A novel geometrically inspired polynomial kernel for robot inverse dynamics

Alberto Dalla Libera and Ruggero Carli

Abstract—In this paper we introduce a novel data driven inverse dynamics estimator based on Gaussian Process Regression. Driven by the fact that the inverse dynamics can be described as a polynomial function on a suitable input space, we propose the use of a novel kernel, called Geometrically Inspired Polynomial Kernel (GIP). The resulting estimator behaves similarly to model based approaches as concerns data efficiency. Indeed, we proved that the GIP kernel defines a finite dimensional Reproducing Kernel Hilbert Space that contains the inverse dynamics function computed through the Rigid Body Dynamics. The proposed kernel is based on the recently introduced Multiplicative Polynomial Kernel, a redefinition of the classical polynomial kernel equipped with a set of parameters which allows for an higher regularization. We tested the proposed approach in a simulated environment, and also in real experiments with a UR10 robot. The obtained results confirm that, compared to other data driven estimators, the proposed approach is more data efficient and exhibits better generalization properties. Instead, with respect to model based estimators, our approach requires less prior information and is not affected by model bias.

I. INTRODUCTION

Learning the inverse dynamics model of a robot directly from data is still a challenging task in robotics, worth of investigation, as demonstrated by several important applications. For instance, by learning such a model, it is possible to design robot controllers based on feed-forward strategies [1] and on more complex Model Predictive Control approaches [2], or to estimate the external forces applied to the end effector without using force sensors [3] or, more in general, to provide robots with proprioceptive sensing capabilities [4]. Learning models directly from data has several advantages. Firstly, the derivation of a model is not always an easy task, and, even when a model is available, its use introduces a bias due to uncertainties on the values of parameters which are assumed known or to assumptions which are just a rough approximation of the real behavior of the robot. Secondly, data driven approaches are not platform dependent, namely the same learning technique can be applied to different physical platforms, leading to considerable advantages in terms of time and costs design.

Several data-driven strategies to learn inverse dynamics have been developed in the literature. The authors in [5] proposed a locally weighted projection of different linear models. A significant number of approaches relies on the development of suitable neural networks algorithms; for instance the authors in [6] resort to the use of a recurrent neural network, while in [7] a LSTM network has been proposed. Another wide class of solutions is based on Gaussian Process Regression (GPR) [8], [9], [10]. Differently from neural networks, GPR provides also a bound on the uncertainty of the estimated model, and this additional information can be exploited in different ways, see for instance in Reinforcement Learning the PILCO algorithm [11].

Although data driven modeling techniques have been applied successfully in several control applications, see for example [11], [12], [13], they are still not able to guarantee the same generalization properties of model based learning techniques. Indeed, data driven approaches capture only similarity between data, without exploiting important features like causality or presence of constraints imposed by physics and geometry. This fact results in a considerable data inefficiency, which is particularly evident in systems with a high number of degrees of freedom. The typical huge amount of data required by standard data driven approaches poses serious limitations on their applicability, mainly due to the high computational burden needed to process all the available information, in addition to the difficulty of guaranteeing good generalization properties.

In this paper we investigate the possibility of developing data driven estimators of robot inverse dynamics exhibiting good generalization properties and high data efficiency. The main contribution of the paper is the design of a data driven inverse dynamics estimator based on GPR, more precisely on a novel kernel function, named Geometrically Inspired Polynomial Kernel (GIP). The main idea supporting our approach is related to the existence of a suitable transformation of the standard inverse dynamics inputs, that are, positions, velocities and accelerations of the generalized coordinates, into an augmented space where the inverse dynamics map is well approximated by a polynomial function. As highlighted in [14], standard polynomial kernels are not widely used in regression problems, since they are prone to overfitting. To overcome this limitation, we adopted a re-parametrization of polynomial kernel, recently introduced in [15]. This variation of the polynomial kernel, named Multiplicative Polynomial Kernel (MPK), allows for a bigger flexibility in neglecting eventual unnecessary basis functions of the corresponding Reproducing Kernel Hilbert Space (RKHS), thus leading to better conditioned problems.

The polynomial-based strategy we introduce is tested both in a simulated environment and with data acquired from real experiments on UR10 robot. Despite the GIP estimator requires minimal prior information compared to model based estimators, the obtained results show that the proposed ap-
proach exhibits comparable performance in terms of accuracy and generalization. Additionally, compared to data driven approaches, our learning algorithm is more data efficient and exhibits better generalization properties.

The paper is organized as follow. In Section II we provide an overview of the main strategies based on GPR and adopted in inverse dynamics learning. In Section III we describe the approach we propose. Firstly, we identify an input transformation that leads to a description of the rigid body dynamics equations in terms of polynomial functions. Secondly we briefly review MPK. Thirdly, we define the GIP kernel. Finally in Section IV we test the proposed estimator in a simulated environment, representing a SCARA robot, and on data coming from real experiments performed with a UR10 robot.

II. ROBOT INVERSE DYNAMICS: LEARNING STRATEGIES

In this Section we briefly review the dynamics model of robot manipulators and the main approaches proposed to deal with the inverse dynamics problem.

Consider a robot manipulator with \( n + 1 \) links and \( n \) joints, and let \( q = [q_1, \ldots, q_n]^T \in \mathbb{R}^n \) be the vector collecting the generalized coordinates associated to the joints and, accordingly, \( \dot{q} \) and \( \ddot{q} \) be the velocity and acceleration vectors, respectively. The inverse dynamics problem consists in estimating the function mapping the triple \( (q, \dot{q}, \ddot{q}) \), into the vector of generalized torques denoted by \( \tau \in \mathbb{R}^n \).

