2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Incremental Semiparametric Inverse Dynamics Learning

Raffaello Camoriano*†‡◇, Silvio Traversaro†,
Lorenzo Rosasco◇‡, Giorgio Metta†, and Francesco Nori†

Abstract— **This paper presents a novel approach for incremental semiparametric inverse dynamics learning. In particular, we consider the mixture of two approaches: Parametric modeling based on rigid body dynamics equations and nonparametric modeling based on incremental kernel methods, with no prior information on the mechanical properties of the system. The result is an incremental semiparametric approach, leveraging the advantages of both the parametric and nonparametric models. We validate the proposed technique learning the dynamics of one arm of the iCub humanoid robot.**

I. INTRODUCTION

In order to control a robot a model describing the relation between the actuator inputs, the interactions with the world and bodies accelerations is required. This model is called the *dynamics* model of the robot. A dynamics model can be obtained from first principles in mechanics, using the techniques of rigid body dynamics (RBD) [1], resulting in a *parametric model* in which the values of physically meaningful parameters must be provided to complete the fixed structure of the model. Alternatively, the dynamical model can be obtained from experimental data using Machine Learning techniques, resulting in a *nonparametric model*.

Traditional dynamics parametric methods are based on several assumptions, such as rigidity of links or that friction has a simple analytical form, which may not be accurate in real systems. On the other hand, nonparametric methods based on algorithms such as Kernel Ridge Regression (KRR) [2], [3], [4], Kernel Regularized Least Squares (KRLS) [5] or Gaussian Processes [6] can model dynamics by extrapolating the input-output relationship directly from the available data[1]. If a suitable kernel function is chosen, then the nonparametric model is a universal approximator which can account for the dynamics effects which are not considered by the parametric model. Still, nonparametric models have no prior knowledge about the target function to be approximated. Therefore, they need a sufficient amount of training examples in order to produce accurate predictions on the entire input space. If the learning phase has been performed offline, both approaches are susceptible to the variation of the mechanical properties over long time spans, which are mainly caused by temperature shifts and wear. Even the inertial parameters can change over time. For example if the robot grasps a heavy object, the resulting change in dynamics can be described by a change of the inertial parameters of the hand. A solution to this problem is to address the variations of the identified system properties by learning *incrementally*, continuously updating the model as long as new data becomes available. In this paper we propose a novel technique that joins parametric and nonparametric model learning in an incremental fashion.

TABLE I: Summary of related works on semiparametric or incremental robot dynamics learning.

Author, Year	Parametric	Nonparametric
Nguyen-Tuong et al., 2010 [7]	Batch	Batch
Gijsberts et al., 2011 [8]	-	Incremental
Tingfan Wu et al., 2012 [9]	Batch	Batch
De La Cruz et al., 2012 [10]	CAD*	Incremental
Um et al., 2014 [11]	CAD	Batch
Camoriano et al., 2015	Incremental	Incremental

* In [10] the parametric part is used only for initializing the nonparametric model.

Classical methods for physics-based dynamics modeling can be found in [1]. These methods require to identify the mechanical parameters of the rigid bodies composing the robot [12], [13], [14], [15], which can then be employed in model-based control and state estimation schemes.

In [7] the authors present a learning technique which combines prior knowledge about the physical structure of the mechanical system and learning from available data with Gaussian Process Regression (GPR) [6]. Similar approaches are presented in [9], [11]. Both techniques require an offline training phase and are not incremental, limiting them to scenarios in which the properties of the system do not change significantly over time.

In [10] an incremental semiparametric robot dynamics learning scheme based on Locally Weighted Projection Regression (LWPR) [16] is presented, that is initialized using a linearized parametric model. However, this approach uses a fixed parametric model, that is not updated as new data becomes available. Moreover, LWPR has been shown to underperform with respect to other methods (e.g. [8]).

In [8], a fully nonparametric incremental approach for inverse dynamics learning with constant update complexity is presented, based on kernel methods [17] (in particular

*Corresponding author.
†iCub Facility, Istituto Italiano di Tecnologia, Via Morego 30, Genoa 16163, Italy. Email: {raffaello.camoriano, silvio.traversaro, giorgio.metta, francesco.nori}@iit.it
◇LCSL, Istituto Italiano di Tecnologia and Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Email: lrosasco@mit.edu
‡DIBRIS, Università degli Studi di Genova, Via All'Opera Pia, 13, Genoa 16145, Italy.

[1]Note that KRR and KRLS have a very similar formulation, and that these are also equivalent to the techniques derived from Gaussian Processes, as explained for instance in Chapter 6 of [4].

KRR) and random features [18]. The incremental nature of this approach allows for adaptation to changing conditions in time. The authors also show that the proposed algorithm outperforms other methods such as LWPR, GPR and Local Gaussian Processes (LGP) [19], both in terms of accuracy and prediction time. Nevertheless, the fully nonparametric nature of this approach undermines the interpretability of the inverse dynamics model.

In this work we propose a method that is incremental with fixed update complexity (as [8]) and semiparametric (as [7] and [9]). The fixed update complexity and prediction time are key properties of our method, enabling real-time performances. Both the parametric and nonparametric parts can be updated, as opposed to [10] in which only the nonparametric part is. A comparison between the existing literature and our incremental method is reported in Table I. We validate the proposed method with experiments performed on an arm of the iCub humanoid robot [20].

