Handbook of linear data-driven predictive control

Annual Reviews in Control 56 (2023) 100914

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Handbook of linear data-driven predictive control: Theory, implementation and design

P.C.N. Verheijen *, V. Breschi, M. Lazar

Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, Eindhoven, 5600 MB, The Netherlands

ARTICLE INFO

Keywords:
Data-driven control
Predictive control
Constrained control
Model free control

ABSTRACT

Data-driven predictive control (DPC) has gained an increased interest as an alternative to model predictive control in recent years, since it requires less system knowledge for implementation and reliable data is commonly available in smart engineering systems. Several data-driven predictive control algorithms have been developed recently, which largely follow similar approaches, but with specific formulations and tuning parameters. This review aims to provide a structured and accessible guide on linear data-driven predictive control methods and practices for people in both academia and the industry seeking to approach and explore this field. To do so, we first discuss standard methods, such as subspace predictive control (SPC), and data-enabled predictive control (DeePC), but we also include newer hybrid approaches to DPC, such as γ–data-driven predictive control and generalized data-driven predictive control. For all presented data-driven predictive controllers we provide a detailed analysis regarding the underlying theory, implementation details and design guidelines, including an overview of methods to guarantee closed-loop stability and promising extensions towards handling nonlinear systems. The performance of the reviewed DPC approaches is compared via simulations on two benchmark examples from the literature, allowing us to provide a comprehensive overview of the different techniques in the presence of noisy data.

1. Introduction

Model predictive control (MPC) (Camacho & Bordons, 2007; Maciejowski, 2002; Rawlings et al., 2017) has become the go-for advanced control method in many application domains, including (but not limited to) mechatronics, robotics, power electronics, automotive systems, and smart infrastructures. The reasons for its popularity are several, comprehending its capability of handling constraints, devising anticipating control actions and delivering optimal performance. At the same time, one of the main issues in designing MPC lies in its inherent dependence on a prediction model, which might be cumbersome to derive from first principles, especially when complex systems have to be controlled. Since obtaining and maintaining accurate first-principles-based models often takes the lion's share of design time and effort, while reliable data are becoming increasingly and readily available in modern, smart engineering systems (see, e.g., Hou and Wang (2013) and Lamnabhi-Lagarrigue et al. (2017) and references therein), data-driven methods have hence recently gained an increased interest.

Roughly speaking, these approaches can be distinguished into two families, based on how data are leveraged. On the one hand, data can be exploited to learn a model for the plant to be controlled by employing system identification tools, e.g., Ljung (1999). The learned model can then be embedded into the MPC problem and used to determine the optimal control action, following a standard (model-based) rationale. Although benefiting from model-based guarantees, this approach is often prone and sensitive to modeling errors, hence requiring the designer to focus more on the system identification phase (e.g., on selecting the right model structure), rather than on the actual control design (Gevers, 2005). Moreover, system identification techniques generally aim at attaining the best possible predictor, while often disregarding its final use.[1] On the other hand, we are recently experiencing the renaissance of Data-driven Predictive Controllers (DPC). This class of approaches directly relies on raw (or slightly pre-processed) data to predict the future response of the controlled system, without requiring any explicit identification of a model prior to control design. Hence, DPC methods directly exploit the available data to fulfill the control objective, while (allegedly) reducing the engineering effort required for control design. Indeed, DPC schemes eliminate the need for a state observer and any explicit identification phase, but they still require the designer to make

* Corresponding author.
E-mail addresses: p.c.n.verheijen@tue.nl (P.C.N. Verheijen), v.breschi@tue.nl (V. Breschi), m.lazar@tue.nl (M. Lazar).

[1] This is not true in control-oriented identification. The reader is referred to Formentin and Chiuso (2021) and the references therein for an overview of control-oriented identification techniques.

https://doi.org/10.1016/j.arcontrol.2023.100914
Received 19 July 2023; Accepted 29 October 2023
Available online 9 November 2023

several choices on a set of critical hyper-parameters. Moreover, where system identification often emphasizes on one-step-ahead prediction, DPC seeks to minimize the predicted tracking errors and control efforts over the entire prediction horizon (Köhler et al., 2022).

According to the rationale introduced in Hou and Wang (2013), DPC approaches can be further classified into *indirect* and *direct* methods. Indirect strategies, like Subspace Predictive Control (SPC) (Favoreel et al., 1999), rely on the identification of prediction matrices directly from data, which are then used to construct the controller. This intermediate (modeling) stage makes these approaches suited to be used in combination with large data-sets, which can thus be leveraged to reduce the predictor's bias without hampering the computational complexity of the design problem. Instead, in direct DPC approaches (such as Data-EnablEd Predictive Control (DeePC) Coulson et al., 2019) future outputs are predicted directly from structured data by relying on behavioral system theory and Willems' fundamental lemma (Willems et al., 2005). In turn, this allows for the employed predictor to be "control-oriented"(i.e., emphasizing on delivering predictions for control), while providing the degrees of freedom to have a direct, tunable trade-off between the bias and variance of the employed predictor. However, direct DPC approaches often scale with the dimension of the available data-set, so that only small sets of data can be used to efficiently solve these problems in real-time. Note that, in the context of DPC the notion of *direct* control is slightly different compared to other data-driven methods (e.g., the VRFT approach Campi et al., 2002), in that DPC techniques still rely on a predictor and they are direct for as far "direct" one can go in predictive control.

Within a deterministic (a.k.a., noise-free) scenario, Berberich et al. (2021b) and Coulson et al. (2019) prove that DeePC is equivalent to MPC, thus connecting the established, model-based predictive control rationale with the data-driven one. Moreover, despite the aforementioned classification, in this deterministic scenario, indirect and direct approaches result to be equivalent, as established in Fiedler and Lucia (2021) for SPC and DeePC. Also in light of these equivalences, a new wave of *hybrid* methods have also recently emerged (see Breschi, Chiuso and Formentin (2023), Dörfler et al. (2022) and Lazar and Verheijen (2023)). These intermediate approaches seek to combine the low dimensional efficiency of indirect methods with the tunable bias/variance trade-off characterizing direct strategies.

The actual differences between direct, indirect and hybrid approaches come into play when noisy data are used. While standard SPC solely relies on the asymptotic (for data-sets of infinite length) unbiasedness of the predictor to cope with noisy data, van Wingerden et al. (2022) introduces an instrumental variable (IV) scheme for noise handling in DeePC, and shows its equivalence with an SPC approach that also employs IVs to counteract noise. Instead, direct and hybrid approaches generally cope with noisy data by introducing a set of slack variables and by augmenting the control cost with regularization terms, whose penalties have then to be tuned to possibly reduce the impact of noise on the controller performance. In particular, Fiedler and Lucia (2021) introduces a set of slack variables to cope with the noise affecting the closed-loop inputs and outputs used to initialize the DeePC scheme at each time step. These slack variables are then regularized to contain their values, ultimately requiring the user to select two (sufficiently large) regularization penalties. By also introducing a slack on the closed-loop outputs used for DeePC initialization, Coulson et al. (2019) further copes with noise via a lasso regularization on the predictor's parameters. Alternative regularization strategies are further introduced in Dörfler et al. (2022), where it also shows that tuning the associated penalties is far from trivial, with sloppy choices of these hyper-parameters potentially resulting in poor closed-loop responses. Meanwhile, Dörfler et al. (2022) also proposes a subspace-driven regularization that recovers the predictions characteristics to SPC for high tuning weights, thus providing some insights on the effect of different choices of the regularization penalty. However, due to the nature of the employed predictor, all the schemes introduced in Dörfler et al. (2022) can only be employed in combination with small data-sets, while it is well known that the use of large data-sets is preferable when handling noisy data to attain reliable predictions, see, e.g., Berberich et al. (2021b), Dörfler et al. (2022) and Lazar and Verheijen (2022). In an effort to reduce the ability to leverage large data-sets, Lazar and Verheijen (2023) embeds SPC predictions generated by using a large set of data with a DeePC scheme of manageable size, to guarantee a tunable bias/variance trade-off. Instead, via LQ-decomposition, Breschi, Chiuso and Formentin (2023) derives a DPC method directly conceived within a stochastic setting (differently from the others) that essentially introduces a tunable bias/variance into SPC while minimizing the online computational load. On this line, a.k.a., by pre-manipulating the data matrices, Zhang et al. (2022) proposes to perform a singular value decomposition on the original Hankel data matrix. DeePC is then designed based on the resulting reduced Hankel matrix. Therein, it was shown that this approach can significantly reduce the computational complexity of DeePC, while improving the accuracy of predictions for noisy data. By solely focusing on a reduction of computational complexity, Baros et al. (2022) introduces an efficient numerical method that exploits the structure of Hankel matrices to efficiently solve quadratic programs (QPs) specific to DeePC.

Analyzing closed-loop stability of DPC algorithms is also a very active research subject, with stability guarantees for DeePC first reported in Berberich et al. (2021b) by means of terminal equality constraints and input–output-to-state stability Lyapunov functions. A relaxation to terminal set conditions for DeePC is further provided in Berberich et al. (2021c). An alternative, dissipativity-based, approach is proposed in Lazar (2021) and Lazar and Verheijen (2023), where terminal inequality constraints and dissipation inequalities involving storage and supply functions are used.

Aside from methodological developments, DPC techniques have also been already successfully tested on a broad range of real-world control problems. Examples span from the application of SPC for the control of nuclear reactors (Vajpayee et al., 2018) and cooling data-centers (Li et al., 2023), to those of DeePC for building temperature regulation (Di Natale et al., 2022), battery charging management (Zhang et al., 2023), quadcopter control (Elokda et al., 2021), or for the regulation of synchronous motor drives (Carlet et al., 2022). Still on the application side, Markovsky et al. (2023) discusses in detail the implementation aspects of DeePC when used for power electronics and power systems control. Readers interested in these applications are referred to Markovsky et al. (2023) and the references therein for additional information.

The remainder of this paper is structured as follows. Section 2 presents the standard notation used in MPC, setting the basis for the introduction of the DPC approaches summarized in Section 3. In Section 4 we then introduce and compare the tuning guidelines of different approaches for noisy data, providing an overview of conditions for guaranteeing closed-loop stability, and pinpointing the reader to some extensions of DPC beyond linear systems. Section 5 subsequently presents a comparison of the closed-loop performance attained with each of the reviewed methods, focusing on two benchmark systems from the literature. The paper ends with some concluding remarks and directions for future work in Section 6.

Notation. Throughout the paper, the sets of natural and real numbers are respectively denoted as $\mathbb{N}$ and $\mathbb{R}$, while we indicate the set of real vectors of dimension $n \times 1$ as $\mathbb{R}^n$. Let $\mathbb{N}_{\geq 1}$ denote the set of natural numbers excluding zero. For any finite collection of $q \in \mathbb{N}_{\geq 1}$ vectors $\{\xi_1, \dots, \xi_q\} \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_q}$ we will make use of the operator $\mathrm{col}(\xi_1, \dots, \xi_q) := [\xi_1^\top, \dots, \xi_q^\top]^\top$ to indicate the vector stacking them in a column. Given a matrix $A \in \mathbb{R}^{n \times m}$, we indicate its Frobenius norm as $\|A\|_{\mathrm{Fro}}$ and its generalized pseudo-inverse as $A^\dagger$. A Hankel matrix constructed using an arbitrary dataset z, with depth N, length T and

starting index i is denoted as:

$$\mathcal{H}^i_{[N,T]}(z) := \begin{bmatrix} z(i) & z(i+1) & \dots & z(i+T-1) \\ z(i+1) & z(i+2) & \dots & z(i+T) \\ \vdots & \vdots & & \vdots \\ z(i+N-1) & z(i+N) & \dots & z(i+T+N-2) \end{bmatrix} \tag{1}$$

where the samples in z can comprise information incoming from multiple channels (e.g., multiple inputs or outputs), i.e., $z(i) = \begin{bmatrix} z_0(i) & z_1(i) & \dots & z_r(i) \end{bmatrix}^T$.

2. Preliminaries on MPC for linear systems

Consider the discrete-time, stochastic linear dynamical system in innovation form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Ke(k), \quad k \in \mathbb{N}, \\ y(k) &= Cx(k) + e(k), \end{aligned} \tag{2}$$

where $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^{n_u}$ are the system's state and control input at time k, respectively, $e(k) \in \mathbb{R}^{n_y}$ is the innovation at the same time instant, here assumed to be the realization of a zero-mean, white stochastic process, $y(k) \in \mathbb{R}^{n_y}$ is the corresponding measured output, and (A, B, K, C) are real matrices of suitable dimensions. Let us assume that (A, B) is controllable, (A, C) is observable, and the matrix $A - KC$ is strictly stable, i.e., all its eigenvalues lie inside the unit circle.

Given a prediction horizon $N \in \mathbb{N}_{\geq 1}$, one of the most streamlined versions of deterministic MPC searches for the optimal input to the system by solving:

$$\underset{\mathbf{u}(k),\mathbf{y}(k)}{\text{minimize}} \quad l_N(y(N|k)) + \sum_{i=0}^{N-1} l(y(i|k), u(i|k)) \tag{3a}$$

$$\text{subject to} \quad \mathbf{y}(k) = \Phi x(k) + \Gamma \mathbf{u}(k), \tag{3b}$$

$$(\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N, \tag{3c}$$

where

$$\mathbf{u}(k) := \text{col}(u(0|k), \dots, u(N-1|k)), \qquad \mathbf{y}(k) := \text{col}(y(1|k), \dots, y(N|k)),$$

denote the N-step-ahead input trajectory computed at time k and the vector of corresponding output predictions $y(i|k)$ at future time $k+i$, for $i = 1, \dots, N$, respectively. By iterating system (2) and omitting the noise-dependent components, the latter is expressed in (3b) as a function of the future inputs $\mathbf{u}(k)$ and the current state $x(k)$, namely (Maciejowski, 2002)

$$\mathbf{y}(k) = \Phi x(k) + \Gamma \mathbf{u}(k), \quad \text{where} \quad \Phi := \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix},$$
$$\Gamma := \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix}. \tag{4}$$

In this review, we focus on the MPC formulation with the typical quadratic stage cost function, i.e.,

$$l(y,u) := (y - r_y)^\top Q (y - r_y) + (u - r_u)^\top R (u - r_u), \tag{5}$$

for some positive definite Q, R matrices and user-defined references $r_y \in \mathbb{R}^{n_y}$ and $r_u \in \mathbb{R}^{n_u}$, that can be either constant or time-varying. The objective function in (3a) additionally features the terminal cost $l_N(y(N|k))$, allowing one to weight the last output prediction $y(N|k)$ differently from the others. If the output is equal to the state, this terminal cost is typically computed using an LQR local control law (Rawlings et al., 2017) to guarantee closed-loop stability. However, in practice this is typically not the case, a common choice for the terminal cost is a scaled version of the output stage cost, i.e., $l_N(y) := \alpha l(y, 0)$, with $\alpha \geq 1$. The feasible sets $\mathbb{Y}$ and $\mathbb{U}$ are here supposed to be convex compact sets that contain the desired references in their interior; most often these sets are polytopes.

Once (3) is solved and, thus, the optimal control sequence $\mathbf{u}^*(k)$ is obtained, only $u^*(0|k)$ is retained and used to feed the system (i.e., $u(k) = u^*(0|k)$), while the remaining predicted inputs are discarded. By working in a receding-horizon fashion, this procedure is repeated again at the next time step, with Problem (3) solved again based on the updated initial state $x(k+1)$.

By looking at (3), it is clear that one is bound to have access to the state $x(k)$ (or its estimate) at each time instant to solve the predictive control problem. However, the state might not be directly measurable in practice, ultimately making it necessary to design a state estimator (like a Kalman filter) and, thus, increasing the complexity of the design procedure. At the same time, a model of system (2) is also clearly needed to construct the problem in (3) at the core of MPC. Such a model is commonly obtained either using first principles (i.e., by hand) or exploiting data in combination with system identification techniques. However, both these approaches have well-known drawbacks. Indeed, first principles modeling becomes cumbersome when the complexity of the system increases, while identification often takes much of the design time and identification techniques generally disregard the ultimate use of the model.

In the next section, we introduce four different data-driven predictive control strategies, that aim at coping with the aforementioned limitations by directly using input–output data instead of an explicit model of the system and a state estimator.

