Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 1584–1590

Embedding the Physics in Black-box Inverse Dynamics Identification: a Comparison Between Gaussian Processes and Neural Networks

Giulio Giacomuzzo * **Niccolo' Turcato** * **Alberto Dalla Libera** * **Ruggero Carli** *

* *Department of Information Engineering, University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy*

(e-mail: giacomuzzo@dei.unipd.it, turcatonic@dei.unipd.it, alberto.dallalibera@unipd.it, carlirug@dei.unipd.it).

Abstract: In recent years, black-box estimators for robot inverse dynamics have drawn the attention of the robotics community. This paper compares two recent black-box approaches that try to improve generalization and data efficiency by embedding the physical laws governing the system dynamics in two different ways. The so-called Deep Lagrangian Networks (DeLaNs) impose the structure of the Lagrangian equations but do not constrain the basis functions used to model the dynamics. Instead, the Gaussian process model based on the recently introduced Geometrically Inspired Polynomial (GIP) kernel constrains the basis functions of the regression problem to a physically inspired finite-dimensional space but does not force structural properties to be guaranteed. We carried out extensive experiments both on simulated and real manipulators with increasing degrees of freedom (DOF). Our results show that: (i) the accuracy of the DeLaNs model deteriorates much more rapidly than the one of the GIP kernel model with the DOF increase. (ii) the GIP kernel model better estimates the different components of the dynamics, namely, the inertial, Coriolis, and gravitational torques, despite not directly imposing structural properties.

Keywords: Learning for control; Machine learning; Nonlinear system identification;

1. INTRODUCTION

Accurate models of the inverse dynamics map, which relates joint trajectories to applied torques, are fundamental for model based control in a large variety of robotics applications, ranging from high-precision trajectory tracking (Khosla and Kanade, 1988; Siciliano et al., 2009; Dalla Libera et al., 2021a) to detection and estimation of contact forces in tasks requiring the interaction with the environment (Dalla Libera et al., 2019; Haddadin et al., 2017; De Santis et al., 2008). The derivation of reliable inverse dynamics models is still an active field of research. Traditional parametric model-based methods derive inverse dynamics models from first order principle of the physics, under the assumption of rigid body dynamics (Sousa and Cortesao, 2014). However, they require a precise knowledge of physical parameters and they could result rather inaccurate in presence of parameter uncertainties or unconsidered dynamics, such as complex frictions or flexibilities (Kwon et al., 2021).

In the past years, black-box approaches drew the attention of the robotics community. Among them, the two main lines of research rely on Neural Networks (Goodfellow et al., 2016) and Gaussian process Regression (GPR) (Rasmussen, 2003). Within these methods, inverse dynamics models are derived directly from experimental data, without requiring deep knowledge on the underlying physical system. This flexibility, however, carries the risk of being relatively local in nature and data-inefficient: learned models could generalize only within a neighborhood of the training trajectories and require huge amounts of data to be trained. This behavior becomes more and more evident as the degrees of freedoms (DOF) increase, limiting the applicability of black-box models in practice.

Several solutions were proposed to overcome the aforementioned limitations (Polydoros et al., 2015; Rueckert et al., 2017; Romeres et al., 2020; Rezaei-Shoshtari et al., 2019). A promising class of solutions proposes to embed insights from physics as a model prior (Dalla Libera and Carli, 2019; Lutter et al., 2019b; Nguyen-Tuong and Peters, 2010; Camoriano et al., 2016). Instead of learning the inverse dynamics in a completely unstructured manner, which makes the problem unnecessarily hard, geometrical and physical properties are exploited to improve generalization and data efficiency.

In the context of neural networks, notable results have been obtained by the so called *Deep Lagrangian Networks (DeLaNs)*, presented in (Lutter et al., 2019b). DeLaNs aim at improving generalization and data efficiency by constraining the network structure. Two feed-forward networks approximate, respectively, the inertia matrix and

10.1016/j.ifacol.2023.10.1858

the potential energy. Then the inverse dynamics model is obtained by combining these two networks and imposing the Lagrangian Mechanics equations. In this way, DeLaNs derive a model that respects physical properties, such as the energy conservation and the symmetry and positiveness of the inertia matrix. The network parameters are optimized in a typical deep learning fashion, minimizing the squared residual of the predicted torques.

In the context of GPR, instead, prior information can be encoded designing an appropriate kernel function. In this sense, inspired by the property of the inverse dynamics map of being a polynomial function in an augmented input space, the authors in (Dalla Libera and Carli, 2019) proposed a black-box polynomial kernel named *Geometrically Inspired Polynomial (GIP)* kernel, based on a novel multiplicative kernel (Dalla Libera et al., 2020). This kernel aims at improving generalization and data efficiency by constraining the regression problem into a finite dimensional space that contains the Lagrangian equations.

The two aforementioned techniques represent two different philosophies of embedding a physical prior in black-box models. DeLaNs impose that the network structure reflects the one of the Lagrangian equations, but use general basis functions to model the inertia matrix and the potential energy. Instead, the GIP kernel constraints the basis functions, but does not guarantee structural properties. In this paper, we compare these two methods investigating both their prediction performance and the structural properties they induce. Our contribution is threefold. First,we review the GIP kernel and DeLaNs highlighting the different philosophies behind the two approaches. Second, we test the prediction capabilities and the structural properties induced by the two methods by comparing their overall estimation accuracy and their capabilities of discriminating the inertial, Coriolis and gravitational components of the inverse dynamics map. Third, we investigate the performance decay of the two methods at the increase of the DOF. We compared the two methods through numerical and real experiments involving manipulators of increasing dimension, from 3 up to 6 DOF. Results show that the GPR model with GIP kernel outperforms DeLaNs as regards the overall accuracy. The gap between the two approaches becomes more and more consistent increasing the manipulator DOF: performance of the DeLaNs model deteriorates rapidly as the number of DOF increases, while the GIP kernel model returns reliable estimates also in the 6 DOF setup. Surprisingly, the GIP kernel model proved to be more performing than DeLaNs also when considering the estimation of the different torques components, despite DeLaNs imposes structural properties.

