DeePC-Hunt: Data-enabled Predictive Control Hyperparameter Tuning via Differentiable Optimization

Michael Cummins [1] **Alberto Padoan** [2] **Keith Moffat** [2] **John Lygeros** [2] **Florian Dörfler** [2]

Abstract

This paper introduces **D**ata-**e**nabl**e**d **P**redictive **C**ontrol **H**yperparameter **Tun**ing via Differen**t**iable Optimization (DeePC-Hunt), a backpropagation-based method for automatic hyperparameter tuning of the Data-enabled Predictive Control (DeePC) algorithm. The necessity for such a method arises from the critical importance of hyperparameter selection to achieve satisfactory closed-loop DeePC performance. The standard methods for hyperparameter selection are to either optimize the open-loop performance, or use manual guess-and-check. Optimizing the open-loop performance can result in unacceptable closed-loop behavior, while manual guess-and-check can pose safety challenges. DeePC-Hunt provides an alternative method for hyperparameter tuning which uses an approximate model of system dynamics and backpropagation to directly optimize hyperparameters for the closed-loop performance of DeePC. Numerical simulations demonstrate the effectiveness of DeePC in combination with DeePC-Hunt in a complex stabilization task for a nonlinear system and its superiority over model-based control strategies in terms of robustness to model misspecifications.

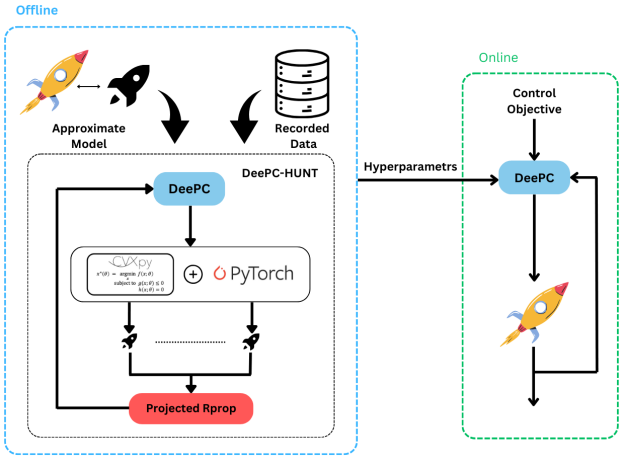


Figure 1. The DeePC-Hunt pipeline and the tools needed for implementation. The DeePC policy is instantiated using CvxpyLayers and PyTorch to enable automatic differentiation. Simulations are carried out on an approximate model of the system and projected resilient backpropagation is used to update the hyperparameters.

1. Introduction

Over the past decade, direct data-driven control methods have experienced a resurgence of interest (Coulson et al., 2019a; van Waarde et al., 2020; Waarde et al., 2020; Xue & Matni, 2021), primarily fueled by the increasing adoption of machine learning techniques and the availability of vast datasets. Unlike traditional control design paradigms that rely on system identification followed by model-based control, these methods compute control actions directly from data. Leveraging tools from behavioral system theory and convex optimization, this class of methods showcased significant promise in simplifying some of the complexities arising in traditional model-based control, while achieving satisfactory control performance with reduced implementation effort.

Among the vast array of methods in this rapidly expanding literature, Data-enabled Predictive Control (DeePC) (Coulson et al., 2019a) stands out as an effective direct data-driven control algorithm. Operating similarly to Model Predictive Control (MPC) in a receding-horizon manner (Borrelli et al., 2017), DeePC circumvents the need for an accurate state-space model, relying instead on a Hankel matrix derived from offline input/output raw data. Although stability guarantees have been established only under specific assumptions, the DeePC algorithm has demonstrated remarkable performance in controlling nonlinear systems affected by

[1]School of Electronic and Electrical Engineering, Trinity College Dublin [2]Department of Electrical Engineering and Information Technology, ETH Zürich. Correspondence to: Michael Cummins <micummin@tcd.ie>.

Workshop on Foundations of Reinforcement Learning and Control at the 41st International Conference on Machine Learning, Vienna, Austria.

noise. This performance is mainly attributed to regularization techniques (Coulson et al., 2019b) and is theoretically justified by tools from distributionally robust optimization (Coulson et al., 2022). However, the DeePC algorithm is often found to be sensitive to the choice of regularization parameters (Dörfler et al., 2023), presenting challenges where experiments are difficult, costly, or unsafe.

Today, the two standard methods for regularization tuning are analytical methods for the open-loop and manual guess-and-check. The analytical methods for the open-loop (Coulson et al., 2019b; Chiuso et al., 2023) are considered overly conservative because the regularization for the open-loop is more conservative than for the closed-loop, as the latter accounts for DeePC's ability to replan at each timestep. Manual guess-and-check methods, on the other hand, optimize the closed-loop performance but require potentially difficult experiments on the real system. Given the conservativeness of open-loop methods and the danger of manual methods, there is a need for the development and adoption of automatic hyperparameter selection strategies.

This paper introduces **D**ata-**e**nabl**e**d **P**redictive **C**ontrol **H**yperparameter T**un**ing via Differen**t**iable Optimization (DeePC-Hunt), a backpropagation-based method designed to automate hyperparameter tuning for DeePC. *DeePC-Hunt interprets DeePC, i.e., the solution of the DeePC optimization, as a control policy, and interprets the DeePC hyperparameters as the parameters of the policy*. Leveraging this DeePC-as-policy interpretation and an approximate model of the system dynamics, DeePC-Hunt optimizes the hyperparameters (to local optimality) using backpropagation (Rumelhart et al., 1986). Specifically, DeePC-Hunt implements a (constrained) variant of the resilient backpropagation algorithm (Riedmiller & Braun, 1993) and widely-used automatic differentiation tools (Paszke et al., 2019) to directly optimize the regularization hyperparameters of a DeePC policy based on the closed-loop performance of the policy deployed on the approximate model. Our findings suggest that DeePC in combination with DeePC-Hunt outperforms comparable model-based control strategies in terms of robustness to model misspecifications, without requiring extensive manual tuning.

