Robust and Kernelized Data-Enabled Predictive Control for Nonlinear Systems

Linbin Huang, Member, IEEE, John Lygeros, Fellow, IEEE, and Florian Dörfler, Senior Member, IEEE

Abstract—This article presents a robust and kernelized data-enabled predictive control (RoKDeePC) algorithm to perform model-free optimal control for nonlinear systems using only input and output data. The algorithm combines robust predictive control and a multistep predictor of nonlinear systems obtained from regularized kernel methods. The latter is based on implicitly learning the nonlinear behavior of the system via the representer theorem. Instead of seeking a model and then performing control design, our method goes directly from data to control. This allows us to robustify the control inputs against the uncertainties in data by considering a min–max optimization problem to calculate the robust and optimal control sequence. We show that by incorporating an appropriate uncertainty set, this min–max problem can be reformulated as a nonconvex but structured minimization problem. By exploiting its structure, we present a projected gradient descent algorithm to effectively solve this problem. Finally, we test the RoKDeePC method on two nonlinear example systems—one academic case study and a grid-forming converter feeding a nonlinear load—and compare it with some existing nonlinear data-driven predictive control methods.

Index Terms—Data-driven control, kernel methods, nonlinear control, predictive control, robust optimization.

I. INTRODUCTION

DATA-DRIVEN control has attracted extensive attention in recent years, as it enables optimal control in scenarios where data are readily available, but the system models are too complex to obtain or maintain [1], [2]; this is often the case, for example, in large-scale energy systems or robotics [3], [4].

One standard approach to data-driven control is indirect, where one first uses data to identify a system model and then performs control design based on the identified model. This paradigm has a long history in control theory, leading to the developments of system identification (SysID) [5], model predictive control (MPC) [6], and so on. Prediction error, maximum likelihood, and subspace methods are popular SysID approaches and have been successfully combined with model-based control design in many industrial applications. Broadly recognized challenges of the indirect approach are incompatible uncertainty estimates and the customization of SysID (e.g., in terms of objective and model order selection) for the ultimate control objective. We refer to [7], [8], [9], and [10] for different approaches. Recent advances in systems theory, statistics, machine learning, and deep learning have introduced many promising techniques to learn the behavior of general dynamical systems from data, e.g., by using Koopman operators, Gaussian processes, kernel methods, and neural networks [11], [12], [13], [14], [15], [16], [17]. The learned models can also be combined with model-based control design (e.g., MPC) to perform optimal control [18], [19], [20], [21], [22], [23]. However, these methods still belong to indirect data-driven control, and one may not be able to provide deterministic guarantees on the actual control performance, especially when complicated structures, e.g., neural networks, are used to learn the system’s behaviors. We note that compared with using neural networks, function estimation using regularized kernel methods has a tractable formulation and solution because of the well posedness of the function classes in reproducing kernel Hilbert spaces (RKHSs) [13], [24].

An alternative formulation is direct data-driven control that aims to go directly from data to control, bypassing the identification step. In recent years, a result formulated by Willems et al. [25] in the context of behavioral system theory, known as the fundamental lemma, has been widely used in formulating direct data-driven control methods, such as [1], [26], [27], and [28]. The fundamental lemma shows that the subspace of input–output trajectories of a linear time-invariant (LTI) system can be obtained from the column span of a data Hankel matrix, which acts as a data-centric representation of the system dynamics. Unlike indirect data-driven control methods, which usually involve a bilevel optimization problem (an inner problem for the SysID step and an outer problem for the control design step), these direct data-driven control methods formulate one single optimization problem to obtain control policy directly from data, which circumvents an explicit step of SysID. In this paradigm, one can conveniently provide robust control against uncertainties in data via regularization and relaxation [9]. For instance, the data-enabled predictive control (DeePC) method uses regularization to robustify the control actions against uncertainties in data [26], [29], [30]. Moreover, performance guarantees on the realized input–output cost can be provided, given that the inherent uncertainty set is large enough [31]. Motivated in part by the desire to extend this direct data-driven approach beyond LTI systems, fundamental...
lemma has been extended for certain classes of nonlinear systems, e.g., Hammerstein–Wiener systems [32], second-order discrete Volterra systems [33], flat nonlinear systems [34], and polynomial time-invariant systems [35], which enables the control design of these classes of nonlinear systems.

In this article, we develop a direct data-driven method for nonlinear systems, referred to as robust and kernelized DeePC (RoKDeePC), which combines kernel methods with a robustified DeePC framework. We use regularized kernel methods [24] to implicitly learn the future behavior of nonlinear systems, and then directly formulate a min–max optimization problem to obtain robust and optimal control sequences. Note that our work focuses on learning the nonlinear predictors rather than developing a nonlinear version of fundamental lemma. We prove that when a weighted two norm is considered in the cost of future outputs, the realized cost of the system is upper bounded by the optimization cost of this optimization problem, leading to deterministic performance guarantees. Furthermore, we demonstrate how this min–max problem can be formulated as a nonconvex yet structured minimization problem when appropriate uncertainty sets are considered. To enable a real-time implementation, we develop a projected gradient descent algorithm exploiting the problem structure to effectively solve the RoKDeePC problem. Unlike learning-based control using neural networks, our method does not require a time-consuming offline learning process and has real-time plug-and-play ability to handle nonlinear systems. We test RoKDeePC on two nonlinear example systems—one academic case study and a grid-forming converter feeding nonlinear systems. The rest of this article is organized as follows. In Section II, we review the linear predictor and then introduce the kernel-based nonlinear predictor using input–output data. Section III proposes the RoKDeePC algorithm and demonstrates its performance guarantees. Section IV develops kernel-based nonlinear predictor using input–output data. We conclude this article in Section VI.

A. Explicit Predictors for LTI Behavior

Consider a discrete-time LTI system

\[ \begin{align*}
    x_{i+1} &= A x_i + B u_i \\
    y_i &= C x_i + D u_i
\end{align*} \]

where \( A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{p \times n}, D \in \mathbb{R}^{p \times m}, \) \( x_i \in \mathbb{R}^n \) is the state of the system at \( t \in \mathbb{Z}_{\geq 0}, u_i \in \mathbb{R}^m \) is the input vector, and \( y_i \in \mathbb{R}^p \) is the output vector. Recall the extended observability matrix

\[ \mathcal{O}_L(A, C) := \text{col}(C, CA, \ldots, CA^{\ell-1}) \]

The lag of system (1) is defined by the smallest integer \( \ell \in \mathbb{Z}_{\geq 0} \), such that the observability matrix \( \mathcal{O}_L(A, C) \) has rank \( n \), i.e., the state can be reconstructed from \( \ell \) measurements. Consider \( L, T \in \mathbb{Z}_{\geq 0} \) with \( T \geq L > \ell \) and length-\( T \) input and output trajectories of (1):

\[ u = \text{col}(u_0, u_1, \ldots, u_{T-1}) \in \mathbb{R}^{mT} \]

and

\[ y = \text{col}(y_0, y_1, \ldots, y_{T-1}) \in \mathbb{R}^{pT} \]

For the inputs \( u \), define the Hankel matrix of depth \( L \) as follows:

\[ \mathcal{H}_L(u) := \begin{bmatrix}
    u_0 & u_1 & \cdots & u_{T-L} \\
    u_1 & u_2 & \cdots & u_{T-L+1} \\
    \vdots & \vdots & \ddots & \vdots \\
    u_{L-1} & u_L & \cdots & u_{T-1}
\end{bmatrix} \]

Accordingly, for the outputs, define the Hankel matrix \( \mathcal{H}_L(y) \). Consider the stacked matrix

\[ \mathcal{H}_L(y, u) = \begin{bmatrix}
    \mathcal{H}_L(u) \\
    \mathcal{H}_L(y)
\end{bmatrix} \]

By [36, Corollary 19], the restricted behavior (i.e., the input–output behavior of trajectories of length \( L \)) equals the image of \( \mathcal{H}_L(y, u) \) if and only if rank(\( \mathcal{H}_L(y, u) \)) = \( mL + n \). This behavioral result can be leveraged for data-driven prediction and estimation as follows. Consider \( T_{\text{ini}}, N, T \in \mathbb{Z}_{\geq 0} \), as well as an input–output time series \( (u^d, y^d) \in \mathbb{R}^{mT} \) and \( y^d \in \mathbb{R}^{pT} \), so that rank(\( \mathcal{H}_{T_{\text{ini}}+N}(u^d, y^d) \)) = \( m(T_{\text{ini}} + N) + n \). Here, the superscript “\( d \)” denotes data collected offline, and the rank condition is met by choosing \( u^d \) to be persistently exciting of sufficiently high order and by assuming that the system is controllable and observable. The Hankel matrix \( \mathcal{H}_{T_{\text{ini}}+N}(u^d, y^d) \) can be partitioned into

\[ \begin{bmatrix}
    U_P \\
    U_F
\end{bmatrix} := \mathcal{H}_{T_{\text{ini}}+N}(u^d) \]

and

\[ \begin{bmatrix}
    \tilde{Y}_P \\
    \tilde{Y}_F
\end{bmatrix} := \mathcal{H}_{T_{\text{ini}}+N}(y^d) \]

where \( U_P \in \mathbb{R}^{mT_{\text{ini}} \times N}, U_F \in \mathbb{R}^{mN \times H}, \tilde{Y}_P \in \mathbb{R}^{pT_{\text{ini}} \times N}, \), \( \tilde{Y}_F \in \mathbb{R}^{pN \times H} \), and \( H_c = T - T_{\text{ini}} - N + 1 \). In the sequel, the data in the partition with subscript \( P \) (for “past”) will be used to implicitly estimate the initial condition of the system, whereas the data with subscript \( F \) will be used to predict the “future” trajectories. In this case, \( T_{\text{ini}} \) is the length of an initial trajectory measured in the immediate past during online operation, and \( N \) is the length of a predicted trajectory starting from the initial trajectory. The fundamental lemma shows that the image of \( \mathcal{H}_{T_{\text{ini}}+N}(u^d, y^d) \) spans all length-(\( T_{\text{ini}} + N \)) trajectories [25].

