UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Learning Algorithms for Robotics Systems

Ph.D. candidate
Alberto Dalla Libera

Advisor
prof. Ruggero Carli

Co-Advisor
prof. Gianluigi Pillonetto

Director & Coordinator
prof. Andrea Neviani

Ph.D. School in
Information Engineering

Department of
Information Engineering
University of Padova
2019

University of Padova
Department of Information Engineering

UNIVERSITÀ DEGLI STUDI DI PADOVA

Ph.D course in: Information Engineering
Curriculum: Information Science and Technology
Cycle: XXXII

Learning Algorithms for Robotics Systems

Director: Prof. Andrea Neviani
Advisor: Prof. Ruggero Carli
Co-Advisor: Prof. Gianluigi Pillonetto

Ph.D. candidate: Alberto Dalla Libera

Year 2019

Abstract

Robotics systems are now increasingly widespread in our day-life. For instance, robots have been successfully used in several fields, like, agriculture, construction, defense, aerospace, and hospitality. However, there are still several issues to be addressed for allowing the large scale deployment of robots. Issues related to security, and manufacturing and operating costs are particularly relevant. Indeed, differently from industrial applications, service robots should be cheap and capable of operating in unknown, or partially-unknown environments, possibly with minimal human intervention.

To deal with these challenges, in the last years the research community focused on deriving learning algorithms capable of providing flexibility and adaptability to the robots. In this context, the application of Machine Learning and Reinforcement Learning techniques turns out to be especially useful.

In this manuscript, we propose different learning algorithms for robotics systems. In Chapter 2, we propose a solution for learning the geometrical model of a robot directly from data, combining proprioceptive measures with data collected with a 2D camera. Besides testing the accuracy of the kinematic models derived with real experiments, we validate the possibility of deriving a kinematic controller based on the model identified.

Instead, in Chapter 3, we address the robot inverse dynamics problem. Our strategy relies on the fact that the robot inverse dynamics is a polynomial function in a particular input space. Besides characterizing the input space, we propose a data-driven solution based on Gaussian Process Regression (GPR). Given the type of each joint, we define a kernel named *Geometrically Inspired Polynomial* (GIP) kernel, which is given by the product of several polynomial kernels. To cope with the dimensionality of the resulting polynomial, we use a variation of the standard polynomial kernel, named *Multiplicative Polynomial* kernel, further discussed in Chapter 6. Tests performed on simulated and real environments show that, compared to other data-driven solutions, the GIP kernel-based estimator is more accurate and data-efficient.

In Chapter 4, we propose a proprioceptive collision detection algorithm based on GPR. Compared to other proprioceptive approaches, we closely inspect the robot behaviors in

quasi-static configurations, namely, configurations in which joint velocities are null or close to zero. Such configurations are particularly relevant in the Collaborative Robotics context, where humans and robots work side-by-side sharing the same environment. Experimental results performed with a UR10 robot confirm the relevance of the problem and the effectiveness of the proposed solution.

Finally, in Chapter 5, we present MC-PILCO, a model-based policy search algorithm inspired by the PILCO algorithm. As the original PILCO algorithm, MC-PILCO models the system evolution relying on GPR, and improves the control policy minimizing the expected value of a cost function. However, instead of approximating the expected cost by moment matching, MC-PILCO approximates the expected cost with a Monte Carlo particle-based approach; no assumption about the type of GPR model is necessary. Thus, MC-PILCO allows more freedom in designing the GPR models, possibly leading to better models of the system dynamics. Results obtained in a simulated environment show consistent improvements with respect to the original algorithm, both in terms of speed and success rate.

Sommario

I robot sono sempre più diffusi nella nostra vita di ogni giorno, e sono stati utilizzati con successo in diversi campi, come l'agricoltura, l'edilizia, la difesa, l'aerospaziale e l'hospitality. Tuttavia, ci sono ancora diversi problemi che devono essere risolti per consentire la distribuzione su larga scala dei robot. Particolarmente rilevanti sono le questioni relative a costi di produzione e costi operativi, nonché le questioni relative alla sicurezza. Infatti, diversamente dalle applicazioni industriali, i robot di servizio dovrebbero essere economici e in grado di funzionare in ambienti sconosciuti o parzialmente sconosciuti, possibilmente il con minimo intervento umano.

Per far fronte a queste sfide, negli ultimi anni la comunità di ricerca si è concentrata sulla derivazione di algoritmi di apprendimento in grado di fornire flessibilità e adattabilità ai robot. In questo contesto, l'applicazione delle tecniche di Machine Learning e Reinforcement Learning risulta particolarmente promettente.

In questo manoscritto, proponiamo diversi algoritmi di apprendimento per sistemi robotici. Nel Capitolo 2 proponiamo una soluzione data-driven per l'apprendimento del modello geometrico del robot, combinando misure propriocettive con dati provenienti da una videocamera 2D. Oltre a testare l'accuratezza dei modelli cinematici derivati con esperimenti reali, abbiamo verificato la possibilità di derivare un controller cinematico basato sul modello identificato.

Nel Capitolo 3 consideriamo il problema della dinamica inversa. La strategia proposta si basa sul fatto che la dinamica inversa è una funzione polinomiale in un partiolare spazio di input. Oltre che caratterizzare questo spazio degli input, proponiamo una soluzione data-driven basata su Gaussian Process Regression (GPR). Dato il tipo di ciascun giunto, abbiamo definito un kernel chiamato *Geometrically Inspired Polynomial* (GIP) kernel, dato dal prodotto di diversi kernel polinomiali. Per far fronte alla dimensionalità del polinomio risultante, abbiamo utilizzato una variante del kernel polinomiale standard, chiamato *Multiplicative Polynomial* kernel, e analizzato in dettaglio nel Capitolo 6. Test effettuati su ambienti simulati e reali mostrano che, rispetto ad altre soluzioni data-driven, lo stimatore basato sul kernel GIP è più preciso e più data-efficient.

Nel Capitolo 4, proponiamo un algoritmo propriocettivo di rilevamento delle collisioni basato su GPR. Rispetto ad altre soluzioni propriocettive, abbiamo posto l'attenzione sui comportamenti dei robot in configurazioni quasi statiche, vale a dire configurazioni in cui le velocità dei giunti sono nulle o quasi nulle. Tali configurazioni sono particolarmente rilevanti nel contesto della robotica collaborativa, in cui umani e robot lavorano fianco a fianco condividendo lo stesss ambiente. I risultati sperimentali ottenuti con un robot UR10 confermano la rilevanza del problema e l'efficacia della soluzione proposta.

Infine, nel Capitolo 5, presentiamo MC-PILCO, il nostro algoritmo di model-based policy search, ispirato all'algoritmo PILCO. Come l'originale algoritmo PILCO, MC-PILCO modella l'evoluzione del sistema basandosi su GPR, e migliora la policy di controllo minimizzando il valore atteso di una certa funzione di costo. Tuttavia, PILCO approssima il valore atteso del costo utilizzando moment matching. Al contratio, in MC-PILCO l'aspettazione del costo è calcolata con un approccio Monte Carlo particle-based, senza considerare alcuna ipotesi sul tipo di modello GPR utilizzato. In questo modo, MC-PILCO consente una maggiore libertà nella progettazione dei modelli GPR. I risultati ottenuti in simulazione mostrano miglioramenti consistenti rispetto all'algoritmo originale, sia in termini di velocità che di percentuale di successo.

Acknowledgments

First, I thank my advisor Prof. Ruggero Carli for supervising me and for the useful advice provided during these three years.

I would like to mention also Prof. Gianluigi Pillonetto, who has always been available for clarification and suggestions.

I would like to thank also all my Ph.D. colleagues in Padova, for the research activities performed together and also for the happy moments outside work.

Besides, I would like to express a special thanks to Diego Romeres, who believed in me, and, together with Daniel Nikovski, gave me the opportunity of spending four months in Boston, working as an intern at MERL. I thank also all the other MERL colleagues.

Finally, I would like to thank my family, my parents Graziella and Carlo, my brother Enrico, and also my girlfriend Annalisa, who always supported me during this period.

Contents

1 Introduction

1.1 New challenges in Robotics

Robotic systems are becoming always more autonomous and reconfigurable, as well as increasingly used in applications "out of the cage". Semi-autonomous and fully autonomous robots have been successfully adopted to accomplish physically intensive and dangerous tasks, in fields like aerospace, defense, agriculture, and construction Hajjaj and Sahari (2016); Augugliaro, Lupashin, Hamer, Male, Hehn, Mueller, Willmann, Gramazio, Kohler, and D'Andrea (2014); Longo and Muscato (2006); Hanjong Joo, ChiSu Son, Kyunghun Kim, Kyunghwan Kim, and Jaejun Kim (2007). Other relevant examples are collaborative applications, in which robots are intended to work side-by-side with humans, possibly entailing also physical interactions Haddadin, De Luca, and Albu-Schäffer (2017). The development of collaborative robots might have a great impact on several domains, like hospitality, health-care and industry. For instance, in the medical field robots might assist humans in rehabilitation from injuries Gelderblom, Wilt, Cremers, and Rensma (2009), or in industrial applications, the robot working with a human being might relieve him from the most physically demanding activities, Masinga, Campbell, and Trimble (2015).

The large-scale use of robots entails also new challenges. A first aspect to be considered is the reduction of manufacturing and setup costs. Indeed, in order to guarantee precision and repeatability, traditional industrial robots are made with high-quality components, entailing manufacturing costs that prevent the large-scale deployment of robots. Moreover, considerable additional costs are due to the employment of high-qualified personnel in setup activities, like calibration and programming. On the other hand, the use of low-quality components augments measurement noise, wear, and uncertainty about the parameters of the robot, possibly leading to consistent discrepancies between the real and the expected behaviors of the robot. In this context, standard identification techniques based on first-principles of physic and prior knowledge about the robot parameters might be not enough robust to deal with unmodeled behaviors and model bias. Consequently, several black-box solutions for deriving estimators of the robot kinematics and dynamics models have been proposed in recent years, relying on the use of tools like Gaussian Process Regression and Neural Networks Rasmussen and Williams (2006); Schölkopf and Smola (2001); Hinton (1988); Goodfellow, Bengio, and Courville (2016). Besides limiting the human-intervention, such techniques allow modeling complex behaviors and reducing effects due to model bias. On the other hand, learning a model purely from data is a challenging task as concerns generalization, namely, the estimation accuracy might decrease significantly when testing the estimator in input locations that are far from the distribution of the training samples. The difficulties of guaranteeing good out of sample performance pose serious questions about the applicability of such techniques in real applications, and motivate the interest in hybrid solutions, merging prior knowledge about the model with information coming from data.

Bringing robots "out of the cage" entails also several issues related to safety, in particular when robots work side-by-side with humans. In the last decade, several efforts have been made to derive planning algorithms that enable robots and humans to share the same workspace. Traditional planning strategies have been modified to minimize the risk and the intensity of collisions. The solutions proposed are based on the use of proximity sensors and 2D cameras, as well as on a probabilistic characterization of human behaviors Ebert and Henrich (2002); Landi, Ferraguti, Costi, Bonfè, and Secchi (2019). However, due to the unpredictability of human behaviors, it is impossible to reduce the collision risk to zero. Moreover, there are applications in which humans and robots need to interact physically. Consequently, it is fundamental that robots are provided with tools for detecting collisions, as well as for estimating their intensity. The strategies proposed can be broadly divided into two sets. The first set accounts for solutions that use ad-hoc sensors to detect collisions, like six-axis force sensors and artificial skins Cirillo, Ficuciello, Natale, Pirozzi, and Villani (2016). The second set instead, groups all the algorithms that detect collisions combining the knowledge of the robot dynamics with proprioceptive measures, such as position, velocities, accelerations, and torques of the joints Haddadin et al. (2017). The main limitations of using ad-hoc sensors are related to the costs and sensitivity of the sensors. For instance, consider a standard manipulator with six degrees of freedom. The cost of providing each joint of the manipulator with a six-axis force sensors might be higher than the cost of the whole manipulator. From this point of view, the second class of solutions seems to be more convenient and has attracted the attention of the research community. The most performing strategies are based on first-principles models of the robot dynamics described by the Lagrangian equations, and on the possibility of measuring joints torques. However, it is worth mentioning that, when considering low costs hardware, typically, robots are not equipped with joints torque sensors. Moreover, as mentioned before, traditional modeling techniques might not be sufficiently accurate, compromising significantly the performance of proprioceptive techniques.

Finally, in recent years, the application of Reinforcement Learning techniques to robotic systems has attracted the attention of the Robotics community. This emphasis is motivated by the impact that Reinforcement Learning could have on the deployment of robotic systems. Indeed, several studies show that a considerable amount of the operating costs are due to setup and programming activities. As well as requiring high-qualified personnel, such activities could be needed frequently, due to several aspects, like, wear, setup changes, or variation of the task to be accomplished. Differently than in industrial applications, the costs due to such activities are not sustainable in the context of service robotics, where setup and task variations are more frequent. Reinforcement Learning techniques could decrease significantly such costs, providing to the robots the capabilities of adapting to tasks and setup variations. Indeed, Reinforcement Learning algorithms have been able to reach and exceed human-level performance in several benchmark problems, such as playing chess, go and shogi Silver, Hubert, Schrittwieser, Antonoglou, Lai, Guez, Lanctot, Sifre, Kumaran, Graepel, Lillicrap, Simonyan, and Hassabis (2018). Despite these remarkable results, the application of Reinforcement Learning to robotic systems is still a challenge, because of the large amount of experience required and the safety risks associated with random exploration. Partial improvements have been obtained relying on Model-Based Reinforcement Learning techniques, see for instance Deisenroth and Rasmussen (2011); Todorov and Li (2005); Levine and Abbeel (2014). Experimental results show that providing an explicit model of the physical system allows considerable decreases of the experience time required to converge to good solutions, while also reducing the risk of damage to the hardware during exploration and policy improvement. However, several improvements are still needed to derive safe and robust algorithms.

1.2 Manuscript overview

This manuscript collects several learning algorithms useful to deal with part of the aforementioned problems. In chapters 2 and 3 we present two black-box solutions for the identification of the robot kinematics and dynamics. In particular, in Chapter 2 we propose a framework to learn the robots' geometrical model from time-series of visual observations, collected with a 2D camera. The framework is particularly useful in contexts where the prior knowledge about the robot geometry is partial, or null. An example is Modular Robotics Hornby, Lipson, and Pollack (2003); Gilpin and Rus (2010); Yim, Duff, and Roufas (2000); Yim, Shen, Salemi, Rus, Moll, Lipson, Klavins, and Chirikjian (2007); Brodbeck and Iida (2012); Guan, Jiang, Zhangy, Zhang, and Zhou (2009); Sprowitz, Pouya, Bonardi, Den Kieboom, Mockel, Billard, Dillenbourg, and Ijspeert (2010). Modular robots are composed of different elementary building blocks, and, interchanging, adding or subtracting these elementary modules, modular robots can variate their geometry. Firstly, the algorithm identifies the robot kinematic structure, namely, a high-level description of the robot that defines the connections between links and joints, as well as the type of each joint. Secondly, a model of the forward kinematics is identified relying on Gaussian Process Regression techniques. The effectiveness of the proposed solution is evaluated with tests in simulated and real environments. Part of the contributions reporter in Chapter 2 are based on Dalla Libera, Terzi, Rossi, Susto, and Carli (2019a)

The inverse dynamics identification problem is considered in Chapter 3. After discussing the different solutions proposed in the literature, we describe our black-box strategy based on Gaussian Process Regression. The main idea supporting our approach relies on the fact that the inverse dynamics is a polynomial function in an augmented space of the inputs traditionally considered in inverse dynamics identification, namely, positions, velocities, and accelerations of the joints. Besides characterizing the polynomial function in terms of the maximum relative degree of each variable, we propose a kernel that encodes all the monomials of the polynomial space containing the inverse dynamics. We named the proposed kernel *Geometrically Inspired Polynomial kernel* because it is fully defined by the robot kinematic structure. Tests performed in simulated and real environments show that, compared to other black-box solutions, the proposed approach is more data-efficient, and performs better in out of sample estimation. Finally, in Chapter 3 we briefly introduce the *Multiplicative Polynomial kernel* a refinement of the standard polynomial kernel, used to derive the *Geometrically Inspired Polynomial kernel.* The properties of the *Multiplicative Polynomial kernel* are investigated in depth in Chapter 6. The contributions of Chapter 3 are based on Dalla Libera and Carli (2020)

Chapter 4 is about proprioceptive collision detection. In this chapter, we propose a collision detection strategy based on Gaussian Process Regression. Considerable attention is devoted to problems raising in *quasi-static* configurations, namely when joints velocities are small or null. Such configurations are particularly relevant in collaborative applications, give that, when robots work side-by-side with humans, joints velocities are limited due to safety reasons. In *quasi-static* configurations behaviors due to friction become particularly relevant and hard to be modeled. Inaccuracies of the inverse dynamics model could compromise also the performance of the collision detection algorithm. Analyzing examples of these singular behaviors in a real setup, we show how traditional identification strategies can not model similar behaviors. Driven by this consideration, we propose a collision detection algorithm based on a modification of Gaussian Process Regression models typically used for identifying the inverse dynamics. Experiments performed with a real robot show the effectiveness of our approach. Part of the contributions presented in Chapter 4 are based on Dalla Libera, Tosello, Pillonetto, Ghidoni, and Carli (2019b)

In Chapter 5 we introduce MC-PILCO (Monte Carlo Probabilistic Inference for Learning Control), a model-based policy gradient algorithm that relies on Gaussian Process Regression and Monte Carlo sampling. The algorithm is inspired by the PILCO (Probabilistic Inference for Learning Control) algorithm and tries to cope with two possible limitations of the original algorithm. In PILCO, the system evolution is modeled with a Gaussian process. The algorithm updates the policy parameters, i.e., the control function, minimizing the expected value of a cost function, computed w.r.t. the probability distribution induced by the Bayesian model learned from data collected during interactions. The cost function is a map of the system states, incorporating information about the task to be solved, e.g., reach a particular state. However, the exact computation of this expectation is not feasible, due to the impossibility of deriving a closed form of the long-term state-distribution. In PILCO, an approximation of the objective function is derived in closed form, relying on moment matching, and assuming that long-term state predictions are Gaussian-distributed. Firstly, to compute the moments in closed form, PILCO requires the use of Radial Basis Functions kernels, preventing the adoption of more structured kernels. Secondly, approximating the expected state distributions with Gaussian variables allows modeling only unimodal distribution, possibly leading to considerable discrepancies between the predicted states and the predicted behaviors.

In our work, we approximate the expectation of the cost function relying on Monte Carlo simulation. In this manner, as well as allowing the freedom of choosing any kernel function to model the system, we do not introduce constraints on the distribution of the long-term predictions. Results obtained in a simulated carpole system prove that the combination of these two aspects leads to consistent performance improvement.

Finally, as mentioned before, in Chapter 6 we further inspect the properties of the *Multiplicative Polynomial Kernel*. The proposed kernel is applied to the identification of Volterra Series, a class of models used in System Identification to represent nonlinear systems. Volterra series are strictly connected with the Taylor expansion of the input-output impulse response, and, when working in discrete time, they are polynomial functions in the lagged inputs. With such models, the problem is the course of dimensionality. The number of coefficients to be identified is proportional to the number of possible monomials, namely, it grows exponentially with the system memory and the polynomial degree. This fact makes the Volterra series identification a particularly challenging task. After providing a formal definition of the proposed kernel, in Chapter 6 we compare its regularization properties with the ones of the standard polynomial kernel. The *Multiplicative Polynomial Kernel* is equipped with a richer set of hyperparameters that provides more flexibility in penalizing the monomials that really influence the system output. This fact leads to better performance in terms of data-efficiency and out of sample accuracy, as confirmed by experimental results. Part of the contributions presented in Chapter 6 are based on Dalla Libera, Carli, and Pillonetto (2019c)

1.3 Background on Gaussian Process Regression

We conclude this chapter providing background notions about Gaussian Process Regression (GPR), given that we will use this tool in all the next chapters. GPR is a Bayesian estimation tool successfully used in different fields, thanks to its capability of representing a wide range of models, as well as providing information about the uncertainty of the estimate. For a detailed discussion about GPR, we refer the interested reader to Rasmussen and Williams (2006) , in particular, Chapter 2.

Let $\boldsymbol{y} \in \mathbb{R}^N$ be a vector of N observations, and denote by X the set of the corresponding n-dimensional input vectors, namely,

$$\boldsymbol{y} = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^T , \qquad X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \; .$$

Then, when modeling the input-output relation with a Gaussian process (GP), the following probabilistic model is assumed

$$\boldsymbol{y} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} = \boldsymbol{f}(X) + \boldsymbol{e}\,,$$

where $\boldsymbol{e}$ is Gaussian i.i.d. noise with standard deviation σ_n, while f is an unknown function of the inputs, modeled as a GP, namely, we have $\boldsymbol{f}(X) \sim N(\boldsymbol{m}_f, K(X,X))$; $\boldsymbol{m}_f$ and $K(X,X)$ define the prior probability distribution of f. The matrix $K(X,X)$, also named *kernel matrix*, is defined through a kernel function, i.e., a function mapping a couple of inputs vector in $\mathbb{R}$. Denote the kernel by $k(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Then, $\mathbb{E}[y_i, y_j]$, the $K(X,X)$ entry at row i and column j, is equal to $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. To be a valid kernel, i.e., to generate valid covariance matrices, $k(\cdot,\cdot)$ must be symmetric and positive semi-definite:

- $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_j, \boldsymbol{x}_i)$;
- $\boldsymbol{v}^T K(X,X)\boldsymbol{v} \geq 0 \quad \forall N \in \mathbb{N}$ and $\forall \boldsymbol{v} \in \mathbb{R}^N$.

Remarkably, starting from the Bayes rule, and exploiting the properties of the Gaussian distribution (see Rasmussen and Williams (2006) for details), we obtain that the posterior probability of f given the training samples $(X, \boldsymbol{y})$ is Gaussian, with mean and covariance that can be computed in closed form. Let $\boldsymbol{x}_*$ be a generic input location, and denote by $p\left(f\left(\boldsymbol{x}_*\right)|X, \boldsymbol{y}\right)$ the posterior probability of $f(\boldsymbol{x}_*)$ given the training samples X and $\boldsymbol{y}$. Then, we have

$$p\left(f\left(\boldsymbol{x}_*\right)|X, \boldsymbol{y}\right) \sim N\left(\hat{f}(\boldsymbol{x}_*), \operatorname{cov}\left(\hat{f}(\boldsymbol{x}_*)\right)\right), \tag{1.1a}$$

$$\hat{f}(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, X)\boldsymbol{\alpha} = K(\boldsymbol{x}_*, X)\left(K(X,X) + \sigma_n I\right)^{-1}\boldsymbol{y}\,, \tag{1.1b}$$

$$\operatorname{cov}\left(\hat{f}(\boldsymbol{x}_*)\right) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - K(\boldsymbol{x}_*, X)\left(K(X,X) + \sigma_n I\right)^{-1} K(\boldsymbol{x}_*, X)^T\,, \tag{1.1c}$$

where $K(\boldsymbol{x}_*, X) = [k(\boldsymbol{x}_*, \boldsymbol{x}_1) \ \ldots \ k(\boldsymbol{x}_*, \boldsymbol{x}_N)]$, and $\boldsymbol{\alpha} = \left(K(X,X) + \sigma_n I\right)^{-1}\boldsymbol{y}$. Notice that $\hat{f}(\boldsymbol{x}_*)$ corresponds to the maximum a posteriori (MAP) estimates. As far as computational complexity is concerned, the number of operation needed to compute (1.1b) scales with the cube of N, due to the matrix inversion required by the computation of alpha. However, notice that $\boldsymbol{\alpha}$ can be computed off-line, given that it is independent on $\boldsymbol{x}_*$. Assuming that $\boldsymbol{\alpha}$ is known, we have that the number of operations required to compute a single estimation scales linearly with the number of samples N, reducing considerably the computational burden.

A crucial aspect of GPR is the definition of the prior distribution. In particular, we focus on the kernel selection. The kernel allows encoding eventual prior information about the behaviors of the process, e.g., stationarity w.r.t. the input locations, smoothness, or linearity. Several options can be considered. We introduce two kernels particularly relevant in this manuscript. We start with the linear kernel. Let $\phi : \mathbb{R}^n \to \mathbb{R}^p$ be a function mapping an input vector in a given p-dimensional feature space. We define the linear kernel with the following expression,

$$k_\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi^T(\boldsymbol{x}_i)\Sigma\phi(\boldsymbol{x}_j)\,, \tag{1.2}$$

where the matrix $\Sigma \geq 0 \in \mathbb{R}^{p\times p}$ contains the kernel hyperparameters. A standard choice consists in assuming Σ is a diagonal matrix, with distinct diagonal elements. In this case, the diagonal elements define the relevance of each feature. The linear kernel is particularly useful because it allows exploiting eventual prior information about the system, namely, the feature space $\phi(\boldsymbol{x})$ can be defined starting from a physical model of the system. Examples of this strategy are reported in Chapter 3 and 4.

The second kernel that we describe is the Radial Basis Functions (RBF) kernel. The RBF kernel belongs to the family of stationary kernels, and it is defined by the following expression

$$k_{RBF}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \lambda e^{-\frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{x}_j||^2_{\Sigma^{-1}}}\,, \tag{1.3}$$

where the scaling factor $\lambda > 0$ and the matrix $\Sigma > 0$ are kernel hyperparameters. A standard choice consists in assuming that Σ has diagonal structure. In this case, the diagonal elements are named *lengthscales*, and they determine the variability of the function w.r.t. the relative input dimension. However, it is worth mentioning that there are also other options. For instance, Σ can be parametrized w.r.t. its Cholesky decomposition $\Sigma = LL^T$, where $L \in \mathbb{R}^{n\times n}$ is a lower triangular matrix. In this case, the set of hyperparameters is composed by λ and all the lower triangular elements of L, with the elements along the diagonal constrained to be positive.

Moreover, we mention three important properties of kernel functions, that allow generating new kernels starting from valid kernel functions:

- The sum of kernels is a valid kernel.
- The product of kernels is a valid kernel.
- Let $a(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ be a deterministic function mapping an input vector in $\mathbb{R}$, and consider a valid kernel function $k(\cdot,\cdot)$, then $g(\boldsymbol{x}, \boldsymbol{x}') = a(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{x}')a(\boldsymbol{x}')$ is a valid kernel function. This property is also named *vertical rescaling* property.

We refer the interested reader to Rasmussen and Williams (2006), Chapter 4.2.4 for the proofs.

Finally, we conclude the GPR background discussing the training procedure. The training of a GP model is composed of two steps. The first step consists in tuning σ_n and the kernel hyperparameters, while the second step consists in deriving the estimator computing $\boldsymbol{\alpha}$ in (1.1b). The main strategies proposed to tune the hyperparameters are marginal likelihood optimization and cross-validation. Cross-validation algorithms select the hyperparameters based on the performance obtained in a validation set. The limitation of such approaches is that the number of configurations to be tested grows extremely fast with the number of parameters. Due to this fact, in this work we rely on marginal likelihood maximization, hereafter equivalently named evidence maximization. The marginal likelihood is the integral of the product between the prior probability and the likelihood,

$$p\left(\boldsymbol{y}|X\right) = \int p(\boldsymbol{y}|\boldsymbol{f}, X)p(\boldsymbol{f}|X)d\boldsymbol{f}\,.$$

Recalling that $p(\boldsymbol{y}|\boldsymbol{f}, X) \sim N(\boldsymbol{f}, \sigma_n^2 I)$ and $p(\boldsymbol{f}|X) \sim N(\boldsymbol{m}_f, K(X, X))$, it can be proved that the logarithm of the previous integral corresponds to

$$-\frac{1}{2}\boldsymbol{y}^T\left(K(X,X)+\sigma_n^2 I\right)^{-1}\boldsymbol{y} - \frac{1}{2}\log\left(|K(X,X)+\sigma_n^2 I|\right) - \frac{N}{2}\log(2\pi)\,. \tag{1.4}$$

Then, to optimize the hyperparameters we maximize the previous expression w.r.t. the kernel hyperparameters. Notice that the resulting optimization problem is not convex.

As far as the implementation is concerned, we implemented all the GPR algorithms in Python. We rely on PyTorch Paszke, Gross, Chintala, Chanan, Yang, DeVito, Lin, Desmaison, Antiga, and Lerer (2017) to enable GPU support, as well as to exploit its auto-differentiation functionalities, for computing the gradient of (1.4) w.r.t. the hyperparameters by backpropagation. The library is publicly available[1].

[1] https://bitbucket.org/AlbertoDallaLibera/gpr-pytorch/src

2 Autonomous learning of the robot kinematics model

2.1 Introduction

Kinematics models are fundamental in Robotics. Several control strategies and trajectory planning algorithms are designed assuming to know a forward kinematics model, see for instance Siciliano, Sciavicco, Villani, and Oriolo (2009), Latombe (2012)

Forward kinematics models describe the map between joints configurations and links poses. When the geometrical parameters of the robot are known, first principles models of this map are derived defining a proper sequence of consecutive roto-translations. A standard way to define and parametrize a suitable sequence of roto-translations is given by the Denavit-Hartenberg (DH) convention.

Typically, precise prior knowledge about the robot geometry is available, since most of the times CAD models are available, and direct measurements of the robot parameters are possible. However, robots are becoming more and more autonomous and re-configurable, and there might be cases in which such information about geometry is not available a priori. Consider for instance Modular Robotics Yim et al. (2000, 2007); Brodbeck and Iida (2012); Guan et al. (2009); Sprowitz et al. (2010), where robots are composed of different elementary building blocks, that can be interchanged, added or removed.

Changing their geometry, modular robots adapt to different environments and tasks. Another relevant application where robot's geometry might change during the execution of a task is automatic adaptation to hardware failures. Hardware failures could result in an unexpected variation of the robot kinematics; consider, for instance, the extreme case of a robot joint getting stuck. However, in some cases, the robot might still be able to accomplish the task, by updating its model and, in turn, adapting its controller.

In the aforementioned situations, where prior knowledge about robot geometry is partial, or null, standard solutions can not be applied to derive kinematics models. In particular, the design of algorithms able to learn a geometrical model of a robot autonomously is essential to reduce set up times and human intervention, together with the associated costs.

In such context, the identification of the forward kinematics is only a subtask of the derivation of a robot geometrical model. The first step consists in identifying the kinematic structure, namely, an high level description of the robot that defines how the different components are connected. More precisely, when considering a manipulator with open kinematic chain, kinematic structure determines the order by which links and joints occur in the kinematic chain, as well as the type of each joint, e.g., if a joint is revolute or prismatic. This high level description is particularly useful in several research topics related to robotics, like, task and motion planning, transfer learning, and meta learning Finn, Yu, Zhang, Abbeel, and Levine (2017). Interestingly, it is also strictly connected with the body structure learning problem, well known in Computer Vision Dantone (2014), but also in other fields like cognitive Neuroscience.

The kinematic structure and forward kinematics identification problem has been already addressed in the robotics community. The proposed approaches can be roughly divided based on the kind of sensors adopted to collect data. In Zhou and Shi (2016), points cloud data have been used. Firstly, links are identified and ordered by clustering points according to their relative distances and by assuming that each cluster corresponds to one link. After that, a forward kinematics model is identified, optimizing the DH parameters with standard gradient descent. A more convenient setup has been considered in Hersch, L. Sauser, and Billard (2008); Lin, Rojas, and Guan (2017); Rühr, Sturm, Pangercic, Beetz, and Cremers (2012); Sturm, Plagemann, and Burgard (2008), where a distinct fiducial marker is attached to each link, making clustering operations not necessary. In particular, in Sturm et al. (2008), the kinematic structure and the forward kinematics are learned simultaneously. The solution is based on the identification of a Bayesian network. The nodes of the network are random variables associated to markers poses and joints signals, while the edges represent the relative transformations between markers. Firstly, a score to all the possible edges is assigned, relying on Gaussian Process Regression (GPR). Then, the kinematic model is learned by looking for the path that connects all the markers and minimizes a cost function suitably defined.

In this paper, we consider the same setup of Sturm et al. (2008) but, instead of learning simultaneously the kinematic structure and the forward kinematics, we propose a two steps procedure. Firstly, the kinematic structure is identified. Secondly, based on the kinematic models identified, we derive a learning algorithm for the forward kinematics based on GPR.

As far as the first step is concerned, the proposed solution is based on checking the feasibility of three systems of equations, which are obtained starting from elementary kinematic relations between pairs of subsequent links and using information extracted from time series of visual data. More precisely, given the measured poses of a couple of markers attached to subsequent links, and the corresponding joint input signal, a linear system of equations holds true if the three elements define a prismatic transformation; instead, a linear and a non linear systems are satisfied if the transformation is revolute. In general, it is possible to exhibit sets of observations for which the systems of equations hold true though the pair of markers and the joint signal considered are not in relation among them. However, by extensive Monte Carlo simulations we show that this false-positive fact is very unlikely to appear. To ensure that the feasibility of the introduced systems of equations is a necessary and sufficient condition, we need to apply our strategy with data obtained from *fully informative* sets of observations; in the paper we exhibit a class of trajectories from which it is possible to properly select observation sets which are fully informative. Compared to the state of the art, the proposed approach is less expensive as regards the computational costs, since it is based on the solution of linear and non linear systems of equations with low dimensionality. Moreover, differently from Sturm et al. (2008), the proposed algorithm fully reconstructs the kinematic structure, including the joint type sequence.