Related work: DeePC-Hunt is motivated and inspired by policy optimization algorithms (Hu et al., 2022), which iteratively refine control policy parameters to minimize cumulative cost using different versions of the (projected) gradient descent algorithm. Unlike MPC or DeePC, DeePC-Hunt yields policies expressed as explicitly differentiable functions mapping states to actions. The implicit function theorem (Amos & Kolter, 2021) can be then used to differentiate the solution map of a Convex Optimisation Control Policy (COCP), where Karush-Kuhn-Tucker (KKT) conditions are differentiated to compute the gradient of the control input with respect to the hyperparameters. The work closest to ours are (Amos et al., 2019; Zuliani et al., 2024), which extend the theory of differentiating KKT conditions theory to MPC policies for automatic parameter tuning via gradient descent methods. However, (Amos et al., 2019; Zuliani et al., 2024) explicitly rely on state-space models, which restrict the model class and are not generally available. A broader approach to policy optimization for COCPs is presented in (Agrawal et al., 2020b), with experimental validation demonstrating the effectiveness of differentiable optimization for tuning a wide range of COCPs. Unlike previous approaches, DeePC-Hunt optimizes DeePC policies by integrating data from a system with data obtained from numerical simulations of an approximate model of the system, resulting in improved closed-loop performance.

Contributions:

i) We introduce the DeePC-Hunt algorithm for tuning the hyperparameters of a DeePC policy.

ii) To efficiently implement DeePC-Hunt, we introduce a variant of the resilient backpropagation algorithm (Riedmiller & Braun, 1993) that supports box constraints, and implement it using automatic differentiation tools (Paszke et al., 2019).

iii) We validate DeePC + DeePC-Hunt on a challenging benchmark task: landing a Vertical Takeoff and Vertical Landing (VTVL) vehicle on an oceanic platform using a realistic Gym environment simulator (Brockman et al., 2016).

Paper organization: The paper is organized as follows. Section 2 presents a concise overview of Differentiable Convex Optimization Layers, DeePC, and a version of Resilient Backpropagation (Riedmiller & Braun, 1993). Section 3 introduces our methodology; DeePC-Hunt. Section 4 demonstrates DeePC + DeePC-Hunt on the VTVL landing task and compares the performance with model-mismatched MPC. Section 5 concludes the paper with a summary of our findings and an outlook for future research directions.

Notation: The set of real numbers is denoted by $\mathbb{R}$. The set of real n-dimensional vectors is denoted by $\mathbb{R}^n$. The set of real $n \times m$-dimensional matrices is denoted by $\mathbb{R}^{n\times m}$. The set of positive integers is denoted by $\mathbb{N}$. The set of non-negative real numbers is denoted by $\mathbb{R}_+$. The set of $n \times n$-dimensional symmetric positive definite matrices is denoted by $\mathbb{S}_+^{n\times n}$. The set of closed, convex cones in $\mathbb{R}^m$ is denoted by $\mathcal{C}^m$. The transpose of the matrix $M \in \mathbb{R}^{p\times m}$ is denoted by $M^\intercal$. The p-norm of the vector $x \in \mathbb{R}^n$ is denoted by $|x|_p$. The weighted norm of a vector $x \in \mathbb{R}^n$ with the positive definite weight matrix $Q \in \mathbb{S}_+^{n\times n}$ is denoted by $|x|_Q^2 = x^\intercal Q x$. The i^{th} entry of a vector $v^{\mathrm{d}} \in \mathbb{R}^n$

is denoted by $v^{\mathrm{d}\,|i}$. The expectation operator is denoted by $\mathbb{E}$. The function $\operatorname{sign} : \mathbb{R} \to \{-1, 0, 1\}$ is defined as -1 if the argument is negative, 1 if the argument is positive, and 0 if the argument is zero. Given two vectors $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$, we define $\operatorname{col}(x_1, x_2) := (x_1^\intercal, x_2^\intercal)^\intercal$. Given $T \in \mathbb{N}$, the *Hankel matrix* of depth $L \in \mathbb{N}$, with $L \leq T$, associated with the vector $w \in \mathbb{R}^{qT}$ is denoted by

$$H_L(w) = \begin{bmatrix} w^1 & w^2 & \cdots & w^{T-L+1} \\ w^2 & w^3 & \cdots & w^{T-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ w^L & w^{L+1} & \cdots & w^T \end{bmatrix}.$$

2. Background

2.1. Differentiable Convex Optimization Layers

Consider the parameterized convex optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \;& f(x|\theta) \\ \text{s.t. } & g(x|\theta) \leq 0, \\ & h(x|\theta) = 0, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^{m_1}$ are convex functions, $h : \mathbb{R}^n \to \mathbb{R}^{m_2}$ is affine and θ represents the vector of parameters of f, g and h. The parameter θ belongs to a given set $\Theta \subseteq \mathbb{R}^\ell$, which represents the problem data. The solution to the class of convex optimization problems of the form (1), known as *disciplined parameterized programs* (Agrawal et al., 2019a), may be defined as a mapping $\theta \mapsto x^*(\theta)$ from the parameter θ to the global solution $x^*(\theta)$ of (1). Formally, we define the solution map of (1) as

$$s(\theta) := \theta \mapsto x^*(\theta). \tag{2}$$

Differentiable convex optimization layers (Agrawal et al., 2019a) provides a method for differentiating the global solution (2) with respect to the parameter θ. To this end, (1) is first transformed into a convex cone problem defined as

$$\begin{aligned} \min_{x,\nu} \;& c^\intercal x \\ \text{s.t. } & Ax + \nu = b, \\ & (x, \nu) \in \mathbb{R}^n \times \mathcal{K}, \end{aligned} \tag{3}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $\mathcal{K} \in \mathcal{C}^p$ are the problem data of (3). Such a transformation is typically performed through domain-specific languages for convex programming, such as CVXPY (CVXPY) (Diamond & Boyd, 2016), and (A, b, c) is determined via a sparse (linear) projection $L(\theta) = (A, b, c)$ (Agrawal et al., 2019b).

Thus, the solution map (2) can be decomposed into component functions:

$$s(\theta) = (R \circ S \circ L)(\theta), \tag{4}$$

where $L : \mathbb{R}^\ell \to \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n$ maps the problem data of (1) to the problem data of (3), $S : \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n \times \mathcal{K}$ is the solution map of (3) and $R : \mathbb{R}^n \times \mathcal{K} \to \mathbb{R}^n$ maps the solution of (3) to the solution of (1). Specifically, $S(A, b, c) = (x^*, \nu^*)$ and $R(x^*, \nu^*) = x^*$. The gradient of the solution map is then formulated as $\nabla s(\theta) = \nabla L(\theta) \nabla S(A, b, c) \nabla R(x^*, \nu^*)$. Since R and L are designed to be linear mappings, calculating their gradient is elementary. $\nabla S(A, b, c)$ may be obtained by differentiating the KKT conditions of (3), as detailed in (Agrawal et al., 2020a) (see also (Agrawal et al., 2019b;a; 2020a) for details).