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
that, \( \text{col}(u_{\text{ini}}, \bar{y}_{\text{ini}}, y) \in \mathbb{R}^{(m+p)T_{\text{ini}}+N} \) is a trajectory of (1) if and only if there exists a vector \( g \in \mathbb{R}^{H} \), so that
\[
\begin{bmatrix}
U_p \\
\bar{Y}_p \\
U_f \\
\bar{Y}_f
\end{bmatrix}
g = \begin{bmatrix}
u_{\text{ini}} \\
\bar{y}_{\text{ini}} \\
u \\
y
\end{bmatrix}.
\]
(3)

The initial trajectory \( \text{col}(u_{\text{ini}}, \bar{y}_{\text{ini}}, y) \) can be thought of as setting the initial condition for the future (to-be-predicted) trajectory \( \text{col}(u, y) \). In particular, if \( T_{\text{ini}} \geq \ell \), for every given future input trajectory \( u \), the future output trajectory \( y \) is uniquely determined through (3) [37].

In this case, one can consider an explicit linear predictor (known as the subspace predictive control predictor [9], [38]).

\[
y = M\text{col}(u_{\text{ini}}, \bar{y}_{\text{ini}}, u)
\]
(4)

where \( M \in \mathbb{R}^{pN \times ((m+p)T_{\text{ini}}+m)N} \) is a linear mapping and can be calculated from data as follows:
\[
M = \bar{Y}_F \begin{bmatrix} U_p & \bar{Y}_p \\ U_f & \bar{Y}_f \end{bmatrix}^\dagger
\]
(5)
where \( \dagger \) denotes the Moore–Penrose pseudoinverse. Note that (5) is the solution of the following linear regression problem:
\[
\min_M \left\| \bar{Y}_F - M \begin{bmatrix} U_p & \bar{Y}_p \\ U_f & \bar{Y}_f \end{bmatrix}_F \right\|_F
\]
(6)

where \( \text{col}(U_{[p]}, \bar{Y}_{[p]}, U_{[f]}, \bar{Y}_{[f]}) \) can be considered as one sample trajectory of \( \text{col}(u_{\text{ini}}, \bar{y}_{\text{ini}}, u, y) \).

In addition to Hankel matrices, one can also use more input–output data to construct (Chinese) Page matrices, mosaic Hankel matrices, or trajectory matrices as data-driven predictors [36], [39], [40].

B. Explicit Kernel-Based Nonlinear Predictors

We now consider (time-invariant) nonlinear systems, where the future outputs are nonlinear functions of the initial trajectory and the future inputs
\[
y_{[1]} = f_i(u_{\text{ini}}, \bar{y}_{\text{ini}}, u), \quad i \in [1:pN]
\]
(7)

which, with a sufficiently large \( T_{\text{ini}} \), also coincides with the nonlinear autoregressive with exogenous inputs (NARX) model [41]. Notice that (7) is a multistep predictor, and we refer to [42] and [43] for an investigation on multistep predictors and state-space models (one-step predictors) in the context of predictive control. In what follows, we focus on how to find the nonlinear functions \( f_i \) that best fit the observed trajectories contained in \( \text{col}(U_p, \bar{Y}_p, U_f, \bar{Y}_f) \).

To reconcile flexibility of the functions class \( f_i \) and well poseness of the function estimation problem, we assume that the functions \( f_i \) are continuous and belong to an RKHS [44].

**Definition 1:** An RKHS over a nonempty set \( \mathcal{X} \) is a Hilbert space of functions \( f : \mathcal{X} \rightarrow \mathbb{R} \), such that for each \( x \in \mathcal{X} \), the evaluation functional \( E_x f := f(x) \) is bounded.

Note that in our case, the set \( \mathcal{X} \) associated with the RKHS of the nonlinear functions \( f_i \) in (7) is a subset of \( \mathbb{R}^{(m+p)T_{\text{ini}}+mN} \).

An RKHS is associated with a positive semidefinite kernel, called reproducing kernel, which encodes the properties of the functions in this space.

**Definition 2:** A symmetric function \( K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is called positive semidefinite kernel if, for any \( h \in \mathbb{N} \)
\[
\sum_{i=1}^{h} \sum_{j=1}^{h} \alpha_i \alpha_j K(x_i, x_j) \geq 0 \quad \forall (x_k, \alpha_k) \in (\mathcal{X}, \mathbb{R}), \quad k \in [1:h].
\]
The kernel section of \( K \) centered at \( x \) is \( K_x(\cdot) = K(x, \cdot) \), \( \forall x \in \mathcal{X} \).

According to the Moore–Aronszajn theorem [45], an RKHS is fully characterized by its reproducing kernel. To be specific, if a function \( f : \mathcal{X} \rightarrow \mathbb{R} \) belongs to an RKHS \( \mathcal{H} \) with kernel \( K \), there exists a sequence \( \alpha_i K_{x_i}(x) \), such that
\[
f(x) = \lim_{h \rightarrow \infty} \sum_{i=1}^{h} \alpha_i K_{x_i}(x)
\]
(8)

for some \( \alpha_i \in \mathbb{R}, x_i \in \mathcal{X} \), for all \( i \in [1:h] \). Note that \( h \) can possibly be a finite number. Moreover, the inner product \( \langle \cdot, \cdot \rangle \) in the RKHS \( \mathcal{H} \) can be expressed as follows:
\[
\langle f, g \rangle = \lim_{h \rightarrow \infty} \sum_{i=1}^{h} \sum_{j=1}^{h} \alpha_i \beta_j K(x_i, x_j)
\]

where \( f, g \in \mathcal{H} \), \( g(x) = \lim_{h \rightarrow \infty} \sum_{j=1}^{h} \beta_j K_{x_j}(x) \), and the induced norm of \( f \) is \( \|f\|_{\mathcal{H}}^2 = \lim_{h \rightarrow \infty} \sum_{i=1}^{h} \sum_{j=1}^{h} \alpha_i \alpha_j K(x_i, x_j) \). In this article, since we are interested in approximating continuous functions, without loss of generality, we assume continuous kernel \( K \) (i.e., \( K(\cdot, \cdot) \) is continuous on \( \mathcal{X} \times \mathcal{X} \)) [46], [47].

We consider now the function estimation problem for the nonlinear system in (7). Since every column in the data matrix \( \text{col}(U_p, \bar{Y}_p, U_f, \bar{Y}_f) \) is a trajectory of the system and, thus, an input–output sample for (7), for each \( i \in [1:pN] \), we search for the best (in a regularized least-square sense) \( f_i \) in an RKHS \( \mathcal{H} \) with kernel \( K \)
\[
\min_{f_i \in \mathcal{H}} \sum_{j=1}^{H_x} \left( \bar{Y}_{[f_i]}(x_j) - f_i(x_j) \right)^2 + \gamma \|f_i\|_{\mathcal{H}}^2
\]
(9)

where \( x_j = \text{col}(U_{[p]}, \bar{Y}_{[p]}, U_{[f]}, \bar{Y}_{[f]}) \). Note that to simplify the forthcoming developments, the formulation in (9) does not consider temporal correlation between different indices \( i \) and causality (i.e., \( y_t \) is not affected by \( u_t+1 \)). In other words, we treat each function in (7) as independent when using data to estimate them. One can further consider causality by restricting the arguments of \( f_i \) to be \( (u_{\text{ini}}, \bar{y}_{\text{ini}}, u_{[1:t]}), \) which will lead to more complicated formulations than those in this article, as different functions \( f_i \) are defined over sets with different dimensions.

The cost function in (9) includes the prediction errors and a regularization term, which is used to avoid overfitting and penalize undesired behaviors. We note that since the functions in an RKHS can be parameterized as in (8), the representer theorem [24] ensures that the minimization problem in (9) can be solved in closed form. In our context, this leads to the following standard result of kernel methods [48].
Lemma 1: The minimizer of (9) admits a closed-form prediction model for (7) as follows:

\[ y = \bar{Y}_F (K + y I)^{-1} k(u_{ini}, \bar{y}_{ini}, u) \tag{10} \]

where \( \bar{K} \in \mathbb{S}^{N^2 \times N^2} \) is the Gram matrix, such that \( \bar{K}_{ij} = K(x_i, x_j) \) and \( \bar{k}() = \text{col}(K_{x_1}(\cdot), K_{x_2}(\cdot), \ldots, K_{x_h}(\cdot)) \).

Equation (10) serves as a data-driven nonlinear explicit predictor, which can be used to predict future behaviors of system (7) and embedded in predictive control algorithms. The choice of kernel \( K \) encodes the properties of the function space. Indeed, (10) shows that the synthesized functions are linear combinations of the kernel sections centered at the sampling points. Hence, one can choose the kernel \( K \) based on a priori knowledge on the system. For example, if one knows that the functions \( f_i (i \in \{1:pN\}) \) are polynomials in inputs and initial data, then a polynomial kernel \( K(x_i, x_j) = (x_i^T x_j + c)^{d_i} \) can be used, such that \( f_i \) belongs to the corresponding RKHS. When the functions \( f_i \) are very high-order polynomials or other complex forms, a Gaussian kernel \( K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/22^2) \) can be used to span the set of continuous functions, such that \( f_i \) belongs to the corresponding RKHS, since the Gaussian kernel is a universal kernel [46]. If a linear kernel is used, then the predictor (10) is similar to the linear predictor (4). Note that \( f_i (i \in \{1:pN\}) \) contains predictors at different time steps, and one should choose the kernel \( K \), such that all the functions \( f_i \) belong to the associated RKHS. For instance, if \( f_i \) are polynomial functions, the predictor of the \( N \)th step could have a (possibly significantly) higher order than that of the first step, and the order of a polynomial kernel \( K \) should be chosen as the highest order in \( f_i \).