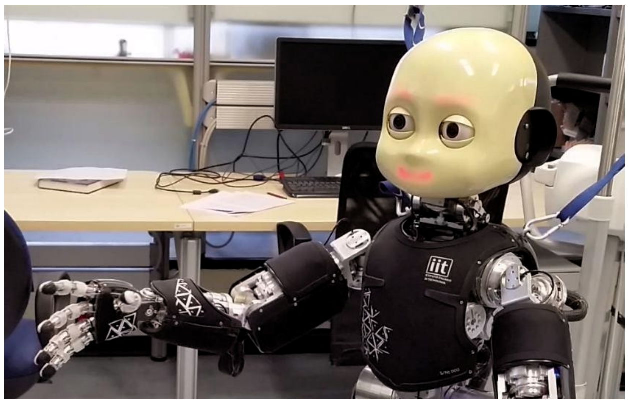


Fig. 1: iCub learning its right arm dynamics.

The article is organized as follows. Section II introduces the existing techniques for parametric and nonparametric robot dynamics learning. In Section III, a complete description of the proposed semiparametric incremental learning technique is introduced. Section IV presents the validation of our approach on the iCub humanoid robotic platform. Finally, Section V summarizes the content of our work.

II. BACKGROUND

A. Notation

The following notation is used throughout the paper.

- The set of real numbers is denoted by $\mathbb{R}$. Let $\mathbf{u}$ and $\mathbf{v}$ be two n-dimensional column vectors of real numbers (unless specified otherwise), i.e. $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, their inner product is denoted as $\mathbf{u}^\top \mathbf{v}$, with "$\top$" the transpose operator.
- The Frobenius norm of either a vector or a matrix of real numbers is denoted by $\| \cdot \|$.
- $I_n \in \mathbb{R}^{n \times n}$ denotes the identity matrix of dimension n; $0_n \in \mathbb{R}^n$ denotes the zero column vector of dimension n; $0_{n\times m} \in \mathbb{R}^{n\times m}$ denotes the zero matrix of dimension $n \times m$.

B. Parametric Models of Robot Dynamics

Robot dynamics parametric models are used to represent the relation connecting the geometric and inertial parameters with some dynamic quantities that depend uniquely on the robot model. A typical example is obtained by writing the robot inverse dynamics equation in linear form with respect to the robot inertial parameters $\boldsymbol{\pi}$:

$$\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = \Phi(\mathbf{x})\boldsymbol{\pi}, \qquad (1)$$

where: $\mathbf{q} \in \mathbb{R}^{n_{dof}}$ is the vector of joint positions, $\boldsymbol{\tau} \in \mathbb{R}^{n_{dof}}$ is the vector of joint torques, $\boldsymbol{\pi} \in \mathbb{R}^{n_p}$ is the vector of the identifiable (base) inertial parameters [1], $\Phi(\mathbf{x}) \in \mathbb{R}^{n_{dof} \times n_p}$ is the "regressor", i.e. a matrix that depends only on the robot kinematic parameters. In the rest of the paper we will indicate with $\mathbf{x}$ the triple given by $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$. Other parametric models write different measurable quantities as a product of a regressor and a vector of parameters, for example the total energy of the robot [21], the instantaneous power provided to the robot [22], the sum of all external forces acting on the robot [23] or the center of pressure of the ground reaction forces [24]. Regardless of the choice of the measured variable $\mathbf{y}$, the structure of the regressor is similar:

$$\mathbf{y} = \Phi(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\pi} = \Phi(\mathbf{x})\boldsymbol{\pi}, \qquad (2)$$

where $\mathbf{y} \in \mathbb{R}^t$ is the measured quantity.

The $\boldsymbol{\pi}$ vector is composed of certain linear combinations of the inertial parameters of the links, the *base inertial parameters* [25]. In particular, the inertial parameters of a single body are the mass m, the first moment of mass $m\mathbf{c} \in \mathbb{R}^3$ expressed in a body fixed frame and the inertia matrix $I \in \mathbb{R}^{3\times3}$ expressed in the orientation of the body fixed frame and with respect to its origin.

In *parametric modeling* of robot dynamics, the regressor structure depends on the kinematic parameters of the robot, that are obtained from CAD models of the robot through kinematic calibration techniques. Similarly, the inertial parameters $\boldsymbol{\pi}$ can also be obtained from CAD models of the robot, however these models may be unavailable (for example) because the manufacturer of the robot does not provide them. In this case the usual approach is to estimate $\boldsymbol{\pi}$ from experimental data [15]. To do that, given n measures of the measured quantity $\mathbf{y}_i$ (with $i = 1 \dots n$), stacking (2) for the n samples it is possible to write:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \Phi(\mathbf{x}_1) \\ \Phi(\mathbf{x}_2) \\ \vdots \\ \Phi(\mathbf{x}_n) \end{bmatrix} \boldsymbol{\pi}. \qquad (3)$$

This equation can then be solved in least squares (LS) sense to find an estimate $\hat{\boldsymbol{\pi}}$ of the base inertial parameters. Given the training trajectories it is possible that not all parameters in $\boldsymbol{\pi}$ can be estimated well as the problem in (3) can be ill-posed, hence this equation is usually solved as a *Regularized Least Squares* (RLS) problem. Defining

$$\overline{\mathbf{y}}_n = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_n \end{bmatrix}, \quad \overline{\mathbf{\Phi}}_n = \begin{bmatrix} \Phi(\mathbf{x}_1) \\ \Phi(\mathbf{x}_2) \\ \vdots \\ \Phi(\mathbf{x}_n) \end{bmatrix},$$

the RLS problem that is solved for the parametric identification is:

$$\hat{\boldsymbol{\pi}} = \underset{\boldsymbol{\pi} \in \mathbb{R}^{n_p}}{\operatorname{argmin}} \left(\|\overline{\mathbf{\Phi}}_n \boldsymbol{\pi} - \overline{\mathbf{y}}_n\|^2 + \lambda \|\boldsymbol{\pi}\|^2 \right), \lambda > 0. \quad (4)$$

