Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 519–524

Forward Dynamics Estimation from Data-Driven Inverse Dynamics Learning ??

Alberto Dalla Libera * **Giulio Giacomuzzo** * **Ruggero Carli** *
Daniel Nikovski ** **Diego Romeres** **

* *Department of Information Engineering, University of Padova, via Gradenigo 6b, Padova, Italy (e-mail: alberto.dallalibera@unipd.it, giulio.giacomuzzo@phd.unipd.it, carlirug@dei.unipd.it).*
** *Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA, United States (e-mail: {nikovski,romeres}@merl.com)*

Abstract: In this paper, we propose to estimate the forward dynamics equations of mechanical systems by learning a model of the inverse dynamics and estimating individual dynamics components from it. We revisit the classical formulation of rigid body dynamics in order to extrapolate the physical dynamical components, such as inertial and gravitational components, from an inverse dynamics model. After estimating the dynamical components, the forward dynamics can be computed in closed form as a function of the learned inverse dynamics. We tested the proposed method with several machine learning models based on Gaussian Process Regression and compared them with the standard approach of learning the forward dynamics directly. Results on two simulated robotic manipulators, a PANDA Franka Emika and a UR10, show the effectiveness of the proposed method in learning the forward dynamics, both in terms of accuracy as well as in opening the possibility of using more structured models.

Keywords: Learning for control; Nonparametric methods; Machine learning;

1. INTRODUCTION

Methods for learning effectively the forward dynamics of mechanical systems, such as various robotic platforms, have been investigated for a long time. The forward dynamics describe the dynamical evolution of a system in response to forces, and are usually used for simulation and control purposes Atkeson and Morimoto (2002); Abbeel et al. (2006); Ng et al. (2006); Romeres et al. (2019a); Dalla Libera et al. (2020b); Ota et al. (2021). Knowledge of the forward dynamics is the basis of any model-based control algorithm, for example in model-based reinforcement learning (MBRL) Amadio et al. (2022, 2023), where it is one element of a Markov decision process (MDP) formulation, generally called the transition function of the MDP, Polydoros and Nalpantidis (2017).

The forward dynamics express the relationship between joint positions, joint velocities, applied forces, and joint accelerations. In RL terms, it represents the transition from the current state to the next state, given the current applied action. A model of the forward dynamics is often given by first-principles equations of motion, the so called rigid body dynamics (RBD). Physical models obtained from first principles are, however, often limited in their ability to describe some real physical phenomena, given our inability to describe certain complex dynamical behaviors of real systems. Moreover, the parameters of these models are often unknown or known only imprecisely. Consequently, in several applications, such models are not sufficiently accurate to describe the system dynamics. In this case, data collected by interacting with the system can be used to improve such models, leading to so called data-driven models.

Data-driven models have been widely studied in the context of inverse dynamics identification. The inverse dynamics is the inverse of the forward dynamics, and it takes as input the trajectory given in terms of joint positions, velocities, and accelerations, and outputs the torques that are causing them. This function is often used in control schemes to improve, for example, the tracking performance of the controller Craig (2005); Nguyen-Tuong et al. (2009); Dalla Libera et al. (2021), see the survey Nguyen-Tuong and Peters (2011) for an overview, and for anomaly detection see e.g., Romeres et al. (2019b).

In the last decades, there has been an increased focus on learning inverse dynamics models of complex robotic systems by means of machine learning, using so called data-driven models, e.g. Gaussian processes regression (GPR) Nguyen-Tuong et al. (2008), deep neural networks Polydoros et al. (2015), support vector machines (SVM), etc. The critical aspect of data-driven solutions is generalization ability, i.e., the phenomenon that estimation accuracy might decrease in configurations that are far from the training samples. For this reason, several studies focused on deriving data-efficient estimators of the inverse dynamics, see, for instance, Nguyen-Tuong et al. (2009); Dalla Libera and Carli (2020); Rezaei-Shoshtari et al. (2019) as black-box solutions. An alternative to black-box solutions are gray-box models, a combination of a physical model and a data-driven model: the physical models exploit prior knowledge, thus increasing data efficiency and generalization, whereas the data driven part compensates for the inaccuracies of the physical model, further improving accuracy. In the GPR literature, these models are named semiparametric models, see for instance, Romeres et al. (2016, 2019a); Camoriano et al. (2016).

When solving the system identification problem for the dynamics of a mechanical system, the inverse dynamics have a significant advantage over the forward dynamics: the inverse dynamics model can be reformulated as a linear model in the

10.1016/j.ifacol.2023.10.1620

inertial parameters Hollerbach et al. (2008), whereas the model of the forward dynamics, in general, does not have such a formulation. This circumstance, while benefiting from the much more extensive set of techniques for learning linear models in comparison to nonlinear ones, also enjoys a structural advantage: learning the inverse dynamics is often better posed than learning the forward dynamics.