III. DEEPC WITH KERNEL METHODS

A. Review of the DeePC Algorithm

The DeePC algorithm proposed in [26] directly uses input–output data collected from the unknown LTI system to predict the future behavior and performs optimal and safe control without identifying a parametric system representation. More specifically, DeePC solves the following optimization problem to obtain the optimal future control inputs:

\[
\min_{u \in \mathbb{R}^N, y \in \mathbb{R}^N} \ell(u) + \|y - r\|_Q \quad \text{s.t. (10)}. \tag{11}
\]

where \( \ell : \mathbb{R}^{mN} \to \mathbb{R} \) is the cost function of control actions, for instance, \( \ell(u) = \|u\|_R \), or \( \ell(u) = \|u\|^2_2 \) (considered in this article). The sets of input and output constraints are defined as \( U = \{u \in \mathbb{R}^{mN} | W_u u \leq w_u \} \) and \( Y = \{y \in \mathbb{R}^{pN} | W_y y \leq w_y \} \). The positive definite matrix \( R \in \mathbb{S}^{mN \times mN} \) and positive semidefinite matrix \( Q \in \mathbb{S}^{pN \times pN} \) are the cost matrices. The vector \( r \in \mathbb{R}^{pN} \) is a prescribed reference trajectory for the future outputs. DeePC involves solving the convex quadratic problem (11) in a receding horizon fashion, that is, after calculating the optimal control sequence \( u^* \), we apply \((u_1, \ldots, u_{N+k-1}) = (u^*_0, \ldots, u^*_{k-1})\) to the system for \( k \leq N \) time steps; then, reinitialize problem (11) by updating \( \text{col}(u_{ini}, \bar{y}_{ini}) \) to the most recent input and output measurements, and setting \( t \) to \( t + k \), to calculate the new optimal control for the next \( k \leq N \) time steps. As in MPC, the control horizon \( k \) is a design parameter.

The standard DeePC algorithm in (11) assumes exact output data \((\bar{Y}_F, \bar{Y}_F, \bar{Y}_{ini})\) generated from the unknown system (1). However, in practice, exact data are not accessible to the controller due to measurement noise. We note that regularized DeePC algorithms can be used to provide robustness against disturbances on the data, which often amount to one-norm or two-norm regularization of \( g \) in the cost function of (11) [29], [31]. In this article, we assume no process or input noise, i.e., the input data \((U_p, \bar{Y}_F, u_{ini})\) are perfectly known.

Exact Data and Inexact Data: Throughout this article, we use \( \bar{Y}_F, \bar{Y}_F, \bar{Y}_{ini} \) to denote the exact (no noise) output data generated from the system, which accurately captures the system dynamics; we use \( \bar{Y}_F, \bar{Y}_F, \bar{Y}_{ini} \) to denote the corresponding inexact (noisy) output data. We use \( \bar{K} \) to denote the Gram matrix calculated from exact output data and \( \bar{K} \) to denote the Gram matrix calculated from inexact output data.

B. RoKDeePC Algorithm

The DeePC algorithm above implicitly incorporates a linear predictor. In this section, we develop the RoKDeePC algorithm using an implicit and robustified version of the kernel-based nonlinear predictor (10) to perform data-driven optimal control for nonlinear systems. We first consider the following certainty-equivalence MPC problem (with a weighted two-norm cost on the future outputs) using the kernel predictor in (10) (referred to as kernel-based MPC in this article):

\[
\min_{u \in \mathbb{R}^N, y \in \mathbb{R}^N} \ell(u) + \|y - r\|_Q \quad \text{s.t. (10)}. \tag{11}
\]

In what follows, we assume that when exact data are available, the predictor (10) can faithfully predict the future behavior of the nonlinear dynamics (7).

Assumption 1: When exact data \((\bar{Y}_F, \bar{Y}_F, \bar{Y}_{ini})\) is available in the kernel-based nonlinear predictor (10), the prediction error is bounded, i.e., there is \( \beta_e \geq 0 \), so that

\[
\|y_{sys} - y_{pred}\|_Q \leq \beta_e \tag{13}
\]

where \( y_{pred} = \bar{Y}_F \bar{K} + y I)^{-1} k(u_{ini}, \bar{y}_{ini}, u_{sys}) \) is the predicted output trajectory based on (10), \( u_{sys} \) is a control sequence applied to the system, and \( y_{sys} \) is the (noise-free) realized output trajectory of the system in response to \( u_{sys} \).

Assumption 1 implicitly requires that sufficiently rich data are used to capture the system dynamics. We refer to [47, Proposition 1] on how to possibly find \( \beta_e \) from exact data. Moreover, according to [47, Proposition 2], one can possibly use more data to obtain a more accurate prediction and, thus, reduce \( \beta_e \). Note that the explicit conditions of (minimal) data requirements to satisfy (13) still remain to be investigated in kernel methods. Since exact data are available, this assumption can generally be satisfied if the unknown dynamics \( f_i \) are contained in the RKHS of the chosen kernel \( K \), e.g., \( f_i \) is polynomial, and a sufficiently high-order
polynomial kernel is used. Otherwise, this assumption can be met by using kernels with the universal approximation property (e.g., a Gaussian kernel) and by implicitly assuming that \( f_i \) is continuous and that \( \text{col}(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u_{\text{sys}}) \) is contained in a compact set [46].

When inexact (noisy) data are used, the prediction error may grow unbounded, and the obtained control sequence from (12) may not result in a robust closed loop. As a first step toward addressing this problem, we rewrite (10) implicitly as follows:

\[
\begin{bmatrix}
\hat{K} + \gamma I \\
\hat{Y}_F
\end{bmatrix} g = \begin{bmatrix}
k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \\
y
\end{bmatrix}
\]

where \( g \) is uniquely determined by \((\hat{K} + \gamma I) g = k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u), \) since \((\hat{K} + \gamma I)\) has full rank, and \( y = \hat{Y}_F g \) predicts the future behavior of the system. The reformulation of (10) into (14) is essential for later steps of robustifying the predictive control problem and providing performance guarantees under noisy output data. We note that, given \( \text{col}(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \), there exists a unique \( g \in \mathbb{R}^{H_f} \) that satisfies (14). Unlike linear systems, however, given an arbitrary \( g \), there may not exist a vector \( \text{col}(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \) that satisfies (14); for linear systems, such a vector exists because of the fundamental lemma. Substituting (14) into (12) leads to

\[
\min_{u \in \mathcal{U}, y \in \gamma, g} \ell(u) + \| y - r \|_Q \quad \text{s.t.} (14).
\]

Note that (15) is equivalent to (12), but it avoids the matrix inversion in (10), which is useful in the later development of robust formulations. When exact data are available, both (12) and (15) achieve satisfactory predictions and control performance, according to Assumption 1. However, in our setting (Assumption 1), the prediction errors cannot be a priori bounded when inexact data are used, which may result in unacceptable control performance. In fact, quantifying the prediction error of (10) with inexact data is a nontrivial task. In what follows, we circumvent this problem by developing a min–max version of (15) to handle inexact data. A necessary step for this purpose is to relax the equality constraint in (15) by penalizing its violation in the cost function (i.e., soft constraints)

\[
\min_{u \in \mathcal{U}, y \in \gamma, g} \ell(u) + \| \hat{Y}_F g - r \|_Q + \lambda_k \| (\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \|
\]

where \( \lambda_k \) is a penalty parameter. We note that with a sufficiently large \( \lambda_k \), the solution to (16) becomes close to the solution to (15) [49]. If the objective function in (15) is Lipschitz continuous, then one can find a finite \( \lambda_k \) that establishes the equivalence of (15) and (16) [50, Proposition 9.1.1]. A robust solution may still not be obtained when inexact output data are used in (16). Hence, we further robustify (16) and propose the following RoKDeePC formulation that leverages inexact output data for nonlinear, robust, and optimal control:

\[
\min_{u \in \mathcal{U}, g \in \mathcal{G}} \max_{\Delta_K, \Delta_Y \in \mathcal{D}_{F1}} \ell(u) + \| (\hat{Y}_F + \Delta_Y) g - r \|_Q \\
+ \lambda_k \| (\hat{K} + \Delta_K + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) + \Delta_k \|
\leq \lambda_k \| (\hat{K} + \Delta_K + \gamma I) g - (k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) + \Delta_k) \|
\]

where \([\Delta_K, \Delta_Y]\) and \( \Delta_f \) are disturbances added to the data matrices, which, respectively, reside in prescribed disturbance sets \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \), and \( \gamma = \{ g \mid (\hat{Y}_F + \Delta_Y) g \in \gamma, \forall \Delta_Y \in \mathcal{D}_2 \} \) robustifies the output constraints.

The above min–max formulation considers the worst-case scenario of the uncertainties added to the data matrices, thereby leading to a robust solution. We will later show that performance guarantees can also be provided by employing this formulation. Note that min–max optimization problems could be, in general, difficult to solve; however, (17) admits a tractable reformulation as a minimization problem if appropriate disturbance sets \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) are considered. The following result shows that the inner max problem of (17) can be resolved analytically for appropriate disturbance sets \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \), resulting in regularized problem formulations.