C. Nonparametric Modeling with Kernel Methods

Consider a probability distribution ρ over the probability space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the input space (the space of the d measured attributes) and $\mathcal{Y} \subseteq \mathbb{R}^t$ is the output space (the space of the t outputs to be predicted). In a nonparametric modeling setting, the goal is to find a function $f^* : \mathcal{X} \to \mathcal{Y}$ belonging to a set of measurable functions $\mathcal{H}$, called *hypothesis space*, such that

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \underbrace{\int_{\mathcal{X} \times \mathcal{Y}} \ell(f(\mathbf{x}), \mathbf{y}) d\rho(\mathbf{x}, \mathbf{y})}_{\mathcal{E}(f)}, \quad (5)$$

where $\mathbf{x} \in \mathcal{X}$ are row vectors, $\mathbf{y} \in \mathcal{Y}$, $\mathcal{E}(f)$ is called *expected risk* and $\ell(f(\mathbf{x}), \mathbf{y})$ is the *loss function*. In the rest of this work, we will consider the squared loss $\ell(f(\mathbf{x}), \mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|^2$. Note that the distribution ρ is unknown, and that we assume to have access to a discrete and finite set of measured data points $S = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ of cardinality n, in which the points are independently and identically distributed (i.i.d.) according to ρ.
In the context of kernel methods [17], $\mathcal{H}$ is a *reproducing kernel Hilbert space* (RKHS). An RKHS is a Hilbert space of functions such that $\exists k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for which the following properties hold:

1) $\forall \mathbf{x} \in \mathcal{X} \quad k_{\mathbf{x}}(\cdot) = k(\mathbf{x}, \cdot) \in \mathcal{H}$
2) $g(\mathbf{x}) = \langle g, k_{\mathbf{x}} \rangle_{\mathcal{H}} \, \forall g \in \mathcal{H}, \mathbf{x} \in \mathcal{X}$,

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ indicates the inner product in $\mathcal{H}$. The function k is a *reproducing kernel*, and it can be shown to be symmetric positive definite (SPD). We also define the kernel matrix $K \in \mathbb{R}^{n \times n} \quad s.t. \quad K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, which is symmetric and positive semidefinite (SPSD) $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, with $i, j \in \{1, \ldots, n\}, n \in \mathbb{N}^+$.

The optimization problem outlined in (5) can be approached empirically by means of many different algorithms, among which one of the most widely used is Kernel Regularized Least Squares (KRLS) [3], [5]. In KRLS, a regularized solution $\hat{f}_\lambda : \mathcal{X} \to \mathcal{Y}$ is found solving

$$\hat{f}_\lambda = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \left(\sum_{i=1}^n \|f(\mathbf{x}_i) - \mathbf{y}_i\|^2 + \lambda \|f\|_{\mathcal{H}}^2 \right), \lambda > 0, \quad (6)$$

where λ is called *regularization parameter*. The solution to (6) exists and is unique. Following the representer theorem [17], the solution can be conveniently expressed as

$$\hat{f}_\lambda(\mathbf{x}) = \sum_{i=1}^n \boldsymbol{\alpha}_i k(\mathbf{x}_i, \mathbf{x}) \quad (7)$$

with $\boldsymbol{\alpha} = (K + \lambda I_n)^{-1} Y \in \mathbb{R}^{n \times t}$, $\boldsymbol{\alpha}_i$ i-th row of $\boldsymbol{\alpha}$ and $Y = \left[\mathbf{y}_1^\top, \ldots, \mathbf{y}_n^\top\right]^\top$. It is therefore necessary to invert and store the kernel matrix $K \in \mathbb{R}^{n \times n}$, which implies $O(n^3)$ and $O(n^2)$ time and memory complexities, respectively. Such complexities render the above-mentioned KRLS approach prohibitive in settings where n is large, including the one treated in this work. This limitation can be dealt with by resorting to approximated methods such as *random features*, which will now be described.

1) Random Feature Maps for Kernel Approximation: The random features approach was first introduced in [18], and since then has been widely applied in the field of large-scale Machine Learning. This approach leverages the fact that the kernel function can be expressed as

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}, \quad (8)$$

where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ are row vectors, $\phi : \mathbb{R}^d \to \mathbb{R}^p$ is a *feature map* associated with the kernel, which maps the input points from the input space $\mathcal{X}$ to a feature space of dimensionality $p \leq +\infty$, depending on the chosen kernel. When p is very large, directly computing the inner product as in (8) enables the computation of the solution, as we have seen for KRLS. However, K can become too cumbersome to invert and store as n grows. A random feature map $\tilde{\phi} : \mathbb{R}^d \to \mathbb{R}^D$, typically with $D \ll p$, directly approximates the feature map ϕ, so that

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} \approx \tilde{\phi}(\mathbf{x}) \tilde{\phi}(\mathbf{x}')^\top. \quad (9)$$

D can be chosen according to the desired approximation accuracy, as guaranteed by the convergence bounds reported in [18], [26]. In particular, we will use random Fourier features for approximating the Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}}. \quad (10)$$