The estimation is typically performed starting from a set of input-output observations which is composed by the set of input locations \( X = \{x(t_1), \ldots, x(t_{N_{TR}})\} \), where \( x(t) = [q(t), \dot{q}(t), \ddot{q}(t)] \), and the corresponding set of outputs \( Y = \{\tau(t_1), \ldots, \tau(t_{N_{TR}})\} \), being \( N_{TR} \) the total number of observations. In the following, when there is no risk of confusion, we will omit the dependence on time \( t \). Moreover by \( y \) we denote the vector obtained stacking together all the elements in \( Y \).

A. Rigid body dynamics estimators

Several approaches which have been proposed to deal with the inverse dynamics problem are based on a rigid body dynamics (RBD) assumption. Under this assumption, the robot dynamics is described as

\[
\tau = B(q) \dot{q} + C(q, \dot{q}) \ddot{q} + g(q),
\]

where \( B(q) \in \mathbb{R}^{n \times n} \) and \( C(q, \dot{q}) \in \mathbb{R}^{n \times n} \) are respectively the inertia matrix and the Coriolis matrix, and \( g(q) \) is the vector accounting for the gravitational contributions, see [16] for a detailed description. The previous equation depends on two sets of parameters, respectively, the kinematic and dynamics parameters. The first set is composed by geometric quantities (i.e., lengths, angles) that, together with \( q \), define the forward kinematic. The second set, instead, contains the masses, centers of mass, and inertia components of the links. Remarkably, it is possible to show that \( B(q) \) is linear respect to the dynamics parameters, see [16]. Specifically, collecting all the dynamics parameters into the vector \( \omega \in \mathbb{R}^{N_{par}} \) (\( N_{par} \) denotes the number of dynamical parameters), \( B(q) \) can be rewritten as

\[
\tau = \Phi(q, \dot{q}, \ddot{q}) \omega = \Phi(x) \omega = [\phi_1^T(x) \ldots \phi_n^T(x)]^T \omega,
\]

for a suitable matrix \( \Phi \in \mathbb{R}^{n \times N_{par}} \) which depends only on the kinematic parameters. Assuming the kinematic parameters to be known, the matrix \( \Phi \) is well defined, and the inverse dynamics problem boils down to the computation of an estimate \( \hat{\omega} \) of \( \omega \).

Several learning techniques, which have been proposed to compute \( \hat{\omega} \), rely on Fisherian techniques, see for example [17]. Though, when the model is sufficiently accurate, these estimators achieve accurate estimates together with good generalization properties, there are several limitations that prevent their applicability in general scenarios. Indeed it is worth stressing that to apply Fisherian techniques the robot kinematic parameters must be precisely known since errors in the knowledge of these parameters introduce model bias that can affect significantly estimation performance.

Moreover there are situations where it is hard to derive \( \Phi \) or where the rigid body assumption is a too rough approximation of the real robot behaviors.

B. Gaussian Process Regression for robot inverse dynamics

To overcome the limitations characterizing estimators based on RBD assumption, several Bayesian approaches have been proposed in the last decade in literature. Most techniques are based on GPR, see [14] for a detailed description. Typically in GPR approaches each joint is treated individually and modeled as a single Gaussian Process. More precisely, when considering the \( i \)-th joint, it is assumed that the output measurements \( y_i = \{\tau_i(t_1), \ldots, \tau_i(t_{N_{TR}})\} \) are generated by the following probabilistic model

\[
y_i = \begin{bmatrix} f_i(x(t_1)) \\ \\
\vdots \\
 f_i(x(t_{N_{TR}})) \\
\end{bmatrix} + \begin{bmatrix} e_i(t_1) \\ \\
\vdots \\
 e_i(t_{N_{TR}}) \\
\end{bmatrix} = f_i(X) + e_i(X),
\]

where \( e_i(X) \) is i.i.d. Gaussian noise with standard deviation \( \sigma_{n_i} \), and \( f_i(X) \) is an unknown function defined as a Gaussian Process, namely \( f_i(X) \sim N(m_i(X), K_i(X, X)) \), being \( m_i \) and \( K_i(X, X) \), respectively, mean and covariance. In particular the matrix \( K_i(X, X) \), called also kernel matrix, is defined through a kernel function \( k_i(\cdot, \cdot) \), i.e., the element in \( h \)-th row and \( j \)-th column is equal to \( k_i(x(t_h), x(t_j)) \) (see [14] for a discussion on kernel functions).

When there is no prior knowledge about the model, the GPR prior can be assigned in a data driven fashion, based on similarity assumptions between data. In GPR a common choice consists in assuming \( m_i(\cdot) = 0 \) and adopting a Radial Basis Function (RBF) as kernel, namely, to assume that the outputs distribution is stationary with respect to the input locations. It is well known that RBF kernels can approximate any continuous functions, thus providing a valid tool to obtain accurate estimates of \( \tau_i \) directly from data. RBF kernel has been successfully applied in several robotics applications,
see for example [11]. However, typically estimators based on RBF kernels well approximate the inverse dynamics only in a neighborhood of the training input locations, exhibiting poor performance in terms of generalization properties. Several strategies have been designed in order to limit the computational complexity and to increase the generalization, see for example [10]. However, when considering robots with considerable degrees of freedom, it is still hard to design inverse dynamics estimators with remarkable generalization properties directly from data, i.e., without exploiting a-priori knowledge.

