The discrete fully probabilistic design algorithm: a tool to design control policies from examples

Enrico Ferrentino EFERRENTINO@UNISA.IT
Dept. of Information and Electrical Eng. & Applied Math. at the University of Salerno, Italy

Pasquale Chiacchio PCHIACCHIO@UNISA.IT
Dept. of Information and Electrical Eng. & Applied Math. at the University of Salerno, Italy

Giovanni Russo GIOVARUSSO@UNISA.IT
Dept. of Information and Electrical Eng. & Applied Math. at the University of Salerno, Italy

Abstract

This paper is concerned with the synthesis of control policies from example datasets for constrained nonlinear systems. The constraints do not need to be fulfilled in the possibly noisy example data, which in turn might be collected from a system that is different from the one under control. We present a discretized design that expounds an algorithm recently introduced in Gagliardi and Russo (2021) for control synthesis from examples. For this discretized design, we discuss a number of key properties and give a design pipeline. The design, which we term as discrete fully probabilistic design algorithm, is benchmarked numerically on an example that involves controlling an inverted pendulum with actuation constraints starting from data collected from a physically different pendulum that does not satisfy the system-specific actuation constraints.

Keywords: Fully probabilistic design, control from examples, data-driven control

1. Introduction

Over the past few years, much research effort has been devoted to the problem of synthesizing control polices directly from data, bypassing the need to devise and identify a mathematical model in the form of difference/differential equations (Hou and Wang, 2013). An appealing *data-driven* control framework is that of designing controllers by using example data (Hanawal et al., 2019; Gagliardi and Russo, 2021; Al Makdah et al., 2021; Tu et al., 2021). Within this *control from examples framework*, one seeks to synthesize policies so that the closed-loop system *tracks* some desired behavior extracted from the examples. In this context, we consider the challenge of synthesizing control policies for nonlinear constrained systems from noisy example data. These data might be collected from a system that is different from the one under control and might not satisfy the system-specific constraints, which might be unknown when the examples are collected.

Statement of the contributions. We introduce, and benchmark, a discretized design that expounds an algorithm presented in Gagliardi and Russo (2021). The algorithm is used for the synthesis of control policies from examples. We term this discretized design *discrete fully probabilistic design algorithm* (DFPD). The proposed design is a numerical/discrete version of the algorithm from Gagliardi and Russo (2021): this latter algorithm can tackle both discrete and continuous control problems by finding an analytical solution. Here, we propose to find a purely numerical solution. As the original algorithm, the numerical/discrete implementation can be used to compute

policies from examples for constrained, possibly stochastic/nonlinear, systems. We discuss a number of properties of DFPD and give a design pipeline for the use of the algorithm. The design, which is numerically benchmarked on an inverted pendulum, is fully documented and the code is openly available[1].

Related work. We briefly survey a number of works that are related to the probabilistic design framework we use in this paper. We leverage a Bayesian approach (Peterka, 1981) to dynamical systems that allows to represent the behaviors of these systems via probability functions. In the context of control, the approach has been leveraged in e.g. Kárný (1996); Kárný and Guy (2006); Kárný and Kroupa (2012); Herzallah (2015); Guy et al. (2018); Pegueroles and Russo (2019) for the design of randomized control policies that enable tracking of a given target behavior. In these papers, the tracking problem is tackled by setting-up an unconstrained optimization problem. Closely related works, include Todorov (2009, 2007); Kappen et al. (2012). These works, by leveraging a similar framework, formalize the control problem as the (unconstrained) problem of minimizing a cost that captures the discrepancy between an ideal probability density function and the actual probability density function of the system under control. An online version of these algorithms has been proposed in Guan et al. (2014): in such a work, by leveraging an average cost formulation, the probability mass function for the state transitions is found. Finally, we also recall here Russo (2021); Garrabe and Russo (2022), where policies are obtained from the minimization of similar costs by leveraging multiple, specialized, datasets. These last papers, together with Gagliardi and Russo (2021), introduce constraints to the probabilistic formulation. Finally, we also recall Gandhi et al. (2021), where a decision making architecture for Robust Model-Predictive Path Integral Control is proposed and this indeed allows the introduction of the constraints (see also references therein).

The paper is organized as follows. After giving the mathematical preliminaries and formalizing the statement of the problem (Section 2), we present the DFPD algorithm. This is done in Section 3, where we also discuss a number of its properties. In Section 4 we describe a design pipeline to use DFPD. The pipeline illustrates a process to determine, via DFPD, control inputs from the data. Finally, the effectiveness of DFPD is illustrated in Section 5 on a pendulum with actuation constraints. Concluding remarks are given in Section 6.

2. Mathematical set-up and problem statement

We denote sets via *calligraphic* capital characters and vectors in **bold**. Multidimensional random variables and their realizations are both denoted by lower-case bold letters. All the random variables we consider are discrete and sampled from probability mass functions (pmfs) with compact supports. Throughout the paper, we also refer to pmfs as (normalized) *histograms* and to its domain as (discrete) *alphabet*. The notation $\mathbf{z} \sim P(\mathbf{z})$ denotes the fact that the random variable $\mathbf{z}$ is sampled from the pmf $P(\mathbf{z})$. Given the set $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$, its *indicator function* is denoted by $\mathbb{1}_{\mathcal{Z}}(\mathbf{z})$, so that $\mathbb{1}_{\mathcal{Z}}(\mathbf{z}) = 1 \, \forall \mathbf{z} \in \mathcal{Z}$ and 0 otherwise. We also let $\mathbb{E}_P[\mathbf{h}(\mathbf{z})] := \sum P(\mathbf{z})\mathbf{h}(\mathbf{z})$, where the sum is implicitly assumed to be taken over the support of $P(\mathbf{z})$, be the expectation of a function, say $\mathbf{h}(\cdot)$, of $\mathbf{z}$. The joint pmf of $\mathbf{z}_1$ and $\mathbf{z}_2$ is denoted by $P(\mathbf{z}_1, \mathbf{z}_2)$ and the conditional pmf of $\mathbf{z}_1$ given $\mathbf{z}_2$ is denoted by $P(\mathbf{z}_1|\mathbf{z}_2)$. The control problem considered in this paper is stated in terms of the Kullback-Leibler (or simply KL) divergence (Kullback and Leibler, 1951). Given the histograms