**Lemma 2:** Consider two disturbance sets

\[
\mathcal{D}_{F1} = \{ [\Delta_K, \Delta_Y] \in \mathbb{R}^{H_f \times (H_f + 1)} \mid \| [\Delta_K, \Delta_Y] \|_F \leq \rho_1 \}
\]

and

\[
\mathcal{D}_{F2} = \{ \Delta_Y \in \mathbb{R}^{N \times H_f} \mid \| Q^{1/2} \Delta_Y \|_F \leq \rho_2 \}.
\]

Given a vector \( u \in \mathcal{U} \) and a vector \( g \in \mathcal{G} \), it holds that

\[
\max_{[\Delta_K, \Delta_Y] \in \mathcal{D}_{F1}} \| (\hat{K} + \Delta_K + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) + \Delta_k \| \\
= \| (\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \| + \rho_1 \sqrt{\| g \|^2 + 1} \tag{19}
\]

\[
\max_{\Delta_Y \in \mathcal{D}_{F2}} \| (\hat{Y}_F + \Delta_Y) g - r \|_Q \\
= \| \hat{Y}_F g - r \|_Q + \rho_2 \| g \|.
\]

**Proof:** The proof is adapted from [51, Th. 3.1]. Consider a fixed \( g \) and a fixed \( u \) in (19). It follows from triangle inequality that

\[
\max_{[\Delta_K, \Delta_Y] \in \mathcal{D}_{F1}} \| (\hat{K} + \Delta_K + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) + \Delta_k \| \\
\leq \| (\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \| \\
+ \max_{[\Delta_K, \Delta_Y] \in \mathcal{D}_{F1}} \| \Delta_K g - \Delta_k \| \\
= \| (\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \| + \rho_1 \sqrt{\| g \|^2 + 1}.
\]

Choose \( \hat{\Delta} = [\hat{\Delta}_K, \hat{\Delta}_Y] \in \mathcal{D}_{F1} \), such that

\[
[\hat{\Delta}_K, \hat{\Delta}_Y] = \frac{\rho_1 \omega}{\sqrt{\| g \|^2 + 1}} \begin{bmatrix} g^T & -1 \end{bmatrix}
\]

where

\[
\omega = \begin{cases} 
(\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) & \text{if } (\hat{K} + \gamma I) g \neq k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \\
\text{any unit-norm vector}, & \text{otherwise}.
\end{cases}
\]

Since \( \hat{\Delta} \) is a rank-1 matrix, we have \( \| \hat{\Delta} \|_F = \| \hat{\Delta} \| = \rho_1 \), and

\[
\| (\hat{K} + \hat{\Delta}_K + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) + \hat{\Delta}_k \| \\
= \| (\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \| + \| \hat{\Delta}_K g - \hat{\Delta}_k \| \\
= \| (\hat{K} + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) \| + \rho_1 \sqrt{\| g \|^2 + 1} \leq \max_{[\Delta_K, \Delta_Y] \in \mathcal{D}_{F1}} \| (\hat{K} + \Delta_K + \gamma I) g - k(u_{\text{ini}}, \tilde{y}_{\text{ini}}, u) + \Delta_k \|.
\]
where the first equality follows from triangle inequality and the fact that \([\bar{\mathbf{K}} + \gamma \mathbf{I}]g - k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u)\) is a nonnegative scalar multiple of \((\Delta_K g - \Delta_k)\). Since the lower bound coincides with the upper bound on the maximization problem (19) for any fixed \(g\) and \(u\), the equality in (19) holds. By following a similar process, one can prove the equality in (20).

**Corollary 1 (Tractable Formulation for RoKDeePC):** By considering \(D_1 = D_{F_1}\) and \(D_2 = D_{F_2}\) in (17), where \(D_{F_1}\) and \(D_{F_2}\) are defined in (18), it holds that

\[
(17) = \min_{\ell(u), g \in \mathcal{G}} \ell(u) + \left\| \bar{\mathbf{Y}}_F - r \right\|_Q + h(g) + \lambda_k \left\| \bar{\mathbf{K}} + \gamma \mathbf{I} \right\| g - k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u) \right\| \tag{21}
\]

where \(h(g) = \lambda_k \rho_1(\|g\|^2 + 1)^{1/2} + \rho_2\|g\|\) is the regularizer. Moreover, the minimizer of (21) coincides with that of (17).

The above result shows that one can get robust control inputs by simply solving a regularized minimization problem. The proposed RoKDeePC algorithm involves solving (21) in a receding horizon manner, similar to the DeePC algorithm. The initial trajectory \((u_{\text{ini}}, \hat{y}_{\text{ini}})\) should be updated every time before solving (21). We remark that \(D_{F_1}\) and \(D_{F_2}\) are unstructured uncertainty sets, which may lead to conservativeness. For instance, \(\bar{\mathbf{K}}\) is a symmetric Gram matrix generated from a kernel function, but \(D_{F_1}\) does not restrict \(\Delta_K\) to be symmetric or incorporate the kernel information; \(\bar{\mathbf{Y}}_F\) may be a Hankel data matrix, but a Hankel structure cannot be imposed on \(\Delta_y\) by considering \(D_{F_2}\). As will be seen in the simulation section, though the considered unstructured sets are conservative, they can lead to satisfactory performance in the presented case studies.

Notice that the cost function in (21) is convex in \(g\), but nonconvex in \(u\). Hence, in general, one may not be able to find the global minimizer. Given a feasible control sequence \(u_{\text{sys}} \in \mathcal{U}\) (e.g., a local minimizer), we define the minimum cost of (21) over \(g\) as follows:

\[
c_{\text{opt}}(u_{\text{sys}}) = \min_{g \in \mathcal{G}} \ell(u_{\text{sys}}) + \left\| \bar{\mathbf{Y}}_F - r \right\|_Q + h(g) + \lambda_k \left\| \bar{\mathbf{K}} + \gamma \mathbf{I} \right\| g - k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}}) \right\| \tag{22}
\]

where \(h(g)\) is the same as that in (21).

**C. Performance Guarantees of RoKDeePC**

In what follows, we show that the proposed RoKDeePC algorithm provides deterministic performance guarantees when applying a (locally) optimal control sequence to the system because of the min–max formulation in (17). We begin by making the following assumption.

**Assumption 2:** For any col\((u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}})\) (contained in a compact set), there exists \([\Delta_K \Delta_y] \in \mathcal{D}_{F_1}\), such that \(\bar{\mathbf{K}} + \Delta_K = \bar{\mathbf{K}}\) and \(k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}}) + \Delta_y = k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}})\); there exists \(\Delta_y \in \mathcal{D}_{F_2}\), such that \(\bar{\mathbf{Y}}_F + \Delta_y = \bar{\mathbf{Y}}_F\), where \(D_{F_1}\) and \(D_{F_2}\) are defined in Lemma 2.

Assumption 2 indicates that sufficiently large uncertainty sets \(D_{F_1}\) and \(D_{F_2}\) are used to cover the deviations of the data matrices induced by noise. This is necessary for (17) to provide a robust solution against the realized uncertainty in practice. Since col\((u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}})\) is contained in a compact set and \(k(\cdot)\) is continuous (by choosing a continuous kernel), Assumption 2 can be easily satisfied when the output noise is bounded and sufficiently large disturbance sets \(D_{F_1}\) and \(D_{F_2}\) are considered, which corresponds to choosing sufficiently large regularization parameters \(\rho_1\) and \(\rho_2\) in (21). Note that it could be nontrivial to find the exact bound of the Gram matrix deviations under output noise. However, since \(\rho_1\) is a to-be-tuned parameter in the control algorithm, we can conservatively use a (sufficiently large) estimate, which avoids deriving the exact bound. We define the realized cost of the system as \(c_{\text{realized}}(u_{\text{sys}}) = \ell(u_{\text{sys}}) + \|y_{\text{sys}} - r\|_Q\) (with \(y_{\text{sys}}\) defined in Assumption 1). The following result shows that the realized cost is upper bounded by the optimization cost in (22) plus the prediction error in (13).

**Theorem 1:** If Assumptions 1 and 2 hold, then there exists a sufficiently large \(\lambda_k\) for (22), such that

\[
c_{\text{realized}}(u_{\text{sys}}) \leq c_{\text{opt}}(u_{\text{sys}}) + \beta_\epsilon, \quad \text{for any feasible } u_{\text{sys}} \in \mathcal{U} \tag{23}
\]

where \(\beta_\epsilon\) is the prediction error bound in Assumption 1.

**Proof:** According to the definition of \(c_{\text{opt}}(u_{\text{sys}})\), if Assumption 2 holds, we have

\[
c_{\text{opt}}(u_{\text{sys}}) = \min_{g \in \mathcal{G}} \max_{\Delta_K \in \mathcal{D}_{F_1}, \Delta_y \in \mathcal{D}_{F_2}} \ell(u_{\text{sys}}) + \left\| \bar{\mathbf{Y}}_F + \Delta_y \right\| g - k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}}) \right\| + \lambda_k \left\| \bar{\mathbf{K}} + \gamma \mathbf{I} \right\| g - k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}}) \right\| \tag{24}
\]

where \(g^*\) minimizes (17) in which \(u = u_{\text{sys}}\). Given a vector \(u_{\text{sys}}\), it follows from (14) that there exists a \(\hat{g}\), such that

\[
\left[\hat{\mathbf{K}} + \gamma \mathbf{I}\right] \hat{g} = \left[\begin{array}{c} k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}}) \\ y_{\text{pred}} - \bar{\mathbf{Y}}_F g^* \end{array}\right] \tag{25}
\]

where \(y_{\text{pred}}\) is the predicted output trajectory from the kernel-based nonlinear predictor. By defining \(\Delta_g = \hat{g} - g^*\), we have

\[
[\hat{\mathbf{K}} + \gamma \mathbf{I}] \Delta_g = \left[\begin{array}{c} k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u_{\text{sys}}) - \left(\hat{\mathbf{K}} + \gamma \mathbf{I}\right) g^* \\ y_{\text{pred}} - \bar{\mathbf{Y}}_F g^* \end{array}\right] \tag{26}
\]

which implies that

\[
\bar{\mathbf{Y}}_F \Delta_g = \bar{\mathbf{Y}}_F \left[\hat{\mathbf{K}} + \gamma \mathbf{I}\right]^{-1} \epsilon = y_{\text{pred}} - \bar{\mathbf{Y}}_F g^*. \tag{27}
\]