Regarding the derivation of a forward kinematics model, we do not rely on the knowledge of a prior model. As mentioned before, the standard procedure consists in deriving a first principles model based on the nominal parameters, and then improving the accuracy of the model through calibration procedures, see for instance Gao, Wang, Lin, and Chen (2009) Lin, Zhao, Ye, and Ding (2017). Typically, due to the non linearity of kinematic models, calibration procedures are effective only when deviations of the geometrical parameters are small. Unfortunately, as described before, it might happen that prior knowledge is not available. In this chapter, we propose a grey-box solution based on GPR. Starting from the knowledge of the kinematic structure, the standard inputs are mapped in a augmented space where the forward kinematics derived with DH is a polynomial function. Then, each element of the output, namely, the pose of the marker attached to the last link, are modeled as a distinct GPR with a proper polynomial kernel. Experimental results show the effectiveness of our solution. Moreover, to further test the accuracy of the derived model, we implemented a simple kinematic controller based on the model learned.

It is worth mentioning that our work significantly extends the seminal intuitions explored in Nguyen, Patel, and Khorasani (1990), where the authors have experimented the use of neural networks, comparing performance of different architectures via experiments performed on a simulated manipulator. Interestingly, results highlight that data efficiency and out of sample accuracy can be significantly improved considering as input space an augmented space built starting from the standard inputs, that are the positions of the joints.

The chapter is organized as follows. In Section 2.2, we describe the setup we consider, and we formally define the problem of interest. In Section 2.3, we provide some background about robot kinematics, and, assuming noiseless measurements, we derive the three systems of equations which represent the theoretical core of the proposed solution. We start Section 2.4 with some considerations about the measurement noises affecting our setup, and then we describe our strategy for kinematic structure identification. In Section 2.5, resorting on the identified kinematic structure, we propose our data-driven algorithm for forward kinematics estimation, as well as designing a simple kinematic controller based on the learned model. Finally, in Section 2.6, we report experimental results.

2.2 Setup description and problem formulation

In this section, we formally describe the setup and the problem we aim at solving. The framework is the same one adopted in Sturm et al. (2008), and it consists in a camera and a robotic arm, composed of n links and $n-1$ joints, forming an open kinematic chain.
Measurements of joints positions are available, and we point out with $q_k(t)$, $k = 1, \dots, n-1$, the k-th component of $\boldsymbol{q}(t)$, the vector of joints positions at time t. In this work only two types of joint are considered, revolute and prismatic. More complex connections can be derived combining these two types.
As far as the camera placement is concerned, we adopt an eyes-to-hand configuration, i.e., the camera is fixed and it is observing the robot; the camera reference frame (RF)

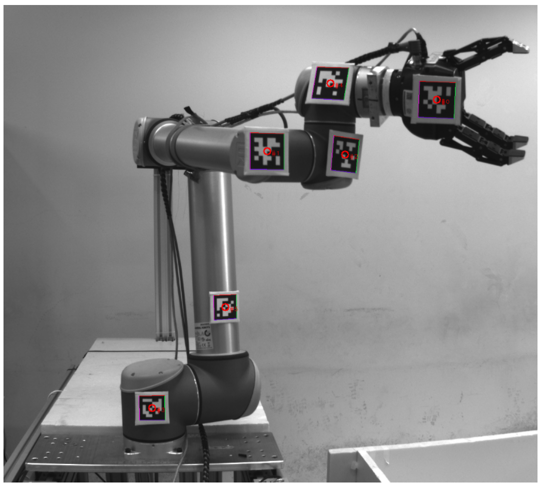

Figure 2.1: Symbolic representation of the setup. C, L and M point out respectively the camera, the link and the markers RF, while joint signals are pointed with q.

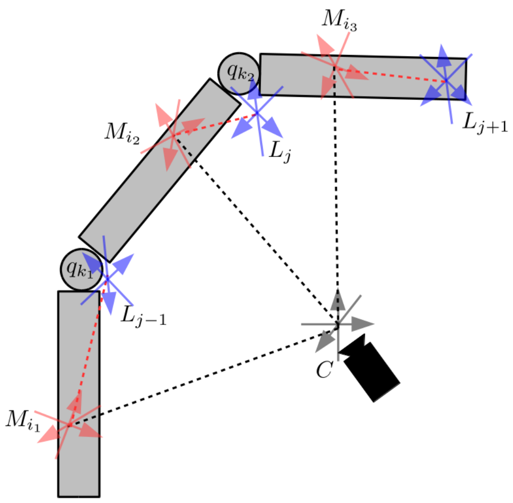

Figure 2.2: A frame collected from the camera with the robot and the fiducial markers visible.

coincides with the world RF. A distinct fiducial marker M_i, $i = 1, \ldots, n$, is attached to each of the L_j, $j = 1, \ldots, n$, links, where the subscript j denotes the position of the link in the kinematic chain; namely L_1 and L_n are, respectively, the first and the last link. Examples of fiducial markers are Fiala (2010); Olson (2011); Wang and Olson (2016). In Figure 2.2, we have reported a frame collected with the camera in our experimental setup, with the markers and the robot visible. A pictorial representation of the overall setup is reported in Figure 2.1. Notice that, in describing the setup of interest, we have used different subscripts for links, markers and joints. This is to stress that we do not know a priori the order with which joints and markers occur in the kinematic chain. For example, given link L_j, we do not know the index i of the corresponding marker M_i, as well as the index k of the joint connecting L_j and L_{j+1}.

We conclude the setup description discussing information obtained processing frames acquired from the camera. Measurements of the markers' poses in the camera RF are obtained applying solutions described in Fiala (2010); Olson (2011); Wang and Olson (2016). Let $\boldsymbol{p}^c_{M_i}(t)$ be the pose of marker M_i at time t, denoted with respect to the camera RF, hereafter indicated by c; then we have

$$\boldsymbol{p}^c_{M_i}(t) = \left[\boldsymbol{l}^c_{M_i}(t)^T \quad \boldsymbol{o}^c_{M_i}(t)^T\right]^T ,$$

where $\boldsymbol{l}^c_{M_i}$ is the vector collecting the three Cartesian coordinates $x^c_{M_i}$, $y^c_{M_i}$ and $z^c_{M_i}$ of the relative translation, while $\boldsymbol{o}^c_{M_i}$ expresses the relative orientation in yaw-pitch-roll convention; namely, $\boldsymbol{o}^c_{M_i} = \left[\gamma^c_{M_i}(t),\ \beta^c_{M_i}(t),\ \alpha^c_{M_i}(t)\right]$ where $\gamma^c_{M_i}(t)$, $\beta^c_{M_i}(t)$ and $\alpha^c_{M_i}(t)$, denote, respectively, the yaw, pitch and roll angles. For future use, we introduce also an alternative expression of relative orientations, based on rotation matrices. Consider marker M_i, its orientation in the camera RF is described by the rotation matrix $R^c_{M_i}$. The relation between the two notations is given by

$$R^c_{M_i} = R_z(\gamma^c_{M_i})R_y(\beta^c_{M_i})R_x(\alpha^c_{M_i}),$$

where R_z, R_y and R_x are the elementary rotation matrices around the z, y and x axes.

To sum up, given T observations, measurements collected are

$$\mathcal{D} = \{(P(t_1), \boldsymbol{q}(t_1)), \dots, (P(t_T), \boldsymbol{q}(t_T))\}, \tag{2.1}$$

where $P(t) = \{\boldsymbol{p}^c_{M_1}(t), \cdots, \boldsymbol{p}^c_{M_n}(t)\}$. For future use, we introduce also the following sets,

$$\begin{aligned} P_i &= \{\boldsymbol{p}^c_{M_i}(t), t = t_1, \dots, t_T\}, \\ \boldsymbol{q}_k &= \{q_k(t), t = t_1, \dots, t_T\}. \end{aligned}$$

From now on, to keep the notation compact, we point out explicitly the dependency on time only when necessary.

The main goal of this work is deriving a forward kinematics model of the manipulator, namely, to reconstruct the relation existing between the RF of the last link and the camera RF as a function of the joints variables. The solution we propose consists in performing the following two steps. First, based on $\mathcal{D}$, we apply a procedure able to identify the kinematic structure of the manipulator (see Sections 2.3 and 2.4). More precisely, the kinematic structure identification problem consists in a classification problem, decomposable into three subtasks:

- Identifying $\mathcal{S}_M = \{M_{i_1}, \dots, M_{i_n}\}$, namely, the sequence of markers associated to the kinematic chain $L_1, \dots, L_n$;
- Identifying $\mathcal{S}_Q = \{Q_{k_1}, \dots, Q_{k_{n-1}}\}$, i.e., the sequence of joint types connecting consecutive links along the kinematic chain, starting from the couple (L_1, L_2), up to the couple (L_{n-1}, L_n); more precisely, Q_{k_j} is a binary variable assuming value 0 (resp. 1) when the joint between L_j and L_{j+1} is prismatic (resp. revolute);
- Identifying $\mathcal{S}_q = \{q_{k_1}, \dots, q_{k_{n-1}}\}$, i.e., the sequence of joint signals associated to the sequence of joint types in $\mathcal{S}_Q$.

As second step, based on the knowledge of $\mathcal{S}_Q$, we run a data-driven algorithm based on GPR to derive a model of the forward kinematics (see Section 2.5).

2.3 Relations between couples of subsequent markers

In this section, we derive the theoretical building blocks of our kinematic structure identification algorithm. We consider ideal conditions, in particular to access to the real values of $\boldsymbol{q}$ and $\boldsymbol{p}^c_{M_i}$. We start considering the case that markers M_{i_1} and M_{i_2} are attached to the consecutive links L_{j_1}, L_{j_2}, connected through joint q_k. Without loss of generality we can assume $j_2 = j_1 + 1$. Elaborating the elementary kinematic relations involved by such assumption, and distinguishing between the case q_k is prismatic or revolute, we derive three systems of equations that are verified when the triplet (M_{i_1}, M_{i_2}, q_k) is connected. In the remaining part of the section instead, we consider the opposite problem, investigating if the aforementioned systems are verified if and only if the triplet (M_{i_1}, M_{i_2}, q_k) is connected.

2.3.1 Background

To provide a mathematical description of the transformations occurring along the kinematic chain we need to define a RF for each link and for each maker. As far as the links are concerned, we adopt the Denavit-Hartenberg (DH) convention; for details we refer the interested reader to Siciliano et al. (2009), chapter 2.8.2. Once the RFs of the links have been assigned, the expression of $R^{L_j}_{L_{j-1}}$, i.e., the relative orientation between the consecutive links L_{j-1} and L_j, is given by

$$R^{L_{j-1}}_{L_j} = R_z(\theta_j) R_x(\alpha_j), \tag{2.2}$$

where α_j a constant parameter (see Figure 2.16 on page 62 of Siciliano et al. (2009) for a pictorial description of α_j and θ_j). In case the joint connecting L_{j-1} and L_j is prismatic, then θ_j is constant and equal to θ^0_j, while, if the joint is revolute and controlled by q_k, it holds $\theta_j = \theta^0_j + q_k$. The relation between the relative positions of L_{j-1} and L_j is described by $\boldsymbol{l}^{L_{j-1}}_{L_j}$, i.e., the expression of the origin of L_j-RF with respect to the origin of L_{j-1}-RF; we have that

$$\boldsymbol{l}^{L_{j-1}}_{L_j} = \begin{bmatrix} 0 \\ 0 \\ d_j \end{bmatrix} + R_z(\theta_j) \begin{bmatrix} a_j \\ 0 \\ 0 \end{bmatrix}, \tag{2.3}$$

where $R_z(\theta_j)$ is defined as before, and a_j is a constant parameter of the kinematic (again see Figure 2.16 on page 62 of Siciliano et al. (2009) for a pictorial description of a_j and d_j). If the joint connecting L_{j-1} and L_j is revolute then d_j is constant and equal to d_j^0, while, if it is prismatic and parametrized by q_k, then it holds $d_j = d_j^0 + q_k$.

Additionally, we need to describe the pose of each marker with respect to the RF of the link it is attached to. For example, suppose that M_{i_1} is attached to L_{j_1}, then position and orientation of M_{i_1} w.r.t. L_{j_1} are described, respectively, by $\boldsymbol{l}_{M_{i_1}}^{L_{j_1}}$ and $R_{M_{i_1}}^{L_{j_1}}$. Observe that, since the markers are assume to be rigidly attached to the corresponding links, $\boldsymbol{l}_{M_{i_1}}^{L_{j_1}}$ and $R_{M_{i_1}}^{L_{j_1}}$ are constant and independent of the joint signals. For later use, it is convenient introducing also $R_{L_{j_1}}^{M_{i_1}} = (R_{M_{i_1}}^{L_{j_1}})^T$ and $\boldsymbol{l}_{L_{j_1}}^{M_{i_1}}$. Similar definitions hold for M_{i_2} and L_{j_2}, assuming M_{i_2} is attached to L_{j_2}.

We stress that $\boldsymbol{l}_{M_{i_1}}^{L_{j_1}}$ and $R_{M_{i_1}}^{L_{j_1}}$ are unknown, and that we do not introduce any limitation on the way the markers are attached to links. From a practical point of view, this fact is very interesting, since it allows adopting the proposed algorithm even in setups different from the one we described in Section 2.2. For instance, it might happen that fiducial markers are not available and the use of ad-hoc computer-vision algorithms is required to detect and track fictitious markers Kim and Yoon (2017); Yun, Lee, Lee, and Kim (2017); Kim, Lee, Kim, Kim, and Han (2012). In this context the markers placement is not controllable, but it still holds that $\boldsymbol{l}_{M_{i_1}}^{L_{j_1}}$ and $R_{M_{i_1}}^{L_{j_1}}$ are constant.

Based on the quantities above introduced, and exploiting standard algebraic properties of rotation matrices, we derive two relations describing the relative pose between consecutive markers. Let $\boldsymbol{l}_{M_{i_2}}^{M_{i_1}}$ and $R_{M_{i_2}}^{M_{i_1}}$ be, respectively, the Cartesian coordinates of the origin of M_{i_2}-RF w.r.t. M_{i_1}-RF, and the relative orientation between M_{i_2}-RF and M_{i_1}-RF. Then we have

$$\boldsymbol{l}_{M_{i_2}}^{M_{i_1}} = \boldsymbol{l}_{L_{j_1}}^{M_{i_1}} + R_{L_{j_1}}^{M_{i_1}} \boldsymbol{l}_{L_{j_2}}^{L_{j_1}} + R_{M_{i_2}}^{M_{i_1}} \left(-\boldsymbol{l}_{L_{j_2}}^{M_{i_2}} \right). \tag{2.4}$$

and

$$R_{M_{i_2}}^{M_{i_1}} = R_{L_{j_1}}^{M_{i_1}} R_{L_{j_2}}^{L_{j_1}}(\theta_{j_2}) R_{M_{i_2}}^{L_{j_2}}. \tag{2.5}$$

Before investigating the above relations distinguishing the case where the joint connecting two successive links is prismatic from the case where the joint is revolute, we remark that, in the described setup, we measure indirectly the relative translation and orientation between markers, since

$$\boldsymbol{l}_{M_{i_2}}^{M_{i_1}} = (R_{M_{i_1}}^c)^T (\boldsymbol{l}_{M_{i_2}}^c - \boldsymbol{l}_{M_{i_1}}^c),$$

and

$$R_{M_{i_2}}^{M_{i_1}} = (R_{M_{i_1}}^c)^T R_{M_{i_2}}^c,$$

where $R^c_{M_{i_1}}$, $R^c_{M_{i_2}}$, $\boldsymbol{l}^c_{M_{i_1}}$ and $\boldsymbol{l}^c_{M_{i_2}}$ are obtained processing the data coming from the camera.

2.3.2 Prismatic joint

Assume that the joint connecting L_{j_1} and L_{j_2} is prismatic and let q_k be the corresponding joint variable. Since in this case the angle θ_j is constant, it follows that the relative orientation between L_{j_1} and L_{j_2} is not affected by variations of the joint variable, that is, the matrix $R^{M_{i_1}}_{M_{i_2}}$ is also constant over the time. By substituting the expression of $\boldsymbol{l}^{L_{j_1}}_{L_{j_2}}$ given in (2.3) into (2.4), the following equation holds

$$\begin{aligned}
\boldsymbol{l}^{M_{i_1}}_{M_{i_2}} =& \boldsymbol{l}^{M_{i_1}}_{L_{j_1}} + R^{M_{i_1}}_{L_{j_1}} \left(\begin{bmatrix} 0 \\ 0 \\ d^0_j + q_k \end{bmatrix} + R_z(\theta^0_j) \begin{bmatrix} a^0_j \\ 0 \\ 0 \end{bmatrix} \right) + R^{M_{i_1}}_{M_{i_2}} \left(-\boldsymbol{l}^{M_{i_2}}_{L_{j_2}} \right) \\
=& \boldsymbol{l}^{M_{i_1}}_{L_{j_1}} + R^{M_{i_1}}_{L_{j_1}} \left(\begin{bmatrix} 0 \\ 0 \\ d^0_j \end{bmatrix} + R_z(\theta^0_j) \begin{bmatrix} a^0_j \\ 0 \\ 0 \end{bmatrix} \right) + R^{M_{i_1}}_{M_{i_2}} \left(-\boldsymbol{l}^{M_{i_2}}_{L_{j_2}} \right) + R^{M_{i_1}}_{L_{j_1}} \begin{bmatrix} 0 \\ 0 \\ q_k \end{bmatrix}, \qquad (2.6)
\end{aligned}$$

where the first three addenda of the right hand side of the last equation are constant, and they can be compacted in the vector $\boldsymbol{l}^{i_1}_{i_2}$, while the last term has constant direction and module that depends on the joint coordinate q_k. In addition, observe that (2.6) is linear w.r.t. $\boldsymbol{l}^{i_1}_{i_2}$ and the third column of $R^{M_{i_1}}_{L_{j_1}}$, that we denote hereafter by $\boldsymbol{z}^{M_{i_1}}_{L_{j_1}}$. Then, we can write

$$\boldsymbol{l}^{M_{i_1}}_{M_{i_2}} = \begin{bmatrix} I_3 & q_k I_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{l}^{i_1}_{i_2} \\ \left(\boldsymbol{z}^{M_{i_1}}_{L_{j_1}} \right) \end{bmatrix} = A(q_k) \boldsymbol{b}^{i_2}_{i_1} , \qquad (2.7)$$

where

$$A(q_k) = \begin{bmatrix} I_3 & q_k I_3 \end{bmatrix}$$

is the vector of regressors, and

$$\boldsymbol{b}^{i_2}_{i_1} = \begin{bmatrix} \boldsymbol{l}^{i_1}_{i_2} \\ \boldsymbol{z}^{M_{i_1}}_{L_{j_1}} \end{bmatrix}$$

is the vector collecting the parameters of the linear system. Since the last three elements of $\boldsymbol{b}^{i_2}_{i_1}$ are the column of a rotation matrix, we have that the following constraint must be satisfied

$$\left(\boldsymbol{z}^{M_{i_1}}_{L_{j_1}} \right)^T \boldsymbol{z}^{M_{i_1}}_{L_{j_1}} = 1.$$

It is easy to see that, by introducing the block diagonal matrix $H = \text{block diag}\,(0_{3\times3}, I_3)$, where $0_{3\times3}$ is a 3×3 matrix of zeros, the last equation can be written in terms of $\boldsymbol{b}_{i_1}^{i_2}$, as

$$\left(\boldsymbol{b}_{i_1}^{i_2}\right)^T H\boldsymbol{b}_{i_1}^{i_2} = 1. \tag{2.8}$$

If, instead of considering a single observation, we consider the T observations in the set $\mathcal{D}$ defined in (2.1), we obtain the following linear system of $3T$ equations,

$$\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right) = \begin{bmatrix} A\left(q_k(t_1)\right) \\ \vdots \\ A\left(q_k(t_T)\right) \end{bmatrix} \boldsymbol{b}_{i_1}^{i_2} = \mathcal{A}(\boldsymbol{q}_k)\boldsymbol{b}_{i_1}^{i_2}, \tag{2.9}$$

where $\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right)$ is the vector collecting the relative translations between markers M_{i_1} and M_{i_2}, namely

$$\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right) = \left[(\boldsymbol{l}_{M_{i_2}}^{M_{i_1}}(t_1))^T, \dots, (\boldsymbol{l}_{M_{i_2}}^{M_{i_1}}(t_T))^T\right]^T.$$

We remark that $\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right)$ contains information that can be indirectly measured, also $\mathcal{A}(\boldsymbol{q}_k)$ is known, while $\boldsymbol{b}_{i_1}^{i_2}$ is the vector of the unknown variables. We have the following Proposition.

Proposition 2.3.1. *Consider two markers M_{i_1} and M_{i_2}, attached to consecutive links connected through a prismatic joint, whose corresponding joint signal is q_k. Then, given a set of observations $\mathcal{D}$, the relative orientation between markers is constant, i.e., $R_{M_{i_2}}^{M_{i_1}}$ is constant, and the linear system in* (2.9) *admits solution in $\boldsymbol{b}_{i_1}^{i_2}$ satisfying the constraint in* (2.8)*.*

2.3.3 Revolute joint

Now assume that the joint connecting L_{j_1} and L_{j_2} is revolute and that q_k is the corresponding joint variable. Starting from (2.4), and substituting $\boldsymbol{l}_{L_{j_2}}^{L_{j_1}}$ with the expression in

(2.3), we have

$$\begin{aligned}
\boldsymbol{l}^{M_{i_1}}_{M_{i_2}} =& \boldsymbol{l}^{M_{i_1}}_{L_{j_1}} + R^{M_{i_1}}_{L_{j_1}} \left(\begin{bmatrix} 0 \\ 0 \\ d_{j_2} \end{bmatrix} + R^{L_{j_1}}_{L_{j_2}}(\theta_{j_2}) \begin{bmatrix} a_{j_2} \\ 0 \\ 0 \end{bmatrix} \right) + R^{M_{i_1}}_{M_{i_2}} \left(-\boldsymbol{l}^{M_{i_2}}_{L_{j_2}} \right) \\
=& \boldsymbol{l}^{M_{i_1}}_{L_{j_1}} + R^{M_{i_1}}_{L_{j_1}} \begin{bmatrix} 0 \\ 0 \\ d_{j_2} \end{bmatrix} + R^{M_{i_1}}_{M_{i_2}} R^{M_{i_2}}_{L_{j_2}} \begin{bmatrix} a_{j_2} \\ 0 \\ 0 \end{bmatrix} + R^{M_{i_1}}_{M_{i_2}} \left(-\boldsymbol{l}^{M_{i_2}}_{L_{j_2}} \right), \qquad (2.10)
\end{aligned}$$

where, as before, the different contributions have been grouped conveniently. In particular the first two terms are constant, while the last two are linear w.r.t. $R^{M_{j_1}}_{M_{j_2}}$. As consequence, (2.10) is linear w.r.t. the vector of variables

$$\bar{\boldsymbol{b}}^{i_1}_{i_2} = \begin{bmatrix} \boldsymbol{l}^{M_{i_1}}_{L_{j_1}} + R^{M_{i_1}}_{L_{j_1}} \begin{bmatrix} 0 \\ 0 \\ d_{j_2} \end{bmatrix} \\ R^{M_{i_1}}_{L_{j_2}} \begin{bmatrix} a_{j_2} \\ 0 \\ 0 \end{bmatrix} - \boldsymbol{l}^{M_{i_2}}_{L_{j_2}} \end{bmatrix},$$

given that

$$\boldsymbol{l}^{M_{i_1}}_{M_{i_2}} = \begin{bmatrix} I_3 & R^{M_{i_1}}_{M_{i_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{l}^{M_{i_1}}_{L_{j_1}} + R^{M_{i_1}}_{L_{j_1}} \begin{bmatrix} 0 \\ 0 \\ d_{j_2} \end{bmatrix} \\ R^{M_{i_1}}_{L_{j_2}} \begin{bmatrix} a_{j_2} \\ 0 \\ 0 \end{bmatrix} - \boldsymbol{l}^{M_{i_2}}_{L_{j_2}} \end{bmatrix} = \bar{A}(\boldsymbol{p}^c_{M_{i_1}}, \boldsymbol{p}^c_{M_{i_2}}) \bar{\boldsymbol{b}}^{i_1}_{i_2}. \qquad (2.11)$$

The last equation decomposes $\boldsymbol{l}^{M_{i_1}}_{M_{i_2}}$ in the sum of two vectors, one that is constant, and one with constant module and direction that depends on the $R^{M_{i_1}}_{L_{j_1}}$.

When considering the set of observations $\mathcal{D}$, we obtain the following $3T$ linear equations

$$\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right) = \begin{bmatrix} \bar{A}\left(\boldsymbol{p}^c_{M_{i_1}}(t_1), \boldsymbol{p}^c_{M_{i_2}}(t_1)\right) \\ \vdots \\ \bar{A}\left(\boldsymbol{p}^c_{M_{i_1}}(t_T), \boldsymbol{p}^c_{M_{i_2}}(t_T)\right) \end{bmatrix} \bar{\boldsymbol{b}}^{i_1}_{i_2} = \bar{\mathcal{A}}(P_{i_1}, P_{i_2}) \bar{\boldsymbol{b}}^{i_1}_{i_2}, \qquad (2.12)$$

where $\boldsymbol{l}\,(P_{i_1}, P_{i_2})$ and $\bar{\mathcal{A}}(P_{i_1}, P_{i_2})$ are known, while $\bar{\boldsymbol{b}}_{i_2}^{i_1}$ is the vector of unknown variables. We have the following Proposition.

Proposition 2.3.2. *Let M_{i_1} and M_{i_2} be two markers attached to consecutive links, and assume that the joint between the two links is revolute. Then, the linear system of equations defined by the set of observations $\mathcal{D}$ in (2.12) admits solution in $\bar{\boldsymbol{b}}_{i_2}^{i_1}$.*

Observe that in (2.12) the dependence on the joint signal is not made explicit, being incorporated into $R_{M_{i_2}}^{M_{i_1}}$. To derive an expression involving the joint signal q_k we analyze the relative orientation between markers M_{i_1} and M_{i_2}. Notice that, differently from the prismatic case, matrix $R_{M_{i_2}}^{M_{i_1}}$ depends on q_k. Expanding $R_{L_{j_2}}^{L_{j_1}}$ with the expression in (2.2), we rewrite (2.5), i.e., the relative orientation between markers as

$$R_{M_{i_2}}^{M_{i_1}} = R_{L_{j_1}}^{M_{i_1}} R_z(q_k + \theta_{j_2}^0) R_x(\alpha_{j_2}) R_{M_{i_2}}^{L_{j_2}} = R_{L_{j_1}}^{M_{i_1}} R_z(q_k) R_{M_{i_2}}^{\bar{L}_{j_2}}, \tag{2.13}$$

where, exploiting the properties of elementary rotation matrices, we have $R_{M_{i_2}}^{\bar{L}_{j_2}} = R_z(\theta_{j_2}^0) R_x(\alpha_{j_2}) R_{M_{i_2}}^{L_{j_2}}$. The matrix equation in (2.13) defines nine non linear equations in the elements of the unknown matrices $R_{M_{i_2}}^{\bar{L}_{j_2}}$ and $R_{L_{j_1}}^{M_{i_1}}$. Additionally, since $R_{M_{i_2}}^{\bar{L}_{j_2}}$ and $R_{L_{j_1}}^{M_{i_1}}$ are rotation matrices, the following constraints must be satisfied

$$R_{L_{j_1}}^{M_{i_1}} \left(R_{L_{j_1}}^{M_{i_1}}\right)^T = I_3 \;,\; R_{M_{i_2}}^{\bar{L}_{j_2}} \left(R_{M_{i_2}}^{\bar{L}_{j_2}}\right)^T = I_3.$$

A more compact description can be obtained relying on the use of rotation angles. Firstly, instead of considering as unknowns the elements of $R_{M_{i_2}}^{\bar{L}_{j_2}}$ and $R_{L_{j_1}}^{M_{i_1}}$, we describe both rotation matrices using the yaw, pitch and roll angles, collected, in the two vectors

$$\boldsymbol{o}_{L_{j_1}}^{M_{i_1}} = [\gamma_{L_{j_1}}^{M_{i_1}}, \beta_{L_{j_1}}^{M_{i_1}}, \alpha_{L_{j_1}}^{M_{i_1}}]^T \;,\; \boldsymbol{o}_{M_{i_2}}^{L_{j_2}} = [\gamma_{M_{i_2}}^{L_{j_2}}, \beta_{M_{i_2}}^{L_{j_2}}, \alpha_{M_{i_2}}^{L_{j_2}}]^T.$$

Then we have

$$R_{L_{j_1}}^{M_{i_1}} = R_z(\gamma_{L_{j_1}}^{M_{i_1}}) R_y(\beta_{L_{j_1}}^{M_{i_1}}) R_x(\alpha_{L_{j_1}}^{M_{i_1}}) \;,\; R_{M_{i_2}}^{\bar{L}_{j_2}} = R_z(\gamma_{M_{i_2}}^{L_{j_2}}) R_y(\beta_{M_{i_2}}^{L_{j_2}}) R_x(\alpha_{M_{i_2}}^{L_{j_2}}). \tag{2.14}$$

In this way, we enforce orthogonality, while reducing the number of unknowns of the system. Secondly, also (2.13) can be rewritten w.r.t. the yaw pitch and roll angles, obtaining

$$\boldsymbol{o}_{M_{i_2}}^{M_{i_1}} = f_o\left(R_{L_{j_1}}^{M_{i_1}} R_z(q_k) R_{M_{i_2}}^{\bar{L}_{j_2}}\right),$$

where f_o is a function that convert a rotation matrix in yaw, pitch and roll angles.

If, instead of considering a single observation, we consider a set of observations $\mathcal{D}$, we obtain $3T$ non linear equations,

$$\begin{bmatrix} \boldsymbol{o}_{M_{i_2}}^{M_{i_1}}(t_1) \\ \vdots \\ \boldsymbol{o}_{M_{i_2}}^{M_{i_1}}(t_T) \end{bmatrix} = \begin{bmatrix} f_o\left(R_{L_{j_1}}^{M_{i_1}} R_z\left(q_k(t_1)\right) R_{M_{i_2}}^{\bar{L}_{j2}}\right) \\ \vdots \\ f_o\left(R_{L_{j_1}}^{M_{i_1}} R_z\left(q_k(t_T)\right) R_{M_{i_2}}^{\bar{L}_{j2}}\right) \end{bmatrix}.$$

In the following, for compactness, we denote the left and the right hand side of the previous equation by, respectively, $\boldsymbol{o}\left(P_{i_1}, P_{i_2}\right)$ and $f_o(\boldsymbol{q}_k, R_{L_{j_1}}^{M_{i_1}}, R_{M_{i_2}}^{\bar{L}_{j2}})$, obtaining

$$\boldsymbol{o}\left(P_{i_1}, P_{i_2}\right) = f_o(\boldsymbol{q}_k, R_{L_{j_1}}^{M_{i_1}}, R_{M_{i_2}}^{\bar{L}_{j2}}). \tag{2.15}$$

We have the following proposition.

Proposition 2.3.3. *Let M_{i_1} and M_{i_2} be two markers attached to consecutive links connected through the revolute joint q_k. Then, given a set of observations $\mathcal{D}$, the corresponding non linear system of equations defined by* (2.14) *and* (2.15) *admits a solution, i.e., there exist two triplets of yaw, pitch and roll angles defining $R_{L_{j_1}}^{M_{i_1}}$ and $R_{M_{i_2}}^{\bar{L}_{j2}}$ satisfying* (2.15).

2.3.4 Fully informative trajectories

In the previous Section we have stated three propositions defining conditions that are verified when M_{i_1} and M_{i_2} are attached to consecutive links. In general, the reverse relations are not true since it is possible to exhibit sets of observations $\mathcal{D}$ such that the conditions of the previous Propositions are satisfied even if the markers are not attached to subsequent links; in the following we will refer to such cases as *false positives.*

False positive observations are strictly related to the sequences of joint configurations which have generated the set of observations $\mathcal{D}$. In the following, we denote by *fully informative* observations a set of observations $\mathcal{D}$ such that conditions defined in Propositions 1, for the prismatic joint, and in Propositions 2 and 3, for the revolute joint, are necessary and sufficient to verify if two markers are attached to consecutive links. A full characterization of this class is not trivial. However, in this chapter, we provide a class of input trajectories from which it is possible to properly select a set of *fully informative* observations.

Definition 2.3.4. Consider a collection of $n-1$ joints trajectories, where in each trajectory only one joint is actuated while all the others are kept stuck. For $k = 1, \ldots, n-1$, assume that the k-th trajectory is obtained varying the joint signal q_k, and that there exist at least two time instants $t_{1,k}$ and $t_{2,k}$ such that

$$\mathrm{mod}\,(q_k(t_{1,k})) \neq \mathrm{mod}\,(q_k(t_{2,k})) \tag{2.16}$$

and $q_w(t_{1,k}) = q_w(t_{2,k})$, $w \neq k$, where $\mathrm{mod}(\cdot)$ is the 2π module operator. Then, we define the set $\bar{\mathcal{D}}$, as the collection of the pairs $(P(t_{1,k}), \boldsymbol{q}(t_{1,k})), (P(t_{2,k}), \boldsymbol{q}(t_{2,k}))$, $k = 1, \dots, n-1$, that is,

$$\bar{\mathcal{D}} = \{(P(t_{1,k}), \boldsymbol{q}(t_{1,k})), (P(t_{2,k}), \boldsymbol{q}(t_{2,k}))\}_{k=0,\dots,n-1}$$

Notice that $\bar{\mathcal{D}}$ has $2(n-1)$ observations, which for typical robots (i.e., $n = 3, 4, 5, 6, 7$) represents a limited number of observations. It is possible to show that $\bar{\mathcal{D}}$ is a fully informative set, and, in particular, we have the following results.