The approximated feature map in this case is $\tilde{\phi}(\mathbf{x}) = \left[e^{i\mathbf{x}\boldsymbol{\omega}_1}, \ldots, e^{i\mathbf{x}\boldsymbol{\omega}_D}\right]$, where

$$\boldsymbol{\omega} \sim p(\boldsymbol{\omega}) = (2\pi)^{-\frac{D}{2}} e^{-\frac{\|\boldsymbol{\omega}\|^2}{2\sigma^2}}, \quad (11)$$

with $\boldsymbol{\omega} \in \mathbb{R}^d$ column vector. The fundamental theoretical result on which random Fourier features approximation relies is Bochner's Theorem [27]. The latter states that if $k(\mathbf{x}, \mathbf{x}')$ is a shift-invariant kernel on $\mathbb{R}^d$, then k is positive definite if and only if it is the Fourier transform of a non-negative measure $p(\omega) \geq 0$. If this holds, by the definition of Fourier transform and after appropriate scaling we can treat $p(\omega)$ as a proper probability distribution, and write

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') = \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i(\mathbf{x} - \mathbf{x}')\boldsymbol{\omega}} d\boldsymbol{\omega}, \quad (12)$$

which can be approximated by performing an empirical average as follows:

$$\begin{aligned} k(\mathbf{x} - \mathbf{x}') &= \mathbb{E}_{\boldsymbol{\omega} \sim p}\left[e^{i(\mathbf{x} - \mathbf{x}')\boldsymbol{\omega}}\right] \approx \\ &\approx \tfrac{1}{D} \textstyle\sum_{k=1}^D e^{i(\mathbf{x} - \mathbf{x}')\boldsymbol{\omega}} = \tilde{\phi}(\mathbf{x}) \tilde{\phi}(\mathbf{x}')^\top. \end{aligned} \quad (13)$$

Therefore, it is possible to map the input data as $\tilde{\mathbf{x}} = \tilde{\phi}(\mathbf{x}) \in \mathbb{R}^D$, with $\tilde{\mathbf{x}}$ row vector, to obtain a nonlinear and nonparametric model of the form

$$\tilde{f}(\mathbf{x}) = \tilde{\mathbf{x}}\tilde{W} \approx \hat{f}_\lambda(\mathbf{x}) = \sum_{i=1}^{n} \boldsymbol{\alpha}_i k(\mathbf{x}_i, \mathbf{x}) \tag{14}$$

approximating the exact kernelized solution $\hat{f}_\lambda(\mathbf{x})$, with $\tilde{W} \in \mathbb{R}^{D \times t}$. Note that the approximated model is nonlinear in the input space, but linear in the random features space. We can therefore introduce the regularized linear regression problem in the random features space as follows:

$$\tilde{W}^\lambda = \underset{\tilde{W} \in \mathbb{R}^{d \times t}}{\operatorname{argmin}} \left(\|\tilde{X}\tilde{W} - Y\|^2 + \lambda \|\tilde{W}\|^2 \right), \lambda > 0, \tag{15}$$

where $\tilde{X} \in \mathbb{R}^{n \times D}$ is the matrix of the training inputs where each row has been mapped by $\tilde{\phi}$. The main advantage of performing a random feature mapping is that it allows us to obtain a nonlinear model by applying linear regression methods. For instance, Regularized Least Squares (RLS) can compute the solution $\tilde{W}^\lambda$ of (15) with $O(nD^2)$ time and $O(D^2)$ memory complexities. Once $\tilde{W}^\lambda$ is known, the prediction $\hat{\mathbf{y}} \in \mathbb{R}^{1 \times t}$ for a mapped sample $\tilde{\mathbf{x}}$ can be computed as $\hat{\mathbf{y}} = \tilde{\mathbf{x}}\tilde{W}^\lambda$.

D. Regularized Least Squares

Let $Z \in \mathbb{R}^{a \times b}$ and $U \in \mathbb{R}^{a \times c}$ be two matrices of real numbers, with $a, b, c \in \mathbb{N}^+$. The Regularized Least Squares (RLS) algorithm computes a regularized solution $W^\lambda \in \mathbb{R}^{b \times c}$ of the potentially ill-posed problem $ZW = U$, enforcing its numerical stability. Considering the widely used Tikhonov regularization scheme, $W^\lambda \in \mathbb{R}^{b \times c}$ is the solution to the following problem:

$$W^\lambda = \underset{W \in \mathbb{R}^{b \times c}}{\operatorname{argmin}} \underbrace{\left(\|ZW - U\|^2 + \lambda \|W\|^2 \right)}_{J(W,\lambda)}, \quad \lambda > 0, \tag{16}$$

where λ is the regularization parameter. By taking the gradient of $J(W, \lambda)$ with respect to W and equating it to zero, the minimizing solution can be written as

$$W^\lambda = (Z^\top Z + \lambda I_b)^{-1} Z^\top U. \tag{17}$$

Both the parametric identification problem (4) and the nonparametric random features problem (15) are specific instances of the general problem (16).

In particular, the parametric problem (4) is equivalent to (16) with:

$$W^\lambda = \hat{\boldsymbol{\pi}}, \quad Z = \overline{\Phi}_n, \quad U = \overline{\mathbf{y}}_n$$

while the random features learning problem (15) is equivalent to (16) with:

$$W^\lambda = \tilde{W}^\lambda, \quad Z = \tilde{X}, \quad U = Y.$$

Hence, both problems for a given set of n samples can be solved applying (17).