1. https://github.com/unisa-acg/discrete-fpd

$P(\boldsymbol{\alpha})$ and $Q(\boldsymbol{\alpha})$ defined over the discrete alphabet $\mathcal{A}$, the KL-divergence is defined as

$$\mathcal{D}_{KL}(P||Q) := \sum_{\alpha \in \mathcal{A}} P(\alpha) \ln \frac{P(\alpha)}{Q(\alpha)}. \tag{1}$$

Formulation of the control problem. Let $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{d_x}$ be the state of a dynamical system, and $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{d_u}$ be the control variable. The time variable is $t \in [0, T]$, $T < +\infty$. The time interval $[0, T]$ is discretized in $n+1$ instants with step $\Delta t > 0$. This yields the sequence of time-instants $t(k) = k\Delta t$, with $k = 0, \ldots, n$ and $T = n\Delta t$. We also let $\mathbf{x}(k)$ and $\mathbf{u}(k)$ be $\mathbf{x}$ and $\mathbf{u}$ evaluated in the discretized time. We recall that $\mathcal{X}$ and $\mathcal{U}$ are both compact and we set

$$\mathcal{X} := \{\mathbf{x}_0, \ldots, \mathbf{x}_{m-1}\}, \quad \mathcal{U} := \{\mathbf{u}_0, \ldots, \mathbf{u}_{z-1}\}\ \forall k. \tag{2}$$

For notational convenience, we use the shorthand notation $\mathbf{x}_j(k)$ (resp. $\mathbf{u}_h(k)$) to denote that $\mathbf{x}$ (resp. $\mathbf{u}$) at time-instant k equals the value of the j-th (resp. h-th) element in $\mathcal{X}$ (resp. $\mathcal{U}$).

Following Peterka (1981); Kárný (1996) one can describe the behavior of a given system by computing the joint pmf $P^n := P\big(\mathbf{x}(0), \ldots, \mathbf{x}(n), \mathbf{u}(0), \ldots, \mathbf{u}(n-1)\big)$, obtained from e.g. state-input data collected from the system. Likewise, one can also define a desired (or reference) behavior for the system in terms of a joint pmf, say $Q^n := Q\big(\mathbf{x}(0), \ldots, \mathbf{x}(n), \mathbf{u}(0), \ldots, \mathbf{u}(n-1)\big)$. Then, by making the standard Markov's assumption, the chain rule for pmfs yields the following convenience factorization:

$$\begin{aligned}
P^n &= \prod_{k=1}^{n} P\big(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{u}(k-1)\big) P\big(\mathbf{u}(k-1)|\mathbf{x}(k-1)\big) P(\mathbf{x}(0)) \\
&=: \prod_{k=1}^{n} \tilde{P}_X^k \tilde{P}_U^{k-1} P(\mathbf{x}(0)) =: \prod_{k=1}^{n} \tilde{P}^k P(\mathbf{x}(0)), \\
Q^n &= \prod_{k=1}^{n} Q\big(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{u}(k-1)\big) Q\big(\mathbf{u}(k-1)|\mathbf{x}(k-1)\big) Q(\mathbf{x}(0)) \\
&=: \prod_{k=1}^{n} \tilde{Q}_X^k \tilde{Q}_U^{k-1} Q(\mathbf{x}(0)) =: \prod_{k=1}^{n} \tilde{Q}^k Q(\mathbf{x}(0)).
\end{aligned} \tag{3}$$

The former term in the first product of the definitions takes the name of *state evolution model*, while the latter is the *randomized control law* (Kárný and Guy, 2006). In the context of this paper, the pmf Q^n is collected from example data. This allows us to state the problem of synthesizing control policies from example data for systems with actuation constraints as follows.

Problem 1 (Global optimization problem) *Find a solution to the optimization problem*

$$\begin{aligned}
\min_{\tilde{P}_U^0, \ldots, \tilde{P}_U^{n-1}} \ & \mathcal{D}_{KL}(P^n||Q^n) \\
\textit{s.t. } & \mathbf{f}_k(p_{hi}^k) \leq \mathbf{0}\ \forall k \\
& \mathbf{g}_k(p_{hi}^k) = \mathbf{0}\ \forall k
\end{aligned} \tag{4}$$

where $\mathbf{f}_k(p_{hi}^k)$, $\mathbf{g}_k(p_{hi}^k)$ are constraint vector functions defining a convex feasibility domain and $p_{hi}^k := P\left(\mathbf{u}_h(k)|\mathbf{x}_i(k)\right)$.

For Problem 1 we make the following observations. Minimizing the KL-divergence amounts at minimizing the discrepancy between P^n and Q^n. These pmfs can be obtained by observing different systems and hence the formulation embeds the possibility of synthesizing policies for a given system by using examples collected on a different one.