Proposition 2.3.5. *Let (M_{i_1}, M_{i_2}, q_k) be a triplet satisfying conditions of Proposition 2.3.1, for the set of observations $\bar{\mathcal{D}}$ defined in Definition 2.3.4. Then, the corresponding links L_{j_1} and L_{j_2} are subsequent in the kinematic chain, and the joint connecting them is prismatic with input signal q_k.*

Proposition 2.3.6. *Let (M_{i_1}, M_{i_2}, q_k) be a triplet satisfying equations in Proposition 2.3.2 and Proposition 2.3.3, for the set of observations $\bar{\mathcal{D}}$ defined in Definition 2.3.4. Then, the corresponding links L_{j_1} and L_{j_2} are subsequent in the kinematic chain, and the joint connecting them is revolute with input signal q_k.*

The proofs of the above Propositions are reported in the Appendix.

We conclude this section with a remark about Propositions 2.3.5 and 2.3.6. The two propositions require suitable excitation assumptions, that might not be verified for general input trajectories. As consequence, in order to apply the results of the two propositions, an ad-hoc actuation strategy should be designed to collect observations satisfying conditions in Definition 2.3.4. However, we would like to point out that the set of observations in (2.3.4) represents just a small subset of the class of *fully informative* observations. Indeed, through extensive Monte Carlo simulations, reported in Section 2.6.1, we show that selecting a set of observations not *fully informative* from generic trajectories seems to be a very unlikely event. In order to not interrupt the flow of the presentation, we leave this discussion to Section 2.6.1. In the next section, we describe our kinematic structure classification algorithm, assuming that the observations collected are *fully informative*, regardless of whether trajectories belong to the class in Definition 2.3.4 or not.

2.4 Proposed approach for kinematic structure identification

In this section, we describe the algorithm we propose to deal with the robot kinematic structure classification problem. Firstly, driven by Proposition 2.3.5 and 2.3.6, we propose an iterative algorithm for the noiseless case. Assuming these ideal conditions, as well as that the set of observations $\mathcal{D}$ is *fully informative*, we can prove the convergence of the algorithm to the correct solution. Then, after discussing the sources of noise present in our setup, we modify the algorithm to deal with the non-ideal conditions.

2.4.1 Kinematic structure classification with ideal conditions

Results of Proposition 2.3.5 and 2.3.6, suggest that to determine if a triplet (M_{i_1}, M_{i_2}, q_k) is connected, and, in case, if q_k is revolute or prismatic, we can check the feasibility of the conditions defined by Propositions 2.3.1, 2.3.2 and 2.3.3. From the computational point of view, a convenient strategy is proposed in Figure 2.3. Specifically, given the pair of markers (M_{i_1}, M_{i_2}) and a joint signal q_k, we firstly evaluate if the markers are connected through a prismatic joint and, in case this test is negative, we secondly evaluates if they are connected through a revolute joint. The first test consists in checking the feasibility of the linear equations defined in Proposition 2.3.1. The second test, instead, is composed by two steps; the first step verifies if the linear equations of Proposition 2.3.2 admit solution, and, in such case, also the second step is performed, which consists in solving the system of non-linear equations of Proposition 2.3.3. Observe that, in this way, the last step is performed only when it is necessary, thus, minimizing its executions. This fact is particularly relevant from the computational point of view, since the non linear test is the most expensive.

Then, to identify the kinematic structure of the robot, we propose an algorithm that applies iteratively the procedure described in the flow chart in Figure 2.3. The algorithm returns $\hat{\mathcal{S}}_M$, $\hat{\mathcal{S}}_Q$ and $\hat{\mathcal{S}}_q$, the estimates of $\mathcal{S}_M$, $\mathcal{S}_Q$ and $\mathcal{S}_q$, namely, the sequences of markers, joint types and joint signals defining the kinematic structure of the robot. $\hat{\mathcal{S}}_M$, $\hat{\mathcal{S}}_Q$ and $\hat{\mathcal{S}}_q$ are initialized as empty sets. First, the algorithm identifies the marker associated to the base link, appends it to $\hat{\mathcal{S}}_M$, and set it as the current marker. The identification of this marker is trivial, given that its pose in the camera RF is constant. After that, the algorithm selects a triplet, adding to the current marker a marker and a joint signal randomly picked among the ones that are not in $\hat{\mathcal{S}}_M$ and $\hat{\mathcal{S}}_q$. Then, the algorithm checks if data associated with the triplet satisfy conditions of Proposition 2.3.1, 2.3.2 and 2.3.3, as described in Figure 2.3. Based on the results obtained, $\hat{\mathcal{S}}_M$, $\hat{\mathcal{S}}_Q$ and

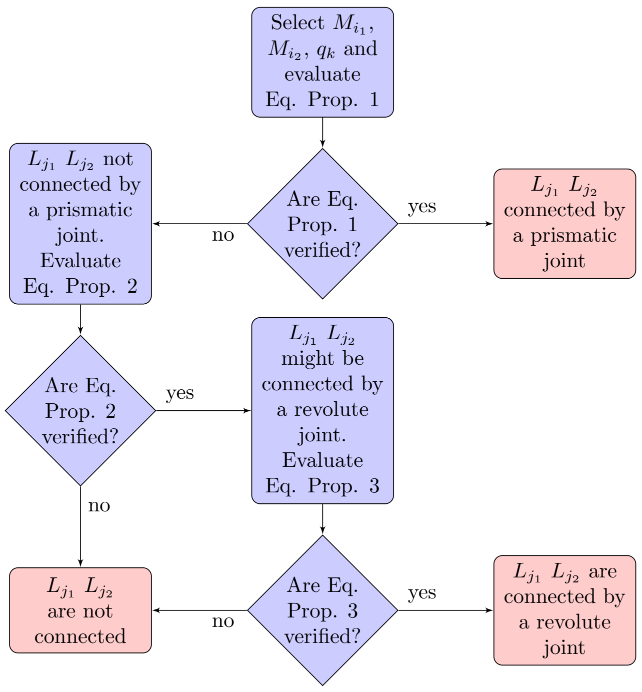
Select M_{i_1}, M_{i_2}, q_k and evaluate Eq. Prop. 1
Are Eq. Prop. 1 verified?
yes
no
L_{j_1} L_{j_2} connected by a prismatic joint
L_{j_1} L_{j_2} not connected by a prismatic joint. Evaluate Eq. Prop. 2
Are Eq. Prop. 2 verified?
yes
no
L_{j_1} L_{j_2} might be connected by a revolute joint. Evaluate Eq. Prop. 3
Are Eq. Prop. 3 verified?
yes
no
L_{j_1} L_{j_2} are connected by a revolute joint
L_{j_1} L_{j_2} are not connected

Figure 2.3: Flow chart of an iteration of the proposed algorithm. Ending conditions are highlighted in red.

$\hat{\mathcal{S}}_q$, are updated. In case that conditions are verified, the marker previously selected is set as current marker, and the procedure iterated over a new triplet.

2.4.2 Sources of noise

The algorithm proposed in the previous subsection can not be applied directly to observations acquired in a real setup, since, due to the presence of noise, the systems of equations defined by Proposition 2.3.1, 2.3.2 and 2.3.3 are, in general, not satisfied even if the triplet considered is connected.

In our setup, the presence of noise is related to several aspects. A significant contribution is due to noise in markers' poses. Markers' poses are computed processing images coming from the camera, that are corrupted by quantization errors due to the digitization of signal. The effects of such errors in the estimated pose depend on different aspects, like the distance between the camera and the marker, or the angles between the camera optical axis and the axes perpendicular to the marker. The dependence on several variables, together with the non linearity of the quantization error, make difficult to derive a rigorous mathematical description of the errors. Nonetheless, several experimental studies have been performed, see for instance Olson (2011), Wang and Olson (2016) and López-Cerón (2016); beside confirming the non linear behavior of the errors, the obtained results show that, for a considerable range of distances and inclination angles, markers' poses are estimated very accurately. For instance, in López-Cerón (2016), when the distance marker-camera is smaller than 4 meters, position errors are in the order of millimeters, while errors on the yaw, pitch and roll angles are less than 10^{-2} radiants. Experimental evidence shows that to obtain similar performance when the distance is higher, a certain inclination angle between the marker and the camera is needed, i.e., the marker must be quite far from being parallel to the camera. Moreover, results show also that outside the angular and distance ranges providing accurate estimates, errors grows rapidly, possibly resulting in the presence of outliers.

We conclude the characterization of noise sources mentioning other two contributions. The first one is the measurement noise of joint positions. Typically, compared to the errors in markers poses, these errors are smaller since encoders measure angular positions with high precision. The second contribution is due to synchronization errors between the camera and the robot, in particular when acquisition is not regulated with an ad-hoc board. Notice that these errors could be particularly evident when the robot is moving at high velocities, and then their effects can be mitigated moving the robot at small velocities.

2.4.3 Kinematic structure classification with non-ideal conditions

In this subsection, we modify the algorithm described in Subsection 2.4.1, to deal with the presence of noise. Our solution is based on the following idea. To exploit results of Proposition 2.3.5 and 2.3.6 even with noisy observations, instead of looking for the exact solutions of the systems defined in Proposition 2.3.1, 2.3.2 and 2.3.3, we solve a least squares problem for each of the three systems, minimizing the residuals. Assuming that observations with outliers can be removed, i.e., that the observations' noises are small, it is reasonable to expect a small residual for a system of equations that would be verified in absence of noise, and, on the contrary, a high residual when not verified. Based on this idea, we introduce a threshold ξ, and a function $f_r(\cdot)$ depending on the residuals: the test on (M_{i_1}, M_{i_2}, q_k) is assumed to be positive if the value of f_r is lower than ξ, and negative otherwise.

Before formulating the three optimization problems related to Proposition 2.3.1, 2.3.2 and 2.3.3, it is worth providing some more details about f_r and the thresholds. As far as f_r is concerned, we consider the same function for all the three tests. In particular, given a vector of residuals $\boldsymbol{r} \in \mathbb{R}^N$, in our algorithm, f_r is defined as

$$f_r(\boldsymbol{r}) = \sqrt{\frac{\boldsymbol{r}^T\boldsymbol{r}}{N}}.$$

The function f_r corresponds to the root of the mean squared residuals, and provides a statistics about the magnitude of the residuals. This value is compared with a threshold that characterizes the expected magnitude of the residuals caused by measurement noises. Notice that the outputs in (2.9) and (2.12) are distances, while in (2.15) angles. Then, also due to the different type of uncertainties with which translation and orientation are estimated, we introduce two thresholds, ξ_t and ξ_o. The first provides a bound on the uncertainties about distances, and it is used to test (2.9) and (2.12), while the second refers to the uncertainty about angles, and it is used to test (2.15). In the remaining part of this section, we describe the optimization problems that the modified algorithm solves for each of the three tests.

Prismatic joint

Given $\boldsymbol{b} \in \mathbb{R}^6$, let $\boldsymbol{r}_{i_1}^{i_2}(\boldsymbol{b})$ be the vector collecting the residuals of linear system in (2.9), namely,

$$\boldsymbol{r}_{i_1}^{i_2}(\boldsymbol{b}) = \boldsymbol{l}(P_{i_1}, P_{i_2}) - \mathcal{A}(\boldsymbol{q}_k)\boldsymbol{b}.$$

Let $\hat{\boldsymbol{b}}_{i_1}^{i_2}$ be a solution of the following constrained linear least squared problem,

$$\hat{\boldsymbol{b}}_{i_1}^{i_2} = \arg\min_{\boldsymbol{b}\in\mathcal{H}} \left(\boldsymbol{r}_{i_1}^{i_2}(\boldsymbol{b})\right)^T \boldsymbol{r}_{i_1}^{i_2}(\boldsymbol{b}), \tag{2.17}$$

where $\mathcal{H} = \{\boldsymbol{b} \in \mathbb{R}^6 \text{ s.t. } \boldsymbol{b}^T H \boldsymbol{b} = 1\}$. Then, the test is assumed to be positive if $f_r\left(\boldsymbol{r}_{i_1}^{i_2}(\hat{\boldsymbol{b}}_{i_1}^{i_2})\right)$ is lower than ξ_t, and negative otherwise.

Notice that Proposition 2.3.5 defines also a condition on the relative orientation between markers, that must be constant. As a consequence, when considering noisy observations, we expect that the elements of $R_{M_{i_2}}^{M_{i_1}}$ have small variances. Based on this observation, in addition to testing the value of f_r, we compute the variances of the measures of $R_{M_{i_2}}^{M_{i_1}}$, and check if the maximum value is below a given threshold.

Revolute joint - linear test

Given $\boldsymbol{b} \in \mathbb{R}^6$, let $\bar{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{b})$ be the vector collecting the residuals of the linear system (2.12), that is

$$\bar{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{b}) = \boldsymbol{l}(P_{i_1}, P_{i_2}) - \bar{\mathcal{A}}(P_{i_1}, P_{i_2})\boldsymbol{b}.$$

Then we solve the following unconstrained linear least squared problem,

$$\hat{\bar{\boldsymbol{b}}}_{i_1}^{i_2} := \arg\min_{\boldsymbol{b}} \left(\bar{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{b})\right)^T \bar{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{b}). \tag{2.18}$$

Observe that, under the assumption that $\bar{A}(P_{i_1}, P_{i_2})$ is full rank, $\hat{\bar{\boldsymbol{b}}}_{i_1}^{i_2}$ can be computed in closed form as

$$\hat{\bar{\boldsymbol{b}}}_{i_1}^{i_2} = \left(\bar{\mathcal{A}}^T(P_{i_1}, P_{i_2})\bar{\mathcal{A}}(P_{i_1}, P_{i_2})\right)^{-1} \bar{\mathcal{A}}^T(P_{i_1}, P_{i_2})\, \boldsymbol{l}(P_{i_1}, P_{i_2}).$$

By a close inspection of (2.12), one can see that $\bar{\mathcal{A}}(P_{i_1}, P_{i_2})$ is full rank, as soon as there are two observations for which the matrix $R_{M_{i_2}}^{M_{i_1}}$ assumes different values; this condition is easily verified in practice[1]. The test is assumed to be positive if $f_r(\hat{\bar{\boldsymbol{b}}}_{i_1}^{i_2}) < \xi_t$, negative otherwise.

Revolute joint - non-linear test

In case that the previous test is positive, the algorithm checks conditions defined by the non linear system in (2.15), to determine if M_{i_1} and M_{i_2} are connected through the revolute joint q_k. Differently from the previous case, the output of the non linear system is a vector of angles. To account for periodicity effects, the residuals are defined as

$$\tilde{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{o}_1, \boldsymbol{o}_2) = \operatorname{mod}\left(\boldsymbol{o}\left(P_{i_1}, P_{i_2}\right) - f_o(\boldsymbol{q}_k, \boldsymbol{o}_1, \boldsymbol{o}_2) + \pi\right) - \pi,$$

where we recall that $\operatorname{mod}(\cdot)$ is the 2π module operator. With a slight abuse of notation, vectors $\boldsymbol{o}_1$ and $\boldsymbol{o}_2 \in \mathbb{R}^3$ are directly inputted to $f_o(\cdot)$, instead of computing the relative rotation matrices with (2.14). Then, the algorithm computes $\hat{\boldsymbol{o}}_{L_{j_1}}^{M_{i_1}}$ $\hat{\boldsymbol{o}}_{M_{i_2}}^{L_{j_2}}$ solving the following non linear least squares problem,

$$\left(\hat{\boldsymbol{o}}_{L_{j_1}}^{M_{i_1}}, \hat{\boldsymbol{o}}_{M_{i_2}}^{L_{j_2}}\right) = \arg\min_{(\boldsymbol{o}_1, \boldsymbol{o}_2)} \left(\tilde{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{o}_1, \boldsymbol{o}_2)\right)^T \tilde{\boldsymbol{r}}_{i_1}^{i_2}(\boldsymbol{o}_1, \boldsymbol{o}_2). \tag{2.19}$$

The test is assumed to be positive if $f_r\left(\tilde{\boldsymbol{r}}_{i_1}^{i_2}(\hat{\boldsymbol{o}}_{L_{j_1}}^{M_{i_1}}, \hat{\boldsymbol{o}}_{M_{i_2}}^{L_{j_2}})\right)$ is lower than ξ_o, negative otherwise.

[1]If not verified, we might compute $\hat{\bar{\boldsymbol{b}}}_{i_1}^{i_2}$ resorting to the pseudo inverse of $\bar{\mathcal{A}}^T(P_{i_1}, P_{i_2})\bar{\mathcal{A}}(P_{i_1}, P_{i_2})$.

2.5 A polynomial-based approach for forward kinematics identification

In this section, we describe the strategy we propose to identify the forward kinematics. Our solution is a grey-box solution based on GPR, in particular, on the design of a suitable polynomial kernel. Minimal prior knowledge about the model is required, since to derive the GPR model we assume to know only the kinematic structure, more specifically, the sequence $\mathcal{S}_Q$; no knowledge about the kinematic parameters is required. The section is organized as follows. First, we formally formulate the forward kinematics identification problem. Second, we state a proposition that shows how the forward kinematics can be described by a polynomial function defined over a suitable input space derived from the standard inputs, that are joints positions. Third, driven by this proposition, we derive a proper GPR model. Finally, we conclude the section proposing a simple kinematic controller based on the models derived.

2.5.1 Forward kinematics and polynomial functions

The forward kinematics identification problem consists in identifying the map that relates the position of the joints with the pose of the last link with respect to the camera RF. In the following, to keep the notation compact, we denote with the subscript M the marker attached to the last link of the robot. Accordingly, the output of the forward kinematics is given by the vector $\boldsymbol{p}_M^c = [(\boldsymbol{l}_M^c)^T, (\boldsymbol{o}_M^c)^T]^T$, or, equivalently, by the pair $(\boldsymbol{l}_{\boldsymbol{M}}^{\boldsymbol{c}}, R_M^c)$, if we express orientations using rotation matrices. For future use, we introduce also the function relating the yaw pitch and roll angles collected in $\boldsymbol{o}_M^c$ with the elements of R_M^c,

$$\begin{aligned}
\alpha_M^c &= \tan^{-1}\left(R^c_{M_{(2,1)}}/R^c_{M_{(1,1)}}\right), \\
\beta_M^c &= \tan^{-1}\left(-R^c_{M_{(3,1)}}/\sqrt{(R^c_{M_{(3,2)}})^2+(R^c_{M_{(3,3)}})^2}\right), \\
\gamma_M^c &= \tan^{-1}\left(R^c_{M_{(3,2)}}/R^c_{M_{(3,3)}}\right),
\end{aligned} \tag{2.20}$$

where $R^c_{M_{(i,j)}}$ is the element of R_M^c in the i-th row i and j-th column (see Siciliano et al. (2009) for the details).

We show how the forward kinematics can be properly described by a polynomial function over a input vector $\boldsymbol{x}$ which depends on $\boldsymbol{q}$ and is formally described as follows. Let N_r and N_p be, respectively, the number of revolute and prismatic joints, and let $\mathcal{I}_r = \{r_1, \dots, r_{N_r}\}$ and $\mathcal{I}_p = \{p_1, \dots, p_{N_p}\}$ be the indices of the revolute and prismatic joints. Then, $\boldsymbol{x}$ is defined as

$$\boldsymbol{x} = \left[\boldsymbol{q}_c, \boldsymbol{q}_s, \boldsymbol{q}_p\right], \tag{2.21}$$

where,

$$\begin{aligned}
\boldsymbol{q}_c &= \Big[\cos(q_{r_1}), \dots, \cos(q_{r_{N_r}})\Big] \in \mathbb{R}^{N_r}, \\
\boldsymbol{q}_s &= \Big[\sin(q_{r_1}), \dots, \sin(q_{r_{N_r}})\Big] \in \mathbb{R}^{N_r}, \\
\boldsymbol{q}_p &= \Big[q_{p_1}, \dots, q_{p_{N_p}}\Big] \in \mathbb{R}^{N_p}.
\end{aligned}$$

Moreover, in the following, we denote by q_{c_b}, q_{s_b} and q_{p_b} the b-th component of $\boldsymbol{q}_c$, $\boldsymbol{q}_s$ and $\boldsymbol{q}_p$, respectively, and by X the set collecting all the $\boldsymbol{x}(t_1), \dots, \boldsymbol{x}(t_T)$. We have the following result.

Proposition 2.5.1. *Consider a manipulator with N_r revolute joints and N_p prismatic joints, and let $\mathcal{I}_r$ and $\mathcal{I}_p$ be, respectively, the indices of the revolute and prismatic joints. Then, the elements of vector $\boldsymbol{l}^c_M$ and matrix R^c_M are polynomial functions in the input vector $\boldsymbol{x}$, defined in (2.21). More specifically,*

(i) the elements of R^c_M are polynomial functions in the elements of $[\boldsymbol{q}_c, \boldsymbol{q}_s]$, with maximum degree N_r and where each monomial satisfies the conditions

$$deg(q_{c_b}) + deg(q_{s_b}) \leq 1 \qquad b = 1, \dots, N_r;$$

(ii) the elements of $\boldsymbol{l}^c_M$ are polynomial functions in the elements of $\boldsymbol{x}$ with maximum degree $N_r + N_p$, and where each monomial satisfies the conditions

$$\begin{aligned}
&deg(q_{p_b}) \leq 1 \qquad b = 1, \dots, N_p, \\
&deg(q_{c_b}) + deg(q_{s_b}) \leq 1 \qquad b = 1, \dots, N_r.
\end{aligned}$$

The proof is reported in the appendix.

2.5.2 GPR based on polynomial kernel

To achieve robustness with respect to noisy observations, we rely on Bayesian techniques, in particular on GPR, that we have reviewed in Section 1.3. For completeness, we briefly review here the main concepts. In the GPR framework the outputs are modeled as a Gaussian process. Let $\boldsymbol{x}(t)$ and $y(t)$ be, respectively, the input and noisy outputs of function f at time t. Then we have

$$\boldsymbol{y} = \begin{bmatrix} y(t_1) \\ \vdots \\ y(t_T) \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}(t_1)) \\ \vdots \\ f(\boldsymbol{x}(t_T)) \end{bmatrix} + \begin{bmatrix} e(t_1) \\ \vdots \\ e(t_T) \end{bmatrix} = f(X) + \boldsymbol{e},$$

where e is Gaussian i.i.d. noise with zero mean and standard deviation σ, while $f(X) \sim N(0, K(X,X))$, with $K(X,X)$ denoting the kernel matrix, whose elements are defined thorough the kernel function $k(\cdot,\cdot)$; in particular the element of K at row i and column j is equal to $k(\boldsymbol{x}(t_i), \boldsymbol{x}(t_j))$. Given a set of training samples $(X, \boldsymbol{y})$, the maximum a posteriori probability (MAP) estimator of f is given in closed form. Let $\boldsymbol{x}(t_*)$ be the value of the input location at a general time instant t_*. Then, the MAP estimate of $f(\boldsymbol{x}(t_*))$ is given by

$$\hat{f}(\boldsymbol{q}(t_*)) = \sum_{h=1}^{N_{TR}} \alpha_i k(\boldsymbol{q}(t_*), \boldsymbol{q}(t_h)),$$

where α_i is the i-th element of the vector $\boldsymbol{\alpha}$, defined as

$$\boldsymbol{\alpha} = \left(K(X,X) + \sigma^2 I\right)^{-1} \boldsymbol{y}.$$

From a functional analysis point of view, the previous Proposition 2.5.1 states that the outputs of the forward kinematics are polynomial functions in the augmented space $\boldsymbol{x}$. Through the GPR framework, adopting suitable kernel functions, we can define a regression problem in the Reproducing Kernel Hilbert Space (RKHS) identified by the the previous proposition Schölkopf and Smola (2001). In particular, we recall two notions that will be used in the next part of this section. The first is the definition of linear kernel, which is given by the following expression,

$$k(\boldsymbol{x}(t_i), \boldsymbol{x}(t_j)) = \boldsymbol{\phi}^T(\boldsymbol{x}(t_i)) \Sigma \boldsymbol{\phi}(\boldsymbol{x}(t_j)),$$

where $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \dots, \phi_p(\boldsymbol{x})] \in \mathbb{R}^p$ is a function mapping $\boldsymbol{x}$ in a given feature space, and $\Sigma > 0 \in \mathbb{R}^{p \times p}$ is a diagonal matrix, whose diagonal elements are the kernel hyperparameters. The RKHS associated to this kernel is the one generated by the functions in $\boldsymbol{\phi}$.

Secondly, we recall that the point-wise product of two kernels is still a kernel. In particular, consider two linear kernels, and denote by $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ the corresponding basis functions sets. Then, the basis functions of the RKHS associated to the the product of this two kernels are obtained by considering all the possible products between an element in $\boldsymbol{\phi}_1$ and an element in $\boldsymbol{\phi}_2$, Schölkopf and Smola (2001) (chapter 13.1).

2.5.3 Proposed approach for forward kinematics identification

In our work we model each element of $\boldsymbol{l}_M^c$ and R_M^c by a distinct Gaussian process. For each model we consider different values of the hyperparameters, as well as different types of kernels.

Driven by Proposition 2.5.1, and the point-wise product property discussed before, we define $k_R(\cdot,\cdot)$, that is, the kernel function modeling the elements of R^c_M, as the product of several elementary building blocks. In particular, each building block is a linear kernel in the b-th component of $\boldsymbol{q}_c$ and $\boldsymbol{q}_s$, with $b = 1, \dots, N_r$, accounting for the contributions of $\cos(q_{r_b})$ and $\sin(q_{r_b})$. Formally k_R is defined as,

$$k_R(\boldsymbol{x}(t_i), \boldsymbol{x}(t_j)) = \prod_{b=1}^{N_r} \boldsymbol{q}^T_{cs_b}(t_i) \Sigma_{R_b} \boldsymbol{q}_{cs_b}(t_j), \tag{2.22}$$

where $\boldsymbol{q}_{cs_b} = [1, q_{c_b}, q_{s_b}]$, and $\Sigma_{R_b} \in \mathbb{R}^{3\times 3}$, $b = 1, \dots, N_r$, are diagonal matrices, whose elements are the hyperparameters of the kernel[2].

Next, we define $k_l(\cdot,\cdot)$, that is, the kernel function modeling the elements of $\boldsymbol{l}^c_M$. Proposition 2.5.1 suggests that, in addition to $\boldsymbol{q}_c$, $\boldsymbol{q}_s$, we need to account also for the contributions linear w.r.t. $\boldsymbol{q}_p$. To do that, we multiply $k_R(\cdot,\cdot)$ by an additional kernel, obtaining,

$$k_l\left(\boldsymbol{x}(t_i), \boldsymbol{x}(t_j)\right) = k_R\left(\boldsymbol{x}(t_i), \boldsymbol{x}(t_j)\right) \bar{\boldsymbol{q}}^T_p(t_i) \Sigma_l \bar{\boldsymbol{q}}_p(t_j), \tag{2.23}$$

where $\bar{\boldsymbol{q}}_p$ is the vector obtained concatenating 1 with $\boldsymbol{q}_p$; the matrix $\Sigma_l \in \mathbb{R}^{(N_p+1)\times(N_p+1)}$ is a diagonal matrix, whose diagonal elements are additional hyperparameters.

We conclude the description of the proposed learning strategy with the following observation. As discussed before, R^c_M is an redundant description of the relative orientation $\boldsymbol{o}^c_M$. To limit the number of models to be learned, in our work we learn only the models of $R^c_{M_{(1,1)}}$, $R^c_{M_{(2,1)}}$, $R^c_{M_{(3,1)}}$, $R^c_{M_{(3,2)}}$ and $R^c_{M_{(3,3)}}$, and we obtain $\boldsymbol{o}^c_M$ applying (2.20).

2.5.4 Kinematic controller based on the GPR model

In this subsection, we introduce a naive kinematic controller based on the forward kinematics model previously described. It is worth stressing that the proposed controller implementation is meant only to test the accuracy of the model obtained, and not to developed a novel control strategy. For this reason we have not addressed several important issues that, instead, should be considered when proposing highly performing control strategies.

We assume that each joint motor is equipped with a low level controller able to follow a desired reference trajectory $\boldsymbol{q}_{des}$, and we denote by δ the sampling time. Typically, when considering trajectories with sufficiently small differences between the desired values at time t and $t + \delta$, the low level controllers are able to follow accurately the reference

[2]Notice that the presence of 1 in $\boldsymbol{q}_{cs_b}$ is needed to consider the monomials with degree zero and one in the RKHS defined by $k_R(\cdot,\cdot)$.

trajectory. Our strategy is based on the fact that the model derived in the previous section is differentiable with respect to $\boldsymbol{q}$. Then, given $\boldsymbol{p}_*$, a target pose of the last marker, we implement an open loop controller that computes a suitable $\boldsymbol{q}_{des}$ trajectory driving the robot from the initial pose $\boldsymbol{p}^c_M(0)$ to $\boldsymbol{p}_*$. The proposed controller is defined by the following expression,

$$\boldsymbol{q}_{des}(t+\delta_T) = \boldsymbol{q}_{des}(t) - \nu \nabla e_p(t), \tag{2.24}$$

where $e_p(t) = ||\boldsymbol{p}_* - \hat{\boldsymbol{p}}^c_M(t)||$, with $\hat{\boldsymbol{p}}^c_M(t)$ equals to the estimated pose at time t, ∇e_p is the gradient of e_p computed w.r.t. $\boldsymbol{q}$, and ν is a parameter that allows to tune the velocity with which the target pose is reached. It is worth stressing that the objective function minimized by the controller is not linear w.r.t. $\boldsymbol{q}$. As a consequence, the performance of this simple gradient-based controller could be limited by the presence of local minima, and not only by inaccuracies of the GPR model.

2.6 Experimental results

We describe the experiments that we performed to test the effectiveness of our solutions. We considered tests both in simulated and real environments. The simulated environments are implemented in MATLAB. Given the robot DH parameters, and the markers poses expressed w.r.t. the links RF, the observations are computed evaluating iteratively (2.2) and (2.3). In the simulated environments we do not consider the presence of noise.
Experiments in a real setup were performed with a UR10 robot, which is a collaborative manipulator with six revolute joint ($n = 7$). As far as the camera is concerned, we used a Microsoft Kinect One (Kinect 2) RGB-D sensor, positioned in front of the robot, at a distance approximately equal to $2[m]$. Instead, as regards the fiducial markers, we considered the AprilTags 36h11 family, proposed in Olson (2011). The frames collected were processed using IAI Kinect2 ROS package that is a collection of tools and libraries for a ROS Interface to the Kinect One, and the ROS wrapper of the AprilTag 3 visual detector Wang and Olson (2016); Brommer, Malyuta, Hentzen, and Brockers (2018); Malyuta, Brommer, Hentzen, Stastny, Siegwart, and Brockers (2019). The interface with the robot and the camera are implemented in ROS Quigley, Conley, Gerkey, Faust, Foote, Leibs, Wheeler, and Ng (2009). In particular, we used the URModernDriver Andersen (2015), which allows collecting the joint positions at $125[Hz]$. The system collects a new sample after finishing processing the previous frame. The time required to process a frame is approximately equal to $1[sec]$. In the real setup we collected two dataset. The first dataset, hereafter denoted by $\mathcal{D}_{tr}$, consists in the collection of 2103 samples, acquired

while the robot follows a trajectory defined by 25 way-points randomly distributed in the available region of the robot workspace. The second dataset instead, denoted by $\mathcal{D}_{test}$, is the collection of 2121 samples, collected requiring the robot to reach the same way-points, but with a different order; by doing so, the trajectories followed by the robot in the two dataset are different.

2.6.1 Empirical results for general trajectories

In this experiment we investigate the effectiveness of the proposed algorithm for general input trajectories, performing a Monte Carlo study in the simulated environment, composed of 64 experiments. In each experiment we simulated a different robot (with $n = 6$), obtaining a dataset of the type defined in (2.1), with $T = 50$. Among the different simulations we let vary several parameters, like the joint type order $\mathcal{S}_Q$, the DH parameters, the relative pose between the camera and the robot, and the markers attachment, namely, $R_{M_i}^{Lj}$ and $\boldsymbol{l}_{M_i}^{Lj}$. As far as the input trajectories are concerned, each joint trajectory is given by the sum of 10 sinusoids, with amplitudes and frequencies randomly selected in each simulation. For each dataset we considered all the possible triplets, i.e., all the pairs of markers and joint signal, for a total number of 4800 triplets, and we checked the feasibility of the conditions defined by Proposition 2.3.1, 2.3.2 and 2.3.3. We assigned class 1 in case that conditions are satisfied, an class 2 otherwise. The triplets are labeled accordingly, namely, we assigned the triplet (M_1, M_2, q_k) to class 1 when the elements of the triplet are connected and q_k is a prismatic (resp. revolute) joint, when evaluating conditions defined by Proposition 2.3.1 (resp. Proposition 2.3.2 and 2.3.3). Results are reported in Figure 2.4 in the form of confusion matrix and shows that the probability of selecting non *fully informative* input trajectories is close to zero, since in each simulations all the triplets have been labeled correctly. Moreover, results evidence that equations defined by Proposition 2.3.2 and 2.3.3 identify only a necessary condition when considered alone, since several false positives occurred. However, considering them together, as done in Proposition 2.3.6, the number of false positives goes to zero, allowing always a correct classification.