E. Recursive Regularized Least Squares (RRLS) with Cholesky Update

In scenarios in which supervised samples become available sequentially, a very useful extension of the RLS algorithm consists in the definition of an update rule for the model which allows it to be incrementally trained, increasing adaptivity to changes of the system properties through time. This algorithm is called Recursive Regularized Least Squares (RRLS). We will consider RRLS with the Cholesky update rule [28], which is numerically more stable than others (e.g. the Sherman-Morrison-Woodbury update rule). In adaptive filtering, this update rule is known as the *QR algorithm* [29].

Let us define $A = Z^\top Z + \lambda I_b$ with $\lambda > 0$ and $B = Z^\top U$. Our goal is to update the model (fully described by A and B) with a new supervised sample $(\mathbf{z}_{k+1}, \mathbf{u}_{k+1})$, with $\mathbf{z}_{k+1} \in \mathbb{R}^b$, $\mathbf{u}_{k+1} \in \mathbb{R}^c$ row vectors.

Consider the Cholesky decomposition $A = R^\top R$. It can always be obtained, since A is positive definite for $\lambda > 0$. Thus, we can express the update problem at step $k+1$ as:

$$\begin{aligned} A_{k+1} &= R_{k+1}^\top R_{k+1} \\ &= A_k + \mathbf{z}_{k+1}^\top \mathbf{z}_{k+1} \\ &= R_k^\top R_k + \mathbf{z}_{k+i}^\top \mathbf{z}_{k+1}, \end{aligned} \tag{18}$$

where R is full rank and unique, and $R_0 = \sqrt{\lambda} I_b$.
By defining

$$\tilde{R}_k = \begin{bmatrix} R_k \\ \mathbf{z}_{k+1} \end{bmatrix} \in \mathbb{R}^{b+1 \times b}, \tag{19}$$

we can write $A_{k+1} = \tilde{R}_k^\top \tilde{R}_k$. However, in order to compute R_{k+1} from the obtained A_{k+1} it would be necessary to recompute its Cholesky decomposition, requiring $O(b^3)$ computational time. There exists a procedure, based on Givens rotations, which can be used to compute R_{k+1} from $\tilde{R}_k$ with $O(b^2)$ time complexity. A recursive expression can be obtained also for B_{k+1} as follows:

$$\begin{aligned} B_{k+1} &= Z_{k+1}^\top U_{k+1} \\ &= Z_k^\top U_k + \mathbf{z}_{k+1}^\top \mathbf{u}_{k+1}. \end{aligned} \tag{20}$$

Once R_{k+1} and B_{k+1} are known, the updated weights matrix W_k can be obtained via back and forward substitution as

$$W_{k+1} = R_{k+1} \setminus (R_{k+1}^\top \setminus B_{k+1}). \tag{21}$$

The time complexity for updating W is $O(b^2)$.

As for RLS, the RRLS incremental solution can be applied to both the parametric (4) and nonparametric with random features (15) problems, assuming $\lambda > 0$. In particular, RRLS can be applied to the parametric case by noting that the arrival of a new sample $(\Phi_r, \mathbf{y}_r)$ adds t rows to $Z_k = \overline{\Phi}_{r-1}$ and $U_k = \overline{\mathbf{y}}_{r-1}$. Consequently, the update of A must be decomposed in t update steps using (20). For each one of these t steps we consider only one row of Φ_r and $\mathbf{y}_r^\top$, namely:

$$\mathbf{z}_{k+i} = (\Phi_r)_i, \quad \mathbf{u}_{k+i} = (\mathbf{y}_r^\top)_i, \quad i = 1 \ldots t$$

where $(V)_i$ is the i-th row of the matrix V. In the nonparametric random features case, RRLS can be applied considering

$$\mathbf{z}_{k+1} = \tilde{\mathbf{x}}_r, \quad \mathbf{u}_{k+1} = \mathbf{y}_r,$$

where $(\tilde{\mathbf{x}}_r, \mathbf{y}_r)$ is the supervised sample at step r.

III. SEMIPARAMETRIC INCREMENTAL DYNAMICS LEARNING

We propose a semiparametric incremental inverse dynamics estimator, designed to have better generalization properties with respect to fully parametric and nonparametric ones, both in terms of accuracy and convergence rates. The estimator, whose functioning is illustrated by the block diagram in Figure 2, is composed of two main parts. The first one is an incremental parametric estimator taking as input the rigid body dynamics regressors $\Phi(x)$ and computing two quantities at each step:

- An estimate $\hat{\mathbf{y}}$ of the output quantities of interest
- An estimate $\hat{\boldsymbol{\pi}}$ of the base inertial parameters of the links composing the rigid body structure

The employed learning algorithm is RRLS. Since it is supervised, during the model update step the measured output $\mathbf{y}$ is used by the learning algorithm as ground truth. The parametric estimation is performed first, and is independent of the nonparametric part. This property is desirable in order to give priority to the identification of the inertial parameters $\boldsymbol{\pi}$. Moreover, since the estimator is incremental, the estimated inertial parameters $\hat{\boldsymbol{\pi}}$ adapt to changes in the inertial properties of the links, which can occur if the end-effector is holding a heavy object. Still, this adaptation cannot address changes in nonlinear effects which do not respect the rigid body assumptions.