2. ROBOT INVERSE DYNAMICS

In this section, we provide the dynamics model of a robotic arm under the rigid body assumption. Consider a robotic arm with $n+1$ links, connected by n joints (each joint can be either revolute or prismatic) and let $\boldsymbol{q} \in \mathbb{R}^n$ be the vector of joint positions. The inverse dynamics is defined as the function mapping the robot state $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{3n}$ into the vector of generalized torques $\boldsymbol{\tau} \in \mathbb{R}^n$. Under the rigid body assumption, the robot dynamics can be derived from the Lagrangian mechanics. Let the Lagrangian be the difference between kinetic energy T and potential energy V, namely $\mathcal{L} = T - V$. Then, the system dynamics satisfies the Euler Lagrange equation, expressed as

$$\boldsymbol{\tau} = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}. \tag{1}$$

The kinetic energy has the form $T = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}$, where $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the symmetric, positive definite inertia matrix. Substituting $\mathcal{L}$ and T in (1) yields to the dynamics equation

$$\boldsymbol{\tau} = \underbrace{M(\boldsymbol{q})\ddot{\boldsymbol{q}}}_{\boldsymbol{m}(\boldsymbol{q},\ddot{\boldsymbol{q}})} + \underbrace{C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}}_{\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})} + \underbrace{\frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}}}_{\boldsymbol{g}(\boldsymbol{q})}, \tag{2}$$

where

$$C(\boldsymbol{q},\dot{\boldsymbol{q}}) = \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\left(\dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right)^T \tag{3}$$

is the Coriolis matrix. Moreover, $\boldsymbol{m}(\boldsymbol{q},\ddot{\boldsymbol{q}})$ represents the inertial torque, $\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})$ accounts for the Coriolis and centripetal torques while $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ is the vector accounting for the gravity contribution. For a more detailed derivation of (2) refer to (Siciliano et al., 2009). In the context of black-box inverse dynamics estimation, the torque map in (2) is modeled as a generic unknown function, namely $\boldsymbol{\tau} = f(\boldsymbol{x})$, and it is estimated from experimental data.

3. GAUSSIAN PROCESSES FOR ROBOT INVERSE DYNAMICS

In GPR the inverse dynamics map is learnt directly from data relying on a probabilistic framework. In particular, each joint torque is modeled as a single Gaussian Process (GP) which is assumed to be conditionally independent of the other given the robot state $\boldsymbol{x}$. Considering the generic i-th joint, let $\mathcal{D}^{(i)} = \{X, \boldsymbol{y}^{(i)}\}$ be a set of N input-output observations, where $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ contains the N robot states and $\boldsymbol{y}^{(i)} \in \mathbb{R}^N$ is the vector collecting the corresponding N noisy measurements of the i-th torque component $\tau^{(i)}$. Then the vector $\boldsymbol{y}^{(i)}$ is assumed to be generated by the following probabilistic model

$$\boldsymbol{y}^{(i)} = \begin{bmatrix} f^{(i)}(\boldsymbol{x}_1) \\ \vdots \\ f^{(i)}(\boldsymbol{x}_N) \end{bmatrix} + \begin{bmatrix} w_1^{(i)} \\ \vdots \\ w_N^{(i)} \end{bmatrix} = \boldsymbol{f}^{(i)}(X) + \boldsymbol{w}^{(i)}, \tag{4}$$

where $\boldsymbol{w}^{(i)}$ is i.i.d Gaussian noise with standard deviation σ_i, while $f^{(i)}(\cdot) : \mathbb{R}^{3n} \to \mathbb{R}$ is an unknown function, modeled a priori as a zero-mean GP, namely $\boldsymbol{f}^{(i)}(X) \sim \mathcal{N}(0, \mathbb{K}^{(i)})$. The elements of the covariance matrix $\mathbb{K}^{(i)}$, named *kernel matrix*, are determined by a kernel function $k^{(i)}(\cdot,\cdot)$. Specifically, the element of $\mathbb{K}^{(i)}$ in position (h,j) is $k^{(i)}(\boldsymbol{x}_h, \boldsymbol{x}_j)$. As illustrated in (Rasmussen, 2003), given the dataset $\mathcal{D}^{(i)}$ and a general input location $\boldsymbol{x}_*$, the posterior distribution of $f^{(i)}(x_*)$ is Gaussian, with mean

$$\hat{f}_*^{(i)} = \boldsymbol{k}_*^{(i)\,T}\boldsymbol{\alpha}^{(i)}$$

and covariance

$$\mathbb{V}[f_*^{(i)}] = k^{(i)}(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^{(i)\,T}(\mathbb{K}^{(i)} + \sigma_i^2\, I)^{-1}\boldsymbol{k}_*^{(i)},$$

where

$$\boldsymbol{k}_*^{(i)} = [k^{(i)}(\boldsymbol{x}_*, \boldsymbol{x}_1), \dots, k^{(i)}(\boldsymbol{x}_*, \boldsymbol{x}_N)]^T,$$
$$\boldsymbol{\alpha}^{(i)} = (\mathbb{K}^{(i)} + \sigma_i^2 I)^{-1} \boldsymbol{y}^{(i)}.$$

Since the posterior distribution of $f^{(i)}(x_*)$ is Gaussian, its maximum a posteriori estimator coincides with its mean $\hat{f}_*^{(i)}$. In the GPR framework the kernel function plays a fundamental role. Indeed, it determines the correlation between samples of $f(\cdot)$ on different input locations, and it encodes prior assumptions on $f(\cdot)$, e.g., its smoothness. An alternative explanation of the kernel role can be derived linking GPR and regression problems in Reproducing Kernel Hilbert Spaces (RKHS). Each positive definite kernel is related to a unique RKHS, which turns out to be the hypothesis space where $f(\cdot)$ is searched. For these reasons, the whole complexity of GPR reduces to the choice of the correct kernel function for the problem at hand.