2.6.2 UR10: kinematic structure classification

In this experiment we test the effectiveness of our kinematic structure classification algorithm with data acquired in the real setup. As done in the previous experiment, we test all the possible triples of markers with the observations in $\mathcal{D}_{tr}$. As concerns the values of the thresholds defined in Section 2.4.3, we set the following values, $\xi_t = 0.02$ and $\xi_o = 0.1$. In Figure 2.5 we reported the confusion matrix obtained. Results confirm

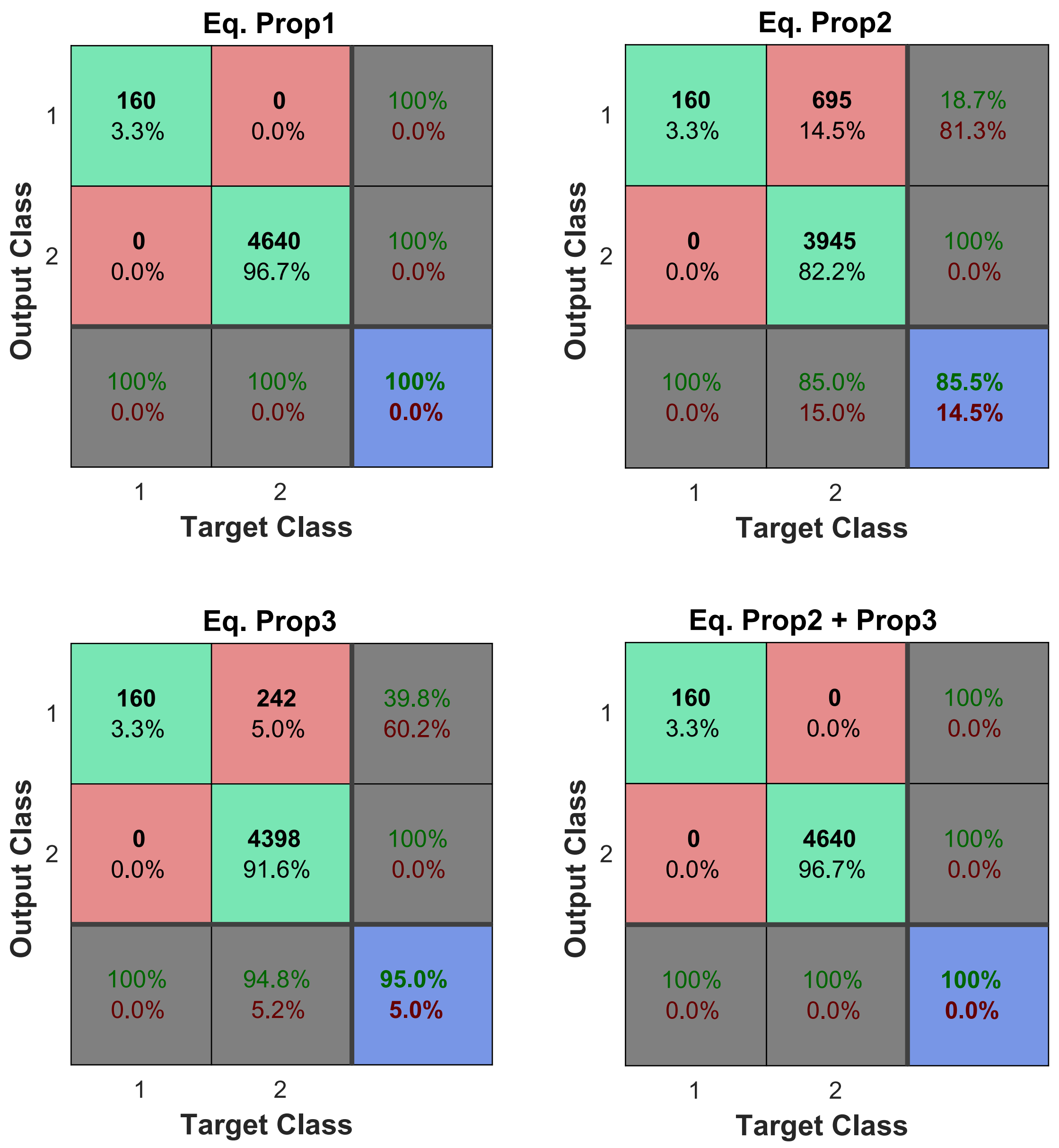

Figure 2.4: Confusion matrices of the Monte Carlo Simulations described in Section 2.6.1. When considering Eq. Prop. 2.3.1, a sample is assigned to Class 1 if its markers and joint identify a prismatic transformation, while in the cases of Eq. Prop. 2.3.2 and 2.3.3 the elements in Class 1 identify revolute transformations.

Table 2.1: RMSEs of the estimation errors obtained in $\mathcal{D}_{test}$. The RMSEs of x, y, and z are expressed in meters, while the RMSEs of γ, $beta$, and α are in radiant.

x	y	z	γ	β	α
0.0052	0.0051	0.0069	0.0169	0.0265	0.0358

the effectiveness of the adaptation proposed to deal with the presence of noise, given that all the triplets are classified correctly.

To check how the number of observations influences the performance of the proposed approach, we analyzed the evolution of the f_r, obtained solving the optimization problems in (2.18) and (2.19), as function of the training samples. In Figure 2.6 we plotted the evolution of the number of misclassified triplets, together with f_{r_1} and f_{r_2}, where f_{r_1} (resp. f_{r_2}) corresponds to the mean of the the f_r values associated to triplets that are connected (resp. not connected). In transparency we plotted the intervals between the minimum and the maximum value of f_{r_1} and f_{r_2}. Observe that in the initial phases the values of f_{r_1} and f_{r_2} are close, and, consequently, several triplets are not classified correctly. We argue that, due to the small variability of the observations, the residuals due to false assumptions are not significantly different from the residuals due to noise. The values of f_{r_2} increase significantly with the number of samples, i.e., with the variability of the observations, allowing the correct classification of all the triplets. In particular, in the test performed, the gap between the values of f_{r_1} and f_{r_2} become sufficiently different after 1300 samples. However, it is worth mentioning that the number of samples required is strictly connected to several aspects, in particular, the variability of the joints trajectories and the measurement noise.

2.6.3 UR10: forward kinematics identification

In this experiment we test our forward kinematics identification algorithm with data acquired in the real setup. We trained the proposed estimator using the observations in $\mathcal{D}_{tr}$. In order to further stress the generalization properties, we used just 100 samples, randomly selected among the 2103 samples in $\mathcal{D}_{tr}$. The training consists in computing (2.5.2), after optimizing the hyperparameters by Marginal Likelihood maximization (see Section 1.3). In Table 2.1 we reported the out of sample performance, measured in terms of Root Mean Squared Error (RMSE) in $\mathcal{D}_{test}$. Results prove the effectiveness of the proposed solution. The RMSEs of the relative translations are in the order of millimeters, while, the RMSEs of the relative orientation are in the order of 0.01 radiants. These values are of the same order of magnitude of the measurement noise present in our setup.

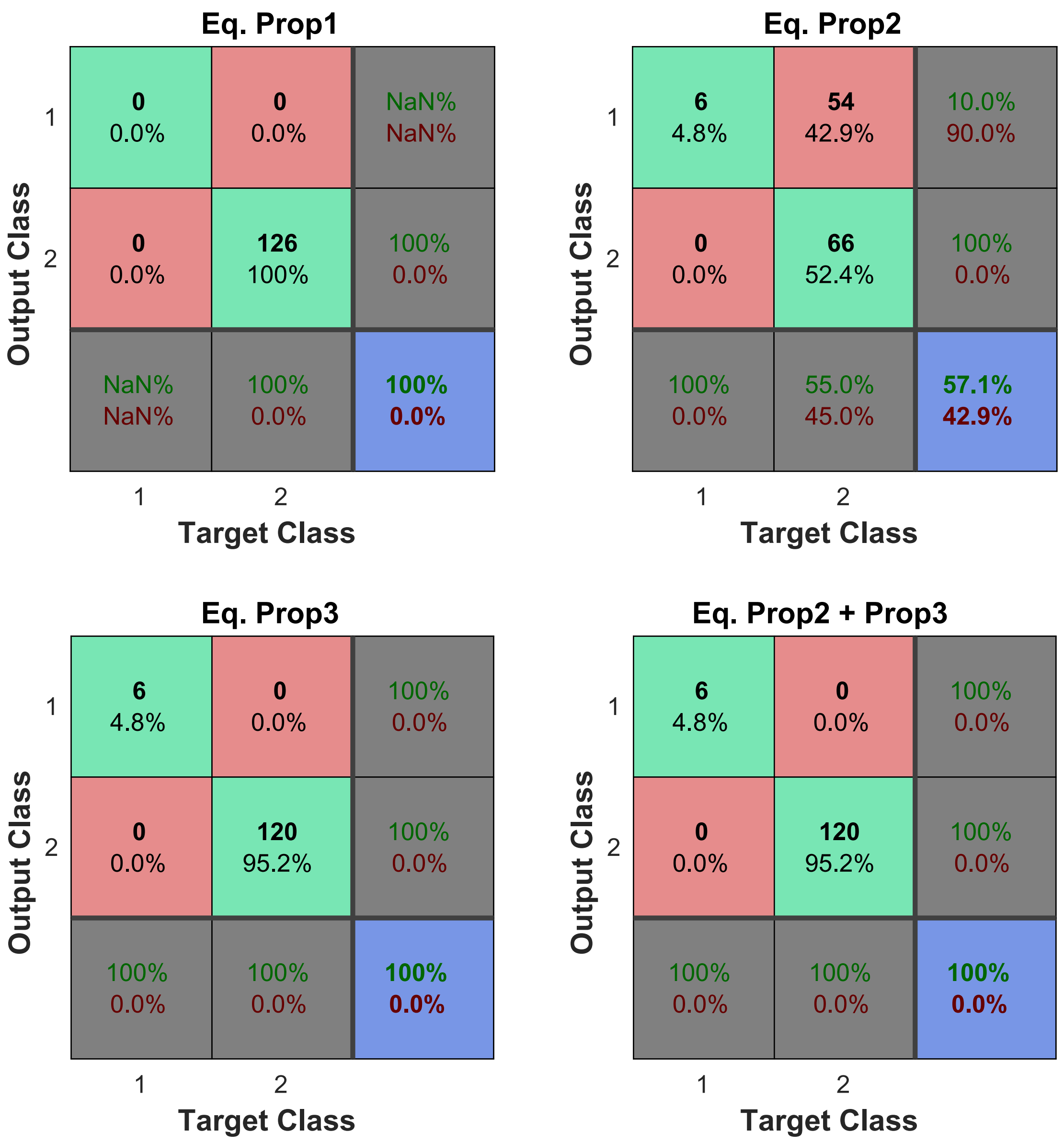

Figure 2.5: Confusion matrices obtained processing all the triplets in $\mathcal{D}_{tr}$. When considering Eq. Prop. 2.3.1, a sample is assigned to Class 1 if its markers and joint identify a prismatic transformation, while in the cases of Eq. Prop. 2.3.2 and 2.3.3 the elements in Class 1 identify revolute transformations.

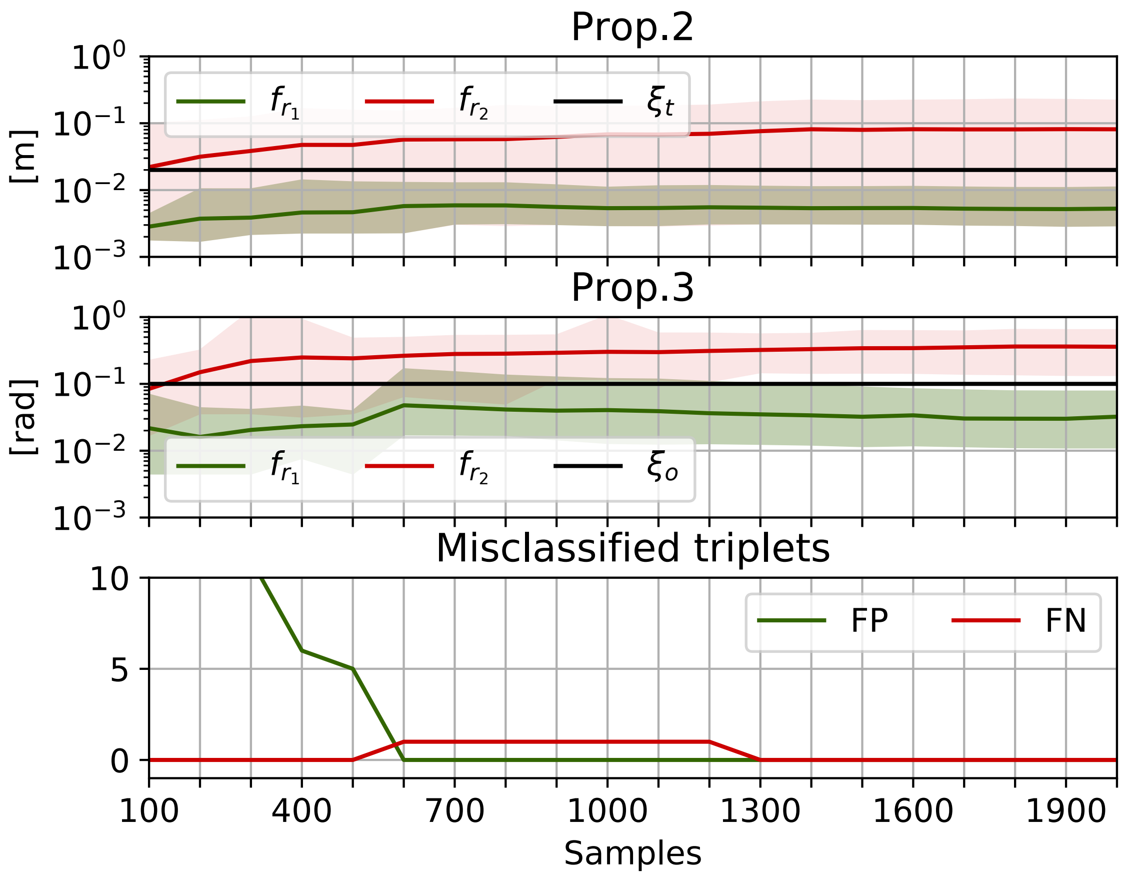

Figure 2.6: Evolution of the f_r mean as function of the samples. The green and red lines represent the mean of the triplets belonging to class 1 and class 2. In transparency we pointed out the intervals between the minimum and the maximum values of f_{r_1} and f_{r_2}. Results are plotted in logarithmic scale. The bottom plot represents the evolution of the misclassified cases; FP is the number of *false positives*, while FN is the number of *false negatives*.

Table 2.2: RMSEs of the positioning errors obtained deriving the controller with $\mathcal{D}_{tr}$ and $\mathcal{D}_{\bar{tr}}$. The RMSEs of x, y, and z are expressed in meters, while the RMSEs of γ, $beta$, and α are in radiant.

Dataset	x	y	z	γ	β	α
$\mathcal{D}_{tr}$	0.0065	0.0111	0.0228	0.1853	0.0853	0.1167
$\mathcal{D}_{\bar{tr}}$	0.0017	0.0016	0.0020	0.0988	0.0269	0.0419

2.6.4 UR10: control performance

In this set of experiments we test the kinematic controller described in (2.24), based on the model learned in the previous experiment with our polynomial-based approach. In particular, to derive the estimator, we used all the observations in $\mathcal{D}_{tr}$. Given a fixed initial configuration of the robot, we computed offline the reference trajectories to be inputted to the low level controller to reach 5 target poses randomly distributed in the robot workspace. After applying the computed control actions, we collected the final poses measured by the camera; the video of the experiment is publicly available[3]. In Table 2.2 we reported the RMSEs of the positioning errors. Results prove that the control actions computed with the naive controller drive the robot close to the target poses. As concerns the relative translations, errors are in the order of centimeters, while orientation errors are in the order of 0.1 radiants.

To evaluate the possibility of improving the accuracy of the controller in an online framework, we performed a second experiment, in which we derive the forward kinematics estimator using $\bar{\mathcal{D}}_{tr}$, the dataset obtained adding to $\mathcal{D}_{tr}$ the observations of the poses reached with the control actions computed with the previous controller. Results show a consistent improvement, likely due to the higher accuracy of the estimator. This fact highlights the importance of defining strategies to sufficiently exciting strategies.

Finally, we point out that the RMSEs obtained do not depend only on the accuracy of forward kinematics model. Significant contributions to the errors are also due to intrinsic limitations of the control strategy implemented, in particular, the presence of local minima in the error function e_p. These errors might be removed implementing more complex control strategies, that, however, are out of the scope of this chapter.

2.7 Conclusions

In this chapter we have introduced a autonomous algorithm for kinematics model identification. In the framework presented, we assume that a distinct marker is attached

[3]https://youtu.be/PgjvfxMArtQ

to each link of the robot, as well as that joint variables are measurable; no prior about the robot geometry is assumed. Starting from a time series of such observations, the proposed algorithm firstly identifies the robot kinematic structure. Secondly, based on the kinematic structure identified, it derives an estimator of the forward kinematics based of GPR, in particular on polynomial kernels. We proved the effectiveness of our approach in several simulated environments, as well as in a real setup, with experiments performed with a UR10 robot.

3
Polynomial-based inverse dynamics identification

3.1 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 Khosla and Kanade (1988) and on more complex Model Predictive Control approaches Poignet and Gautier (2000), or to provide robots with proprioceptive sensing capabilities Haddadin et al. (2017), Siciliano and Villani (2000).

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 design time and costs.

Several data-driven strategies to learn inverse dynamics have been developed. In Vijayakumar and Schaal (2000), the authors proposed a locally weighted projection of different linear models. A significant number of approaches rely on neural networks; for instance, the authors in Polydoros, Nalpantidis, and Krüger (2015) resort to the use of a recurrent neural network, while in Rueckert, Nakatenus, Tosatto, and Peters (2017) an LSTM network has been proposed. Another wide class of solutions is based on Gaussian Process Regression (GPR), Nguyen-Tuong, Peters, Seeger, and Schölkopf (2008), Romeres, Zorzi, Camoriano, and Chiuso (2016), Camoriano, Traversaro, Rosasco, Metta, and Nori (2016) and Nguyen-Tuong, Seeger, and Peters (2009). Differently, from neural networks, GPR provides also a bound on the uncertainty of the estimates; this additional information can be exploited in different ways, see for instance in Reinforcement Learning the PILCO algorithm Deisenroth and Rasmussen (2011).

Although data-driven modeling techniques have been applied successfully in several control applications, see for example Deisenroth and Rasmussen (2011), Levine and Abbeel (2014), Hewing, Liniger, and Zeilinger (2018), 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 the 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 (DOF). 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 chapter, 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 this work 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 inputs, that are, positions, velocities, and accelerations, of the generalized coordinates, into an augmented space, where the inverse dynamics map derived with the Lagrangian equations is a polynomial function. Inspired by this property, we propose a model based on the *Multiplicative Polynomial Kernel* (MPK), recently introduced in Dalla Libera et al. (2019c), which is a re-parameterization of the standard polynomial kernel. As shown in Dalla Libera et al. (2019c), and also in Chapter 6, compared to the standard polynomial kernel, MPK parametrization allows for greater flexibility in neglecting eventual unnecessary basis functions of the corresponding Reproducing Kernel Hilbert Space (RKHS), leading to higher generalization performance.

The idea of mapping the standard inputs in a space where the inverse dynamics is a polynomial function has been explored also in Ge and Hang (1998). In Ge and Hang (1998), the authors have introduced the same augmented space we consider in this work but have modeled only the elements of the inertia matrix and the potential energy as polynomials in this augmented space. However, instead of exploiting the encoding properties of polynomial kernels, they have designed a structural network inspired by the Lagrangian equations, where each element of the inertia matrix and the potential energy are a linear combination of monomials belonging to a particular class of polynomials. As consequence, compared to our approach, in the model proposed in Ge and Hang (1998) the number of parameters to be identified grows quickly w.r.t. the robot DOF, i.e., w.r.t. the number of possible monomials, increasing the risk of overfitting.

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

The chapter is organized as follows. In Section 3.2, we provide an overview of the main strategies based on GPR adopted in inverse dynamics learning. In Section 3.3 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 3.4, 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.

3.2 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 $\boldsymbol{q} = [q_1, \dots, q_n]^T \in \mathbb{R}^n$ be the vector collecting the generalized coordinates associated to the joints; accordingly, let $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ be the velocity and acceleration vectors, respectively. The inverse dynamics problem consists in estimating the function mapping the triple $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ into the vector of generalized torques, denoted by $\boldsymbol{\tau} \in \mathbb{R}^n$. The estimation is typically performed starting from a set of input-output observations, which is composed by the set of input locations $X = \{\boldsymbol{x}(t_1), \dots, \boldsymbol{x}(t_{N_{TR}})\}$, where $\boldsymbol{x}(t) = [\boldsymbol{q}(t)\, \dot{\boldsymbol{q}}(t)\, \ddot{\boldsymbol{q}}(t)]$, and the set of noisy outputs Y associated to $\{\boldsymbol{\tau}(t), t = t_1, \dots, t_{N_{TR}}\}$, 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.

3.2.1 Rigid body dynamics estimators

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

$$\boldsymbol{\tau} = B(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})\,, \tag{3.1}$$

where $B(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ and $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n \times n}$ are, respectively, the inertia matrix and the Coriolis matrix, and $\boldsymbol{g}(\boldsymbol{q})$ is the vector accounting for the gravitational contributions, see Siciliano et al. (2009). The previous equation depends on two sets of parameters, the kinematic and dynamics parameters. The first set is composed by geometric quantities (i.e., lengths, angles) that, together with $\boldsymbol{q}$, 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 (3.1) is linear w.r.t. the dynamics parameters, see Siciliano et al. (2009). Specifically, denoting by $\boldsymbol{w}^d$ the vector collecting all the dynamics parameters, (3.1) can be rewritten as

$$\boldsymbol{\tau} = \Phi^d(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\,\boldsymbol{w}^d = \Phi^d(\boldsymbol{x})\,\boldsymbol{w}^d = \begin{bmatrix}\boldsymbol{\phi}_1^d(\boldsymbol{x}) & \dots & \boldsymbol{\phi}_n^d(\boldsymbol{x})\end{bmatrix}^T \boldsymbol{w}^d, \tag{3.2}$$

for a suitable matrix $\Phi^d \in \mathbb{R}^{n \times N_{par}}$, which depends only on the kinematic parameters. Then, assuming the kinematic parameters to be known, the inverse dynamics problem boils down to the computation of $\hat{\boldsymbol{w}}^d$, an estimate of $\boldsymbol{w}^d$.

In several solutions, $\hat{\boldsymbol{w}}^d$ is computed relying on Fisherian techniques, see for example Sousa and Cortesão (2014). When the model is accurate, and the signal to noise ratio is sufficiently high, these estimators achieve accurate estimates, together with good generalization properties. However, besides the presence of noise, several aspects could limit the performance of such approaches. Indeed, it is worth stressing that errors in the knowledge of the kinematic parameters introduce model bias. Moreover, there are situations where it is hard to derive (3.2), or where the rigid body assumption is a too rough approximation of the real robot behaviors.

3.2.2 Gaussian Process Regression for robot inverse dynamics

To overcome the limitations characterizing estimators based on the RBD assumption, several Bayesian approaches have been proposed in the last decade. Most techniques are based on GPR, see Section 1.3. Typically, in GPR approaches, each joint is treated individually and modeled as a single Gaussian process (GP). More precisely, referring to the notions introduced in Section 1.3, when considering the *i-th* joint, it is assumed that $\boldsymbol{y}_i$, the vector collecting the measure associated $\tau_i(t)$ with $t = t_1, \dots, t_{N_{TR}}$, is generated by the following probabilistic model

$$\boldsymbol{y}_i = \begin{bmatrix} f_i(\boldsymbol{x}(t_1)) \\ \vdots \\ f_i(\boldsymbol{x}(t_{N_{TR}})) \end{bmatrix} + \begin{bmatrix} e_i(t_1) \\ \vdots \\ e_i(t_{N_{TR}}) \end{bmatrix} = \boldsymbol{f}_i(X) + \boldsymbol{e}_i, \tag{3.3}$$

where $\boldsymbol{e}_i$ is i.i.d. Gaussian noise with standard deviation σ_{n_i}, and f_i is an unknown function defined as a GP, namely, $\boldsymbol{f}_i(X) \sim N(\boldsymbol{m}_i(X), K_i(X,X))$, being $\boldsymbol{m}_i$ 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 $k_i(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j))$.

We recall that in this probabilistic framework the maximum a posteriori probability (MAP) estimator of f_i is given in close form. Let $\boldsymbol{x}(t_*)$ be a general input location. Then, as proved in Rasmussen and Williams (2006) (see Chapter 2), the MAP estimate of $f_i(\boldsymbol{x}(t_*))$ is given by

$$\hat{f}_i(\boldsymbol{x}(t_*)) = \sum_{h=1}^{N_{TR}} \alpha_i k_i(\boldsymbol{x}(t_*), \boldsymbol{x}(t_h)), \tag{3.4}$$

where α_i is the i-th element of the vector $\boldsymbol{\alpha}$, defined as

$$\boldsymbol{\alpha} = \left(K(X,X) + \sigma_{n_i}^2 I\right)^{-1} \boldsymbol{y}_i. \tag{3.5}$$

Before providing a brief overview about GPR based solutions, we comment about the computational cost of (3.4) and (3.5). Due to the matrix inversion in (3.5), the number of operations required to derive the f_i estimator is proportional to the cube of N_{TR}. However, observe that $\boldsymbol{\alpha}$ depends only on the training data, and then its computation can be performed offline. As consequence, once $\boldsymbol{\alpha}$ is computed, the cost of evaluating (3.4), grows linearly w.r.t. N_{TR}, thus allowing the use of GPR estimators in real time applications, see for example Hewing et al. (2018) and Nguyen-Tuong et al. (2009).

Non-Parametric prior: RBF Kernel

When no prior knowledge about the model is available, the GPR prior can be defined in a data-driven way, relaying on the so called Non-Parametric prior (NP) kernels. An option consists in assuming that the distribution of the outputs is stationary w.r.t. the input locations, and then, considering $\boldsymbol{m}_i(\cdot) = 0$, and adopting a Radial Basis Function (RBF) as kernel. For sake of clarity we recall the RBF kernel definition previously provided in Section 1.3,

$$k_{RBF}(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j)) = \lambda_{RBF} e^{-\frac{1}{2}||\boldsymbol{x}(t_h)-\boldsymbol{x}(t_j)||^2_{\Sigma^{-1}_{RBF}}},$$

where λ_{RBF} and Σ_{RBF} are the kernel hyperparameters, which are typically tuned from data by Marginal Likelihood (ML) maximization.

It is well known that RBF kernels can approximate any continuous functions, thus providing a valid tool to obtain accurate estimates of τ_i directly from data. The RBF kernel has been successfully applied in several robotics applications, see for example Deisenroth and Rasmussen (2011). However, 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. Several strategies have been designed in order to limit the computational complexity and increase the generalization, see for example Nguyen-Tuong et al. (2009). 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.

Parametric prior

In case a RBD model is given, starting from (3.2), and modeling $\boldsymbol{w}^d$ as a Gaussian variable, with mean $\bar{\boldsymbol{w}}^d$ and covariance Σ_w, it is possible to derive a linear kernel that inherits all the positive aspects of the RBD estimators, but acts in a Bayesian framework. Consider the measures of the i-th joint torques, we have that

$$\boldsymbol{y}_i = f(X) + \boldsymbol{e} = \Phi_i^d(X)\,\boldsymbol{w}^d + \boldsymbol{e},$$

where $\Phi_i^d(X)$ is the matrix with rows equal to $\boldsymbol{\phi}_i^d(\boldsymbol{x}(t_j))$, $j = 1, \ldots, N_{TR}$. Assuming $\boldsymbol{w}^d \sim N\left(\boldsymbol{w}^d, \Sigma_{w^d}\right)$, we have that the prior mean and the prior covariance are defined by the following expressions,

$$\begin{aligned} \boldsymbol{m}_i(X) &= \Phi_i^d(X)\bar{\boldsymbol{w}}^d, \\ k_i(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j)) &= \left(\boldsymbol{\phi}_i^d(\boldsymbol{x}(t_h))\right)^T \Sigma_{w^d}\boldsymbol{\phi}_i^d(\boldsymbol{x}(t_j)), \end{aligned} \tag{3.6}$$

where $\bar{\boldsymbol{w}}^d$ and Σ_{w^d} are the kernel hyperparameters. When the above kernel is used alone, we obtain the so called Parametric Prior (PP) estimators.

Semi-Parametric prior

Instead, when the kernel in (3.6) is used together with a NP kernel, we obtain the so called Semi-Parametric Prior (SP) estimators. Adopting an RBF kernel as data-driven kernel, we have

$$k_i(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j)) = \left(\boldsymbol{\phi}_i^d(\boldsymbol{x}(t_h))\right)^T \Sigma_{w^d} \boldsymbol{\phi}_i^d(\boldsymbol{x}(t_j)) + k_{RBF}(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j)), \tag{3.7}$$

see for example Romeres et al. (2016), Camoriano et al. (2016) and Nguyen-Tuong and Peters (2010). The rationale behind the use of kernel in (3.7) is the following: the first term allows to exploit the prior knowledge coming from the RBD, providing generalization, while $k_{RBF}(\cdot,\cdot)$ improves estimate in a neighborhood of the training locations, compensating for model bias or un-modeled behaviors. We remark that estimators based on (3.6) and (3.7) are model-based estimators, since their kernel functions are derived starting from (3.2).

3.3 Proposed Approach: Geometrically Inspired Polynomial Kernel

In this section, we propose a novel kernel that allows estimating the inverse dynamics without requiring prior knowledge of the model, preserving the fact of having a good generalization and high accuracy. This section is organized as follows. Firstly, we state Proposition 3.3.1, which characterizes the inverse dynamics from the functional analysis point of view. Given the type of each joint, i.e., prismatic or revolute, Proposition 3.3.1 defines a transformation of the input $\boldsymbol{x}$ where the inverse dynamics is a polynomial function. Then, we briefly review the *Multiplicative Polinomial Kernel*, and finally, we define the proposed kernel function, named *Geometrically Inspired Polynomial kernel.*

3.3.1 Polynomial characterization of the rigid-body model

In the following, we restrict our study to manipulators where each joint is either revolute or prismatic. 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 $I_r = \left\{i_{r_1}, \dots, i_{r_{N_r}}\right\}$ and $I_p = \left\{i_{p_1}, \dots, i_{p_{N_p}}\right\}$ the sets containing, respectively, the revolute and prismatic joints indexes. We start our analysis by defining

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

In the following, we denote by q_{c_b} the element in $\boldsymbol{q}_c$ associated to joint i_{r_b}, i.e., $\cos(q_{i_{r_b}})$ (similar definitions hold for q_{s_b} and q_{p_b}). For later convenience we define also $\boldsymbol{q}_{cs} \in \mathbb{R}^{2N_r}$, the vector obtained concatenating $\boldsymbol{q}_c$ and $\boldsymbol{q}_s$. In addition, we denote by $\dot{\boldsymbol{q}}_v$ the vector collecting the elements of the set

$$\left\{\dot{q}_i \dot{q}_j,\ 1 \le i \le n,\ i \le j \le n\right\},$$

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

Finally, we introduce a compact notation to identify a particular set of inhomogeneous polynomial functions. Let $\boldsymbol{a}$ be the vector containing the m variables $a_1, \dots, a_m$. We denote by $\mathbb{P}_{[p]}\left(\boldsymbol{a}_{[p_1]}\right)$ the set of polynomial functions of degree not greater than p defined over the variables in $\boldsymbol{a}$, such that each variable a_i appears with degree not greater than p_1. Similar definitions hold in case the inputs set accounts for more input vectors.

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

$$\bar{\boldsymbol{x}} = \left[\boldsymbol{q}_c, \boldsymbol{q}_s, \boldsymbol{q}_p, \dot{\boldsymbol{q}}_v, \ddot{\boldsymbol{q}}\right]. \tag{3.8}$$

We have the following result.

Proposition 3.3.1. *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 (3.1) belongs to $\mathbb{P}_{(2n+1)}\left(\boldsymbol{q}_{c(2)}, \boldsymbol{q}_{s(2)}, \boldsymbol{q}_{p(2)}, \dot{\boldsymbol{q}}_{v(1)}, \ddot{\boldsymbol{q}}_{(1)}\right)$. Namely, each $\tau_i(\cdot)$ is a polynomial function in $\bar{\boldsymbol{x}}$, of degree not greater than $2n+1$, such that: (i) each element of $\boldsymbol{q}_c$, $\boldsymbol{q}_s$ and $\boldsymbol{q}_p$ appear with degree not greater than 2, and (ii) each element of $\dot{\boldsymbol{q}}_v$ and $\ddot{\boldsymbol{q}}$ appear with degree not greater than 1. Moreover, for any monomial of the aforementioned polynomial, the sum of the q_{c_b} and q_{s_b} degrees is equal or lower than two, namely, it holds*

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

The proof is reported in the Appendix.

Remark 3.3.2. The result stated in Proposition 3.3.1 is related to the modeling property used in Ge and Hang (1998), though some important differences are present. Indeed, in Ge and Hang (1998), the authors have modeled with polynomial functions the potential energy and the elements of the inertia matrix, while Proposition 3.3.1 establishes that the whole inverse dynamics is a polynomial function in the augmented space. Moreover, Proposition 3.3.1 provides more strict constraints on the degrees of $\cos(q_i)$, $\sin(q_i)$, thus better characterizing the maximum degree with which each variable can appear in the different monomials, and, in turn, decreasing the number of possible monomials.