In case a RBD model is given, starting from (2), and modeling $w$ as a Gaussian variable with mean $\bar{w}$ and covariance $\Sigma_w$, it is possible to derive a linear kernel that inherits all the positive aspects of the RBD estimators, but acting in a Bayesian framework, namely considering uncertainties and noises. Specifically, let $i$ be again the index of the considered joint, then

$$m_i(X) = \Phi_i(X)\bar{w}$$

$$k_i(x(t_h), x(t_j)) = \phi_i(x(t_h))\Sigma_w\phi_i^T(x(t_j)), \quad (4)$$

where $\Phi_i(X)$ is the matrix collecting all the rows $\phi_i(x(t_j))$, $j = 1, \ldots, N_{TR}$.

The above kernel can be used alone, leading to the so called Parametric Prior (PP) estimators, or together with a data driven kernel, leading to the so called Semi Parametric (SP) prior estimators. In the latter case, when adopting a RBF kernel as data driven kernel, we have

$$k_i(x(t_h), x(t_j)) = \phi_i(x(t_h))\Sigma_w\phi_i^T(x(t_j)) + k_{RBF}(x(t_h), x(t_j)), \quad (5)$$

see for example [9] and [18]. The rationale behind the use of kernel in (5) is the following: the first term allows to exploit the prior knowledge coming from the RBD, thus providing good generalization properties, while $k_{RBF}(\cdot, \cdot)$ improves estimate in a neighborhood of the training locations, compensating for possible errors due to model bias or un-modeled elements, like complex friction behaviors.

We remark that estimators based on both (4) and (5) are model based estimators, given that their kernel functions are derived starting from (2).

III. PROPOSED APPROACH: PHYSICALLY INSPIRED POLYNOMIAL KERNEL

As highlighted in the previous Section, in general, data driven estimators exhibit poor generalization properties. To achieve good generalization performance it is necessary to include in the definition of the prior information coming from RBD models, that, however, might introduce bias or be hard to derive. The goal of this section is to propose a novel kernel that allows to estimate the inverse dynamics directly from data, preserving the fact of having good generalization properties and high data accuracy.

This Section is organized as follows. Firstly we states Proposition [1] that characterizes the inverse dynamics from the functional analysis point of view. Given the type of each joint, i.e. prismatic or revolute, Proposition [1] defines a transformation of the input $x$ where the inverse dynamics is a polynomial function. Then we briefly review the Multiplicative Polinomial Kernel, recently introduced in [15], which represents a redefinition of the classical polynomial kernels used in the literature. Finally we define the proposed kernel function, named Geometrically Inspired Polynomial kernel.

A. Polynomial characterization of the rigid-body model

In the following we restrict our study to manipulators where each joint is either revolute or prismatic; more complex joint types can be viewed as a combination of the previous two ones. Let $N_r$ and $N_p$ be the number of revolute and prismatic joints, respectively, where $N_r + N_p = n$, and let us denote by $I_r = \{i_{r_1}, \ldots, i_{r_{N_r}}\}$ and $I_p = \{i_{p_1}, \ldots, i_{p_{N_p}}\}$ the sets containing, respectively, the revolute and prismatic joint indexes.

We start our analysis by introducing the following vectors

$$q_c = [\cos(q_{i_{r_1}}), \ldots, \cos(q_{i_{r_{N_r}}})] \in \mathbb{R}^{N_r},$$

$$q_s = [\sin(q_{i_{r_1}}), \ldots, \sin(q_{i_{r_{N_r}}})] \in \mathbb{R}^{N_r},$$

$$q_p = [q_{i_{p_1}}, \ldots, q_{i_{p_{N_p}}}] \in \mathbb{R}^{N_p}.$$

In the following we denote by $q_c$ the element in $q_c$ associated to joint $i_{r_1}$, i.e. $\cos(q_{i_{r_1}})$ (similar definitions hold for $q_s$ and $q_p$). For later convenience we define also $q_{cs} \in \mathbb{R}^{2N_r}$, the vector obtained concatenating $q_c$ and $q_s$. In addition by $\dot{q}_v$ we denote the vector stacking together the elements of the set

$$\{\dot{q}_v, \dot{q}_v\}, \quad 1 \leq i \leq n, \quad i \leq j \leq n,$$

that is, the set containing all the possible pairwise products of components of $\dot{q}_v$. Notice that $\dot{q}_v \in \mathbb{R}^{n(n+1)/2}$.

Finally, we introduce a compact notation to identify a particular set of inhomogeneous polynomial functions. Let $\alpha$ be the vector containing the $m$ variables $a_1, \ldots, a_m$. We denote by $F_{[p]}(\alpha_{[p]})$ the set of polynomial functions of degree not greater than $p$ defined over the variables in $\alpha$, such that each variable $a_i$ appears with degree not greater than $p_i$. Similar definitions hold in case the input sets accounts for more input vectors.

Now we consider the transformation $F : \mathbb{R}^{3n} \to \mathbb{R}^\gamma$ where $\gamma = 2N_r + N_p + (n+1)/2 + n$, which maps the input location $x$ into the element $\bar{x} \in \mathbb{R}^\gamma$ defined as

$$\bar{x} = [q_{c}, q_{s}, q_{p}, \dot{q}_v, \ddot{q}_v].$$

(6)

We have the following result.