The second estimator is also RRLS-based, fully nonparametric and incremental. It leverages the approximation of the kernel function via random Fourier features, as outlined in Section II-C.1, to obtain a nonlinear model which can be updated incrementally with constant update complexity $O(D^2)$ (see Section II-E). This estimator receives as inputs the current vectorized $\mathbf{x}$ and $\hat{\mathbf{y}}$, normalized and mapped to the random features space approximating an infinite-dimensional feature space introduced by the Gaussian kernel. The supervised output is the residual $\triangle\mathbf{y} = \mathbf{y} - \hat{\mathbf{y}}$. The nonparametric estimator provides as output the estimate $\triangle\tilde{\mathbf{y}}$ of the residual, which is then added to $\hat{\mathbf{y}}$ to obtain the semiparametric estimate $\tilde{\mathbf{y}}$. Similarly to the parametric part, in the nonparametric one the estimator's internal nonlinear model can be updated during operation, which constitutes an advantage in the case in which the robot has to explore a previously unseen area of the state space, or when the mechanical conditions change (e.g. due to wear, tear or temperature shifts).

IV. EXPERIMENTAL RESULTS

A. *Software*

For implementing the proposed algorithm we used two existing open source libraries. For the RRLS learning part we used GURLS [30], a regression and classification library based on the Regularized Least Squares (RLS) algorithm, available for Matlab and C++. For the computations of the regressors $\Phi(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ we used iDynTree[2] (see [31]), a C++ dynamics library designed for free floating robots. Using SWIG [32], iDynTree supports calling its algorithms in several programming languages, such as Python, Lua and Matlab. For producing the presented results, we used the Matlab interfaces of iDynTree and GURLS.

B. *Robotic Platform*

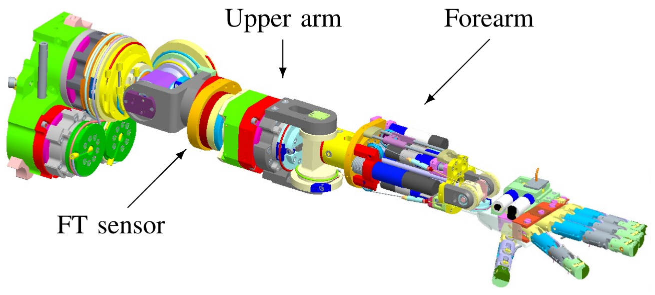


Fig. 3: CAD drawing of the iCub arm used in the experiments. The six-axis F/T sensor used for validation is visible in the middle of the upper arm link.

iCub is a full-body humanoid with 53 degrees of freedom [20]. For validating the presented approach, we learned the dynamics of the right arm of the iCub as measured from the proximal six-axis force/torque (F/T) sensor embedded in the arm (see Figure 3). The considered output $\mathbf{y}$ is the reading of the F/T sensor, and the inertial parameters $\boldsymbol{\pi}$ are the base parameters of the arm [33]. As $\mathbf{y}$ is not a input variable for the system, the output of the dynamic model is not directly usable for control, but it is still a proper benchmark for the dynamics learning problem, as also shown in [8]. Nevertheless, the joint torques could be computed seamlessly from the F/T sensor readings if needed for control purposes, by applying the method presented in [34].

C. *Validation*

We now present the results of the experimental validation of the proposed semiparametric model. The model includes a parametric part, based on physical modeling. This part is expected to provide acceptable prediction accuracy for the force components in the whole workspace of the robot, since it is based on prior knowledge about the structure of the robot itself, which does not abruptly change as the trajectory changes. On the other hand, the nonparametric part can provide higher prediction accuracy in specific areas of the input space for a given trajectory, since it also models nonrigid body dynamics effects by learning directly from data. To provide empirical foundations to the above insights, a validation experiment has been set up using the right arm of the iCub humanoid robot, considering as input the positions, velocities and accelerations of the 3 shoulder joints and of

[2]`https://github.com/robotology/idyntree`

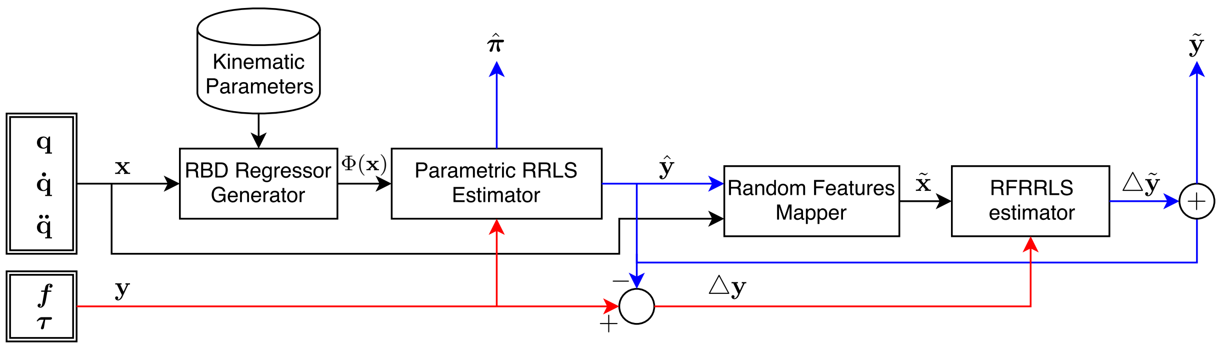


Fig. 2: Block diagram displaying the functioning of the proposed prioritized semiparametric inverse dynamics estimator. $\boldsymbol{f}$ and $\boldsymbol{\tau}$ indicate measured force and torque components, concatenated in the measured output vector $\mathbf{y}$. The parametric part is composed of the RBD regressor generator and of the parametric estimator based on RRLS. Its outputs are the estimated parameters $\hat{\boldsymbol{\pi}}$ and the predicted output $\hat{\mathbf{y}}$. The nonparametric part maps the input to the random features space with the Random Features Mapper block, and the RFRRLS estimator predicts the residual output $\triangle\tilde{\mathbf{y}}$, which is then added to the parametric prediction $\hat{\mathbf{y}}$ to obtain the semiparametric prediction $\tilde{\mathbf{y}}$.