The CvxpyLayers (Agrawal et al., 2019a) Python package integrates CVXPY and PyTorch (Paszke et al., 2019) to enable the transformation (1) $\mapsto$ (3) and the differentiation of (4) with respect to θ. DeePC, explained in the following section, can be interpreted as a convex optimization problem of the form (1). CvxpyLayers will be later used in the implementation of DeePC-Hunt to instantiate the DeePC policy (6), allowing us to differentiate the control inputs with respect to the regularization parameters.

2.2. The DeePC Algorithm

Consider a discrete-time dynamical system described by the equations

$$\begin{aligned} x_{k+1} &= f(x_k, u_k), \\ y_k &= h(x_k, u_k), \end{aligned} \tag{5}$$

where $u \in \mathbb{R}^m$ is the input of the system, $x \in \mathbb{R}^n$ is the state of the system and $y \in \mathbb{R}^p$ is the output of the system, respectively.

The DeePC algorithm leverages raw data matrices derived from offline input/output measurements of system (5) as a predictive model. Assuming the availability of offline/training data recorded from the (true) system (5), let $u^{\mathrm{d}} = \operatorname{col}(u^{\mathrm{d}|1}, \ldots, u^{\mathrm{d}|T}) \in \mathbb{R}^{mT}$ and $y^{\mathrm{d}} = \operatorname{col}(y^{\mathrm{d}|1}, \ldots, y^{\mathrm{d}|T}) \in \mathbb{R}^{pT}$ be vectors containing an input sequence of length $T \in \mathbb{N}$ applied to system (5) and the corresponding output sequence, respectively. Let $q = m + p$ and define the data vector $w^{\mathrm{d}} = \operatorname{col}(u^{\mathrm{d}|1}, \ldots, u^{\mathrm{d}|T}, y^{\mathrm{d}|1}, \ldots, y^{\mathrm{d}|T}) \in \mathbb{R}^{qT}$. For given initial and future time horizons $T_{\mathrm{ini}} \in \mathbb{N}$ and $T_{\mathrm{f}} \in \mathbb{N}$, we define the input and output data matrices as

$$\begin{pmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \end{pmatrix} = H_{T_{\mathrm{ini}}+T_{\mathrm{f}}}(u^{\mathrm{d}}), \quad \begin{pmatrix} Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{pmatrix} = H_{T_{\mathrm{ini}}+T_{\mathrm{f}}}(y^{\mathrm{d}}), .$$

where $H_t(z)$ converts the vector $z \in \mathbb{R}^T$ into a t-tall Hankel matrix with $T - t + 1$ columns. Furthermore, we define the past and future combined input-and-output data matrices as $W_{\mathrm{p}} = (U_{\mathrm{p}}^\intercal, Y_{\mathrm{p}}^\intercal)^\intercal$ and $W_{\mathrm{f}} = (U_{\mathrm{f}}^\intercal, Y_{\mathrm{f}}^\intercal)^\intercal$, respectively.

The DeePC algorithm operates in a receding-horizon fashion by iteratively solving an optimization problem using a data

matrix as a predictive model. Given a reference trajectory $w^{\mathrm{ref}} = \mathrm{col}(u^{\mathrm{ref}|1}, \dots, u^{\mathrm{ref}|T_{\mathrm{f}}}, y^{\mathrm{ref}|1}, \dots, y^{\mathrm{ref}|T_{\mathrm{f}}}) \in \mathbb{R}^{qT_{\mathrm{f}}}$, a vector of past input/output data at time k $w_k^{\mathrm{ini}} = \mathrm{col}(u_k^{\mathrm{ini}}, y_k^{\mathrm{ini}}) \in \mathbb{R}^{qT_{\mathrm{ini}}}$, where $u_k^{\mathrm{ini}} = \mathrm{col}(u_{k-T_{\mathrm{ini}}}, \dots, u_{k-1}) \in \mathbb{R}^{mT_{\mathrm{ini}}}$ and $y_k^{\mathrm{ini}} = \mathrm{col}(y_{k-T_{\mathrm{ini}}}, \dots, y_{k-1}) \in \mathbb{R}^{pT_{\mathrm{ini}}}$, an input constraint set $\mathcal{U} \subseteq \mathbb{R}^{mT_{\mathrm{f}}}$, an output constraint set $\mathcal{Y} \subseteq \mathbb{R}^{pT_{\mathrm{f}}}$, a positive-definite input cost matrix $R \in \mathbb{S}_+^{m\times m}$, a positive-definite output cost matrix $Q \in \mathbb{S}_+^{p\times p}$, a regularization function $\psi : \mathbb{R}^{T-T_{\mathrm{ini}}-T_{\mathrm{f}}+1} \times \mathbb{R}^{pT_{\mathrm{ini}}} \to \mathbb{R}_+^r$, and a weight vector $\lambda \in \mathbb{R}_+^r$, the DeePC optimization is:

$$\begin{aligned} \min_{u,y,g,\sigma_y} \ & \sum_{i=1}^{T_{\mathrm{f}}} |y_i - y^{\mathrm{ref}|i}|_Q^2 + |u_i - u^{\mathrm{ref}|i}|_R^2 + \lambda^\intercal \psi(g, \sigma_y), \\ \text{s.t.} \ & \begin{pmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{f}} \end{pmatrix} g = \begin{pmatrix} u_k^{\mathrm{ini}} \\ y_k^{\mathrm{ini}} \\ u \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \sigma_y \\ 0 \\ 0 \end{pmatrix}, \\ & (u, y) \in \mathcal{U} \times \mathcal{Y}. \end{aligned} \tag{6}$$

Noting that (6) is a parameterized convex optimization problem (λ is the parameter), we define its solution map in terms of λ, w_k^{ini} and w_k^{ref} as

$$\mathcal{S}_{w_{\mathrm{d}}}\left(\lambda, w_k^{\mathrm{ini}}, w^{\mathrm{ref}}\right) := \left(\lambda, w_k^{\mathrm{ini}}, w^{\mathrm{ref}}\right) \mapsto (w^*, g^*, \sigma_y^*), \tag{7}$$

where $w^* = \mathrm{col}(u^*, y^*)$ and (w^*, g^*, σ_y^*) is the optimal solution of the optimization problem (6) . Furthermore, defining the (linear) projection

$$\mathcal{P}(w^*, g^*, \sigma_y^*) = u_0^*,$$

where $u_0^* \in \mathbb{R}^m$ is the vector defined by the first m entries of w^*, we rewrite the DeePC control policy as

$$\pi_{w_{\mathrm{d}}}^{\lambda}(w_k^{\mathrm{ini}}, w^{\mathrm{ref}}) := (\mathcal{P} \circ \mathcal{S}_{w_{\mathrm{d}}})\left(\lambda, w_k^{\mathrm{ini}}, w^{\mathrm{ref}}\right). \tag{8}$$

The original formulation of the DeePC algorithm presented in (Coulson et al., 2019a) employs a one-norm regularizer on the decision variable $g \in \mathbb{R}^{T-T_{\mathrm{ini}}-T_{\mathrm{f}}+1}$, namely

$$\lambda^\intercal \psi(g, \sigma_y) = \lambda^1 |g|_1,$$

which promotes the selection of low-complexity models.