3. The Discrete fully probabilistic design algorithm

The pseudo-code for the DFPD algorithm is given in Algorithm 1. The algorithm takes as input the probabilistic descriptions $\tilde{P}_X^{k+1}\big(\mathbf{x}(k+1)|\mathbf{x}_i,\mathbf{u}_h\big)$, $\tilde{Q}_X^{k+1}\big(\mathbf{x}(k+1)|\mathbf{x}_i,\mathbf{u}_h\big)$, $\tilde{Q}_U^{k}\big(\mathbf{u}(k)|\mathbf{x}_i\big)$ and the constraints of Problem 1 (optional). The algorithm then outputs the optimal solution to Problem 1[2]. The core of the procedure is a backward recursion that solves, at each iterate, the following

Problem 2 (Local optimization problem) *Find a solution, for each $\mathbf{x}_i \in \mathcal{X}$, to the following*

$$\begin{aligned}\min_{p_{1i},\dots,p_{zi}} \; & d\left(\mathbf{x}_i\right) = p_{1i}\ln p_{1i} + p_{1i}(d_{1i}^x + r_{1i} - a_{1i}) + \ldots + p_{zi}\ln p_{zi} + p_{zi}(d_{zi}^x + r_{zi} - a_{zi}) \\ \textit{s.t.} \; & \sum_{h=1}^{z} p_{hi} = 1 \\ & p_{hi} \geq 0 \; \forall h \\ & \mathbf{f}_k(p_{hi}) \leq \mathbf{0} \\ & \mathbf{g}_k(p_{hi}) = \mathbf{0}\end{aligned} \tag{5}$$

where $p_{hi} = p_{hi}^k$ are the probabilities at the current iterate, $d_{hi}^x = \mathcal{D}_{KL}(\tilde{P}_X^{k+1}||\tilde{Q}_X^{k+1})$, $a_{hi} = \ln\left(Q\left(\mathbf{u}_h(k)|\mathbf{x}_i(k)\right)\right)$ and

$$r_{hi} = \sum_{j=0}^{m-1} P\big(\mathbf{x}_j(k+1)|\mathbf{x}_i(k),\mathbf{u}_h(k)\big)d\big(\mathbf{x}_j(k+1)\big). \tag{6}$$

At each step of the optimization horizon, and for each state $\mathbf{x}_i \in \mathcal{X}$, the algorithm computes the scalars d_{hi}, a_{hi} and r_{hi}, needed for the resolution of Problem 2. We note that, for the k-th step and the i-th state, the computation of r_{hi} requires the recursion of the optimal cost $d(\mathbf{x}(k+1))$ *for all states*. Therefore, at k, each single cost $d(\mathbf{x}_i(k))$ is temporarily stored in the auxiliary variable $c(\mathbf{x}_i)$ in order to be reused at $k-1$. We now give the following

Proposition 1 *Algorithm 1 returns the optimal solution to Problem 1.*

Proof The proof follows the same technical derivations of Gagliardi and Russo (2021) where an analytical solution to Problem 1 is given via a backward recursion and a primal-dual argument. The only difference is that, for numerical convenience, Algorithm 1 tackles, at each k, a reformulation of the problem solved in the paper mentioned above. This reformulation is in fact Problem 2. For completeness, a derivation of this problem is given in Appendix **??**. ■

We also make the following

2. As noted in Gagliardi and Russo (2021), given the convexity of the cost, if the constraints define a convex set, the existence of the solution is guaranteed.

Algorithm 1: Discrete fully probabilistic design algorithm

Inputs: $\tilde{P}_X^{k+1}\big(\mathbf{x}(k+1)|\mathbf{x}_i,\mathbf{u}_h\big)$, $\tilde{Q}_X^{k+1}\big(\mathbf{x}(k+1)|\mathbf{x}_i,\mathbf{u}_h\big)$, $\tilde{Q}_U^{k}\big(\mathbf{u}(k)|\mathbf{x}_i\big)$, constraints of Problem 1 (optional)

Output: solution to Problem 1

Initialize: $d(\mathbf{x}_i) \leftarrow 0\ \forall i = 0,\dots,m-1$

Main loop:

for $k \leftarrow n-1$ *to* 0 **do**

  for ***each*** $\mathbf{x}_i \in \mathcal{X}$ **do**

    $d_{hi}^x \leftarrow \mathcal{D}_{KL}(\tilde{P}_X^{k+1}||\tilde{Q}_X^{k+1})\ \forall h$

    $a_{hi} \leftarrow \ln\Big(\tilde{Q}_U^k(\mathbf{u}_h|\mathbf{x}_i)\Big)\ \forall h$

    $r_{hi} \leftarrow \sum_{x_j\in\mathcal{X}} \tilde{P}_X^{k+1}(\mathbf{x}_j|\mathbf{x}_i,\mathbf{u}_h)d(\mathbf{x}_j)\ \forall h$

    Find $\tilde{P}_U^k$ by solving Problem 2 with the coefficients above

    Store optimal cost $c(\mathbf{x}_i)$

  end

  $d(\mathbf{x}_i) \leftarrow c(\mathbf{x}_i)\ \forall i = 0,\dots,m-1$

end

Remark 2 *Consider a receding horizon set-up with time-invariant constraints and where, at each k, the optimization problem solved in the receding horizon window remains the same. Further, note that Algorithm 1, by construction, finds all the possible (for each state) optimal conditional probability functions $\tilde{P}_U^0$. This implies that, in principle, for this special case, one can find the policy offline. Specifically, once the problem is solved offline, the control input can be simply obtained by sampling, at each k, from $\tilde{P}_U^0$.*