3.1 Geometrically Inspired Polynomial Kernel (GIP)

The GIP kernel is based on the fact that the dynamics equations in (2) are polynomial functions with respect to the elements of a suitable transformation of $\boldsymbol{x}$. In the following, for completeness, we review the GIP kernel structure. Let N_r and N_p be the number of revolute and prismatic joints, respectively, where $N_r + N_p = n$, and let $I_r = \{i_{r_1}, \dots, i_{r_{N_r}}\}$ and $I_p = \left\{i_{p_1}, \dots, i_{p_{N_p}}\right\}$ be the sets containing, respectively, the revolute and prismatic joints indexes. Accordingly, we introduce the vectors

$$\boldsymbol{q}_c = \left[\cos\left(q_{i_{r_1}}\right), \dots, \cos\left(q_{i_{r_{N_r}}}\right)\right] \in \mathbb{R}^{N_r},$$
$$\boldsymbol{q}_s = \left[\sin\left(q_{i_{r_1}}\right), \dots, \sin\left(q_{i_{r_{N_r}}}\right)\right] \in \mathbb{R}^{N_r},$$
$$\boldsymbol{q}_p = \left[q_{i_{p_1}}, \dots, q_{i_{p_{N_p}}}\right] \in \mathbb{R}^{N_p}.$$

In the following, we denote by q_{c_b} the element in $\boldsymbol{q}_c$ associated to joint i_{r_b}, i.e. $q_{c_b} = \cos(q_{i_{r_b}})$ (analogous definitions hold for q_{s_b} and q_{p_b}). For later convenience we define also $\boldsymbol{q}_{cs} \in \mathbb{R}^{2N_r}$, the vector obtained concatenating $\boldsymbol{q}_c$ and $\boldsymbol{q}_s$. Finally, we introduce the vector $\dot{\boldsymbol{q}}_v$ which collects the elements of the set $\{\dot{q}_i \dot{q}_j,\, 1 \le i \le n\,,\, i \le j \le n\}$, that is, the set containing all the possible pairwise products of components of $\dot{\boldsymbol{q}}$. Observe that $\dot{\boldsymbol{q}}_v \in \mathbb{R}^{n(n+1)/2}$.

Interestingly, in (Dalla Libera and Carli, 2019) it has been shown that each component of the vector of generalized torques τ is a polynomial function defined over the elements of $\bar{\boldsymbol{x}} = \left[\boldsymbol{q}_c, \boldsymbol{q}_s, \boldsymbol{q}_p, \dot{\boldsymbol{q}}_v, \ddot{\boldsymbol{q}}\right]$, of degree not greater than $2n+1$, such that: (i) each element of $\boldsymbol{q}_c$, $\boldsymbol{q}_s$ and $\boldsymbol{q}_p$ appear with degree not greater than 2; (ii) each element of $\dot{\boldsymbol{q}}_v$ and $\ddot{\boldsymbol{q}}$ appear with degree not greater than 1; (iii) in any monomial of the polynomial, the sum of the q_{c_b} and q_{s_b} degrees is equal or lower than two, that is,

$$deg\left(q_{c_b}\right) + deg\left(q_{s_b}\right) \le 2\ .$$

To properly embed the previous properties into the GIP kernel, the authors in (Dalla Libera and Carli, 2019) relies on the use of the multiplicative kernel (denoted as MPK) introduced in (Dalla Libera et al., 2021b). Specifically the MPK kernel of order p is defined as the product of p linear kernels,

$$k^{(p)}_{\text{MPK}}\left(\boldsymbol{x}_h, \boldsymbol{x}_j\right) = \prod_{s=1}^{p} \left(\sigma_s^2 + \boldsymbol{x}_h^T \Sigma_s \boldsymbol{x}_j\right), \tag{5}$$

where the $\Sigma_s \in \mathbb{R}^{d \times d}$ are distinct diagonal matrices. The diagonal elements, together with the parameters σ_s^2, compose the hyperparameters set of the MPK kernel. In the following, we use the notation $\bar{\boldsymbol{x}}_i$ to indicate that the elements of the vector $\bar{\boldsymbol{x}}$ refers to information contained in state $\boldsymbol{x}_i$ and similarly for $\boldsymbol{q}_{cs,i}$, $\boldsymbol{q}_{p,i}$, $\boldsymbol{q}_{av,i}$, $\boldsymbol{q}_{cs_b,i}$. To comply with the constraints on the maximum degree of each term as above reported, the GIP kernel is given by the product of $N_r + N_p + 1$ MPK kernels where

- N_r kernels have $p = 2$ and each of them is defined on a 2-dimensional input space given by $\boldsymbol{q}_{cs_b} = [q_{c_b}, q_{s_b}]$, with $b \in I_r$;
- N_p kernels have $p = 2$ and each of them is defined on a 1-dimensional input which is one of the $\boldsymbol{q}_p$ components;
- a single kernel with $p = 1$ defined on the input vector $\boldsymbol{q}_{av} = [\ddot{\boldsymbol{q}}\,, \dot{\boldsymbol{q}}_v]$.

