JOURNAL PAPER

Tracking Control of Non-minimum Phase Systems: A Kernel-based Approach

Mohammadmahdi Mehrabi[a] and Keivan Ahmadi[a]

[a]Department of Mechanical Engineering, University of Victoria, Victoria, British Columbia, Canada V8W 2Y2.

ARTICLE HISTORY
Compiled October 4, 2023

ABSTRACT
Feedforward control with model inversion can achieve high-accuracy output tracking, but it generates unbounded control input for non-minimum phase (NMP) models due to unstable poles. Pseudo-inversion methods use parametric regression to bound the control input based on the desired trajectory and the system dynamics. We propose a new non-parametric pseudo-inversion method that uses Bayesian inference and kernel functions to design optimal and flexible control trajectories. Compared to existing methods, the presented approach can handle arbitrary types of NMP systems and trajectories and embed desirable features in the control input. We also develop a recursive limited-preview version that is computationally efficient and suitable for online and adaptive applications. We present closed-form equations for both versions and compare their performances with existing methods in benchmark examples.

KEYWORDS
Feedforward Control; Nonminimum Phase System; Bayesian Estimation; Model Inversion; Kernel Estimation

1. Introduction

Tracking control aims to minimize the error between the system output and the desired trajectories. Feedforward controllers achieve this goal by compensating for the system's known dynamics and disturbances before the associated error appears in the output. Therefore, even when a feedback controller is used, adding a feedforward controller can augment its performance by pre-compensating for the system dynamics (Sharma, & Pradhan, 2017). In other cases where a feedback device is not available, feedforward control is the only feasible solution (Duan, Yoon, & Okwudire, 2018).

Feedforward control based on model inversion eliminates tracking error by filtering the reference (desired) trajectory ($y_d(t)$) through the inverse of the system's model(Butterworth, Pao, & Abramovitch, 2012). The schematic block-diagram of this control strategy is depicted in Fig.1, where $C(z^{-1})$ is the inverse of the system model $G(z^{-1})$. Model inversion, however, is not applicable to non-minimum phase (NMP) systems. In the discrete-time domain, the transfer function of the NMP system includes zeros that are outside the unit circle on the complex plane. Those zeros become unsta-

CONTACT Keivan Ahmadi. Email: kvahmadi@uvic.ca

ble poles for the inverse filter, leading to unbounded control inputs ($u(t)$). NMP systems are commonly encountered in applications with transport delay, non-collocated sensors and actuators, or fast sampling rates (AAström, Hagander, & Sternby, 1984; Butterworth, Pao, & Abramovitch, 2011; Dai, Li, Zhu, & Zhang, 2019; Miu, 2012). Approximate model inversion, stable (or direct) model inversion, and pseudo-inversion are the three notable strategies in the literature to treat NMP systems in feedforward control.

Various methods with the approximate model inversion approach have been presented to determine a stable inverse model for NMP systems. The non-minimum phase zeros ignored (NPZ-Ignored) method simply ignores the NMP zeros of the forward system in the inverse model. The resulting inverse model is therefore stable but both the phase and magnitude of the generated output deviate from the desired trajectory (Butterworth et al., 2012; Rigney, Pao, & Lawrence, 2009). The zero phase error tracking control (ZPETC) method approximates a stable inverse that does not cause phase difference but suffers from magnitude error (Tomizuka, 1987). Alternatively, in zero magnitude error tracking control (ZMETC), the magnitude error is eliminated, but phase error is allowed. Advanced variants of this approach such as truncated series (TS) result in better tracking, but they are not applicable when one of the zeros is on the unit circle (i.e. non-hyperbolic systems) (Gross, & Tomizuka, 1994). Instead of inverting the forward model, the inverse model can also be identified directly by input-output data and system identification methods (Blanken, Meijdenberg, & Oomen, 2018; Blanken, & Oomen, 2020; Zhou, Helwa, & Schoellig, 2018; Zundert, & Oomen, 2018). However, because the knowledge of the future output is required to determine the optimal input, the inverse models resulting from approximate inversion are usually non-causal and their experimental identification is complex. For example, Blanken and Oomen (Blanken et al., 2020) adopted a kernel-based system identification method that uses non-casual kernels to impose a prior on the identified inverse model. Data-driven methods such as neural networks can also be used to identify such a model, but they are difficult to train online and their stability cannot be investigated rigorously (Chou, Duan, & Okwudire, 2021).

Inspired by the non-causality of the approximate inverse models, the direct or stable inversion methods leverage a known preview of the future desired trajectory to compute sufficient pre-actuation that cancels the effect of NMP zeros (Devasia, Chen, & Paden, 1996; Hunt, Meyer, & Su, 1996; Marconi, Marro, & Melchiorri, 2001). In theory, this method can entirely eliminate tracking error for any NMP system, but the length of the required preview and thereby the amount of the required pre-actuation tends to infinity as the system zeros approach the unit circle (i.e. non-hyperbolic or near non-hyperbolic systems). The limited-preview variants of this approach have been presented for restrictive types of systems and trajectories, or at the expense of allowing limited tracking error (Piazzi, & Visioli, 2005; Zou, & Devasia, 1999). Unlike the approximate and stable inversion methods where a closed-form inverse model is determined, the pseudo-inversion approach uses regression to design a bounded input based on the prior knowledge of the system dynamics and desired trajectory (Ramani, Duan, Okwudire, & Ulsoy, 2017; Romagnoli, & Garone, 2019). In the most generalized form of pseudo-inversion, the Filtered Basis Functions (FBF) method parametrizes the input trajectory ($u(t)$) as a linear sum of a set of basis functions whose contributions are determined by minimizing the tracking error. This technique is shown to be applicable to a wide range of systems and trajectories, even when the zero is located on or close to the unit circle. In addition, this method is versatile for designing input trajectories that strike a balance between tracking error and input smoothness. For

instance, feedforward control with FBF was used to design an input trajectory with an optimal frequency spectrum that does not excite unwanted structural vibrations in the feed drive of a computer numerical control (CNC) machine tool(Okwudire, Ramani, & Duan, 2016). Because prior knowledge of the entire desired trajectory is required for designing the corresponding input trajectory, the FBF method is not efficient for following long trajectories or online applications where only a limited preview of the trajectory is available. To address this issue, Duan et al. (Duan et al., 2018) presented the limited-preview version of the FBF method that leverages the local property of spline basis functions to recursively determine the optimal input trajectory for only a short window of the desired trajectory at a time. They showed that the limited preview version maintains the versatility and efficiency of the full-preview FBF.