The rationale behind Remark 2 is as follows. Let us assume that t_0 is the current time and Problem 1 is solved via Algorithm 1. In this case, $t = t_0 + k\Delta t$, so that a finite horizon T of n optimization steps is considered. Although the current state $\mathbf{x}(t_0) = \mathbf{x}(k=0)$ is known, Algorithm 1 solves Problem 1 $\forall \mathbf{x}(t_0) \in \mathcal{X}$. Let us name this optimal solution $P(\mathbf{u}(k)|\mathbf{x}(k))$. In a receding horizon setup, $P(\mathbf{u}(0)|\mathbf{x}(0))$ is the control action at $t = t_0$. Now, let us consider a generic time $t_1 > t_0$ at which Problem 1 is solved again with the same strategy, with $t = t_1 + k\Delta t$ and $k = 0,\dots,n-1$ corresponding to the same finite horizon T and with $\mathbf{x}(t_1) \in \mathcal{X}$. Let us name this optimal solution $P_{t_1}(\mathbf{u}(k)|\mathbf{x}(k))$. If the probabilistic models and the constraints are time-invariant, then $P_{t_1}(\mathbf{u}(k)|\mathbf{x}(k)) = P(\mathbf{u}(k)|\mathbf{x}(k))\ \forall k$. As a consequence, in our receding horizon setup, the control action at $t = t_1$ is, again, $P(\mathbf{u}(0)|\mathbf{x}(0))$. Last, let us assume to pick $t_1 = t_i + i\Delta t$, with $i \in \mathbb{N}_0$ and that a control action is taken at every t_1. In view of the above, the optimal control action is $P(\mathbf{u}(0)|\mathbf{x}(0))\ \forall t_1$, as long as $\mathbf{x}(t_1) = \mathbf{x}(k=0) \in \mathcal{X}$, that is true by construction.

Remark 3 *The downside of the approach in Remark 2 is that, if the state space is large, the off-line planning approach is infeasible in any practical situation. In this case, a receding horizon strategy might still be the most effective option, possibly with a set constraint like $\mathbf{x}(0) \in \mathcal{X}_0$, where $\mathcal{X}_0$ is an arbitrarily large neighborhood of the current state $\mathbf{x}_0$. This would require re-executing the receding horizon optimization only when $\mathbf{x}(0)$ gets far enough from $\mathbf{x}_0$.*

4. A proposed design pipeline: from data to control inputs

Before proceeding with the illustration of the pipeline to compute the inputs required by Algorithm 1, we note here that, as for any other control approach that relies solely on the available data, these need to be sufficiently informative. In this paper we do not consider the problem of obtaining sufficiently informative datasets and refer to e.g. (Van Waarde et al., 2020; De Persis and Tesi, 2020; van Waarde et al., 2020; Colin et al., 2020) for recent results on this topic.

The first step of the pipeline is to gather the data, which are then processed to obtain the probability functions required as input by Algorithm 1. The last, optional, step of the pipeline consists in formalizing the constraints for Problem 1.

Data gathering. We refer to the system under control as *target system* and we term as *reference system* the one that is used to collect example data. In what follows, we do not require any knowledge on the (possibly) nonlinear and stochastic dynamics that is generating the data. Also, data for the target and reference system might be generated from different unknown dynamics. Our starting point is the collection of the data to compute $\tilde{Q}_X^k$, $\tilde{P}_X^k$ and $\tilde{Q}_U^k$. Namely, we assume the availability of the following data recorded within the observation window $[0, T]$: the triplets $\{\mathbf{x}^t(k), \mathbf{u}^t(k), \mathbf{x}^t(k+1)\}$ and $\{\mathbf{x}^r(k), \mathbf{u}^r(k), \mathbf{x}^r(k+1)\}$, and the pairs $\{\mathbf{x}^r(k), \mathbf{u}^r(k)\}$, where $\mathbf{x}^t(k)$ and $\mathbf{x}^r(k)$ are the target and reference systems' states and $\mathbf{u}^t(k)$ and $\mathbf{u}^r(k)$ are the target and reference systems' inputs at $t(k)$, respectively. Before illustrating how these data are used, we remark here that for physical applications (modeled via continuous dynamics) the data give a view of a discretized version of the processes. Hence, the probability functions that we obtain implicitly depend on the (discretization) step Δt. From the practical viewpoint, this parameter needs to be in accordance with the duration of the full control cycle.

Remark 4 *For time-invariant, we drop the superscripts in the above probability functions as these are stationary. Due to the effects of the discretization and quantization, stationary probability functions might, in principle, still be obtained from time-varying dynamical processes.*

Quantization. Once the data are obtained, these need to be quantized to obtain discrete sets of the form (2) required by Algorithm 1. We let $\boldsymbol{\Delta}\mathbf{x}$ and $\boldsymbol{\Delta}\mathbf{u}$ be the uniform quantization steps for state and input. Then, each given observation of the state, say $\mathbf{x}$, is mapped onto $\mathbf{x}_j$ defined in (2) if

$$\mathbf{x}_j - \frac{\boldsymbol{\Delta}\mathbf{x}}{2} \leq \mathbf{x} < \mathbf{x}_j + \frac{\boldsymbol{\Delta}\mathbf{x}}{2}. \tag{7}$$

Analogously, a given observed input, say $\mathbf{u}$, is mapped onto $\mathbf{u}_h$ defined in (2) if

$$\mathbf{u}_h - \frac{\boldsymbol{\Delta}\mathbf{u}}{2} \leq \mathbf{u} < \mathbf{u}_h + \frac{\boldsymbol{\Delta}\mathbf{u}}{2}. \tag{8}$$

At the boundaries, $\mathbf{x}$ is mapped onto $\mathbf{x}_0$ when $\mathbf{x} < \mathbf{x}_0 + \Delta\mathbf{x}/2$ and onto $\mathbf{x}_{m-1}$ when $\mathbf{x} \geq \mathbf{x}_{m-1} - \Delta\mathbf{x}/2$. Equivalent considerations hold for the inputs.