The resulting kernel for the i-th component of the vector of generalized torques τ is

$$k^{(i)}\left(\bar{\boldsymbol{x}}_h, \bar{\boldsymbol{x}}_j\right) = \tag{6}$$
$$k_{cs}\left(\boldsymbol{q}_{cs,h}, \boldsymbol{q}_{cs,j}\right) \times k_p\left(\boldsymbol{q}_{p,h}, \boldsymbol{q}_{p,j}\right) \times k^{(1)}_{\text{MPK}}\left(\boldsymbol{q}_{av,h}, \boldsymbol{q}_{av,j}\right),$$

with

$$k_{cs}\left(\boldsymbol{q}_{cs,h}, \boldsymbol{q}_{cs,j}\right) = \prod_{b=1}^{N_r} k^{(2)}_{\text{MPK}}\left(\boldsymbol{q}_{cs_b,h}, \boldsymbol{q}_{cs_b,j}\right),$$
$$k_p\left(\boldsymbol{q}_{p,h}, \boldsymbol{q}_{p,j}\right) = \prod_{b=1}^{N_p} k^{(2)}_{\text{MPK}}\left(q_{p_b,h}, q_{p_b,j}\right).$$

The GIP kernel induces a finite-dimensional RKHS containing the dynamics in (2). As a consequence, the regression task is restricted to a limited hypothesis space, specifically tailored to the inverse dynamics problem, leading to higher accuracy and better data efficiency.

4. DEEP LAGRANGIAN NETWORKS (DELANS)

Deep Neural Networks are function approximation machines that have become ubiquitous in the recent years due to their nice property of being universal approximators (Goodfellow et al., 2016). DeLaNs aim at improving generalization and data efficiency in the context of inverse dynamics estimation by encoding known physical properties of Lagrangian systems. In particular, DeLaNs model the potential energy $V(\boldsymbol{q})$ and the inertia matrix $M(\boldsymbol{q})$ as feed-forward neural networks. In order to ensure the symmetry of the inertia matrix, rather than modeling $M(\boldsymbol{q})$ directly, DeLaNs exploit the Cholesky decomposition $M(\boldsymbol{q}) = L(\boldsymbol{q})L(\boldsymbol{q})^T$ and model only the lower triangular matrix $L(\boldsymbol{q})$. Thus, $M(\boldsymbol{q})$ and $V(\boldsymbol{q})$ are estimated as $\hat{M}(\boldsymbol{q}) = \hat{L}(\boldsymbol{q}, \phi)\hat{L}(\boldsymbol{q}, \phi)^T$,$\hat{V}(\boldsymbol{q}) = \hat{V}(\boldsymbol{q}; \psi)$, where $\hat{\cdot}$ represents an approximation, while ϕ and ψ denote the network parameters. In order to ensure the positive definiteness of $\hat{M}(\boldsymbol{q})$, the diagonal l_D and off-diagonal l_O elements of $\hat{L}(\boldsymbol{q})$ are learned separately. Diagonal elements are obtained using a non-negative activation function such as ReLu. Off-diagonal elements, instead, are obtained with a linear activation function. Moreover, a positive scalar

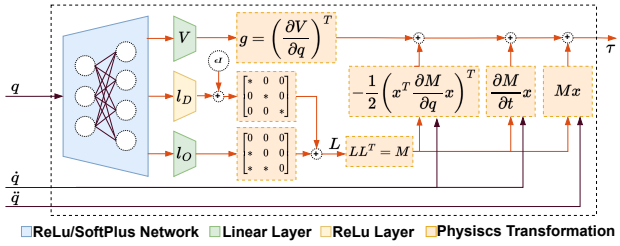


Fig. 1. The computational graph of the DeLaN.

ϵ is added to the diagonal elements, which forces the eigenvalues of the inertia matrix to be positive. Since both L and V rely on the same physical system, the network parameters between them can be shared. In this case, l_D, l_O and V are obtained using three heads with different activation functions. The corresponding inverse dynamics map $\hat{\boldsymbol{\tau}}$ can be obtained from (1) as

$$\hat{\boldsymbol{\tau}} = \hat{M}\ddot{\boldsymbol{q}} + \hat{\dot{M}}\dot{\boldsymbol{q}} - \frac{1}{2}\left(\dot{\boldsymbol{q}}^T \frac{\partial \hat{M}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}\right)^T + \frac{\partial \hat{V}}{\partial \boldsymbol{q}}, \tag{7}$$

where the dependency of $\hat{M}$ and $\hat{V}$ on $\boldsymbol{q}$, ϕ and ψ has been omitted. DeLaNs architecture is shown in Figure 1. The network parameters are learned by minimizing the squared residual of the Euler-Lagrange differential equations, which leads to

$$\phi^*, \psi^* = \arg\min_{\phi,\psi} \|\hat{\boldsymbol{\tau}} - \boldsymbol{\tau}\|^2_{\mathbf{W}_\tau} + \lambda\Omega(\phi, \psi), \tag{8}$$