Following the pseudo-inversion approach, this paper presents a new non-parametric regression method to design an optimal input trajectory for feedforward tracking control. Instead of parametrizing the input trajectory with basis functions, this method models it as a Gaussian process whose mean and covariance matrix are determined by Bayesian inference according to the desired output trajectory and the model of the system dynamics. Kernels are used to impose a prior on the covariance of the input trajectory, and the hyperparameters of the kernels are adjusted to minimize tracking error or to balance it with input smoothness. Using benchmark examples, we show that the presented method minimizes tracking error by generating bounded inputs for NMP systems even when zeros are near or on the unit circle. Furthermore, we extend this method to recursively compute the input trajectory in small batches based on the limited preview of the desired trajectory, making it suitable for online and adaptive applications. We show that the parameters of the recursive formula and the hyperparameters of the kernel can be adjusted to generate bounded inputs for NMP systems with arbitrary zero locations. Additionally, the types of kernel functions that can be used with the presented method are virtually unlimited. Therefore, input trajectories with desired characteristics can be designed by choosing appropriate kernel functions. In all the studied benchmark examples, the performance of the presented method is compared with the full and limited-preview versions of the FBF method to highlight these advantages.

In the next section, we give a brief overview of the studied tracking control problem and the general pseudo-inversion approach to solving it. Also in that section, we present both the full- and limited-preview versions of the new kernel-based methods for pseudo-inversion. Subsequently, in Section 3, we apply the presented method to benchmark examples where their performances are compared with the FBF method. The design parameters' influence on the stability of the recursive method is also studied in that section.

2. Feedforward tracking control by pseudo-inversion

The block diagram of plant-inversion feedforward tracking control for a Single Input Single Output (SISO) system in the discrete-time domain is shown in Fig.1. The desired trajectory, $y_d(t), t \in \mathbb{N}_0$, is filtered through the tracking controller, $C(z^{-1})$, to provide input $u(t)$ to the plant $G(z^{-1})$. The model, $G(z^{-1})$, can be the transfer function of the plant or the overall closed-loop system if the plant is controlled by feedback control. The filter $C(z^{-1})$ is designed to minimize the tracking error of the output, $e(t) = y(t) - y_d(t)$. We assume the plant model ($G(z^{-1})$) is non-minimum phase and thus the ideal tracking controller $C(z^{-1})$ cannot be obtained by the simple

inversion of $G(z^{-1})$.

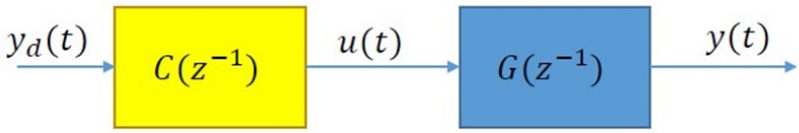

Figure 1. Plant inversion feedforward control scheme

Instead of determining the optimal filter $C(z^{-1})$, the Filtered Basis Functions (FBF) method (Ramani et al., 2017) uses regression to directly determine the input sequence $u(t)$. The general methodology in FBF comprises parametrizing the input trajectory, $u(t)$, as a linear combination of a set of basis functions, and then determining their optimal weight to minimize tracking error. This method was shown to be highly effective in minimizing tracking errors for general NMP systems. Here, we present an alternative, kernel-based, form of the FBF method which is more flexible and easier to implement in regularizing the control input and minimizing tracking error. A general overview of the FBF method is explained in the following section, followed by first presenting the full-preview formulation of the kernel-based method and then extending it to the limited-preview version for online and adaptive applications.

2.1. Overview of Filtered Basis Functions (FBF) method

Let $u(t), t = 0..E$, be the input to the linear time-invariant system. In the FBF method, $u(t)$ is approximated by a linear combination of a set of n basis functions, $\varphi_i(t), i = 1..n$:

$$u(t) = \sum_{i=0}^{n} \gamma_i \varphi_i(t) \tag{1}$$

Subsequently, the system output can also be expressed as a linear combination of the same basis functions filtered by the plant dynamics:

$$y(t) = g(t) * u(t) = \sum_{i=0}^{n} \gamma_i \tilde{\varphi}_i(t); \quad \tilde{\varphi}_i(t) = g(t) * \varphi_i(t) \tag{2}$$

where $*$ stands for discrete-time convolution, $g(t) = \mathcal{Z}^{-1}[G(z^{-1})]$ is the impulse response, and $\tilde{\varphi}_i$ are filtered basis functions (FBF). The weight of contribution from each basis function is determined by minimizing the squared tracking error for the

entire trajectory comprising $t = 0..E$:

$$\underbrace{\begin{bmatrix} \tilde{\varphi}_0(0) & \tilde{\varphi}_1(0) & \dots & \tilde{\varphi}_n(0) \\ \tilde{\varphi}_0(1) & \tilde{\varphi}_1(1) & \dots & \tilde{\varphi}_n(1) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\varphi}_0(E) & \tilde{\varphi}_1(E) & \dots & \tilde{\varphi}_n(E) \end{bmatrix}}_{\tilde{\mathbf{\Phi}}} \underbrace{\begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_n \end{bmatrix}}_{\boldsymbol{\gamma}} + \mathbf{e} = \underbrace{\begin{bmatrix} y_d(0) \\ y_d(1) \\ \vdots \\ y_d(E) \end{bmatrix}}_{\mathbf{y}_d} \tag{3}$$

where $\mathbf{e} = [e(0)...e(E)]^T$ is the vector of tracking errors. The least squares approximation of $\boldsymbol{\gamma}$ in Eq.(3) is unique if the FBF are linearly independent. Linearly independent FBF can be created by appropriately selecting the initial conditions in filtering (Ramani et al., 2017). Alternatively, regularization can be used to remove ill-conditioning from matrix $\tilde{\mathbf{\Phi}}$ in Eq. (3). For instance, least squares approximation with Tikhonov regularization for $\boldsymbol{\gamma}$ is the solution to the following optimization problem:

$$\hat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}} |\mathbf{y}_d - \tilde{\mathbf{\Phi}}\boldsymbol{\gamma}|^2 + \alpha \boldsymbol{\gamma}^T \boldsymbol{\gamma} \tag{4}$$

where α is regularization coefficient. The solution of the quadratic optimization problem in Eq.(4) is expressed as follows:

$$\hat{\boldsymbol{\gamma}} = (\tilde{\mathbf{\Phi}}^T \tilde{\mathbf{\Phi}} + \alpha \mathbf{I})^{-1} \tilde{\mathbf{\Phi}}^T \mathbf{y}_d \tag{5}$$

Estimated input $\hat{\mathbf{u}} = [\hat{u}(0)...\hat{u}(E)]^T$ is obtained by substituting $\hat{\boldsymbol{\gamma}}$ for $\boldsymbol{\gamma}$ in Eq.(1):

$$\hat{\mathbf{u}} = \mathbf{\Phi}(\tilde{\mathbf{\Phi}}^T \tilde{\mathbf{\Phi}} + \alpha \mathbf{I})^{-1} \tilde{\mathbf{\Phi}}^T \mathbf{y}_d \tag{6}$$

where $\mathbf{\Phi}$ is the matrix of basis functions, similar to $\tilde{\mathbf{\Phi}}$.

