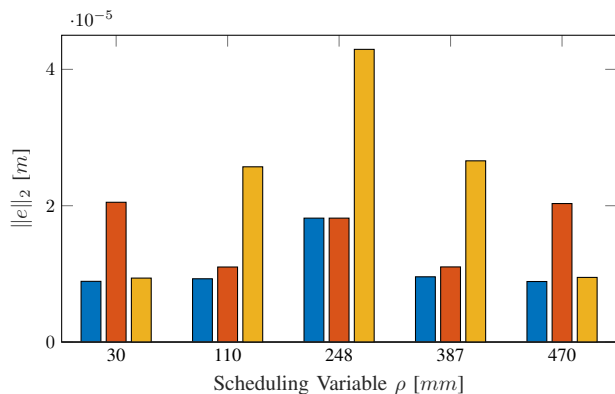
Fig. 7. Error 2-norm for GP snap (■), position-independent snap (■) and acceleration feedforward (■). The GP feedforward has comparable performance for all positions, while both the position-independent snap and acceleration feedforward have higher error 2-norms for certain positions.
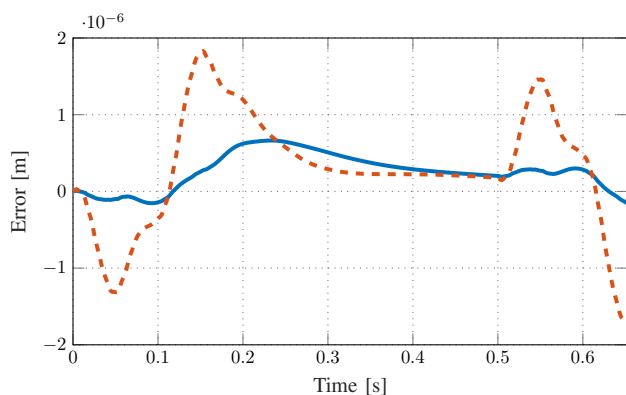

Fig. 8. Time domain error for GP snap feedforward (——) and position-independent snap feedforward (- - -) for the scheduling variable $\rho = 30$ mm. The maximum error for GP snap feedforward is roughly 3 times lower than the maximum error for position-independent snap feedforward.

VI. Conclusions

This work describes a method to use a GP to model the snap feedforward parameter as a function of position. A GP works especially well since it is non-parametric and therefore does not assume a parametric form between the position and the snap feedforward parameter, which is typically unknown. The framework is applied to a flexible beam, which shows the dependency of the snap feedforward parameter on position. GP snap feedforward shows a significant performance increase compared with position-independent snap feedforward.

Future research on this topic is directed at integrating and testing the framework for MIMO systems and adding other position-dependent feedforward parameters or effects. Furthermore, a method to automatically and optimally determine the training inputs in (25) is investigated. Lastly, experimental validation confirming the practical applicability of the framework is a subject of ongoing research.

References

[1] R. de Rozario, R. Voorhoeve, W. Aangenent, and T. Oomen, "Global feedforward control of spatio-temporal mechanical systems: With application to a prototype wafer stage," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 575–14 580, 2017, 20th IFAC World Congress.

[2] J. Bolder, T. Oomen, S. Koekebakker, and M. Steinbuch, "Using iterative learning control with basis functions to compensate medium deformation in a wide-format inkjet printer," *IFAC Mechatronics*, vol. 24, pp. 944–953, 12 2014.

[3] T. Oomen, "Learning for advanced motion control," in *IEEE 16th Int. Workshop on Advanced Motion Control*. IEEE, 9 2020, pp. 65–72.

[4] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Syst. Mag.*, vol. 26, pp. 96–114, 6 2006.

[5] M. J. C. Ronde, "Feedforward control for lightweight motion systems," Ph.D. dissertation, Eindhoven University of Technology, 2014.

[6] M. M. da Silva, W. Desmet, and H. V. Brussel, "Design of mechatronic systems with configuration-dependent dynamics: Simulation and optimization," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*. IEEE, 2007, pp. 1–6.