Contributions. In this work, we consider an alternative approach to learn the forward dynamics by taking into consideration the more favorable structural properties of the inverse dynamics. Instead of using the standard approach which learns a GP model directly on the forward dynamics, we propose to learn a data-driven inverse dynamics model by means of GPR, and then, compute the forwards dynamics with an exact and deterministic transformation from the learned inverse dynamics model. Experimental results show that, when compared to the standard approach, our strategy leads to better performance in terms of data efficiency and accuracy. Moreover, this strategy allows computing the forward dynamics with kernel types that were previously possible and specialized only to learn the inverse dynamics.

The paper is organized as follows. Section 2 provides background formulation of dynamics models and GPR. The proposed approach is described in Section 3, while experiments are reported in Section 4. Section 5 draws the conclusions.

2. BACKGROUND

We start by providing the background formulation of the robot dynamics. Then, we describe GPR for inverse and forward dynamics identification, with details about the black-box priors adopted in this work.

2.1 Rigid Body Dynamics

Consider a mechanical system with n degrees of freedom and denote with $\boldsymbol{q}_t \in \mathbb{R}^n$ its generalized coordinates at time t; $\dot{\boldsymbol{q}}_t$ and $\ddot{\boldsymbol{q}}_t$ are the velocity and the acceleration of the joints, respectively. The generalized torques, i.e., the control input of the system, are denoted by $\boldsymbol{\tau}_t \in \mathbb{R}^n$. For compactness, we will denote explicitly the dependencies on t only when strictly necessary. Under rigid body assumptions, the dynamics equations of a mechanical system are described by the following matrix equation, called rigid body dynamics (RBD):

$$B(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{1}$$

where $B(\boldsymbol{q})$ is the inertia matrix, while $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\boldsymbol{g}(\boldsymbol{q})$, and $\boldsymbol{F}(\dot{\boldsymbol{q}})$ account for the contributions of fictitious forces, gravity, and friction, respectively, see Siciliano et al. (2010) for a more detailed description. For compactness, we introduce also $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}})$, and the symbols $\hat{B}(\boldsymbol{q})$ and $\hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ will denote the estimates of $B(\boldsymbol{q})$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$.

Linear Model of Inverse Dynamics The model in eq. (1) is linear w.r.t. the dynamics parameters, i.e., mass, center of mass, inertia, and friction coefficients of the links, see Siciliano et al. (2010). When neglecting friction, the number of dynamics parameters of each link is $p = 10$, one for mass, three for center of mass, and six for the inertia tensor. Let $\boldsymbol{w} \in \mathbb{R}^{n \cdot p}$ be the vector collecting the dynamics parameters of all the links, then

$$\boldsymbol{\tau} = \Phi\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) \boldsymbol{w} = \begin{bmatrix} \boldsymbol{\phi}^{(1)}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) \\ \vdots \\ \boldsymbol{\phi}^{(n)}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) \end{bmatrix} \boldsymbol{w}, \tag{2}$$

where $\Phi\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) \in \mathbb{R}^{n \times (n \cdot p)}$ depends only on the kinematics parameters of the robot, i.e., its geometry.

The forward dynamics is the map that, given the current position, velocity, and torque $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}$, outputs the acceleration $\ddot{\boldsymbol{q}}$. This model is needed for simulation and prediction purposes.

Forward dynamics learning is known to be more complex than learning the inverse dynamics. This can be seen directly in the RBD model given by physics. From eq. (1), we have

$$\ddot{\boldsymbol{q}} = B^{-1}\left(\boldsymbol{q}\right)\left(\boldsymbol{\tau} - C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} - \boldsymbol{g}\left(\boldsymbol{q}\right)\right), \tag{3}$$

Equation (3) cannot be written, in general, as a linear function of the dynamics parameters, as it can be done for the inverse dynamics, and the presence of the inverse of the inertia matrix creates a nonlinear dependence on the parameters. This complexity would be encoded in the black-box map that a machine learning algorithm would attempt to describe, if trying to learn the forward dynamics directly, i.e. $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau} \longrightarrow \ddot{\boldsymbol{q}}$.

2.2 GPR for Inverse Dynamics Identification

GPR provides a solid probabilistic framework to identify the inverse dynamics from data. Typically, in GPR, each joint torque is modeled by a distinct and independent GP. Consider an input/output data set $\mathcal{D} = \left\{\boldsymbol{y}^{(i)}, X\right\}$, where $\boldsymbol{y}^{(i)} \in \mathbb{R}^N$ is a vector containing N measurements of $\tau^{(i)}$, the i-th joint torque, while $X = \{\boldsymbol{x}_{t_1} \dots \boldsymbol{x}_{t_N}\}$; the vector $\boldsymbol{x}_t$ contains the position, velocity, and acceleration of the joints at time t, hereafter designated as GP input. The probabilistic model of $\mathcal{D}$ is