2.2. Full-preview kernel-based pseudo-inversion

Without parametrizing the input by basis functions, Eqs.2 and 3 can be combined and expressed in the following discrete-time convolution form:

$$\underbrace{\begin{bmatrix} g(0) & 0 & \dots & 0 \\ g(1) & g(0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g(E) & g(E-1) & \dots & g(0) \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(E) \end{bmatrix}}_{\mathbf{u}} + \mathbf{e} = \underbrace{\begin{bmatrix} y_d(0) \\ y_d(1) \\ \vdots \\ y_d(E) \end{bmatrix}}_{\mathbf{y}_d} \tag{7}$$

Regularized Least Squares (ReLS) estimation for the input sequence $\mathbf{u}$ can be obtained as the solution to the following quadratic optimization problem:

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} |\mathbf{y}_d - \mathbf{G}\mathbf{u}|^2 + \alpha \mathbf{u}^T \mathbf{P}^{-1} \mathbf{u} \tag{8}$$

where α is regularization coefficient and $\mathbf{P}$ is regularization matrix. The solution to the quadratic optimization problem in Eq.(8) is expressed as follows:

$$\hat{\mathbf{u}} = (\mathbf{P}\mathbf{G}^T\mathbf{G} + \alpha\mathbf{I})^{-1}\mathbf{P}\mathbf{G}^T\mathbf{y}_d \tag{9}$$

Notice that the ReLS in Eq.(9) becomes equivalent to the FBF solution in (Ramani et al., 2017) when the regularization matrix is $\mathbf{P} = \mathbf{\Phi}^T\mathbf{\Phi}$.

Kernel-based regression takes a probabilistic approach to solving the regularized regression problem in Eq.9 where regularization is imposed by kernels. This section provides a brief description of kernel-based regression in the context of pseudo-inversion for tracking control; more in-depth theory can be found in (Pillonetto, Dinuzzo, Chen, De Nicolao, & Ljung, 2014) and (Rasmussen, 2003).

Let the input sequence $\mathbf{u}$ be a random process with zero-mean normal prior distribution and assume that the tracking error $\mathbf{e}$ is an independent and identically distributed zero-mean Gaussian process with covariance σ, expressed as follows:

$$\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \lambda\mathbf{K}_\beta), \text{ and } \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I}) \tag{10}$$

where $\mathbf{0}$ is a $E+1$ dimensional vector of zeros and $\lambda\mathbf{K}_\beta$ is a square $E+1$ dimensional symmetric positive definite covariance matrix, also known as *kernel* matrix. The output of the causal linear time-invariant system to such input is expressed by the discrete-time convolution sum in Eq.(7). The prior distribution of the joint probability of the output and input is therefore also normal, expressed as follows:

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{u}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma}_y & \mathbf{\Sigma}_{yu}\\ \mathbf{\Sigma}_{uy} & \lambda\mathbf{K}_\beta\end{bmatrix}\right) \tag{11}$$

where $\mathbf{\Sigma}_{yu} = \mathbf{\Sigma}_{uy}^T = \lambda\mathbf{G}\mathbf{K}_\beta$ and $\mathbf{\Sigma}_y = \lambda\mathbf{G}\mathbf{K}_\beta\mathbf{G}^T + \sigma^2\mathbf{I}$.

By applying Bayes' rule to joint Gaussian distributions, the associated conditional distribution can also be expressed as the following Gaussian:

$$p\left(\mathbf{u}|\mathbf{y}_d\right) = \mathcal{N}\left(\hat{\mathbf{u}}, \mathbf{\Sigma}_{u|y_d}\right) \tag{12}$$

where

$$\begin{aligned}\hat{\mathbf{u}} &= \left(\frac{\mathbf{G}^T\mathbf{G}}{\sigma^2} + (\lambda\mathbf{K}_\beta)^{-1}\right)^{-1}\frac{\mathbf{G}^T}{\sigma^2}\mathbf{y}_d \quad \text{and}\\ \mathbf{\Sigma}_{u|y_d} &= \left(\frac{\mathbf{G}^T\mathbf{G}}{\sigma^2} + (\lambda\mathbf{K}_\beta)^{-1}\right)^{-1}\end{aligned} \tag{13}$$

Equation (13) is further simplified by assuming $\lambda = \frac{\sigma^2}{\alpha}$:

$$\hat{\mathbf{u}} = \left(\mathbf{G}^T\mathbf{G} + \left(\frac{1}{\alpha}\mathbf{K}_\beta\right)^{-1}\right)^{-1}\mathbf{G}^T\mathbf{y}_d \tag{14}$$

It is easy to verify that Eq.(14) reduces to Eq. (9) when $\mathbf{K}_\beta = \mathbf{P} = \mathbf{\Phi}^T\mathbf{\Phi}$. Therefore, when the kernel matrix is created from the basis functions, the Maximum A Posteriori (MAP) estimate of the kernel-based input becomes equal to the input estimated by the

FBF method. Also, both the kernel matrix and associated basis functions perform the same job as the regularization matrix in the ReLS method. Despite these similarities, regression with kernels offers important advantages in designing the optimal input sequence because any positive-definite matrix can be used as a kernel without necessarily knowing the corresponding basis functions. This enables a much wider search domain for feasible solutions than regression with basis functions. Besides, the number of hyperparameters in the kernel-based method is usually much fewer than those in the FBF method, as will be shown in Section **??**.

2.3. Limited preview

Despite the discussed advantages of the kernel-based approach, the computational cost of inverting the $E+1$ dimensional square kernel matrix $\mathbf{K}_\beta$ in Eq.14 could make this method impractical. In this section, we present a recursive variant of the kernel-based method, where the input trajectory is computed recursively in batches of L_w time samples. While the computational cost of the kernel-based approach increases cubically by the size of the trajectory (i.e. $O(E^3)$), it only increases linearly in the recursive method. Besides, because the method is recursive and only requires the knowledge of the desired trajectory at the current batch of time samples, it can be used for online limited-preview implementation.

Suppose the total desired trajectory is divided into consecutive segments of L_w time samples with L_o overlapping samples. The goal is to compute the control input for the current segment, $\mathbf{u}_c \in \mathbb{R}^{L_w}$, based on the control input in the preceding segments, $\mathbf{u}_p \in \mathbb{R}^H$, and the desired trajectory for the current segment, $\mathbf{y}_c \in \mathbb{R}^{L_w}$:

$$\mathbf{u}_c = f(\mathbf{u}_p, \mathbf{y}_c) \tag{15}$$

where we assume the number of significant delays in $g(t)$ to be $H+1$ and f is a linear map from $\mathbb{R}^H \times \mathbb{R}^{L_w}$ to $\mathbb{R}^{L_w}$.

