Feedback Linearization Robot Control based on Gaussian Process Inverse Dynamics Model

Alberto Dalla Libera
Dept. of Information Eng.
Università di Padova
Padova, Italy
dallaliber@dei.unipd.it

Fabio Amadio
Dept. of Information Eng.
Università di Padova
Padova, Italy
amadiofa@dei.unipd.it

Ruggero Carli
Dept. of Information Eng.
Università di Padova
Padova, Italy
carlirug@dei.unipd.it

Diego Romeres
Mitsubishi Electric
Research Laboratories
Cambridge, Massachusetts
romeres@merl.com

***Abstract*—In this paper, we analyze the implementation of feedback linearization control scheme based on full data-driven inverse dynamics models. We made no use of physical models in the definition of the inverse dynamics, that has been learned entirely from previously recorded data via Gaussian Process Regression (GPR). The resulting controller has been tested in a simulated 7 dof manipulator to solve a trajectory tracking problem. Different kernel functions have been tested, in particular we analyzed the performance obtained by Squared Exponential (SE) kernel and the recently introduced Geometrically Inspired Polynomial (GIP) kernel. Results show that GIP obtains better tracking precision and it is more robust w.r.t. the presence of an initial tracking errors. On the contrary, poor generalization properties of SE kernel deeply undermine control performance when the robot is located far from the poses seen during training.**

***Index Terms*—feedback linearization control, inverse dynamics estimation, gaussian process**

I. Introduction

Feedback linearization offers the possibility to design robot controllers that could be considerably more precise and energy-efficient than standard linear feedback control [1]. Its major drawback is that it needs a precise inverse dynamics model to work properly. However, for many systems, such accurate models cannot be obtained using standard rigid body formulation due to unmodeled non-linearities or system misspecifications. Gaussian Process Regression (GPR) [2] provides a powerful data-driven technique for solving the inverse dynamics identification problem, overcoming the aforementioned limitations. In this work, we analyze the implementation of a feedback linearization control scheme based on a GP inverse dynamics model [3]. In particular, we show the importance that the choice of the kernel function plays in the success of the proposed controller. We compare the results obtained by the *Squared Exponential* (SE) kernel [2], a standard choice in many GPR applications, and the recently introduced *Geometrically Inspired Polynomial* (GIP) kernel [4]. The implemented controller have been tested in a simulated 7 degrees of freedom (dof) manipulator to solve a trajectory tracking problem. The remainder of the paper is organized as follows. In Sec. II, we provide background notions about robot dynamics and control, as well as GPR. In Sec. III we describe the GP-based feedback linearization controller. Experiments are reported in Sec. IV, and conclusions are drawn in Sec. V.

II. Background

In this section, we provide background notions about robot dynamics, as well as introducing the trajectory tracking problem and describing feedback linearization control. Then, we describe GPR for inverse dynamics identification, detailing the black-box priors adopted in this work.

A. Robot dynamics and control

Consider a mechanical systems with n dof, 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, respectively, the joint velocities and the accelerations. The generalized torques are denoted with $\boldsymbol{\tau}_t \in \mathbb{R}^n$. We will denote explicitly the dependencies on t only when strictly necessary. Under rigid body assumptions, the dynamics equations of mechanical systems are described by the following matrix equation

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

where $B(\boldsymbol{q})$ is the inertia matrix, while $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\boldsymbol{g}(\boldsymbol{q})$, and $\boldsymbol{F}(\dot{\boldsymbol{q}})$ accounts, respectively, for the contributions of fictitious forces, gravity, and friction. For compactness, we introduce also $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}})$. In the following we will denote with $\hat{B}(\boldsymbol{q})$ and $\hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the estimates of $B(\boldsymbol{q})$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. The trajectory tracking problem consists in designing a controller able to follow a reference $\boldsymbol{r}_t, \dot{\boldsymbol{r}}_t, \ddot{\boldsymbol{r}}_t$. In feedback linearization control the control input is

$$\boldsymbol{a} = \ddot{\boldsymbol{r}} + K_p \boldsymbol{e} + K_d \dot{\boldsymbol{e}}, \tag{2a}$$

$$\boldsymbol{\tau} = \hat{B}(\boldsymbol{q})\boldsymbol{a} + \hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{2b}$$

Assuming that the model is known exactly, i.e., $\hat{B}(\boldsymbol{q}) = B(\boldsymbol{q})$ and $\hat{\boldsymbol{n}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, combining (1) and (2) and recalling that $B(\boldsymbol{q})$ is invertible it can be proved that the dynamics of the tracking error is described by the following second order linear differential equation,

$$\ddot{\boldsymbol{e}} + K_d \dot{\boldsymbol{e}} + K_p \boldsymbol{e} = 0, \tag{3}$$

which is asymptotically stable if $K_p > 0$ and $K_d > 0$.