By substituting (26) and (27) into (24), we obtain

\[
c_{\text{opt}}(u_{\text{sys}}) \geq \ell(u_{\text{sys}}) + \left\| y_{\text{pred}} - r - \bar{\mathbf{Y}}_F \left[\hat{\mathbf{K}} + \gamma \mathbf{I}\right]^{-1} \epsilon \right\|_Q + \lambda_k \|\epsilon\| \tag{28}
\]

where the second and the forth inequalities hold because of the reverse triangle inequality, and the third inequality

\[
\geq \epsilon \left| y_{\text{sys}} - y_{\text{pred}} \right|_Q \geq \epsilon \left| c_{\text{realized}}(u_{\text{sys}}) - \beta_\epsilon \right|_Q \tag{28}
\]
is satisfied if we take $\lambda_k$ large enough (by assumption) to ensure

$$\lambda_k^2 I \succeq (\hat{K} + \gamma I)^{-1} \hat{Y}_F^T \hat{Q} \hat{Y}_F (\hat{K} + \gamma I)^{-1}.$$ 

Hence, $\lambda_k \|e\| \geq \|\hat{K} + \gamma I\|^{-1} \|e\|_Q$, and the fifth inequality follows from Assumption 1 and the definition of $c_{\text{realized}}(u_{\text{sys}})$. This completes the proof.

The above result links the optimization cost to the realized cost, namely, the realized cost is always bounded by the optimization cost plus the prediction error $\beta e$ even when inexact data are used. This also justifies the design of the cost function of the RoKDeePC in (21) and shows that one should try to find not only a feasible solution but ideally a minimizer (or better the global minimizer) to reduce the optimization cost and possibly the realized cost. Note that Theorem 1 holds for both $\ell(u) = \|u\|_g$ and $\ell(u) = \|u\|_g^2$, as we assume no input noise in this article (i.e., the realized input cost equals the input cost in the optimization problem).

**Remark 1:** In Assumption 1, we assume that the kernel-based predictor (10) gives good predictions (i.e., the prediction error is bounded by $\beta e$) when exact data are used. Then, we circumvent the problem of finding the prediction error bound under noisy data. To be specific, starting from Assumption 1, we show that when inexact output data are used, though the noisy prediction error cannot be a priori bounded, the control performance (i.e., the realized cost) can be guaranteed using the min–max formulation (17). Note that $\beta e$ appears in the performance guarantee, while it is not needed in implementing the proposed control method.

It can be observed that when inexact data are used in the predictor (10), the prediction becomes

$$y = \hat{Y}_F (\hat{K} + \gamma I)^{-1} k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u)$$

$$= (\hat{Y}_F - \Delta y) (\hat{K} - \Delta K + \gamma I)^{-1} [k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u) - \Delta_k].$$

(29)

We can see from (29) that it is nontrivial to derive the bound of prediction error when inexact data are used, since the uncertainty matrix $\Delta K$ is within the inverse operation. However, our approach circumvents the step of deriving this prediction error under inexact data and directly robustifies the optimal control problem based on Assumption 1.

**IV. GRADIENT DESCENT METHODS TO SOLVE RoKDeePC**

The RoKDeePC problem in (21) is nonconvex in $u$ unless the kernel function is linear in $u$. One may use a generic nonlinear optimization toolbox (e.g., IPOPT [52]) to solve this problem. However, solving a nonconvex problem is, in general, time-consuming, especially considering that the decision variable $g$ in (21) can be high dimensional. We propose customized gradient descent methods exploiting the structure of (21) to solve the RoKDeePC problem more efficiently. Although a global minimum may still not be reached (as one may only achieve a local minimizer by using gradient descent methods to solve a nonconvex problem), the numerical study in Section V suggests that these methods generally lead to fast calculations and satisfactory performance.

**A. Gradient Descent Algorithm for RoKDeePC**

Notice that the optimization problem in (21) is nonconvex in $u$ but convex in $g$. Moreover, the dimension of $g$ is generally much higher than that of $u$. Due to this favorable structure, we apply a gradient descent method to update $u$, while at each iteration, we solve a convex optimization problem to update $g$.

This coincides with viewing (21) as follows:

$$\min_{u \in \mathcal{U}} \min_{g \in \mathcal{G}} c(u, g)$$

where $c(u, g)$ is defined in (30).

Note that the input and output constraints can be handled using projected gradient algorithms. We summarize the algorithm for solving RoKDeePC in Algorithm 1. In Step 2, the projection of $u$ into the set $\mathcal{U}$ admits a closed-form solution if upper and lower bounds for the elements in $u$ are considered. Step 3 requires solving a second-order cone program (SOCP), which could make the algorithm time-consuming as one may need to solve the SOCP $i_{\text{max}}$ times. To reduce the computational burden, in what follows, we consider an equivalent cost function, which is quadratic in $g$ and, thus, admits a closed-form solution for Step 3 when the output constraints are inactive.

**Algorithm 1 Projected Gradient Algorithm to Solve (21)**

**Input:** cost function:

$$c(u, g) = \ell(u) + \|\hat{Y}_F g - r\|_Q + h(g) + \lambda_k \|\hat{K} + \gamma I\|^{-1} \|k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u) - \Delta_k\|_2,$$

(30)

where $h(g)$ is the same as that in (21); initial vectors $u = u(0) = 0, g = \hat{g}(0) = 0$; constraint sets $\mathcal{U}$, $\mathcal{G}$; step size $\alpha$; maximal iteration number $i_{\text{max}}$; convergence threshold $\xi$.

**Initialization:** $i = 0$.

**Iterate until** $|c(u(i), g(i)) - c(u(i-1), g(i-1))| < \xi$ or $i \geq i_{\text{max}}$:

1) $i \leftarrow i + 1$

2) $u(i) = \arg \min_{u \in \mathcal{U}} \|u - (u(i-1) - \alpha \frac{\partial c(u, g)}{\partial u}|_{u = u(i-1)} \|_2$

3) $g(i) = \arg \min_{g \in \mathcal{G}} c(u(i), g)$

**Output:** (sub)optimal control sequence $u^* = u(i)$.

**B. Quadratic Reformulation**

To enable a fast calculation of Step 3, we consider the following cost function that is quadratic in $g$:

$$c_q(u, g) = \ell(u) + \|\hat{Y}_F g - r\|_Q^2 + \lambda_k \|\hat{K} + \gamma I\|^{-1} \|k(u_{\text{ini}}, \hat{y}_{\text{ini}}, u)\|^2.$$ 

(31)

The following result shows the connection between the cost functions in (30) and (31).

**Proposition 1:** For any $u$, if $g^* \in \mathbb{R}^H$ is a minimizer of

$$\min_{g \in \mathcal{G}} c_q(u, g)$$

(32)

then $g^*$ also minimizes

$$\min_{g \in \mathcal{G}} c(u, g)$$

(33)
with  
\[ \lambda_k = \begin{cases} 
\lambda'_k \| (\mathbf{K} + \gamma I)g^* - k(u_{ini}, \hat{y}_{ini}, u) \|, & \text{if } \hat{Y}_T g^* \neq r \\
\lambda''_k \| (\mathbf{K} + \gamma I)g^* - k(u_{ini}, \hat{y}_{ini}, u) \|, & \text{otherwise}
\end{cases} \]  
(34) 

\[
\frac{\lambda_k \rho_1 \| g^* \|^2}{\| g^* \|^2 + 1} + \rho_2 \\
= \begin{cases} 
\frac{\lambda_k \rho_1 \| g^* \|^2}{\| g^* \|^2 + 1}, & \text{if } \hat{Y}_T g^* \neq r \\
\frac{\lambda''_k \| g^* \|^2}{\| g^* \|^2}, \text{ otherwise}
\end{cases} \]  
(35) 

Moreover, \( \lambda_k \) in (34) is monotonically increasing in the \( \lambda'_k \) chosen in (31); if \( \rho_2 (\rho_1) \) remains constant, \( \rho_1 (\rho_2) \) in (35) is monotonically increasing in the \( \lambda_k \) chosen in (31). 

Proof: See the Appendix. 

The above result shows that using the cost function \( c_k(u, g) \) in the RoKDeePC is equivalent to setting different parameters \((\lambda_k, \rho_1, \rho_2)\) to the problem in (21) every time solving it during the receding horizon implementation, as the equivalent values of \( \lambda_k, \rho_1, \) and \( \rho_2 \) are related to \( g^* \). This may lead to conservativeness, as we need to choose a robust pair of \((\lambda'_k, \lambda_k)\). As a remedy, one can also update \( \lambda'_k \) if the obtained \( \lambda_k \) is not large enough. To be specific, during the online implementation, after solving (32) and before applying the control sequence to the system, one can compute \( \lambda_k, \rho_1, \) and \( \rho_2 \) (by fixing a desired value of \( \rho_1 \) or \( \rho_2 \) and then compute the other) according to (34) and (35). If the obtained \( \lambda_k, \rho_1, \) and \( \rho_2 \) are not sufficiently large to satisfy Assumption 2 and Theorem 1, then \( \lambda'_k \) and \( \lambda_k \) should be increased to implicitly increase \( \lambda_k, \rho_1, \) and \( \rho_2 \). One can repeat the above process until the obtained \( \lambda_k, \rho_1, \) and \( \rho_2 \) are satisfactory, and then apply the control sequence to the system. Of course, this requires performing more computation as one may need to solve (32) multiple times.