the elbow joint, and as outputs the 3 force and 3 torque components measured by the six-axis F/T sensor in-built in the upper arm. We employ two datasets for this experiment, collected at $10Hz$ as the end-effector tracks (using the Cartesian controller presented in [35]) circumferences with $10cm$ radius on the transverse (XY) and sagittal (XZ) planes[3] at approximately $0.6m/s$. The total number of points for each dataset is 10000, corresponding to approximately 17 minutes of continuous operation. The steps of the validation experiment are the following:

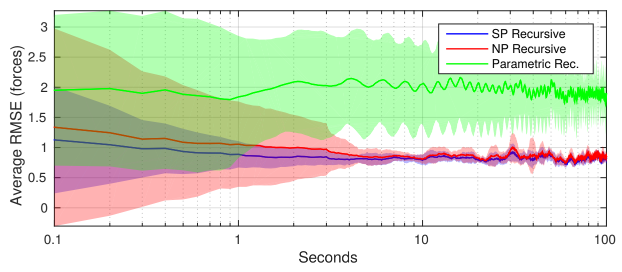
1) Initialize the recursive parametric, nonparametric and semiparametric models to zero. The inertial parameters are also initialized to zero
2) Train the models on the whole XY dataset (10000 points)
3) Split the XZ dataset in 10 sequential parts of 1000 samples each. Each part corresponds to 100 seconds of continuous operation
4) Test and update the models independently on the 10 splitted datasets, one sample at a time.

In Figure 4 we present the means and standard deviations of the average root mean squared error (RMSE) of the predicted force and torque components on the 10 different test sets for the three models, averaged over a 3-seconds sliding window. After few seconds, the nonparametric (NP) and semiparametric (SP) models provide more accurate predictions than the parametric (P) model with statistical significance. At steady state, their force prediction error is approximately $1N$, while the one of the P model is approximately two times larger. Similarly, the torque prediction error is $0.1Nm$ for SP and NP, which is considerably better than the $0.4Nm$ average RMSE of the P model. It shall also be noted that the mean average RMSE of the SP model is lower than the NP one, both for forces and torques. However, this slight difference is not very significant, since it is relatively small with respect to the standard deviation. Given these experimental results, we can conclude that in terms of predictive accuracy the proposed incremental semiparametric method outperforms the incremental parametric one and matches the fully nonparametric one. The SP method also shows a smaller standard deviation of the error with respect to the competing methods. Considering the previous results and observations, the proposed method has been shown to be able to combine the main advantages of parametric modeling (i.e. interpretability) with the ones of nonparametric modeling (i.e. capacity of modeling nonrigid body dynamics phenomena). The incremental nature of the algorithm, in both its P and NP parts, allows for adaptation to changing conditions of the robot itself and of the surrounding environment. Note that the introduction of a forgetting factor to assign smaller weights to older or less relevant samples might be necessary to avoid saturation phenomena in long time spans and properly track the changing system dynamics (see [36] and references therein). This interesting aspect will be addressed in future work.

V. CONCLUSIONS

We presented a novel incremental semiparametric modeling approach for inverse dynamics learning, joining together the advantages of parametric modeling derived from rigid body dynamics equations and of nonparametric Machine Learning methods. A distinctive trait of the proposed approach lies in its incremental nature, encompassing both the parametric and nonparametric parts and allowing for the prioritized update of both the identified base inertial parameters and the nonparametric weights. This feature is key to enabling robotic systems to adapt to mutable conditions of the environment and their own mechanical properties throughout extended periods. We validated our approach on the iCub humanoid robot, by analyzing the performances of a semiparametric inverse dynamics model of its right arm, comparing them with the ones obtained by state of the art fully nonparametric and parametric approaches.

[3]For more information on the iCub reference frames, see `http://eris.liralab.it/wiki/ICubForwardKinematics`



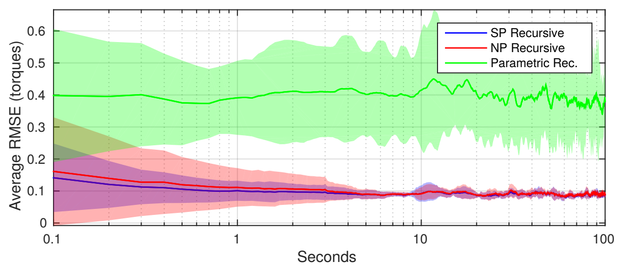


Fig. 4: Predicted forces (top) and torques (bottom) components average RMSE (y axis) in time (x axis), averaged over a 30-samples window for the recursive SP (blue), NP (red) and P (green) estimators. The transparent areas correspond to the standard deviation over 10 repetitions.

ACKNOWLEDGMENT

This paper was supported by the FP7 EU projects CoDyCo (No. 600716 ICT-2011.2.1), Koroibot (No. 611909 ICT-2013.2.1), WYSIWYD (No. 612139 ICT-2013.2.1), and Xperience (No. 270273 ICT-2009.2.1), by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216, and by the FIRB project RBFR12M3AC, funded by the Italian Ministry of Education, University and Research.