$$\boldsymbol{y}^{(i)} = \begin{bmatrix} f^{(i)}\left(\boldsymbol{x}_{t_1}\right) \\ \vdots \\ f^{(i)}\left(\boldsymbol{x}_{t_N}\right) \end{bmatrix} + \begin{bmatrix} e^{(i)}_{t_1} \\ \vdots \\ e^{(i)}_{t_N} \end{bmatrix} = \boldsymbol{f}^{(i)}(X) + \boldsymbol{e}^{(i)},$$

where $\boldsymbol{e}^{(i)}$ is i.i.d. Gaussian noise with standard deviation σ_i, while $f^{(i)}(\cdot)$ is an unknown function modeled a priori as a GP, namely, $f^{(i)}(\cdot) \sim N(m_{f^i}(X), \mathbb{K}^{(i)}(X, X))$. $m_{f^i}(X)$ denotes the prior mean, and, generally, it is assumed to be equal to zero when no prior knowledge is available. The covariance matrix $\mathbb{K}^{(i)}(X, X)$ is defined through a kernel function $k^{(i)}(\cdot, \cdot)$. Specifically, the covariance between $f^{(i)}\left(\boldsymbol{x}_{t_j}\right)$ and $f^{(i)}\left(\boldsymbol{x}_{t_l}\right)$, i.e., the element of $\mathbb{K}^{(i)}(X, X)$ at row *j* and column *l*, is equal to $k^{(i)}\left(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_l}\right)$. Exploiting the properties of Gaussian distributions, it can be proven that the posterior distribution of $f^{(i)}$ given $\mathcal{D}$ in a general input location $\boldsymbol{x}_*$ is Gaussian, see Rasmussen and Williams (2005) for a comprehensive description. Then, the maximum a posteriori estimator corresponds to the mean, which is given by the following expression

$$\hat{f}^{(i)}(\boldsymbol{x}_*) = \mathbb{K}^{(i)}\left(\boldsymbol{x}_*, X\right)\boldsymbol{\alpha}^{(i)} + m_{f^i}(\boldsymbol{x}_*), \tag{4}$$

where

$$\boldsymbol{\alpha}^{(i)} = \left(\mathbb{K}^{(i)}\left(X, X\right) + \sigma_i^2 I\right)^{-1}\left(\boldsymbol{y}^{(i)} - m_{f^i}(X)\right),$$

$$\mathbb{K}^{(i)}\left(\boldsymbol{x}_*, X\right) = \left[k^{(i)}\left(\boldsymbol{x}_*, \boldsymbol{x}_{t_1}\right) \dots k^{(i)}\left(\boldsymbol{x}_*, \boldsymbol{x}_{t_N}\right)\right].$$

Different solutions proposed in the literature can be grouped roughly based on the definition of the GP prior. In this paper,

we will consider two black-box approaches, where the prior is defined without exploiting prior information about the physical model, and assuming $m_{f^i}(X) = 0$.

Squared Exponential kernel The Squared Exponential (SE) kernel defines the covariance between samples based on the distance between GP inputs, see, for instance, Rasmussen and Williams (2005), and it is defined by the following expression

$$k_{SE}(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_l}) = \lambda e^{-\left\|\boldsymbol{x}_{t_j} - \boldsymbol{x}_{t_l}\right\|^2_{\Sigma^{-1}}}; \tag{5}$$

λ and Σ are kernel hyperparameters. The former is a positive scaling factor, and the latter is a positive definite matrix which defines the norm used to compute the distance between inputs. A common choice consists in considering Σ to be diagonal, with the positive diagonal elements named lengthscales.

Geometrically Inspired Polynomial kernel The Geometrically Inspired Polynomial (GIP) kernel has been recently introduced in Dalla Libera and Carli (2020). This kernel is based on the property that the dynamics equations in (1) are a polynomial function in a proper transformation of the GP input, fully characterized only by the type of each joint. Specifically, $\boldsymbol{q}$ is mapped to $\tilde{\boldsymbol{q}}$, the vector composed by the concatenation of the components associated with a prismatic joint and the sines and cosines of the revolute coordinates. As proved in Dalla Libera and Carli (2020), the inverse dynamics in (1) is a polynomial function in $\ddot{\boldsymbol{q}}$, $\dot{\boldsymbol{q}}$ and $\tilde{\boldsymbol{q}}$, where the elements of $\ddot{\boldsymbol{q}}$ have maximum relative degree of one, whereas the ones of $\dot{\boldsymbol{q}}$ and $\tilde{\boldsymbol{q}}$ have maximum relative degree of two. To exploit this property, the GIP kernel is defined through the sum and the product of different polynomial kernels (Dalla Libera et al. (2020a)), hereafter denoted as $k_P^{(p)}(\cdot,\cdot)$, where p is the degree of the polynomial kernel. In particular, we have

$$\begin{aligned} &k_{GIP}(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_l}) = \\ &\left(k_P^{(1)}(\ddot{\boldsymbol{q}}_{t_j}, \ddot{\boldsymbol{q}}_{t_l}) + k_P^{(2)}(\dot{\boldsymbol{q}}_{t_j}, \dot{\boldsymbol{q}}_{t_l})\right) k_Q(\tilde{\boldsymbol{q}}_{t_j}, \tilde{\boldsymbol{q}}_{t_l}), \end{aligned} \tag{6}$$

where, in its turn, k_Q is given by the product of polynomial kernels with degree two, see Dalla Libera and Carli (2020) for all the details. In this way, the GIP kernel allows defining a regression problem in a finite-dimensional function space where (1) is contained, leading to better data efficiency in comparison with the SE kernel.