Moreover, when the output constraints are inactive, (32) admits a closed-form solution as follows:

\[ g^* = M_r r + M_k k(u_{ini}, \hat{y}_{ini}, u) \]  
(36) 

where \( M_r = [\hat{Y}_T^\top Q \hat{Y}_T + \lambda_k I + \lambda'_k (\mathbf{K} + \gamma I)]^{-1} \hat{Y}_T^\top Q \) and \( M_k = [\hat{Y}_T^\top Q \hat{Y}_T + \lambda_k I + \lambda'_k (\mathbf{K} + \gamma I)]^{-1} \mathbf{K} \). Notice that (36) requires simply a linear mapping to obtain \( g^* \), which can greatly accelerate the gradient descent algorithm when the output constraints are inactive. In summary, we consider the following modified RoKDeePC optimization problem:

\[ \min_{u \in \mathcal{U}, g \in \mathcal{G}} c_q(u, g) \]  
(37) 

which is related to the original formulation in (21) via Proposition 1, and we implement the following modifications to Algorithm 1 to solve (37).

1) Change the cost function from \( c(u, g) \) to \( c_q(u, g) \).

2) Change Step 3 in the iteration to:

\[ g(i) = M_r r + M_k k(u_{ini}, \hat{y}_{ini}, u(i)) \text{ if } g(i) \notin \mathcal{G}, g(i) \leftarrow \arg \min_{g \in \mathcal{G}} c_q(u(i), g). \]

Step 3 becomes easier to solve if the output constraint set \( \mathcal{G} \) is a nonempty box constraint (i.e., upper and lower bounds for the outputs are considered), as the constraint set \( \mathcal{G} \) can be formulated as \( \mathcal{G} = \{ g | G_1 g + 1 \| g \| \leq q_1 \} \) for some \( G_1 \) and \( q_1 \) [31], [53]. The computation cost can be further reduced by assuming that \( \| g \| \) is bounded and (conservatively) restricts \( \mathcal{G} \) to be a polyhedron \( \mathcal{G} = \{ g | G_1 g \leq q' \} \) for some \( q' \). In our applications, we do not find these assumptions to be limiting. The step size \( \alpha \) in the algorithm (i.e., the so-called learning rate in learning-based algorithms) is a design parameter, which can be chosen via, for instance, a line search that satisfies the Wolfe [54] conditions to ensure convergence to a local minimum [55]. As an alternative, one can also choose a sufficiently small (constant) step size to ensure convergence (to a local minimum), as the objective function combining (37) with (36) is \( L \)-smooth [56], [57].

We remark that one can also assume a kernel function that is nonlinear in \((u_{ini}, \hat{y}_{ini})\) but linear in \( u \), such that (31) becomes a convex quadratic program, which can be more efficiently solved using standard solvers. For instance, one may consider:

\[ K(x_i, \text{col}(u_{ini}, \hat{y}_{ini})) = \exp \left( \frac{\| x_{i,1} - \text{col}(u_{ini}, \hat{y}_{ini}) \|^2}{2 \sigma^2} \right) + x_{i,2}^2 u \]  
(38) 

where \( x_i = \text{col}(x_{i,1}, x_{i,2}), \) Note that (38) is a combination of two positive semidefinite kernels and, thus, also satisfies the positive semidefinite property. If the future outputs of the system are linear in \( u \), such a kernel can be used to accurately capture the system’s dynamics. Otherwise, the obtained kernel-based predictor is the linear-input model that best fits the observed data, which may not lead to an accurate prediction.

V. SIMULATION RESULTS

In this section, we test RoKDeePC on two example nonlinear systems to illustrate its effectiveness. We will compare the performance when different kernel functions are used, and we will compare the RoKDeePC algorithm with the traditional DeePC algorithm, the kernel-based MPC in (12), and the Koopman-based MPC in [19]. Note that the kernel-based MPC in (12) is an indirect data-driven control approach, as it first uses data to obtain a model/predictor of the system and then performs control design. By comparison, RoKDeePC is a direct data-driven control method that avoids an explicit step of SysID [i.e., no equality constraint is enforced in (37)]. We will compare the performance of these two methods under different noise levels. One can consider advanced versions of these two methods (e.g., with terminal constraints and chance constraints) to possibly improve their performance in the future work. The simulation files that implement the RoKDeePC algorithm and the other methods for comparison are available at: https://github.com/Linbin-Huang/RoKDeePC.git.

A. Example 1: A Nonlinear Single-Input-Single-Output Model

We start with an academic case study to compare different predictors. Consider the following discrete-time nonlinear model of a polynomial single-input single-output system:

\[ y_t = 4 y_{t-1} u_{t-1} - 0.5 y_{t-1} + 2 u_{t-1} u_t + u_t. \]  
(39)
where $\psi$, and ten thin plate spline radial basis functions (defined by $R(x, y)$ to the hyperparameter tuning of Gaussian processes [12].

Estimation to tune the parameter in Gaussian kernel, similar noise-free data are available. For instance, the Gaussian kernel that the kernel predictors have satisfactory performance when parameters in these kernel functions are chosen to ensure $K(x, y)$ non-satisfactory performance when noise-free data are available. For instance, the Gaussian kernel has only one parameter to tune, and thus, an appropriate value can be easily found with a line search in the cross validation, i.e., choose the value that minimizes the total prediction error on the test dataset; one can also use maximal likelihood estimation to tune the parameter in Gaussian kernel, similar to the hyperparameter tuning of Gaussian processes [12].

For the Koopman-based predictor, one can consider $x_i = (u_{i-1}, y_{i-1})$ as the state variables of the above system. Similar to [19], we consider the lifting functions in the form of $\phi_i(x_i, u_i) = \psi_i(x_i) + \mathcal{L}_i(u_i)$ (40) where $\psi_i : \mathbb{R}^n \rightarrow \mathbb{R}$ is, in general, nonlinear, but $\mathcal{L}_i : \mathbb{R}^m \rightarrow \mathbb{R}$ is linear. The lifting functions $\psi_i$ are chosen to be the linear and quadratic terms of the state variables and ten thin plate spline radial basis functions (defined by $\psi(x) = \|x - x_0\|^2 \log(\|x - x_0\|)$ whose center is $x_0$, with centers selected randomly from $[-1.5, 1.5]^2$. Under this setting, the input-output data can be used in the extended dynamic mode decomposition (EDMD) [11] to solve for the parametric system representation (Koopman-based predictor). The lifting functions in (40) lead to a quadratic program for the MPC formulation, which can be solved efficiently in practice.

Fig. 1 compares the open-loop predictions of 50 time steps obtained by employing different predictors, and the corresponding prediction errors are shown in Table I. For the linear predictor and the kernel predictor (whose prediction horizon is $N = 5$), the time steps from $iN + 1$ to $(i + 1)N$ ($i \in [1:9]$) are predicted based on the previous predictions. It can be seen that the linear predictor cannot make an accurate prediction due to the strong nonlinearity of the system. The Koopman-based predictor also cannot make an accurate prediction, because (40) assumes independence between $x_i$ and $u_i$, which is not the case in (39). Moreover, we observe similar prediction errors even with more data in the EDMD, e.g., $T = 200,000$ (data not shown). The structure in (40) enables fast calculations of the associated MPC problem, but restricts the applicability in some classes of nonlinear systems. One can also assume a more general form of $\phi_i$ to improve the prediction accuracy, but it is out of the scope of this article.

By comparison, all kernel predictors can predict the system’s behaviors with satisfactory accuracy. Moreover, when there is no measurement noise, the predictor with polynomial kernel almost perfectly predicts the system’s behaviors, as the future outputs are indeed polynomial functions of $(u_{i-1}, y_{i-1}, u_i)$ due to the bilinear structure in (39).

We next compare the time-domain performance when different control methods are applied: 1) DeePC with quadratic regularization [30]; 2) Koopman-based MPC [19]; 3) nominal (i.e., nonrobustified) Kernel-based MPC in (12) (the cost function is modified to be $\ell(u) + \|y - r\|_Q^2$); and 4) RoKDeePC in (37). We consider the following input and output costs in all these methods: $\ell(u) + \|y - r\|_Q^2 = \sum_{i=1}^{N} (|u_i|^2 + 10^3 |y_i - r_i|^2) + \sum_{i=2}^{N} 10^2 |u_i - u_{i-1}|^2$, where $r_i$ is the output reference over time. This cost function penalizes the control inputs, the tracking errors, and the rates of changes of control inputs. The other parameters of the RoKDeePC are as follows: $\lambda_c^2 = 10^8$ and $\lambda_p = 1$. The control horizon is $k = 1$ in all the methods. We assume no input and output constraints and focus on comparing the tracking performance. Hence, no projection is needed in the gradient descent algorithm.

Fig. 2 shows the time-domain responses of the system when different control methods are applied, where the output reference $r_i$ steps from 0 to 0.1 at $t = 0.1$ s and then to 0.05 at $t = 0.3$ s. The sampling time of the discrete-time system is 2 ms. It can be seen that when DeePC or

| Prediction Error $\sum_i \|\Delta y_i\|^2$ | Without measurement noise | With white noise (variance: $10^{-3}$) |
|------------------------------------------|---------------------------|-------------------------------|
| Linear predictor                        | 0.2378                    | 0.2384                        |
| Koopman predictor (linear in $u_i$)     | 0.1927                    | 0.2175                        |
| Kernel predictor (polynomial kernel)    | 0.0008                    | 0.0486                        |
| Kernel predictor (Gaussian kernel)      | 0.0051                    | 0.0463                        |
| Kernel predictor (exponential kernel)   | 0.0030                    | 0.0456                        |

Table I: Prediction comparison under different noise levels.