Similar to Section 2.2, let $[\mathbf{u}_p^T, \mathbf{u}_c^T]^T$ be zero-mean Gaussian random vector with $\lambda\tilde{\mathbf{K}}_\beta \in \mathbb{R}^{H+L_w}$ covariance matrix:

$$\begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_c \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \lambda\tilde{\mathbf{K}}_\beta \right) \tag{16}$$

Then, $\mathbf{y}_c$ can be expressed as follows:

$$\begin{gathered} \mathbf{y}_c = \tilde{\mathbf{G}} \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_c \end{bmatrix} + \mathbf{e}_c \\ \tilde{\mathbf{G}} = \begin{bmatrix} g(H) & \cdots & g(0) & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & g(H) & \cdots & g(1) & g(0) & 0 & 0 \\ \cdots & \cdots & g(H) & \cdots & g(1) & g(0) & 0 \\ \cdots & \cdots & \cdots & g(H) & \cdots & g(1) & g(0) \end{bmatrix} \end{gathered} \tag{17}$$

where $\mathbf{e}_c$ is the tracking error in the current window, which is also assumed to be independent and identically distributed zero-mean Gaussian process with covariance σ. Because of the linear relationship between $\mathbf{y}_c$ and input commands in Eq.17, the joint

distribution of $[\mathbf{u}_c^T, \mathbf{u}_p^T, \mathbf{y}_c^T]^T$ is also zero mean Gaussian with the following covariance:

$$\begin{bmatrix} \mathbf{u}_c \\ \mathbf{u}_p \\ \mathbf{y}_c \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{\Sigma}_2 \\ \mathbf{\Sigma}_2^T & \mathbf{\Sigma}_3 \end{bmatrix}\right) \tag{18}$$

where

$$\begin{aligned} \mathbf{\Sigma}_1 &= \mathbf{K}_3 \\ \mathbf{\Sigma}_2 &= \begin{bmatrix} \mathbf{K}_2, & \begin{bmatrix} \mathbf{K}_2 & \mathbf{K}_3 \end{bmatrix} \tilde{\mathbf{G}}^T \end{bmatrix} \\ \mathbf{\Sigma}_3 &= \begin{bmatrix} \mathbf{K}_1, & \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix} \tilde{\mathbf{G}}^T \\ \tilde{\mathbf{G}} \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_2 \end{bmatrix}^T & , \tilde{\mathbf{G}}\mathbf{K}\tilde{\mathbf{G}}^T + \sigma^2 \mathbf{I}_{L_w \times L_w} \end{bmatrix} \end{aligned} \tag{19}$$

and $\mathbf{K}_1 \in \mathbb{R}^{H \times H}$, $\mathbf{K}_2 \in \mathbb{R}^{H \times L_w}$, and $\mathbf{K}_3 \in \mathbb{R}^{L_w \times L_w}$ are the submatrices of $\tilde{\mathbf{K}}_\beta$:

$$\lambda \tilde{\mathbf{K}}_\beta = \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_2 \\ \mathbf{K}_2^T & \mathbf{K}_3 \end{bmatrix} \tag{20}$$

Following Baye's rule for joint Gaussian distribution, the posterior distribution of the input command $\mathbf{u}_c$ is obtained as follows:

$$p(\mathbf{u}_c \mid [\mathbf{u}_p^T, \mathbf{y}_c^T]^T) = \mathcal{N}(\hat{\mathbf{u}}_c, \mathbf{\Sigma}_{u_c|uy}) \tag{21}$$

where

$$\begin{aligned} \hat{\mathbf{u}}_c &= \mathbf{\Sigma}_2 \mathbf{\Sigma}_3^{-1} \begin{bmatrix} \mathbf{u}_p \\ \mathbf{y}_c \end{bmatrix} \\ \mathbf{\Sigma}_{u_c|uy} &= \mathbf{\Sigma}_1 - \mathbf{\Sigma}_2 \mathbf{\Sigma}_3^{-1} \mathbf{\Sigma}_2^T \end{aligned} \tag{22}$$

The $\mathbf{\Sigma}_2 \mathbf{\Sigma}_3^{-1}$ term in Eq.22 is a $L_w \times (L_w + H)$ constant matrix that is computed once at the beginning and used at every recursive estimation of $\hat{\mathbf{u}}_c$. Therefore, the computation cost does not increase by increasing the size of the total trajectory. The input sequence for the first segment (where $\mathbf{u}_p$ is unknown) is simply obtained by setting $E = H - 1$ in Eq.14. Figure 2 shows a schematic of updating $\mathbf{u}_p$ and $\mathbf{u}_c$ vectors

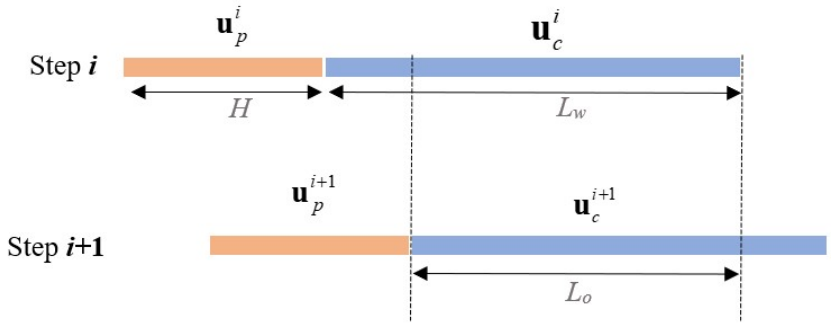

Figure 2. The schematic of the moving preview, $\mathbf{u}_p$, and $\mathbf{u}_c$ vectors in two consecutive steps

in two consecutive preview windows. At each recursion, only the first $L_w - L_o$ entries of the newly computed $\mathbf{u}_c$ vector are added to the next step's $\mathbf{u}_p$ vector. Due to the

Table 1. Parameters of the system transfer function in the three studied examples

Example	p	K	a
1	0.5	-0.5	2
2	0.5	-500	1.001
3	0.5	0.25	-1

Table 2. Parameters of full-preview and limited-preview formulations

parameter	Value
Binary sequence band	[0,1]
Binary sequence limits	$\pm 10^5 mm/s^2$
E (Full-preview)	100
α (Full-preview Example 1)	$1E-5$
α (Full-preview Example 2)	$1E-8$
α (Full-preview Example 3)	$1E-14$
σ^2 (Limited-preview)	$1E-5$
E (Limited-preview)	$1E+4$
L_w	50
H	50
L_o	45

causality of the system, each entry in $\mathbf{u}_c$ is only determined by the succeeding entries in the corresponding $\mathbf{y}_c$, to the point that the last entry of $\mathbf{u}_c$ is only affected by the last entry of $\mathbf{y}_c$. As a result, the estimation variance becomes progressively larger at the later entries of $\mathbf{u}_c$. Hence, we keep the first $L_w - L_o$ entries of $\mathbf{u}_c$ in each step and discard the rest.

3. Results and discussion

In this section, both full and limited-preview kernel-based methods are used to determine the input sequence for benchmark NMP systems from (Ramani et al., 2017) and (Butterworth et al., 2012). All of the examples are first-order discrete-time systems with $T = 0.1$ ms time intervals and a pole at 0.5. All of the systems are NMP with zeros at various distances from the unit circle, which also include nonhyperbolic and near hyperbolic cases. The plant transfer function is expressed as follows, and the corresponding parameters for each example are shown in Table 1:

$$G(z) = K\frac{z-a}{z-p} \tag{23}$$

The acceleration profile of the desired trajectory is a pseudorandom binary sequence in Matlab 2022 whose parameters are listed in table 2 along with other parameters related to the full-preview and limited-preview formulations. The velocity and position profiles of the desired trajectory are obtained by numerically integrating acceleration.