3.3.2 Multiplicative Polynomial Kernel

From a functional analysis point of view, Proposition 3.3.1 states that the inverse dynamics derived through the RBD belongs to the finite dimensional space of polynomial functions. A suitable set of basis functions for this space is given by the set of all the monomials in $\mathbb{P}_{(2n+1)}\left(\boldsymbol{q}_{c(2)}, \boldsymbol{q}_{s(2)}, \boldsymbol{q}_{p(2)}, \dot{\boldsymbol{q}}_{v(1)}, \ddot{\boldsymbol{q}}_{(1)}\right)$, heareafter denoted by vector $\boldsymbol{\phi}_{\mathbb{P}}\left(\boldsymbol{x}\right) \in \mathbb{R}^{N_{\mathbb{P}}}$, being $N_{\mathbb{P}}$ its cardinality. Unfortunately, $N_{\mathbb{P}}$ grows rapidly with the number of joints. To provide a couple of examples, when considering a SCARA robot, i.e., $n = 4$, we have that $N_{\mathbb{P}} = 1647$, while, for a standard six DOF robot, like the UR10, we have that $N_{\mathbb{P}} = 302615$. Clearly, when considering GPR based approaches, the computational and memory requirements induced by the dimension of $\boldsymbol{\phi}_{\mathbb{P}}\left(\boldsymbol{x}\right)$ prevent the possibility of adopting the linear kernel $k\left(\boldsymbol{x}\left(t_k\right), \boldsymbol{x}\left(t_j\right)\right) = \left(\boldsymbol{\phi}_{\mathbb{P}}\left(\boldsymbol{x}\left(t_j\right)\right)\right)^T \Sigma_w \boldsymbol{\phi}_{\mathbb{P}}\left(\boldsymbol{x}\left(t_j\right)\right)$, as done in (3.6).

A compact solution that allows to overcome this problem consists in assuming that the target function $f_i\left(\cdot\right)$ belongs to the Reproducing Kernel Hilbert Space (RKHS) associated to a polynomial kernel, see Rasmussen and Williams (2006). More precisely, when considering the space of inhomogeneous polynomials defined on the components of $\boldsymbol{x} \in \mathbb{R}^d$, with maximum degree p, the polynomial kernel is classically defined as

$$k_{pl}^{(p)}\left(\boldsymbol{x}\left(t_h\right), \boldsymbol{x}\left(t_j\right)\right) = \left(\sigma_p^2 + \boldsymbol{x}^T\left(t_h\right) \Sigma_p \boldsymbol{x}\left(t_j\right)\right)^p, \tag{3.9}$$

where $\sigma_p^2 > 0$ and $\Sigma_p > 0$ are the kernel hyperparameters, see Rasmussen and Williams (2006). Unfortunately, as highlighted in Rasmussen and Williams (2006) (chapter 4.2.2), the kernel function in (3.9) 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 3.3.1.

A valid alternative to (3.9) is represented by MPK, recently introduced in Dalla

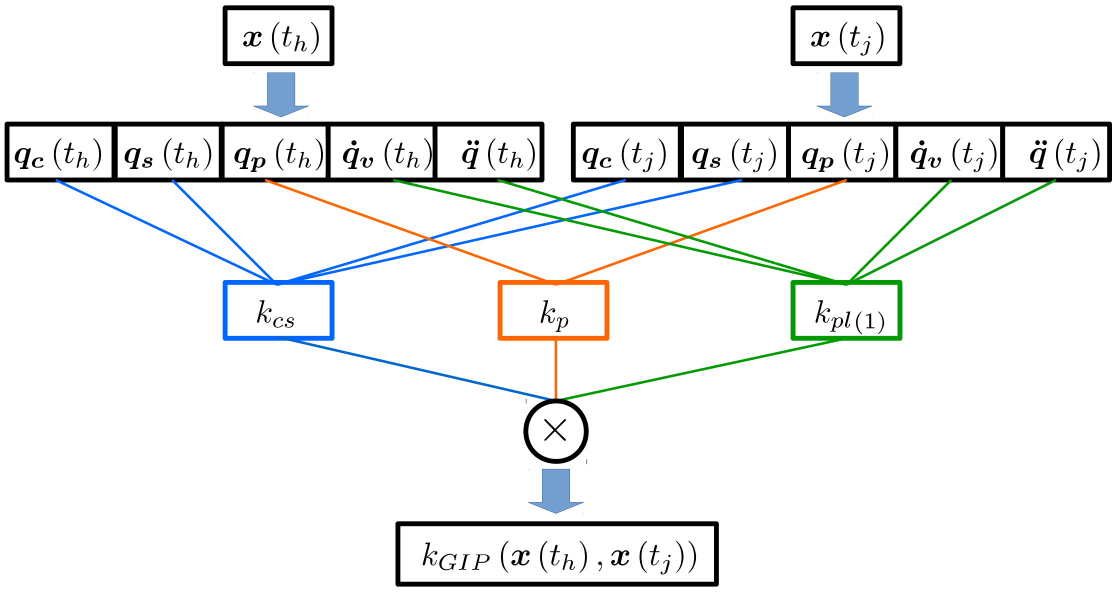

Figure 3.1: Schematic representation of the GIP kernel.

Libera et al. (2019c), and further discuss in Chapter 6. When considering the space of inhomogeneous polynomial with maximum degree p, the MPK is defined as the product of p linear kernels,

$$k_{mpk}^{(p)}(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j)) = \prod_{s=1}^{s=p} \left(\sigma_s^2 + \boldsymbol{x}^T(t_h)\Sigma_s \boldsymbol{x}(t_j)\right), \tag{3.10}$$

where the $\Sigma_s \in \mathbb{R}^{d\times d}$ matrices are distinct diagonal matrices. The diagonal elements, together with the parameters σ_s^2, compose the hyperparameters set of the MPK.

Observe that the RKHSs identified by (3.10) and (3.9) contain the same basis functions. However, as discussed in Chapter 6, (3.10) is equipped with a richer set of hyperparameters, that can be tuned by ML maximization, and allows to better select the monomials that really influence the system output.

3.3.3 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 model each joint torque with a zero-mean GP, and, driven by Proposition 3.3.1, we assume that the inverse dynamics is a polynomial in the input space $\bar{\boldsymbol{x}}$, defined in (3.8). 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 (3.10), where

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

The resulting kernel for the i-th joint is

$$\begin{aligned} k_i\left(\bar{\boldsymbol{x}}\left(t_h\right), \bar{\boldsymbol{x}}\left(t_j\right)\right) &= k_{cs}\left(\boldsymbol{q}_{cs}\left(t_h\right), \boldsymbol{q}_{cs}\left(t_j\right)\right) \\ &\quad k_p\left(\boldsymbol{q}_p\left(t_h\right), \boldsymbol{q}_p\left(t_j\right)\right) \\ &\quad k^{(1)}_{mpk}\left(\boldsymbol{q}_{av}\left(t_h\right), \boldsymbol{q}_{av}\left(t_j\right)\right), \end{aligned} \tag{3.11}$$

with

$$\begin{aligned} k_{cs}\left(\boldsymbol{q}_{cs}\left(t_h\right), \boldsymbol{q}_{cs}\left(t_j\right)\right) &= \prod_{b=1}^{N_r} k^{(2)}_{mpk}\left(\boldsymbol{q}_{cs_b}\left(t_h\right), \boldsymbol{q}_{cs_b}\left(t_j\right)\right), \\ k_p\left(\boldsymbol{q}_p\left(t_h\right), \boldsymbol{q}_p\left(t_j\right)\right) &= \prod_{b=1}^{N_p} k^{(2)}_{mpk}\left(q_{p_b}\left(t_h\right), q_{p_b}\left(t_j\right)\right). \end{aligned}$$

In Figure 3.1 we reported a schematic representation of the GIP kernel.

3.4 Experimental Results

We tested the proposed approach both in a simulated environment and in a real environment. Regarding technical aspects, we implemented all the considered algorithms in Python. To speed up algebraic operations, we largely exploited the functionalities provided by Pytorch Paszke et al. (2017). The code[1] and the datasets[2] are publicly available.

3.4.1 Simulated SCARA robot

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

[1]https://bitbucket.org/AlbertoDallaLibera/gip_kernel
[2]https://mega.nz/#F!fbBBSCCK!NwRs60ace05mTe2Ot5fz-Q

Robot. The SCARA is a 4 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 were computed through (3.1), assuming complete knowledge of the joint trajectories. Equation (3.1) was derived using the python package *Sympybotics*[3].

Estimation accuracy

In the first experiment, we tested the estimators' accuracy. The proposed approach is compared with both model-based and data-driven estimators.As far as model-based estimators are concerned, we implemented three solutions. The first estimator is a classical Fisherian estimator (FE), based on (3.2); in particular we considered the implementation provided by *Sympybotics*, see Sousa and Cortesão (2014) for details. The other two model-based estimators are the PP and SP kernel-based estimators, where the model-based component is defined as in (3.6). In all the three solutions, we computed the Φ^d matrix in (3.2) assuming the nominal kinematic parameters provided by the manufacturer. To account for behaviors due to model bias, we varied the values of the kinematic parameters of the model generating data around the nominal values, so that the Φ^d matrix used by the FE and the PP and SP kernels is different from the one generating data. Parameters perturbations are uniform random variables, ranging $[-0.05, 0.05]\, m$ for lengths, and $[-5, 5]\, deg$ for angles. Instead, as far as data-driven approaches are concerned, we tested an RBF kernel-based estimator and a neural network. The neural network is a fully connected network with two hidden layers, each of which is composed of 400 sigmoids. Recurrent architectures have not been considered since typically they are applied to deal with the on-line adaptation problem, which is out of the scope of this work.

To obtain statistically significant results, we performed a Monte Carlo analysis, composed of 20 experiments. A single experiment consists in sampling a model perturbation and performing two simulations, one for the training set and one for the test set. In each simulation, joints follow a distinct random trajectory, given by the sum of 200 sinusoids with random amplitudes and angular velocities sampled in the range $[-2, 2]\, rad/sec$. A zero-mean Gaussian noise with standard deviation $0.01\, Nm$ was added to the output of (3.1), resulting in a high signal to noise ratio. Both the training and test dataset are composed of 2000 samples.

The hyperparameters of the GPR based estimators were trained by ML maximization (see Section 1.3). As concerns $\bar{\boldsymbol{w}}^d$ and Σ_{w^d}, i.e., the prior distribution of the model-based kernels, we considered $\bar{\boldsymbol{w}}^d = 0$, and Σ_{w^d} equals to a diagonal matrix with distinct diagonal elements. Instead, as concerns the optimization of neural network parameters, the Mean

[3]https://github.com/cdsousa/SymPyBotics

Squared Error (MSE) was considered as loss function, defined as

$$MSE(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \sum_{j=1}^{N} \left(y_i(t_j) - \hat{y}_i(t_j)\right)^2 / N,$$

with N equals to the number of samples. Both for GPRs and the neural network, we used Adam as optimizer Kingma and Ba (2015). Further details about the optimization, size of the batch and number of epochs, are provided in the source code.

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

$$nMSE(\boldsymbol{\tau}_i, \hat{\boldsymbol{y}}_i) = MSE(\boldsymbol{\tau}_i, \hat{\boldsymbol{y}}_i)/Var\left(\boldsymbol{\tau}_i\right).$$

In Figure 3.2, we have plotted the obtained $nMSE$s 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 $nMSE$s of RBF kernel-based estimator and neural network estimator are in most of the trials higher than 10%. Instead, the GIP kernel-based estimator provides accurate estimates, as proven by $nMSE$s 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, the GIP kernel-based estimator performs similarly to the model-based approaches. Actually, in joint 2 and 3, the proposed approach outperforms FE and the PP kernel-based estimator, whose performances are affected by model bias. Results confirm also the validity of semiparametric schemes, proving that the 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 the RBF component might not be effective in compensating for model bias, in particular when the performance of the data-driven estimator is low, as proven by the $nMSE$s in joint 3, where the GIP kernel-based estimator is more accurate than SP. Finally, a comparison of the FE and PP performance suggests that Fisherian approaches are more sensitive to model bias; notice, in particular, the $nMSE$s obtained in joint 1 and 4.

Data efficiency

In the second experiment, we tested the data efficiency of different estimators. Since our focus is on comparing data-driven approaches, model bias was not 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

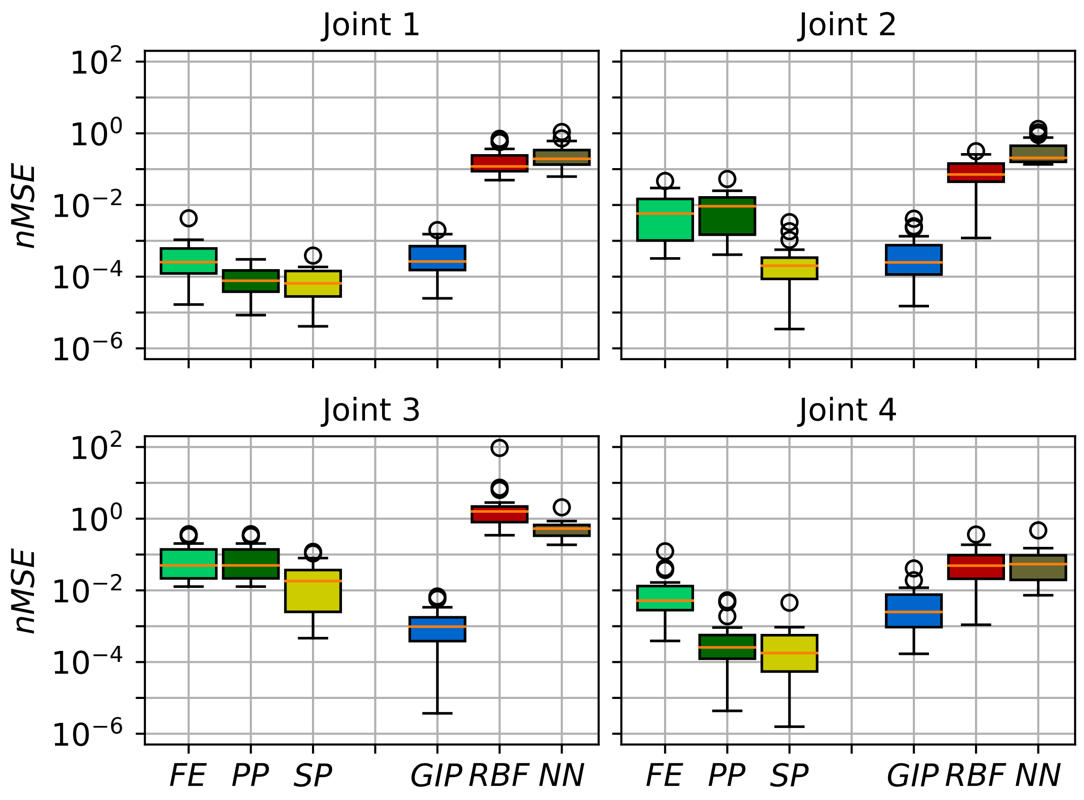

Figure 3.2: Boxplot of the $nMSE$s obtained in 20 experiments. Model-based estimators (FE, PP, and SP) are on the left, while the data-driven estimators (GIP, RBF, and NN, the fully connected neural network) are on the right.The upper and the lower vertical edge of the box represent the first and the third quartile, while the median is pointed out by the orange line. The black vertical lines indicate variability outside the first and third quartile; circles point out outliers. Results are plotted in logarithmic scale.

scenario, where data are generated with the robot nominal parameters, the performance of PP might be considered as the baseline of an optimal solution.

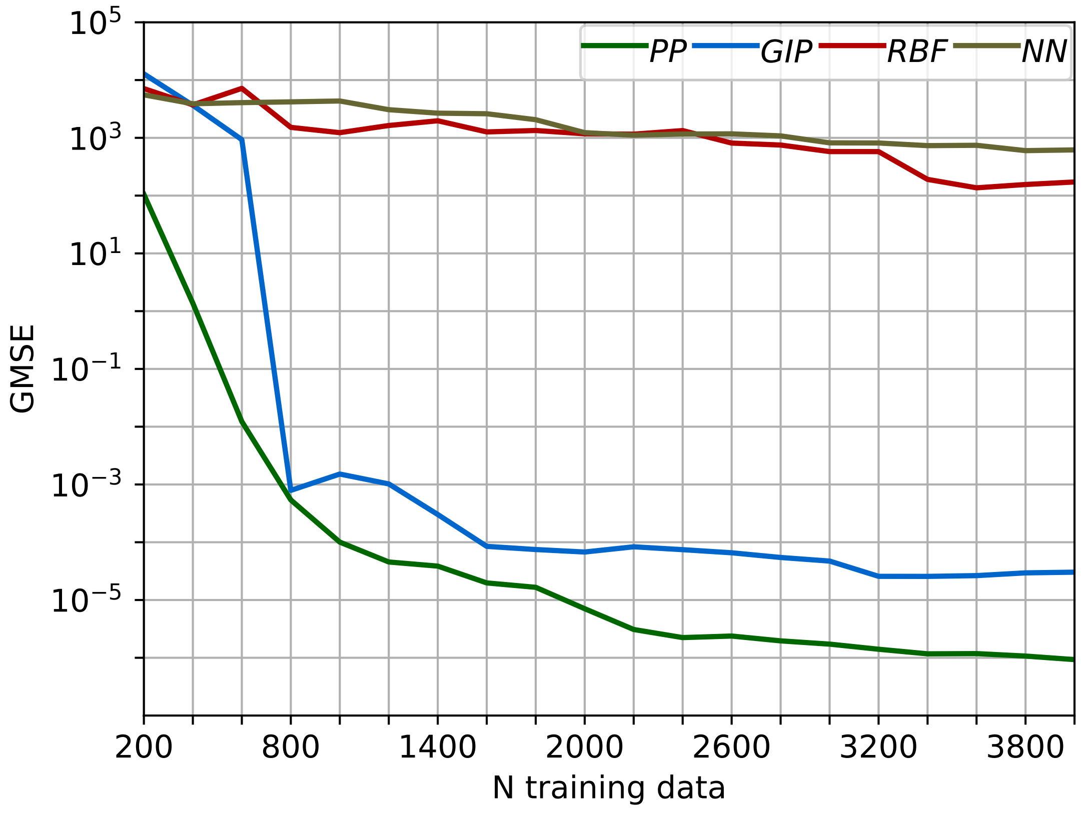

Figure 3.3: Comparison of the neural network (NN) and the PP, RBF and GIP estimators in terms of accuracy and data efficiency. The plot shows the evolution of $GMSE$ in the test set, as a function of the training samples available. Results are plotted in logarithmic scale.

The experiment is composed of training and test simulation, with joints trajectories generated as in the previous experiment; each dataset accounts for 4000 samples. Results are reported in Figure 3.3, where we have plotted the evolution of the Global Mean Squared Error ($GMSE$), i.e., the sum of the $MSE(\boldsymbol{\tau}_i, \hat{\boldsymbol{y}}_i)$s of the four joints, as function of the number of training samples used to train and derive estimators. The evolutions of the errors show that the proposed solution outperforms the other data-driven estimators, both in terms of accuracy and data efficiency, given that its $GMSE$ is lower and decreases faster. As in the previous experiment, the GIP kernel-based estimator behaves more similarly to the model-based approach than to the other data-driven solutions, proving its data efficiency.

3.4.2 UR10 robot

We used a Universal Robots UR10 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 $\boldsymbol{i}$. Assuming that the behaviors due to elasticity are negligible, i.e., $\boldsymbol{\theta} = K_r\boldsymbol{q}$, where $\boldsymbol{\theta}$ contains the motor angles and K_r is the diagonal matrix of the gear reduction ratio, the inverse dynamics in (3.1) can be rewritten as

$$K_{eq}\boldsymbol{i} = B_{eq}\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}} + C\left(\boldsymbol{q},\dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{g}\left(\boldsymbol{q}\right) + F_v\dot{\boldsymbol{q}} + F_c sign\left(\dot{\boldsymbol{q}}\right),$$

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

The interface with the robot is based on ROS Quigley et al. (2009), through the *ur_modern_driver*[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 collected dataset is publicly available, and it has been designed to stress generalization properties. 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. Instead, the test dataset is composed of two types 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 optimization procedures and the considered estimators are the same of the previous experiment. Due to space constraints, we neglect the FE, which performs similarly to PP. Given the higher complexity of the UR10 inverse dynamics, the number of hidden units of the neural network was increased to 600. Concerning the derivation of the GPR based estimators, i.e., the computation of (3.4) and (3.5), we used a subset of the training data, in order to reduce the computational burden induced by (3.5); in particular, we selected 4000 samples, downsampling with constant step the 40000 training 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 3.4, confirm the

[4]https://github.com/ThomasTimm/ur_modern_driver

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 is 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 confirms 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 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 Dalla Libera et al. (2019b) and also in Chapter 4, 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.

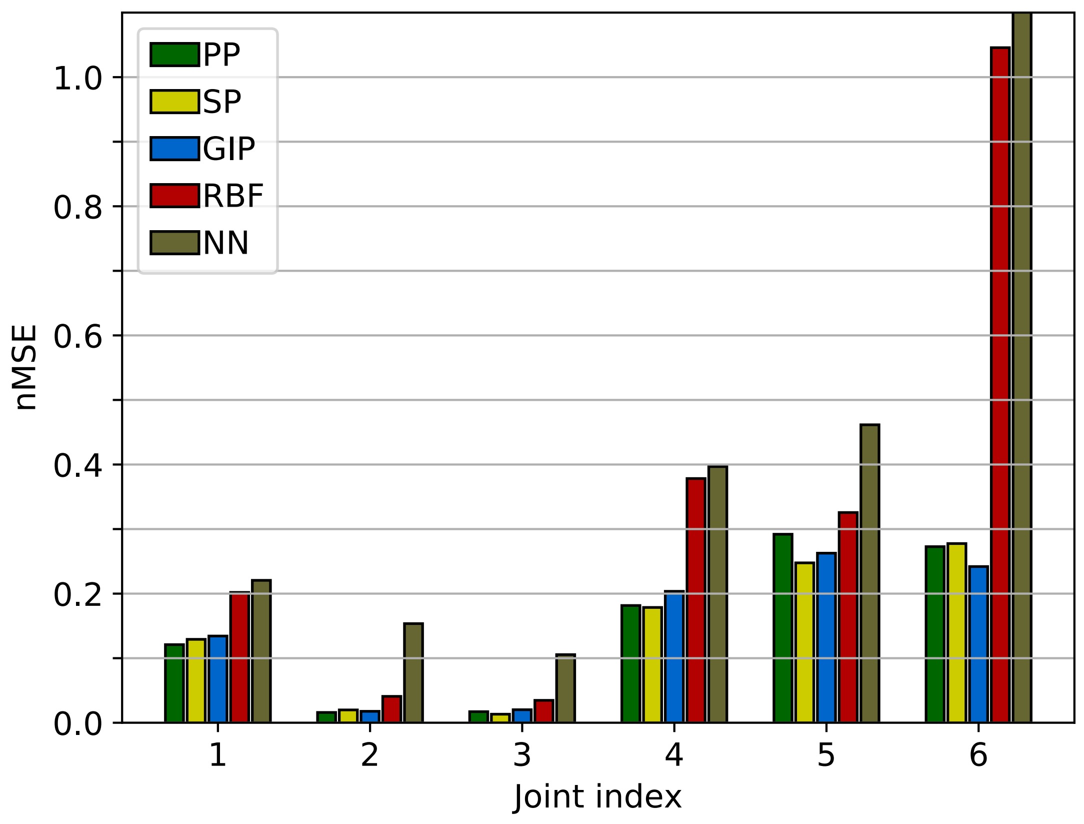

Figure 3.4: Comparison of the neural network (NN) and the PP, RBF and GIP kernel based estimators in the UR10 inverse dynamics prediction.

3.5 Conclusion

In this chapter, 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 a real system, this property allows deriving accurate inverse dynamics estimators directly from data, without the need of prior knowledge about the model and using a small amount of data. Indeed, numerical results show that the 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.

4 Proprioceptive collision detection via Gaussian Process Regression

4.1 Introduction

Collaborative Robotics has attracted increasing interest over the last decade in the research community, mainly due to the fact that the design of robots able to collaborate with humans might have a great impact on several domains. For instance, in the medical field robots might assist humans in rehabilitation from injuries Gelderblom et al. (2009), or in industrial applications, the robot working with a human being might relieve him from the most physically demanding activities, Masinga et al. (2015).

Human-robot collaboration is a challenging topic under different points of view but, likely, the most critical aspects are related to safety. Indeed, when robots and humans work side-by-side, they need to share their workspace, and, in these circumstances, robots should avoid dangerous and unexpected collisions with humans. When camera and proximity sensors are available, one might develop motion planning algorithms minimizing the collision probability Ebert and Henrich (2002). However, it is impossible to reduce the collision risk to zero. Clearly, in this context, it is fundamental that robots are provided with robust strategies that can promptly detect collisions. Moreover, once a collision has been detected, the robot has to classify such collision, in particular discriminating between intended and unintended contacts, and it has to react accordingly.

In order to detect the interaction with the external environment, robots might be endowed with specific sensors, like artificial skins or force-sensors. However, this approach might have some limitations. Indeed artificial skins do not provide information about the collision intensity Cirillo et al. (2016), while six-axis force-sensors are expensive and highly sensitive to environmental parameters like temperature and humidity.

A solution alternative to the use of additional sensors is proprioceptive collision detection (CD) Haddadin et al. (2017). Proprioceptive collision detection algorithms identify when an external force is applied using only proprioceptive sensors, namely joint torque sensors and current sensors, besides the joint coordinates. We refer the interested reader to Haddadin et al. (2017) for an overview of the main state of the art collision detection algorithms. All the proposed approaches require the definition of a monitoring signal $\boldsymbol{s}(t)$ and a threshold $\boldsymbol{\sigma}_{CD}$. The algorithms assume that a collision occurred when $\boldsymbol{s}$ exceeds $\boldsymbol{\sigma}_{CD}$, see Le, Choi, and Kang (2013),Villagrossi, Beschi, Simoni, and Pedrocchi (2018), De Luca, Albu-Schaffer, Haddadin, and Hirzinger (2006) and De Luca and Mattone (2003). The different strategies can be classified based on the way the monitoring signal is defined. The most direct and intuitive technique is to define $\boldsymbol{s}$ as the estimate of the external torques Le et al. (2013),Villagrossi et al. (2018). Instead, in De Luca et al. (2006) $\boldsymbol{s}$ is defined as the estimate of the power due to external forces. Another solution is proposed in De Luca and Mattone (2003), where the authors define a monitoring signal based on the generalized momentum. It is worth remarking that this class of solutions requires accurate knowledge of the robot dynamics model since they assume to know both the kinetics and dynamics parameters. Typically the former parameters are known, while the latter ones are estimated resorting to Fisherian estimators Sousa and Cortesão (2014).

In this work, to detect if an interaction has occurred, we propose a novel approach based on the Gaussian Process Regression (GPR) framework. This approach has minimal sensing requirements since it needs only to measure the joint coordinates and the motor currents. In this work, we extend the GPR algorithms based on semi-parametric priors (i.e., composed by the sum of a parametric component and a non-parametric component) developed to learn the robot inverse dynamics, see Section 3.2.2. Compared to the standard approach, our algorithm can efficiently deal also with *quasi-static* configurations, namely, when the robot is stuck or the joints' velocities are very low. Specifically, relying on an enlarged set of input features and designing proper kernel structure, our estimator can model the complex behaviors due to static frictions and kinetic frictions at low velocities.

The main contribution of this chapter is the derivation of a GPR-based learning algorithm able to estimate the currents due to external forces. As showed by our numerical results, the estimator can detect when an interaction with the external environment has occurred.

The chapter is organized as follows: in Section 4.2 we briefly review the state of the art proprioceptive collision detection algorithms that are based on external torques estimation. In Section 4.3 we present our collision detection strategy based on Gaussian Regression. In Section 4.4 we briefly recall standard GPR techniques adopted in the learning of the robot inverse dynamics, highlighting via a numerical example the limitations of these approaches when used to detect collision in *quasi-static* configurations. Then, in Section 4.5 we formally describe our learning algorithm and in Section 4.6 we show numerical results obtained using a UR10 robot.

4.2 CD via direct estimation of external torques

In this section, we describe a collision detection strategy based on the monitoring of the external torques estimates, also named *direct estimation*, see Haddadin et al. (2017), Le et al. (2013) and Villagrossi et al. (2018). When a collision occurs, an external force $\boldsymbol{F}^e(t)$ is applied to the robot, and, consequently, the joints are subject to an additional external torque $\boldsymbol{\tau}^e(t)$. Consider an n joints manipulator and let $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$, $\ddot{\boldsymbol{q}}(t)$ and $\boldsymbol{\tau}(t) \in \mathbb{R}^n$ be, respectively, the vectors of joints positions, velocities, accelerations and joint torques at time t; in the following, to keep the notation compact, we point out explicitly the time dependence only when it is necessary. Then we have

$$\boldsymbol{\tau} = M(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{\tau}^{\epsilon} - \boldsymbol{\tau}^{e}, \tag{4.1}$$

where $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the generalized inertia matrix, $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ models the effects due to the gravitational force, and $\boldsymbol{\tau}^{\epsilon} \in \mathbb{R}^n$ describes the torques related to the unmodeled dynamic behaviors, mainly frictions Siciliano et al. (2009).

Collision detection through direct monitoring of $\boldsymbol{\tau}^e$ defines the monitoring signals $\boldsymbol{s}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau})$ equal to $\hat{\boldsymbol{\tau}}^e(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau})$, the estimate of $\boldsymbol{\tau}^e$ obtained from equation (4.1). Let $\hat{M}(\boldsymbol{q})$, $\hat{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\hat{\boldsymbol{g}}(\boldsymbol{q})$ and $\hat{\boldsymbol{\tau}}^{\epsilon}$ be, respectively, the estimates of $M(\boldsymbol{q})$, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$, $\boldsymbol{g}(\boldsymbol{q})$ and $\boldsymbol{\tau}^{\epsilon}$, and denote by $\boldsymbol{y}$ a noisy measure of $\boldsymbol{\tau}$, then we have

$$\hat{\boldsymbol{\tau}}^e = \hat{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \hat{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \hat{\boldsymbol{g}}(\boldsymbol{q}) - \hat{\boldsymbol{\tau}}^{\epsilon} - \boldsymbol{y}. \tag{4.2}$$

Ideally, we should have $\boldsymbol{s}(\cdot) = 0$ when $\boldsymbol{\tau}^e = 0$; in practice, due to the model inaccuracies and the measurement noise, it happens that the monitoring signal is different from zero even when no external forces are applied. Consequently, the introduction of a threshold $\boldsymbol{\sigma}_{CD}$ to be compared with the value of $\boldsymbol{s}$ is necessary. Then, we define the binary collision function $f_{CD}(\cdot)$ as

$$f_{CD}(\boldsymbol{s}) = \begin{cases} \text{TRUE}, & if\ |\boldsymbol{s}| \geq \boldsymbol{\sigma}_{CD} \\ \text{FALSE}, & if\ |\boldsymbol{s}| < \boldsymbol{\sigma}_{CD} \end{cases},$$

where $|\cdot|$, $\geq$ and $<$ are element-wise operators and $|\boldsymbol{s}| \geq \boldsymbol{\sigma}_{CD}$ if the relation holds at least for one component. The value of $\boldsymbol{\sigma}_{CD}$ is set by cross-validation with the purpose of limiting the number of false positives and false negatives. Typically the identification of $\boldsymbol{\sigma}_{CD}$ is done observing the evolution of $\boldsymbol{s}(\cdot)$ obtained while the robot is moving with $\boldsymbol{\tau}^e = 0$ for a time interval sufficiently large from the statistical point of view, see for examples Haddadin et al. (2017).

Finally observe that in (4.2), to compute $\hat{\boldsymbol{\tau}}^e$, the model of the robotic arm is assumed known. This model depends on the kinematic parameters and dynamics parameters. Typically, kinematic parameters are known quite accurately, while dynamics parameters are estimated by resorting to some Fisherian approach Sousa and Cortesão (2014).

4.3 GPR for Proprioceptive collision detection

In this work, we propose an approach based on the GPR framework to solve the CD problem. GPR is used to build the monitoring signal $\boldsymbol{s}$. In the following, instead of measuring directly the joint torque $\boldsymbol{\tau}$, we assume to measure $\boldsymbol{i}$, the currents of the joints motors; this is due to the fact that in our experimental setup we have access to $\boldsymbol{i}$, and not to $\boldsymbol{\tau}$. However, it is worth stressing that a current-based approach has minimal requirements as far as the number of sensors employed is concerned.

To consider the motor currents $\boldsymbol{i}$ instead of $\boldsymbol{\tau}$, we need to include the mechanical equations of the motors in the robotic arm model, as well as to assume that behaviors due to elasticity are negligible. Let $\boldsymbol{\theta}(t)$, $\dot{\boldsymbol{\theta}}(t)$ and $\ddot{\boldsymbol{\theta}}(t)$ be the angular position, velocity, and acceleration of the motors. Then the mechanical equations of the motors are

$$J_m \ddot{\boldsymbol{\theta}} + B_m \dot{\boldsymbol{\theta}} + \boldsymbol{\tau}^l = K_\tau \boldsymbol{i}, \tag{4.3}$$

where $\boldsymbol{\tau}^l$ are the torques due to the load, and J_m, B_m and $K_\tau \in \mathbb{R}^{n\times n}$ are diagonal matrices containing respectively the rotors inertia, the motors damping coefficients and the torques-current ratios. When the behaviors due to the elasticity of the gears are negligible, it holds $\dot{\boldsymbol{\theta}} = K_r \dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{\theta}} = K_r \ddot{\boldsymbol{q}}$, and $\boldsymbol{\tau}^l = K_r^{-1}\boldsymbol{\tau}$, with $K_r \in \mathbb{R}^{n\times n}$ equals to the diagonal matrix containing the gear reduction ratios. Substituting these equations in (4.3), we can express $\boldsymbol{\tau}$ as function of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ and $\boldsymbol{i}$, and equation (4.1) becomes:

$$M_{eq}\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}} + C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{g}\left(\boldsymbol{q}\right) - \boldsymbol{\tau}^{\epsilon} - \boldsymbol{\tau}^{e} + B_{eq}\dot{\boldsymbol{q}} = K_{eq}\boldsymbol{i}, \tag{4.4}$$

where, for compactness, we defined $M_{eq}(\boldsymbol{q}) = M(\boldsymbol{q}) + K_r^2 J_m$, $B_{eq} = K_r^2 B_m$ and $K_{eq} = K_\tau K_r$.