(a) Without measurement noise. (b) White noise (variance: $10^{-3}$).
TABLE II

| noise level | realized costs | RoKDeePC | Kernel-based MPC | RoKDeePC | Kernel-based MPC | RoKDeePC | Kernel-based MPC |
|-------------|---------------|----------|-----------------|----------|-----------------|----------|-----------------|
| variance: $10^{-4}$ | mean | 8.92 | 8.51 | (-4.6%) | 6.07 | 6.16 | (+1.5%) |
| | standard deviation | 4.17 | 3.72 | (-11%) | 1.97 | 2.01 | (+2.0%) |
| variance: $10^{-3}$ | mean | 25.06 | 28.19 | (+12%) | 25.11 | 29.37 | (+17%) |
| | standard deviation | 17.99 | 19.24 | (+6.9%) | 15.31 | 17.66 | (+15%) |
| variance: $1.5 \times 10^{-3}$ | mean | 30.76 | 37.54 | (+22%) | 31.89 | 40.71 | (+28%) |
| | standard deviation | 24.29 | 28.56 | (+16%) | 22.15 | 27.03 | (+22%) |

We remark that with the gradient descent algorithm in Section IV, the RoKDeePC has a reasonable running time. On an Intel Core i7-9750H CPU with 16-GB RAM, it requires approximately 0.2 s to solve (37) (with any of the considered kernels) every time in the above simulations. The kernel-based MPC requires similar time to solve (12) using a similar gradient descent algorithm. By comparison, the Koopman-based MPC and DeePC only require to solve quadratic programs, which take about 2 and 10 ms, respectively, using OSQP [61] as the solver. However, their performance is vastly inferior. To compare the robustness of the RoKDeePC and the kernel-based MPC, the above simulations were repeated 100 times with different datasets to construct the Hankel matrices and different random seeds to generate the measurement noise. Table II displays the mean and standard deviation of realized closed-loop costs, i.e., $\sum_{t=1}^{200} (\|\tilde{u}_t\|^2 + 10^3 \|\tilde{y}_t - r_t\|^2) + \sum_{t=2}^{200} 10^2 \|\tilde{u}_t - \tilde{u}_{t-1}\|^2$, where $\tilde{u}_t$ and $\tilde{y}_t$ are obtained from the system in the simulations. It shows that with the increase of the measurement noise level, the RoKDeePC achieves superior performance than the (certainty equivalence) kernel-based MPC. For instance, when the variance of the measurement noise is $1.5 \times 10^{-3}$, the averaged closed-loop costs of the RoKDeePC are, respectively, 22% (Gaussian kernel), 28% (exponential kernel), and 27% (polynomial kernel) lower than those of the kernel-based MPC. We attribute this performance gap to the robustness of the RoKDeePC, namely, the control sequence obtained from RoKDeePC is robust against the uncertainties in data; by comparison, the kernel-based MPC implicitly assumes certainty equivalence and may not lead to a robust solution. Moreover, we observe that the Gaussian kernel and the exponential kernel have better performance than the polynomial kernel under inexact measurements, even though the future behaviors of the systems can be described by polynomial functions.

**B. Example 2: A Grid-Forming Converter With Nonlinear Load**

We now test the performance of the RoKDeePC in a grid-forming converter that is feeding a local nonlinear load, as shown in Fig. 3. The grid-forming converter employs virtual synchronous machine control for synchronization, which emulates the swing equation of synchronous generators to generate the frequency [62], [63], [64]. The converter is connected to a local nonlinear load, whose active and reactive power

---

Fig. 2. Time-domain responses of the system with different control methods and different measurement noise levels. (a) Without measurement noise. (b) White noise (variance: $10^{-3}$). (c) White noise (variance: $10^{-4}$).

Koopman-based MPC is applied, the output cannot accurately track the reference, because the linear predictor and the Koopman predictor (linear in $u_t$) cannot make an accurate prediction (Table I). Note that one can use, for example, robust Koopman-based MPC approaches, and bilinear Koopman formulations, to possibly improve the performance [58], [59], [60]. By comparison, the RoKDeePC and the kernel-based MPC both achieve satisfactory performance even under measurement noise. In this case study, we observe that the performance is not sensitive to the hyperparameters in the kernel functions. For instance, we observe similar performance when the parameter in the Gaussian kernel varies from 0.01 to 1.

---

We now test the performance of the RoKDeePC in a grid-forming converter that is feeding a local nonlinear load, as shown in Fig. 3. The grid-forming converter employs virtual synchronous machine control for synchronization, which emulates the swing equation of synchronous generators to generate the frequency [62], [63], [64]. The converter is connected to a local nonlinear load, whose active and reactive power...
consumptions are (in per-unit values)

\[ P_{\text{Load}} = 0.3 + 0.2 U^3 + 10 \Delta U^2 + 5 \Delta U \]

\[ Q_{\text{Load}} = 0.04 + 8 \Delta U^2 + 2 \Delta U \]

where \( U \) is the terminal voltage magnitude of the load and \( \Delta U = U - U_0 \) is the voltage deviation from its nominal value \( U_0 = 1 \) p.u. The other system parameters are as follows: \( L_F = 0.2 \) p.u., \( C_F = 0.07 \) p.u., \( L_g = 0.4 \) p.u., \( \omega = 100\pi \) rad/s, \( P_0 = 0.5 \) p.u., \( \omega = 0.2 \), and \( D = 0.08 \).

We apply the RoKDeePC algorithm to perform optimal voltage (or reactive power) control in the converter. As shown in Fig. 3, the input of the converter system is the internal voltage magnitude of the converter (i.e., \( U_g^* \)), and the output can either be the reactive power \( Q_F \) or the terminal voltage deviation of the load \( \Delta U \). Since the converter system is more complex than (39), we choose \( T_{\text{ini}} = 4 \), \( T = 1500 \), \( N = 6 \), and a control horizon of \( k = 1 \). The other settings of the RoKDeePC are the same as those in Section V-A (cost function, excitation signals, and so on). We again assume no input and output constraints and focus on comparing the tracking performance. For the Koopman-based MPC, we consider \( (u_{\text{ini}}, y_{\text{ini}}) \) as the state variables of the system, and the choices of the lifting functions are the same as those in the previous subsection.

Fig. 4(a) shows the reactive power responses of the converter in grid-connected mode (i.e., the breaker in Fig. 3 is closed), and we choose the reactive power \( Q_F \) to be the output of the system. The sampling time of the system is 2 ms. The reference for \( Q_F \) steps from 0 to 0.1 p.u. at \( t = 0.1 \) s, and then to 0.05 p.u. at \( t = 0.3 \) s. It can be seen that the converter has satisfactory performance when the RoKDeePC is applied (with Gaussian, exponential, or polynomial kernel). However, when the Koopman-based MPC or DeePC is applied, there are severe tracking errors; with the former, we also observe some oscillations that persist even with more data to solve the EDMD problem and a longer prediction horizon (e.g., \( T = 20000 \) and \( N = 20 \)). Similar to the results in Section V-A, this is because of the strong nonlinearity of the systems and the special structure of the lifting functions in (40) that are linear in \( u_t \). With the kernel-based MPC in (12) (with Gaussian, exponential, or polynomial kernel), the system became unstable, and thus, the time-domain responses are not displayed in Fig. 4.

Fig. 4(b) shows the time-domain responses of the voltage deviation when its reference steps from 0 to 0.05 p.u. at \( t = 0.1 \) s, and then to 0.025 p.u. at \( t = 0.3 \) s. We observe that the RoKDeePC (with Gaussian, exponential, or polynomial kernel) achieves the best performance, whereas the Koopman-based MPC and DeePC lead to substantial tracking error. The tracking error is about 10% when the kernel in (38) is used in the RoKDeePC to enable faster calculations.
VI. CONCLUSION

RoKDeePC, an algorithm to perform direct data-driven control for nonlinear systems based on implicitly predicting the future outputs of the system using the representer theorem, was presented. RoKDeePC involves solving a data-driven optimal control problem in a receding-horizon manner, which does not require any time-consuming offline learning step. Moreover, we demonstrate how to provide end-to-end robustification for the control sequence against measurement noise in the output data, leading to a performance guarantee for inexact data conditions. This performance guarantee requires the assumption of bounded prediction errors when exact data are used to predict the future outputs. To handle the nonconvexity of the optimal control problem, we exploited its structure and developed projected gradient descent algorithms to enable fast calculations of the (sub)optimal solutions and, thus, real-time implementations in real-world applications. RoKDeePC was tested in two example nonlinear systems (including a high-fidelity converter feeding a nonlinear load) and showed excellent performance. A comparison with related nonlinear data-driven MPC methods showed the favorable performance of our approach. Future work should include explicit conditions on data requirements, closed-loop performance, stability, optimal and autonomous tuning of hyperparameters, and correlated multistep predictors for RoKDeePC. One can also consider advanced versions of RoKDeePC (e.g., with terminal constraints and chance constraints) to improve the performance.

APPENDIX

PROOF OF PROPOSITION 1

Proof: Since \( \mathcal{Y} \) is a polyhedron, we can compactly rewrite \( G \) as \( \hat{G} = \{ g | Gg + N_G(g) \leq q \} \), where \( N_G(g) = [||G^{(1)}g|| \ | ||G^{(2)}g|| \ | \cdots ||G^{(n_q)}g||]^\top \) for some \( G \in \mathbb{R}^{n_q \times n_h}, q \in \mathbb{R}^{n_q}, \) and \( G^{(i)} \in \mathbb{R}^{n \times n_h}, i \in [1:n_q] \) [53].