- Define the mean m_{f^i} equal to the *i*-th output of (1) or (2). Typically, in this case, the covariance is defined through an SE kernel, which aims at compensating inaccuracies of the prior mean.
- Define $m_{f^i} = 0$, and the kernel as the sum of two kernels. The first kernel is a linear kernel derived from (2) assuming that $\boldsymbol{w} \sim N(0, \Sigma)$, whereas the second is a standard SE kernel. Then, $k_{SP}^{(i)}$, namely, the semiparametrical kernel of the *i*-th joint, is defined by the following expression:

  $$k_{SP}^{(i)}(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_l}) = \phi^{(i)}(\boldsymbol{x}_{t_j}) \Sigma \phi^{(i)}(\boldsymbol{x}_{t_l})^T + k_{SE}^{(i)}(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_l}),$$

  where matrix Σ is a tunable hyperparameter, typically assumed diagonal, expressing the prior covariance of $\boldsymbol{w}$, Romeres et al. (2016, 2019a); Camoriano et al. (2016).

We remark that we limit our investigation to black-box solutions, since we want to test the proposed approach w.r.t. standard direct learning of the forward dynamics (which cannot use semiparametric kernels). However, we would like to stress that our approach is fully compatible wih any kernel function.

2.3 GPR for forward dynamics identification

The GPR framework presented for the inverse dynamics learning can be applied also to the forward dynamics. When considering the *i*-th joint, the input of the GP is the vector containing $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$, while the output is the *i*-th component of $\ddot{\boldsymbol{q}}$. However, w.r.t. the inverse dynamics, the choices for the GP prior are limited. (i) The GIP kernel cannot be applied, since it is based on the assumption that $\boldsymbol{\tau}$ is a polynomial function in a proper transformation of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$, but there is not an equivalent property for $\ddot{\boldsymbol{q}}$. (ii) Due to the fact that there is not an equivalent relation (2), i.e., $\ddot{\boldsymbol{q}}$ are not linear w.r.t. dynamics parameters, in general it is not possible to formulate a semiparametric kernel. Then, the options commonly available are:

- If no prior is available, assume $m_{f^i} = 0$, and define the covariance a priori as a SE kernel (or any non-structured kernel) with GP input $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$;
- In case that (1) is known, a so called residual model can be used. The prior knowledge can be exploited by defining the mean m_{f^i} to be equal to the *i*-th component of $M(\boldsymbol{q})^{-1}(\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}))$, and the covariance through an SE kernel, to compensate for eventual inaccuracies of the forward physical model.

3. ESTIMATING FORWARD DYNAMICS FROM INVERSE DYNAMICS LEARNING

In this section, we describe the proposed approach to estimating the forward dynamics (3) as a function of the inverse dynamics learned using GPR. In particular, we discuss how the physical equations of the RBD (1) entail an exact relationship that can be leveraged to compute gravitational contributions, inertial contributions, and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ as a function of the inverse dynamics evaluated in a subset of the inputs. We assume that a distinct GP is used for each of the n degrees of freedom, and we denote by $\hat{f}^{(i)}(\cdot)$, $i = 1 \dots n$ the estimator of the *i*-th joint torque obtained by applying (4). For convenience, from here on, we will point out explicitly the different components of the GP input, namely, the input of $\hat{f}^{(i)}$ will be $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ instead of $\boldsymbol{x}$, which comprises the concatenation of $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$. It is worth mentioning that the proposed approach is inspired by the strategy adopted in Newton-Euler algorithms, see De Luca and Ferrajoli (2009).

The proposed approach consists of (i) learning the inverse dynamics, which is a function suitable for identification purposes given its linear formulation, and (ii) using the learned model to have a closed-form deterministic transformation that estimates the forward dynamics. The method is described in Algorithm 1.

Assumption 1. The dependency of the dynamical components $B(\boldsymbol{q})\ddot{\boldsymbol{q}}$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ w.r.t. the input quantities $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ in (1) is exact. That is, the inertial contributions do not depend on $\dot{\boldsymbol{q}}$, the $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ do not depend on $\ddot{\boldsymbol{q}}$, and there are no terms with cross dependency that appear in the equations of motions.