Remark 5 *While it is reasonable to assume that the reference and target systems share a common state space (i.e. the two systems should have similar capabilities in order to deliver similar behaviors), inputs can be quite different, depending on the systems' dynamical characteristics. Thus input data might need to be normalized prior to the application of (8).*

Computation of the probability functions. As in e.g. Gagliardi and Russo (2021), we obtain the probability functions by computing the empirical distributions from the data. This can be achieved by defining three counting functions: (i) $c_X : \mathcal{X} \to \mathbb{N}$ counts the occurrences of a given state in the collected reference (or target) system data; (ii) $\mathbf{c}_{X|X,U} : \mathcal{X} \times \mathcal{U} \to \mathbb{N}^m$, for each visited state and selected input in the same state, counts the occurrences of every state in the following time instant in the collected reference (or target) system data; (iii) $\mathbf{c}_{U|X} : \mathcal{X} \to \mathbb{N}^z$, for each visited state, counts the occurrences of every selected input in the collected reference system data. The probabilistic functions for the target thus be computed, for each i and h, as:

$$\tilde{Q}_X(\mathbf{x}|\mathbf{x}_i, \mathbf{u}_h) = \frac{\boldsymbol{\tau}_{X|X,U}(\mathbf{x}_i, \mathbf{u}_h)}{\boldsymbol{\tau}^h_{U|X}(\mathbf{x}_i)}, \quad \tilde{Q}_U(\mathbf{u}|\mathbf{x}_i) = \frac{\boldsymbol{\tau}_{U|X}(\mathbf{x}_i)}{\tau_X(\mathbf{x}_i)}. \tag{9}$$

Since the counting functions are vector functions, the superscript indicates the element in the vector. In the above expressions, we have $\tau_X(\mathbf{x}) := o_s + c_X(\mathbf{x}), \boldsymbol{\tau}_{X|X,U}(\mathbf{x}, \mathbf{u}) := o_n + \mathbf{c}_{X|X,U}(\mathbf{x}, \mathbf{u})$ and $\boldsymbol{\tau}_{U|X}(\mathbf{x}) := o_i + \mathbf{c}_{U|X}(\mathbf{x})$, with o_s, o_n and o_i being small constants (offsets) added in order to avoid divisions by 0 in (9) which would happen when a given event is not seen in the data, as well as the computation of $\log(0)$ for, e.g., the coefficients in Problem 2. Clearly, the same steps can be followed to obtain $\tilde{P}_X(\mathbf{x}|\mathbf{x}_i, \mathbf{u}_h)$ and these are omitted here for brevity.

Remark 6 *A possible choice for the offsets o_s is to set $o_s \leq 1/m$. Once o_s is fixed, all the other offsets must be designed so as to guarantee that probabilities sum to one. Then one needs to set $o_i = o_s/z$ and $o_n = o_i/m$.*

Remark 7 *When an input is never selected in a given state or a state is never visited, the data do not contain any information about $\tilde{Q}_X(\mathbf{x}|\mathbf{x}_i, \mathbf{u}_h)$ and $\tilde{Q}_U(\mathbf{u}|\mathbf{x}_i)$ respectively. In these cases the best we can do is assuming that any state can be reached at the next time step with equal probability by applying $\mathbf{u}_h$ in $\mathbf{x}_i$ and that any input is selected with equal probability when in $\mathbf{x}_i$, respectively. Equations in (9) automatically implement this condition.*

Respecting the range of feasible inputs. Problem (2) accounts for soft and hard constraints imposed through probabilities. Here we discuss about the possibility of manipulating the input domains so as to respect the range of feasible inputs, regardless of the optimization performed by Algorithm 1. Let us assume that the reference and target systems are commanded with inputs $\mathbf{v}_{r,t}$, and that the (symmetrical) ranges of feasible inputs are $-\overline{\mathbf{v}}_{r,t} \leq \mathbf{v}_{r,t} \leq \overline{\mathbf{v}}_{r,t}$, where $\pm\overline{\mathbf{v}}_{r,t}$ represent the boundaries, extracted, for instance, from the plants' datasheet. The inputs of the probabilistic model are the normalized ones, i.e.

$$\mathbf{u}_r = \frac{\mathbf{v}_r}{\overline{\mathbf{v}}_r}, \mathbf{u}_t = \frac{\mathbf{v}_t}{\overline{\mathbf{v}}_t} \tag{10}$$

which implies $-\mathbf{1} \leq \mathbf{u}_{\{r,t\}} \leq \mathbf{1}$, with $\mathbf{1}$ representing the vector of ones of appropriate size.

Once the control policy $\tilde{P}^U_k$ is available, the probabilistic controller reconstructs the target system input by inverting (10), hence setting $\mathbf{u}_0 = -\mathbf{1}, \mathbf{u}_{z-1} = \mathbf{1}$ in (2) ensures that the ranges of feasible inputs are respected without defining explicit constraints in Problem 1.

Adding constraints in the optimization problem. This last step in the pipeline is optional and is only requested if the problem has actuation constraints, beyond respecting the range of feasible inputs. We recall that the constraint functions that the algorithm takes as input are generic and one only needs to verify that the set is convex (recall that convexity of the set guarantees the existence

of an optimal solution to Problem 1). Constraints of practical interest, which can be captured with a proper choice of constraint functions, include moment constraints (Georgiou and Lindquist, 2003; Pavon and Ferrante, 2006; Zhu and Baggio, 2019) and bound constraints (McKinnon and Schoellig, 2019; Nakka et al., 2021). For example, the inequality constraint $\mathbb{E}_{\tilde{P}^k_U}[U^i] - m_i \leq 0$ expresses the fact that the i-th moment of the control input is not larger than m_i. Bound constraints instead formalize the fact that $\mathbb{P}(\mathbf{U}_k \in \bar{\mathcal{U}}_k) \geq 1 - \varepsilon$, where $\bar{\mathcal{U}}_k \subset \mathcal{U}$ is and $\varepsilon \geq 0$. That is, the constraint captures the fact that the control variable belongs to some (e.g. desired) $\bar{\mathcal{U}}_k$ with some desired probability. This constraint can be included in Algorithm 1 as $\mathbb{E}_{\tilde{f}^k_{\mathbf{U}}}\left[\mathbb{1}_{\bar{\mathcal{U}}_k}(\mathbf{U}_k)\right]$.