Alternatively, as detailed in (Dörfler et al., 2023), one may use the identification-induced regularizer

$$\lambda^\intercal \psi(g, \sigma_y) = \lambda^0 |(I - \Pi)g|_s,$$

with $s \in \mathbb{N}$ and

$$\Pi = \begin{pmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{pmatrix}^\dagger \begin{pmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{pmatrix}.$$

This regularizer leads to consistent predictions and connects to classic Subspace Predictive Control (SPC) algorithms (Favoreel & De Moor, 1999).

In this work, we leverage the regularization function

$$\lambda^\intercal \psi(g, \sigma_y) = \lambda^0 |(I - \Pi)g|_2^2 + \lambda^1 |g|_1.$$

This choice leads to improved performance, as suggested by findings in (Dörfler et al., 2023) and validated by our numerical simulations. Furthermore, a one-norm regularization is used to promote sparsity in the slack variable σ_y (Coulson et al., 2019a) yielding the regularization function

$$\lambda^\intercal \psi(g, \sigma_y) = \lambda^0 |(I - \Pi)g|_2^2 + \lambda^1 |g|_1 + \lambda^2 |\sigma_y|_1 \tag{9}$$

with the parameter vector $\lambda = \mathrm{col}(\lambda^0, \lambda^1, \lambda^2)$.

2.3. Projected Resilient Backpropagation

Consider the constrained optimization problem

$$\min_{\lambda \in \Lambda} \phi(\lambda), \tag{10}$$

where $\Lambda \subseteq \mathbb{R}_+^r$ and $\phi : \mathbb{R}^r \to \mathbb{R}$. Finding a locally optimal solution $\lambda^* \in \mathbb{R}_+^r$ may be challenging with a first-order iterative optimization algorithm (e.g., gradient descent); for example, the objective function ϕ is a convex quadratic function with a large condition number. Resilient backpropagation (Riedmiller & Braun, 1993) is a heuristic first-order iterative optimization algorithm that mitigates this issue by optimizing over each scalar element of the decision variable $\lambda = \mathrm{col}(\lambda^0, \dots, \lambda^{r-1})$ using only the *sign* of the gradient and an adaptive step-size. The update rule of the algorithm is defined as

$$\begin{aligned} \lambda_{k+1}^i &= \lambda_k^i - \eta_k^i \mathrm{sign}(\nabla_{\lambda^i} \phi(\lambda_k^i)), \\ \eta_{k+1}^i &= \mathrm{Rprop}(\eta_k^i, \lambda_k^i, \lambda_{k+1}^i | \eta^{\max}, \eta^{\min}, \beta, \alpha) \\ &= \begin{cases} \min\{\alpha \eta_k^i, \eta_{\max}\}, & \text{if } \nabla_{\lambda^i} \phi(\lambda_{k+1}^i) \nabla_{\lambda^i} \phi(\lambda_k^i) < 0, \\ \max\{\beta \eta_k^i, \eta_{\min}\}, & \text{if } \nabla_{\lambda^i} \phi(\lambda_{k+1}^i) \nabla_{\lambda^i} \phi(\lambda_k^i) > 0, \\ \eta_k^i, & \text{if } \nabla_{\lambda^i} \phi(\lambda_{k+1}^i) \nabla_{\lambda^i} \phi(\lambda_k^i) = 0, \end{cases} \end{aligned} \tag{11}$$

where $\mathrm{Rprop} : \mathbb{R}_+ \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the step-size generator and $(\eta^{\max}, \eta^{\min}, \beta, \alpha) \in \mathbb{R}_+^4$ are design parameters that represent the minimum step-size, maximum step-size, decay factor, and growth factor, respectively. Resilient backpropagation is particularly useful in the context of DeePC-Hunt, because the (locally) optimal DeePC regularization parameters can be orders of magnitude from an initial guess (e.g., $\lambda^{0*} = 1$ and $\lambda_1^* = 10^5$).

In addition to the challenge of the scale of the regularization, each regularization term λ^i must be non-negative. To enforce this constraint, we define the projection operator as

$$P_\Lambda(\lambda) = \operatorname*{argmin}_{\mu \in \Lambda} \frac{1}{2} |\lambda - \mu|^2,$$

where we recall that $\Lambda \subseteq \mathbb{R}^r_+$. Redefining (11), we obtain the modified iteration scheme

$$\begin{aligned}\gamma^i_{k+1} &= \gamma^i_k - \eta^i_k \text{sign}(\nabla_{\lambda^i}\phi(\lambda^i_k)),\\ \eta^i_{k+1} &= \text{Rprop}(\eta^i_k, \gamma^i_k, \gamma^i_{k+1} | \eta^{\max}, \eta^{\min}, \beta, \alpha),\\ \lambda_{k+1} &= P_\Lambda\left(\gamma_{k+1}\right),\end{aligned} \tag{12}$$

which ensures that the constraint $\lambda^k \in \Lambda$ is satisfied. To the best of the authors' knowledge, this paper is the first to employ a version of resilient backpropagation that ensures the feasibility of the iterates of (11) through a projection scheme.

3. DeePC-Hunt

As noted in the introduction and Section 2.2, the regularization parameter λ has a significant impact on the closed-loop performance of DeePC. DeePC-Hunt tunes λ to optimize the closed-loop performance of a DeePC policy implemented on a simplified or incorrect surrogate model of the system. The intuition for this approach is that the optimal regularization parameters of a DeePC controller are "close" for "similar" systems. By optimizing λ with respect to the closed-loop performance on the surrogate model, we hope to find a λ that also works for the true system. Thus, the surrogate model is used to find λ^*, but is not used to control the system; only the data from the true system is used by the DeePC policy.