We start by, respectively, rewriting (32) and (33) as follows:

\[
\min_{g \in \hat{G}} \| \hat{Y}_{\bar{F}g} - r \|_Q^2 + \lambda_h \| Vg - v \|_2^2 + c_u
\]

\[
\min_{g \in \hat{G}} \| \hat{Y}_{\bar{F}g} - r \|_Q + h(g) + \lambda_h \| Vg - v \|_2 + c_u
\]

where \( c_u = \ell(u) \) is a constant, \( V = \hat{K} + \gamma I \), \( \nu = \kappa(u_{\text{ini}}, \hat{y}_{\text{ini}}, u) \), and \( h(g) = \lambda_h \rho_1 (\| g \|_2 + 1)^{1/2} + \rho_2 \| g \|_2 \).

Consider the Lagrangian of (41)

\[
\mathcal{L}_q (g, \mu_q) = \| \hat{Y}_{\bar{F}g} - r \|_Q^2 + \lambda_g \| g \|_2^2 + \lambda_k \| Vg - v \|_2^2 + \mu_q^\top (Gg + N_G(g) - q)
\]

where \( \mu_q \) is the vector of the dual variables. Since \( \hat{G} \) is nonempty, there exists a solution \((g^*, \mu^*_q)\) to the Karush–Kuhn–Tucker (KKT) conditions of (41) which can be reorganized as follows:

\[
\begin{align*}
2\hat{Y}_{\bar{F}g}^\top Q \hat{Y}_{\bar{F}g} - r + \lambda_h \| g \|_2^2 + 2\lambda_k (Gg + N_G(g) - q) = 0 \\
\mu_q^\top (Gg + N_G(g) - q) = 0 \\
\mu_q \geq 0
\end{align*}
\]

where \( g^* \) is a minimizer of (41) and (32).

Consider the Lagrangian of (42)

\[
\mathcal{L}(g, \mu) = \| \hat{Y}_{\bar{F}g} - r \|_Q + \lambda_h \rho_1 \sqrt{\| g \|_2^2 + 1} + \mu_2 \| Vg - v \|_2^2 + \mu^\top (Gg + N_G(g) - q)
\]

where \( \mu \) is the vector of the dual variables. By choosing \( \lambda_h, \rho_1, \) and \( \rho_2 \) according to (34) and (35), it can be verified that \((g^*, \mu^*, y^*, w^*, z^*)\), where

\[
\left( \mu^*, y^* \right) = \begin{cases} \\
\left( \frac{\mu_q^*}{2}, 0 \right), & \text{otherwise} \\
\frac{\mu_q^*}{2}, & \hat{Y}_{\bar{F}g}^* \neq r \\
\frac{\mu_q^*}{2}, & \hat{Y}_{\bar{F}g}^* = r \\
\end{cases}
\]

\[
z^* = \begin{cases}
\lambda_h V^\top (Vg^* - v), & \text{if } Vg^* \neq v \\
0, & \text{otherwise}
\end{cases}
\]

\[
w^* = \rho_2 g^*/\| g^* \|_2 \text{ if } g^* \neq 0, \text{ and } w^* = 0 \text{ otherwise, and } g^* \text{ as before, satisfy the KKT conditions of (42)}
\]

\[
\begin{align*}
q + \lambda_h \rho_1 g + w + z + G^\top \mu \\
+ \sum_{i \in [n_q]} \mu_{q[i]} \left( \frac{G^{(i)}(G^\top g)}{\| G^{(i)}g \|} \right) = 0 \\
\| \hat{Y}_{\bar{F}g}^* - r \|_Q \geq \| \hat{Y}_{\bar{F}g} - r \|_Q + y^\top (\hat{g} - g) \\
\forall \hat{g} \in \mathbb{R}^H \\
\rho_2 \| g \|_2 \geq \rho_2 \| g \|_2 + w^\top (\hat{g} - g) \forall \hat{g} \in \mathbb{R}^H \\
\lambda_h \| Vg - v \|_2 \geq \lambda_h \| Vg - v \|_2 + z^\top (\hat{g} - g) \forall \hat{g} \in \mathbb{R}^H \\
\mu^\top (Gg + N_G(g) - q) = 0 \\
Gg + N_G(g) \leq q \\
\mu \geq 0.
\end{align*}
\]

Thus, the vector \( g^* \) is also a minimizer of (42) and (33).

Next, we prove the monotonic relationship between \( \lambda_h \) and \( \lambda' \). Let \( g_1 \) be the minimizer of (41) with \( \lambda'_h = \lambda'_{h1} > 0 \) (and the minimizer of (42) with \( \lambda_h = \lambda_{h1} \)), and \( g_2 \) the minimizer of (41) with \( \lambda'_h = \lambda'_{h2} > \lambda'_{h1} \) (and the minimizer of (42) with \( \lambda_h = \lambda_{h2} \)). If \( g_1 = g_2 \), we directly obtain \( \lambda_{h1} < \lambda_{h2} \) according to (44). If \( g_1 \neq g_2 \), according to the definitions of \( g_1 \) and \( g_2 \), we have

\[
\| \hat{Y}_{\bar{F}g_1} - r \|_Q^2 + \lambda_h \| g_1 \|_2^2 + \lambda'_{h1} \| Vg_1 - v \|_2^2 < \| \hat{Y}_{\bar{F}g_2} - r \|_Q^2 + \lambda_h \| g_2 \|_2^2 + \lambda'_{h2} \| Vg_2 - v \|_2^2
\]

which can be reorganized as follows:

\[
\| \hat{Y}_{\bar{F}g_1} - r \|_Q^2 + \lambda_h \| g_1 \|_2^2 + \lambda'_{h1} (\| Vg_1 - v \|_2^2 - \| Vg_2 - v \|_2^2) < \| \hat{Y}_{\bar{F}g_2} - r \|_Q^2 + \lambda_h \| g_2 \|_2^2 + \lambda'_{h2} \| Vg_2 - v \|_2^2
\]

Moreover, we have

\[
\| \hat{Y}_{\bar{F}g_1} - r \|_Q^2 + \lambda_h \| g_1 \|_2^2 + \lambda'_{h2} \| Vg_2 - v \|_2^2 < \| \hat{Y}_{\bar{F}g_1} - r \|_Q^2 + \lambda_h \| g_1 \|_2^2 + \lambda'_{h2} \| Vg_1 - v \|_2^2
\]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
which can be reorganized as follows:
\[
\| \hat{Y}_{FG1} - r \|_Q^2 + \lambda_R \| g_1 \|^2 + \lambda'_2 (\| V g_1 - v \| - \| V g_2 - v \|)^2 \\
> \| \hat{Y}_{FG2} - r \|_Q^2 + \lambda_R \| g_2 \|^2. \quad (46)
\]

By combining (45) and (46), we obtain
\[
\lambda'_{k1} (\| V g_1 - v \| - \| V g_2 - v \|)^2 \\
< \lambda'_{k2} (\| V g_1 - v \| - \| V g_2 - v \|)^2
\]
which indicates that \( \| V g_1 - v \|^2 > \| V g_2 - v \|^2 \) as \( \lambda'_{k2} > \lambda'_{k1} > 0 \). Then, we have \( \| V g_1 - v \| > \| V g_2 - v \| \).

According to the definitions of \( \lambda_{k1} \) and \( \lambda_{k2} \), we also have
\[
\| \hat{Y}_{FG1} - r \|_Q + h(g_1) + \lambda_{k1} (\| V g_1 - v \| - \| V g_2 - v \|) \\
< \| \hat{Y}_{FG2} - r \|_Q + h(g_2)
\]
leading to
\[
\lambda_{k1} \left( \| V g_1 - v \| - \| V g_2 - v \| \right) > 0 \\
< \lambda_{k2} (\| V g_1 - v \| - \| V g_2 - v \|)^2
\]
It can then be deduced that \( \lambda_{k1} < \lambda_{k2} \). Hence, \( \lambda_k \) is increasing with the increase of \( \lambda'_k \). By following a similar process, one can also prove that \( \rho_1 (\rho_2) \) is increasing with the increase of \( \lambda_R \). This completes the proof.

ACKNOWLEDGMENT

The authors would like to thank Liviu Aolaritei, Francesco Micheli, and Jianzhe Zhen for fruitful discussions.

REFERENCES

[1] I. Markovsky, L. Huang, and F. Dörfler, “Data-driven control based on the behavioral approach: From theory to applications in power systems,” IEEE Control Syst., vol. 43, no. 5, pp. 28–68, Oct. 2023.

[2] I. Markovsky and F. Dörfler, “Behavioral systems theory in data-driven analysis, signal processing, and control,” Annu. Rev. Control, vol. 52, pp. 42–64, Jan. 2021.

[3] L. Huang, J. Coulson, J. Lygeros, and F. Dörfler, “Decentralized data-enabled predictive control for power system oscillation damping,” IEEE Trans. Control Syst. Technol., vol. 30, no. 3, pp. 1065–1077, May 2022.

[4] E. Elokda, J. Coulson, P. N. Beuchat, J. Lygeros, and F. Dörfler, “Data-enabled predictive control for quadcopters,” Int. J. Robust Nonlinear Control, vol. 31, no. 18, pp. 8916–8936, Dec. 2021.

[5] L. Huang, System Identification: Theory for the User, Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.

[6] M. Morari and J. H. Lee, “Model predictive control: Past, present, and future,” Comput. Chem. Eng., vol. 23, nos. 4-5, pp. 667–682, May 1999.

[7] H. Hjalmarsson, M. Gevers, and F. de Bruyne, “For model-based control design, closed-loop identification gives better performance,” Automatica, vol. 32, no. 12, pp. 1659–1673, Dec. 1996.

[8] S. Formentin and A. Chiuso, “CoRe: Control-oriented regularization for system identification,” in Proc. IEEE Conf. Decis. Control (CDC), Dec. 2018, pp. 2523–2528.

[9] F. Dörfler, J. Coulson, and I. Markovsky, “Bridging direct and indirect data-driven control formulations via regularizations and relaxations,” IEEE Trans. Autom. Control, vol. 68, no. 2, pp. 883–897, Feb. 2023.