5. Benchmarking the DFPD algorithm on an inverted pendulum

We now numerically investigate the effectivenes of the DFPD algorithm by using an inverted pendulum with actuation constraints as test-bed. We first describe the environment, then we describe how the inputs to Algorithm 1 are obtained. We finally discuss the numerical results. The code to replicate the results can be found at https://github.com/unisa-acg/discrete-fpd.

The environment. We consider the task of stabilizing a pendulum on its unstable equilibrium. The pendulum we want to stabilize (i.e. the target system) has parameters that are different from the one used to collect the data (i.e. the reference system). Specifically, the parameters of the reference system were as follows: rod length, $l_r = 0.2$ m, mass, $m_r = 0.5$ kg, friction $b_r = 8 \cdot 10^{-5}$ Nm/(deg/s). Instead, for the target pendulum we have $l_t = 0.4$ m, $m_t = 1$ kg, $b_t = 1 \cdot 10^{-5}$ Nm/(deg/s). As usual, the input to the pendulum is the torque applied to its hinged end, i.e. $\mathbf{u} = u$, while the state is $\mathbf{x} = [x_1, x_2]^T$, with x_1 being the angular position and $x_2 = \dot{x}_1$ the angular velocity. Finally, in one of our simulations, we impose the constraint that the target system cannot use a torque that is larger than 50% of the torque capacity, say $\tau_{t,max} = 11.5$ Nm, i.e. $-0.5 \leq \mathbf{u} \leq 0.5$.

Obtaining the inputs to Algorithm 1. Data were obtained by simulating the nonlinear dynamics of the target and the reference system[3]. The dynamics of the two systems were stochastic: zero mean Gaussian noise processes with variances $\sigma_r^2 = 20$ and $\sigma_t^2 = 10$ were additively added to the acceleration of the reference and target dynamics, respectively. Data were generated by making the reference system follow a path that takes it from the stable equilibrium state $\mathbf{x} = [-\pi/2, 0]^T$ to the unstable equilibrium state $\mathbf{x} = [\pi/2, 0]^T$. Examples were generated via a model-based PID controller, simulating a number of control policies that differed in the time law used by the pendulum to reach the target position. The time parametrization was obtained through the phase plane technique, see e.g. Slotine and Yang (1989). We used this technique as it allowed us to formulate the trajectory generation problem for the pendulum with a reduced set of parameters. In fact all the trajectories belong to one of two classes having either one or three switching points in the phase plane, that could be suitably assigned through randomized uniform sampling in a given range. The switching points were then connected through linear interpolation. Figure 1 reports position, velocity and torque signals over 100 simulations, where 30% of trajectories belong to the 1-switching-point class, while the remaining ones belong to the 3-switching-point class. In order to collect data on the state evolution model of the target pendulum, we provided it with open loop torques that excited the system at different configurations. All simulations for the reference and target systems were executed with $\Delta t = 0.01$ s.

3. Mathematical models are only used to generate the data and are not leveraged to compute the control action

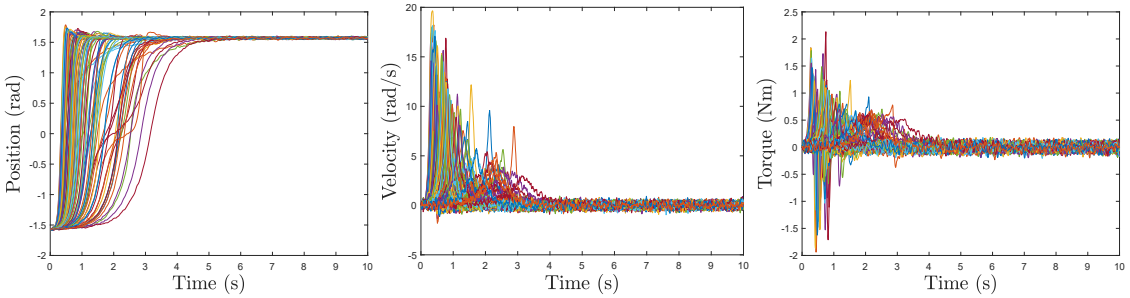

Figure 1: Reference system data over 100 simulations

The generation of the probabilistic model followed the arguments of Section 4. First, the state and input domains were discretized, then the probabilistic models were computed as in (9), assuming time invariance. Discrete state domain boundaries were determined on the basis of the maximum and minimum values observed in the reference and target data, thus $\mathbf{x}_0 = [-1.5872, -1.8107]^T$ and $\mathbf{x}_{m-1} = [1.9583, 19.6537]^T$. The state discretization step $\Delta\mathbf{x}$ was selected from a trade-off between accuracy and computation time. In our numerical experiments we set $\Delta\mathbf{x} = [0.1223, 0.7402]^T$, as this empirically yielded satisfactory results. For the inputs, we observed that the target system is bigger and heavier than the reference one, from which one should expect higher torques. In our implementation, we considered symmetric torque limits and inputs of the reference system were normalized as $u = \tau_r/\tau_{r,max}$, while those of the target system as $u = \tau_t/\tau_{t,max}$, so that u takes the common semantics of an effort capacity. In these expressions τ_r and τ_t are the observed torques of the reference and target system respectively, $\tau_{r,max}$ and $\tau_{t,max}$ are the maximum torques of the reference and target system respectively, as observed from data. Finally we set $\Delta u = 0.0513$ and, in (2), $u_0 = -1$ and $u_{z-1} = 1$, so as to respect the ranges of feasible inputs by construction. Given this set-up, we computed the probability functions $\tilde{Q}_X$, $\tilde{Q}_U$ and $\tilde{P}_X$ needed by Algorithm 1. For the last experiment, as last input to Algorithm 1, we defined a constraint on the torque of the pendulum (see the first paragraph of this section). The constraint was captured via a bound constraint.