Linear spline kernel function is used to regularize the input estimation problem:

$$K(i,j) = \min\{i,j\} \tag{24}$$

where $K(i,j)$ is the (i,j) entry of the kernel matrix $\mathbf{K}_\beta$. The control input could also

Table 3. Parameters of full-preview (FP) and limited-preview (LP) formulations of FBF method

parameter	Value
n (Number of control points in FP)	E
L (ratio of time steps to control points in LP)	10
L_H (Previous window time step size in LP)	20
L_c (Current window time step size in LP)	200
n_{up} (Number of updated control points in each step in LP)	2

be smoothed by cubic spline functions by using the following kernel:

$$K(i,j) = \frac{ij\min\{i,j\}}{2} - \frac{(\min\{i,j\})^3}{6} \tag{25}$$

See (Pillonetto et al., 2014) for more details about the kernels in Eqs.(24) and (25). A wide range of kernels could also be used without necessarily knowing their corresponding basis functions (e.g. see Chapter 4 in (Rasmussen, 2003)). Each kernel function would typically include a few (design) hyperparameters that must be determined by optimization. The spline kernel functions in Eqs.(24) and (25) do not include any hyperparameters and therefore the only design parameter is regularization coefficient, α in Eq.(14). In this paper, α was selected by manual tuning to minimize tracking error in all three examples. Also, the results are compared with the full-preview and limited-preview versions of FBF method introduced in (Duan et al., 2018) and (Ramani et al., 2017). Spline basis function is used for limited preview FBF, and Discrete Cosine Transform (DCT) is used for full preview FBF. The related parameters for each method are listed in table 3.

3.1. Full-preview implementation

In this section, the full-preview kernel-based estimate in Eq.14 is used to determine the input sequence for each example, and the results are compared with ReLS (Eq. 9) and FBF method. To study the regularizing effect of kernels, the ReLS method with unity regularization matrix $\mathbf{P} = \mathbf{I}$ is used. Regularization coefficient α in Eq.(9) was selected by manual tuning to minimize tracking error. The value of α for each example is shown in Table 2. The resulting input sequences and the corresponding tracking errors for three examples are shown in Figs.3 to 5, respectively. As shown in Figs.3 to 5, tracking error is minimized by ReLS, kernel-based, and FBF methods for all three examples. In example 1, all three methods show similar performances with the difference that the input sequences from ReLS and FBF increase sharply at the end of the trajectory while the kernel-based input remains smooth. This difference is due to the smoothing effect of the linear spline kernel matrix. Example 2 with the positive near hyperbolic zero is the most challenging system to control, which is controlled efficiently with all of the methods. The negative nonhyperbolic zero in Example 3 causes fluctuations in the input profiles in all of the methods, but the output trajectory is tracked properly. Increasing the number of control points in the FBF method reduces oscillations of the input sequence but increases the tracking error. The Root Mean Square (RMS) of tracking error in Example 1 is similar for all three methods. Tracking error in the kernel-based method could be further improved by reducing the regularization coefficient; however, this would yield a less smooth input sequence.

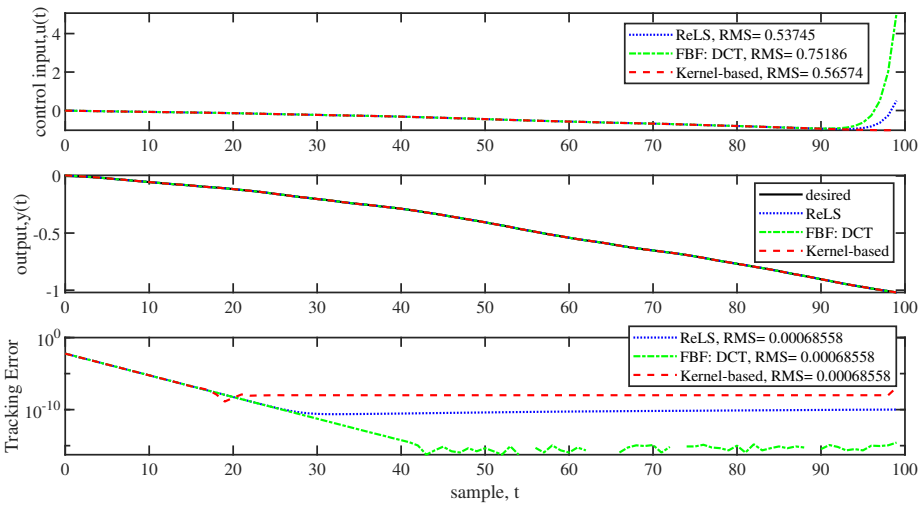

Figure 3. Example 1, zero at $a = 2$, Full preview; Comparison of the generated control inputs, predicted outputs, and tracking errors of the ReLS, FBF, and Kernel-based methods

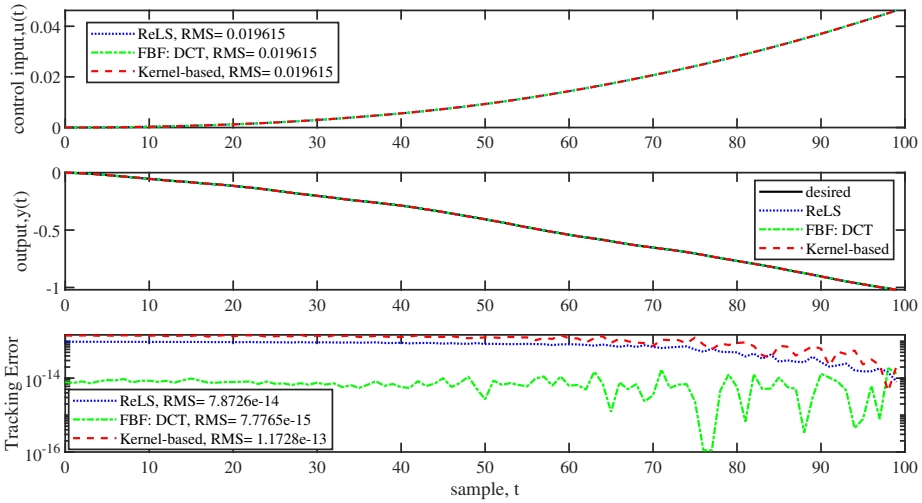

Figure 4. Example 2, zero at $a = 1.001$, Full preview; Comparison of the generated control inputs, predicted outputs, and tracking errors of the ReLS, FBF, and Kernel-based methods

3.2. Recursive limited-preview implementation

In this section, input trajectories for the three examples are computed recursively using Eq.22, and the results are compared with the limited-preview FBF in (Duan et al., 2018). Linear spline kernels are used in this section as well. Compared to the full-preview examples, one hundred times more samples were generated to demonstrate the computational efficiency of the recursive method (i.e. E=10,000 instead of 100). Despite this substantial increase in the length of the trajectory, the computation time does not change noticeably compared to the batch computation of the 100-samples in the previous section. The resulting input sequences and the corresponding trajectories and tracking errors are shown in Figs.6 to 8. In examples 1 and 3, the recursive