3. Data-driven predictive control

To overcome the first limitation of MPC mentioned in the previous section, the dependence of problem (3) on the (possibly inaccessible) state $x(k)$ has to be replaced with that on the measured inputs and outputs. To this end, along with the "future horizon" N already featured in (3), let us consider the "past horizon" $T_{\text{ini}} \in \mathbb{N}_{\geq 1}$. According to this new horizon, we construct the following sequences of past inputs/outputs

$$\mathbf{u}_{\text{ini}}(k) := \text{col}(u(k-T_{\text{ini}}), \dots, u(k-1)), \quad \mathbf{y}_{\text{ini}}(k) := \text{col}(y(k-T_{\text{ini}}+1), \dots, y(k)),$$

that we will then use to characterize the initial condition of the system at time k. It is worth pointing out that T_{ini} is inherently linked to the order of the system. In particular, it should be selected so that $T_{ini} \geq n$, as discussed in more detail in the next section.

Meanwhile, to shift from the model-based predictor in (3b) to its data-driven counterpart, we suppose to have access to an *informative* set of input/output pairs $\{\bar{u}, \bar{y}\}$ of length T, where informativity is here guaranteed by the fact that the available input sequence is persistently exciting of order $T_{\text{ini}} + N + n$ according to the following definition.

Definition 3.1 (*Persistence of Excitation*)**.** The input data sequence $\bar{u}$ is called persistently exciting of order L if the rank of the Hankel matrix satisfies

$$\text{rank}\left(\mathcal{H}^0_{[L,T]}(\bar{u})\right) = Ln_u. \tag{6}$$

In turn, this definition implies that T is sufficiently long, i.e.,

$$T \geq n_u(T_{\text{ini}} + N + n). \tag{7}$$

These data are used to construct the following Hankel matrices:

$$\begin{aligned} \mathbf{U}_p &:= \mathcal{H}^0_{[T_{\text{ini}},T]}(\bar{u}), \quad & \mathbf{Y}_p &:= \mathcal{H}^1_{[T_{\text{ini}},T]}(\bar{y}), \\ \mathbf{U}_f &:= \mathcal{H}^{T_{\text{ini}}}_{[N,T]}(\bar{u}), \quad & \mathbf{Y}_f &:= \mathcal{H}^{T_{\text{ini}}+1}_{[N,T]}(\bar{y}). \end{aligned} \tag{8}$$

These matrices are at the core of the four data-driven predictive control approaches reviewed in this section, namely SPC (Favoreel et al., 1999), DeePC (Coulson et al., 2019), γ-DDPC (Breschi, Fabris et al., 2023) and GDPC (Lazar & Verheijen, 2023). Apart from reviewing their main features, for each data-driven control strategy, we include a table summarizing numbers and properties to facilitate their comparison by the interested user.

Remark 3.2 (*Handguide to Construct the Hankel Matrices*)**.** Let us consider the generic input/output time series denoted as:

$$U := \begin{bmatrix} \bar{u}(0) & \dots & \bar{u}(T+T_{\text{ini}}+N) \end{bmatrix}, \quad Y := \begin{bmatrix} \bar{y}(0) & \dots & \bar{y}(T+T_{\text{ini}}+N) \end{bmatrix}$$

In MATLAB, the corresponding Hankel matrices (8) can be constructed with the following code-snippet:

```
Up = zeros(Tini*nu, T); Uf = zeros(N*nu, T);
Yp = zeros(Tini*ny, T); Yf = zeros(N*ny, T);

for i = 1:Tini
    Up((i-1)*nu+1:i*nu, :) = U(:, i   :i+T-1);
    Yp((i-1)*ny+1:i*ny, :) = Y(:, i+1:i+T);
end
for i = 1:N
    Uf((i-1)*nu+1:i*nu, :) = U(:, ...
        i+Tini:i+Tini+T-1);
    Yf((i-1)*ny+1:i*ny, :) = Y(:, ...
        i+Tini+1:i+Tini+T);
end
```

3.1. Subspace predictive control

Subspace Predictive Control (SPC) (Favoreel et al., 1999) can be seen as the precursor of recently developed data-driven predictive control techniques, being one of the first attempts to directly incorporate results from subspace identification (van Overschee & De Moor, 1996) into the MPC design problem. SPC is an *indirect* data-driven approach, that revolves around the estimation of prediction matrices from input/output data rooted in the following relationship:

$$\mathbf{y}(k) = \underbrace{\begin{bmatrix} P_1 & P_2 & \Gamma \end{bmatrix}}_{:=\Theta} \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \\ \mathbf{u}(k) \end{bmatrix}, \tag{9}$$

according to which future outputs can be described as a linear combination of previous inputs/outputs and future inputs, with Γ defined as in (4). By relying on this relation, the estimation of the unknown matrix Θ can be carried out by solving the least-squares problem (Favoreel et al., 1999; Verheijen et al., 2021):

$$\begin{aligned} &\min_{\Theta} \left\| \mathbf{Y}_f - \Theta \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix} \right\|_{\text{Fro}}^2, \quad \text{with solution} \quad \Theta^* = \begin{bmatrix} P_1 & P_2 & \hat{\Gamma} \end{bmatrix} \\ &= \mathbf{Y}_f \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}^{\dagger}, \end{aligned} \tag{10}$$

where $\hat{\Gamma}$ is the data-driven estimation of Γ. Note that, in a noise-free setting, $P_1\mathbf{u}_{\text{ini}}(k) + P_2\mathbf{y}_{\text{ini}}(k) = \Phi x(k)$ and $\hat{\Gamma} = \Gamma$ (see Verheijen et al. (2021)), hence allowing to establish the equivalence between MPC and SPC. In a deterministic case, the prediction matrices P_1 and P_2 can be derived from the state-space system with the following definitions (Verheijen et al., 2021):

$$\begin{aligned} &\mathbf{X}_p := \begin{bmatrix} x(0) & \dots & x(T) \end{bmatrix}, \quad \mathbf{X}_f := \begin{bmatrix} x(T_{\text{ini}}) & \dots & x(T+T_{\text{ini}}) \end{bmatrix}, \\ &\mathcal{C}_{\text{ini}} = \begin{bmatrix} A^{T_{\text{ini}}-1}B & \dots & AB & B \end{bmatrix}, \end{aligned} \tag{11}$$

and

$$\begin{aligned} \mathbf{Y}_p &= \Phi_{\text{ini}}\mathbf{X}_p + \Gamma_{\text{ini}}\mathbf{U}_p, \\ \mathbf{Y}_f &= \Phi\mathbf{X}_f + \Gamma\mathbf{U}_f, \\ \mathbf{X}_f &= A^{T_{\text{ini}}}\mathbf{X}_p + \mathcal{C}_{\text{ini}}\mathbf{U}_p. \end{aligned} \tag{12}$$

Under the assumptions that system (2) is observable and $T_{\text{ini}} \geq n$, then there exists an M such that $A^{T_{\text{ini}}} + M\Phi_{\text{ini}} = 0$ and the following relations hold:

$$\begin{aligned} \mathbf{Y}_f &= \Phi A^{T_{\text{ini}}}\mathbf{X}_p + \Phi\mathcal{C}_{\text{ini}}\mathbf{U}_p + \Gamma\mathbf{U}_f, \\ &= \Phi\left(\mathcal{C}_{\text{ini}} + M\Gamma_{\text{ini}}\right)\mathbf{U}_p - \Phi M\mathbf{Y}_p + \Gamma\mathbf{U}_f. \end{aligned} \tag{13}$$

Algorithm 1 SPC algorithm

Input: $\mathbf{U}_p, \mathbf{Y}_p, \mathbf{U}_f, \mathbf{Y}_f$, $(N, T_{\text{ini}}, Q, R, r_y, r_u)$

Before closing the control loop:

1: Obtain Θ^* from (10) and extract P_1, P_2 and $\hat{\Gamma}$.

2: Construct controller as in (14).

During the control loop:

1: Measure $y(k)$ and construct $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$.

2: Solve control law (14) and obtain $\mathbf{u}^*(k)$.

3: Apply $u(k) = u^*(0|k)$ to the system.

Table 1
SPC in a nutshell.

SPC	
Problem size	$N(n_u + n_y)$
Memory requirement	$(Nn_y) \times (T_{\text{ini}}(n_u + n_y) + Nn_u)$
Tunable bias–variance trade-off	No
Suitable for large data-sets	Yes
Additional tuning weights	–

Meanwhile, if the (more realistic) scenario in which the available data are noisy is considered, the predictor in (10) is known to asymptotically (i.e., for $T \to \infty$) result in an unbiased estimate of the true system matrices.

The SPC control problem is thus similar to (3), yet replacing (3b) with (9), namely

$$\underset{\mathbf{u}(k),\mathbf{y}(k)}{\text{minimize}} \quad l_N(y(N|k)) + \sum_{i=0}^{N-1} l(y(i|k), u(i|k)) \tag{14a}$$

$$\text{subject to} \quad \mathbf{y}(k) = P_1\mathbf{u}_{\text{ini}}(k) + P_2\mathbf{y}_{\text{ini}}(k) + \hat{\Gamma}\mathbf{u}(k), \tag{14b}$$

$$(\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N. \tag{14c}$$

For implementation details for this type of controllers, the interested reader is referred to Algorithm 1, while an overview of numbers and properties of SPC is provided in Table 1.

Remark 3.3 (*Reducing Computational Time with SPC*)**.** As for MPC, the control problem in (14) can be easily rewritten in a condensed form, reducing the optimization variables to $\mathbf{u}(k)$. This manipulation is generally not as trivial for the forthcoming data-driven predictive control schemes.

3.2. Data-enabled predictive control

Instead of relying on an estimate of the prediction matrices as in SPC, Data-EnablEd Predictive Control (DeePC) is a direct data-driven control method that predicts future output sequence $\mathbf{y}(k)$ directly from Hankel matrix data. This is possible thanks to the following deterministic result.

Lemma 3.4 (*Willems' Fundamental Lemma van Waarde, De Persis et al., 2020; Willems et al., 2005*)**.** *Consider system (2) for $e(k) = 0$, $\forall k \in \mathbb{N}$, and let $\mathbf{X}_p := \begin{bmatrix} x(0) & \dots & x(T) \end{bmatrix}$. If the measured input sequence $\bar{u}$ is persistently exciting of order $T_{ini} + N + n$, then the following statements hold.*

(i) *The matrix $\begin{bmatrix} \mathbf{X}_p^T & \mathbf{U}_p^T & \mathbf{U}_f^T \end{bmatrix}^T$ has full row rank, i.e.,*

$$\text{rank}\left(\begin{bmatrix} \mathbf{X}_p \\ \mathbf{U}_p \\ \mathbf{U}_f \end{bmatrix}\right) = n + n_u(T_{ini} + N), \tag{15}$$

in turn, implying that

$$\operatorname{rank}\left(\begin{bmatrix}\mathbf{U}_p\\ \mathbf{U}_f\\ \mathbf{Y}_f\end{bmatrix}\right)=\operatorname{rank}\left(\begin{bmatrix}0 & I & 0\\ 0 & 0 & I\\ \Phi A^{T_{ini}} & \Phi\mathcal{C}_{ini} & \Gamma\end{bmatrix}\begin{bmatrix}\mathbf{X}_p\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\right)=n+n_u(T_{ini}+N), \tag{16}$$

and, through the relations in (12), resulting in

$$\operatorname{rank}\left(\begin{bmatrix}\mathbf{Y}_p\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\right)=\operatorname{rank}\left(\begin{bmatrix}\Phi_{ini} & \Gamma_{ini} & 0\\ 0 & I & 0\\ 0 & 0 & I\end{bmatrix}\begin{bmatrix}\mathbf{X}_p\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\right)=n+n_u(T_{ini}+N), \tag{17}$$

where $\mathcal{C}_{ini}=\begin{bmatrix}A^{T_{ini}-1}B & \dots & AB & B\end{bmatrix}$.

(ii) *Every input/output trajectory* $\{\mathbf{u}_{ini}(k),\mathbf{y}_{ini}(k),\mathbf{u}(k),\mathbf{y}(k)\}$ *is a trajectory of system (2) if and only if*

$$\begin{bmatrix}\mathbf{U}_p\\ \mathbf{Y}_p\\ \mathbf{U}_f\\ \mathbf{Y}_f\end{bmatrix}\mathbf{g}(k)=\begin{bmatrix}\mathbf{u}_{ini}(k)\\ \mathbf{y}_{ini}(k)\\ \mathbf{u}(k)\\ \mathbf{y}(k)\end{bmatrix}. \tag{18}$$

for some real vector $\mathbf{g}(k)\in\mathbb{R}^T$.

Proof sketch.

(i) Define:

$$\mathbf{X}_p:=A^n\mathbf{X}_{p-n}+\mathcal{C}_n\mathbf{U}_{p-n}, \tag{19}$$

as the relation between $\mathbf{X}_p$ and its delayed version by n (the system order), considering the controllability matrix $\mathcal{C}_n:=\begin{bmatrix}A^{n-1}B & \dots & AB & B\end{bmatrix}$. Assuming that the system is controllable, i.e., $\operatorname{rank}(\mathcal{C}_n)=n$, and additionally supposing that $\mathbf{U}_{p-n}$ has full row rank, and that the rank of $\begin{bmatrix}\mathbf{U}_{p-n}^T & \mathbf{U}_p^T & \mathbf{U}_f^T\end{bmatrix}^T$ is equal to $n_u(T_{\text{ini}}+N+n)$, then, it holds that:

$$\begin{aligned}\operatorname{rank}\left(\begin{bmatrix}\mathbf{X}_p\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\right)&=\operatorname{rank}\left(\begin{bmatrix}A^n & \mathcal{C}_n & 0 & 0\\ 0 & 0 & I & 0\\ 0 & 0 & 0 & I\end{bmatrix}\begin{bmatrix}\mathbf{X}_{p-n}\\ \mathbf{U}_{p-n}\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\right)\\&=\operatorname{rank}\left(\begin{bmatrix}\mathcal{C}_n & 0 & 0\\ 0 & I & 0\\ 0 & 0 & I\end{bmatrix}\begin{bmatrix}\mathbf{U}_{p-n}\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\right).\end{aligned} \tag{20}$$

Following Definition 3.1, the rank condition on $\begin{bmatrix}\mathbf{U}_{p-n}^T & \mathbf{U}_p^T & \mathbf{U}_f^T\end{bmatrix}^T$ is equivalent to stating that the input must be persistently exciting of order $T_{\text{ini}}+N+n$. Also, $A^n\mathbf{X}_{p-n}$ does not add any necessary information to attain full rank in (20). Furthermore, it also cannot invalidate the rank condition, which is proven in, for example, van Waarde, De Persis et al. (2020, Theorem 1) and Berberich et al. (2023, Section III).

(ii) Any input/output pair $\{\hat{\mathbf{u}},\hat{\mathbf{y}}\}$ is a trajectory of the system if there exists a $\hat{x}_0$ such that:

$$\hat{\mathbf{y}}=\Phi\hat{x}_0+\Gamma\hat{\mathbf{u}}. \tag{21}$$

Since $\begin{bmatrix}\mathbf{X}_p^T & \mathbf{U}_p^T & \mathbf{U}_f^T\end{bmatrix}^T$ has full row rank, we know that there exists a $\mathbf{g}(k)$ such that

$$\begin{bmatrix}\mathbf{X}_p\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\mathbf{g}(k)=\begin{bmatrix}x(k-T_{\text{ini}})\\ \mathbf{u}_{\text{ini}}(k)\\ \mathbf{u}(k)\end{bmatrix}, \tag{22}$$

for any pair of inputs and states (De Persis & Tesi, 2020). Therefore:

$$\begin{bmatrix}\mathbf{U}_p\\ \mathbf{Y}_p\\ \mathbf{U}_f\\ \mathbf{Y}_f\end{bmatrix}\mathbf{g}(k)=\begin{bmatrix}0 & I & 0\\ \Phi_{\text{ini}} & \Gamma_{\text{ini}} & 0\\ 0 & 0 & I\\ \Phi A^{T_{\text{ini}}} & \Phi\mathcal{C}_{\text{ini}} & \Gamma\end{bmatrix}\begin{bmatrix}\mathbf{X}_p\\ \mathbf{U}_p\\ \mathbf{U}_f\end{bmatrix}\mathbf{g}(k)=$$

$$=\begin{bmatrix}0 & I & 0\\ \Phi_{\text{ini}} & \Gamma_{\text{ini}} & 0\\ 0 & 0 & I\\ \Phi A^{T_{\text{ini}}} & \Phi\mathcal{C}_{\text{ini}} & \Gamma\end{bmatrix}\begin{bmatrix}x(k-T_{\text{ini}})\\ \mathbf{u}_{\text{ini}}(k)\\ \mathbf{u}(k)\end{bmatrix}=\begin{bmatrix}\mathbf{u}_{\text{ini}}(k)\\ \mathbf{y}_{\text{ini}}(k)\\ \mathbf{u}(k)\\ \mathbf{y}(k)\end{bmatrix}. \tag{23}$$

Furthermore, through the rank condition, we know that if $\{\mathbf{u}_{\text{ini}}(k),\ \mathbf{y}_{\text{ini}}(k)\}$ is a trajectory of the system, $x(k-T_{\text{ini}})$ is determined. Then, for any $\mathbf{u}(k)$, a unique $\mathbf{y}(k)$ exists in the behavior of the system. □

Remark 3.5. Through the lens of the proof, the first condition of the lemma is not "if and only if", since the matrix can be full rank for lower order of excitation if

$$\operatorname{rank}(\mathcal{C}_m)=n,\quad \text{with } m<n,\quad (\text{e.g. } \operatorname{rank}(B)=n)$$

or the initial state $x(0)$ adds linearly independent information through $A^n\mathbf{X}_{p-n}$ (e.g. the initial state is not an equilibrium). However, in a data-driven setting, neither of these are assumed to be known in general.