Assumption 1 is a fairly mild assumption commonly assumed in classical manuscripts. The estimation of each dynamical component is described in the following.

In Algorithm 1, initially the inverse dynamics, $\hat{f}^{(i)}$, are learned for each DoF of the mechanical system using GPR as described in Section 2.2. These models are now considered known and fixed. Then, the inertia matrix $B(\boldsymbol{q})$, the Coriolis and gravitational forces $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$ are estimated as a

function of the learned inverse dynamics. Finally, the forward dynamics can be computed according to the physical laws (3).

Algorithm 1 Inverse2Forward

collect data $\mathcal{D} = \{(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t), \boldsymbol{\tau}_t\}$
Inverse Dynamics Learning
for $i = 1...n_{DoF}$ **do**
$\hat{f}^i \leftarrow$ learn inverse dynamic, from (4)
end for
Forward Dynamics Estimation
$\hat{\boldsymbol{g}}(\boldsymbol{q}) \leftarrow$ estimate gravitational component, from (7)
$\hat{B}(\boldsymbol{q}) \leftarrow$ estimate inertial component, from (8)
$\hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leftarrow$ estimate Coriolis and gravitaional component, from (9)
Compute $\ddot{\boldsymbol{q}} = \hat{B}^{-1}(\boldsymbol{q})\left(\boldsymbol{\tau} - \hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right)$

Gravitational contribution. The motion equations in (1) show that the torque components due to the gravitational contributions account for all the terms that depend only on $\boldsymbol{q}$. Consequently, to obtain $g^{(i)}(\boldsymbol{q})$, i.e., the estimate of the i-th gravitational contribution in the configuration $\boldsymbol{q}$, we evaluate $\hat{f}^{(i)}$ by setting $\dot{\boldsymbol{q}} = \boldsymbol{0}, \ddot{\boldsymbol{q}} = \boldsymbol{0}$. Then, the estimate of $\boldsymbol{g}(\boldsymbol{q})$ is

$$\hat{\boldsymbol{g}}(\boldsymbol{q}) = \begin{bmatrix} \hat{g}^{(1)}(\boldsymbol{q}) \\ \vdots \\ \hat{g}^{(n)}(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \hat{f}^{(1)}(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{0}) \\ \vdots \\ \hat{f}^{(n)}(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{0}) \end{bmatrix}. \tag{7}$$

Inertial contributions. The inertial contributions, i.e., $B(\boldsymbol{q})\ddot{\boldsymbol{q}}$, account for all the contributions that depend simultaneously on $\boldsymbol{q}$ and $\ddot{\boldsymbol{q}}$. Consequently, to estimates these contributions, we evaluate the GP models in $(\ddot{\boldsymbol{q}}, \boldsymbol{0}, \boldsymbol{q})$, and subtract the gravitational contribution defined and computed previously. In particular, to obtain the element $\hat{B}_{ij}(\boldsymbol{q})$, i.e., the estimate of the $B(\boldsymbol{q})$ element in position (i, j), we set all the accelerations to zero, except for the j-th component. Denoting with $\boldsymbol{1}_j$ the vector with all elements equal to zero except for the j-th element, which equals one, we have

$$\hat{B}_{ij}(\boldsymbol{q}) = \hat{f}^{(i)}(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{1}_j) - \hat{g}^{(i)}(\boldsymbol{q}). \tag{8}$$

Estimation of $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. The vector $\boldsymbol{n}(\boldsymbol{q})$, defined under (1), contains all the contributions that do not depend on $\ddot{\boldsymbol{q}}$. Then, $\hat{n}^{(i)}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, i.e., the estimate of the i-th component of $\boldsymbol{n}(\boldsymbol{q})$, is computed by evaluating the i-th GP model by setting $\ddot{\boldsymbol{q}} = \boldsymbol{0}$. Therefore, we can compute:

$$\hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} \hat{n}^{(1)}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \vdots \\ \hat{n}^{(n)}((\boldsymbol{q}, \dot{\boldsymbol{q}})) \end{bmatrix} = \begin{bmatrix} \hat{f}^{(1)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{0}) \\ \vdots \\ \hat{f}^{(n)}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{0}) \end{bmatrix}. \tag{9}$$

The proposed Algorithm 1 is based on well-known first-principle relationships. Yet, it offers a useful bridge between modern machine learning techniques and the principles of classical mechanics that, as will be seen in Section 4, improves significantly upon the standard approach to learn directly the forward dynamics.

A further advantage w.r.t. standard approaches is that more structured kernels can be utilized to learn the forward dynamics. As described in Sections 2.2-2.3, the GIP and semiparametric kernels cannot be used to directly learn the forward dynamics. However, these kernels can be used to learn the inverse dynamics in the initial steps of Algorithm 1, and consequently used to learn the forward dynamics at the end of Algorithm 1.