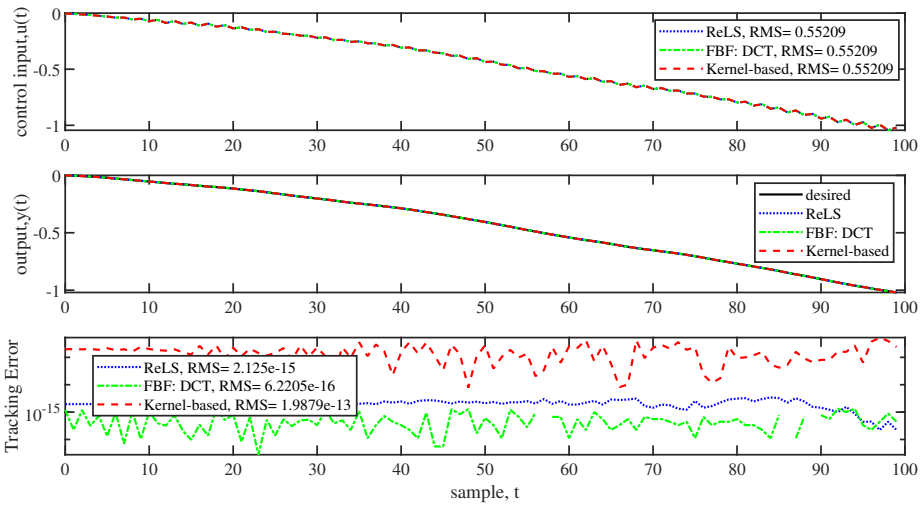

Figure 5. Example 3, zero at $a = -1$, Full preview; Comparison of the generated control inputs, predicted outputs, and tracking errors of the ReLS, FBF, and Kernel-based methods

method generates stable input commands and tracking errors are comparable to the full-preview method. They are also less than the tracking error obtained by applying the limited-preview FBF method. In example 2, where the zero is adjacent to the unit circle, the tracking error becomes unstable for both kernel-based and FBF methods. A longer trajectory had to be used in this example to demonstrate the divergence of tracking error in the kernel-based method.

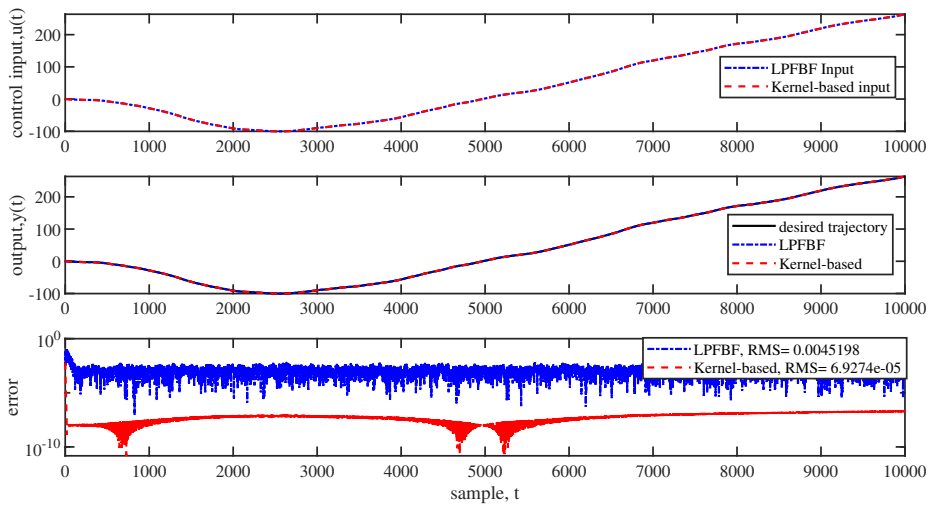

Figure 6. Example 1, zero at $a = 2$, Limited preview; Comparison of the generated control inputs, predicted outputs, and tracking errors of the ReLS, FBF, and Kernel-based methods

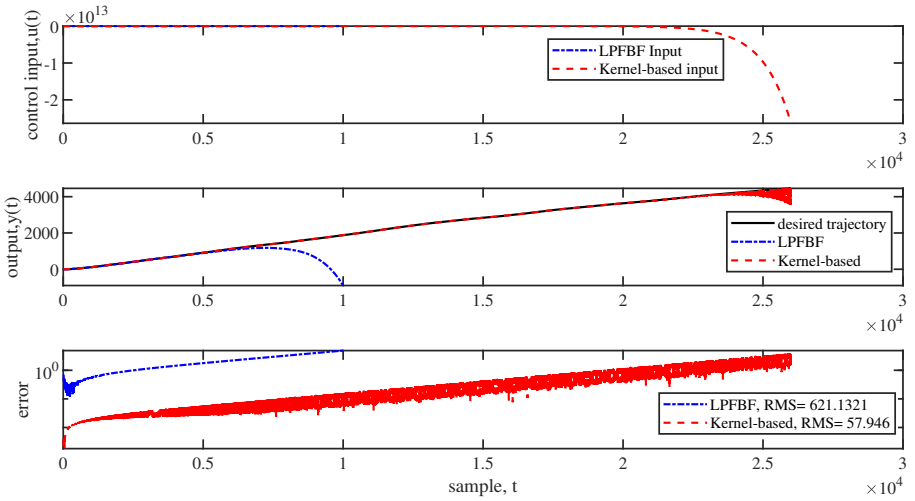

Figure 7. Example 2, zero at $a = 1.001$, limited preview; Comparison of the generated control inputs, predicted outputs, and tracking errors of the ReLS, FBF, and Kernel-based methods

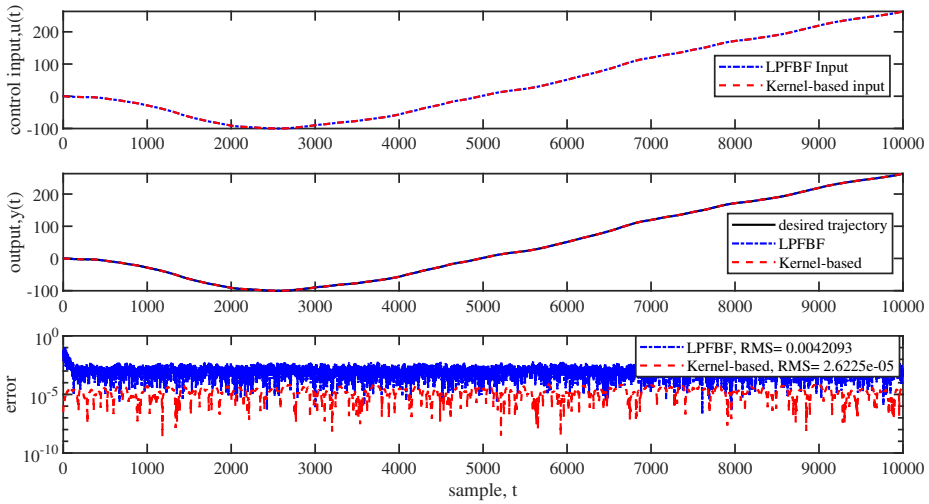

Figure 8. Example 3, zero at $a = -1$, limited preview; Comparison of the generated control inputs, predicted outputs, and tracking errors of the ReLS, FBF, and Kernel-based methods

3.3. The effect of design parameters on tracking error and recursion stability

Here, we study the influence of various parameters of the algorithm on its performance and stability. These parameters consist of the window parameters in recursive formulation, i.e. H, $L = L_w - L_o$, and L_w, and the hyperparameter σ^2. In each of the following figures, only one of the design parameters is varied while the rest remain at the values shown in Table 2. Figure 9 (Top left) shows the RMS tracking errors for 6 values of L and a set of zero values between -2 and +2. L is the number of retained elements after computing the input vector at each recursion, and $L_w - L$ is the number of overlapping elements. As shown in Fig. 9 (top left), increasing L (shorter overlaps) increases the

tracking error. This is because fewer output entries contribute to estimating the last entries of the input batch. The error tends to infinity for zeros near 1 due to the instability of the algorithm. In those cases, the greatest eigenvalue of the transition matrix also becomes greater than 1. Increasing the overlap between segments decreases the error and improves stability. Nonetheless, no significant improvement in stability is observed for L below 30 samples.