B. GPR for inverse dynamics identification

In GPR each joint torque is modeled with a distinct and independent GP. Consider an input/output dataset $\mathcal{D} = \{\boldsymbol{y}^{(i)}, X\}$, where $\boldsymbol{y}^{(i)} \in \mathbb{R}^N$ is a vector collecting N measurements of $\tau^{(i)}$, the *i*-th joint torque, while $X = \{\boldsymbol{x}_{t_1} \dots \boldsymbol{x}_{t_N}\}$; $\boldsymbol{x}_t$ is the vector collecting the position, velocity and acceleration of the joints at time t, hereafter denoted GP input. The probabilistic model of $\mathcal{D}$ is

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

where $\boldsymbol{w}^{(i)}$ 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, $f^{(i)}(\cdot) \sim N(0, \mathbb{K}^{(i)}(X,X))$. The covariance matrix $\mathbb{K}^{(i)}(X,X)$ is defined through a kernel function $k^{(i)}(\cdot,\cdot)$. Specifically, the covariance between $f^{(i)}(\boldsymbol{x}_{t_j})$ and $f^{(i)}(\boldsymbol{x}_{t_l})$, i.e., the element of $\mathbb{K}^{(i)}(X,X)$ at row *j* and column *l*, is equal to $k^{(i)}(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_l})$. Given a general input location $\boldsymbol{x}_*$, the maximum a posteriori estimator is

$$\hat{f}^{(i)}(\boldsymbol{x}_*)) = \mathbb{K}^{(i)}(\boldsymbol{x}_*, X)\,\boldsymbol{\alpha}^{(i)}, \text{ where} \tag{4a}$$

$$\boldsymbol{\alpha}^{(i)} = (\mathbb{K}^{(i)}(X,X) + \sigma_i^2 I)^{-1}\boldsymbol{y}^{(i)}, \tag{4b}$$

$$\mathbb{K}^{(i)}(\boldsymbol{x}_*, X) = \left[k^{(i)}(\boldsymbol{x}_*, \boldsymbol{x}_{t_1}) \dots k^{(i)}(\boldsymbol{x}_*, \boldsymbol{x}_{t_N})\right], \tag{4c}$$

see [2] for details. In this work, we will consider the two following kernels:

- Squared Exponential kernel (SE):

The SE kernel [2] defines the covariance between samples based on the distances between GP inputs, and it is defined by the following expression

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

where λ is a scaling factor and Σ is a positive definite diagonal matrix which defines the norm used.

- Geometrically inspired Polynomial kernel (GIP):

This kernel is based on the property that the dynamics equations in (1) are a polynomial function in $\ddot{\boldsymbol{q}}$, $\dot{\boldsymbol{q}}$ and $\tilde{\boldsymbol{q}}$, where $\tilde{\boldsymbol{q}}$ is a transformation of $\boldsymbol{q}$. Specifically, $\tilde{\boldsymbol{q}}$ is the vector composed by the concatenation of the prismatic coordinates and the sines and cosines of the revolute coordinates. As proved in [4], the elements of $\ddot{\boldsymbol{q}}$ have maximum relative degree one, while the ones of $\dot{\boldsymbol{q}}$ and $\tilde{\boldsymbol{q}}$ have maximum relative degree two. Then, the GIP kernel is defined through the sum and the product of different polynomial kernel [5], denoted as $k_P^{(p)}(\cdot,\cdot)$, where p is the degree of the polynomial kernel. In particular, we have

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

where, in its turns, k_Q is given by the product of polynomial kernel with degree two, see [4] 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 w.r.t. the SE kernel.