where $\|\cdot\|_{\mathbf{W}}$ denotes a weighted norm with weight matrix $\mathbf{W}$. The regularization term $\Omega(\phi, \psi)$, instead, represents the L_2-norm of the network parameters and it is necessary due to the ill-posedness of the non regularized problem. The Euler Lagrange equation (1), indeed, is invariant to linear transformations and hence the Lagrangian satisfying (1) is not unique. The addition of a regularization term helps obtaining a unique solution. Note that the optimization problem in (8) imposes the learned model to respect the Lagrangian Mechanics. Besides that, as suggested in (Lutter et al., 2019a), it could be interesting to impose also the energy conservation property. This can be done by adding to the objective function of (8) the residual of the estimated instantaneous change in energy. To this aim, note that total instantaneous change in energy for a Lagrangian system without friction equals the work done by the actuators, i.e. $\dot{E} = \dot{\boldsymbol{q}}^T\boldsymbol{\tau}$. The estimated instantaneous energy change, instead, is given by

$$\hat{\dot{E}} = \dot{\boldsymbol{q}}^T \hat{M}\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \hat{\dot{M}}\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \frac{\partial \hat{V}}{\partial \boldsymbol{q}}. \tag{9}$$

Including also the energy conservation in (8) leads to

$$\phi^*, \psi^* = \arg\min_{\phi,\psi} \|\hat{\boldsymbol{\tau}} - \boldsymbol{\tau}\|^2_{\mathbf{W}_\tau} + \|\hat{\dot{E}} - \dot{E}\|^2_{\mathbf{W}_E} + \lambda\Omega(\phi, \psi). \tag{10}$$

The resulting optimization problem is solved using standard gradient-based optimization techniques with back-propagation. We refer to (Lutter et al., 2019b) for a deeper explanation on how the derivatives involved in (7) and (9) can be efficiently computed.

Remark: DeLaNs propose to model in a black-box fashion only the system energy, rather than the entire inverse dynamics map, and then derive the inverse dynamics imposing the Lagrangian mechanics equations. Moreover, the structure of the neural networks used to model the system energy is designed to impose known physical properties, such as the positiveness and symmetry of the inertia matrix. This approach allows obtaining a physically consistent model, which assures energy conservation and can be used to describe both the forward and the inverse dynamics, being the inertia matrix non-singular by design. However, no assumptions are made on the basis functions exploited to model the system energy. As a result, the regression problem is defined in a very general search space. In contrast, the GPR model with GIP kernel does not impose the structure of the inverse dynamics, but properly selects the basis functions used to model the unknown torque, which restricts the problem to a finite dimensional space where the inverse dynamics is contained. The resulting model is not guaranteed to be physically consistent, but has high data efficiency and generalization capabilities (Dalla Libera and Carli, 2019; Giacomuzzo et al., 2022).

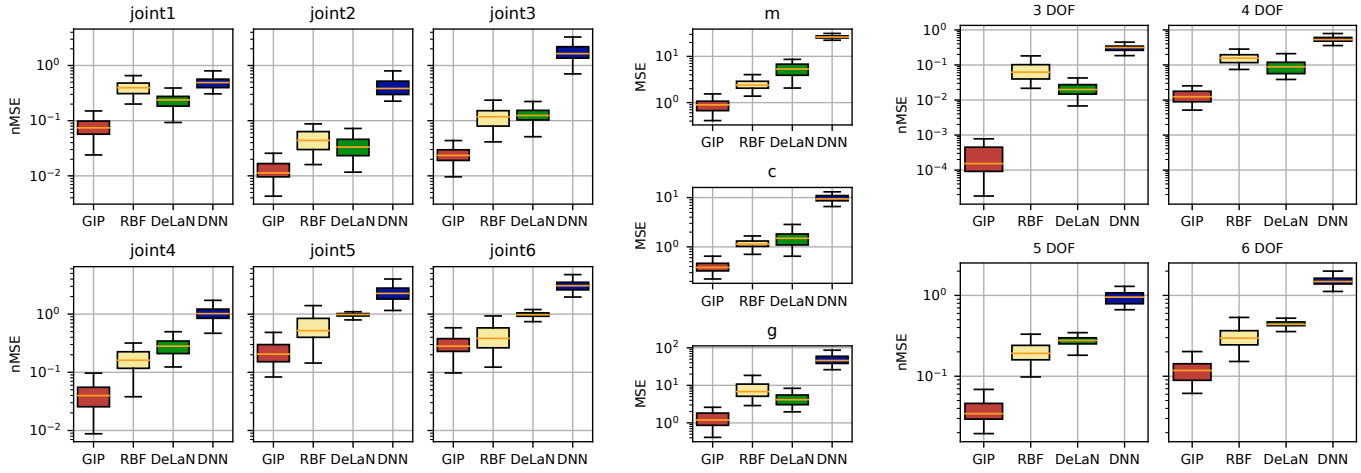
5. TORQUE DECOMPOSITION ESTIMATION

Besides the overall torque, in several practical applications we are interested in computing also its single components. In this section we describe a possible way to obtain the different torque contributions from a black-box estimator of (2). In particular, let $\hat{f}(\cdot)$ be the outcome of the estimator representing the overall torque. We are interested in computing the corresponding decomposition, namely the inertial torque $\hat{\boldsymbol{m}}(\boldsymbol{q}, \ddot{\boldsymbol{q}})$, the coriolis and centripetal torque $\hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the gravitation torque $\hat{\boldsymbol{g}}(\boldsymbol{q})$, which can be evaluated as follows:

- *Gravitational contribution* $\boldsymbol{g}(\boldsymbol{q})$ accounts for all terms that depend only on $\boldsymbol{q}$, as shown in (2). We can thus obtain the estimate of $\boldsymbol{g}(\boldsymbol{q})$ by setting $\dot{\boldsymbol{q}} = 0$, $\ddot{\boldsymbol{q}} = 0$, which leads to $\hat{\boldsymbol{g}}(\boldsymbol{q}) = \hat{f}(\boldsymbol{q}, 0, 0)$.
- *Inertial contribution* $\boldsymbol{m}(\boldsymbol{q}, \ddot{\boldsymbol{q}})$ can be obtained from (2) by setting $\dot{\boldsymbol{q}} = 0$ and then subtracting the gravity contribution previously computed, namely $\hat{\boldsymbol{m}}(\boldsymbol{q}, \ddot{\boldsymbol{q}}) = \hat{f}(\boldsymbol{q}, 0, \ddot{\boldsymbol{q}}) - \hat{f}(\boldsymbol{q}, 0, 0)$.
- *Coriolis and centripetal contribution* $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be obtained setting $\ddot{\boldsymbol{q}} = 0$ and then subtracting the gravity term, namely $\hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \hat{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, 0) - \hat{f}(\boldsymbol{q}, 0, 0)$.

6. EXPERIMENTS

To assess the performance of the approaches we illustrated, we performed extended experiments on simulated and real setups. In both the cases, we considered a Franka Emika Panda robot, which is a 7 DOF collaborative manipulator with only revolute joints. The experiments presented in this section aim to investigate three aspects: (i) the generalization, namely the prediction accuracy on unseen trajectories; (ii) the accuracy of the torque decomposition; (iii) the performance degradation at the increase of the complexity of the manipulator. To this aim we considered 4 versions of the Panda robot, each one obtained by locking the last $7-k$ joints while leaving the first k joints free to move, with $k \in [3, 6]$. Within all the considered experiments we compare the performance of the GIP and DeLaN estimators. For the seek of completeness, we include as baselines the performance of a feedforward deep neural network (DNN) and a vanilla GP with *Radial Basis Function* (RBF), kernel which is defined as



(a) Simulated Panda robot with last joint locked (b) Decomposition (c) Analysis of performance with increasing DOF

Fig. 2. Summary of the results obtained on the simulation experiments described in section 6.1. (a) shows the boxplots of the torque nMSE on each single joint, for the 6 DOF setup. (b) shows the boxplots of the MSE on torque decomposition averaged over all the joints, for the 6 DOF setup. (c) shows the boxplots of the torque nMSE averaged over all the joints, for the four different configurations of the Panda robot considered (from 3 to 6 DOFs).

$$k_{RBF}(\boldsymbol{x}, \boldsymbol{x}') = \lambda e^{-\|\boldsymbol{x}-\boldsymbol{x}'\|^2_{\Sigma^{-1}}}, \tag{11}$$

where λ and Σ are the kernel hyperparameters. In particular, λ is a scaling factor, while Σ is a positive definite matrix, often chosen as diagonal, with the diagonal elements named lengthscales. Regarding the DeLaN model, we considered the same architecture introduced in (Lutter and Peters, 2021). In particular, the network involved to learn the mass matrix and potential energy is composed by two hidden layers of 64 neurons each and SoftPlus as activation function. The feedforward DNN, instead, is composed by two hidden layers of 512 neurons each and Sigmoid as activation function. DeLaN models are trained minimizing the loss function in (10), while the feedforward DNN is trained minimizing the torque Mean Squared Error (MSE). The hyperparameters of the GP kernels, instead, are optimized by marginal likelihood maximization (Rasmussen, 2003). All the optimization problems are solved applying Stochastic Gradient Descent using Adam as optimizer (Kingma and Ba, 2014). In order to speed up the algebraic computations involved, all the presented estimators are implemented with the Python library PyTorch (Paszke et al., 2019). The section is organized as follows: in subsections 6.1 and 6.2 we present the results obtained in simulation and with the real Panda robot, respectively.

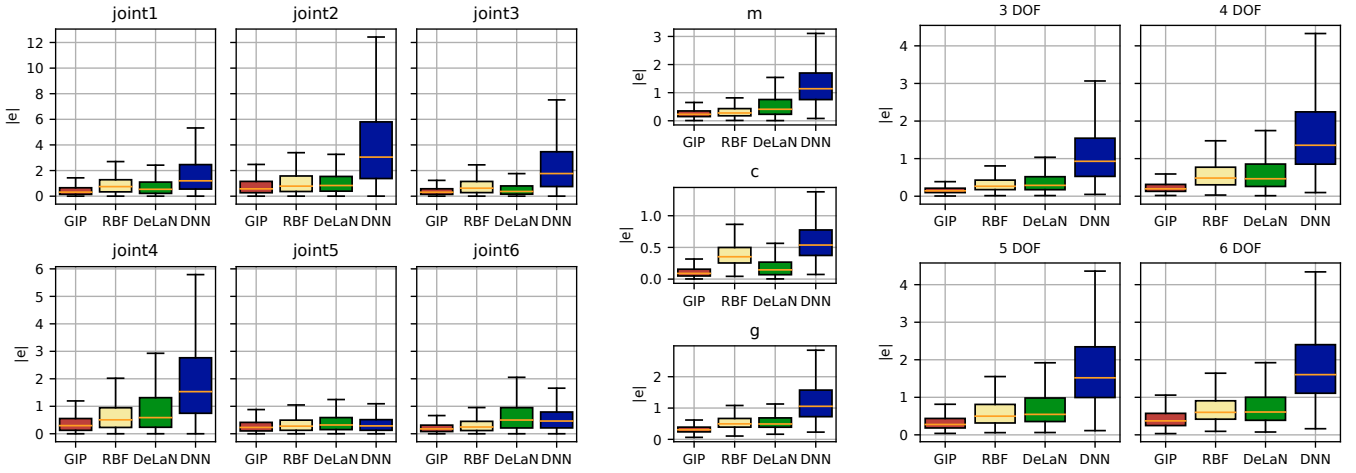
6.1 Simulation experiment

Experiments have been carried out in SymPyBotics (Sousa, 2014), simulating the 4 versions of the Panda robot. In order to obtain statistically relevant results, for each version of the robot, we tested the regression performance of the two families of models with a Monte Carlo (MC) analysis consisting of 50 experiments. Each experiment consists in generating one training trajectory and 50 test trajectories. The training and test trajectories are obtained by imposing to each joint a random position path, composed of the sum of 50 sinusoids. In details, the reference position of the i-th joint is defined as