Figure 9 (top right) shows the RMS tracking error versus zero locations for different values of H, the number of considered past inputs. Changing H does not affect stability, but increasing H leads to lowering the tracking error. There is no significant change in error for $H > 40$ because the impulse response diminishes after that.

Figure 9 (bottom left) shows the influence of window size L_w on the RMS tracking error for various zero locations. As expected, a larger window size leads to a lower error, and also, the effect of L_w on stability is more significant than other parameters. In fact, for $L_w = 10$ the algorithm becomes unstable for all of the systems with zeros between 1 and 2.

The 3-dimensional plot in Fig. 9 (bottom right) shows the relationship between the maximum eigenvalues of the transition matrix and the hyperparameter σ^2 for a set of systems with different zeros. The regularization hyperparameter σ^2 plays a major role in the stability of the system. As the figure shows, in the absence of regularization ($\sigma^2 = 0$) two regions near -1 and 1 are unstable. However, by adding regularization, we can stabilize the algorithm for the systems with zero near -1 and a wider range of systems with zeros near +1.

An unlimited range of kernel types could be used with this algorithm, some of which may be more effective in stabilizing it. Using kernels with more hyperparameters will make the design more flexible and widen the range of systems that could be controlled by this algorithm. This, however, would increase the computational effort needed for tuning those hyperparameters.

4. Conclusions

A kernel-based method was presented for designing optimal input sequences for NMP systems. We showed that the function of kernels in this method is closely related to the basis functions in the FBF method. If the basis functions are known, the equivalent kernels can be easily determined; however, any positive definite kernel can also be used without knowing the associated basis functions. This property makes the kernel-based approach more powerful in searching for bounded inputs for NMP systems with arbitrary zeros and desired trajectories.

An important advantage of the presented kernel-based approach over the FBF method is its flexibility and ease of use for regularizing (or smoothing) the estimated input sequence. In FBF, regularization is controlled by the type of basis functions, the number of included basis functions, and their corresponding filter initial conditions. In the kernel-based approach, regularization is controlled by the kernel type and its hyperparameters, which are typically a few. For instance, the spline kernel used in this work does not have any hyperparameters. Therefore, regularization with kernels seems easier than with their corresponding basis functions. Moreover, virtually unlimited types of positive definite functions are available that can be used for regularization without necessarily knowing their corresponding basis functions. Plus, new kernels can also be generated by multiplying or adding others.

A recursive version of the kernel-based method was also presented for online limited-

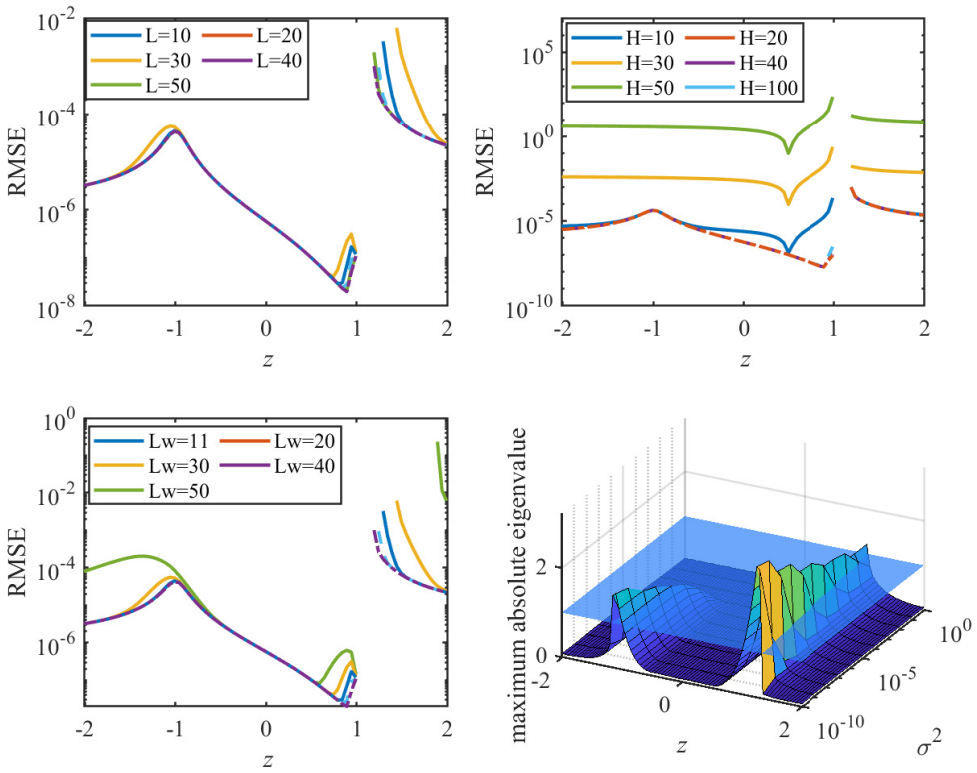

Figure 9. Top left: The effect of updating and overlapping elements on the RMSE in different zero locations, Top right: The effect of preceding elements on the RMSE in different zero locations, Bottom left: The effect of current window elements on the RMSE in different zero locations, Bottom right: The effect of the hyperparameter σ^2 (regularization parameter) on the stability and the maximum absolute value of the transition matrix eigenvalues in different zero locations

preview implementation. The method's computational efficiency in designing long trajectories was demonstrated in several examples, and its performance was compared with limited-preview FBF method. The recursive and full-preview methods resulted in comparable tracking errors, but the tracking error becomes unstable for both kernel-based and FBF methods if the zero is close to +1.

Hyperparameters in the kernel-based method must be tuned by optimization. In common regression problems where observations are stochastic, hyperparameters are usually determined by minimizing cross-validation errors or maximizing the marginal likelihood of the observed data. In tracking error minimization, due to the deterministic nature of the observed data (i.e. desired trajectory), cross-validation or marginal likelihood cannot be used for tuning hyperparameters. Instead, a new cost function comprising tracking error and control effort could be minimized to determine optimal hyperparameters.

Disclosure Statement

The authors report there are no competing interests to declare

Funding

This work was supported by the National Research Council Canada under Grant number DHGA-108-1.