Despite being a deterministic result, Willems' Lemma establishes the minimum requirement for the input data sequence to be persistently exciting for the unknown system one aims at controlling, ultimately providing a first guideline on how to design the data collection experiment. At the same time, it provides a strategy to predict future outputs directly from data through (18), on which DeePC relies. However, because of its deterministic nature, Willems' Lemma is not sufficient on its own when the available data are noisy. Indeed, as soon as the available data are corrupted by noise, the property in (16) ceases to hold, and the matrix $\begin{bmatrix}\mathbf{U}_p^T & \mathbf{U}_f^T & \mathbf{Y}_f^T\end{bmatrix}^T$ becoming full row rank. In turn, this implies that any arbitrary pair of input/output data can be a "trajectory of the system", likely leading to inaccurate predictions.

To circumvent that problem, DeePC features a regularization term steering towards $\mathbf{g}(k)$ to be small, leading to the following optimization problem:

$$\begin{aligned}\underset{\mathbf{u}(k),\mathbf{y}(k),\mathbf{g}(k),\sigma_y(k)}{\text{minimize}}\quad & l_N(y(N|k))+\sum_{i=0}^{N-1}l(y(i|k),u(i|k))+\lambda_g l_g(\mathbf{g}(k))\\ &+\lambda_y\|\sigma_y(k)\|_2^2 && \text{(24a)}\\ \text{subject to}\quad & \begin{bmatrix}\mathbf{U}_p\\ \mathbf{Y}_p\\ \mathbf{U}_f\\ \mathbf{Y}_f\end{bmatrix}\mathbf{g}(k)=\begin{bmatrix}\mathbf{u}_{\text{ini}}(k)\\ \mathbf{y}_{\text{ini}}(k)+\sigma_y(k)\\ \mathbf{u}(k)\\ \mathbf{y}(k)\end{bmatrix}, && \text{(24b)}\\ & (\mathbf{y}(k),\mathbf{u}(k))\in\mathbb{Y}^N\times\mathbb{U}^N. && \text{(24c)}\end{aligned}$$

where $l_g(\mathbf{g}(k))$ is an user-defined regularization function over $\mathbf{g}(k)$. This regularizer is commonly chosen as $\|\mathbf{g}(k)\|_2^2$, so often that we will from now on refer to the scheme with this regularization as *"DeePC"*. Additionally, $\sigma_y(k)$ is an auxiliary variable introduced to prevent the problem from turning infeasible if $\{\mathbf{u}_{\text{ini}}(k),\mathbf{y}_{\text{ini}}(k)\}$ is not in the behavioral set (e.g., in the presence of a disturbance). Note that, this additional variable introduces an additional degree of freedom in the optimization problem and, thus, λ_y is often chosen very large not to give the estimator too much freedom. At the same time, $\sigma_y(k)$ becomes unnecessary when T_{ini} and T are properly chosen (see Breschi, Chiuso and Formentin (2023, Lemma 4)), even in the presence of noise.

Remark 3.6 (*Practical Note on the System Order in a Data-Driven Framework*)**.** Without knowledge of the system, accurately estimating the state order n is not trivial. Instead, we advise generating an input signal such that

$$\operatorname{rank}\left(\begin{bmatrix}\mathbf{U}_p\\ \mathbf{Y}_p\\ \mathbf{U}_f\end{bmatrix}\right)=n_u(T_{\text{ini}}+N)+T_{\text{ini}}n_y$$

and $T\geq T_{\text{ini}}(n_y+n_u)+Nn_u$, while choosing T_{ini} large enough, i.e., $T_{\text{ini}}\gg n$. See Section 4.1 for more information about generating informative data.

Remark 3.7 (*Π–Regularization*)**.** Although minimizing $\mathbf{g}(k)$ in the cost function is essential to get accurate predictions, considering a Tikhonov regularization $l_g(\mathbf{g}(k)) = \|\mathbf{g}(k)\|_p$ does not result in an unbiased predictor (Dörfler et al., 2022). To circumvent this issue, the following alternative is proposed in Dörfler et al. (2022):

$$l_g := \|(I - \Pi)\mathbf{g}(k)\|_2^2, \quad \text{where } \Pi := \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}, \tag{25}$$

that can be used when $T > T_{\text{ini}}(n_y + n_u) + Nn_u$, in turn ensuring that the right-kernel of $\begin{bmatrix} \mathbf{U}_p^T & \mathbf{Y}_p^T & \mathbf{U}_f^T \end{bmatrix}^T$ exists even under noisy conditions (and preventing $\Pi = I$). For increasing values of λ_g, this regularization choice steers $\mathbf{g}(k)$ towards the SPC (unbiased) predictor. However, its characteristic weights make the regularization term completely dense and, thus, the corresponding quadratic program much more time-consuming to solve with respect to other data-driven predictive control approaches.

Remark 3.8 (*On the Precursors of DeePC*)**.** Although the first complete rendition of the DeePC algorithm is provided in Coulson et al. (2019) and Yang and Li (2015) already introduced a control algorithm leveraging the behavioral framework from Willems et al. (2005). Nonetheless, differently from DeePC, the control scheme proposed in Yang and Li (2015) relies on the splitting of $\mathbf{g}(k)$ into a component over the past window and one over the future window, i.e.,

$$\begin{bmatrix} \mathbf{u}(k) \\ \mathbf{y}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{U}_f \\ \mathbf{Y}_f \end{bmatrix} (\Delta\mathbf{g}(k) + \mathbf{g}_0(k)), \quad \text{where } \mathbf{g}_0(k) = \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \end{bmatrix}. \tag{26}$$

However, this formulation does not include a constraint forcing $\Delta\mathbf{g}(k)$ to belong to the null space of $\begin{bmatrix} \mathbf{U}_p^T & \mathbf{Y}_p^T \end{bmatrix}^T$, hence it does not result in trajectories that are consistent with the considered initial conditions $(\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k))$.

Remark 3.9 (*On the Notion of Informativity*)**.** As of this point, we have only considered that our data is the trajectory of a single system, the true one. However, any system that contains the data/trajectory in its behavioral representation is a valid candidate for the "true" system. Compared to SPC, which provides the optimal linear unbiased estimator that fits the data-set, DeePC optimizes over a set of predicted trajectories spanned by a linear system of equations defined by the Hankel matrix data. If there is no noise, these trajectories collapse to a unique trajectory. In the presence of noisy data, DeePC allows for optimizing the bias/variance trade-off as it optimizes over a set of trajectories close to the true system trajectory. The regularization penalty in the cost function hence prevents the optimal trajectory from deviating too much from the unbiased estimated system trajectory (note that we do consider noise to be Gaussian, and thus, unbounded). This is different than other informativity-based data-driven control frameworks (van Waarde, Eising et al., 2020), which aim to design a robust controller that stabilizes all possible systems that are compatible with the data. However, such frameworks assume that the noise is bounded.

The reader is referred to Algorithm 2 for a sketch of the main steps to be carried out in DeePC, while its main features are listed in Table 2. This summary clearly highlights one of the main limitations of DeePC with respect to its practical application, i.e., the fact that the complexity of the DeePC quadratic program scales with the amount of data used to describe the system's behavior. DeePC thus requires the designer to undertake an important trade-off between *online computational efficiency* and *prediction accuracy*. Indeed, the length of the Hankel matrices T must be larger than both T_{ini} and N, and T_{ini} should be chosen as large as possible to enhance prediction accuracy. At the same time, the larger the data-set used, the more complex (24) becomes, especially when the Π-regularization in (25) is used.

Algorithm 2 DeePC algorithm

Input: $\mathbf{U}_p, \mathbf{Y}_p, \mathbf{U}_f, \mathbf{Y}_f, (N, T_{\text{ini}}, Q, R, r_y, r_u, \lambda_g, \lambda_y)$

Before closing the control loop:

1: *(Optionally)* Construct Π as in (25).
2: Build controller as in (24).
3: Tune λ_g.

During the control loop:

1: Measure $y(k)$ and construct $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$.
2: Solve control law (24) and obtain $\mathbf{u}^*(k)$.
3: Apply $u(k) = u^*(0|k)$ to the system.

3.3. γ–Data driven predictive control

The first hybrid method bridging between SPC and DeePC we introduce in this review is γ-DDPC (Breschi, Chiuso and Formentin, 2023; Breschi, Fabris et al., 2023). By rooting its derivation from a stochastic perspective on the predictive control problem, γ-DDPC relies on an a-priori operation on the Hankel matrices to enhance prediction accuracy, whilst not requiring an estimate of the prediction matrices and, thus, leveraging the degrees of freedom that characterize DeePC.

To review this alternative data-driven control technique, let us first define

$$\mathbf{Z}_p := \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \end{bmatrix}, \quad \mathbf{z}_{\text{ini}}(k) := \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \end{bmatrix},$$

respectively grouping the "past" data and the initial condition characterizing (18). Moreover, let us further introduce the following LQ decomposition of the Hankel matrices featured in the predictor of DeePC[2]:

$$\begin{bmatrix} \mathbf{Z}_p \\ \mathbf{U}_f \\ \mathbf{Y}_f \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix}. \tag{27}$$

where the matrices $\{L_{ii}\}_{i=1}^3$ are all non-singular (under the stochastic conditions of persistence of excitation stated in Breschi, Chiuso and Formentin (2023, Lemma 3)), and Q_i have orthonormal rows, i.e., $Q_i Q_i^\top = I$, for $i = 1, \dots, 3$, $Q_i Q_j^\top = 0$, $i \neq j$.

This last decomposition represents the key step of the γ-DDPC scheme. Indeed, right-multiplying both sides of (27) by $\mathbf{g}(k)$ yields

$$\begin{bmatrix} \mathbf{z}_{\text{ini}}(k) \\ \mathbf{u}(k) \\ \mathbf{y}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_p \\ \mathbf{U}_f \\ \mathbf{Y}_f \end{bmatrix} \mathbf{g}(k) = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} \gamma_1(k) \\ \gamma_2(k) \\ \gamma_3(k) \end{bmatrix}, \tag{28}$$

allowing one to replace the predictor exploited in DeePC (see (24b)) with a new one in the new variables $\gamma_i(k) = Q_i \mathbf{g}(k)$, for $i = 1, 2, 3$. Note that, unlike DeePC, the dimensions of these new optimization variables do not depend on T anymore, making the control problem computationally tractable even when large data-sets are used. The number of optimization variables can be further reduced by pre-computing $\gamma_1(k)$. Indeed, since $\mathbf{z}_{\text{ini}}(k)$ is fixed, $\gamma_1(k)$ can be set to

$$\gamma_1^*(k) = L_{11}^{\dagger} \mathbf{z}_{\text{ini}}(k). \tag{29}$$

[2] This decomposition can be directly obtained using the MATLAB command `[L,Q] = qr([Up; Yp; Uf; Yf]',0); L=L'; Q=Q';`. For MATLAB versions R2022a and higher, the second argument of `qr.m` can be replaced with `"econ"`.

Table 2
DeePC in a nutshell.

DeePC		
	Without Π regularization	With Π regularization
Problem size	$N(n_u+n_y)+T+T_{\text{ini}}n_y$	$N(n_u+n_y)+T+T_{\text{ini}}n_y$
Memory requirement	$(T_{\text{ini}}+N)(n_u+n_y)\times T$	$(T_{\text{ini}}+N)(n_u+n_y)\times T+T\times T$
Tunable bias–variance trade-off	Yes	Yes
Suitable for large data-sets	No	No
Additional tuning weights	λ_g,λ_y	λ_g,λ_y

Algorithm 3 γ-DDPC control algorithm

Input: $\mathbf{U}_p,\mathbf{Y}_p,\mathbf{U}_f,\mathbf{Y}_f$, $(N,Q,R,r_y,r_u,T_{\text{ini}},\beta_2,\beta_3)$

Before closing the control loop:

1: Compute and extract $L_{11},L_{21},L_{22},L_{31},L_{32},L_{33}$ from LQ-decomposition, see (27).

2: Pre-compute $\begin{bmatrix}L_{21}\\L_{31}\end{bmatrix}L_{11}^{\dagger}$.

2: Build controller as in (30).

3: Tune β_2,β_3.

During the control loop:

1: Measure $y(k)$ and construct $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$ to get $\mathbf{z}_{\text{ini}}(k)$.

2: Solve control law (30) and obtain $\mathbf{u}^*(k)$.

3: Apply $u(k)=u^*(0|k)$ to the system.

Table 3
γ-DDPC in a nutshell.

γ-DDPC	
Problem size	$2N(n_u+n_y)$
Memory requirement	$N(n_u+n_y)\times(N+T_{\text{ini}})(n_u+n_y)$
Tunable bias–variance trade-off	Yes
Suitable for large data-sets	Yes
Additional tuning weights	β_2,β_3

Based on both the LQ-decomposition and the pre-computation of $\gamma_1(k)$, the γ-DDPC problem thus reduces to

$$\underset{\mathbf{u}(k),\mathbf{y}(k),\gamma_2(k),\gamma_3(k)}{\text{minimize}} \quad l_N(y(N|k))+\sum_{i=0}^{N-1} l(y(i|k),u(i|k))+l_\gamma(\gamma_2(k),\gamma_3(k),\beta) \tag{30a}$$

$$\text{subject to} \quad \begin{bmatrix}\mathbf{u}(k)\\\mathbf{y}(k)\end{bmatrix}=\begin{bmatrix}L_{22}&0\\L_{32}&L_{33}\end{bmatrix}\begin{bmatrix}\gamma_2(k)\\\gamma_3(k)\end{bmatrix}+\begin{bmatrix}L_{21}\\L_{31}\end{bmatrix}L_{11}^{\dagger}\mathbf{z}_{\text{ini}}(k), \tag{30b}$$

$$(\mathbf{y}(k),\mathbf{u}(k))\in\mathbb{Y}^N\times\mathbb{U}^N. \tag{30c}$$

where $l_\gamma(\gamma_2(k),\gamma_3(k),\beta)=\beta_2\|\gamma_2(k)\|^2+\beta_3\|\gamma_3(k)\|^2$, with $\beta=\begin{bmatrix}\beta_2&\beta_3\end{bmatrix}^T$, is a regularization term that can be used to trade-off between bias and variance as in DeePC. The reader is referred to Algorithm 3 for a sketch of the main steps to be carried out in γ-DDPC, while its main features are listed in Table 3.

Remark 3.10 (*Linking γ-DDPC and SPC*). By setting $\gamma_3=0$ (a.k.a., $\beta_3\to\infty$) and $\beta_2=0$, γ-DDPC can be related with SPC based on the following relation (Favoreel et al., 1999):

$$\begin{bmatrix}P_1&P_2&\hat{\Gamma}\end{bmatrix}=\begin{bmatrix}L_{31}&L_{32}\end{bmatrix}\begin{bmatrix}L_{11}&0\\L_{21}&L_{22}\end{bmatrix}^{\dagger}. \tag{31}$$

This equality holds also when $L_{33}=0$, which is true in a noise-free setting.

3.4. Generalized data-driven predictive control

The last (hybrid) approach that we review in this survey explicitly merges SPC and DeePC, towards benefiting from the positive aspects of the two methods, i.e., the possibility of leveraging large data-sets and the advantages of using an unbiased predictor as in SPC, and the possibility to tune the bias–variance trade-off guaranteed by DeePC.

To this end, Generalized Data-Driven Predictive Control (GDPC) (Lazar & Verheijen, 2023) parametrizes the input sequence as the sum of two terms, namely a free, *fixed* input sequence $\mathbf{u}_b(k)$, and a forced one, that is actually *optimized* by solving a quadratic program on-line.