III. GP-based Feedback Linearization Control

In this section, we describe the implemented controller, hereafter denoted as GP Feedback Linearization (GP-FL). GP-FL estimates directly the control input in (2) using the GP models. The estimate of (2b) at time t is obtained evaluating the n GP models with GP-input given by the concatenation of $\boldsymbol{q}_t$, $\dot{\boldsymbol{q}}_t$ and $\boldsymbol{a}_t = \ddot{\boldsymbol{r}}_t + K_p\boldsymbol{e}_t + K_d\dot{\boldsymbol{e}}_t$. Then, we have

$$\boldsymbol{\tau}_t = \begin{bmatrix} \hat{f}^{(1)}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{a}_t) \\ \vdots \\ \hat{f}^{(n)}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{a}_t) \end{bmatrix}, \tag{7}$$

where, with abuse of notation, we pointed out explicitly the different components of the GP-input, instead of using their concatenation.

IV. Experiments

Experiments have been carried out in PyBullet [6], simulating a 7 dof manipulator, with a control frequency of 1000 Hz. First, we collected data for inverse dynamics identification by employing a hand-tuned PD controller to track a random reference trajectory. For each joint, the reference is 50 seconds of Gaussian noise filtered with a low-pass filter (with cut frequency 1 Hz). The dataset used to train the GP models is composed of 5000 samples, obtained downsampling the data collected. Two different GP inverse dynamics model have been trained on the recorded data-set: the first employs a SE kernel, while the second a GIP kernel. For each dof $j = 1, \dots, 7$, the reference joint position is given by $r_t^{(j)} = 0.165\ t\ sin(2\pi F_j\ t)$ where the frequencies F_j are randomly sampled from $\mathcal{N}(0.5, 1)$. The control horizon has been set to 5 [s]. In Fig. 1, and 2 are reported, respectively, the evolution of the joint angles and control torques obtained by the two controllers, starting with initial tracking error null. One can note that GP-FL controller with SE kernel works properly when the amplitudes of the reference oscillations are low, but it starts failing suddenly towards the end of the control horizon, when null torques are given to all joints. This is due to the fact that when the reference trajectory crosses regions that are far from the training samples the GP estimator based on SE kernel fails to provide meaningful predictions. Indeed, the control torques goes to zero, which is the prior mean [2]. Instead, thanks to the better generalization of GIP kernel, the controller based on GIP kernel is more robust and it is always able to track the desired reference, also in unexplored areas of the state-space.

We have tested the two controllers also in presence of initial tracking errors. Results highlight the same issues. The estimator based on SE kernel is not effective. In fact, the initial error make the magnitude of the $\boldsymbol{a}_t$ term so high that its distance from accelerations observed during training is big, leading to null torques. Instead, GIP kernel has no problem in handling the presence of a starting error. In Fig. 3, we plotted the tracking errors obtained with the controller based on GIP kernel and the one based on the true model, when starting with an initial error of 5.73 [deg] for all the joints. In both cases the evolution of the errors follows (3), confirming the effectiveness of the

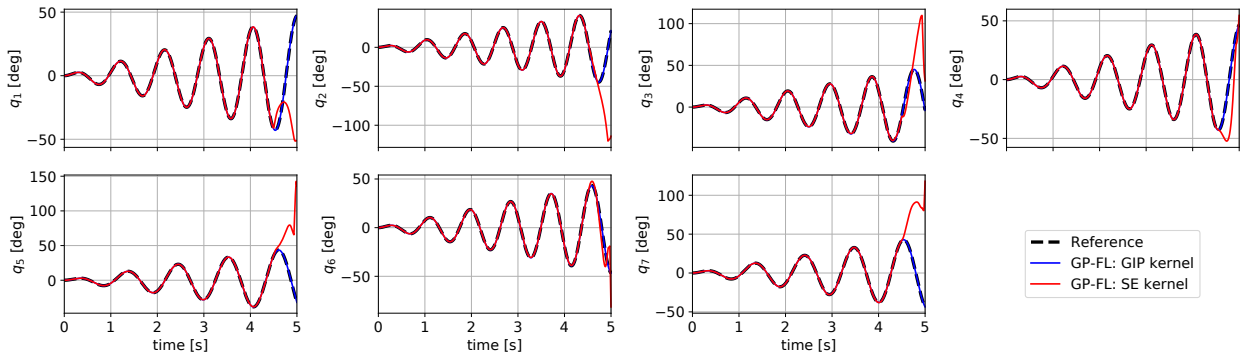

Fig. 1. Reference and actual joint trajectories obtained by the GP-FL controllers with SE kernel and GIP kernel.