Note that the method is not restricted to any function approximator to compute the inverse dynamics, e.g. neural networks could be used too; rather, it offers a general methodology, easy to implement, that reduces the problem of learning the forward dynamics to the more convenient inverse dynamics learning.

4. EXPERIMENTS

This section compares the learning performance of our approach with standard GP-based forward dynamics learning. Specifically, we implemented Algorithm 1 using as prior $m_{f^i} = 0$ and the two black-box kernels introduced in (5) and (6). The models obtained are hereafter referred to as SE and

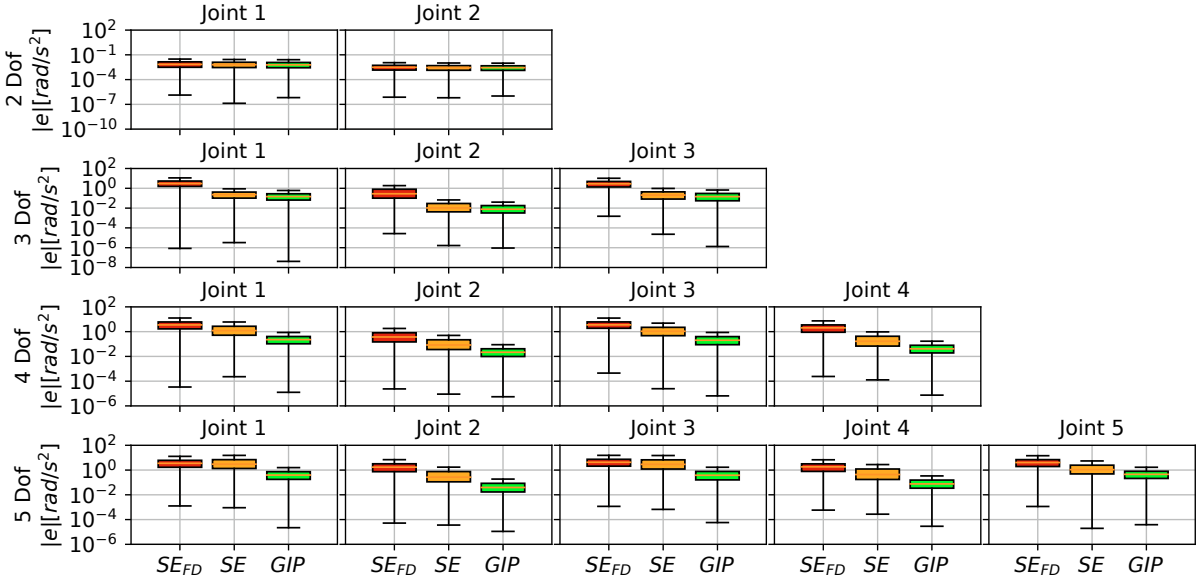

Fig. 1. Distribution of the acceleration residual as a function of the DoFs on the simulated PANDA robot.

GIP estimators. As a baseline, we implemented the standard black-box forward dynamics estimator with $m_{f^i} = 0$ and SE kernel described in Section 2.3, hereafter denoted by SE_{FD}.

We carried out two experiments on simulated setups. The first investigated the behavior of the estimators as the DoFs increase on a simulated PANDA Franka Emika robot. The second, instead, analyzed the data-efficiency performance on a UR10 robot. Training and test data sets of the kind: $\mathcal{D} = \{(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t), \boldsymbol{\tau}_t\}$ are obtained by generating joint trajectories and evaluating (1) to compute the joint torques. The dynamics equations (1) are derived using the Python package *Sympybotics* [1], Sousa and Cortesão (2014). All the estimators were implemented in Python, starting from *gpr-pytorch* [2], a library for GPR based on *pytorch*, Paszke et al. (2017).