The surrogate model is described by the equations

$$\begin{aligned}\tilde{x}_{k+1} &= \tilde{f}\left(\tilde{x}_k, \tilde{u}_k\right),\\ \tilde{y}_k &= \tilde{h}\left(\tilde{x}_k, \tilde{u}_k\right),\end{aligned} \tag{13}$$

where $\tilde{u} \in \mathbb{R}^m$, $\tilde{x} \in \mathbb{R}^n$ and $\tilde{y} \in \mathbb{R}^p$, respectively. The functions $\tilde{f}$ and $\tilde{h}$ approximate the input-output behavior of system (5). The approximate model may be derived from the physics of (a simplified version of) the system (5) or through the use of model reduction techniques (see, e.g., (Antoulas, 2005) for an overview of available techniques).

Without loss of generality, the timestep k at which DeePC-Hunt determines the optimal λ is set to 1. Given the approximate model (13), the vector w^{ini}_1 consisting of the prior T_{ini} measurements at timestep 1, the reference trajectory w^{ref}, and the DeePC policy $\pi^\lambda_{w_\text{d}}$ (Section 2.2), we define the approximate closed-loop cost, $\tilde{C}^{\pi^\lambda_{w_\text{d}}} : \mathbb{R}^{qT_{\text{ini}}} \times \mathbb{R}^{qT_\text{f}} \to \mathbb{R}_+$, over the N-long horizon (note: N may be different than T_f) as

$$\tilde{C}^{\pi^\lambda_{w_\text{d}}}\left(w^{\text{ini}}_1, w^{\text{ref}}\right) = \sum_{i=1}^{N} |\tilde{y}_i - y^{\text{ref}|i}|^2_Q + |u^*_i - u^{\text{ref}|i}|^2_R, \tag{14}$$

where u^*_i is the input produced by $\pi^\lambda_{w_\text{d}}\left(w^{\text{ini}}_i, w^{\text{ref}}\right)$ and $\tilde{y}_i$ is the output given by (13) when u^*_i is applied. At each timestep i, w^{ini}_{i+1} is given by inserting u^*_i and $\tilde{y}_i$ into w^{ini}_i, and removing the measurements furthest in the past corresponding to time $i - T_{\text{ini}}$. $\tilde{x}_0$, the initial state required by (13), is determined from w^{ini}_1 using a Kalman Filter or another state estimation method.

The aim is for λ to be effective for all initial conditions. Therefore, the expected closed-loop cost over a probability distribution of initial conditions w^{ini}_1 is optimized. (A similar procedure could be followed for w^{ref}, if the DeePC policy needs to work for a distribution of reference trajectories as well.) For simplicity, the w^{ini}_1 distribution is taken to be the uniform distribution over the columns of W_p, denoted as $\mathcal{D}(W_\text{p})$. The expected closed-loop cost is

$$\mathbb{E}_{w^{\text{ini}}_1 \sim \mathcal{D}(W_\text{p})}\left[\tilde{C}^{\pi^\lambda_{w_\text{d}}}\left(w^{\text{ini}}_1, w^{\text{ref}}\right)\right].$$

The operators, $\Psi(\cdot)$, $\Omega_u(\cdot)$, and $\Omega_y(\cdot)$, are defined for w^{ini}_i such that $\Psi(w^{\text{ini}}_i) = \tilde{x}_{i-1}$ and $\Omega_u(w^{\text{ini}}_i) = (u^{\text{ini}}_{i-T_{\text{ini}}+2}, \ldots, u^{\text{ini}}_i)$, and $\Omega_y(w^{\text{ini}}_i) = (y^{\text{ini}}_{i-T_{\text{ini}}+2}, \ldots, y^{\text{ini}}_i)$. The constrained, non-convex DeePC-Hunt optimization problem is

$$\min_{\lambda \in \Lambda} \mathbb{E}_{w^{\text{ini}}_1 \sim \mathcal{D}(W_\text{p})}\left[\tilde{C}^{\pi^\lambda_{w_\text{d}}}\left(w^{\text{ini}}_1, w^{\text{ref}}\right)\right], \tag{15}$$

which is equivalent to

$$\begin{aligned}\min_{\lambda \in \Lambda}\ & \mathbb{E}_{w^{\text{ini}}_1 \sim \mathcal{D}(W_\text{p})}\left[\sum_{i=1}^{N} |\tilde{y}_i - y^{\text{ref}|i}|^2_Q + |u^*_i - u^{\text{ref}|i}|^2_R\right],\\ \text{s.t. } & \tilde{x}_0 = \Psi(w^{\text{ini}}_1),\\ & \tilde{x}_{i+1} = \tilde{f}\left(\tilde{x}_i, \tilde{u}_i\right),\\ & \tilde{y}_i = \tilde{h}\left(\tilde{x}_i, \tilde{u}_i\right),\\ & w^{\text{ini}}_{i+1} = \text{col}(\Omega_u(w^{\text{ini}}_i), \tilde{u}_{i+1}, \Omega_y(w^{\text{ini}}_i), \tilde{y}_{i+1})\\ & \tilde{u}_i = \pi^\lambda_{w_\text{d}}\left(w^{\text{ini}}_i, w^{\text{ref}}\right)\end{aligned}$$

Note that this problem is a bi-level optimization problem since $\pi^\lambda_{w\text{d}}$ is itself an optimization problem. Recalling (7) and (8), $\pi^\lambda_{w\text{d}}$ is the solution map of a disciplined parameterized program with a linear projection applied. Therefore, $\pi^\lambda_{w\text{d}}$ may be differentiated using CvxpyLayers, which leverages the technique discussed in Section 2.1. Thus, the projected resilient backpropagation method described in (12) can be used to compute a locally optimal solution λ^* of (15).

Note that $\mathcal{D}(W_\text{p})$ is an empirical distribution over recorded trajectories and it is therefore possible to compute (15). Furthermore, (15) is an empirical risk minimization problem with dataset $\mathcal{D}(W_\text{p})$ and hypothesis $\pi^\lambda_{w\text{d}}$. If the sample size of $\mathcal{D}(W_\text{p})$ is large, one would require numerous simulations to compute the constraints/objective and take a single gradient step. To alleviate this issue, a similar approach is taken to Stochastic Gradient Descent (SGD) methods by computing an unbiased estimate of (15) in the form of Monte

Carlo samples, which is common practice when performing empirical risk minimization with large datasets (Ryu & Yin, 2022).

Pseudo code for the sample-estimate/Monte Carlo DeePC-Hunt implementation is given in Algorithm 1.