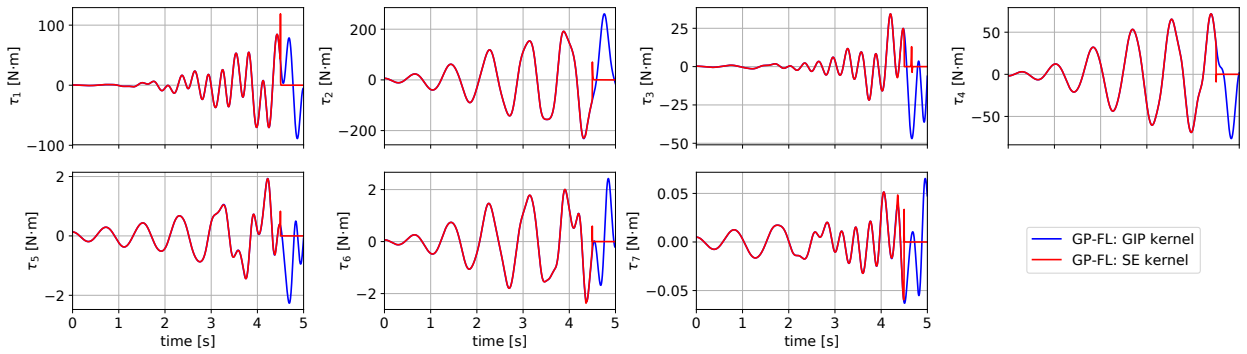

Fig. 2. Applied torques obtained by the GP-FL controllers with SE kernel and GIP kernel.

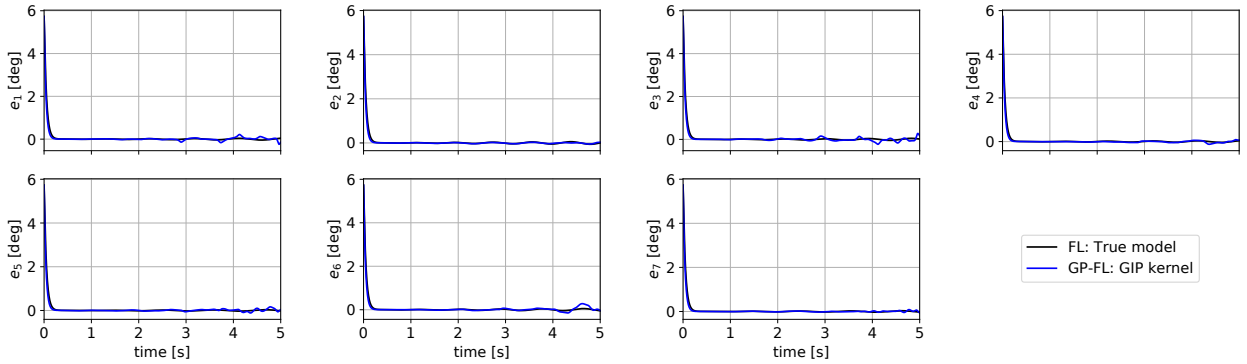

Fig. 3. Error evolution obtained by the GP-FL controllers with GIP kernel and by the FL controller based on the true model.

proposed solution. Limited oscillations around zero can be observed with the controller based on GIP when the reference trajectory is far from the training data.

V. Conclusions

We analyzed the application of GP-based inverse dynamics models for feedback linearization control of robotic systems, focusing on the effects of the kernel adopted. In particular, we compared the performance obtained by using SE or GIP kernels in a simulated trajectory tracking problem. GP-FL control with SE kernel showed critical problems related to the poor generalization properties. On the contrary, GIP kernel was effective in learning inverse dynamics models that can be used for the design of GP-based feedback linearization control.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[2] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[3] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *2008 American Control Conference*. IEEE, 2008, pp. 212–217.

[4] A. Dalla Libera and R. Carli, "A data-efficient geometrically inspired polynomial kernel for robot inverse dynamic," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 24–31, 2020.

[5] A. D. Libera, R. Carli, and G. Pillonetto, "A novel multiplicative polynomial kernel for volterra series identification," *arXiv preprint arXiv:1905.07960*, 2019.

[6] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.