4.1 Performance as a Function of DoFs

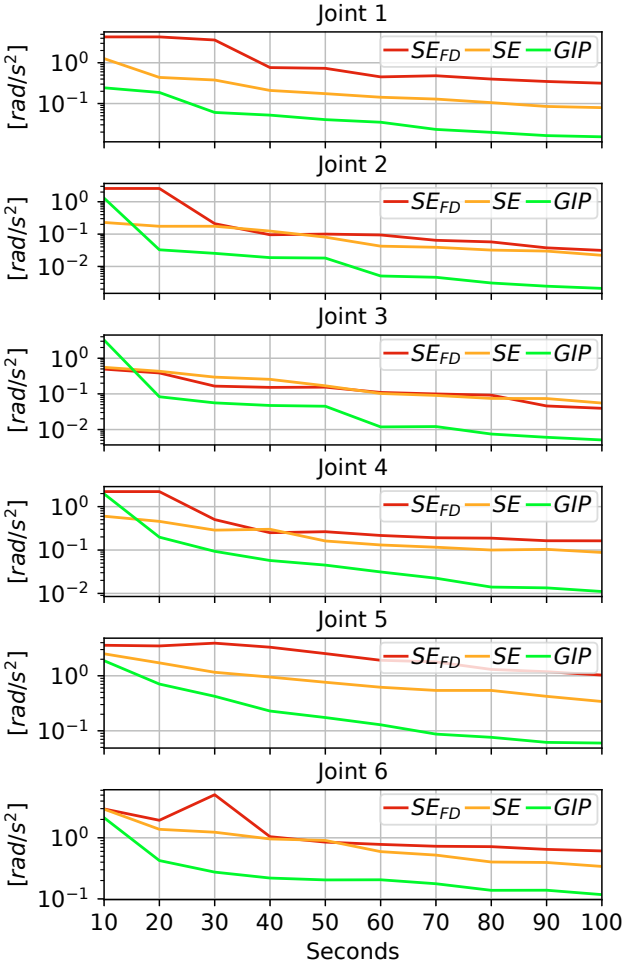

Fig. 2. Median of the acceleration error modules as a function of the training data (in seconds), for each DoF.

We compared the performance of the three estimators on a simulated PANDA robot as the number of DoFs increased from two to five. For each DoF, we collected a training and a test data set. The reference trajectories followed by the robot were different realization of Gaussian noise filtered with a low-pass filter, with cut-off frequency $1Hz$. The trajectories lasted 100 seconds, collected at $100Hz$ for a total of 10000 samples per data set. The joint torques were corrupted by Gaussian noise with standard deviation $0.01Nm$. We computed the three models following Alg 1 and trained the kernel hyperparameters by maximizing the marginal likelihood of the training data, Rasmussen and Williams (2005).

Kernel	2 DoF	3 DoF	4 DoF	5 DoF
SE	0.0101	0.0347	0.2282	0.2636
GIP	0.0016	0.0088	0.0465	0.0413

Table 1. Inverse dynamics RMSEs in Nm of the SE and GIP estimators as a function of the DoF.

For each DoF, Figure 1 reports the distribution of the acceleration error modules. It is particularly interesting to compare the performance of the SE_{FD} and SE estimators, since they adopt the same kernel to model, respectively, accelerations and torques. The proposed SE estimator performs similarly to SE_{FD} in the 2-DoF experiment, which is a simple test, and it ouptperforms the baseline in all the other experiments with higher DoFs. These results show that the proposed solution can improve the standard direct learning of the forward dynamics. Interestingly, the GIP estimator outperforms the baseline approach and the SE estimator for all the DoFs and for all the joints. The performance gap between the GIP and SE estimators increases with the increase of the DoF. While the distributions of the estimation errors of GIP and SE are similar for two and three DoF, GIP significantly outperforms SE for four and five DoF. The deterioration of the SE performance is due to the limited generalization properties of the SE kernel, which affects the accuracy of the inverse dynamics model when the DoF increases. Table 1 reports the Root Mean Squared Errors of the torques estimates for each DoF tested. The RMSEs of the SE estimator grow much rapidly with the DoF, compared with the GIP estimator. Torque estimation errors are amplified in Algorithm 1, limiting also the accuracy of the forward dynamics estimation.

Thanks to the higher data efficiency and generalization of the GIP kernel (see the RMSEs in Table 1), the GIP estimator estimates accelerations accurately also in the setup with 5 DoF. The performance of the GIP estimator shows another advantage of the proposed approach, that is, the possibility of using data-efficient solutions proposed for inverse dynamics. As discussed in Section 2, the forward dynamics in (3) does not admit convenient representation like the linear model in (2) or the GIP polynomial representation described in Dalla Libera and Carli (2020). These advantages can be further exploited by relying on the SP kernels described in Section 2.2. Using these kernels in simulated data would be unfair, as we have the exact knowledge of the physical model generating the data.

4.2 Data Efficiency Performance

We compared the data efficiency of the SE_{FD}, SE and GIP estimators by evaluating their accuracy as a function of the amount of training data. In this experiment, we considered a simulated UR10 robot. As in the previous experiment, the reference trajectories followed in the training and test data sets are distinct realizations of Gaussian noise filtered at $1Hz$. In Fig. 2, we reported the medians of the acceleration error modules as a function of the number of seconds of training samples used. We used the first $10, 20, \ldots, 100$ seconds of training data to derive the 3 estimators, after optimizing the kernel hyperparameters by maximizing the marginal likelihood of the training data. Then,

[1] https://github.com/cdsousa/SymPyBotics
[2] https://bitbucket.org/AlbertoDallaLibera/gpr-pytorch

we tested the derived estimators in the whole test data set.
The SE_{FD} and SE estimators perform similarly for joints 2 and 3, but the SE estimator outperforms SE_{FD} for the other joints. This confirms the efficacy of the proposed method as a general method that, with the same model, outperforms the standard approach to learning the forward dynamics directly. Figure 2 shows that the GIP estimator significantly outperforms the other models, both in terms of accuracy and data efficiency. After around only $30s$ of data the GIP estimators performs like the other estimators after $100s$. The performance of the GIP estimator confirms that the proposed approach can significantly improve the direct forward dynamics learning by exploiting the data-efficient solutions proposed for inverse dynamics, which are not available for forward dynamics models.