Algorithm 1 DeePC-Hunt

Given: Batch size $b \in \mathbb{R}_+$, closed-loop time horizon $N \in \mathbb{N}$, policy π_{w_d}, reference trajectory $w^{\mathrm{ref}} \in \mathbb{R}^{qT_\mathrm{f}}$, training time $t_{max} \in \mathbb{N}$, $(\eta^{\max}, \eta^{\min}, \beta, \alpha) \in \mathbb{R}_+^4$, and approximate dynamics $\tilde{f}$ and $\tilde{g}$.
Initialise: η_0^i, λ_0^i, γ_0^i, $\forall i = 1, \ldots, r$
for $k = 1 : t_{max}$ **do**
  for $j = 1 : B$ in parallel **do**
    $w_1^{\mathrm{ini},j} \sim \mathcal{D}(W_\mathrm{p})$
    $\tilde{x}_0^j = \Psi(w_1^{\mathrm{ini},j})$
    for $i = 0 : N-1$ **do**
      $\tilde{u}_i^j \leftarrow \pi_{w_\mathrm{d}}^{\lambda^k}\left(w_i^{\mathrm{ini},j}, w^{\mathrm{ref}}\right)$
      $\tilde{y}_i^j \leftarrow \tilde{h}\left(\tilde{x}_i^b, \tilde{u}_i^j\right)$
      $\tilde{x}_{i+1}^j \leftarrow \tilde{f}\left(\tilde{x}_i^j, \tilde{u}_i^j\right)$
      $w_{i+1}^{\mathrm{ini},j} \leftarrow \mathrm{col}(\Omega_u(w_i^{\mathrm{ini},j}), \tilde{u}_i^j, \Omega_y(w_i^{\mathrm{ini},j}), \tilde{y}_i^j)$
    end for
  end for
  $J(\lambda^k) \leftarrow \frac{1}{b}\sum_{j=1}^b \tilde{C}^{\pi_{w_\mathrm{d}}^\lambda}\left(w^{\mathrm{ini},j}, w^{\mathrm{ref}}\right)$
  for $i = 1 : r$ in parallel **do**
    $\gamma_{k+1}^i \leftarrow \gamma_k^i - \eta_k^i \mathrm{sign}(\nabla_{\lambda^i} J(\lambda_k^i))$
    $\eta_{k+1}^i \leftarrow \mathrm{Rprop}\left(\eta_k^i, \gamma_k^i, \gamma_{k+1}^i \,|\eta^{\max}, \eta^{\min}, \beta, \alpha\right)$
  end for
  $\lambda_{k+1} \leftarrow P_\Lambda\left(\gamma_{k+1}\right)$
end for

equations of motion

$$\begin{aligned} m\ddot{x} &= F_s \cos(\theta) - F_E \sin(\varphi + \theta) l_1, \\ m\ddot{y} &= F_s \cos(\theta) - F_E \sin(\varphi + \theta) l_1 - mg, \\ m\ddot{\theta} &= F_E \sin(2\pi - \varphi) l_1 - F_s l_2, \end{aligned} \tag{16}$$

where $x(t) \in \mathbb{R}$ is the horizontal position of the rocket, $y(t) \in \mathbb{R}$ is the vertical position of the rocket, $\theta(t) \in \mathbb{R}$ is the vertical pitch of the rocket, $F_s(t) \in \mathbb{R}$ is the force exerted by the side engines, $F_E(t) \in \mathbb{R}_+$ is the force exerted by the main engine, $\varphi(t) \in \mathbb{R}$ is the heading angle of the main engine, $m \in \mathbb{R}$ is the mass of the rocket, $l_1 \in \mathbb{R}_+$ is the length of the portion of the rocket below the center of gravity, and $l_2 \in \mathbb{R}_+$ is the length of the portion of the rocket above the center of gravity, respectively. A free-body diagram representing all forces acting on the VTVL vehicle is given in Fig. 2.

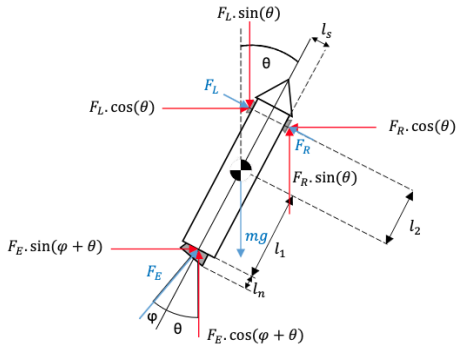


Figure 2. Free body diagram of the VTVL rocket taken from (Ferrante, 2017) and the corresponding equations of motion.

4. Numerical Simulation: landing a VTVL vehicle

4.1. Rocket Lander Gym Environment

The performance of DeePC-Hunt is demonstrated by considering the task of safely landing a Vertical Takeoff and Vertical Landing (VTVL) rocket on an oceanic platform. The objective is to land the VTVL vehicle on the designated landing pad in a vertical position, as illustrated in Fig. 3 and Fig. 4. The problem is motivated and inspired by a version of the Lunar Lander benchmark problem introduced in (Ferrante, 2017) and implemented using OpenAI's Gym Library (Brockman et al., 2016).

The dynamics of the VTVL vehicle are modeled by the

For simplicity, it is assumed that the landing pad remains stationary, perfect state information is available, and control inputs are chosen as $u = (F_E, F_s, \varphi) \in \mathbb{R}^3$, thus facilitating full actuation. The state and control variables are also required to satisfy given state and input box constraints $x(t) \in X$, $y(t) \in Y$, $\theta(t) \in \Theta$, and $u(t) \in U$, respectively.

4.2. MPC Policy

With perfect knowledge of the system parameters, MPC can be used to land the VTVL vehicle on the designated landing pad (Rawlings et al., 2017). Linearizing around an appropriately selected equilibrium point and discretizing the resulting system using zero-order-hold sampling gives the

following MPC control scheme

$$\begin{aligned} \min_{x\in\mathbb{R}^{6N},u\in\mathbb{R}^{3N}} \quad & \sum_{i=1}^{T_\mathrm{f}} |x_i - r|_Q^2 + |u_i|_R^2 \\ \text{s.t. } & x_0 = \hat{x}_0, \\ & x_{i+1} = Ax_i + Bu_i, \\ & (x,u) \in (X \times Y \times \mathbb{R}^2 \times \Theta \times \mathbb{R} \times U)^{T_\mathrm{f}}, \end{aligned} \tag{17}$$

where r is the reference trajectory for the state.

Numerical simulations suggest that the performance of policy (17) is highly sensitive on the quality of the state-space model (A, B). To illustrate this point, two distinct models are considered. The first model, named Model A, leverages the exact parameters of model (16). Model B leverages inaccurately estimated parameters adjusted by increasing l_1 by 33% and by decreasing l_2 and m by 25% and 50%, respectively.