REFERENCES

[1] R. Featherstone and D. E. Orin, "Dynamics." in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 35–65.
[2] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. pp. 55–67, 1970.
[3] C. Saunders, A. Gammerman, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables." in *ICML*, J. W. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 515–521.
[4] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
[5] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, no. 190, pp. 131–154, 2003.
[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[7] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics." in *ICRA*. IEEE, 2010, pp. 2677–2682.
[8] A. Gijsberts and G. Metta, "Incremental learning of robot dynamics using random features." in *ICRA*. IEEE, 2011, pp. 951–956.
[9] T. Wu and J. Movellan, "Semi-parametric Gaussian process for robot system identification," in *IROS*, Oct 2012, pp. 725–731.
[10] J. Sun de la Cruz, D. Kulic, W. Owen, E. Calisgan, and E. Croft, "On-Line Dynamic Model Learning for Manipulator Control," in *IFAC Symposium on Robot Control*, vol. 10, no. 1, 2012, pp. 869–874.
[11] T. T. Um, M. S. Park, and J.-M. Park, "Independent joint learning: A novel task-to-task transfer learning scheme for robot models," in *ICRA*. IEEE, 2014, pp. 5679–5684.
[12] K. Yamane, "Practical kinematic and dynamic calibration methods for force-controlled humanoid robots." in *Humanoids*. IEEE, 2011, pp. 269–275.
[13] S. Traversaro, A. D. Prete, R. Muradore, L. Natale, and F. Nori, "Inertial parameter identification including friction and motor dynamics." in *Humanoids*. IEEE, 2013, pp. 68–73.
[14] Y. Ogawa, G. Venture, and C. Ott, "Dynamic parameters identification of a humanoid robot using joint torque sensors and/or contact forces." in *Humanoids*. IEEE, 2014, pp. 457–462.
[15] J. Hollerbach, W. Khalil, and M. Gautier, "Model identification," in *Springer Handbook of Robotics*. Springer, 2008, pp. 321–344.
[16] S. Vijayakumar and S. Schaal, "Locally Weighted Projection Regression: Incremental Real Time Learning in High Dimensional Space." in *ICML*, P. Langley, Ed. Morgan Kaufmann, 2000, pp. 1079–1086.
[17] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. MIT Press, 2002.
[18] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *NIPS*. Curran Associates, Inc., 2007, pp. 1177–1184.
[19] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model Learning with Local Gaussian Process Regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.
[20] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, "The iCub Humanoid Robot: An Open-systems Platform for Research in Cognitive Development," *Neural Netw.*, vol. 23, no. 8-9, pp. 1125–1134, Oct. 2010.
[21] M. Gautier and W. Khalil, "On the identification of the inertial parameters of robots," in *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*. IEEE, 1988, pp. 2264–2269.
[22] M. Gautier, "Dynamic identification of robots with power model," in *ICRA*, vol. 3. IEEE, 1997, pp. 1922–1927.
[23] K. Ayusawa, G. Venture, and Y. Nakamura, "Identifiability and identification of inertial parameters using the underactuated base-link dynamics for legged multibody systems," *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 446–468, 2014.
[24] J. Baelemans, P. van Zutven, and H. Nijmeijer, "Model parameter estimation of humanoid robots using static contact force measurements," in *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, Oct 2013, pp. 1–6.
[25] W. Khalil and E. Dombre, *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.
[26] A. Rahimi and B. Recht, "Uniform approximation of functions with random bases," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, Sept 2008, pp. 555–561.
[27] W. Rudin, *Fourier Analysis on Groups*, ser. A Wiley-interscience publication. Wiley, 1990.
[28] Å. Björck, *Numerical Methods for Least Squares Problems*. Siam Philadelphia, 1996.
[29] A. H. Sayed, *Adaptive Filters*. Wiley-IEEE Press, 2008.
[30] A. Tacchetti, P. K. Mallapragada, M. Santoro, and L. Rosasco, "GURLS: a least squares library for supervised learning," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3201–3205, 2013.
[31] F. Nori, S. Traversaro, J. Eljaik, F. Romano, A. Del Prete, and D. Pucci, "icub whole-body control through force regulation on rigid noncoplanar contacts," *Frontiers in Robotics and AI*, vol. 2, no. 6, 2015.
[32] D. M. Beazley *et al.*, "SWIG: An easy to use tool for integrating scripting languages with C and C++," in *Proceedings of the 4th USENIX Tcl/Tk workshop*, 1996, pp. 129–139.
[33] S. Traversaro, A. Del Prete, S. Ivaldi, and F. Nori, "Inertial parameters identification and joint torques estimation with proximal force/torque sensing," in *ICRA*, 2015.
[34] S. Ivaldi, M. Fumagalli, M. Randazzo, F. Nori, G. Metta, and G. Sandini, "Computing robot internal/external wrenches by means of inertial, tactile and F/T sensors: Theory and implementation on the iCub," in *Humanoids*, Oct 2011, pp. 521–528.
[35] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *IROS*, Oct 2010, pp. 1668–1674.
[36] C. Paleologu, J. Benesty, and S. Ciochină, "A robust variable forgetting factor recursive least-squares algorithm for system identification," *Signal Processing Letters, IEEE*, vol. 15, pp. 597–600, 2008.