The free input is used to construct the corresponding output predictions $\mathbf{y}_b(k)$ according to the SPC rationale, i.e., via (9). Therefore, GDPC initially requires one to use the (potentially large) data-set of length T to estimate the prediction matrices $\begin{bmatrix}P_1&P_2&\hat{\Gamma}\end{bmatrix}$ and to construct $\mathbf{y}_b(k)$ accordingly as

$$\mathbf{y}_b(k)=P_1\mathbf{u}_{\text{ini}}(k)+P_2\mathbf{y}_{\text{ini}}(k)+\hat{\Gamma}\mathbf{u}_b(k) \tag{32}$$

Then, by considering a reduced length $n_u(N+T_{\text{ini}}+n)\le T_g\le T$, one can construct the GDPC Hankel matrices

$$\begin{aligned}\bar{\mathbf{U}}_p&:=\begin{bmatrix}\bar{\mathbf{u}}(0,T_{\text{ini}})&\dots&\bar{\mathbf{u}}(T_g-1,T_{\text{ini}})\end{bmatrix},\\ \bar{\mathbf{Y}}_p&:=\begin{bmatrix}\bar{\mathbf{y}}(1,T_{\text{ini}})&\dots&\bar{\mathbf{y}}(T_g,T_{\text{ini}})\end{bmatrix},\\ \bar{\mathbf{U}}_f&:=\begin{bmatrix}\bar{\mathbf{u}}(T_{\text{ini}},N)&\dots&\bar{\mathbf{u}}(T_{\text{ini}}+T_g-1,N)\end{bmatrix},\\ \bar{\mathbf{Y}}_f&:=\begin{bmatrix}\bar{\mathbf{y}}(T_{\text{ini}}+1,N)&\dots&\bar{\mathbf{y}}(T_{\text{ini}}+T_g,N)\end{bmatrix},\end{aligned} \tag{33}$$

where

$$\begin{aligned}\bar{\mathbf{u}}(k,j)&:=\text{col}(\bar{u}(k),\dots,\bar{u}(k+j-1)),\\ \bar{\mathbf{y}}(k,j)&:=\text{col}(\bar{y}(k),\dots,\bar{y}(k+j-1)),\end{aligned}$$

then formulating the GDPC problem

$$\underset{\mathbf{u}(k),\mathbf{y}(k),\mathbf{g}(K),\sigma_y(k)}{\text{minimize}} \quad l_N(y(N|k))+\sum_{i=0}^{N-1} l(y(i|k),u(i|k))+\lambda_g l_g(\mathbf{g}(k))+\lambda_y\|\sigma_y(k)\| \tag{34a}$$

$$\text{subject to} \quad \begin{bmatrix}\bar{\mathbf{U}}_p\\\bar{\mathbf{Y}}_p\\\bar{\mathbf{U}}_f\\\bar{\mathbf{Y}}_f\end{bmatrix}\mathbf{g}(k)=\begin{bmatrix}0\\\sigma_y(k)\\\mathbf{u}(k)-\mathbf{u}_b(k)\\\mathbf{y}(k)-\mathbf{y}_b(k)\end{bmatrix} \tag{34b}$$

$$(\mathbf{y}(k),\mathbf{u}(k))\in\mathbb{Y}^N\times\mathbb{U}^N, \tag{34c}$$

where $l_g(\mathbf{g}(k))$ can be chosen similarly to the methods used in DeePC. However, since we already steer the estimate towards $\mathbf{u}_b(k)$, $\mathbf{y}_b(k)$, the selection of $l_g(\mathbf{g}(k))=\|\mathbf{g}(k)\|_2^2$ is the most computationally efficient choice. Based on the results of van Waarde, De Persis et al. (2020), we stress that the Hankel matrices $\bar{\mathbf{U}}_p,\bar{\mathbf{U}}_f,\bar{\mathbf{Y}}_p,\bar{\mathbf{Y}}_f$ in (33) do not have to be *necessarily* constructed from $\mathbf{U}_p,\mathbf{Y}_p,\mathbf{U}_f,\mathbf{Y}_f$. Moreover, although $\mathbf{u}_b(k)$ can in principle be freely chosen by the user, the closer it is to the optimal input $\mathbf{u}^*(k)$, the more accurate the prediction of $\mathbf{y}(k)$ is. The approach suggested in Lazar and Verheijen (2023) to comply with this is to select $\mathbf{u}_b(k)$ as the unconstrained solution to the SPC problem, namely

$$\begin{aligned}\mathbf{u}_b^*(k)&=\arg\min_{\mathbf{u}_b(k)}\ (\mathbf{y}_b(k)-\mathbf{r}_y(k))^T\Omega(\mathbf{y}_b-\mathbf{r}_y(k))+(\mathbf{u}_b(k)-\mathbf{r}_u(k))^T\\ &\quad\times\Psi(\mathbf{u}_b(k)-\mathbf{r}_u(k))\\ &=\arg\min_{\mathbf{u}_b(k)}\ \mathbf{u}_b(k)^T\left(\Psi+\hat{\Gamma}^T\Omega\hat{\Gamma}\right)\mathbf{u}_b(k)+2\mathbf{u}_b(k)^T\\ &\quad\times\left(\hat{\Gamma}^T\Omega\left(P_1\mathbf{u}_{\text{ini}}(k)+P_2\mathbf{y}_{\text{ini}}(k)-\mathbf{r}_y(k)\right)-\Psi\mathbf{r}_u(k)\right)\\ &=-\left(\Psi+\hat{\Gamma}^T\Omega\hat{\Gamma}\right)^{-1}\left(\hat{\Gamma}^T\Omega\left(P_1\mathbf{u}_{\text{ini}}(k)+P_2\mathbf{y}_{\text{ini}}(k)-\mathbf{r}_y(k)\right)-\Psi\mathbf{r}_u(k)\right),\end{aligned} \tag{35}$$

Algorithm 4 GDPC control algorithm

Input: $\mathbf{U}_p, \mathbf{Y}_p, \mathbf{U}_f, \mathbf{Y}_f$, $(N, T_{\text{ini}}, Q, R, r_y, r_u, \lambda_g)$

Before closing the control loop:

1: Obtain Θ from (10) and extract P_1, P_2 and $\hat{\Gamma}$.
2: Pick a suitable T_g and construct $\bar{\mathbf{U}}_p, \bar{\mathbf{Y}}_p, \bar{\mathbf{U}}_f, \bar{\mathbf{Y}}_f$, see (33).
3: Build controller as in (34).
4: Compute $(\Psi + \hat{\Gamma}^T \Omega \hat{\Gamma})^{-1}$.
5: Tune λ_g.

During the control loop:

1: Measure $y(k)$ and construct $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$.
2: Compute $\mathbf{u}_b^*(k)$ from (35).
3: Obtain $\mathbf{y}_b(k) = P_1 \mathbf{u}_{\text{ini}}(k) + P_2 \mathbf{y}_{\text{ini}}(k) + \hat{\Gamma} \mathbf{u}_b(k)$.
2: Solve control law (34) and obtain $\mathbf{u}^*(k)$.
3: Apply $u(k) = u^*(0|k)$ to the system.

where $\Omega := \text{diag}\{Q, \dots, Q, \alpha Q\}$, $\Psi := \text{diag}\{R, \dots, R\}$, and $\mathbf{r}_y(k), \mathbf{r}_u(k)$ are the respective output and input set-point vectors. Since $\left(\Psi + \hat{\Gamma}^T \Omega \hat{\Gamma}\right)^{-1}$ can be computed a-priori, the computational load online is negligible.

Remark 3.11 (*About Ill-Conditioned Problems*)**.** In contrast to DeePC, $\mathbf{g}(k)$ must satisfy

$$\begin{bmatrix} \bar{\mathbf{U}}_p \\ \bar{\mathbf{Y}}_p \end{bmatrix} \mathbf{g}(k) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Since $\{0, 0\}$ this is a trajectory of every linear system, $\sigma_y(k)$ should be unnecessary for this problem. However, under noisy conditions, small sizes of T_g (i.e., $T_g \leq N n_u + T_{\text{ini}}(n_u + n_y)$) can turn the problem infeasible since the space of trajectories over which the solver can freely optimize $\mathbf{u}(k)$ over is smaller than the dimension of $\mathbf{u}(k)$. Therefore, if it is required to use a smaller size of T_g, the slack $\sigma_y(k)$ must be used in GDPC. Also, in our experience, including $\sigma_y(k)$ improves numerical stability.

The main steps to be performed when using GDPC in combination with (35) are finally summarized in Algorithm 4, while its main features are delineated in Table 4.

4. Guidelines in design, implementation and tuning

Now that the algorithmic background on the surveyed data-driven control techniques has been reviewed, it is important to provide practitioners with a set of guidelines on how to actually make these data-driven controllers work. In this section, we then introduce a set of recommendations on how to use them and make some considerations that can help improve your data-driven predictive controller.

4.1. Generating experiment data

Clearly, at the core of any data-driven control strategy lies the availability of an "informative" set of input/output data, that have to be gathered from the controlled plant. The design of the input used in this data collection phase thus becomes crucial for the whole data-driven design to succeed. In particular, these inputs must be persistently exciting with an order of $T_{\text{ini}} + N + n$ (see Lemma 3.4) or higher.

When experiments can be performed on the plant in open-loop, input design can be carried out as it is commonly done in system identification. Therefore, conventional choices for the inputs are Gaussian white noises, Pseudo-Random Binary Signals (PRBS) and multi-sines (Ljung, 1999). Although the asymptotic convergence of the predictor's bias is guaranteed in open-loop for any persistently exciting input, the input magnitude should strongly excite the system with respect to the expected noise levels as it is practically impossible to work with infinite size data-sets. We refer the reader to the results of Coulson et al. (2023) for exact lower bounds on the input magnitude.

Meanwhile, when no guarantees are available on the stability of the system to be controlled, the data collection experiments should be conducted in closed-loop, while still guaranteeing the required level of excitation. In this case, one can consider the "direct method", in which the reference used to carry out the closed-loop experiments is strongly persistently exciting (Van den Hof & Schrama, 1995). Alternatively, one can add a persistently exciting input disturbance to the nominal closed-loop input.

Remark 4.1 (*The Limited Energy Case*)**.** When only limited energy is transferred to the system, one can select inputs that have increased spectral energy for specific frequency bands. While potentially improving prediction accuracy at these frequencies, the choice of these inputs sacrifices accuracy outside such bands. Therefore, it is a viable experiment design option only when one seeks optimal performance within a specific frequency band, while in all other cases, it is advisable to excite all frequencies.

4.2. Design of the data matrices

The data-driven predictors featured in all the reviewed direct control strategies rely on the construction of a set of Hankel matrices built from a single experiment. Nonetheless, these key elements of the data-driven control problems can also be constructed in other ways.

Chinese page matrices

Instead of using Hankel matrices to store the data, they can also be structured into a (Chinese) Page matrix, i.e.,

$$\mathcal{P}_N(z) := \begin{bmatrix} z(0) & z(N) & \dots & z(L) \\ z(1) & z(N+1) & \dots & z(L+1) \\ \vdots & \vdots & & \vdots \\ z(N-1) & z(2N-1) & \dots & z(L+N-1) \end{bmatrix}.$$

Based on this alternative choice, every measured sample only appears once in the data matrices, in turn resulting in favorable stochastic properties (van den Hof, 1983). Moreover, the use of this alternative structure enables one to consider a larger set of behaviors, while still allowing to contain the dimension of the control problem. Given the issues linked to its memory requirements, this is particularly desirable in DeePC (see Table 2). For these reasons, DeePC has indeed been tested with both Hankel and Page matrices of the same size, evidencing improved closed-loop performance when Page matrices have been used Coulson (2022, Section 3.2.1).

Multiple data-sets

Following the result of van Waarde, De Persis et al. (2020), the Hankel matrices do not necessarily have to be constructed with data gathered from one single experiment. Indeed, multiple Hankel matrices generated with diverse data-sets can be concatenated (i.e., $\mathbf{U}_p = \begin{bmatrix} \mathbf{U}_p^1 & \mathbf{U}_p^2 & \dots & \mathbf{U}_p^m \end{bmatrix}$), provided that each sub-Hankel matrix $\mathbf{U}_p^i$ is constructed from one data-set only and that the concatenation still satisfies the minimum conditions, i.e., persistency of excitation and the minimum length of T.

4.3. A note on the implementation of DPC on the actual system: the initial window

All discussed data-driven predictive controllers estimate the free response using a past window of input–output data, $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$. When the controller is implemented on the system, this data has not yet been measured. Therefore, between $t = 0$ and $t = T_{\text{ini}} T_s$, it is advised to let the system run in an open loop sequence (this includes using $u = 0$) or using any previously applied control laws, and only activate the data-driven controller once $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$ are collected. Note that, for any run time less than $t = T_{\text{ini}} T_s$, no converging statement can

Table 4
GDPC in a nutshell.

GDPC	
Problem size	$N(n_u+n_y)+T_g+T_{\text{ini}}n_y$
Memory requirement	$(T_{\text{ini}}+N)(n_u+n_y)\times T_g + Nn_u \times Nn_u + Nn_y \times (T_{\text{ini}}(n_u+n_y)+Nn_u)$
Tunable bias–variance trade-off	Yes
Suitable for large data-sets	Yes
Additional tuning weights	$\lambda_g, T_g, \lambda_y$

be made about the estimate of the free response, regardless of the noise variance.

4.4. Selection of T_{ini}

Let us consider the following relationship (Knudsen, 2001):

$$x(k) = (A-KC)^{T_{\text{ini}}}x(k-T_{\text{ini}}) + \sum_{i=0}^{T_{\text{ini}}-1}\left((A-KC)^i(Bu(k-i-1) + Ky(k-i-1))\right). \tag{36}$$

Since $|\text{eig}(A-KC)|<1$, then

$$\lim_{T_{\text{ini}}\to\infty} x(k) = \sum_{i=0}^{T_{\text{ini}}-1}\left((A-KC)^i(Bu(k-i-1)+Ky(k-i-1))\right), \tag{37}$$

implying that the state of the system can be reconstructed exactly by simply looking at a (long) past horizon of inputs and (noisy) outputs. This relation shows that the choice of T_{ini} entails an engineering trade-off. Indeed, one would aim for large T_{ini} in order to better reconstruct the system's state. On the other hand, increasing the past window means that the initial time in which the controller has to run in open-loop increases, along with the complexity of the data-driven procedures.

Possible automated strategies to tune T_{ini} while attaining the previous trade-off can be taken from system identification. As an example, one can identify ARX models of increasing order with the available data and, then, select the order leading to the model with minimum Akaike's Information Criterion (AIC) (Akaike, 1976). It is worth remarking that the accuracy of these models is not relevant for the sake of control design since they are solely used to select T_{ini}. At the same time, for reduced levels of noise, this strategy will lead to values of T_{ini} that are close to the true system order n, generally resulting in $T_{\text{ini}} \gg n$ when the data become more noisy.

Remark 4.2 (T_{ini} *and* n)**.** The vector $\begin{bmatrix}\mathbf{u}_{\text{ini}}^T & \mathbf{y}_{\text{ini}}^T\end{bmatrix}^T$ can be seen as a non-minimal state realization for (2). Under this lens, in a deterministic setting, the controllability of the corresponding non-minimal system is guaranteed only when $T_{\text{ini}} = n$. Nonetheless, this does not hold anymore when the data are noisy.

4.5. Offset-free tracking by augmenting the controller with integral action

In classical MPC, offset-free tracking can be achieved by either (*i*) computing an input reference $\mathbf{r}_u$ that corresponds to the desired output set point $\mathbf{r}_y$, or (*ii*) augmenting the system in an integral action. However, these approaches are both not straightforwardly implementable in a data-driven setting, due to the lack of an explicit model for the controlled system. Nonetheless, offset-free tracking can be still attained in data-driven control through the result of Verheijen et al. (2021, Thrm 1), which allows one to construct a rate-based, data-driven controller following the steps summarized in Algorithm 5. Note that this result is rooted in the augmentation of the system's model with an integrator, leading to its order increasing of n_y. Therefore, for the data-driven control to be well-posed, $\bar{u}$ must be persistently exciting with an order of $N + T_{\text{ini}} + n + n_y$.