Results. We obtained the optimal policy to control the target system from data collected from the reference system via Algorithm 1. In particular, by leveraging the stationarity of the system and of the constraints, we used the observation given in Remark 2 to retrieve the optimal policy. Namely, once the problem was solved offline, we sampled, at each k, the control input from the conditional probability function $\tilde{P}_U^0$. This policy was the one used at run-time to generete the actual torque inputs to the pendulum. Specifically, at each control cycle: (i) the current state (pendulum position and velocity) is read; (ii) quantization is retrieved; (iii) the right histogram for $\tilde{P}_U^0(\mathbf{u}|\mathbf{x}_j)$ is queried from the memory (note indeed that $\tilde{P}_U^0(\mathbf{u}|\mathbf{x}_j)$ is conditioned and hence the histogram depends on the current state of the pendulum). From the histogram, the control input with the highest-probability value was selected and the torque was obtained as $\tau_t = u \cdot \tau_{t,max}$. The discrete FPD optimization was executed with $n = 10$. The pendulum position, velocity and torque obtained with the FPD control policy are reported in blue in Figure 2: the data driven control policy correctly stabilizes the pendulum around the state $\mathbf{x} = [\pi/2, 0]^T$, imitating the behavior of the reference system in Figure 1. In Figure 2 we also report, in red, the result of using the same data-driven control policy on the reference system. The figure shows that, when the policy from the reference system is exported to control the target system, this is not able to stabilize the pendulum on the desired equilibrium.

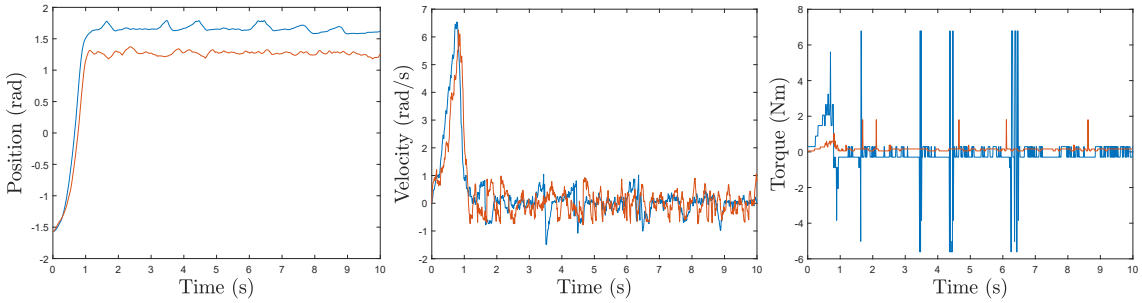

Figure 2: Comparison between data-driven control policy applied to the target (blue) and reference (blue) system

The reason for this is in the fact that the parameters of the two systems were different. In these simulations there were no constraints and our last validation step consists in verifying the ability of the algorithm to stabilize the pendulum even in the presence of the constraints mentioned in the first paragraph of this section. In Figure 3 the behavior of the closed loop system is shown when these constraints were added. The figure clearly shows that DFPD was able to stabilize the pendulum in the presence of these constraints.

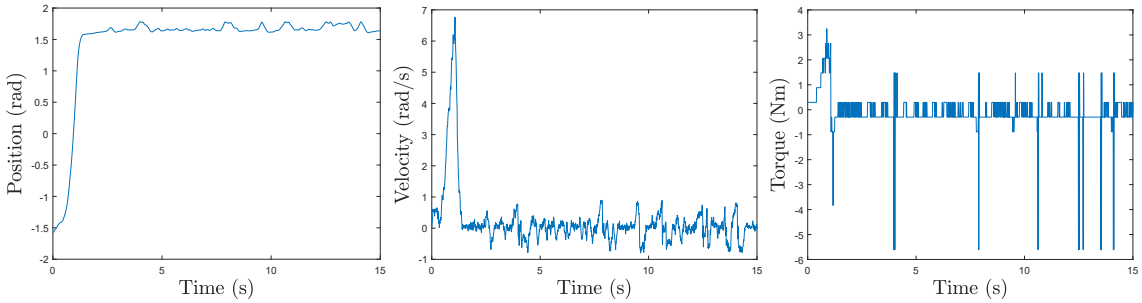

Figure 3: Target pendulum trajectory through data-driven control with restricted torque constraints

6. Conclusions

We considered the problem of designing control policies from example data for constrained systems having a possibly stochastic and nonlinear dynamics. In this context, we introduced the DFPD, a design that expounds an algorithm from Gagliardi and Russo (2021) for the synthesis of control policies from data collected from a system that is different from the one under control, without requiring that the constraints are already fulfilled in the possibly noisy example data. After discussing a number of properties of the numerical design and presenting a pipeline for its use, we benchmarked numerically the DFPD on an example that involves controlling an inverted pendulum. The pendulum was affected by actuation constraints and the demonstration data were collected from a physically different pendulum that does not satisfy these constraints. Our simulations highlight that DFPD is able to stabilize the pendulum directly from the data while satisfy the constraints embedded in the problem formulation.

References

Abed AlRahman Al Makdah, Vishaal Krishnan, and Fabio Pasqualetti. Learning robust feedback policies from demonstrations, 2021. URL https://arxiv.org/abs/2103.16629.