**Proposition 1:** Consider a manipulator with $n + 1$ links and $n$ joints, divided in $N_r$ revolute joints and $N_p$ prismatic joints, subject to $n = N_r + N_p$. Then, the inverse dynamics of each joint obtained through the rigid body model in (1) belongs to $\mathbb{F}^{(2n+1)}(q_{c}, q_{s}, q_{p}, \dot{q}_v, \ddot{q}_v)$, where $\tau_i(\cdot)$ is a polynomial function in $\bar{x}$, of degree not greater than $2n + 1$, such that: (i) each element of $q_{c}$, $q_{s}$ and $q_{p}$ appear with degree not greater than 2, and (ii) each element of $\dot{q}_v$ and $\ddot{q}_v$ appear with degree not greater
than 1. Moreover for any monomial of the aforementioned polynomial the sum of the $q_{cs}$ and $q_{sb}$ degrees is equal or lower than two, namely it holds
\[
\text{deg} (q_{cs}) + \text{deg} (q_{sb}) \leq 2.
\]
The proof is reported in the Appendix.

B. Multiplicative Polynomial Kernel

From a functional analysis point of view, Proposition 1 states that the inverse dynamics function defined through a rigid-body model belongs to a finite dimensional space. In particular, the set of basis functions accounts for all the possible monomials in $\mathbb{P}_{(2n+1)} \{ q_{c(2)}, q_{s(2)}, q_{p(2)}, \dot{q}_{v(1)}, \ddot{q}_{v(1)} \}$. Unfortunately, the cardinality of this set grows rapidly with the input dimension $d$ and the polynomial degree $p$. For instance consider a polynomial function with degree $p$, in the generic $d$-th dimensional input vector $x$. Then $\phi_{pl(p)}(x)$, the row vector containing all the possible monomials in $\mathbb{P}_{(p)}(x)$, has dimension $(d+p)$.

\[
\phi_{pl(p)}(x) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ \cdots \\ x_1^p \\ x_2 \\ \cdots \\ x_2^p \\ \vdots \\ x_d \\ \cdots \\ x_d^p \end{pmatrix}, \quad (7)
\]

where $\sigma_i$ are the kernel hyperparameters, see [14] (chapter 4.2.2).

An elegant and compact solution that allows to overcome this problem consists in assuming that the target function $\tau_i(\cdot)$ belongs to the Reproducing Kernel Hilbert Space (RKHS) associated to a polynomial kernel, see [14]. More precisely, when considering the space of inhomogeneous polynomials defined on the components of $x \in \mathbb{R}^d$, with maximum degree $p$, the polynomial kernel is classically defined as
\[
k_{pl(p)}(x(t_h), x(t_j)) = \begin{pmatrix} \sigma_p^2 + x^T(t_h) \Sigma_p x(t_j) \end{pmatrix}^p, \quad (8)
\]

where $\sigma_p > 0$ and $\Sigma_p > 0$ are the kernel hyperparameters, see [14] (chapter 4.2.2).

Unfortunately, as highlighted in [14], the kernel function in (8) is not widely used in regression problems, since it is prone to overfitting, in particular when considering high dimensional inputs and $p > 2$, that is exactly the situation identified in Proposition 1.

A valid alternative to (8) is represented by MPK, recently introduced in [15]. When considering the space of inhomogeneous polynomial with maximum degree $p$ the MPK is defined as the product of $p$ linear kernels,
\[
k_{pl(p)}(x(t_h), x(t_j)) = \prod_{s=1}^{s=p} \begin{pmatrix} \sigma_s^2 + x^T(t_h) \Sigma_s x(t_j) \end{pmatrix}, \quad (8)
\]

where, for $s = 1, \ldots, p$, the $\Sigma_s \in \mathbb{R}^{d \times d}$ matrices are distinct diagonal matrices. The diagonal elements, together with the parameters $\sigma_s^2$, $s = 1, \ldots, p$, compose the hyperparameters set, and they are constrained to be equal or greater than zero, i.e. $\sigma_s^2 \geq 0$, $\Sigma_s \geq 0$, $s = 1, \ldots, p$.

Observe that the RKHSs identified by (8) and (7) contain the same basis functions. However, as discussed in [15], (8) is equipped with a richer set of parameters that allows to better select the monomials that really influence the system output. In [15] it has been shown that hyperparameters can be estimated from data by marginal-likelihood optimization, increasing significantly the prediction capability of the model and reducing the risk of overfitting.

C. Geometrically inspired polynomial kernel for robot inverse dynamics

In this subsection we describe the GIP kernel we propose to model the robot inverse dynamics. Our approach requires minimal information, since we assume to know only the joints type. We assume each joint torque to be described by a zero mean Gaussian process and, driven by Proposition 1, we model the inverse dynamics as a polynomial in the input space $\bar{x}$ in (6). In order to comply with the constraints on the maximum degree of each term we adopt a kernel function given by the product of $N_r + N_p + 1$ kernels of the type defined in equation (8), where

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

The resulting kernel for the $i$-th joint is
\[
k_i(x(t_h), x(t_j)) = k_{cs}(q_{cs}(t_h), q_{cs}(t_j)) k_{p}(q_p(t_h), q_p(t_j)) \bar{k}_{pl(p)}(q_{av}(t_h), q_{av}(t_j)), \quad (9)
\]

with
\[
k_{cs}(q_{cs}(t_h), q_{cs}(t_j)) = \prod_{b=1}^{b=N_r} \bar{k}_{pl(2)}(q_{csb}(t_h), q_{csb}(t_j)),
\]
\[
k_{p}(q_p(t_h), q_p(t_j)) = \prod_{b=1}^{b=N_p} \bar{k}_{pl(2)}(q_{pb}(t_h), q_{pb}(t_j)).
\]

In Figure 1 we reported a schematic representation of the GIP kernel.
We have tested the novel proposed approach both in a simulated environment and in a real environment. Regarding technical aspects, we have implemented all the algorithms we have considered in Python. In order to speed up algebraic operations and training procedures we largely exploited the technical aspects, we have implemented all the algorithms considered are GIP, RBF and NN, the fully connected neural network (on the right). Results are plotted in logarithmic scale.