Instead of estimating $\boldsymbol{\tau}^e$ from (4.2), we propose to learn a GPR model that provides an estimate of $\boldsymbol{i}$, denoted as $\hat{\boldsymbol{i}}$, when $\boldsymbol{\tau}^e = 0$; more specifically we train a suitable GPR model for $\boldsymbol{i}$, over a sufficiently rich dataset containing only collision-free trajectories. Then, the monitoring signal $\boldsymbol{s}$ is defined as the difference between the measured current $\boldsymbol{i}$ and $\hat{\boldsymbol{i}}$. Clearly, if no collision has occurred, i.e., $\boldsymbol{\tau}^e$ is effectively null, we expect $\hat{\boldsymbol{i}}$ to be close to $\boldsymbol{i}$, and, in turn, $\boldsymbol{s}$ to be small; viceversa if a collision has happened, i.e, $\boldsymbol{\tau}^e \neq 0$, then $\hat{\boldsymbol{i}}$ should be significantly different from $\boldsymbol{i}$, and $\boldsymbol{s}$ should become sufficiently large to detect the contact.

We stress the fact that in this work we focus only on the development of GPR models able to produce proper monitoring signals, and we do not discuss any strategy to design the threshold $\boldsymbol{\sigma}_{CD}$; $\boldsymbol{\sigma}_{CD}$ might be set using standard rules Haddadin et al. (2017), like cross-validation. It is worth mentioning that GPR techniques return also the confidence intervals of the computed estimates. This fact might be exploited to define input dependent thresholds less prone to false positive. We let the discussion about the definition of $\boldsymbol{\sigma}_{CD}$ as future works.

4.4 GPR for robot inverse dynamics: critical issues

In the first part of this section, we briefly review standard GPR based solutions adopted to identify the robot inverse dynamics, already presented in Section 3.2.2. Moreover, we discuss possible improvements of solutions based on Parametric Prior. Finally, through a numerical example, we highlight some critical issues that emerge when these standard models are used to solve the CD problem.

4.4.1 GPR for inverse dynamics: compensation of kinetic friction

We recall that, when modeling the inverse dynamics relying on GPR, a typical strategy consists in modeling each joint torque output with an independent Gaussian process (GP). Let $(X, \boldsymbol{y}_i)$ be an input-output dataset relative to the i-th joint torque, where $X = \{\boldsymbol{x}(t_1), \ldots, \boldsymbol{x}(t_{N_{TR}})\}$, with $\boldsymbol{x}(t) = [\boldsymbol{q}(t)\ \dot{\boldsymbol{q}}(t)\ \ddot{\boldsymbol{q}}(t)]$, and $\boldsymbol{y}_i$ is the vector collecting the noisy measure of the i-th joint torque, $\tau_i(t)$ with $t = t_1, \dots, t_{N_{TR}}$. Then, as described in Section 3.3, $\boldsymbol{y}_i$ is modeled as the sum of an unknown function f_i, representing the i-th torque, and Gaussian i.i.d. noise, with zero mean and standard deviation σ_n. The unknown function f_i is a GP, with prior mean m_i and prior covariance defined through a kernel function $k_i(\cdot, \cdot)$. Remarkably, the posterior distribution of the target function can be computed in closed form. In particular, we have that, given a general input vector $\boldsymbol{x}(t_*)$, the maximum a posteriori estimate of $f_i(t_*)$, i.e., the estimate of $\tau_i(\boldsymbol{x}(t_*))$, is given in closed form by (3.4).

The crucial aspect in GPR is the definition of the f_i prior distribution. In Section 3.2.2, we have seen that the different solutions proposed for inverse dynamics can be grouped into three classes:

- Non-Parametric prior (NP), like the RBF kernel;
- Parametric prior (PP);
- Semi-Parametric prior (SP).

PP and SP solutions are based on the linear model of the inverse dynamics w.r.t. $\boldsymbol{w}^d \in \mathbb{R}^{N_{par}}$, the vector collecting the dynamics parameters, i.e., masses, centers of mass and inertia of the links. In particular, recalling (3.2), we have

$$\tau_i(t) = \left(\boldsymbol{\phi}_i^d(\boldsymbol{x}(t))\right)^T \boldsymbol{w}^d \,,$$

where $\boldsymbol{\phi}_i^d(\boldsymbol{x}_t) \in \mathbb{R}^{N_{par}}$ is a given vector that depends on the kinematics parameters, see Siciliano et al. (2009). The previous equation has been derived assuming $\boldsymbol{\tau}^\epsilon$ equal to zero. A refinement of this model can be obtained including also some terms modeling the frictions effects, namely, considering $\boldsymbol{\tau}^\epsilon = \boldsymbol{\tau}^f$, with $\boldsymbol{\tau}^f$ equals to the torques due to frictions. The simplest and most used model to describe the torque applied to the i-th joint by frictions, denoted as τ_i^f, is given by

$$\tau_i^f(t) = \begin{cases} \tau_i(t) & \text{if } \dot{q}_i(t) = 0 \ , \ \tau_i(t) \leq F_i^s \\ F_i^k sign(\dot{q}_i(t)) + F_i^v \dot{q}_i(t) & \text{if } |\dot{q}_i(t)| > 0 \end{cases} \,, \tag{4.5}$$

where F_i^s, F_i^k and F_i^v are, respectively, the static friction coefficient, the kinetic friction coefficient and the viscous friction coefficient of the i-th joint Dupont (1990). Notice that, when $\dot{q}_i$ is not null, τ_i^f is linear w.r.t. F_i^k and F_i^v, and hence the behaviors due to the kinetic frictions can be easily merged in (3.2), leading to the augmented equation

$$\tau_i(\boldsymbol{x}(t)) = \left[\left(\boldsymbol{\phi}_i^d(\boldsymbol{x}(t))\right)^T \quad \left(\boldsymbol{\phi}_i^f(\boldsymbol{x}(t))\right)^T \right] \begin{bmatrix} \boldsymbol{w}^d \\ \boldsymbol{w}_i^f \end{bmatrix} := \boldsymbol{\phi}_i^T(\boldsymbol{x}(t))\, \boldsymbol{w_i} \tag{4.6}$$

where $\boldsymbol{\phi}_i^f(\boldsymbol{x}(t)) = [sign(\dot{q}_i) \quad \dot{q}_i]$ and $\boldsymbol{w}_i^f = \left[F_i^k \quad F_i^v\right]^T$. Finally, as done in (3.6), assuming $\boldsymbol{w}_i \sim N(\bar{\boldsymbol{w}}_i, \Sigma_{w_i})$, we obtain the following prior

$$\begin{aligned} m_i(x(t)) &= \phi_i^T(x(t))\bar{\boldsymbol{w}}_i\,, \\ k_i(\boldsymbol{x}(t_h), \boldsymbol{x}(t_j)) &= \boldsymbol{\phi}_i^T(\boldsymbol{x}(t_h))\, \Sigma_{w_i} \boldsymbol{\phi}_i(\boldsymbol{x}(t_j)) \end{aligned} \tag{4.7}$$

4.4.2 Limitations of proprioceptive CD with standard GPR approach

In this subsection we describe a simple experiment that highlights the limitations of the GPR models previously introduced when working in the *quasi-static* configurations, namely configurations with joints velocity close to zero. The experiment has been performed with a UR10 robot and consists of a succession of rest phases (all the joints stuck and parallel to the ground) and moving phases. In the moving phases, only the first joint is actuated. The values of q_1 in the rest phase are sequentially $\frac{\pi}{2}$, 2.09, $\frac{\pi}{2}$, 0.52 $[rad]$. The evolution of q_1 and $\dot{q}_1$, together with the current i_1, are reported in Figure 4.1.

It is particularly interesting analyzing the value of i_1 during the rest phases. Firstly, notice that i_1 is significantly different from zero. However, due to the UR10 geometry, during the rest phases, the torque applied to the first joint should be null, given that the robot is not moving, and gravitational contribution are null. Second, observe that in the rest phases with $q_1 = \pi/2$, highlighted by the grey bars, despite the robot is in the same configuration $\boldsymbol{x}$, the current i_1 assumes three different values. Referring to the GPR notation, the function $f_1(\cdot)$ attains different values in the same input location $\boldsymbol{x}$. The differences among these values are so significant that they can not be explained by only the presence of noise in the measurements.

These two unexpected behaviors are due to the complexity of joint frictions in *quasi-static* configurations, Dupont (1990). Indeed, as described by (4.5), when the joint velocity is close to zero, torques generated by frictions are highly non-linear, as well as dependent on $\boldsymbol{\tau}$. Similar behaviors have been highlighted also in Vuong and Jr (2007). The authors observed that the friction model in (4.5) is accurate only when the module of joint velocity is greater than a certain threshold σ_v. The value of σ_v depends on several aspects, like the type of motors used, or the materials the joint is made. This fact results in the difficulty of characterizing a priori σ_v. As suggested also in Vuong and Jr (2007),

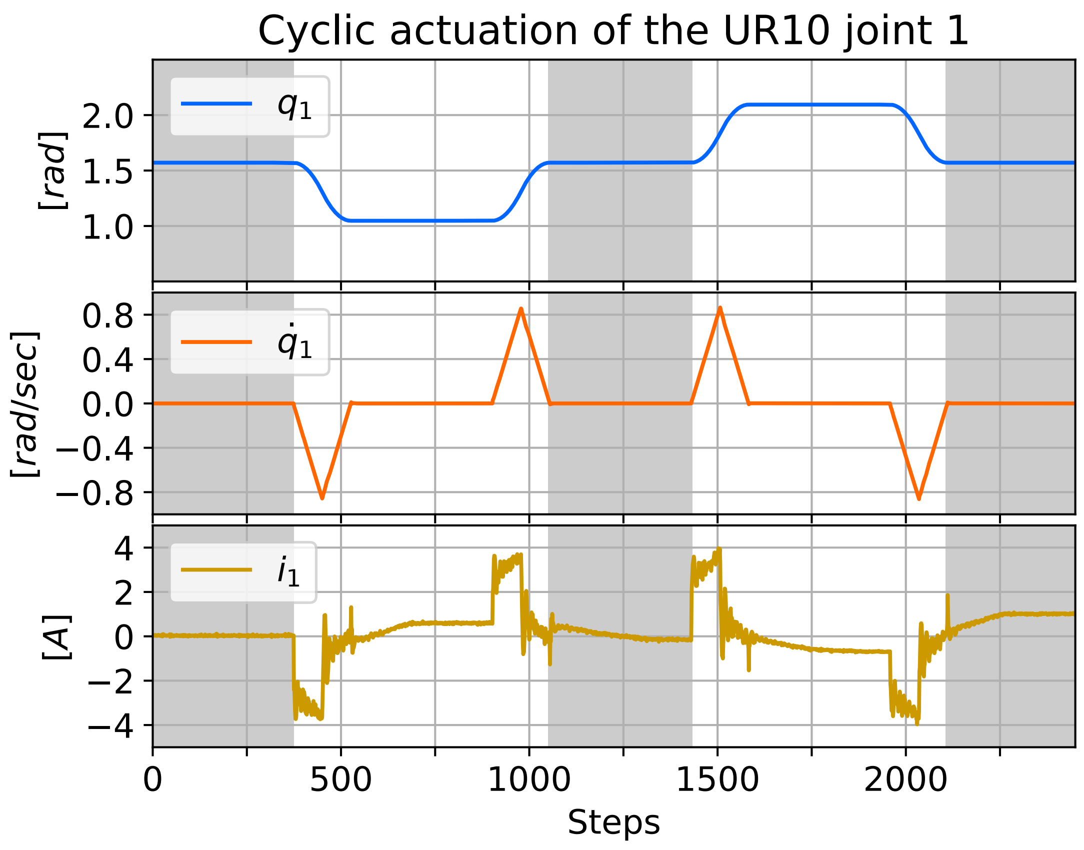

Figure 4.1: Cyclic actuation of the UR10 first joint (all the other joint are not moving). q_1, $\dot{q}_1$ and i_1 denote, respectively, the position, the velocity and the motor current of the first joint. The grey bars highlights the rest phases associated to the same robot configuration.

σ_v can be identified by cross-validation, measuring the out of sample performance. In our setup we have found $\sigma_v = 10^{-2}[rad/sec]$.

As confirmed by experimental results reported in Section 4.6.1, the two aspects previously highlighted could compromise significantly the performance of standard GPR models proposed to identify the inverse dynamics, leading to a high value of σ_{CD}. In particular, PP-based estimators could suffer due to the inaccuracy of the friction model introduced in (4.6). Instead, NP approaches could not be effective due to the significantly different values assumed by the target function f_1 in the same input location $\boldsymbol{x}$.

4.5 Semi-Parametric GP with NP friction compensation

The proposed solution for learning the motor currents is motivated by the following observations. (i) Experimental results in Section 4.6.1 show that, when working in a *dynamic* configuration, standard GPR approaches provide accurate estimates. (*ii*) When dealing with the *quasi-static* configuration, we need to include additional features in the input space, to avoid that the same input is mapped into different outputs. (*iii*) We need to model the discontinuity due to the different behaviors of static frictions and kinetic frictions, i.e., we need to provide a unified framework capturing the behaviors in both scenarios, *dynamic* and *quasi-static*.

Based on the above observations, we propose to model the function f_i adopting an SP-based GP, with the PP-based component f_{PP_i} that includes also the effects of the friction, and the NP-based component f_{NP_i} specifically thought to capture the behaviors generated by the frictions in *quasi-static* configurations. Before formally describing the model we consider, we provide some more details about the second and third observations above.

4.5.1 Additional features

Notice, from Equation (4.5), that when the velocity is null, important contributions are given by $\boldsymbol{\tau}$, that is a term related to the action of the controller. Consequently, in *quasi-static* configurations, it might be necessary to add to the GPR inputs some features related to the control actions. We stress the fact that, from a control point of view, we are operating in a black box context since we do not have access to the low-level controller of the UR robot that we used in our experiments.

Driven by such consideration, we defined the following augmented input vector to model the complex behaviors happening in *quasi-static*,

$$\bar{\boldsymbol{x}}(t) = \left[\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t), \boldsymbol{e}^q(t), \dot{\boldsymbol{e}}^q(t), \boldsymbol{i}^c(t)\right] , \tag{4.8}$$

where $\boldsymbol{e}^q(t)$ and $\dot{\boldsymbol{e}}^q(t)$ denote, respectively, the joint position and velocity errors at time t, while $\boldsymbol{i}^c(t)$ are the currents required by the controllers of the motors at the instant t.

The rationale behind the choice of adopting this set of features is the following: the variables $\boldsymbol{e}^q$ and $\dot{\boldsymbol{e}}^q$ allow to model proportional and derivative contributions, while the $\boldsymbol{i}^c$ bring information about non linear control actions (i.e., saturation) and dynamic contribution (i.e., integral contribution).

4.5.2 Modeling friction discontinuity through GPR

In our approach, the discontinuity of the behaviors due to static and kinematic frictions is modeled through an ad-hoc NP kernel. In particular, we exploited the *vertical rescaling* property of the kernels, introduced in Section 1.3. The NP-based component f_{NP_i} of our model is scaled by the function

$$a_i(\boldsymbol{x}(t)) = \begin{cases} 0 & if \ |\dot{q}_i(t)| \geq \sigma_v \\ 1 & if \ |\dot{q}_i(t)| < \sigma_v \end{cases} .$$

Then, it turns out that

$$f_{NP_i}(\bar{\boldsymbol{x}}(t)) = a_i(\boldsymbol{x}(t)) f^s_{NP_i}(\bar{\boldsymbol{x}}(t)), \tag{4.9}$$

where $f^s_{NP_i}$ is a GP with zero mean and covariance defined through an RBF kernel. Observe that, in this way, the NP-based component acts only when the i-th link is in *quasi-static* configurations, with the specific task of capturing the behaviors due to frictions at low velocity.

4.5.3 Proposed Algorithm: SP GPR with friction compensation

To sum up, we model each joint torque with the following expression

$$\begin{aligned} f_i(\bar{\boldsymbol{x}}(t)) =& f_{PP_i}(\boldsymbol{x}(t)) + a_i(\boldsymbol{x}(t)) f^s_{NP_i}(\bar{\boldsymbol{x}}(t)) \\ =& \boldsymbol{\phi}_i^T(\boldsymbol{x}(t))\boldsymbol{w_i} + a_i(\boldsymbol{x}(t)) f^s_{NP_i}(\bar{\boldsymbol{x}}(t)) \end{aligned} \tag{4.10}$$

where f_{PP_i} and $f^s_{NP_i}$ are independent GPs with zero mean. Then, assuming $\boldsymbol{w}_i \sim N(0, \Sigma_{w_i})$, and defining the $f^s_{NP_i}$ covariance through an RBF kernel, we obtain

$$k_i(\boldsymbol{x}(t_k), \boldsymbol{x}(t_j)) = \boldsymbol{\phi}_i^T(\boldsymbol{x}(t_k))\Sigma_{w_i}\boldsymbol{\phi}_i(\boldsymbol{x}(t_j)) + a_i(\boldsymbol{x}(t_k)) k_{RBF}(\bar{\boldsymbol{x}}(t_k), \bar{\boldsymbol{x}}(t_j)) a_i(\boldsymbol{x}(t_j)).$$

4.6 Experiments

A Universal Robots UR10[1] is used for the experiments. It is a collaborative industrial robot with 6-degrees of freedom. The interface with the UR10 is based on ROS (Robot Operating System, Quigley et al. (2009)), through the *ur_modern_driver*[2]. Data are acquired with a sampling time of $8 \cdot 10^{-3} sec$. The data processing and the derivation of the physical model are implemented in MATLAB, while the GPR in Python, to exploit the PyTorch computational advantages during the model optimization Paszke et al. (2017).

[1] www.universal-robots.com/UR10
[2] https://github.com/ThomasTimm/ur_modern_driver

The root mean squared error (RMSE) between $\boldsymbol{i}$ and $\hat{\boldsymbol{i}}$ has been considered in order to evaluate the algorithms accuracy. Given N samples we have

$$RMSE(X) = \sqrt{\frac{\sum_{k=1}^{N}(i_i(\boldsymbol{x}(t_k)) - \hat{i}_i(\boldsymbol{x}(t_k)))^2}{N}}.$$

We tested three GPR models, the PP-based estimator with linear features modeling kinetic frictions defined in (4.7), the SP-based estimator defined in (3.7), except for the PP component, that is defined as in (4.7), and finally the proposed approach defined in Section 4.5.3, hereafter denoted by FC.

We performed two experiments. The first one compare the accuracy of the aforementioned GP models in *quasi-static* and *dynamic* configurations. Instead, the second one validates the possibility of using the GPR models of the currents to detect collisions due to human-robot interaction.

4.6.1 Random exploration of the workspace

In this experiment, we compare estimation performance. We collected two datasets. The first, used as the training dataset, consists of a set of trajectories collected requiring to the end-effector to reach 200 random points (for a total of 80000 samples) randomly distributed within a hemisphere of the robot workspace. Instead, the second dataset is composed by 22000 samples collected requiring the robot to reach 50 random points inside the previous hemisphere, and to track a circle of radius $30[cm]$ at a tool speed of $30[mm/s]$. The hyperparameters of the three estimators have been trained by marginal likelihood maximization, see Section 1.3. As far as the practical derivation of the estimators, i.e., the computation of the posterior distribution, we computed (1.1) after downsampling the training dataset with constant step, obtaining 5000 samples.

In Figure (4.2) we reported the RMSEs obtained in the test dataset, distinguishing between *quasi-static* and *dynamic* configurations. Results show the effectiveness of the proposed learning strategy in improving the model accuracy in *quasi-static* configurations, given that, for all the joints, the FC's RMSE is lower than the one obtained with the other standard GPR models. Moreover, observe that the RMSEs of SP are higher than the ones obtained with PP, namely, the addition of the RBF kernel with standard inputs decreases the performance when $\dot{q} < \sigma_v$. This is probably due to the behavior showed in Figure 4.1, i.e., the presence of significantly different values of the target function associated with the same input location. This fact further underlines the importance of considering the augmented state introduced in this work.

As concerns performance when $|\dot{q}| > 0$, the RMSEs obtained with PP and SP are similar. This fact suggests that the parametric kernel proposed in (4.7) and the friction model in (4.5) are quite accurate when $|\dot{q}| > \sigma_v$. Finally, observe that joint 1 RMSE of the FC-based estimator is significantly smaller than that of the PP-based estimator. This fact highlights another interesting fact. The addition of the NP-based friction compensation can improve also the accuracy of the PP-based component.

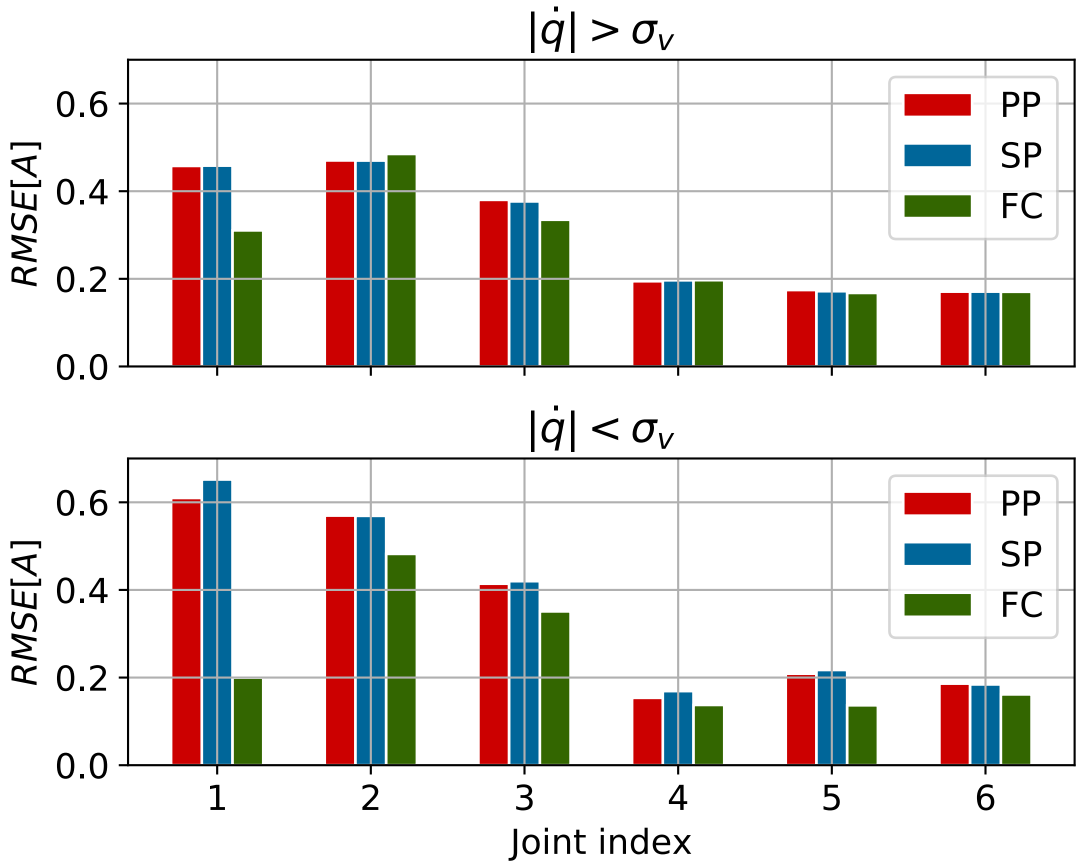

Figure 4.2: Comparison of the out of sample RMSEs obtained with the different GPR models; performance in *quasi-static* configurations are in the bottom graph, while *dynamic* configuration are in the top graph.

4.6.2 Detection of human-robot interaction

In order to validate the CD strategy proposed, we used the estimators derived in the previous experiment on a real test case: the detection of human-robot interaction. We tested the algorithms both in *dynamic* and *quasi-static* configurations. Due to the similarity of PP and SP performance, we reported only the SP performance. In the first part of the experiment, the end-effector of the robot is tracking a circle, while in the last

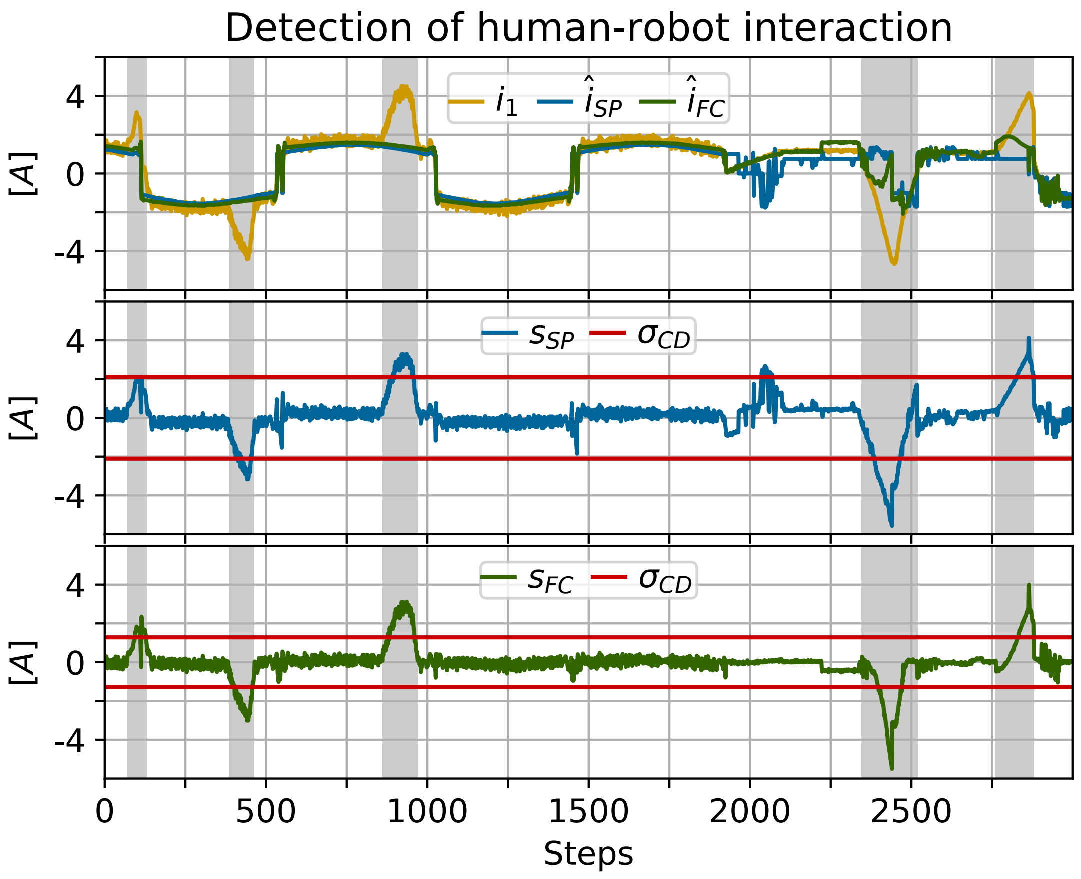

Figure 4.3: Detection of human-robot interaction. The grey bars highlight the intervals in which a human user apply a force to the first joint of the robot. In the top graph we represented the evolution of i_1, together with the estimates returned by SP and FC. In the middle graph (resp. bottom) we represented the monitoring signal obtained with the SP estimator (resp. FC estimator), together with the relative σ_{CD}.

part it stays in the final configuration. A human user applies an external force to the first robot joint five times, three during the moving phase, and two during the *quasi-static* phase[3]. Results are reported in Figure 4.3, where we plotted the evolution of i_1, together with the estimates obtained with SP and FC, the monitoring signals s_{SP} and s_{FC}, and the thresholds σ_{CD}; the σ_{CD} values have been set equal to the maximum value of the estimation error in the test set of the previous experiment. The gray bar highlights the time intervals in which the interactions occurred.

The results show that the monitoring signal derived with the FC estimator allows detecting all the interactions. Instead, using the monitoring signal and the σ_{CD} obtained with the SP estimator the first contact is not detected. This is mainly due to the higher

[3]The experiment is visible at https://youtu.be/2jJS8ajXhEw

value of σ_{CD}. With the proposed approach, we have $\sigma_{CD} = 1.28[A]$, while with the SP estimator we obtained $\sigma_{CD} = 2.10[A]$; for completeness we report also the value obtained with the PP estimator, that, as expected, is similar to the one of SP, and equal to $2.09[A]$.

4.7 Conclusions

In this work we validated the use of GPR to solve the proprioceptive collision detection problem, focusing on the definition of a good monitoring signal. The proposed approach has minimal requirements in terms of sensors since only joint coordinates and motor currents are needed. The proposed monitoring signal corresponds to the estimate of the currents due to external torques. In particular, we focused on *quasi-static* configurations, that are particularly relevant in collaborative robotics. To deal with the non-linear effects and the unmodeled behaviors due to static frictions, we modified the standard GPR SP-based estimators used in the robot inverse dynamics, considering an augmented set of input features, as well as designing an ad-hoc NP-component ables to model the discontinuity between static and kinetic frictions. The proposed approach has been tested in a UR10. The experimental results show the effectiveness of the proposed solution.

5

MC-PILCO: policy-search using Monte Carlo sampling

5.1 Introduction

Reinforcement Learning (RL) has seen explosive growth in recent years. RL algorithms have been able to reach and exceed human-level performance in several benchmark problems, such as playing chess, go and shogi Silver et al. (2018). Despite these remarkable results, the application of RL to real physical systems (e.g., robotic systems) is still a challenge, because of the large amount of experience required and the safety risks during exploration. To overcome these limitations, Model-Based RL (MBRL) techniques have been developed Deisenroth and Rasmussen (2011); Todorov and Li (2005); Levine and Abbeel (2014). Providing an explicit model of the physical system allows drastic decreases in the experience time required to converge to good solutions, while also reducing the risk of damage to the hardware during exploration and policy improvement.

MBRL methods are effective as much as their models resemble accurately the real systems. Hence, deterministic models suffer dramatically from this kind of issue, and the use of stochastic models becomes necessary to capture model uncertainty. Gaussian processes (GPs) Rasmussen and Williams (2006) are a class of Bayesian models commonly used in RL methods precisely for their intrinsic capability to handle uncertainty and provide principled stochastic predictions Rasmussen and Kuss (2004)Berkenkamp, Turchetta, Schoellig, and Krause (2017). GPs are not the only model structure that has been used to build uncertainty-aware models. Other possibilities are ensembles of probabilistic deep neural networks Chua, Calandra, McAllister, and Levine (2018)Kurutach, Clavera, Duan, Tamar, and Abbeel (2018).

In this chapter, we propose MC-PILCO, a Monte Carlo particle-based version of the algorithm PILCO (Probabilistic Inference for Learning Control) Deisenroth and Rasmussen (2011). PILCO is a model-based policy search algorithm that uses GP models to approximate the expected long-term cost, a function of the system state incorporating the task to be solved. The policy parameters are optimized relying on a gradient-basted strategy, with the aim of finding the policy that minimizes an assigned cost. PILCO manages to achieve substantial data-efficiency in solving different control problems Durrant-Whyte, Roy, and Abbeel (2012), but it comes with two main limitations due to the method used to generate long-term predictions. In PILCO predicted state distributions are approximated with a Gaussian distribution by moment matching. However, this kind of approximation allows modeling only unimodal distributions, which might be a too rough approximation of the real system behavior. Remarkably, the first and the second moments, and the policy gradient are computed in closed form. However, the computation of the moments is tractable only when considering the Radial Basis Function (RBF) kernel. This might not be the most convenient choice, being that this kind of kernel shows poor generalization properties.

In this work, we tackled these two limitations. Adopting a Monte Carlo sampling approach, we approximate the expected cost simulating the evolution of several state particles, based on the learned model. The policy gradient is obtained by backpropagation of the associated stochastic computational graph. This approach does not make any assumption on prediction probability distributions, and it does not require a specific type of kernel function, providing more flexibility in the design of the GP models.

Similar particle-based approaches have been tried before in the context of PILCO, obtaining poor performance. Mchutchon and College (2014) sees the cause in the presence of various local minima in the optimization process, while in Parmas, Rasmussen, Peters, and Doya (2018) poor performances are attributed to a hopelessly large gradient variance when using particles. In both cases, they still keep RBF kernels. On the contrary, the method we propose gives the possibility to choose any kind of kernel function. For many systems, the principal dynamic behaviors can be described by polynomial relations. Hence, in our method, we combine RBF with polynomial kernels to obtain a more effective representation of the system dynamics. Experimental results performed in a simulated environment proved the effectiveness of the solution, and the importance of using also structured kernels, like the polynomial kernel, to model the system.

This chapter is structured as follows. In Section 5.2 we state the general setting of model-based policy gradient methods. In Section 5.3 we present modeling approaches with GPs. In Section 5.4 we present MC-PILCO, our proposed algorithm. Finally, in Section 5.5 we present experimental results.