[7] R. Voorhoeve, R. de Rozario, W. Aangenent, and T. Oomen, "Identifying position-dependent mechanical systems: A modal approach applied to a flexible wafer stage," *IEEE Trans. Contr. Syst. Technol.*, vol. 29, pp. 194–206, 1 2021.

[8] T. Oomen, "Control for precision mechatronics," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. Springer London, 2020, pp. 1–10.

[9] M. Boerlage, R. Tousain, and M. Steinbuch, "Jerk derivative feedforward control for motion systems," in *Proc. American Control Conference (ACC'04)*, vol. 5. IEEE, 2004, pp. 4843–4848 vol.5.

[10] M. Boerlage, "MIMO jerk derivative feedforward for motion systems," in *Proc. American Control Conference*, vol. 2006. IEEE, 2006.

[11] R. Voorhoeve, N. Dirkx, T. Melief, W. Aangenent, and T. Oomen, "Estimating structural deformations for inferential control: a disturbance observer approach," *IFAC-PapersOnLine*, vol. 49, pp. 642–648, 2016.

[12] R. de Rozario and T. Oomen, "Frequency response function identification of periodically scheduled linear parameter-varying systems," in *Mechanical Systems and Signal Processing*, vol. 148, 2 2021.

[13] N. Kontaras, M. Heertjes, and H. Zwart, "Continuous compliance compensation of position-dependent flexible structures," *IFAC-PapersOnLine*, vol. 49, no. 13, pp. 76–81, 2016, 12th IFAC Workshop on Adaptation and Learning in Control and Signal Processing ALCOSP 2016.

[14] M. G. Wassink, M. van de Wal, C. Scherer, and O. Bosgra, "LPV control for a wafer stage: beyond the theoretical solution," *Control Engineering Practice*, vol. 13, pp. 231–245, 2 2005.

[15] R. Toth, M. van de Wal, P. S. C. Heuberger, and P. M. J. V. den Hof, "LPV identification of high performance positioning devices," in *Proc. American Control Conference (ACC'11)*. IEEE, 6 2011, pp. 151–158.

[16] R. de Rozario, T. Oomen, and M. Steinbuch, "Iterative learning control and feedforward for LPV systems: Applied to a position-dependent motion system," in *Proc. American Control Conference (ACC'17)*. IEEE, 5 2017, pp. 3518–3523.

[17] S. Formentin, D. Piga, R. Tóth, and S. M. Savaresi, "Direct learning of LPV controllers from data," *Automatica*, vol. 65, pp. 98–110, 2016.

[18] C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2004, vol. 3176.

[19] G. Pillonetto, F. Dinuzzo, T. Chen, G. D. Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, pp. 657–682, 3 2014.

[20] J. van de Wijdeven and O. Bosgra, "Using basis functions in iterative learning control: analysis and design theory," *International Journal of Control*, vol. 83, pp. 661–675, 4 2010.

[21] W. Gawronski, *Advanced Structural Dynamics and Active Control of Structures*, W. K. Gawronski, Ed. Springer New York, 2004.

[22] S. O. R. Moheimani, D. Halim, and A. J. Fleming, *Spatial Control of Vibration - Theory and Experiments*. World Scientific Publishing Co. Pte. Ltd., 2003, vol. 10.

[23] D. Y. Abramovitch, "Trying to keep it real: 25 years of trying to get the stuff I learned in grad school to work on mechatronic systems," in *2015 IEEE Conference on Control Applications (CCA)*. IEEE, 9 2015, pp. 223–250.

[24] J. Bolder and T. Oomen, "Rational basis functions in iterative learning control—with experimental verification on a motion system," *IEEE Trans. Contr. Syst. Technol.*, vol. 23, pp. 722–729, 3 2015.

[25] P. Lambrechts, M. Boerlage, and M. Steinbuch, "Trajectory planning and feedforward design for electromechanical motion systems," *Control Engineering Practice*, vol. 13, pp. 145–157, 2 2005.