IV. EXPERIMENTAL RESULTS

We have tested the novel proposed approach both in a simulated environment and in a real environment. Regarding technical aspects, we have implemented all the algorithms we have considered in Python. In order to speed up algebraic operations and training procedures we largely exploited the functionalities provided by Pytorch [19]. The code[1] and the datasets[1] are publicly available.

A. Simulated SCARA robot

In order to evaluate the benefits of the GIP kernel we first tested the proposed approach in a simulated environment. We considered a SCARA robot, more precisely an AdeptOne Robot. The SCARA is a 4 degrees of freedom (DOF) robot manipulator, with three revolute joints (joint 1, 2 and 4) and a prismatic joint (joint 3). As far as data generation is concerned, joint torques have been computed through Eq. (1), assuming complete knowledge of the trajectories followed by the joints. Equation (1) has been derived using the python package SymPyBotics[2].

1) Estimation accuracy: In the first experiment we have tested estimators accuracy. The proposed approach has been compared with respect to both model based and data driven estimators. As far as model based estimators are concerned, we consider two options, i.e., PP and SP kernel based estimators, where the model based component is defined as in Eq. (1) and computed assuming the nominal kinematic parameters provided by the manufacturer. To account for behaviors due to model bias, the kinematic parameters of the model generating data have been varied around the nominal values, so that the model based component of the PP and SP kernel is computed assuming a model different from the one generating data. Perturbations have been modeled as uniform random variables, ranged between \([-0.05, 0.05]\) m for lengths, and \([-5, 5]\) deg for angles. Instead, as far as data driven approaches are concerned, a RBF kernel based estimator and a neural network have been tested. The neural network is a fully connected network with two hidden layers, each of which is composed by 400 sigmoids. The obtained training and test dataset are composed by 2000 samples.

The GPR based estimators have been trained by Marginal Likelihood maximization, while the optimization of neural network parameters has been performed considering the Mean Squared Error (MSE) as loss function, defined as

\[
MSE(y, \hat{y}) = \frac{1}{N_{TR}} \sum_{j=1}^{N_{TR}} (y_j - \hat{y}_j)^2.
\]

Performance are compared by Normalized Mean Squared Error (nMSE) in the test set, defined as

\[
nMSE(\tau, \hat{\tau}) = \frac{1}{N_{TR}} \sum_{j=1}^{N_{TR}} (\tau_j - \hat{\tau}_j)^2 / Var(\tau).
\]

Both for GPR and the neural network we have used Adam as optimizer [20].

In Figure 2, we have plotted the obtained nMSEs through a boxplot. Results show that the proposed approach outperforms other data driven estimators, which are not able to learn accurately the inverse dynamics of the SCARA robot using just 2000 samples. Indeed, except for joint 4, the nMSEs of RBF kernel based estimator and of neural network estimator are in most of the trials higher than 10%. Instead, the GIP kernel based estimator provides accurate estimates, as proven by nMSEs values, that are always below 1%, with the exception of joint 4, where two outliers are present, probably due to training inputs not sufficiently exciting. Moreover, GIP kernel based estimator performs similarly to the model based approaches. Actually in joint 2 and 3 the proposed approach outperforms PP kernel based estimator, whose performance are affected by model bias. Finally, results confirm also validity of semi parametric schemes, proving that addition of a data driven component can compensate for model bias, given that SP kernel based estimator outperforms PP. Anyway we highlight that in hybrid schemes data driven component might not be
effective in compensating for model bias, in particular when performance of data driven estimator are low, as proven by the \( nMSEs \) in joint 3, where the GIP kernel based estimator is more accurate than SP.

2) Data efficiency: In the second experiment we have tested the data efficiency of different estimators. Since our focus is on comparing data driven approaches, model bias has not been considered, in favor of greater results interpretability. The GIP kernel based estimator is compared with the other data driven estimators, and also with the PP kernel based approach. In this ideal scenario, where data are generated with the robot nominal parameters, performance of PP might be considered as baseline of an optimal solution.

The experiment is composed by a training and a test simulation, with joint trajectories generated as in the previous experiment; each simulation accounts for 4000 samples. Results are reported in Figure 3 where we have plotted the evolution of the Global Mean Squared Error (GMSE) as function of the number of training samples used to train and derive estimators. The GMSE is defined as the sum of the \( MSE(\tau_i, \hat{\tau}_i)\)s of the four joints. The evolutions of the errors show that the proposed solution outperforms the other data driven estimators in terms of both accuracy and data efficiency, given that its GMSE is lower and decreases faster. As in the previous experiment, GIP kernel based estimator behaves more similarly to the model based approach than to the other data driven solutions, proving its data efficiency.