5.2 Model-Based Policy Gradient

In this section, we introduce the standard framework considered in MBRL algorithms. Consider the discrete-time system described by the unknown transition function f,

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w},$$

where $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ and $\boldsymbol{u}_t \in \mathbb{R}^{d_u}$ are, respectively, the state and the inputs at time step t, while $\boldsymbol{w} \sim \mathcal{N}(0, \Sigma_{\boldsymbol{w}})$ is a Gaussian random variable modeling system noise. It is also defined an instantaneous cost function $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$ that measures the penalty for being in state $\boldsymbol{x}_t$ and choosing input $\boldsymbol{u}_t$. The cost function must incorporate all the information needed to fulfill the desired task (for example, drive the system to a particular state). The objective is to find a deterministic control policy $\pi : \boldsymbol{x} \mapsto \boldsymbol{u} = \pi(\boldsymbol{x})$ that minimizes the expected long-term cost over a fixed finite time horizon N, i.e.,

$$J = \sum_{t=0}^{N} \mathbb{E}_{\boldsymbol{x}_t} \left[c(\boldsymbol{x}_t, \pi(\boldsymbol{x}_t)) \right], \tag{5.1}$$

with the initial state distributed according to $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$. We consider the control policy as a parametric function, and so we indicate it by $\pi_{\boldsymbol{\theta}}$, being $\boldsymbol{\theta}$ the parameters vector. Hence, the expected long-term cost will depend on $\boldsymbol{\theta}$ too, and we refer to it by $J(\boldsymbol{\theta})$ to make explicit this dependency.

A model-based approach for learning a policy that successfully controls the system consists, generally, in three main phases, which are repeated until the task is fulfilled:

- *Model Learning*: the data collected from all the previous interactions are used to build a model of the system dynamics (at the first iteration data are collected applying random exploratory controls to build an initial model);

- *Policy Update*: policy is updated in order to minimize $J(\boldsymbol{\theta})$ according to the current model;

- *Apply Policy*: the actual policy is applied to the system and the measures are stored for improving the model.

Model-based policy gradient methods exploit the learned model to predict the state evolution when applying current policy. These predictions are used to estimate $J(\boldsymbol{\theta})$ and its gradient $\nabla_{\boldsymbol{\theta}} J$, in order to update policy parameters $\boldsymbol{\theta}$ with a gradient-descent approach.

5.3 GPR and dynamics model

In this section, we describe the standard model learning framework based on Gaussian process regression considered in MBRL. Firstly, we describe the model for one-step-ahead prediction, as well as recalling background notions about GPR (see Section 1.3). Finally, we discuss long term predictions, focusing, in particular, on the strategy used in PILCO for propagating the uncertainty.

5.3.1 GPR and one-step-ahead predictions

The state of a mechanical system is typically defined as $\boldsymbol{x}_t = [\boldsymbol{q}_t^T, \dot{\boldsymbol{q}}_t^T]^T$, where $\boldsymbol{q}_t \in \mathbb{R}^{\frac{d_x}{2}}$ is the vector collecting the generalized coordinates of the system at time step t, and, as usual, $\dot{\boldsymbol{q}}_t$ represents the derivative of $\boldsymbol{q}_t$ w.r.t. time. A common strategy with GPR-based approaches consists in modeling the evolution of each state dimension with a distinct GP. Denote by $\Delta_t^{(i)} = x_{t+1}^{(i)} - x_t^{(i)}$ the difference between the value of the i-th component at time $t+1$ and t, and by $y_t^{(i)}$ the noisy measure of $\Delta_t^{(i)}$, with $i \in \{1, \dots, d_x\}$. Moreover, let $\tilde{\boldsymbol{x}}_t = [\boldsymbol{x}_t, \boldsymbol{u}_t]$ be the vector collecting the state and the input at time t. Then, given a vector of n output measures $\boldsymbol{y}_i = [y_{t_1}^{(i)}, \dots, y_{t_n}^{(i)}]^T$, and the set of relative inputs $\tilde{X} = \{\tilde{\boldsymbol{x}}_{t_1}, \dots, \tilde{\boldsymbol{x}}_{t_n}\}$, GPR assumes the following probabilistic model,

$$\boldsymbol{y}_i = \begin{bmatrix} \Delta_{t_1}^{(i)} \\ \vdots \\ \Delta_{t_n}^{(i)} \end{bmatrix} + \begin{bmatrix} e_{t_1}^{(i)} \\ \vdots \\ e_{t_n}^{(i)} \end{bmatrix} = \begin{bmatrix} h_i(\tilde{\boldsymbol{x}}_{t_1}) \\ \vdots \\ h_i(\tilde{\boldsymbol{x}}_{t_n}) \end{bmatrix} + \begin{bmatrix} e_{t_1}^{(i)} \\ \vdots \\ e_{t_n}^{(i)} \end{bmatrix} = \boldsymbol{h}_i(\tilde{X}) + \boldsymbol{e}_i,$$

where $\boldsymbol{e}^{(i)}$ is Gaussian i.i.d. noise with standard deviation σ_i, and $h_i(\tilde{\boldsymbol{x}}_t) = \Delta_t^{(i)}$ is a unknown function modeled a priori as a zero-mean Gaussian process. In particular, we have $\boldsymbol{h}_i \sim \mathcal{N}(0, K_i(\tilde{X}, \tilde{X}))$, with the covariance $K_i(\tilde{X}, \tilde{X})$ defined through a kernel function $k_i(\cdot, \cdot)$, namely, the element in j-th row and k-th column is given by $k_i(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_k)$. A crucial aspect of GPR is the kernel choice. The kernel function encodes prior assumptions about the process. A convenient option for modeling continuous functions is the Radial

Basis Function (RBF) kernel, used also in PILCO, and defined as

$$k_{RBF}(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_k) := \lambda^2 e^{-||\tilde{\boldsymbol{x}}_j - \tilde{\boldsymbol{x}}_k||^2_{\Lambda^{-1}}}, \tag{5.2}$$

where the scaling factor λ and the matrix Λ are kernel hyperparameters which can be estimated by marginal likelihood maximization. Typically, Λ is assumed to be diagonal, with the diagonal elements named length-scales.

Remarkably, as discussed in Section 1.3, the posterior distribution of $h_i(\cdot)$ can be computed in closed form. Let $\tilde{\boldsymbol{x}}_t$ be a general augmented state at time t. Then, the distribution of $\hat{\Delta}_t^{(i)}$, the estimate of $\Delta_t^{(i)}$, is Gaussian, with mean and variance given by

$$\mathbb{E}[\hat{\Delta}_t^{(i)}] = k_i(\tilde{\boldsymbol{x}}_t, \tilde{X})\Gamma_i^{-1}\boldsymbol{y}_i, \tag{5.3}$$

$$var[\hat{\Delta}_t^{(i)}] = k_i(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_t) - k_i\Gamma_i^{-1}k_i^T(\tilde{\boldsymbol{x}}_t, \tilde{X}), \tag{5.4}$$

with Γ_i and $k_i(\tilde{\boldsymbol{x}}_t, \tilde{X})$ defined as

$$\Gamma_i = (K_i(\tilde{X}, \tilde{X}) + \sigma_i^2 I),$$
$$k_i(\tilde{\boldsymbol{x}}_t, \tilde{X}) = [k_i(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t_1}), \dots, k_i(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t_n})].$$

Consequently, we have the following posterior distribution for the estimated state at time $t+1$

$$p(\hat{\boldsymbol{x}}_{t+1}|\tilde{\boldsymbol{x}}_t, \tilde{X}, \boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}), \tag{5.5}$$

where

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{x}_t + \left[\mathbb{E}[\hat{\Delta}_t^{(1)}] \dots \mathbb{E}[\hat{\Delta}_t^{(d_x)}]\right]^T, \tag{5.6}$$

$$\Sigma_{t+1} = blkdiag\left(\left[var[\hat{\Delta}_t^{(0)}] \dots var[\hat{\Delta}_t^{(d_x)}]\right]\right). \tag{5.7}$$

5.3.2 Long-term predictions: Moment Matching

In MBRL the policy $\pi_{\boldsymbol{\theta}}$ is evaluated and improved based on $p(\hat{\boldsymbol{x}}_1) \dots p(\hat{\boldsymbol{x}}_N)$. To compute $p(\hat{\boldsymbol{x}}_1)$ we have to marginalize (5.5) over the initial state distribution $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$, and then iterate the procedure to obtain the subsequent distributions. However, the exact computation of the resulting integrals is not tractable. In PILCO, the predicted state distributions are approximated relying on moment matching. Specifically, assuming as kernel function the RBF kernel, the first and the second moment of $p(\hat{\boldsymbol{x}}_1)$ are computed in closed form. Then the $p(\hat{\boldsymbol{x}}_1)$ distribution is approximated with a Gaussian distribution, with mean and variance equal to the moments computed previously. Finally, the subsequent probability distributions are computed iterating the procedure. For details, we refer the reader to Deisenroth, Fox, and Rasmussen (2015).

We conclude this section by discussing two possible limitations of this approach. Firstly, to make the computation of the first and second moment tractable, the authors assumed to use only the RBF kernel. It is well known that such kernel can approximate accurately the target function only in a neighborhood of the training input locations; poor generalization properties might slow down significantly the learning process. Secondly, approximating each $p(\hat{\boldsymbol{x}}_t)$ with a Gaussian distribution allows modeling only unimodal distributions. This might be a too rough approximation of the real system behaviors.

5.4 MC-PILCO

In this section, we present the proposed approach. As far as model learning is concerned, we rely on GPR. However, compared to PILCO, we consider a different strategy to approximate the distribution of long-term predictions. In our algorithm, the expected value of the long-term cost is approximated using particles-based methods. Besides avoiding behaviors due to the unimodality of Gaussian approximation, this strategy allows using any kind of kernel, providing more freedom as concerns modeling.

MC-PILCO is summed up in pseudo-code in Algorithm 1, and consists in the iteration of three main steps, namely, update the GP models, update the policy parameters, and apply the policy. In its turn, the policy update is composed of three steps, iterated N_{opt} times:

- simulate the evolution of M particles, based on the current $\pi_{\boldsymbol{\theta}}$ and on the GP models learned from the previously observed data;
- compute $\hat{J}(\boldsymbol{\theta})$, an approximation of the expected-long-term cost, based on the evolution of the M particles;
- update the policy parameters $\boldsymbol{\theta}$ based on $\nabla_{\boldsymbol{\theta}}\hat{J}(\boldsymbol{\theta})$, the gradient of $\hat{J}(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$, computed by backpropagation.

In the next two subsections, we describe the particles-based strategy adopted to approximate $J(\boldsymbol{\theta})$, and we discuss the technique used to compute $\nabla_{\boldsymbol{\theta}}\hat{J}(\boldsymbol{\theta})$. Finally, in the last subsection, we describe the GP models considered in our algorithm.

5.4.1 Particles-based Policy Evaluation

We start describing the particles based strategy adopted to approximate the expected long-term-cost. We assume that a one-step-ahead prediction model of the type described

Algorithm 1: MC-PILCO

```
init policy $\pi_{\boldsymbol{\theta}}(\cdot)$, cost $c(\cdot)$, kernel $k(\cdot,\cdot)$, policy optimization steps $N_{opt}$, number of
  particles $M$
Apply random control to system and collect data
while task not learned do
    Learn GP model from sampled data;
    for $i = 1...N_{opt}$ do
        predict $M$ particles rollouts from GP model when applying current policy;
        compute $\hat{J}(\boldsymbol{\theta})$ from particles (5.8);
        compute $\nabla_{\boldsymbol{\theta}}\hat{J}$ through backpropagation;
        gradient-based policy update (e.g., using Adam);
    end
    apply updated policy to system and collect data
end
return final policy $\pi_{\boldsymbol{\theta}^*}(\cdot)$, learned GP model;
```

in Section 5.3 is available. Namely, given a current state $\boldsymbol{x}_t$ and the input selected by the policy $\boldsymbol{u}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$, $p(\hat{\boldsymbol{x}}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$, the distribution of the predicted state at the next time-step is Gaussian, with the mean and the variance defined by (5.6) and (5.7). No assumptions about the kernel function used are necessary.

Monte Carlo sampling Caflisch (1998) provides a simple method to approximate the integrals in (5.1). The basic idea consists in simulating the evolution of a sufficient number of particles, and approximating each $\mathbb{E}_{\boldsymbol{x}_t}[c(\boldsymbol{x}_t, \pi(\boldsymbol{x}_t))]$ with the sample mean of the costs associated to the states and the inputs assumed by the particles at time t. Specifically, we sample M particles from the initial state distribution $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$. Each one of the M particles is propagated using the GP dynamics (5.5). Let $\boldsymbol{x}_t^{(m)}$ be the state of the m-th particle at time t, with $m = 1, \dots, M$. At time step t, the actual policy $\pi_{\boldsymbol{\theta}}$ is evaluated to compute the associated control. The GP model provides the Gaussian distribution $p(\boldsymbol{x}_{t+1}^{(m)}|\boldsymbol{x}_t^{(m)}, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(m)}))$ from which $\boldsymbol{x}_{t+1}^{(m)}$, the state of the particle at the next time step, is sampled. This process is iterated until N samples are generated for each particle. The process is illustrated in Figure 5.1 for sake of clarity. Finally, the estimate of the expected long-term cost is computed with the following expression,

$$\hat{J}(\boldsymbol{\theta}) = \sum_{t=0}^{N} \left(\frac{1}{M} \sum_{m=1}^{M} c(\boldsymbol{x}_t^{(m)}, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(m)})) \right) . \tag{5.8}$$

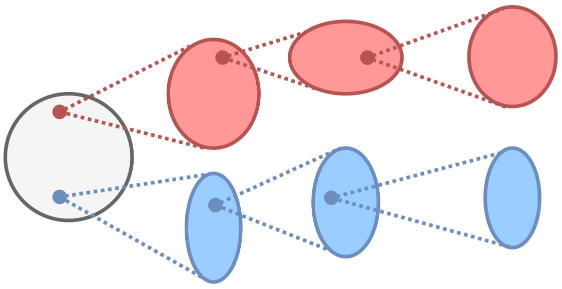

Figure 5.1: Example of two particles propagating through the stochastic model (Gaussian distributions represented as ellipses).

5.4.2 Policy Gradient through Backpropagation

To compute the gradient of (5.8) w.r.t. the policy parameters we rely on techniques based on stochastic computational graphs. We used the reparameterization trick to propagate the gradient through the stochastic nodes; for details we refer the reader to Kingma and Welling (2013). Then, $\nabla_{\boldsymbol{\theta}}\hat{J}$ is computed simply applying the chain rule.

5.4.3 Proposed GP models

In this subsection, we describe a general framework that we propose to model the evolution of mechanical systems. Following ideas presented in Romeres, Jha, Dalla Libera, Yerazunis, and Nikovski (2019), instead of modeling the evolution of each state dimension independently, we halved the number of GPs to be learned, exploiting the intrinsic correlations between the state components $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. Indeed, when considering a sufficiently small sampling time T_s (small w.r.t. the application), it is reasonable assuming constant accelerations between two consecutive time-steps, obtaining the following evolution of $\boldsymbol{q}_t$,

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + T_s\dot{\boldsymbol{q}}_t + \frac{T_s}{2}(\dot{\boldsymbol{q}}_{t+1} - \dot{\boldsymbol{q}}_t). \tag{5.9}$$

Let $\mathcal{I}_{\boldsymbol{q}}$ (respectively $\mathcal{I}_{\dot{\boldsymbol{q}}}$) be the ordered set of the state dimensions indices associated to $\boldsymbol{q}$ (respectively $\dot{\boldsymbol{q}}$). In our framework, we learn only $d_x/2$ GPs, each of which models the evolution of a distinct $\Delta_t^{(i_k)}$, with $i_k \in \mathcal{I}_{\dot{\boldsymbol{q}}}$. The predicted change in velocity, $\Delta_t^{(i_k)}$, will be sampled from GPs, while the position change is computed according to (5.9) and the sampled $\Delta_t^{(i_k)}$. Besides limiting the number of GPs to be learned, this strategy allows considering correlations between $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$.

One of the advantages of the policy update framework described in Section 5.4.1-5.4.2 is that it is independent of the structure of the GP models. Consequently, w.r.t. PILCO, several different options can be considered to model the evolution of the systems, for example, adopting more kernel structures, e.g., the polynomial kernel, or model-based kernels, like the ones proposed in Romeres et al. (2019); for the sake of generality, we do not rely on model-based kernels in this work. In the proposed algorithm, the kernel of each GP is given by the sum of two kernels. The first is an RBF kernel of the type defined in (5.2), while the second is a Multiplicative Polynomial Kernel (MP kernel), already introduced in Chapter 3, and further discussed in Chapter 6. For sake of clarity, we recall the definition of the MP kernel of degree d,

$$k_{MP}^{(d)}(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_k) := \prod_{r=1}^{d} \left(\sigma_{MP_r}^2 + \tilde{\boldsymbol{x}}_j^T \Sigma_{MP_r} \tilde{\boldsymbol{x}}_k \right),$$

where the $\Sigma_{MP_r} \geq 0$ matrices are distinct diagonal matrices. The diagonal elements of the Σ_{MP_r}, together with the $\sigma_{MP_r}^2$ elements are the kernel hyperparameters. Then, we have that each k_{i_k} is defined by the following expression

$$k_{i_k}(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_k) = k_{RBF}(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_k) + k_{MP}^{(d)}(\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_k). \tag{5.10}$$

The idea motivating the aforementioned kernel is the following. MP kernel allows capturing eventual modes of the system that are polynomial functions in $\tilde{\boldsymbol{x}}$, while the RBF kernel models more complex behaviors not captured by the polynomial kernel.

5.5 Experimental Results

We tested the proposed algorithm in a simulated cartpole swing-up task, a standard benchmark for control and RL. MC-PILCO is compared with the original PILCO algorithm. For this reason, we kept the same system settings indicated in the PILCO code[1]. We implemented MC-PILCO in python, exploiting the PyTorch functionalities to compute gradients, while we run PILCO with the original MATLAB code. This section is structured in two parts, the first compares the policy learning performance of the two algorithms, while the second regards an analysis of the modeling results.

5.5.1 Cartpole Swing-up Policy Learning

The objective of the task is to complete a swing-up of the system: bring and balance the pendulum in the vertical upright position. The cartpole state is $\boldsymbol{x} := [q_1, \dot{q}_1, q_2, \dot{q}_2]$, where q_1 is the position of the cart along the horizontal axis, and q_2 is the angle of the pendulum.

[1]Code available at `http://mlg.eng.cam.ac.uk/pilco/`

We have that $q_2 = 0$ when the pendulum is hanging downwards, and $q_2 = \pi$ when it is upright. The system input is the horizontal force applied to the cart, the maximum applicable force is 10 [N]. The control horizon considered is 3 [s], with a sampling time of 0.1 [s]. As far as the policy is concerned, we adopted a single-layered RBF network of size 100 as control policy. The policy $\pi_{\boldsymbol{\theta}}$ takes as input $\boldsymbol{x}_{ext} := [q_1, \dot{q}_1, \dot{q}_2, cos(q_2), sin(q_2)]$, where the *cos* and *sin* transformations are considered due to the periodicity of the angles. Policy output is limited in the range [-10,10] [N] using an hyperbolic tangent function, properly scaled, i.e., $u(\boldsymbol{x}_{ext}) = 10 \cdot tanh(\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_{ext}))$. The cost function is given by

$$c(\boldsymbol{x}) = 1 - e^{-\left(\frac{q_1}{l_1}\right)^2 - \left(\frac{|q_2|-\pi}{l_2}\right)^2}.$$

Cost values are bounded between 1 and 0, and $c = 0$ when the pendulum is in equilibrium at $q_2 = \pi$ (or $q_2 = -\pi$) and the cart is positioned in $q_1 = 0$. The cost function depends upon two parameters, fixed as $l_1 = 1.5$ and $l_2 = 1$. We considered $M = 100$ particles to approximate the expected cost, and we updated the policy using Adam Kingma and Ba (2015). The learning rate is decreased gradually as the model becomes more accurate with new data from experience, in this way, during the first iterations we have more exploration in the policy parameters, while towards the end we learn more finely the optimal policy. On the other hand, the number of optimization steps increases with the amount of experience; at the beginning, the model is not trustworthy, and there is no reason to heavily optimize the policy w.r.t. its predictions.

The performance of the algorithms is measured by considering the evolution of two indicators as functions of the experienced time. First, we consider the amount of time in which the pole is balanced in the vertical upright position, considering a tolerance of 10 degrees. Second, we consider the average distance of the cart from the center position in the horizontal axis. Experiments were repeated for 30 different random seeds. The results comparing the proposed MC-PILCO with PILCO are reported in Figure 5.2. After 4 interactions (a total experience of 12 [s]) MC-PILCO manages to balance the pendulum upright for an average time of 1.69 [s], while PILCO for 1,45 [s]. Both algorithms reach a similar average distance of the cart from zero, 0.16 [m] for MC-PILCO, and 0.19 [m] for PILCO (detailed results in Figure 5.2).

We consider a trial successful when the policy maintains the pendulum in the upright for at least the last 1 [s]. In Table 5.1 the evolution of the success rates is reported. MC-PILCO reaches a success rate of 88.5% outperforming PILCO which obtains 76.7%. Notice also the considerable discrepancies between the success rate obtained after 6 [s]. Probably due to the greater data-efficiency of the kernels used by MC-PILCO, its success rate is twice the PILCO success rate. Finally, in Figure 5.3 we show the trajectory obtained with a successful policy learned by MC-PILCO after 12[s] of experience, together with the predicted particles evolution.

We tested the proposed algorithm also using the cartpole environment of MuJoCo, Todorov, Erez, and Tassa (2012). Results obtained are similar to ones obtained in the PILCO environment. The video of the experiment is publicly available[2].

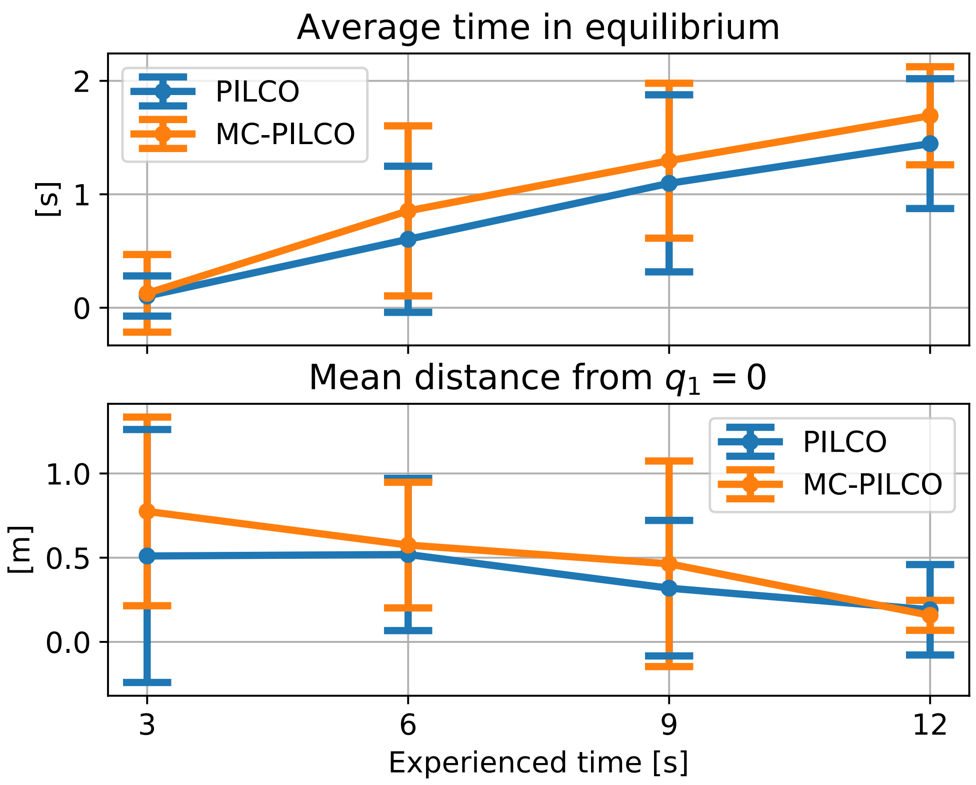

Figure 5.2: Performance observed w.r.t. the amount of interaction time with the cartpole system. Mean results are plotted at each trial (each one long 3 [s]), together with the standard deviation confidence intervals. The top graph represents the equilibrium time, while the bottom graph represents the average distance from $q_1 = 0$. Numerical values are reported in Table 5.2 and 5.3.

5.5.2 Cartpole Model Learning

The cartpole system is now used to compare the models used by the two algorithms. We trained the GP models of PILCO and MC-PILCO with the same training data, composed of 3 [s] of random exploration. We compare the respective long-terms predictions under

[2]https://youtu.be/r74tCtVOWGc

Success Rate *(%)*	*3 s*	*6 s*	*9 s*	*12 s*
MC-PILCO	3.8%	38.5%	69.2%	88.5%
PILCO	0.0%	20.0%	63.3%	76.7%

Table 5.1: MC-PILCO and PILCO success rates w.r.t. experienced time.

Time Up *(s)*	*3 s*	*6 s*	*9 s*	*12 s*
MC-PILCO	0.13 s	0.85 s	1.30 s	1.69 s
PILCO	0.10 s	0.60 s	1.10 s	1.45 s

Table 5.2: MC-PILCO and PILCO mean time in vertical upright position w.r.t. experienced time.

Distance *(m)*	*3 s*	*6 s*	*9 s*	*12 s*
MC-PILCO	0.77 m	0.57 m	0.46 m	0.16 m
PILCO	0.51 m	0.52 m	0.32 m	0.19 m

Table 5.3: MC-PILCO and PILCO mean distance of the cart from the center position w.r.t. experienced time.

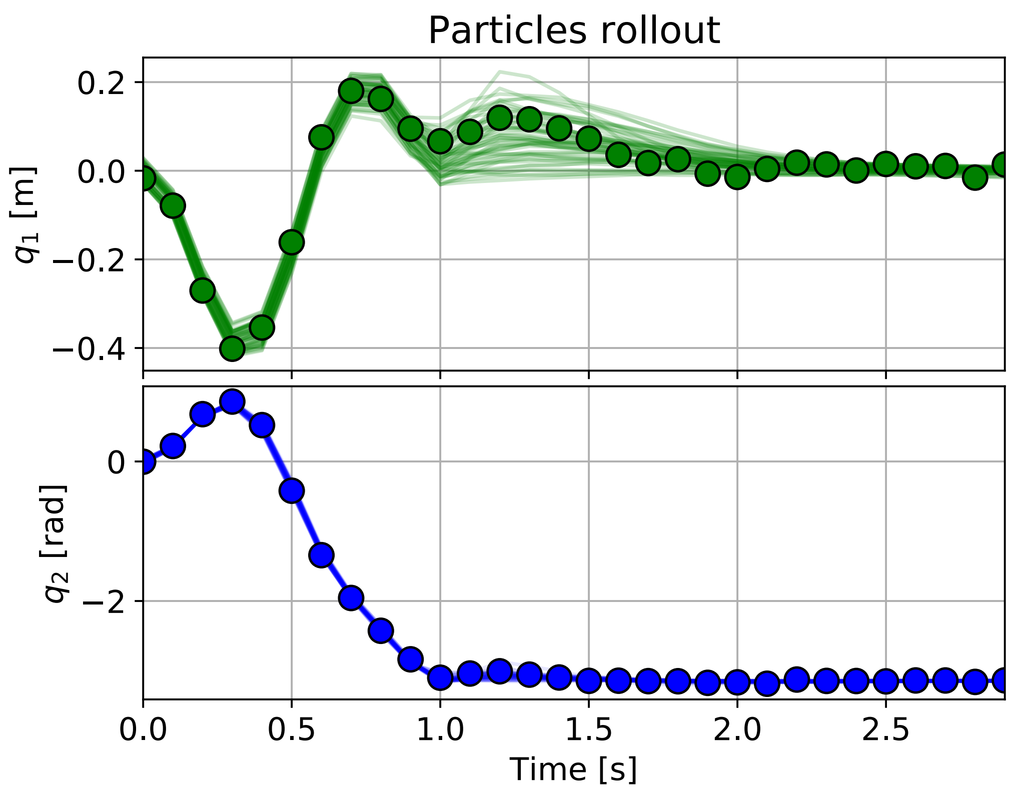

Figure 5.3: Cart position q_1 (top) and pendulum angle q_2 (bottom) particles' rollouts predicted by the model. Dots represent the real trajectories.

the same control policy. The results are showed in Figure 5.4. MC-PILCO particles are very accurate, being that some particles manage to follow the true trajectory for almost all the three seconds, only with the few exploration data at disposal. On the other hand, predicted state distribution used by PILCO is accurate enough until approximately 1.5 [s]. Later on, its variance increases drastically. This is due to the Gaussian unimodal distribution that PILCO assumes to obtain long-term prediction via moment matching.

Finally, we investigate the effect of choosing different kernels in the model learning framework described in Section 5.4.3. We compared the performance obtained using the RBF kernel, and the proposed kernel in (5.10). Models are trained with the same training data, and the predicted particles are compared with the same policy (in this case a learned policy that correctly executes the swing-up). Results are presented in Figure 5.5. The model adopting the combination of MP and RBF kernels outperforms the one using only RBF kernel. Some particles follow the real trajectory for almost the whole 3 [s] horizon, and no particle predicted by the RBF GP model follows the real trajectory for more than 1 [s]. Furthermore, not accurate particles produced by the GP with MP and RBF kernel do not diverge from the real trajectory as much as the particles obtained by the model with the RBF kernel. This difference in modeling accuracy is summed up by the root mean squared error (RMSE) of the two models. The RMSE obtained with the kernel in (5.10) is lower than the one obtained using only the RBF kernel, almost an order of magnitude along the whole prediction horizon. The results obtained stress once again the critical importance of the kernel choice, and its effects on accurate modeling of the dynamics, that inevitably affect the performance of the policy learning process.

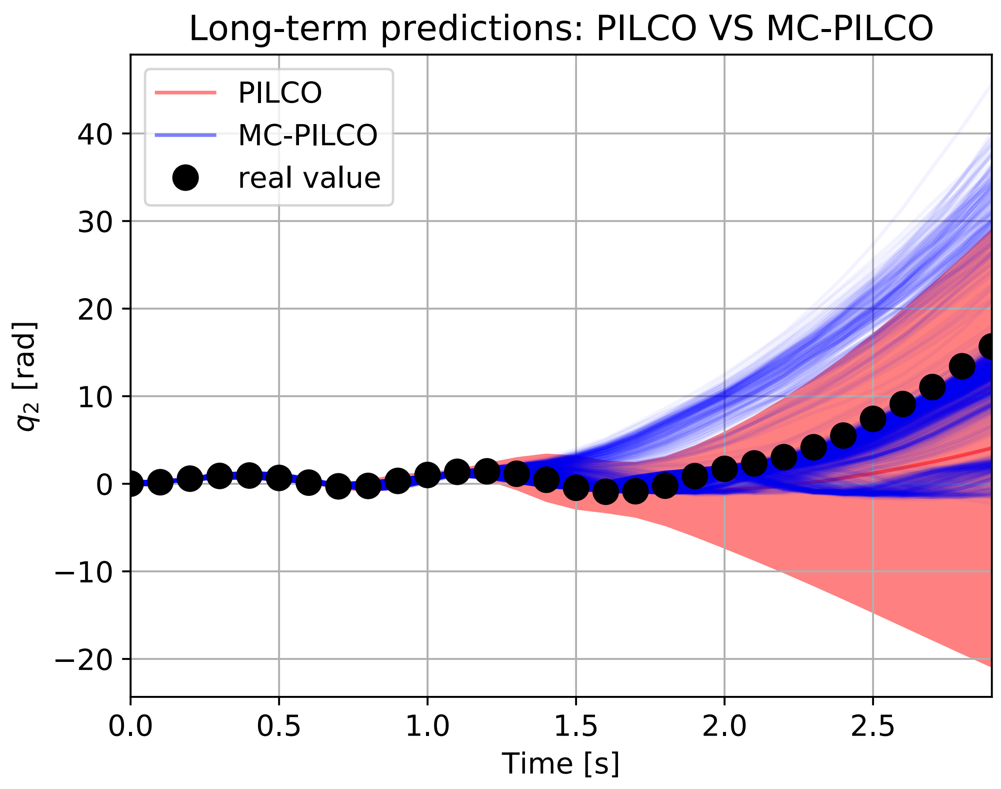

Figure 5.4: PILCO and MC-PILCO (with MP+RBF kernel) predictions of the q_2 (pendulum angle) trajectory. For PILCO we indicated the 99% confidence interval with the red shaded area. MC-PILCO particles are plotted in blue.

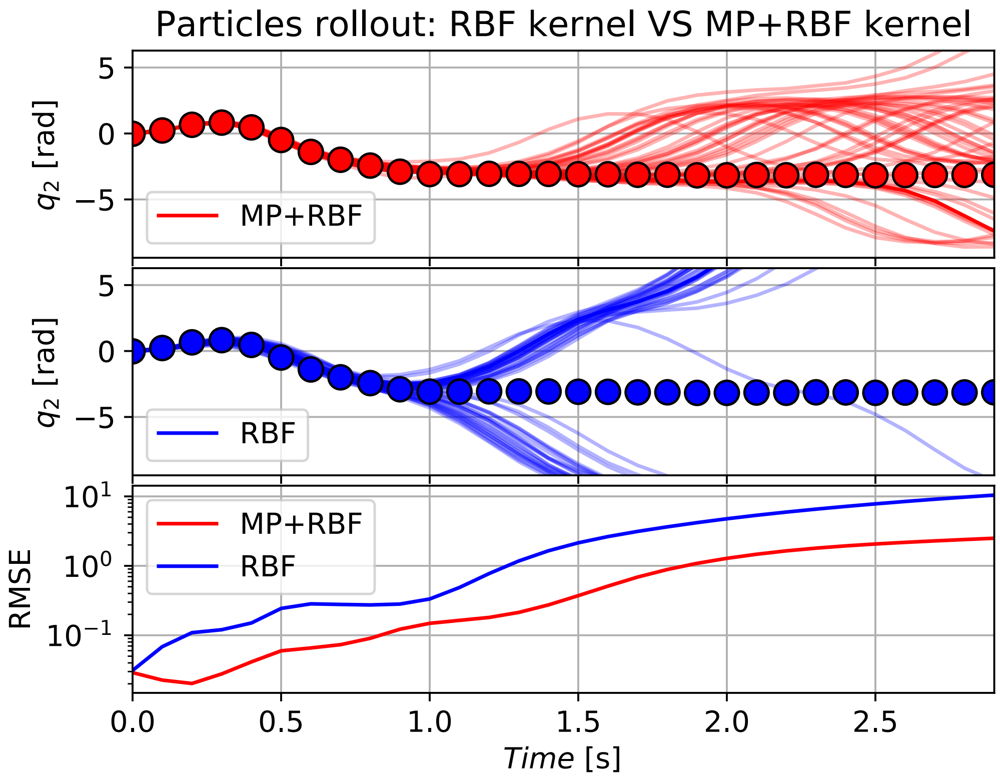

Figure 5.5: Comparison of the particles predicted by a GP model with a MP+RBF kernel, and one with a RBF kernel. Dots represents the real angle trajectory. The evolution of the RMSEs is shown in the bottom plot.