$$q_i^r(t) = \sum_{l=1}^{50} \frac{a}{\omega_f\, l}\, sin(\omega_f\, l\, t) - \frac{b}{\omega_f\, l}\, cos(\omega_f\, l\, t), \tag{12}$$

with $\omega_f = 0.02\,\text{rad/s}$, while a and b are sampled from a uniform distribution ranging in $[-c,\ c]$, with c chosen in order to respect the robot limits on joint position, velocity and acceleration. A zero-mean Gaussian noise with standard deviation 0.01 N m was added to the resulting torques of the training dataset, in order to simulate measurement noise. All the generated datasets are composed of 500 samples. Each inverse dynamics estimator is trained on the noisy training dataset and then tested on each noiseless test datasets. We compared the prediction performance evaluating the normalized Mean Squared Error (nMSE) of the estimated torques on the test sets, which is the ratio between the mean squared error and the variance of the corresponding joint torque. On the same test trajectories, we compare also the ability to reconstruct the different torque components, namely the inertia torque $\boldsymbol{m}(\boldsymbol{q}, \ddot{\boldsymbol{q}})$, the Coriolis and centripetal torque $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the gravitational torque $\boldsymbol{g}(\boldsymbol{q})$, in terms of Mean Squared Error (MSE) [1] . Figures 2a and 2b report respectively the nMSE distributions on the torque estimates for each joint, and the joints average MSE distribution on the torques decomposition estimates for the 6 DOF simulated robot. Figure 2c instead shows the average torque nMSE over all the joints with the increase of DOF for the manipulator. As expected, one can see that, in general, the GIP kernel outperforms the RBF kernel, while DeLaN outperforms the feedforward DNN. That is, within each family of models, the approach that exploits knowledge of physics yields better accuracy both estimation of torque and reconstruction of its components. It is worth noticing that, while the overall torque model is learned in a supervised fashion, the measurements of the single torque components are not exploited during the training phase, and thus the decomposition is learned completely unsupervised. The

[1] With the considered robot structure, the gravity term of the first joint is identically null, which prevents the use of the nMSE score.



(a) Real Panda robot with last joint locked (b) Decomposition (c) Analysis of performance with increasing DOF

Fig. 3. Summary of the results obtained on the real experiments described in section 6.2. (a) boxplots of the absolute torque error on each single joint, for the 6 DOF setup. (b) boxplots of the absulute error on torque decomposition averaged over all the joints, for the 6 DOF setup. (c) boxplots of the absolute torque prediction error averaged over all the joints, for the four different configurations of the Panda robot considered (from 3 to 6 DOFs).

better performance of GIP kernel suggests that it induces, in the resulting model, better structural properties. This intuition is confirmed also considering the evolution of the normalized errors at the rise of the robot complexity. If the number DOFs are increased, indeed, the GIP based estimator remains reasonably accurate, while the performance of DeLaN model deteriorates.

6.2 Real experiment

We tested the considered approaches on the physical Panda robot, using the ROS (Koubâa et al., 2017) interface provided by Franka Emika. We collected a training and test trajectory for the 4 versions of the manipulator with incremental DOFs (from 3 to 6), using the same trajectory generation procedure described in (12), but with 75 sinusoids instead of 50. The robot is equipped with torque sensors, therefore we use their measurements as targets of the inverse dynamics estimators. Joints position and velocities are also provided by the robot's interface, while joint accelerations are computed through numerical differentiation. The robot's interface also provides the inertial matrix together with the coriolis and the gravitational torque components. Each collected dataset accounts for about 2900 samples collected at 48 Hz, the training datasets are downsampled to provide 500 samples, in order to reduce the computational burden required by the optimizations. The physically inspired models, namely the GIP and DeLaN estimators, cannot be used directly on real data due to the presence of friction. Both the models, indeed, are designed to catch only the rigid body dynamics and they must be adapted to incorporate the action of non conservative forces. The literature addresses the modeling of actuator friction with several approaches. Most of them consider the motor friction to depend only on the joint velocity $\dot{\boldsymbol{q}}_i$. In particular, in this work we consider a friction model obtained as combination of Coulomb and viscous frictions, namely $\tau_{f,i} = F_c \operatorname{sign}(\dot{\boldsymbol{q}}_i) + F_v \dot{\boldsymbol{q}}_i$. This model can be easily implemented in the DeLaN architecture by augmenting the standard output with an additional term, linear with respect to the joint velocities and their sign. Accordingly, the loss function of (10) can be updated by adding the estimated energy dissipated by the friction component to (9). The GP model with GIP kernel is instead extended by summing, for each joint estimator, a linear kernel with features equal to the i-th joint velocity and its sign. When estimating the components of the torque with the approach described in sec. 5, it is necessary to remove the friction contribute from the total torque estimate. This can be done in a natural way both for the DeLaN and GIP models, due their structural properties, but it is not possible with the baseline models, since they are completely black-box. Figures 3a and 3b show respectively the absolute error distributions on the torque estimates for each joint, and the joints average absolute error distributions on the test trajectory for the real Panda robot with its last joint locked (6 DOF). Figure 3c instead shows the average torque absolute error over all the joints with the increase of DOF for the manipulator. The results presented in this section confirm the considerations of Sec. 6.1: physics-informed machine learning approaches perform better than their respective baseline. Moreover, results show that the GP model with GIP kernel is superior than DeLaN in both estimation of torque and reconstruction of the single torque components.