B. UR10 robot

A Universal Robots UR10 has been used to test the proposed approach in a real setup. The UR10 robot is a 6 DOF collaborative manipulator, where all the joints are revolute. This robot is not equipped with joint torque sensors, but one can directly measure the motor currents \( i \). Based on this knowledge and, assuming that the behaviors due to elasticity are negligible, i.e. \( \theta = K_r q \), where \( \theta \) contains the motor angles and \( K_r \) is the diagonal matrix of the gear reduction ratio, the inverse dynamics in (11) can be rewritten as

\[
K_{eq}i = B_{eq}(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) + F_v \dot{q} + F_c sign(\dot{q}),
\]

where \( F_v + F_c sign(\dot{q}) \) accounts for the motors frictions and \( B_{eq}(q) = B(q) + K_r^2 B_m \), with \( B_m \) equal to the diagonal matrix of the rotor inertias; the \( K_{eq} \) matrix is defined as \( K_r K_r \), where \( K_r \) is the diagonal matrix containing the torque-current coefficients of the motors.

The interface with the robot is based on ROS [21], through the ur_modern_drive\(^4\) and data are acquired with a sampling time of \( 8 \cdot 10^{-3} \) sec. The driver provides joints positions, velocities and currents, while accelerations are computed through causal numerical differentiation. The dataset collected is publicly available, and it has been designed in order to stress generalization properties. With respect to the dataset proposed in [5], [6] and other works, the robot workspace is explored more extensively, and it is not composed of repetitive movements. The training set accounts for 40000 samples, collected through a random exploration of the robot workspace, requiring the end-effector to reach a series of random points with variable velocity. The test dataset instead is composed of two type of trajectories, for a total number of 25312 points. 22324 points have been collected through a random exploration similar to the one described for the training dataset. The remaining samples come from the trajectory obtained requiring the end-effector to track a circle of radius 30 cm at a tool speed of 30 mm/s.

The estimators we have tested and the optimization procedure are the same of the previous experiment. Due to the higher complexity of the UR10 inverse dynamics, the number of hidden units of the neural network has been increased to 600. Instead, regarding the practical implementation of the GPR estimators, to deal with the computational burden and memory requirements of the GPR, we downsampld the 40000 training samples with a constant step, obtaining 4000 samples. The kinematic parameters considered in the derivation of the model driven components are the nominal values provided by the manufactures.

The results obtained in the real setup, and reported in Figure 4 confirm the behaviors obtained in the simulative setup. The proposed approach outperforms the other data driven estimators in all the joints, confirming that data efficiency is crucial to derive inverse dynamics estimators with good generalization properties. GIP performance are close to the ones of the model based estimators, and in joints 5 and 6 the proposed approach slightly outperforms the PP estimator, that, as explained before, might be affected by model errors. SP performance confirm that model errors can be compensated by the data driven component, even though, as proven by the \( nMSE \) in joint 6, the improvement might not be so significant when data driven estimates are not accurate. Finally, we remark that the \( nMSEs \) obtained by

\(^4\)https://github.com/ThomasTimm/ur_modern_driver
the PP, SP and GIP estimators are close to the limit imposed by the signal to noise ratio. Indeed we quantified a noise variance approximately equal to $0.03[A^2]$, leading to a ratio between the noise variance and the output variance equal to $[0.0125, 0.0018, 0.0037, 0.1607, 0.2528, 0.2637]$. These values are close to the nMSEs obtained, except for the first link. This issue has already been observed in [22], where experiments in a similar set up showed that currents at low velocities are corrupted by non Gaussian noise, limiting significantly estimation performance in the first joint.

V. CONCLUSION

In this paper we introduced a novel polynomial kernel to deal with the data driven inverse dynamics identification. Compared to other data driven approaches, the proposed kernel based estimator, called GIP kernel, is more data efficient. As proven by experiments in a simulated environment and in a real system, this property allows to derive accurate inverse dynamics estimators directly from data, without the need of a priori knowledge about the model and using a small amount of data. Indeed numerical results show that GIP kernel based estimator exhibits behaviors similar to the ones of model based approaches, in terms of accuracy, generalization and data efficiency. However, compared to model based solutions, the proposed approach has two main advantages. The first is that, since our algorithm estimates the inverse dynamics directly from data, it is not affected by model bias. Secondly our algorithm is convenient from an implementation point of view, given its generality and hence the possibility of applying the same approach to different physical systems. As future plans, we envision to extend the proposed approach to the forward dynamics learning, and design Reinforcement Learning algorithms based on GIP kernels.

APPENDIX

We prove Proposition 1 by inspection, analyzing individually all the terms in (1), i.e., the $B(q)\dot{q}$ and $C(q, \dot{q})\dot{q}$ contributions and the gravity term $g(q)$. Firstly, we provide a characterization of the elements of $B(q)$ as polynomials in $q$, $q_\alpha$ and $q_\beta$. It is known that the inertia matrix is given by

$$B(q) = \sum_{i=1}^{n} m_i J_i^T J_i + J_i^T R_{i}^0 I_i^T R_{i}^0 J_i,$$

where $m_i$ and $I_i^0$ are respectively the $i$-th link mass and inertia matrix, expressed in a reference frame (RF) solidal with the $i$-th link. $J_i$ and $J_{\omega i}$ are, respectively, the linear and angular Jacobians of the $i$-th RF, i.e. $\dot{c}_i = J_i \dot{q}$ and $\omega_i = J_{\omega i} \dot{q}$, where $c_i$ denotes the position of the center of mass of the $i$-th link, while $\omega_i$ denotes the angular velocity of the $i$-th RF. To derive the $J_i$ and $J_{\omega i}$ expressions we need to introduce some notations regarding the kinematic. Adopting the Denavit-Hartenberg (DH) convention, the $R_i^{-1}$ and $I_i^{-1}$ variables, which denote respectively the $i$-th RF orientation and translation with the respect to the previous RF, are given by the following expressions