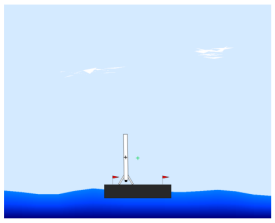
The performance of the MPC policy is illustrated in Fig. 3. Fig. 3a and 3b illustrate the trajectory of the VTVL vehicle under nominal conditions, confirming the effectiveness of a well-estimated model coupled with a linearized MPC scheme (17). However, performance significantly deteriorated when the incorrect system model was used. Figures 3c and 3d demonstrate that even minor perturbations impacting the information regarding the center of gravity led to adverse outcomes, resulting in an unsuccessful landing.

4.3. DeePC-Hunt Policy

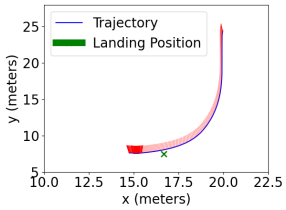
Two DeePC policies are considered: one with λ optimized using DeePC-Hunt with Model A, and the other with Model B. For both policies, we adopt the DeePC formulation (6), employing the regularization function (9), input constraints $\mathcal{U} = U^{T_\mathrm{f}}$, and output constraints $\mathcal{Y} = (X \times Y \times \mathbb{R}^2 \times \Theta \times \mathbb{R})^{T_\mathrm{f}}$. We set $T_\mathrm{f} = 10$, $T_\mathrm{ini} = 1$, and $\lambda^0 = (50, 50, 1000)$.

The offline/training data collection process involves applying a persistently exciting input trajectory $u_\mathrm{d} \in \mathbb{R}^{mT}$, with length $T \in \mathbb{N}$, to the VTVL vehicle and observing the corresponding output trajectory $y_\mathrm{d} \in \mathbb{R}^{pT}$. The inputs are determined via a Pseudo-Random Binary Sequence (PRBS) scheme, initiated from an equilibrium point $y_\mathrm{eq} \in \mathbb{R}^p$. The offline/training data w_d is used to build the Hankel matrices in (6) and the hyperparameters λ are determined by running DeePC-Hunt with either Model A or B, resulting in the DeePC policies $\pi_{w_\mathrm{d}}^{\lambda_A}$ and $\pi_{w_\mathrm{d}}^{\lambda_B}$, respectively.

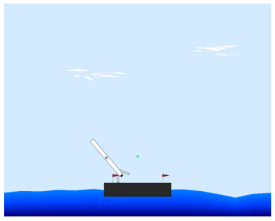
For the DeePC-Hunt training routine outlined in Algorithm 1, we set $t_\mathrm{max} = 100$, $r = 3$, $N = 20$, $B = 1$, $(\eta^\mathrm{max}, \eta^\mathrm{min}, \beta, \alpha) = (10^2, 10^{-3}, 1.2, 0.5)$, and $\Lambda = \{(\lambda^0, \lambda^1, \lambda^2) \in \mathbb{R}^3 \mid \lambda^i \in [\lambda^\mathrm{lower}, \lambda^\mathrm{upper}]\}$ where $\lambda^\mathrm{lower} = 10^{-5}$ and $\lambda^\mathrm{upper} = 10^5$.



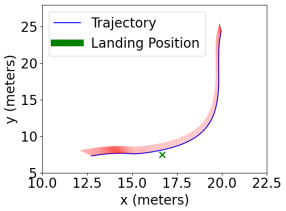
(a) Final position of the VTVL vehicle using the parameters $m = 530.41$kg, $l_1 = 2.85$m, $l_2 = 2.14$m



(b) Trajectory of VTVL vehicle using the correct parameters $m = 530.41$kg, $l_1 = 2.85$m, $l_2 = 2.14$m



(c) Final position of the VTVL vehicle using the incorrect parameters $m = 265.201$kg, $l_1 = 2.14$m, $l_2 = 2.85$m



(d) Trajectory of the VTVL vehicle using the incorrect parameters $m = 265.201$kg, $l_1 = 2.14$m, $l_2 = 2.85$m

Figure 3. Performance of the MPC policies with $Q = \mathrm{diag}(100, 10, 5, 1, 3000, 30)$, $R = \mathrm{diag}(0.01, 0.01, 0.01)$, $X = [0, 33.33]$, $Y = [0, 26.66]$, and $\Theta = [-0.61, 0.61]$, $U = [0, 16118.5] \times [0, 322.37] \times [-0.26, 0.26]$ and $N = 10$.

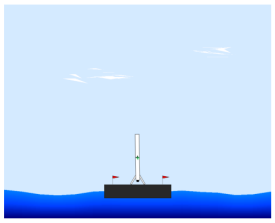
Figure 4 illustrates the performance of the DeePC policies in combination with DeePC-Hunt. Both DeePC policies, trained using accurately and inaccurately estimated models, exhibit satisfactory performance from the predefined position, displaying nearly identical trajectories. This highlights the effectiveness of DeePC in conjunction with DeePC-Hunt, even in scenarios with inaccurately modeled dynamics.

4.4. Evaluation of Control Performance

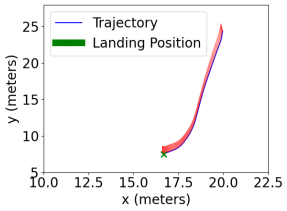
To perform a comprehensive comparison, we introduce two metrics. The first metric evaluates the closed-loop system cost $I : \mathcal{U} \times \mathcal{Y} \to \mathbb{R}_+$, defined as

$$I(y,u) = \sum_{i=1}^{N} |y_i - y^{\mathrm{ref}|i}|_Q^2 + |u_i - u^{\mathrm{ref}|i}|_R^2,,$$

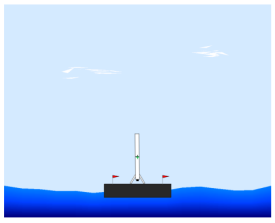
where N_π is the number of time steps it took the policy π to land the VTVL vehicle. The second metric assesses the empirical success rate across various initial conditions. We uniformly sample $k = 50$ initial states within a specified range $(x, y) \in X \times Y$ and compute the percentage of successful landings from these initial positions. Results for both MPC and DeePC in combination with DeePC-Hunt are presented in Table 1.



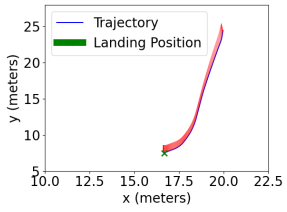
(a) Final position of rocket using DeePC (B) with $\lambda_B = (27.475, 2.128, 946.05)$.