7. CONCLUSIONS

In this paper, we compared DeLaNs with the GP model equipped with GIP kernel. The two recent black-box approaches follow two alternative routes to improve generalization and data efficiency by embedding the system dynamics' physical laws. DeLaNs impose the structure of the Lagrangian equations but do not constrain the basis functions used. Instead, the GIP kernel constrains the basis functions of the regression problem to a physically inspired finite-dimensional space but does not impose structural properties. We compared the two strategies on

manipulators of increasing degrees of freedom (DOF) both in simulated and real setups. Results show that: (i) the exploitation of knowledge of physics in black-box estimator can improve regression performances. (ii) the DeLaNs accuracy deteriorates rapidly with the DOF increase. For instance, on the 6-DOF setup, DeLaNs predictions are inaccurate, while the GIP kernel model still performs well. (iii) the GIP kernel model better estimates the different components of the dynamics despite not directly imposing structure. Our results suggest that the regularization provided by basis functions selection is particularly powerful in black-box inverse dynamics identification. On the other hand, the derivation of physically consistent models is crucial in several applications, such as control and simulation tasks. We think that merging the GIP kernel with the DeLaNs is an interesting research line worth of further future investigation.

REFERENCES

Camoriano, R., Traversaro, S., Rosasco, L., Metta, G., and Nori, F. (2016). Incremental semiparametric inverse dynamics learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 544–550.

Dalla Libera, A., Amadio, F., Nikovski, D., Carli, R., and Romeres, D. (2021a). Control of mechanical systems via feedback linearization based on black-box gaussian process models. In *2021 European Control Conference (ECC)*.

Dalla Libera, A. and Carli, R. (2019). A data-efficient geometrically inspired polynomial kernel for robot inverse dynamic. *IEEE Robotics and Automation Letters*, 5(1), 24–31.

Dalla Libera, A., Carli, R., and Pillonetto, G. (2020). A novel multiplicative polynomial kernel for volterra series identification. In *2020 World Congress of the International Federation of Automatic Control (IFAC)*.

Dalla Libera, A., Carli, R., and Pillonetto, G. (2021b). Kernel-based methods for volterra series identification. *Automatica*, 129, 109686.

Dalla Libera, A., Tosello, E., Pillonetto, G., Ghidoni, S., and Carli, R. (2019). Proprioceptive robot collision detection through gaussian process regression. In *2019 American Control Conference (ACC)*, 19–24.

De Santis, A., Siciliano, B., De Luca, A., and Bicchi, A. (2008). An atlas of physical human–robot interaction. *Mechanism and Machine Theory*, 43(3), 253–270.

Giacomuzzo, G., Turcato, N., Libera, A.D., and Carli, R. (2022). Advantages of a physics-embedding kernel for robot inverse dynamics identification. In *2022 30th Mediterranean Conference on Control and Automation (MED)*, 355–361.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Haddadin, S., De Luca, A., and Albu-Schäffer, A. (2017). Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6), 1292–1312.

Khosla, P.K. and Kanade, T. (1988). Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators. *The International Journal of Robotics Research*, 7(1), 18–28.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Koubâa, A. et al. (2017). *Robot Operating System (ROS).*, volume 1. Springer.

Kwon, J., Choi, K., and Park, F.C. (2021). Kinodynamic model identification: A unified geometric approach. *IEEE Transactions on Robotics*, 37(4), 1100–1114.

Lutter, M., Listmann, K., and Peters, J. (2019a). Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7718–7725.

Lutter, M. and Peters, J. (2021). Combining physics and deep learning to learn continuous-time dynamics models. *CoRR*, abs/2110.01894.

Lutter, M., Ritter, C., and Peters, J. (2019b). Deep lagrangian networks: Using physics as model prior for deep learning. *CoRR*, abs/1907.04490.

Nguyen-Tuong, D. and Peters, J. (2010). Using model knowledge for learning inverse dynamics. In *2010 IEEE International Conference on Robotics and Automation*, 2677–2682.

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. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Polydoros, A.S., Nalpantidis, L., and Krüger, V. (2015). Real-time deep learning of robotic manipulator inverse dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3442–3448.

Rasmussen, C.E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, 63–71. Springer.

Rezaei-Shoshtari, S., Meger, D., and Sharf, I. (2019). Cascaded gaussian processes for data-efficient robot dynamics learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6871–6877.

Romeres, D., Zorzi, M., Camoriano, R., Traversaro, S., and Chiuso, A. (2020). Derivative-free online learning of inverse dynamics models. *IEEE Transactions on Control Systems Technology*, 28(3), 816–830.

Rueckert, E., Nakatenus, M., Tosatto, S., and Peters, J. (2017). Learning inverse dynamics models in o(n) time with lstm networks. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 811–816.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). Modelling, planning and control. *Advanced Textbooks in Control and Signal Processing. Springer,*.

Sousa, C.D. and Cortesao, R. (2014). Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach. *The International Journal of Robotics Research*, 33(6), 931–944.

Sousa, C.D. (2014). Sympybotics v1.0. URL `https://doi.org/10.5281/zenodo.11365`.