$$R_i^{-1} = R_x(\theta_i) R_z(\alpha_i),$$
$$I_i^{-1} = [0,0,d_i]^T + R_z(\theta_i) [a_i,0,0]^T,$$

where $R_x$ and $R_z$ are the elementary rotation matrices around the $x$ and $z$ axis, while $a_i$ and $\alpha_i$ are two constant geometrical parameters, see [16]. The definitions of $d_i$ and $\theta_i$ depend on the joint interconnecting the $i$-th link with the previous link. When the joint is revolute, $d_i$ is constant and $\theta_i = \theta_0 + q_i$, thus the only terms that depend on $q$ are $\cos(q_i)$ and $\sin(q_i)$ contained in $R_i^{-1}$. Referring to the polynomial notation previously introduced, we can write that the elements of $R_i^{-1}$ functions in $\mathbb{P}(1)\left(\cos(q_i),\sin(q_i)\right)$. In case the joint is prismatic, $\theta_i$ is constant and $d_i = d_0 + q_i$, and consequently the only $q$ dependent terms are in $I_i^{-1}$. In particular the elements of $I_i^{-1}$ belong to $\mathbb{P}(1)\left(q_i\right)$.

The $J_{\omega i}$ matrix projects all the angular velocities $\omega_i^{-1}$ in the base frame. Adopting the DH convention, $\omega_i^{-1} = \lambda_i [0,0,\dot{q}_i]^T$, with $\lambda_i = 1$ if the joint is revolute, and $\lambda_i = 0$ if it is prismatic. Then, summing all the angular velocities projected in the base frame through the $R_i^{-1} = \prod_{b=0}^{i} R_{b}^{-1}$ matrices, and remarking that $\omega_i = \sum_{j=1}^{i} \lambda_j R_{j-1}^{-1} \omega_j^{-1}$, we obtain

$$\omega_i = \begin{bmatrix} R_0^0 & 0 \\ 0 & \lambda_1 \end{bmatrix}, \ldots, \begin{bmatrix} R_{i-1}^0 & 0 \\ 0 & \lambda_{i-1} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & \lambda_i \end{bmatrix}, \begin{bmatrix} 0 & (3, n-i) \end{bmatrix} \dot{q},$$

where $0 (3, n-i)$ is a $3 \times (n-i)$ matrix containing only zero elements. The last equation implies

$$J_{\omega i} = \begin{bmatrix} R_0^0 & 0 \\ 0 & \lambda_1 \end{bmatrix}, \ldots, \begin{bmatrix} R_{i-1}^0 & 0 \\ 0 & \lambda_{i-1} \end{bmatrix}, \begin{bmatrix} 0 & (3, n-i) \end{bmatrix}.$$

Exploiting the properties of the rotation matrices, we obtain that

$$R_i^0 J_{\omega i} = \begin{bmatrix} R_i^0 & 0 \\ 0 & \lambda_i \end{bmatrix}, \ldots, \begin{bmatrix} R_{i-1}^i & 0 \\ 0 & \lambda_{i-1} \end{bmatrix}, \begin{bmatrix} 0 & (3, n-i) \end{bmatrix} \dot{q}.$$
Let \( \{I_r \leq i\} \) be the set containing the revolute joint indexes lower or equal than \( i \), and let \( q_c, (\{I_r \leq i\}) \) be the corresponding subset. Recalling that the elements of \( R_{i-1}^k \) are functions in \( \mathbb{P}_{(1)} \left( \cos(q_1), \sin(q_1) \right) \) with maximal degree one, and that \( R_{i}^k = \prod_{k=j}^{i} R_{i}^j \) with \( j > k \), it follows that the \( J^T R_i R_i J_i \) elements belong to \( \mathbb{P}_{(2)} \left( \{I_r \leq i\} \right) \). And for each joint the torque is a function in each monomial of the following constraint holds
\[
deg (q_{c_i}) + deg (q_{s_i}) \leq 2, \tag{10}
\]

To derive a similar characterization of the \( J_i \) elements we analyze the \( c_i \) expression. The position of the \( i-th \) center of mass in the base frame is \( c_i = \sum_{i=1}^{n=1} R_{i-1}^0 R_{i-1}^1 \), implying that the \( c_i \) elements are functions in \( \mathbb{P}_{(i)} \left( q_{c_i}, (\{I_r \leq i\}) \right) \), \( q_{s_i}, (\{I_r \leq i\}) \), \( q_p, (\{I_p \leq i\}) \), and in each monomial the \( q_c \), \( q_s \), and \( q_p \) degrees are constrained by the following inequality
\[
deg (q_{c_i}) + deg (q_{s_i}) \leq 1. \tag{11}
\]
Since \( \dot{c}_i = \dot{J}_i q_i \) and since the derivative of \( \cos (q_j) \), \( \sin (q_j) \) does not increase the degree of these terms when inequality \( \{11\} \) holds, it follows that the \( J_i \) elements belong to the same functional space of \( c_i \). Consequently the elements of \( J_i^T J_i \) are functions in \( \mathbb{P}_{(2)} (q_{c_i}, (\{I_r \leq i\})_{(2)}) \), \( q_{s_i}, (\{I_r \leq i\}), q_p (\{I_p \leq i\})_{(2)} \). Hence, before the nominal monomial the \( q_{c_i} \) and \( q_{s_i} \) degrees are subject to inequality \( \{10\} \).