It is worth remarking that the rate-based strategy summarized in Algorithm 5 does not require the specification of any input reference,

Algorithm 5 Embed Integral action into any data-driven predictive controller

Input: $\bar{u}, \bar{y}$

Before closing the control loop:

1: Construct $\Delta\bar{u}$ from $\bar{u}$ as $\Delta\bar{u}(i) = \bar{u}(i) - \bar{u}(i-1)$.
2: Build Hankel matrices $\mathbf{Y}_p$, $\mathbf{Y}_f$ from $\bar{y}$ and $\Delta\mathbf{U}_p$, $\Delta\mathbf{U}_f$ from $\Delta\bar{u}$.
3: Perform any steps of your preferred DPC rule using $\Delta\mathbf{U}_p$ and $\Delta\mathbf{U}_f$, while considering a rate-based control law.

During the control loop:

1: Measure $y(k)$ and construct $\Delta\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$.
2: Solve the control law and obtain $\Delta\mathbf{u}^*(k)$.
3: Apply $u(k) = u(k-1) + \Delta u^*(0|k)$ to the system.

while the approach inherits the capabilities of rejecting constant disturbances typical of MPC schemes with integral action. At the same time, it is important to highlight that noise has an increased influence at low frequencies, as predictions are based on the linear combination of the outputs and the differences of consecutive inputs. As a possible countermeasure to reduce this effect, one could select an input used to collect data that has higher energy at lower frequencies. Alternatively, one could also think of designing $\Delta\bar{u}$ first, then performing the experiments on the system with $\bar{u}(i) = \bar{u}(i-1)+\Delta\bar{u}(i)$. Nonetheless, even though this choice resolves the aforementioned issue, it also implies that one has to feed an integrating input to the open-loop system, which can be undesirable.

4.6. Tuning guidelines

While it can be assumed that there is some engineering experience in choosing the right Q and R (as it is also required in tuning standard MPC), DeePC, γ-DDPC and GDPC all require the choice of regularization penalties (see Tables 2–4). However, these hyper-parameters are often cumbersome to tune in practice. Indeed, when no simulations can be performed on a black-box model of the system, tuning requires closed-loop calibration experiments that can be unsafe (e.g., leading to plant destabilization). Therefore, we now provide a set of analytical tools that can be used by the user to pick these tuning variables, without requiring additional closed-loop experiments.

4.6.1. Tuning guidelines for DeePC

As shown in Lazar and Verheijen (2022), the DeePC cost function can be rewritten into a Tikhonov regularization problem as follows:

$$\|\mathcal{A}\mathbf{g}(k)-\mathbf{b}\|_2^2 + \lambda_g\|\mathbf{g}(k)\|_2^2, \quad \text{where } \mathcal{A} := \text{diag}(\Omega^{\frac{1}{2}}, \Psi^{\frac{1}{2}})\begin{bmatrix}\mathbf{Y}_f\\ \mathbf{U}_f\end{bmatrix}, \quad \mathbf{b} := \text{diag}(\Omega^{\frac{1}{2}}, \Psi^{\frac{1}{2}})\begin{bmatrix}\mathbf{r}_y\\ \mathbf{r}_u\end{bmatrix}, \tag{38}$$

with $\Omega = \text{diag}(Q,\dots,Q,\alpha Q)$ and $\Psi = \text{diag}(R,R,\dots,R)$ following the cost function defined in (5). This enables practitioners to exploit existing tuning strategies for the regularization penalty λ_g for this class of problems, eventually avoiding lengthy and (potentially) unsafe trial-and-error (closed-loop) procedures. In Lazar and Verheijen (2022), λ_g is tuned through the Hanke–Raus rule (Hanke & Raus, 1996), i.e., by considering the following score function of λ_g:

$$\psi(\lambda_g) := \langle \mathbf{b}, \lambda_g^2(\mathcal{A}\mathcal{A}^T + \lambda_g I)^{-3}\mathbf{b}\rangle^{\frac{1}{2}}, \tag{39}$$

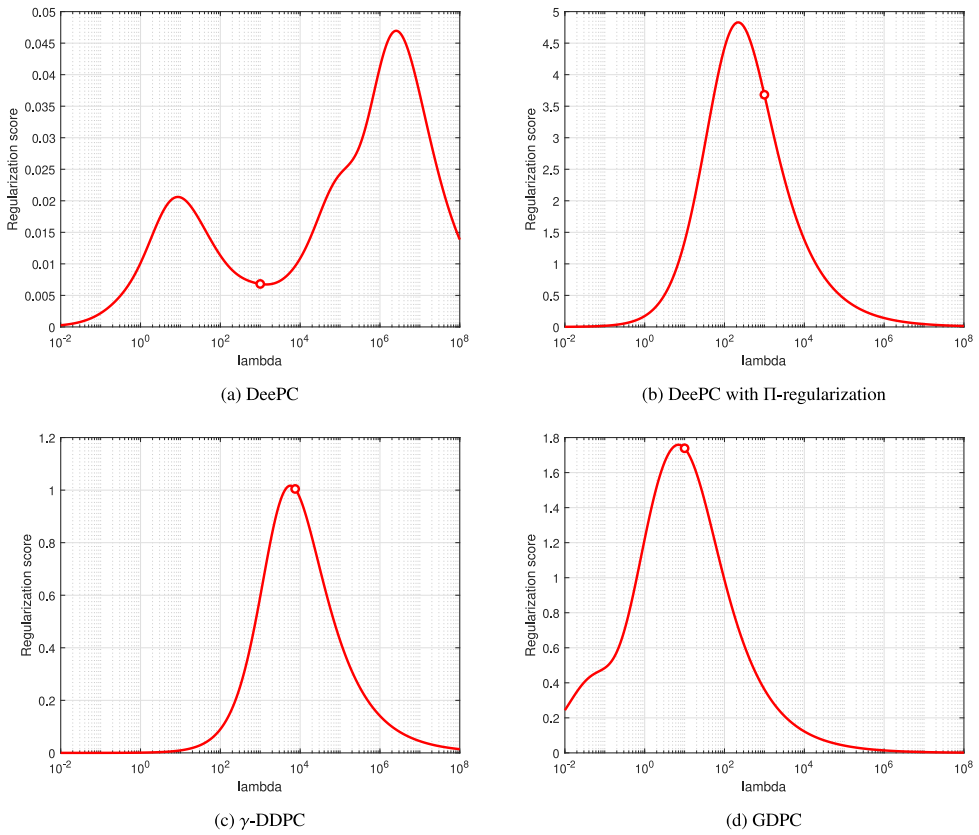

Fig. 1. Score function $\psi(\lambda_g)$ plotted against λ_g for the example in Section 5.1. The chosen values of λ_g are the circle displayed on top of the score function, with the one chosen for DeePC corresponding to the value of λ_g selected for DeePC with Π-regularization.

and using its plot on a semi-logarithmic scale with respect to the regularization penalty to select the latter. It is worth remarking that the aforementioned tuning rationale can be employed when using the Π-regularization (see Remark 3.7) simply by imposing (Lazar & Verheijen, 2022):

$$\mathcal{A} = \mathcal{A}(I - \Pi). \tag{40}$$

Next, we discuss how to interpret this plot in a practical case to select λ_g. Let us specifically consider Fig. 1, which illustrates the Hanke–Raus plots for all the reviewed DPC approaches for the case study we will analyze in Section 5.1, and let us focus on Figs. 1(a)–1(b). From Fig. 1(b), it is evident that $\psi(\lambda_g)$ tends to zero when λ_g is approaching the two regularization extremes, i.e., $\lambda_g = 0$ and $\lambda_g \to \infty$. Indeed, for $\lambda_g = 0$, the solution of (38) leads to over-fit the available noisy data, while it tends to neglect the control objective for $\lambda_g \to \infty$, leading to undesired results in both cases (as summarized in Table 5). Instead, the peak between these extremes displayed in Fig. 1(b) represents a *transitioning regime*, where both objectives are close to being equally treated and, thus, values λ_g proximal to those characterizing these peaks are preferable. At the same time, it is generally desirable to select regularization penalties that favor $\mathbf{g}(k)$ such that the system dynamics are respected, making higher values of λ_g preferable. From experimental experience, it appears that choosing values just after the peak shows to have satisfactory performance. This can be a useful guideline in selecting an initial value for λ_g, which can then (if possible) be refined through experiments on the system until the desired performance is achieved. While a similar tuning rationale can be employed with γ-DDPC and GDPC, since their Hanke–Raus plots are more relatable to those of DeePC with Π regularization, the same cannot be said for standard DeePC. Indeed, as clear from Fig. 1(a), the Hanke–Raus plot of standard DeePC presents multiple peaks, that further shift and change in number under different noise conditions. Our experiments nonetheless indicate that using the Π-regularized (Fig. 1(b)) Hanke–Raus plot to obtain a suitable value of λ_g for DeePC can be a viable practical solution to circumvent this issue.

Remark 4.3 (*Limits in the Applicability of the Hanke–Raus Rule*)**.** If both the input and output references are set to zero, then the Hanke–Raus rule becomes undetermined, since $\mathbf{g}(k) = 0$ would minimize the cost-function, regardless of the weight in λ_g (note that in its current form, fitting $\mathbf{g}(k)$ to the initial trajectory is not considered in the problem). Therefore, this approach is viable as long as $\mathbf{r}_y, \mathbf{r}_u$ in $\mathbf{b}$ are not both zero.

4.6.2. *Tuning guidelines for γ-DDPC*

Differently from DeePC, γ-DDPC requires the user to tune two parameters, β_2 and β_3 (see (30)). While these parameters can be ideally set to $\beta_2 = 0$ and $\beta_3 \to \infty$ when an infinite data-set is available (Breschi, Chiuso and Formentin, 2023), their choice might become critical when a limited number of data is available. Since the data at hand to design the controller are finite in practice, we now summarize possible tuning strategies for β_2 and β_3 to cope with this practical limitation.[3]

[3] We now drop the dependence on k, as the tuning rationale we review can be exploited for both off-line and on-line calibration of β_2 and β_3. Mind that the dependence on k should be reintroduced, whenever on-line tuning is considered.

Table 5
Tuning extremes in DeePC.

$\lambda_g \to 0$	$\lambda_g \to \infty$	$\lambda_g(I-\Pi) \to 0$	$\lambda_g(I-\Pi) \to \infty$
$\mathbf{u}^*(k) = \mathbf{r}_u$	$\mathbf{u}^*(k) = 0$	$\mathbf{u}^*(k) = \mathbf{r}_u$	$\mathbf{u}^*(k) = \mathbf{u}^*_{\text{SPC}}(k)$

Tuning β_2. Let us consider $\beta_3 \to \infty$ and, accordingly, $\gamma_3 = 0$. In this case, the only hyper-parameter left to be tuned in (30) is the penalty on the 2-norm of γ_2, namely β_2.

As shown in Breschi, Chiuso, Fabris et al. (2023), the role of β_2 depends on the features of the input data $\bar{\mathbf{u}}$ used to construct the output predictor. Indeed, when such a sequence is white, the 2-norm regularization on γ_2 in (30) is asymptotically (i.e., for $T \to \infty$) equivalent to a penalization of the input. In this scenario, β_2 (up to a scaling factor) thus represents an additional penalty on the predicted input aside from the weight already characterizing the stage cost. Therefore, choosing $\beta_2 = 0$ could be a viable option not to change the importance ratio between the two design objectives (with respect to the one dictated by Q and R in (5)). Instead, the equivalence between regularization and input penalization does not hold anymore when $\bar{\mathbf{u}}$ is not white, making $\beta_2 = 0$ not an advisable choice when the data are finite, due to the link between γ_2 and the output predictor variance discussed in Breschi, Fabris et al. (2023).

With finite data, one can generally pick β_2 to *contain* the variance of the output predictor, in turn allowing the designer to avoid undesired effects due to prediction errors. In particular, it is desirable for the control sequence

$$\mathbf{u} = L_{21}L_{11}^{\dagger}\mathbf{z}_{\text{ini}} + L_{22}\gamma_2, \tag{41}$$

not to (wrongly) leverage prediction errors to make the predicted tracking error small, *i.e.,*

$$\varepsilon = \mathbf{y} - \mathbf{r}_y = L_{31}L_{11}^{\dagger}\mathbf{z}_{\text{ini}} + L_{32}\gamma_2 - \mathbf{r}_y \simeq 0. \tag{42}$$

To this end, β_2 can be selected either off-line or on-line for the following relationship to hold as tightly as possible:

$$\|L_{33}^{-1}\varepsilon\|_2^2 \simeq \frac{N}{T}\left(\|\gamma_1^*\|_2^2 + \|\gamma_2^*(\beta_2)\|_2^2\right). \tag{43}$$

where T denotes the number of columns of the Hankel matrices.

Alternatively, one can exploit the Hanke–Raus rule already introduced for the DeePC scheme in Section 4.6.1 to tune β_2, by imposing:

$$\begin{aligned}&\psi(\beta_2) := \langle \mathbf{b}(\gamma_1^*), \beta_2^2(\mathcal{A}\mathcal{A}^T + \beta_2 I)^{-3}\mathbf{b}(\gamma_1^*)\rangle^{\frac{1}{2}}, \quad \text{where}\\ &\quad \mathcal{A} := \operatorname{diag}(\Omega^{\frac{1}{2}}, \Psi^{\frac{1}{2}})\begin{bmatrix} L_{32} \\ L_{22}\end{bmatrix}, \quad \mathbf{b}(\gamma_1^*) := \operatorname{diag}(\Omega^{\frac{1}{2}}, \Psi^{\frac{1}{2}})\begin{bmatrix} \mathbf{r}_y - L_{31}\gamma_1^* \\ \mathbf{r}_u - L_{21}\gamma_1^*\end{bmatrix},\end{aligned} \tag{44}$$

with γ_1^* defined as in (29). Due to the dependence on γ_1^*, the choice of β_2 returned by the Hanke–Raus heuristics thus changes based on the initial condition $\mathbf{z}_{\text{ini}}$, in line with the tuning rationale in (43).

Tuning β_3. As discussed in Breschi, Fabris et al. (2023), γ_3 can be seen as a slack accounting for the effect that future noise realizations on the quality of the predicted output. Therefore, β_3 can be used to prevent that the input sequence $\mathbf{u}(k)$ is designed by over-fitting noise. Specifically, by following the same rationale previously exploited to obtain (43) and imposing $\beta_2 = 0$, β_3 can be chosen to make the relation

$$\|\gamma_3^*(\beta_3)\|_2^2 \simeq \frac{N}{T}\left(\|\gamma_1^*\|_2^2 + \|\gamma_2^*(\beta_3)\|_2^2\right), \tag{45}$$

hold as closely as possible. Alternatively, also in this case one can employ the Hanke–Raus rule by setting $\mathcal{A}$ and $\mathbf{b}$ as follows:

$$\begin{aligned}&\psi(\beta_3) := \langle \mathbf{b}(\gamma_1^*, \gamma_2), \beta_3^2(\mathcal{A}\mathcal{A}^T + \beta_3 I)^{-3}\mathbf{b}(\gamma_1^*, \gamma_2)\rangle^{\frac{1}{2}}, \quad \text{where}\\ &\quad \mathcal{A} := \Omega^{\frac{1}{2}}L_{33}, \quad \mathbf{b}(\gamma_1^*, \gamma_2) := \Omega^{\frac{1}{2}}\left(\mathbf{r}_y - L_{31}\gamma_1^* - L_{32}\gamma_2\right).\end{aligned} \tag{46}$$

Table 6
Tuning extremes in γ-DDPC for finite datasets.

	$\beta_2 \to 0$	$\beta_2 \to \infty$
$\beta_3 \to 0$	$\mathbf{u}^*(k) = \mathbf{r}_u$	$\mathbf{u}^*(k) = 0$
$\beta_3 \to \infty$	$\mathbf{u}^*(k) = \mathbf{u}^*_{\text{SPC}}(k)$	$\mathbf{u}^*(k) = 0$

Table 7
Tuning extremes in GDPC.

$\lambda_g \to 0$	$\lambda_g \to \infty$
$\mathbf{u}^*(k) = \mathbf{r}_u$	$\mathbf{u}(k) = \mathbf{u}_b(k), \quad \mathbf{y}(k) = \mathbf{y}_b(k)$ whilst respecting constraints

Note that, $\mathbf{b}$ depends on γ_1^*, i.e., on $\mathbf{z}_{\text{ini}}$, and γ_2. Therefore, the Hanke–Raus rule would return different values of β_3 depending on the initial conditions and the optimization variable γ_2, thus being aligned with the tuning rationale in (45). It is further worth remarking that the choice β_3 has similar effects to that of λ_g in the DeePC with Π-regularization. Indeed, for increasing values of β_3, the predictor exploited in γ-DDPC with $\beta_2 = 0$ approaches the one used in SPC (see (18)). The same happens when $\lambda_g \to \infty$ in (24) + (25) (see Table 6).