5. CONCLUSIONS

In this work, we present a black-box GP-based strategy to learn the forward dynamics model. The proposed algorithm defines a prior on the inverse dynamics function instead of directly modeling the forward dynamics. Based on the learned inverse dynamics model, the algorithm computes individual dynamics components, i.e., inertial, gravitational, and Coriolis contributions, to estimate the forward dynamics. Experiments carried out in simulated environments show that the proposed strategy can be more accurate and data-efficient than directly learning accelerations in a black-box fashion. The advantages w.r.t. the standard approach are particularly relevant when considering data-efficient kernels, such as the GIP kernel, which are not available to model directly the forward dynamics. The proposed method is general and can be adapted to any estimator, both black-box or gray-box. Next, we will test the approach both on data from real robots and make use of semiparametric kernels, and on MBRL algorithms as transition function.

REFERENCES

Abbeel, P., Coates, A., Quigley, M., and Ng, A. (2006). An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*.

Amadio, F., Dalla Libera, A., Antonello, R., Nikovski, D., Carli, R., and Romeres, D. (2022). Model-based policy search using monte carlo gradient estimation with real systems application. *IEEE Transactions on Robotics*.

Amadio, F., Libera, A.D., Nikovski, D., Carli, R., and Romeres, D. (2023). Learning control from raw position measurements. *arXiv preprint arXiv:2301.13183*.

Atkeson, C.G. and Morimoto, J. (2002). Nonparametric representation of policies and value functions: A trajectory-based approach.

Camoriano, R., Traversaro, S., Rosasco, L., Metta, G., and Nori, F. (2016). Incremental semiparametric inverse dynamics learning. *arXiv preprint arXiv:1601.04549*.

Craig, J.J. (2005). *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall Upper Saddle River.

Dalla Libera, A. and Carli, R. (2020). A data-efficient geometrically inspired polynomial kernel for robot inverse dynamic. *IEEE Robotics and Automation Letters*, 5(1), 24–31.

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

Dalla Libera, A., Carli, R., and Pillonetto, G. (2020a). A novel multiplicative polynomial kernel for volterra series identification. *IFAC-PapersOnLine*, 53(2), 316–321.

Dalla Libera, A., Romeres, D., Jha, D.K., Yerazunis, B., and Nikovski, D. (2020b). Model-based reinforcement learning for physical systems without velocity and acceleration measurements. *IEEE Robotics and Automation Letters*, 5(2).

De Luca, A. and Ferrajoli, L. (2009). A modified newton-euler method for dynamic computations in robot fault detection and control. In *2009 IEEE International Conference on Robotics and Automation*, 3359–3364.

Hollerbach, J., Khalil, W., and Gautier, M. (2008). Model identification. In *Handbook of Robotics*, 321–344. Springer.

Ng, A.Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. (2006). Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, 363–372. Springer.

Nguyen-Tuong, D. and Peters, J. (2011). Model learning for robot control: a survey. *Cognitive Processing*, 12(4).

Nguyen-Tuong, D., Seeger, M., and Peters, J. (2008). Computed torque control with nonparametric regression models. In *2008 American Control Conference*, 212–217. IEEE.

Nguyen-Tuong, D., Seeger, M., and Peters, J. (2009). Model learning with local gaussian process regression. *Advanced Robotics*, 23(15), 2015–2034.

Ota, K., Jha, D.K., Romeres, D., van Baar, J., Smith, K.A., Semitsu, T., Oiki, T., Sullivan, A., Nikovski, D., and Tenenbaum, J.B. (2021). Data-efficient learning for complex and real-time physical problem solving using augmented simulation. *IEEE Robotics and Automation Letters*, 4241–4248.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Polydoros, A.S. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2), 153–173.

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. IEEE.

Rasmussen, C.E. and Williams, C.K.I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

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., Jha, D.K., DallaLibera, A., Yerazunis, B., and Nikovski, D. (2019a). Semiparametrical gaussian processes learning of forward dynamical models for navigating in a circular maze. In *2019 International Conference on Robotics and Automation (ICRA)*, 3195–3202. IEEE.

Romeres, D., Jha, D.K., Yerazunis, W., Nikovski, D., and Dau, H.A. (2019b). Anomaly detection for insertion tasks in robotic assembly using gaussian process models. In *2019 18th European Control Conference (ECC)*. IEEE.

Romeres, D., Zorzi, M., Camoriano, R., and Chiuso, A. (2016). Online semi-parametric learning for inverse dynamics modeling. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2945–2950. IEEE.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated.

Sousa, C. and Cortesão, R. (2014). Physical feasibility of robot base inertial parameter identification:a linear matrix inequality approach. *The International Journal of Robotic Research*.