Data Availability

All the codes and data that support the findings of this study are openly available in DDM_UVIC repository at https://github.com/DDM-UVIC/Tracking-Control-of-Non-minimum-Phase-Systems-A-Kernel-based-Approach

Notes on contributors

Mohammadmahdi Mehrabi is a Ph.D. student in Mechanical Engineering at the University of Victoria, Canada. He received his Bachelors's and Master's degrees in Mechanical Engineering from Iran University of Science and Technology and Sharif University of Technology, respectively. His research is focused on system identification, mechanical vibrations, and pattern recognition.

Keivan Ahmadi is an Associate Professor in the Mechanical Engineering Department at the University of Victoria (UVic). He completed his Ph.D. at the University of Waterloo, followed by postdoctoral fellowships at the University of British Columbia (UBC) and Pratt and Whitney Canada. His research focuses on the dynamics of machining processes and machine tool vibrations.

References

AAström, K., Hagander, P. & Sternby, J. (1984). Zeros of sampled systems. *Automatica.* **20**, 31-38

Biagiotti, L., Califano, F. & Melchiorri, C. (2016). Repetitive control of non-minimum phase systems along B-spline trajectories. *2016 IEEE 55th Conference On Decision And Control (CDC).* pp. 5496-5501

Biagiotti, L., Califano, F. & Melchiorri, C. (2019). Repetitive control meets continuous zero phase error tracking controller for precise tracking of B-spline trajectories. *IEEE Transactions On Industrial Electronics.* **67**, 7808-7818

Blanken, L., Meijdenberg, I. & Oomen, T. (2018). Inverse system estimation for feedforward: A kernel-based approach for non-causal systems. *IFAC-PapersOnLine.* **51**, 1050-1055

Blanken, L. & Oomen, T. (2020). Kernel-based identification of non-causal systems with application to inverse model control. *Automatica.* **114** pp. 108830

Bolderman, M., Lazar, M. & Butler, H. (2021). Physics–guided neural networks for inversion–based feedforward control applied to linear motors. *2021 IEEE Conference On Control Technology And Applications (CCTA).* pp. 1115-1120

Butterworth, J., Pao, L. & Abramovitch, D. (2012). Analysis and comparison of three discrete-time feedforward model-inverse control techniques for nonminimum-phase systems. *Mechatronics.* **22**, 577-587

Butterworth, J., Pao, L. & Abramovitch, D. (2008). Architectures for tracking control in atomic force microscopes. *IFAC Proceedings Volumes.* **41**, 8236-8250

Butterworth, J., Pao, L. & Abramovitch, D. (2011). Fitting discrete-time models to frequency responses for systems with transport delay. *ASME International Mechanical Engineering Congress And Exposition.* **54938** pp. 1321-1328

Chou, C., Duan, M. & Okwudire, C. (2021). A Hybrid Filtered Basis Functions Approach for Tracking Control of Linear Systems with Unmodeled Nonlinear Dynamics. *2021 IEEE 17th International Conference On Automation Science And Engineering (CASE)*. pp. 1176-1181

Dai, L., Li, X., Zhu, Y. & Zhang, M. (2019). Quantitative analysis on tracking error under different control architectures and feedforward methods. *2019 American Control Conference (ACC)*. pp. 5680-5686

Devasia, S., Chen, D. & Paden, B. (1996). Nonlinear inversion-based output tracking. *IEEE Transactions On Automatic Control*. **41**, 930-942

Duan, M., Yoon, D. & Okwudire, C. (2018). A limited-preview filtered B-spline approach to tracking control–With application to vibration-induced error compensation of a 3D printer. *Mechatronics*. **56** pp. 287-296

Duan, M., Ramani, K. & Okwudire, C. (2015). Tracking control of non-minimum phase systems using filtered basis functions: a NURBS-based approach. *Dynamic Systems And Control Conference*. **57243** pp. V001T03A006

Fujimoto, H., Hori, Y. & Kawamura, A. (2001). Perfect tracking control based on multirate feedforward control with generalized sampling periods. *IEEE Transactions On Industrial Electronics*. **48**, 636-644

Gopalswamy, S. & Karl Hedrick, J. (1993). Tracking nonlinear non-minimum phase systems using sliding control. *International Journal Of Control*. **57**, 1141-1158

Gross, E. & Tomizuka, M. (1994). Experimental Flexible Beam Tip Tracking Control with a Truncated Series Approximation to Uncancelable Inverse Dynamics. *IEEE Transactions On Control Systems Technology*. **2**, 382-391

Hunt, L., Meyer, G. & Su, R. (1996). Noncausal inverses for linear systems. *IEEE Transactions On Automatic Control*. **41**, 608-611

Lee, H. & Kim, H. (2017). Trajectory tracking control of multirotors from modelling to experiments: A survey. *International Journal Of Control, Automation And Systems*. **15**, 281-292

Marconi, L., Marro, G. & Melchiorri, C. (2001). A solution technique for almost perfect tracking of non-minimum-phase, discrete-time linear systems. *International Journal Of Control*. **74**, 496-506

Miu, D. (2001). Mechatronics: electromechanics and contromechanics. (Springer Science & Business Media)

Okwudire, C., Ramani, K. & Duan, M. (2016). A trajectory optimization method for improved tracking of motion commands using CNC machines that experience unwanted vibration. *CIRP Annals*. **65**, 373-376

Piazzi, A. & Visioli, A. (2005). Using stable input–output inversion for minimum-time feedforward constrained regulation of scalar systems. *Automatica*. **41**, 305-313

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G. & Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*. **50**, 657-682

Ramani, K., Duan, M., Okwudire, C. & Galip Ulsoy, A. (2017). Tracking control of linear time-invariant nonminimum phase systems using filtered basis functions. *Journal Of Dynamic Systems, Measurement, And Control*. **139**

Rasmussen, C. (2003). Gaussian processes in machine learning. *Summer School On Machine Learning*. pp. 63-71

Rigney, B., Pao, L. & Lawrence, D. (2009). Nonminimum phase dynamic inversion for settle time applications. *IEEE Transactions On Control Systems Technology*. **17**, 989-1005

Risuleo, R., Bottegal, G. & Hjalmarsson, H. (2015). On the estimation of initial conditions in kernel-based system identification. *2015 54th IEEE Conference On Decision And Control (CDC)*. pp. 1120-1125

Romagnoli, R. & Garone, E. (2019). A general framework for approximated model stable inversion. *Automatica*. **101** pp. 182-189

Sharma, K. & Pradhan, R. (2017). Design of a novel feed-forward control strategy for a non-minimum phase system. *IOP Conference Series: Materials Science And Engineering*. **225**, 012119

Tomizuka, M. (1987). Zero phase error tracking algorithm for digital control.

Zhou, S., Helwa, M. & Schoellig, A. (2018). An inversion-based learning approach for improving impromptu trajectory tracking of robots with non-minimum phase dynamics. *IEEE Robotics And Automation Letters*. **3**, 1663-1670

Zou, Q. & Devasia, S. (1999). Preview-based stable-inversion for output tracking of linear systems.

Zundert, J. & Oomen, T. (2018). On inversion-based approaches for feedforward and ILC. *Mechatronics*. **50** pp. 282-291