4.6.3. Tuning guidelines for GDPC

Designing a generalized data-driven predictive controller requires the user to select two hyper-parameters, namely online Hankel matrix size T_g and the regularization parameter λ_g, that are used to control the bias/variance trade-off. Even though a systematic approach is not yet available to tune them in the GDPC framework (see Lazar and Verheijen (2023)), we can still provide some intuitive guidelines on the selection of T_g and λ_g.

The matrix size T_g should satisfy the requirements for the data-driven predictor to be meaningful, while it should be kept small enough for the control action to be computed within the system's sampling period.

Meanwhile, λ_g should be tuned by considering the (undesired) effects that the choice of one of its two extreme values has on the final solution, which are summarized in Table 7. For this, consider the same approach as for γ–DDPC and DeePC, using the Tikhonov regularization, with the following modification:

$$\begin{aligned}&\mathbf{y}_b = \hat{\Gamma}\mathbf{u}_b, \quad \mathbf{u}_b = (\Psi + \hat{\Gamma}^T\Omega\hat{\Gamma})^{-1}(\hat{\Gamma}^T\Omega\mathbf{r}_y + \Psi\mathbf{r}_u)\\ &\mathcal{A} := \operatorname{diag}(\Omega^{\frac{1}{2}}, \Psi^{\frac{1}{2}})\begin{bmatrix}\bar{\mathbf{Y}}_f \\ \bar{\mathbf{U}}_f\end{bmatrix}, \quad \mathbf{b} := \operatorname{diag}(\Omega^{\frac{1}{2}}, \Psi^{\frac{1}{2}})\begin{bmatrix}\mathbf{r}_y - \mathbf{y}_b \\ \mathbf{r}_u - \mathbf{u}_b\end{bmatrix},\end{aligned} \tag{47}$$

where the result hereof can be seen in Fig. 1(d).

4.7. Stability criteria for DPC in the deterministic, noise-free case

As with any control law, guaranteeing stability is a crucial aspect also in data-driven predictive control. However, with the lack of a proper model of the system, many classical stability guarantees are not trivial for DPC. Nonetheless, a few methods to guarantee closed-loop stability exist, which we will discuss here.

Long prediction horizon

Due to the guarantee of optimality of the MPC control law, the relation with an LQR controller bearing the same stage cost function, and the equivalence relation between DPC algorithms and MPC ones in the noise-free case, see, e.g., Lazar and Verheijen (2022), we have that DPC controllers will be stabilizing if the prediction horizon N is chosen sufficiently large. The value of the prediction horizon is linked to the approximation of the infinite horizon cost by the finite horizon cost, which is typically related to the settling time for open-loop stable systems. Exact lower bounds for N for model-based predictive control can be found, for example, in Boccia et al. (2014).

Trajectory based terminal criteria

To provide a representation-independent stability analysis for various DPC algorithms, we adopt a common predictor description based on truncated behavior (Coulson et al., 2019) $\mathcal{B}_N(\mathcal{P}|\mathbf{u}_{\text{ini}}, \mathbf{y}_{\text{ini}})$ that contains all input and output sequences of length N that are compatible with initial conditions $(\mathbf{u}_{\text{ini}}, \mathbf{y}_{\text{ini}})$ and the underlying system representation (behavior). The symbol $\mathcal{P}$ refers to the set of elements specific to the system representation, e.g., for SPC $\mathcal{P} = \{(P_1, P_2, \Gamma)\}$ and for DeePC $\mathcal{P} = \{(\mathbf{U}_p, \mathbf{Y}_p, \mathbf{U}_f, \mathbf{Y}_f), \mathbf{g}\}$.

From the results of Berberich et al. (2021b) (which extends the work in Berberich et al. (2020)), stability can be enforced by pushing the last n inputs and outputs towards a stabilizing terminal trajectory. Note that, this is a relaxation over setting the last n inputs and outputs equal to the reference, as terminal equality constraints are more likely to generate infeasible problems. Consider the following modified cost function:

$$\underset{\mathbf{u}(k),\mathbf{y}(k),\xi_n(k)}{\text{minimize}} \quad \sum_{i=0}^{N-1} l(y(i|k), u(i|k)) + l_T(\boldsymbol{\xi}_n(k)) \tag{48a}$$

$$\text{subject to} \quad (\mathbf{y}(k), \mathbf{u}(k)) \in \mathcal{B}(\mathcal{P}|\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k)) \tag{48b}$$

$$(\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N, \tag{48c}$$

$$\boldsymbol{\xi}_n(k) \in \Xi_n, \boldsymbol{\xi}_n(k) = \begin{bmatrix} \text{col}\,(u(N-n-1|k), \dots, u(N-n|k)) \\ \text{col}\,(y(N-n|k), \dots, y(N|k)) \end{bmatrix}, \tag{48d}$$

where $(\mathbf{y}(k), \mathbf{u}(k)) \in \mathcal{B}(\mathcal{P}|\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k))$ denotes either (14b), (24b), (30b) or (34b). Let $l_T(\boldsymbol{\xi}_n(k)) = \boldsymbol{\xi}_n(k)^T P \boldsymbol{\xi}_n(k)$ for some $P \succ 0$ satisfy a Lyapunov condition, with the set Ξ_n be an invariant and constraints admissible set. To design a P, consider the following state-space realization (Berberich et al., 2021c):

$$\begin{aligned} \boldsymbol{\xi}_n(k+1) &= \left[\begin{array}{cccc|cccc} 0 & I & & & 0 & \cdots & \cdots & 0 \\ \vdots & & \ddots & & \vdots & & & \vdots \\ 0 & & & I & 0 & & & 0 \\ 0 & \cdots & \cdots & 0 & 0 & \cdots & \cdots & 0 \\ \hline 0 & \cdots & \cdots & 0 & 0 & I & & \\ \vdots & & & \vdots & \vdots & & \ddots & \\ 0 & \cdots & \cdots & 0 & 0 & & & I \\ b_n & \cdots & \cdots & b_1 & a_n & \cdots & \cdots & a_1 \end{array}\right] \boldsymbol{\xi}_n(k) + \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ I \\ \hline 0 \\ \vdots \\ 0 \\ b_0 \end{array}\right] u(k) \\ &= \tilde{A}\boldsymbol{\xi}_n(k) + \tilde{B}u(k). \end{aligned} \tag{49}$$

Then it can be shown as in Proposition 10 of Berberich et al. (2021c), that a matrix P and gain $\tilde{K}$ that satisfy the condition

$$(\tilde{A} + \tilde{B}\tilde{K})^T P (\tilde{A} + \tilde{B}\tilde{K}) - P \preceq -\begin{bmatrix} 0 & \\ & \Omega \end{bmatrix} - \tilde{K}^T \Psi \tilde{K} \tag{50}$$

yield a closed-stable data-driven predictive controller. Therein, it is also shown how P and $\tilde{K}$ can be computed using only input–output data and solving a linear matrix inequality. Alternatively, the method for computing stabilizing local controllers from input–output data is presented in De Persis and Tesi (2020). A suboptimal but more practical approach is to design an arbitrary stabilizing $\tilde{K}$ such that $|\text{eig}(\tilde{A} + \tilde{B}\tilde{K})| \leq 1$ and then solve (50) which becomes a linear matrix inequality P. Then, the terminal set can be computed as a polyhedral invariant set for $\tilde{A} + \tilde{B}\tilde{K}$ or the largest admissible level set $\{\xi_n : \xi_n^T P \xi_n \leq c\}$ for some $c > 0$ can be used as a terminal set. However, the latter choice yields a convex quadratically constrained QP problem instead of a QP that must be solved online.

This method has also the advantage that it guarantees recursive feasibility of the corresponding DPC optimization problem. In general, to guarantee feasibility, especially in the presence of noisy data, soft constraints should be used where appropriate, i.e., for output constraints and terminal constraints.

Table 8
Flight control: parameters of the running cost and length of the initial condition.

N	T_{ini}	R	Q	$\mathbf{r}_u$
20	20	$0.01 I_2$	$10 I_2$	0

Dissipativity based stability guarantee

Following the results of Lazar (2021) and Lazar and Verheijen (2023) dissipativity of the closed-loop response is implicitly guaranteed for any model-based or data-driven predictive control closed-loop system with the optimal cost as the storage function and a supply function generated by the stage cost. Then, asymptotic stability can be enforced by defining a constant $0 < \rho < 1$ and adding the following constraint to the predictive control optimization problem:

$$l(y(N|k), u(N-1|k)) \leq (1-\rho) l(y(k - T_{\text{ini}}), u(k - T_{\text{ini}})). \tag{51}$$

The above condition yields that the DPC value function is a Lyapunov function, without utilizing a special terminal cost, as in the method above, and without requiring knowledge of a system model or of a locally stabilizing control law and corresponding Lyapunov function. Due to the quadratic cost defined in $l(y, u)$, constraint (51) turns the problem into a convex quadratically constrained QP.

4.8. Beyond linear data-driven predictive control

Although the scope of this handbook is to review DPC techniques for LTI plants, most systems are to some extent not linear and/or time invariant. In this subsection, we discuss a few methods that compensate for some nonlinear or time-varying dynamics, while retaining the linear control law.

Stochastic and robust approaches to DPC. It can be of interest to guarantee that the optimized control action satisfies constraints even if the predicted response is uncertain. Some stochastic implementations to consider are chance-constrained DPC (Coulson et al., 2021), stochastic DPC (Pan et al., 2023) or tube-based stochastic DPC (Kerz et al., 2023).

Adaptive data-driven predictive control. In classical adaptive MPC, the prediction model is updated every time instance to match the system dynamics. Given that one way to update the prediction model is using recent input–output data (Bitmead et al., 1990), the extension towards data-driven predictors is not surprising. Adaptive DPC can be for instance realized using recursive SPC (Verheijen et al., 2022), updating Hankel matrices online in DeePC (Berberich et al., 2021a) or updating the LQ matrices using Givens rotation matrix (Mardi & Wang, 2009) (to give a brief overview). By defining the predictor based on recent data, the resulting prediction is a linear estimate over the recent operating point. However, as the input data is now correlated with the output noise, convergence of the prediction bias to zero is not guaranteed. Consider the aforementioned references on how to tackle this issue.

Linear parameter varying and feedback linearization. Alternative approaches to the above methods include using a linear-parameter-varying (LPV) and feedback linearization description. In the LPV approach to data-driven predictive control (Verhoek et al., 2021), a nonlinear system is embedded into an LPV representation using a parameter that substitutes nonlinear terms in the system representation. This results in LPV Hankel matrices within a DeePC algorithm. Nonlinear systems can also be transformed into a linear representation using feedback linearization, which linearizes the system based on a nonlinear transformation mapping. A data-driven implementation of this technique is considered in Alsalti et al. (2023). The research in this area is very active and it is expected to generate more approaches in the near future, see also Markovsky et al. (2023) for a more general overview.

Table 9
Flight control: regularization parameters and data-set lengths.

	T	T_g	λ_g	λ_y	β_2	β_3
SPC	2500	–	–	–	–	–
DeePC	250	–	1000	10^8	–	–
γ-DDPC	2500	–	–	–	0	7500
GDPC	2500	125	10	–	–	–

5. Numerical examples

We now consider two benchmark examples to compare the performance of the reviewed data-driven predictive control schemes, which we quantitatively compare via the following criteria:

$$\text{ISE} = \sum_{k=T_{\text{ini}}}^{t_{\max}} \|y(k) - r(k)\|_2^2, \quad \text{IAE} = \sum_{k=T_{\text{ini}}}^{t_{\max}} \|y(k) - r(k)\|_1,$$
$$\text{Input Energy} = \sum_{k=T_{\text{ini}}}^{t_{\max}} \|u(k)\|_2^2, \tag{52}$$

respectively evaluating the closed-loop tracking performance achieved with each control scheme and the input effort required to attain them over a closed-loop simulation of t_{max} steps. Performance is assessed in both examples by performing 30 Monte Carlo runs, each using a different noise realization to generate the data set and the corresponding closed-loop simulation, to assess the impact of different noise realizations on the closed-loop behavior of the two benchmark systems. Meanwhile, we empirically evaluate the computational complexity of all the reviewed predictive control schemes by looking at the mean CPU time required to compute the control action over each simulation step, thus analyzing their suitability for embedded applications.[4] In both the considered case studies, the quadratic programs (QPs) in (14), (24), (30) and (34) have been solved at each simulation step via `OSQP`. Moreover, whenever regularization is employed, the Hanke–Raus rule is used to calibrate the regularization parameters off-line. Note that, in γ-DDPC, β_2 is always set to zero and the bias/variance trade-off for the employed predictor is controlled by calibrating β_3. The reader is referred to Section 4 and, specifically, to Fig. 1 for additional details on their choice. We wish to stress here that for the analysis performed in both case studies to be comparable, we only consider DeePC with the "standard" 2-norm regularization, rather than Π-regularization. Indeed, in the first benchmark example, we observed the time-to-solve for DeePC with Π-regularization exceeded the system's sampling time.

To compare the considered DPC approaches against the standard model-based strategy, for each example we further design an MPC scheme by identifying a state-space model for the controlled system using N4SID (van Overschee & De Moor, 1996), assuming the exact knowledge of the system's order. Since we assume not to have direct access to the system's state, in our comparisons we consider two alternative state estimators. Specifically, we employ both a Luenberger observer[5] and a Kalman filter. The latter is designed by setting the covariance of the measurement noise R_{kal} to its estimate provided by N4SID, the covariance of the process noise Q_{kal} as $Q_{\text{kal}} = \hat{K} R_{\text{kal}} \hat{K}^T$, where $\hat{K}$ is the disturbance model obtained with N4SID, while arbitrarily initializing the state covariance as $P_{\text{kal}} = 10^3 I_n$. Note that, these two estimators have been chosen to mimic what a practitioner would do in a first attempt to estimate the system's state, since these approaches are easy to deploy, they follow established rules-of-thumb and require low engineering skills to attain reasonable performance.

Remark 5.1 (*On the Aim of the Numerical Analysis*)**.** We wish to stress that our goal with the subsequent numerical analysis is not to draw any conclusion on which strategy has to be preferred over the others (i.e., we do not aim to answer the question *Which method is better?*) but to provide the practitioner with empirical evidence to select the method that is most suited for the application at hand.

5.1. Data-driven predictive flight control

Let the discrete-time linear model system we aim at controlling be featured by the following matrices (Camacho & Bordons, 2007, Section 6.4):

$$A = \begin{bmatrix} 0.9997 & 0.0038 & -0.0001 & -0.0322 \\ -0.0056 & 0.9648 & 0.7446 & 0.0001 \\ 0.0020 & -0.0097 & 0.9543 & -0.0000 \\ 0.0001 & -0.0005 & 0.0978 & 1.0000 \end{bmatrix},$$
$$B = \begin{bmatrix} 0.0010 & 0.1000 \\ -0.0615 & 0.0183 \\ -0.1133 & 0.0586 \\ -0.0057 & 0.0029 \end{bmatrix}, \quad K = \mathbf{0}_{4\times 2}, \tag{53}$$
$$C = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & -1.0000 & 0 & 7.7400 \end{bmatrix},$$

which are obtained by discretizing the dynamics characterizing the longitudinal motion of a Boeing 747 with a zero-order-hold for a sampling time $T_s = 0.1$ [s]. The considered system has two inputs, namely the throttle u_1 and the angle of the elevator u_2, and two outputs, corresponding to the aircraft's longitudinal velocity and climb rate.

The control goal we aim at achieving is to increase the longitudinal velocity ($y_1(t)$) to 10 ft/s from the initial (0 ft/s) velocity while keeping the climb rate $y_2(t)$ at zero (equivalently, $r_y = \begin{bmatrix} 10 & 0 \end{bmatrix}^T$) and constraining the inputs and outputs as follows:

$$\mathbb{U} := \left\{ u \in \mathbb{R}^2 \ : \ \begin{bmatrix} -20 \\ -20 \end{bmatrix} \le u \le \begin{bmatrix} 20 \\ 20 \end{bmatrix} \right\}$$
$$\mathbb{Y} := \left\{ y \in \mathbb{R}^2 \ : \ \begin{bmatrix} -25 \\ -15 \end{bmatrix} \le y \le \begin{bmatrix} 25 \\ 15 \end{bmatrix} \right\}.$$

To achieve this, all the data-driven predictive control schemes with running cost as in (5) are designed by using the parameters reported in Table 8, equal to those reported in Lazar and Verheijen (2023), while the regularization penalties obtained for each scheme are reported in Table 9.