Given the characterization of \( J_i^T J_i \) and \( J_i^T R_i^0 R_i^1 R_i^0 J_i \), we obtain that the \( B (q) \) elements are functions in \( \mathbb{P}_{(2n)} (q_{c_1}, q_{s_1}, q_{p_2}) \), where in each monomial the \( q_{c_1} \) and \( q_{s_1} \) degrees are subject to inequality \( \{10\} \). Then the \( B (q \dot{q}) \) are functions in \( \mathbb{P}_{(2n+1)} (q_{c_1}, q_{s_1}, q_{p_2}, \dot{q}_1, \dot{q}_1) \).

As reported in \[16\], the \( i-th \) element of the \( C (q, \dot{q}) \) \( \dot{q} \) product is equal to
\[
\sum_{j=1}^{n} c_{ij} \dot{q}_j = \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial b_{ij}}{\partial q_k} - \frac{1}{2} \frac{\partial b_{ij}}{\partial q_k} \dot{q}_k \dot{q}_j .
\]

Since the \( B (q) \) elements belong to \( \mathbb{P}_{(2n)} \), \( q_{c_1}, q_{s_1}, q_{p_2} \), and \( \{10\} \) holds true, also the \( b_{ij} \) partial derivatives belong to \( \mathbb{P}_{(2n)} \), \( q_{c_1}, q_{s_1}, q_{p_2} \), with \( \{10\} \) satisfied. Indeed, for each monomial in \( b_{ij} \), the derivation respect to \( q_p \) decreases the degree by one, while the derivation respect to \( q \) or \( q_s \) does not alter the monomial degree. Then we obtain that the elements of \( C (q, \dot{q}) \) \( \dot{q} \) functions are in \( \mathbb{P}_{(2n+1)} (q_{c_1}, q_{s_1}, q_{p_2}, \dot{q}_1, \dot{q}_1) \).

Regarding \( g (q) \), we observe that the \( i-th \) element is given by \( \partial U / \partial q_i \), where by definition the potential energy \( U = \sum_{i=1}^{n} g_0^i c_i \), with \( g_0 \) denoting the vector of the gravitational acceleration. Then the elements of \( g (q) \) are functions in the same space of the \( J_i \) elements.

To conclude the proof we just need to sum all the contributions and to note that for each joint the torque is a function in
\[
\mathbb{P}_{(2n+1)} (q_{c_1}, q_{s_1}, q_{p_2}, \dot{q}_1, \dot{q}_1) \text{, with each monomial satisfying } \{10\}.
\]

**References**

[1] P. Kholiya and T. Kanade, “Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators,” *IJRR*, vol. 7, no. 1, pp. 18 – 28, February 1988.

[2] P. Poignet and M. Gautier, “Nonlinear model predictive control of a robot manipulator,” in *AMC*, March 2000, pp. 401–406.

[3] B. Siciliano and L. Villani, *Robot Force Control*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[4] S. Haddadin, A. D. Luca, and A. Albu-Schaffer, “Robot collisions: A survey on detection, isolation, and identification,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, Dec 2017.

[5] S. Vijayakumar and S. Schaal, “Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space,” in *ICML*, 2000, pp. 1079–1086.

[6] A. S. Polycoros, L. Nalpanidis, and V. Krger, “Real-time deep learning of robotic manipulator inverse dynamics,” in *IROS*, Sep. 2015, pp. 3442–3448.

[7] E. Rueckert, M. Nakatensu, S. Tosatto, and J. Peters, “Learning inverse dynamics models in o(n) time with lstm networks,” in *Humanoids*, Nov 2017, pp. 811–816.

[8] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schlkopf, “Learning inverse dynamics: A comparison,” 01 2008, pp. 13–18.

[9] D. Nguyen-Tuong, M. Seeger, and J. Peters, “Local gaussian process regression for real time online model learning and control,” in *NIPS*, Max-Planck-Gesellschaft. Red Hook, NY, USA: Curran, June 2009, pp. 1193–1200.

[10] M. P. Deisenroth and C. E. Rasmussen, “Plics: A model-based and data-efficient approach to policy search,” in *ICML*, 2011.

[11] S. Levine, Z. Popovic, and V. Koltun, “Nonlinear inverse reinforcement learning with gaussian processes,” in *NIPS*, 2011, pp. 19–27.

[12] L. Hewing, A. Liniger, and M. Zeilinger, “Cautious nmpc with gaussian process dynamics for autonomous miniature race cars,” 06 2018, pp. 1341–1348.

[13] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, Jan. 2006.

[14] A. D. Libera, R. Carli, and G. Pillonetto, “A novel multiplicative polynomial kernel for volterra series identification,” *arXiv:1905.07960*, May 2019.

[15] P. Sanz, “Robotics: Modeling, planning, and control (siciliano, b. et al; 2009) [on the shelf],” *IEEE Robotics Automation Magazine*, vol. 16, no. 4, pp. 101–101, December 2009.

[16] C. a. D. Sousa and R. Cortesão, “Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach,” *Int. J. Rob. Res.*, vol. 33, no. 6, pp. 931–944, May 2014. [Online]. Available: [http://dx.doi.org/10.1177/0278364913514587](http://dx.doi.org/10.1177/0278364913514587).

[17] D. Nguyen-Tuong and J. Peters, “Using model knowledge for learning inverse dynamics,” in *ICRA*, May 2010, pp. 2673–2682.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[19] P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2015.

[20] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[21] A. D. Libera, E. Tosello, G. Pillonetto, S. Ghidoni, and R. Carli, “Proprioeptive robot collision detection through gaussian process regression,” *arXiv:1905.08689*, May 2019.