Kévin Colin, Xavier Bombois, Laurent Bako, and Federico Morelli. Data informativity for the open-loop identification of mimo systems in the prediction error framework. *Automatica*, 117: 109000, 2020. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2020.109000.

C. De Persis and P. Tesi. Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2020.

Davide Gagliardi and Giovanni Russo. On a probabilistic approach to synthesize control policies from example datasets. *Automatica*, (in press), 2021. URL https://arxiv.org/pdf/2005.11191.pdf.

Manan S. Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A. Theodorou. Robust model predictive path integral control: Analysis and performance guarantees. *IEEE Robotics and Automation Letters*, 6(2):1423–1430, 2021. doi: 10.1109/LRA.2021.3057563.

Emiland Garrabe and Giovanni Russo. On the design of autonomous agents from multiple data sources. *IEEE Control Systems Letters*, 6:698–703, 2022. doi: 10.1109/LCSYS.2021.3086058.

T.T. Georgiou and A. Lindquist. Kullback-Leibler approximation of spectral density functions. *IEEE Transactions on Information Theory*, 49(11):2910–2917, 2003. doi: 10.1109/TIT.2003.819324.

Peng Guan, Maxim Raginsky, and Rebecca M. Willett. Online Markov Decision processes with Kullback–Leibler control cost. *IEEE Transactions on Automatic Control*, 59(6):1423–1438, 2014. doi: 10.1109/TAC.2014.2301558.

Tatiana V. Guy, Siavash Fakhimi Derakhshan, and Jakub Štěch. Lazy fully probabilistic design: Application potential. In Francesco Belardinelli and Estefanía Argente, editors, *Multi-Agent Systems and Agreement Technologies*, pages 281–291, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01713-2.

M. Hanawal, H. Liu, H. Zhu, and I. Paschalidis. Learning policies for Markov Decision Processes from data. *IEEE Transactions on Automatic Control*, 64:2298–2309, 2019.

Randa Herzallah. Fully probabilistic control for stochastic nonlinear control systems with input dependent noise. *Neural networks*, 63:199–207, 2015. doi: 10.1016/j.neunet.2014.12.004.

Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3 – 35, 2013. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2012.07.014.

H.J. Kappen, Gomez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87:159–182, 2012. doi: 0.1007/s10994-012-5278-7.

M. Kárný. Towards fully probabilistic control design. *Automatica*, 32(12):1719–1722, 1996. doi: 10.1016/s0005-1098(96)80009-4.

M. Kárný and T. V. Guy. Fully probabilistic control design. *Systems & Control Letters*, 55(4): 259–265, 2006. doi: 10.1016/j.sysconle.2005.08.001.

Miroslav Kárný. Towards fully probabilistic control design. *Automatica*, 32(12):1719–1722, dec 1996. doi: 10.1016/s0005-1098(96)80009-4. URL https://doi.org/10.1016%2Fs0005-1098%2896%2980009-4.

Miroslav Kárný and Tatiana V. Guy. Fully probabilistic control design. *Systems & Control Letters*, 55(4):259–265, apr 2006. doi: 10.1016/j.sysconle.2005.08.001. URL https://doi.org/10.1016%2Fj.sysconle.2005.08.001.

Miroslav Kárný and Tomáš Kroupa. Axiomatisation of fully probabilistic design. *Information Sciences*, 186(1):105 – 113, 2012. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2011.09.018.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, mar 1951. doi: 10.1214/aoms/1177729694. URL https://doi.org/10.1214%2Faoms%2F1177729694.

Christopher D. McKinnon and Angela P. Schoellig. Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks. *IEEE Robotics and Automation Letters*, 4(2):2180–2187, 2019. doi: 10.1109/LRA.2019.2901638.

Yashwanth Kumar Nakka, Anqi Liu, Guanya Shi, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Chance-constrained trajectory optimization for safe exploration and learning of nonlinear systems. *IEEE Robotics and Automation Letters*, 6(2):389–396, 2021. doi: 10.1109/LRA.2020.3044033.

M. Pavon and A. Ferrante. On the Georgiou-Lindquist approach to constrained Kullback-Leibler approximation of spectral densities. *IEEE Transactions on Automatic Control*, 51(4):639–644, 2006. doi: 10.1109/TAC.2006.872755.

Bernat Guillen Pegueroles and Giovanni Russo. On robust stability of fully probabilistic control with respect to data-driven model uncertainties. In *2019 18th European Control Conference (ECC)*, pages 2460–2465, 2019. doi: 10.23919/ECC.2019.8795901.

Václav Peterka. *Bayesian approach to system identification*, pages 239–304. Elsevier, 1981. doi: 10.1016/b978-0-08-025683-2.50013-2.

Giovanni Russo. On the crowdsourcing of behaviors for autonomous agents. *IEEE Control Systems Letters*, 5(4):1321–1326, 2021. doi: 10.1109/LCSYS.2020.3034750.

J.-J. E. Slotine and H. S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, 1989. doi: 10.1109/70.88024. URL https://doi.org/10.1109%2F70.88024.

Emanuel Todorov. Linearly-solvable Markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009. ISSN 0027-8424. doi: 10.1073/pnas.0710743106.

Stephen Tu, Alexander Robey, Tingnan Zhang, and Nikolai Matni. On the sample complexity of stability constrained imitation learning, 2021. URL https://arxiv.org/abs/2102.09161.

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

Henk J. van Waarde, Claudio De Persis, M. Kanat Camlibel, and Pietro Tesi. 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.

Bin Zhu and Giacomo Baggio. On the existence of a solution to a spectral estimation problem à la Byrnes–Georgiou–Lindquist. *IEEE Transactions on Automatic Control*, 64(2):820–825, 2019. doi: 10.1109/TAC.2018.2836984.