To construct the SPC, γ-DDPC and GDPC controllers we employ data-sets of lengths reported in Table 9, all generated by exciting the system with two pseudo-random binary sequences (PRBS), one for each of the system's inputs, both with amplitude 3. Throughout the data collection phase and when closing the loop, we assume that only the outputs of the system are corrupted by noise (as evident in (53)), with the measurement noise being zero-mean, white, Gaussian distributed, with standard deviation $0.25 I_2$. Note that, this choice yields a signal-to-noise ratio (SNR) of 39 [dB] when considering the data sequences of length $T = 2500$ used in SPC, γ-DDPC and GDPC and 26 [dB] when limiting T to 250 (see Table 8). For implementation reasons, we exploit a set of input/output data still generated as discussed above, but of length 2540, to carry out the identification step needed to construct the MPC scheme used to compare DPC approaches with their model-based counterpart. Moreover, we set $Q_{\text{obs}} = I_4$ and $R_{\text{obs}} = 0$ when designing the Luenberger observer.

[4] All simulations have been carried out on a laptop with Intel i7-9750H CPU, 16 GB of RAM, running MATLAB R2021a.

[5] The Luenberger observer has been computed in Matlab through the command `K = dlqr(A', C', Q_obs, R_obs)';`.

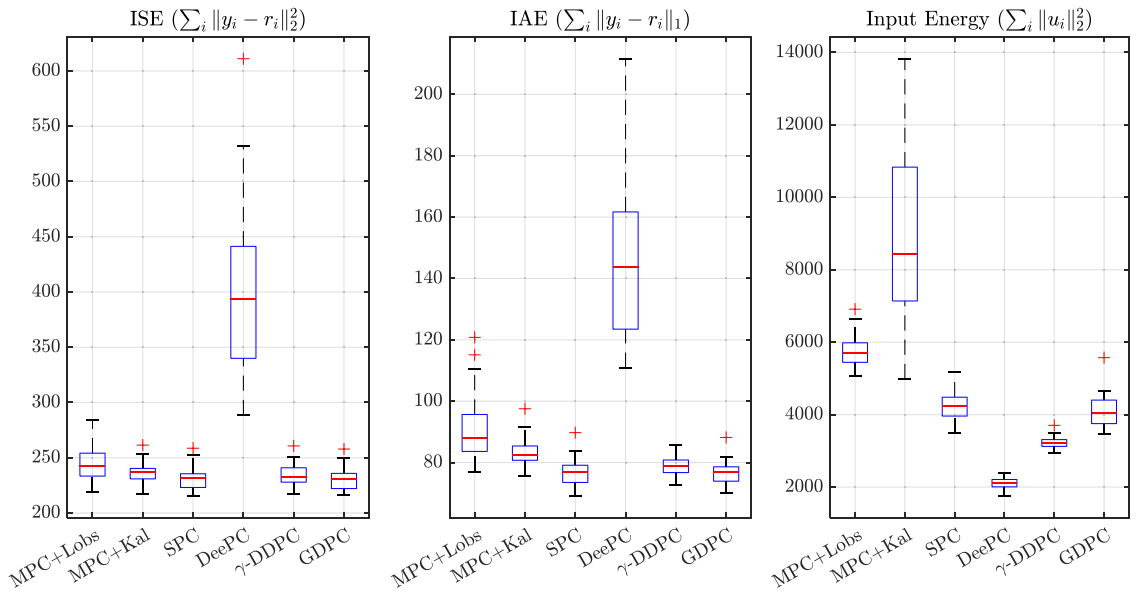

Fig. 2. Flight control: performance indexes over 30 Monte Carlo runs.

The results displayed in Fig. 2 suggest that all controllers handle noise well, showing generally low dispersion and similar averages of the performance indexes. The main differences can be observed between DeePC and all the other approaches, with DeePC resulting in a higher variance of ISE and IAE with the lowest input energy. Such differences can be related to the reduced dimension of the data-set used to construct this DPC scheme and the consequent lower SNR characterizing the available data.

Such a reduction in the data length is however inevitable for DeePC, as shown in Table 10. Indeed, despite our choice of T, DeePC still takes the longest time to be solved on average. Nonetheless, this result does not hamper the applicability of DeePC for this example, since all DPC problems can be solved within the required sampling period $T_s = 100$ [ms]. At the same time, despite considering datasets of the same dimension in (24) and (34), GDPC results to be more efficient than DeePC since it does not need the additional slack σ_y to cope with noise. Among the available approaches, γ-DDPC is instead the one with an average computational time closer to MPC and SPC, highlighting the possible advantages of pre-processing the data matrices prior to the controller deployment.

By looking at Fig. 2 it is also clear that both GDPC and γ-DDPC display improved performance over SPC, thus illustrating the benefits of having a predictor with tunable bias/variance trade-off. This is evident from the trajectories shown in Fig. 3, which also clearly indicate that both the considered model-based controllers show a significant input variance compared to the data-driven methods, while the resulting closed-loop output is not noticeably better.

5.2. Data-driven predictive control of a four tank system

Consider now the matrices

$$A = \begin{bmatrix} 0.921 & 0 & 0.041 & 0 \\ 0 & 0.918 & 0 & 0.033 \\ 0 & 0 & 0.924 & 0 \\ 0 & 0 & 0 & 0.937 \end{bmatrix}, \quad B = \begin{bmatrix} 0.017 & 0.001 \\ 0.001 & 0.023 \\ 0 & 0.061 \\ 0.072 & 0 \end{bmatrix}, \tag{54}$$

$$K = \mathbf{0}_{4\times 2}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

characterizing the linearized state-space model of a four tank system (Berberich et al., 2021b; Raff et al., 2006; Zhang et al., 2022). In this case, our control objective is to steer the measured levels of water in the tanks (y_1, y_2) to $r_y = \begin{bmatrix} 0.65 & 0.77 \end{bmatrix}^T$, while satisfying

$$\mathbb{U} := \left\{ u \in \mathbb{R}^2 \ : \ \begin{bmatrix} -2 \\ -2 \end{bmatrix} \leq u \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\}$$

$$\mathbb{Y} := \left\{ y \in \mathbb{R}^2 \ : \ \begin{bmatrix} -2 \\ -2 \end{bmatrix} \leq y \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\}.$$

To this end, we consider the parameters reported in Table 11, where Q, R and T_{ini} are equal to those used in Zhang et al. (2022), however, we considered a longer prediction horizon to ensure closed-loop stability. Indeed, as shown in Berberich et al. (2021b), this system becomes closed-loop unstable for the MPC controller if short horizons are selected. The hyper-parameters chosen for each DPC strategy are instead reported in Table 12, while (differently from the previous example) we design the Luenberger observer used in MPC by setting the state cost to $Q_{\text{obs}} = I_4$ and the output cost to $R_{\text{obs}} = 10 \cdot I_2$.

The data used to construct the DPC controllers are generated by feeding the system with a PRBS signal of amplitude of 1 on both input channels, while the outputs are corrupted with zero-mean, Gaussian distributed, white noise with standard deviation $0.02 I_2$, yielding a SNR of 12 [dB] when $T = 2500$ and of 10 [dB] for $T = 500$ (see Table 12). The models used in the MPC schemes are also constructed with data generated as described above, but the number of data samples used for identification grows to 2600 (note that this corresponds to the total data needed to construct the Hankel matrices with length $T = 2500$, and thus shares the same SNR).

The performance indexes attained by closing the loop in this setting are reported in Fig. 4, highlighting that the performance of all the tested control schemes is comparable (indeed only small variations in their indexes can be observed). This indicates that, when the hyper-parameters are properly tuned, all controllers can perform just as one another despite the presence of measurement noise corrupting the data. This is further shown in Fig. 5, where the main differences can be noticed in the slightly higher sensitivity to noise in the choice of the input for MPC and the resulting output for the MPC scheme with Kalman filter. Still, the trends in the mean and dispersion of the quality indexes achieved in this case, along with the general features of the closed-loop trajectories, are consistent with those characterizing the flight control example (see Fig. 2), strengthening the conclusion drawn with our two numerical examples. Last but not least, Table 13 once more allows us to draw the same conclusion on the computational

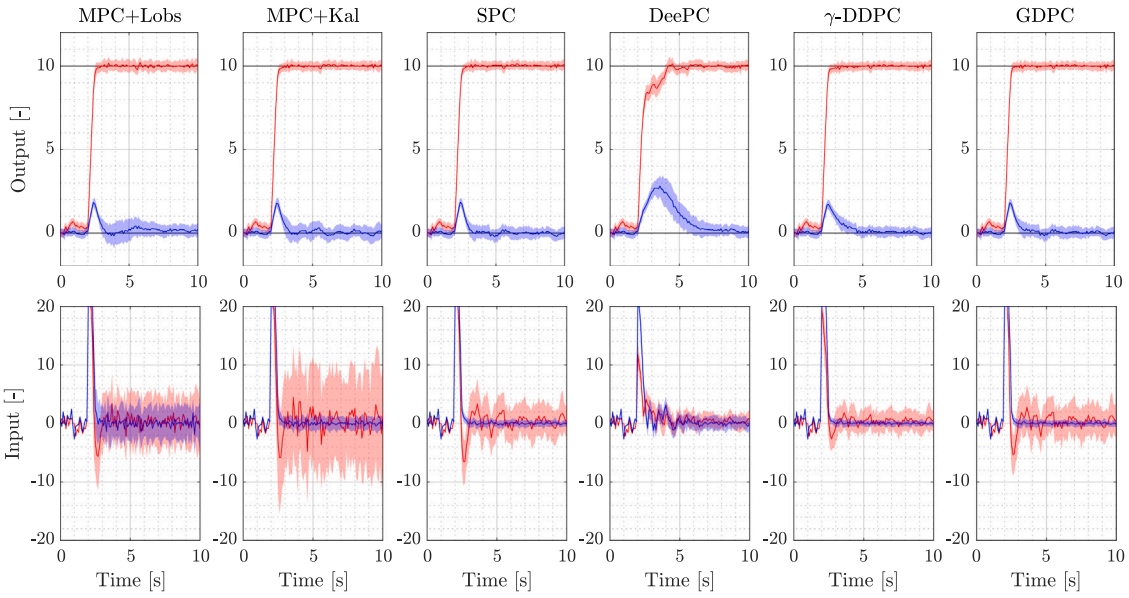

Fig. 3. Flight control: closed-loop mean (lines) input and output trajectories, with their standard deviation (shaded areas) over 30 Monte Carlo runs.

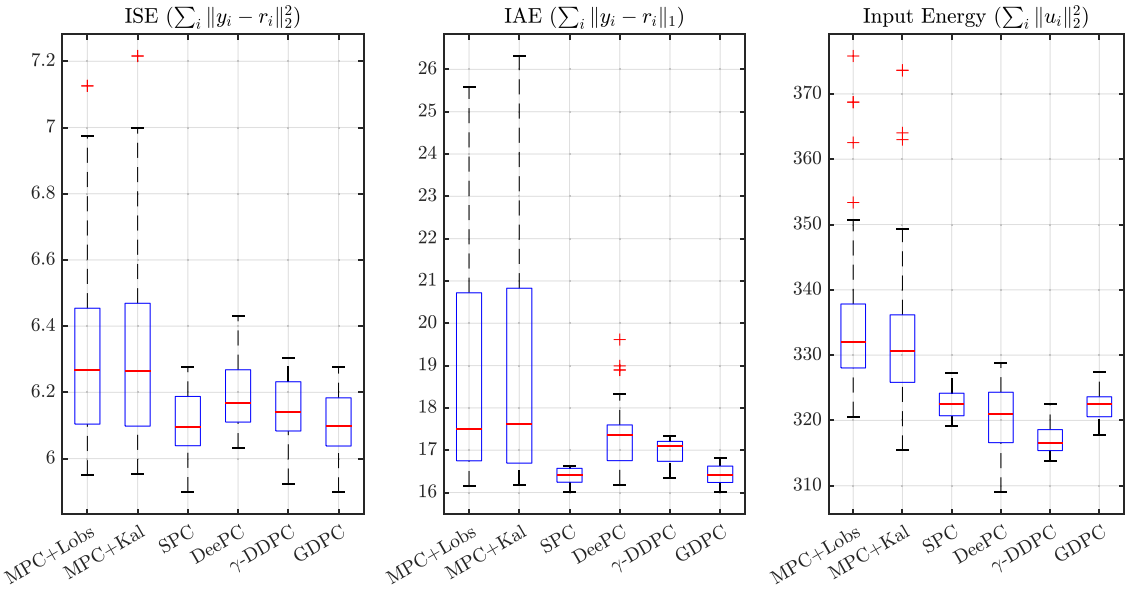

Fig. 4. Four tank system: performance indexes over 30 Monte Carlo runs.

Table 10
Flight control: average CPU time [ms] per sampling interval. Note that the system's sampling time is $T_s = 100$ [ms].

MPC+Lobs	MPC+Kal	SPC	DeePC	γ-DDPC	GDPC
3.5 ms	3.5 ms	4.1 ms	17.6 ms	4.8 ms	14.8 ms

Table 11
Four tank system: parameters of the running cost and length of the initial condition.

N	T_{ini}	R	Q	$\mathbf{r}_u$
70	30	$0.1I_2$	$15I_2$	$0I_2$

Table 12
Four tank system: regularization parameters and data-set lengths.

	T	T_g	λ_g	λ_y	β_2	β_3
SPC	2500	–	–	–	–	–
DeePC	500	–	10	10^6	–	–
γ-DDPC	2500	–	–	–	0	30
GDPC	2500	250	1	10^6	–	–

Table 13
Four Tank system: average CPU time [ms] per sampling interval.

MPC+Lobs	MPC+Kal	SPC	DeePC	γ-DDPC	GDPC
6.1 ms	6.2 ms	9.7 ms	104.1 ms	21 ms	53.4 ms

complexity of the considered approaches, indicating again that DeePC is the most time-consuming approach.

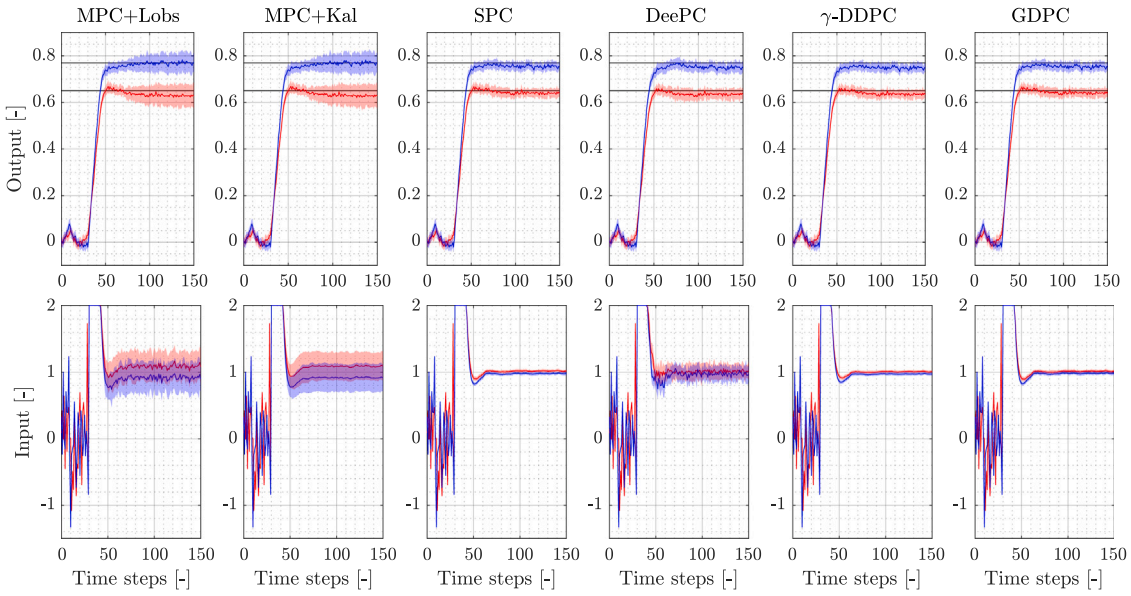

Fig. 5. Four tank system: closed-loop mean (lines) input and output trajectories, with their standard deviation (shaded areas) over 30 Monte Carlo runs.

5.3. Discussion and recommendations

The results that we obtained shed light on how identified model-based state-space MPC in combination with an observer or Kalman filter estimator compares with SPC and other data-driven predictive controllers in the presence of noisy data. From the obtained results we now see that identified model-based state-space MPC is not (necessarily) providing the best achievable performance. Data-driven methods can outperform the model-based controllers, even with appropriate tuning.