(b) Trajectory of rocket using DeePC (B) with $\lambda_B = (27.475, 2.128, 946.05)$.



(c) Final position of rocket using DeePC (A) with $\lambda_A = (49.837, 8.364, 1000.05)$.



(d) Trajectory of rocket using DeePC (A) with $\lambda_A = (49.837, 8.364, 1000.05)$.

Figure 4. Performance of the DeePC policies with $Q = \text{diag}(100, 10, 5, 1, 3000, 30)$, $R = \text{diag}(0.01, 0.01, 0.01)$, $X = [0, 33.33]$, $Y = [0, 26.66]$, and $\Theta = [-0.61, 0.61]$, $U = [0, 16118.5] \times [0, 322.37] \times [-0.26, 0.26]$ and $N = 10$.

	Cost ($\times 10^3$)	**% Success**
MPC (A)	**11.228**	46%
DeePC (A)	16.678	**56%**
MPC (B)	**8.213**	22%
DeePC (B)	12.165	**66%**

Table 1. Results for DeePC and MPC policies using both models. The DeePC policy trained with model A and B, converges to $\lambda_A = (49.837, 8.364, 1000.05)$ and $\lambda_B = (27.475, 2.128, 946.05)$, respectively. We track the averaged closed-loop cost over $k = 50$ simulations with different initial conditions. When sampling the initial conditions, we set $X = [6.67, 26.66]$ and $Y = [18.66, 24]$.

DeePC notably outperforms MPC in success rate, especially under model inaccuracies. However, a trade-off emerges as MPC exhibits lower closed-loop costs upon successful landings. This discrepancy primarily arises from the impact of regularization parameters on the cost function in (6), prioritizing regularization (g, σ_y) over progress towards the setpoint w^{ref}. Employing model B with DeePC-Hunt yields significantly improved performance over direct MPC use.

Our findings suggest promising implications for real-world systems, indicating that even with a poor approximation of true dynamics, effective regularization parameters can be easily identified, ensuring good performance on the true system with a DeePC policy. Given its ease of implementation, DeePC-Hunt appears as a practical solution for automated parameter tuning. It is particularly advantageous when exploring multiple parameters is costly, while developing a poorly approximated model is easy.

5. Conclusion

The paper has introduced DeePC-Hunt, a backpropagation-based method for automatic hyperparameter tuning of the DeePC algorithm. We have demonstrated the efficacy of backpropagation for hyperparameter tuning. Furthermore, we have shown that DeePC-Hunt outperforms model-mismatched MPC on a challenging nonlinear control task, without requiring manual tuning. Our findings suggest promising implications for real-world systems—even with a poor approximation of true dynamics, effective regularization parameters can be identified offline that provide good closed-loop performance on the true system. Given its ease of implementation, DeePC-Hunt appears to be a practical solution for automated DeePC parameter tuning.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, Z. Differentiable convex optimization layers, 2019a.

Agrawal, A., Verschueren, R., Diamond, S., and Boyd, S. A rewriting system for convex optimization problems, 2019b.

Agrawal, A., Barratt, S., Boyd, S., Busseti, E., and Moursi, W. M. Differentiating through a cone program, 2020a.

Agrawal, A., Barratt, S., Boyd, S., and Stellato, B. Learning convex optimization control policies. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 361–373. PMLR, 10–11 Jun 2020b.

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks, 2021.

Amos, B., Rodriguez, I. D. J., Sacks, J., Boots, B., and Kolter, J. Z. Differentiable MPC for end-to-end planning and control, 2019.

Antoulas, A. C. *Approximation of large-scale dynamical systems*. SIAM, 2005.

Borrelli, F., Bemporad, A., and Morari, M. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym, 2016. URL https://arxiv.org/abs/1606.01540.

Chiuso, A., Fabris, M., Breschi, V., and Formentin, S. Harnessing the final control error for optimal data-driven predictive control, 2023.

Coulson, J., Lygeros, J., and Dörfler, F. Data-enabled predictive control: In the shallows of the DeePC. In *2019 18th European Control Conference (ECC)*, pp. 307–312, 2019a. doi: 10.23919/ECC.2019.8795639.

Coulson, J., Lygeros, J., and Dörfler, F. Regularized and distributionally robust data-enabled predictive control. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 2696–2701, 2019b. doi: 10.1109/CDC40024.2019.9028943.

Coulson, J., Lygeros, J., and Dörfler, F. Distributionally robust chance constrained data-enabled predictive control. *IEEE Transactions on Automatic Control*, 67(7):3289–3304, 2022. doi: 10.1109/TAC.2021.3097706.

Diamond, S. and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Dörfler, F., Coulson, J., and Markovsky, I. Bridging direct and indirect data-driven control formulations via regularizations and relaxations. *IEEE Transactions on Automatic Control*, 68(2):883–897, 2023. doi: 10.1109/TAC.2022.3148374.

Favoreel, W. and De Moor, B. SPC: Subspace predictive control. *IFAC Proceedings Volumes*, 32, 01 1999. doi: 10.1016/S1474-6670(17)56683-5.

Ferrante, R. A robust control approach for rocket landing. 2017. URL https://api.semanticscholar.org/CorpusID:245352859.

Hu, B., Zhang, K., Li, N., Mesbahi, M., Fazel, M., and Başar, T. Towards a theoretical foundation of policy optimization for learning control policies, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Rawlings, J., Mayne, D., and Diehl, M. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017. ISBN 9780975937730.

Riedmiller, M. and Braun, H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, 1993. doi: 10.1109/ICNN.1993.298623.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Ryu, E. K. and Yin, W. *Large-Scale Convex Optimization: Algorithms 38;amp; Analyses via Monotone Operators*. Cambridge University Press, 2022.

van Waarde, H. J., De Persis, C., Camlibel, M. K., and Tesi, P. Willems' fundamental lemma for state-space systems and its extension to multiple datasets. *IEEE Control Systems Letters*, 4(3):602–607, 2020. doi: 10.1109/LCSYS.2020.2986991.

Waarde, H., Eising, J., Trentelman, H., and Camlibel, K. Data informativity: A new perspective on data-driven analysis and control. *IEEE Transactions on Automatic Control*, PP:1–1, 01 2020. doi: 10.1109/TAC.2020.2966717.

Xue, A. and Matni, N. Data-driven system level synthesis. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pp. 189–200. PMLR, 07 – 08 June 2021.

Zuliani, R., Balta, E. C., and Lygeros, J. BP-MPC: Optimizing the closed-loop performance of MPC using backpropagation, 2024.