Even though SPC does not allow tuning the bias/variance trade-off, its closed-loop performance is unbiased and robust while providing the fastest CPU times. Hence, when an unbiased predictor is desired, SPC is likely to be the choice one should settle on. On the other hand, DeePC has the largest freedom to optimize the variance/bias trade-off, but its performance highly depends on λ_g and it is fragile in the presence of noisy data. DeePC can be robustified using the Π regularization, at the cost of increased computational complexity because the Hessian of the corresponding DeePC optimization problem is no longer sparse/diagonal. In general, the computational complexity of DeePC increases with the data size. Hence DeePC can be the preferred designer's choice when there is little or no noise and the minimal data size can be used. DeePC provides also the most direct approach to obtaining a predictive controller from data.

γ-DDPC and GDPC offer a middle ground alternative between SPC and DeePC, i.e., they allow some bias/variance trade-off, while still partially relying on the unbiased predictor of SPC. As such, their computational complexity is in between SPC and DeePC, with γ-DDPC being computationally more efficient than GDPC in general. On the other hand, GDPC offers more flexibility, as it can be tuned such that it recovers SPC or DeePC under appropriate settings. Hence, it enables a triple trade-off among bias, variance and computational complexity. If a bias/variance trade-off is desired with the least computational complexity, γ-DDPC offers a suitable solution. If one is interested in optimizing the bias/variance/computational complexity trade-off, GDPC provides a suitable framework.

6. Conclusions

In this review, we provided a bird's-eye view of data-driven predictive control, by looking at it through the lens of a practitioner approaching this class of techniques for the first time. By focusing on four DPC techniques, namely subspace predictive control, data-enabled predictive control, γ-DDPC and generalized data-driven predictive control, by providing a thorough description of the core technicalities needed by the potential designer for their implementation and usage. This methodological review is paired with a comprehensive overview of the main tuning knobs of these techniques and a summary of off-the-shelf strategies that can be employed by practitioners to select them. Two benchmark simulation examples were presented to highlight the main features, similarities, strengths and weaknesses of each approach in practice.

We conclude by distilling the guidelines presented into a set of "*golden rules*" of data-driven predictive control:

1. Use as much data as you can possibly collect (i.e., T very large), while minding its effect on the computational complexity of the DPC algorithm of choice;
2. Pick T_{ini} sufficiently large, $T_{\text{ini}} \gg n$ and, possibly, as large as you can;
3. Generate input data with a large order of persistency of excitation (PRBS, white noise, etc.) and compare

$$\operatorname{rank}\left(\begin{bmatrix}\mathbf{U}_p\\ \mathbf{Y}_p\\ \mathbf{U}_f\end{bmatrix}\right)$$

   with $n_u(T_{\text{ini}}+N)+T_{\text{ini}}n_y$ to validate the informativity/suitability of the collected data for DPC;
4. Tune Q and R first, then use the Hanke–Raus plots to pick a suitable λ_g, or use the rules tailored to γ-DDPC (see Section 4.6.2) when this method is employed;
5. Test the numerical stability and consistency of the DPC algorithm of choice using multiple QP solvers (e.g., Matlab quadprog, OSQP);
6. If persistent disturbances are present/expected, use an off-set free formulation of your DPC algorithm of choice using Algorithm 5.

With these rules of thumb, the controller will perform optimally since all the system properties that should be assumed are taken as sufficiently large.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The first and the third authors gratefully acknowledge the financial support of the research program AquaConnect, funded by the Dutch Research Council (NWO, grant-ID P19-45) and public and private partners of the AquaConnect consortium and coordinated by Wageningen University and Research.

References

Akaike, H. (1976). Canonical correlation analysis of time series and the use of an information criterion. In R. Mehra, & D. Lainiotis (Eds.), *System identification: Advances and case studies* (pp. 27–96). New York: Academic Press.

Alsalti, M., Lopez, V. G., Berberich, J., Allgöwer, F., & Müller, M. A. (2023). Data-based control of feedback linearizable systems. *IEEE Transactions on Automatic Control*, 1–8. http://dx.doi.org/10.1109/TAC.2023.3249289.

Baros, S., Chang, C.-Y., Colón-Reyes, G. E., & Bernstein, A. (2022). Online data-enabled predictive control. *Automatica*, *138*, Article 109926.

Berberich, J., Iannelli, A., Padoan, A., Coulson, J., Dörfler, F., & Allgöwer, F. (2023). A quantitative and constructive proof of Willems' Fundamental Lemma and its implications. In *IEEE CSL presented at ACC 2023* (pp. 4155–4160).

Berberich, J., Köhler, J., Müller, M. A., & Allgöwer, F. (2020). Data-driven tracking MPC for changing setpoints. *IFAC-PapersOnLine*, *53*(2), 6923–6930, 21st IFAC World Congress.

Berberich, J., Köhler, J., Müller, M. A., & Allgöwer, F. (2021a). Data-driven model predictive control: closed-loop guarantees and experimental results. *at - Automatisierungstechnik*, *69*(7), 608–618. http://dx.doi.org/10.1515/auto-2021-0024.

Berberich, J., Köhler, J., Müller, M. A., & Allgöwer, F. (2021b). Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, *66*(4), 1702–1717.

Berberich, J., Köhler, J., Müller, M. A., & Allgöwer, F. (2021c). On the design of terminal ingredients for data-driven MPC. *IFAC–PapersOnLine*, *54*(6), 257–263, 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021.

Bitmead, R. R., Gevers, M., & Wertz, V. (1990). *Adaptive optimal control the thinking man's GPC*. Prentice Hall.

Boccia, A., Grüne, L., & Worthmann, K. (2014). Stability and feasibility of state constrained MPC without stabilizing terminal constraints. *Systems & Control Letters*, *72*, 14–21.

Breschi, V., Chiuso, A., Fabris, M., & Formentin, S. (2023). On the impact of regularization in data-driven predictive control. arXiv:2304.00263.

Breschi, V., Chiuso, A., & Formentin, S. (2023). Data-driven predictive control in a stochastic setting: a unified framework. *Automatica*, *152*, Article 110961. http://dx.doi.org/10.1016/j.automatica.2023.110961.

Breschi, V., Fabris, M., Formentin, S., & Chiuso, A. (2023). Uncertainty-aware data-driven predictive control in a stochastic setting. arXiv:2211.10321.

Camacho, E. F., & Bordons, C. (2007). *Model predictive control*. Springer Verlag.

Campi, M., Lecchini, A., & Savaresi, S. (2002). Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, *38*(8), 1337–1346.

Carlet, P. G., Favato, A., Bolognani, S., & Dörfler, F. (2022). Data-driven continuous-set predictive current control for synchronous motor drives. *IEEE Transactions on Power Electronics*, *37*(6), 6637–6646.

Coulson, J. (2022). *Data-enabled predictive control theory and practice* (Ph.D. thesis), ETH Zürich.

Coulson, J., Lygeros, J., & Dörfler, F. (2019). Data-enabled predictive control: In the shallows of the DeePC. In *18th european control conference* (pp. 307–312). Napoli, Italy.

Coulson, J., Lygeros, J., & Dörfler, F. (2021). Distributionally robust chance constrained data-enabled predictive control. *IEEE Transactions on Automatic Control*, *67*(7), 3289–3304.

Coulson, J., Waarde, H. J. v., Lygeros, J., & Dörfler, F. (2023). A quantitative notion of persistency of excitation and the robust fundamental lemma. *IEEE Control Systems Letters*, *7*, 1243–1248. http://dx.doi.org/10.1109/LCSYS.2022.3232303.

De Persis, C., & Tesi, P. (2020). Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, *65*(3), 909–924. http://dx.doi.org/10.1109/TAC.2019.2959924.

Di Natale, L., Lian, Y., Maddalena, E. T., Shi, J., & Jones, C. N. (2022). Lessons learned from data–driven building control experiments: Contrasting Gaussian process-based MPC, bilevel DeePC and deep reinforcement learning. In *2022 IEEE 61st conference on decision and control (CDC)* (pp. 1111–1117).

Dörfler, F., Coulson, J., & Markovsky, I. (2022). Bridging direct & indirect data-driven control formulations via regularizations and relaxations. *IEEE Transactions on Automatic Control*, 1. http://dx.doi.org/10.1109/TAC.2022.3148374.

Elokda, E., Coulson, J., Beuchat, P. N., Lygeros, J., & Dörfler, F. (2021). Data-enabled predictive control for quadcopters. *International Journal of Robust and Nonlinear Control*, *31*, 8916–8936.

Favoreel, W., De Moor, B., & Gevers, M. (1999). SPC: Subspace predictive control. In *Proc. of the 14th IFAC world congress* (pp. 4004–4009). Elsevier.

Fiedler, F., & Lucia, S. (2021). On the relationship between data–enabled predictive control and subspace predictive control. In *IEEE proc. of the european control conference (ECC)* (pp. 222–229). Rotterdam, The Netherlands.

Formentin, S., & Chiuso, A. (2021). Control-oriented regularization for linear system identification. *Automatica*, *127*, Article 109539.

Gevers, M. (2005). Identification for control: From the early achievements to the revival of experiment design. *European Journal of Control*, *11*(4), 335–352.

Hanke, M., & Raus, T. (1996). A general heuristic for choosing the regularization parameter in ill-posed problems. *SIAM Journal on Scientific Computing*, *17*(4), 956–972.

Hou, Z.-S., & Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, *235*, 3–35.

Kerz, S., Teutsch, J., Brüdigam, T., Leibold, M., & Wollherr, D. (2023). Data-driven tube-based stochastic predictive control. *IEEE Open Journal of Control Systems*, *2*, 185–199. http://dx.doi.org/10.1109/OJCSYS.2023.3291596.

Knudsen, T. (2001). Consistency analysis of subspace identification methods based on a linear regression approach. *Automatica*, *37*(1), 81–89. http://dx.doi.org/10.1016/S0005-1098(00)00125-4, URL: https://www.sciencedirect.com/science/article/pii/S0005109800001254.

Köhler, J., Wabersich, K. P., Berberich, J., & Zeilinger, M. N. (2022). State space models vs. multi-step predictors in predictive control: Are state space models complicating safe data-driven designs? In *2022 IEEE 61st conference on decision and control (CDC)* (pp. 491–498). http://dx.doi.org/10.1109/CDC51059.2022.9992373.

Lamnabhi-Lagarrigue, F., Annaswamy, A., Engell, S., Isaksson, A., Khargonekar, P., Murray, R. M., Nijmeijer, H., Samad, T., Tilbury, D., & Van den Hof, P. (2017). Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Annual Reviews in Control*, *43*, 1–64.

Lazar, M. (2021). A dissipativity-based framework for analyzing stability of predictive controllers. In *IFAC PapersOnLine 54–6, 7th IFAC conference on nonlinear model predictive control* (pp. 159–165). Bratislava, Slovakia.

Lazar, M., & Verheijen, P. C. N. (2022). Offset–free data–driven predictive control. In *2022 IEEE 61st conference on decision and control (CDC)* (pp. 1099–1104). http://dx.doi.org/10.1109/CDC51059.2022.9992453.

Lazar, M., & Verheijen, P. C. N. (2023). Generalized data–driven predictive control: Merging subspace and Hankel predictors. *Mathematics*, *11*(9), http://dx.doi.org/10.3390/math11092216, URL: https://www.mdpi.com/2227-7390/11/9/2216.

Li, Z., Wang, H., Fang, Q., & Wang, Y. (2023). A data-driven subspace predictive control method for air-cooled data center thermal modelling and optimization. *Journal of the Franklin Institute*, *360*(5), 3657–3676. http://dx.doi.org/10.1016/j.jfranklin.2023.02.007, URL: https://www.sciencedirect.com/science/article/pii/S001600322300087X.

Ljung, L. (1999). *System identification: Theory for the user* (2nd ed.). Prentice Hall PTR.

Maciejowski, J. M. (2002). *Predictive control with constraints*. Prentice Hall.

Mardi, N. A., & Wang, L. (2009). Subspace-based model predictive control of time-varying systems. In *Proceedings of the 48h IEEE conference on decision and control (CDC) held jointly with 2009 28th chinese control conference* (pp. 4005–4010). http://dx.doi.org/10.1109/CDC.2009.5400285.

Markovsky, I., Huang, L., & Dörfler, F. (2023). Data–driven control based on the behavioral approach: From theory to applications in power systems. *IEEE Control Systems*.

van Overschee, P., & De Moor, B. (1996). *Subspace identification for linear systems*. Kluwer Academic Publishers.

Pan, G., Ou, R., & Faulwasser, T. (2023). Towards data-driven stochastic predictive control. *International Journal of Robust and Nonlinear Control*, http://dx.doi.org/10.1002/rnc.6812.

Raff, T., Huber, S., Nagy, Z. K., & Allgower, F. (2006). Nonlinear model predictive control of a four tank system: An experimental stability study. In *2006 IEEE conference on computer aided control system design, 2006 IEEE international conference on control applications, 2006 IEEE international symposium on intelligent control* (pp. 237–242). http://dx.doi.org/10.1109/CACSD-CCA-ISIC.2006.4776652.

Rawlings, J. B., Mayne, D. Q., & Diehl, M. M. (2017). *Model predictive control: Theory, computation, and design* (2nd ed.). Nob Hill Publishing.

Vajpayee, V., Mukhopadhyay, S., & Tiwari, A. P. (2018). Data-driven subspace predictive control of a nuclear reactor. *IEEE Transactions on Nuclear Science*, *65*(2), 666–679. http://dx.doi.org/10.1109/TNS.2017.2785362.

van den Hof, P. M. J. (1983). *Approximate realization of noisy linear systems: the Hankel and Page matrix approach* (Master's thesis), the Netherlands: Eindhoven University of Technology.

Van den Hof, P. M. J., & Schrama, R. J. P. (1995). Identification and control – Closed-loop issues. *Automatica*, *31*(12), 1751–1770.

Verheijen, P. C. N., Gonçalves da Silva, G. R., & Lazar, M. (2021). Data–driven rate–based integral predictive control with estimated prediction matrices. In *IEEE Proc. of the 25th international conference on system theory, control and computing (ICSTCC)* (pp. 630–636). Sinaia, Romania.

Verheijen, P., Gonçalves da Silva, G. R., & Lazar, M. (2022). Recursive data–driven predictive control with persistence of excitation conditions. In *2022 IEEE 61st conference on decision and control (CDC)* (pp. 467–473). http://dx.doi.org/10.1109/CDC51059.2022.9992366.

Verhoek, C., Abbas, H. S., Tóth, R., & Haesaert, S. (2021). Data-driven predictive control for linear parameter-varying systems. *IFAC-PapersOnLine*, *54*(8), 101–108. http://dx.doi.org/10.1016/j.ifacol.2021.08.588, 4th IFAC Workshop on Linear Parameter Varying Systems LPVS 2021.

van Waarde, H. J., De Persis, C., Camlibel, M. K., & Tesi, P. (2020). Willems' fundamental lemma for state-space systems and its extension to multiple datasets. *IEEE Control Systems Letters*, *4*(3), 602–607.

van Waarde, H. J., Eising, J., Trentelman, H. L., & Camlibel, M. K. (2020). Data informativity: A new perspective on data-driven analysis and control. *IEEE Transactions on Automatic Control*, *65*(11), 4753–4768.

Willems, J. C., Rapisarda, P., Markovsky, I., & De Moor, B. L. (2005). A note on persistency of excitation. *Systems & Control Letters*, *54*(4), 325–329.

van Wingerden, J.-W., Mulders, S. P., Dinkla, R., Oomen, T., & Verhaegen, M. (2022). Data-enabled predictive control with instrumental variables: the direct equivalence with subspace predictive control. In *2022 IEEE 61st conference on decision and control (CDC)* (pp. 2111–2116). http://dx.doi.org/10.1109/CDC51059.2022.9992824.

Yang, H., & Li, S. (2015). A data–driven predictive controller design based on reduced Hankel matrix. In *IEEE proc. of the 10th asian control conference (ASCC)* (pp. 1–7). Kota Kinabalu, Malaysia: IEEE.

Zhang, K., Chen, K., Lin, X., Zheng, Y., Yin, X., Hu, X., Song, Z., & Li, Z. (2023). Data-enabled predictive control for fast charging of lithium-ion batteries with constraint handling. arXiv:2209.12862.

Zhang, K., Zheng, Y., & Li, Z. (2022). Dimension reduction for efficient data-enabled predictive control. http://dx.doi.org/10.48550/ARXIV.2211.03697, arXiv.