5.6 Conclusions

In this chapter, we proposed MC-PILCO, a particle-based version of PILCO, that requires neither Gaussian approximations in the computation of the policy gradient nor a mandatory RBF kernel for the GPs. The possibility of choosing different kernel functions provides great flexibility to the algorithm. We adopted a combination of MP and RBF functions as kernel function. This choice enhanced considerably the modeling capacity of GP dynamic models. This crucial aspect allows MC-PILCO to obtain better performance than PILCO. As future works, we aim to test MC-PILCO on more complex environments, to explore its capacity to handle higher dimensional states. Besides, we would like to apply it to the control of a real system.

6 Multiplicative Polynomial Kernel and Volterra Series

6.1 Introduction

In many real applications, linear models cannot adequately describe dynamic systems. This can be due to the presence of saturations, quantizers or static nonlinearities at the input and/or the output Ljung (1999)[Section 5]. Even if some insight on the nonlinearities can be available, the formulation of parametric models from finite data records is a difficult task Haber and Unbehauen (1990); Lind and Ljung (2008). In particular, nonlinear system identification is often seen as an extended parametric regression where the choice of regressors and basis functions plays a crucial role.

In this context, Volterra series are especially useful since they can represent a broad range of nonlinear systems Rugh (1980). In discrete-time, Volterra series are connected with Taylor expansions of the input-output map. Considering systems with finite memory (current output depends on a finite number m of past inputs and outputs), the r-th order Volterra series is represented as the convolution between the Volterra maps[1] and all the possible monomials up to order r (function of past inputs and outputs). Identification

[1]These are typically called Volterra kernels in the literature but we adopt this terminology to avoid confusion with the concept of kernels taken from machine learning and introduced later on.

is difficult due to the curse of dimensionality: the number of monomials grows quickly w.r.t. the adopted polynomial degree r and system memory m. Hence, a careful selection of the relevant components to be included in the model is crucial to control complexity, a problem known as regressor selection.

Suboptimal solutions are often adopted, e.g., greedy approaches like forward orthogonal least squares Chen, Billings, and Luo (1989); Billings, Chen, and Korenberg (1989) and its many variants Hong, Mitchell, Chen, Harris, Li, and Irwin (2008)[Section 3]. Another approach uses variance analysis (ANOVA) Lind and Ljung (2005). These regressor selection methods have however difficulties in handling high-dimensional spaces, as e.g., illustrated in Lind and Ljung (2008) where the divide-and-conquer method TILIA is introduced to mitigate this problem. An interesting option is joint estimation and variable selection whose aim is to automatically set to zero groups of variables in the regression vector. This can be performed exploiting the ℓ_1-norm that leads to LASSO Tibshirani (1996), also implementable using LARS Efron, Hastie, Johnstone, and Tibshirani (2004), a less greedy version of classical forward selection.

Alternative solutions rely on kernel-based regularization Schölkopf and Smola (2001). Positive definite kernels are especially important since they implicitly include a large (possibly infinite) number of basis functions. They also define particular spaces, the so called Reproducing Kernel Hilbert spaces (RKHS), and the related norms can be exploited to control complexity. In particular, regularized least squares, also called regularization networks in Poggio and Girosi (1990), determine the unknown system by minimizing an objective sum of two terms. The first is a quadratic loss, accounting for data fit, and the second is a regularization term given by the RKHS squared norm. The balance between these two contributions is given by the regularization parameter, typically estimated from data through Marginal likelihood maximization or cross validation Hastie, Tibshirani, and Friedman (2001); Rasmussen and Williams (2006); Pillonetto and Chiuso (2015).

The crucial aspect in the design of a regularization network is the kernel choice. In system identification the Gaussian kernel is often used to embed just expected smoothness of the input-output map, see e.g., Espinoza, Suykens, and De Moor (2005); Li, Li, Su, and Chun (2006); Xu and Chen (2009) and also Frigola, Lindsten, Schon, and Rasmussen (2013); Frigola and Rasmussen (2013); Hall, Rasmussen, and Maciejowski (2012) for state-space approaches. However, this kernel has some limitations in system identification. In those regions of the regressor space where few data are collected, output prediction just decays to zero. In this chapter, we instead focus on more structured kernels connected with Volterra series. They are discussed in Franz and Schölkopf (2006) using the polynomial kernel, which encodes all the monomials up to the desired degree r. Such implicit representation makes computationally feasible the handling of high-order Volterra series. However, the associated regularization networks have some drawbacks and can overfit the data as stated in Rasmussen and Williams (2006) (chapter 4.2.2). The reason is that the norm associated with the polynomial kernel is sometimes not able to well control the high number of monomials (implicitly) introduced in the estimation process.

More recent work on regularized Volterra series can be found in Birpoutsoukis, Marconato, Lataire, and Schoukens (2017a). Here, authors do not use kernels to encode monomials but focus on how to control the variance of the monomial coefficients via regularization. This is performed extending some ideas developed for linear system identification in Pillonetto and De Nicolao (2010), embedding in a Volterra map smooth-exponential decay concepts, i.e., the fact that, as the lag increases, inputs should have less effect on the output. In this way, the system memory m is replaced by continuous hyperparameters that are connected with memory fading concepts and can be estimated from data. Despite the remarkable results on both simulated and real data Birpoutsoukis, Csurcsia, and Schoukens (2017b), one limitation is that computational and memory requirements increase quickly with the polynomial degree r and the memory m.

The approaches proposed in Franz and Schölkopf (2006) and Birpoutsoukis et al. (2017a) are in some sense complementary. The first one uses the kernel to handle efficiently a large number of monomials but regularization is not always so effective. The second one refines this aspect but the price to pay is that implicit encoding of monomials is absent. In this work, we propose new techniques that overcome such dichotomy. To obtain this, two new kernels are introduced.
The first new kernel, which we call *Multiplicative Polynomial Kernel* (MPK), builds upon the polynomial kernel. For Volterra series of order r, it consists of the product of r basic building blocks. Each block consists of a linear kernel containing hyperparameters that permit monomial selection. This is a fundamental novelty w.r.t. the polynomial kernel that does not promote any sparsity, possibly returning solutions too rich of monomials. Hence, we will see that MPK allows a finer regularization, discovering those system parts that really influence the output and improving prediction capability.
Then, inspired by Birpoutsoukis et al. (2017a), we will show that MPK features can be further improved. Similarly to MPK, the second new kernel is the product of r linear kernels but with different hyperparameters able to model smooth exponential decay of Volterra coefficients. For this reason, it is called *Smooth Exponentially Decaying MPK* (SED-MPK). Hence, SED-MPK combines the nice features of Birpoutsoukis et al. (2017a) with implicit kernel encoding: handling of high-order Volterra series is so made possible.

The chapter is organized as follows. In Section 6.2 we provide background notions about discrete-time Volterra series, and we describe some solutions proposed to identify such models. We start Section 6.3 analyzing the regularization properties of standard polynomial kernel, highlighting possible limitations. Then, we introduce the MPK, discussing the advantages due to the MPK parametrization. In Section 6.4, we describe the SE-MPK, providing insights about the regularization induced by the SE-MPK parameters w.r.t. the coefficients of the Volterra map. Finally, in Section 6.5 we report experimental results.

6.2 Background

6.2.1 Volterra series

Consider a discrete time system, and let z_k be its output at time k. Assume that the system has finite memory m, and let $\boldsymbol{u}_k = [u_k, \ldots, u_{k-m}]$ be the vector containing the lagged inputs influencing the system response at time k. When modeling the system response with a truncated Volterra series of order r, the noisy output y_k is defined by the sum of $r+1$ Volterra functions and the measurement noise $e_k \sim N\left(0, \sigma_n^2\right)$. Specifically, one has

$$y_k = z_k + e_k = h_0 + \sum_{i=1}^{r} H_i(\boldsymbol{u}_k) + e_k \ , \tag{6.1}$$

where h_0 represents the zero-order Volterra function, constant and independent of the inputs, while $H_i(\boldsymbol{u}_k)$ are the higher-order contributions. Each $H_i(\boldsymbol{u}_k)$ is the convolution between a Volterra map h_i and all the possible monomials of degree i built with the components of $\boldsymbol{u}_k$. The Volterra map h_i is function of i variables $\{\tau_j\}_{j=1}^{i}$ that may assume values $\{0, 1, \ldots, m\}$ and represent input lags. In other words, they select (possibly repeated) components from $\boldsymbol{u}_k$ to define a monomial. Then, the expression of $H_i(\boldsymbol{u}_k)$ is

$$H_i\left(\boldsymbol{u}_k\right) = \sum_{\tau_1=0}^{m} \cdots \sum_{\tau_i=0}^{m} h_i\left(\tau_1, \ldots, \tau_i\right) \prod_{\tau=\tau_1}^{\tau_i} u_{k-\tau} \ . \tag{6.2}$$

For example, let $m = 2$ so that $\boldsymbol{u}_k = [u_k \ u_{k-1} \ u_{k-2}]$, with $i = 4$ and $\tau_1 = 0, \tau_2 = 0, \tau_3 = 1, \tau_4 = 2$. Note that τ_1 and τ_2 both select u_k while τ_3 and τ_4 choose, respectively, u_{k-1} and u_{k-2}. Then $h_4\left(0, 0, 1, 2\right)$ is a coefficient that multiplies the monomial $u_k^2 u_{k-1} u_{k-2}$.

Commonly, Volterra maps are assumed to be symmetric with respect to the input lags. Given a set of i input lags, the h_i value is equal for all the possible permutation of the lags. For instance, coming back to the previous example, under symmetry assumptions one has $h_4\left(0, 0, 1, 2\right) = h_4\left(0, 1, 0, 2\right) = h_4\left(1, 0, 2, 0\right) = \ldots$.

Alternatively, (6.2) can be rewritten more compactly with an inner product. Let

$\boldsymbol{\phi}_i(\boldsymbol{u}_k)$ be the vector collecting all the distinct monomials in $\boldsymbol{u}_k$ with degree i, and $\boldsymbol{w}_i$ be the vector function of the coefficients h_i, multiplied by opportune constants to account for the repetitions due to symmetry. Specifically, consider the generic monomial $\prod_{j=0}^{m} u_{k-j}^{d_j}$, where $\boldsymbol{d} = [d_0, \dots, d_m]$ are the relative degrees of each variable, and $\sum_{j=0}^{m} d_j = i$. Then, the multiplicative constant cited above is equal to the multinomial coefficient $\binom{i}{d_0, \dots, d_m}$, hereafter denoted just by $\binom{i}{\boldsymbol{d}}$. For an opportune permutation of the h_i and $\boldsymbol{w}_i$ elements, one then has

$$H_i(\boldsymbol{u}_k) = \boldsymbol{\phi}_i^T(\boldsymbol{u}_k)\boldsymbol{w}_i. \tag{6.3}$$

6.2.2 Volterra maps identification via regularized least squares

Combining (6.1) and (6.3), we have

$$y_k = \boldsymbol{\phi}^T(\boldsymbol{u}_k)\boldsymbol{w} + e_k,$$

with

$$\boldsymbol{\phi}^T(\boldsymbol{u}_k) = \left[1, \boldsymbol{\phi}_1^T(\boldsymbol{u}_k), \dots \boldsymbol{\phi}_r^T(\boldsymbol{u}_k)\right] \in \mathbb{R}^n, \tag{6.4a}$$

$$\boldsymbol{w}^T = \left[h_0, \boldsymbol{w}_1^T, \dots, \boldsymbol{w}_r^T\right] \in \mathbb{R}^n, \tag{6.4b}$$

where $n = 1 + \sum_{i=1}^{r} n_i$, with n_i equals to the dimension of $\boldsymbol{w}_i$. System identification then reduces to obtaining an estimate $\hat{\boldsymbol{w}}$ of $\boldsymbol{w}$ from the data set

$$D = \{(y_k, \boldsymbol{u}_k),\ k = 1\,,\ \dots\,,\ N\}\,. \tag{6.5}$$

For instance, one can use least squares:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\|\boldsymbol{y} - \Phi^T\boldsymbol{w}\right\|,$$

where $\boldsymbol{y} = [y_1, \dots y_N]^T$, and the regression matrix Φ^T is

$$\Phi^T = \begin{bmatrix}\boldsymbol{\phi}(\boldsymbol{u}_1) & \dots & \boldsymbol{\phi}(\boldsymbol{u}_N).\end{bmatrix}^T.$$

This approaches suffers of the curse of dimensionality. The number of parameters n grows quickly with r and m, entailing high variance in the estimate. In Birpoutsoukis et al. (2017a), such problem has been addressed for Volterra series of order two exploiting regularized least squares. In particular, the smooth exponential decay of the Volterra coefficients is enforced by a suitable positive definite matrix (also called kernel) denoted by P. The estimator becomes

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left\| \boldsymbol{y} - \Phi^T \boldsymbol{w} \right\|^2 + \boldsymbol{w}^T P^{-1} \boldsymbol{w}, \tag{6.6}$$

with $\hat{w}$ available in closed form as

$$\hat{\boldsymbol{w}} = \left(\Phi\Phi^T + P^{-1} \right)^{-1} \Phi \boldsymbol{y}. \tag{6.7}$$

The (i, j)-entry of the matrix P can be seen as a similarity measure between the i-th and the j-th component of the vector $\boldsymbol{w}$ in (6.4), i.e., between two coefficients associated to two monomials. In Birpoutsoukis et al. (2017a), such matrix is block diagonal, i.e., $P = \text{block diag}([P_0, P_1, P_2])$. Let us discuss the three sub-blocks. First, P_0 is a positive scalar that regularizes h_0 in (6.4). Second, $P_1 \in \mathbb{R}^{n_1 \times n_1}$ and has to account for $\boldsymbol{w}_1$. This block contains the linear part of the system. Assume that the monomials in $\boldsymbol{\phi}_1(\boldsymbol{u}_k)$ are ordered as $\boldsymbol{w}_1 = [h_1(0), \ldots, h_1(m)]^T$, and let $P_1(i, j)$ describe the interaction between $h_1(i)$ and $h_1(j)$. Then, exploiting the DC kernel proposed in Pillonetto, Dinuzzo, Chen, De Nicolao, and Ljung (2014), the (i, j)-entry of P_1 is defined as

$$P_1(i, j) = c \cdot e^{-\alpha|i-j|} e^{-\beta \frac{|i+j|}{2}},$$

where c, α and β are hyperparameters that can be tuned from data. The variable c is the scale factor, while the other two regulate the impulse response exponential decay. The important contribution of Birpoutsoukis et al. (2017a) is the definition of the third matrix $P_2 \in \mathbb{R}^{n_2 \times n_2}$ that has to describe the interactions between all the second-order monomials, whose coefficients are contained in $\boldsymbol{w}_2$. Each monomial is in one-to-one correspondence with two lags assuming values in the set $\{0, 1, \ldots, m\}$ (their ordering is irrelevant in view of the nature of $\boldsymbol{w}_2$). Let the i-th and j-th monomial be given in terms of the couples $[\tau_1, \tau_2]$ and $[\tau_1', \tau_2']$, respectively. Let also $[v, u]$ and $[v', u']$ be the components of the two vectors w.r.t. the reference frame rotated of $\pi/4[rad]$ around the axis perpendicular to the plane identified by vectors $[1, 0]$ and $[0, 1]$. Then, the following definition holds

$$P_2(i, j) = c_2 p_v(i, j) p_u(i, j), \tag{6.8a}$$

$$p_v(i, j) = e^{-\alpha_v ||v| - |v'||} e^{-\beta_v \frac{||v| + |v'||}{2}}, \tag{6.8b}$$

$$p_u(i, j) = e^{-\alpha_u ||u| - |u'||} e^{-\beta_u \frac{||u| + |u'||}{2}}, \tag{6.8c}$$

where the hyperparameters are now c_2, α_v, α_u, β_v and β_u. Note that p_v (and similarly

p_u) is the product of two exponentials. The first one measures the similarity between two monomials, whereas the second one decreases if the lags projections assume large values. For instance, if $[\tau_1, \tau_2]$ coincide with $[\tau_1', \tau_2']$ the first factor is equal to one (the maximum value). Then, if $[\tau_1, \tau_2]$ and, hence, also $[v, u]$ are large, the corresponding monomial coefficient will be much penalized. In fact, one expects that monomials built with larger input lags will have less influence on output prediction.
Numerical results reported in Birpoutsoukis et al. (2017a) prove that the addition of the regularization term is crucial to control estimator's variance. Regarding computational issues, (6.7) shows that the number of operations scales with the cube of n (the number of distinct monomials), while the storage requirements are proportional to the square of n. Thus, there is a direct dependence on the number of coefficients of the Volterra maps. Unfortunately, this number grows rapidly with the system memory and with the order of the Volterra series. This makes already hard to introduce monomials of degree three.

6.2.3 Polynomial kernel and Volterra series

An alternative technique is regularized system identification in a RKHS defined by a kernel function $k(\boldsymbol{u}_t, \boldsymbol{u}_j)$. Under mild assumptions, a kernel function admits a (possibly infinite) expansion in terms of basis functions ψ_q, i.e.,

$$k(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_q \lambda_q \psi_q(\boldsymbol{u}_t) \psi_q(\boldsymbol{u}_j), \tag{6.9}$$

where λ_q are positive scalars. Any function in the induced RKHS has then the representation

$$f(\boldsymbol{u}_k) = \sum_q c_q \psi_q(\boldsymbol{u}_k), \tag{6.10}$$

for suitable coefficients c_q.
A widely used estimator of the input-output map is

$$\hat{f} = \arg\min_f \sum_{t=1}^{N} (y_t - f(\boldsymbol{u}_t))^2 + \gamma \|f\|_H^2, \tag{6.11}$$

where γ is the regularization parameter that trades-off data fit and the penalty term $\|\cdot\|_H^2$, given by the squared RKHS norm. According to the *representer theorem* Schölkopf, Herbrich, and Smola (2001), one has

$$\hat{f}(\boldsymbol{u}_k) = \sum_{t=1}^{N} \alpha_t k(\boldsymbol{u}_k, \boldsymbol{u}_t), \tag{6.12}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ is given by

$$\boldsymbol{\alpha} = (K + \gamma I_N)^{-1}\boldsymbol{y}, \tag{6.13}$$

with K the so called kernel matrix whose (t, j) entry is $k(\boldsymbol{u}_t, \boldsymbol{u}_j)$. So, even if the RKHS can implicitly encode a very large (possibly infinite) number of basis functions ψ_q, the estimate always belongs to a finite-dimensional subspace and can be computed with a finite number of operations.

A class of kernel-based solutions for Volterra series identification has been proposed in Franz and Schölkopf (2006). The authors rely on the use of polynomial kernels which are strongly connected with Volterra series since they encode monomials in $\boldsymbol{u}_k$. Let us consider

$$\sum_{i=1}^{r} \rho_i \left(\boldsymbol{u}_t^T \boldsymbol{u}_j\right)^i + \rho_0, \tag{6.14}$$

where the $\rho_i \geq 0$ are tunable hyperparameters. As discussed in Schölkopf and Smola (2001), each term $\left(\boldsymbol{u}_t^T \boldsymbol{u}_j\right)^i$ is the homogeneous polynomial kernel of degree i that encodes all the monomials of degree i. So the ψ_q in (6.9) are the elements of $\boldsymbol{\phi}_i$ in (6.4).

The computational bottleneck of kernel-based methods is the matrix inversion in (6.13) that requires $O(N^3)$ operations. The memory requirements to store the kernel matrix are $O(N^2)$. However, notice that there is no direct dependence on the number of basis functions encoded in the kernel. So, computational complexity does not depend on the number of Volterra parameters and this allows handling also high-order series.

6.3 Multiplicative Polynomial Kernel

Now, we show that in important cases the kernel hyperparameters in (6.14) do not provide enough flexibility to penalize the different monomials. For our purposes, it is useful also to recall the following two facts. First, given the kernel expansion (6.9) in terms of the eigenvalues λ_q and the independent basis functions ψ_q, one has

$$f(\boldsymbol{u}_k) = \sum_q c_q \psi_q(\boldsymbol{u}_k) \implies \|f\|_H^2 = \sum_q \frac{c_q^2}{\lambda_q}.$$

So, smaller λ_q give more penalty to the coefficients of the corresponding ψ_q. Second, as said, in the context of our polynomial kernels, the ψ_q belong to the set of monomials contained in

$$\boldsymbol{\phi}^T(\boldsymbol{u}_k) = \left[1, \boldsymbol{\phi}_1^T(\boldsymbol{u}_k), \ldots \boldsymbol{\phi}_r^T(\boldsymbol{u}_k)\right].$$

It is also useful now to denote the components of the blocks $\boldsymbol{\phi}_i$, i.e., the monomials of degree i, by $\phi_1^{(i)}, \phi_2^{(i)}, \ldots$. So, one has

$$\boldsymbol{\phi}_i(\boldsymbol{u}_k) = \left[\phi_1^{(i)}(\boldsymbol{u}_k)\ \phi_2^{(i)}(\boldsymbol{u}_k)\ \ldots\right]^T. \tag{6.15}$$

Each monomial of degree i is in one-to-one correspondence with a $m+1$-dimensional vector whose components are the relative degrees of the $\boldsymbol{u}_k$ components. In particular, we will use $\boldsymbol{d}_q^{(i)}$ to denote the vector that contains the relative degrees of the monomial $\phi_q^{(i)}$. We will also indicate with $\mathcal{D}_i$ the following set

$$\mathcal{D}_i = \left\{ \boldsymbol{d}^{(i)} = [d_0^{(i)}\ d_1^{(i)}\ \ldots\ d_m^{(i)}] \text{ s.t. } \sum_{\tau=0}^{m} d_\tau^{(i)} = i \right\}, \tag{6.16}$$

that is thus associated with all the monomials $\boldsymbol{\phi}_i$ in (6.15).

6.3.1 Penalties induced by the polynomial kernels

We focus on the penalties induced by the single homogenous kernels $\left(\boldsymbol{u}_t^T \boldsymbol{u}_j\right)^i$ that compose (6.14). We also introduce the positive semidefinite diagonal matrix $\Sigma^{(i)} \in \mathbb{R}^{(m+1)\times(m+1)}$ to define the generalized building block as

$$k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \left(\boldsymbol{u}_t^T \Sigma^{(i)} \boldsymbol{u}_j\right)^i, \tag{6.17a}$$

$$\Sigma^{(i)} = \operatorname{diag}(\boldsymbol{\sigma}^{(i)}), \tag{6.17b}$$

$$\boldsymbol{\sigma}^{(i)} = [\sigma_0^{(i)}, \ldots, \sigma_m^{(i)}]. \tag{6.17c}$$

The overall kernel turns out to be

$$K_{pk}^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{i=1}^{r} k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) + \rho_0. \tag{6.18}$$

Since $k_{pk}^{(i)}$ encodes all the monomials of degree i (if the $\Sigma^{(i)}$ is full-rank), using (6.15) one must have

$$k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{q=1}^{n_i} \lambda_q^{(i)} \phi_q^{(i)}(\boldsymbol{u}_t) \phi_q^{(i)}(\boldsymbol{u}_j). \tag{6.19}$$

But we can also find another representation through the multinomial theorem. Using (6.16) and (6.17), one has

$$k_{pk}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \left(\sum_{\tau=0}^{m} \sigma_\tau^{(i)} u_{t-\tau} u_{j-\tau}\right)^i = \sum_{\boldsymbol{d}^{(i)} \in \mathcal{D}_i} \binom{i}{\boldsymbol{d}^{(i)}} \prod_{\tau=0}^{m} \left(\sigma_\tau^{(i)} u_{t-\tau} u_{j-\tau}\right)^{d_\tau^{(i)}} \tag{6.20}$$

where here, and in what follows, we have used the fact that $\binom{i}{\boldsymbol{d}^{(i)}}$ is the number of all the different lags vectors that identify the same monomial.
If $\boldsymbol{d}_q^{(i)}$ contains the relative degrees $[d_{q,0}^{(i)}\ d_{q,1}^{(i)}\ \dots\ d_{q,m}^{(i)}]$ of the monomial $\phi_q^{(i)}$, equating the right hand sides of (6.19) and (6.20) we have

$$\lambda_q^{(i)} = \binom{i}{\boldsymbol{d}_q^{(i)}} \prod_{\tau=0}^{m} (\sigma_\tau^{(i)})^{d_{q,\tau}^{(i)}}. \tag{6.21}$$

The expression (6.21) highlights a first interesting issue. A part of the penalty is assigned only on the basis of the relative degrees associated to the monomial. Due to the behaviour of the multinomial coefficient, monomials composed by mixed terms are promoted. The gap between the penalties assigned to monomials with mixed terms and monomials composed by just one term becomes also particularly relevant when i grows. The reconstruction of the input-output function exploiting this kind of penalties might not be suitable, and higher flexibility could be desirable.

In (6.21), the presence of the multinomial coefficient $\binom{i}{\boldsymbol{d}^{(i)}}$ is due to repetitions induced by symmetry. To avoid this fact, and for visualization purposes, we express the penalties induced by $\lambda_q^{(i)}$ in the coefficients of the i-th Volterra map that define the input-output formulation (6.2). Note that the set of ordered lags $(\tau_1, \dots, \tau_i)$ identifies univocally i, q and, hence, the monomial $\phi_q^{(i)}$ and the vector $\boldsymbol{d}_q^{(i)}$. The opposite is not true due to the possible change of ordering: all the permutations of $(\tau_1, \dots, \tau_i)$ lead to the same monomial. In particular, due to symmetry, the $\lambda_q^{(i)}$ contribution is equally divided among all the coefficients of h_i associated to the $\binom{i}{\boldsymbol{d}_q^{(i)}}$ permutations of $(\tau_1, \dots, \tau_i)$. Then, the penalty assigned to the coefficient $h_i(\tau_1, \dots, \tau_i) = h_i(\boldsymbol{\tau})$ is inversely proportional to

$$\frac{\lambda_q^{(i)}}{\binom{i}{\boldsymbol{d}_q^{(i)}}} = \prod_{\tau=0}^{m} (\sigma_\tau^{(i)})^{d_{q,\tau}^{(i)}}. \tag{6.22}$$

The equality (6.22) permits to analyze the effects of the $\sigma_\tau^{(i)}$ in the penalties assigned to the Volterra maps. Firstly, notice that when considering $\sigma_\tau^{(i)} = 1\ \forall \tau = 0 \dots, m$, as done in Franz and Schölkopf (2006), the penalties assigned to the coefficients of the h_i maps are flat, given that all the coefficients are equally penalized. Secondly, acting on the values of the $\sigma_\tau^{(i)}$, we can promote or penalize the monomials containing $u_{k-\tau}$. However, from (6.22), we can see that promotions or penalizations are rigid, in the sense that by increasing $\sigma_\tau^{(i)}$ we promote simultaneously all the monomials of degree i in which $u_{k-\tau}$ appears. There might be cases in which more flexibility is needed to penalize $u_{k-\tau}$ by also accounting for its relative degree.

Remark 6.3.1. An alternative to (6.18) relies on the use of the inhomogeneous polynomial kernel of order r:

$$\left(1+\boldsymbol{u}_t^T\Sigma\boldsymbol{u}_j\right)^r, \tag{6.23}$$

where $\Sigma \in \mathbb{R}^{(m+1)\times(m+1)}$ is typically diagonal. The inhomogeneous polynomial kernel encodes all the monomials with degree up to r Schölkopf and Smola (2001). Similar considerations can be done for this model by recalling that (6.23) can be expressed as sum of r homogeneous kernels sharing the same $\Sigma = \Sigma^{(i)}$, multiplied by opportune coefficients, namely,

$$\left(1+\boldsymbol{u}_t^T\Sigma\boldsymbol{u}_j\right)^r = \sum_{i=0}^{r}\binom{r}{i}k_{pk}^{(i)}(\boldsymbol{u}_t,\boldsymbol{u}_j).$$

The last equation shows that the use of the inhomogeneous polynomial kernel further reduces the flexibility in determining penalties. In fact, the same hyperparameters determine simultaneously the penalties assigned to monomials with different degrees, while, through the model (6.17), one can exploit $\boldsymbol{\sigma}^{(i)}$ to tune specific regularization for each block $\boldsymbol{\phi}_i$ in (6.15).

6.3.2 Multiplicative Polynomial Kernel

In view of the limitations of the currently used polynomial kernels described above, the MPK is now introduced. Let $k_{mpk}^{(i)}$ denote a novel kernel that, similarly to the homogeneous polynomial kernel, encodes all the monomials of degree i, i.e., all those contained in $\boldsymbol{\phi}_i$. But its structure is different, being the product of i linear kernels, i.e.,

$$k_{mpk}^{(i)}(\boldsymbol{u}_t,\boldsymbol{u}_j) = \prod_{p=1}^{i}\left(\boldsymbol{u}_t^T\Sigma_p\boldsymbol{u}_j\right), \tag{6.24a}$$

$$\Sigma_p = \operatorname{diag}(\boldsymbol{\sigma}_p), \tag{6.24b}$$

$$\boldsymbol{\sigma}_p = [\sigma_0^{(p)},\dots,\sigma_m^{(p)}]. \tag{6.24c}$$

Differently from $k_{pk}^{(i)}$ in (6.17), the $k_{mpk}^{(i)}$ assigns assigns a distinct set of hyperparameters to each factor. It is easy to see that (6.24) is a well-defined kernel function, being the product of valid kernels Rasmussen and Williams (2006). Also, it admits an expansion in terms of the elements in $\boldsymbol{\phi}_i$. Then, we propose the following kernel to model Volterra series of order r:

$$K_{mpk}^{(r)}(\boldsymbol{u}_t,\boldsymbol{u}_j) = \sum_{i=1}^{r}\rho_i k_{mpk}^{(i)}(\boldsymbol{u}_t,\boldsymbol{u}_j) + \rho_0. \tag{6.25}$$

6.3.3 Penalties induced by the MPK

As done with (6.17), we analyze the regularization properties of (6.25) focusing on the single bulding blocks $k^{(i)}_{mpk}$. Using the same kind of notation adopted in the previous subsection, an expression of the $\lambda^{(i)}_q$ is obtained by expanding (6.24), isolating the terms with relative degrees $\boldsymbol{d}^{(i)}_q$, and, finally, summing their coefficients.
To perform such computation, let us introduce the following set

$$\mathcal{T}^{(i)}_q = \left\{ (\tau_1, \ldots, \tau_i) \text{ s.t. } \prod_{\tau=\tau_1}^{\tau_i} u_{k-\tau} = \prod_{\tau=0}^{m} u_{k-\tau}^{d^{(i)}_{q,\tau}} \right\}, \tag{6.26}$$

Each vector in $\mathcal{T}^{(i)}_q$ contains i lags associated with the same $\boldsymbol{d}^{(i)}_q$. Then, summing all the contributions associated to the elements of $\mathcal{T}^{(i)}_q$, one obtains

$$\lambda^{(i)}_q = \sum_{(\tau_1,\ldots,\tau_i)\in\mathcal{T}^i_q} \left(\prod_{p=1}^{i} \sigma^{(p)}_{\tau_p} \right). \tag{6.27}$$

The last equation shows that the MPK parameters allow for a more flexible regularization. In particular, tuning opportunely the hyperparameters, it is possible to promote or penalize the relative degree with which each lagged input appears. For example, consider the lagged input $u_{k-\tau}$, and the coefficients $\sigma^{(1)}_\tau, \ldots, \sigma^{(i)}_\tau$. Then, by inspection of (6.27), it comes that, in order to promote monomials in which $u_{k-\tau}$ appear with relative degree $d \leq i$, at least d of the $\sigma^{(1)}_\tau, \ldots, \sigma^{(i)}_\tau$ hyperparameters should assume large values. Otherwise, all the products in (6.27) and λ_q become small, much penalizing the monomials containing the powers $u^d_{k-\tau}, \ldots, u^i_{k-\tau}$. Standard polynomial kernels do not provide such possibility, given that, as showed before, changing the values of the hyperparameters promote, or penalize, simultaneously all the monomials containing $u_{k-\tau}$, regardless of the relative degree.

It is worth stressing that MPK obtains more flexibility by increasing the number of hyperparameters, possibly leading to overfitting. In particular, when considering the polynomial kernel of degree i, MPK has $(m+1)i$ additional hyparameters. However, as it will be seen through extensive numerical experiments, when considering sufficiently exciting input trajectories the prediction capability of MPK outperforms constantly that of the homogeneus and inhomogeneus polynomial kernel.

6.3.4 Re-parametrization of the diagonal Σ_p matrices

To optimize the Σ_p matrices, we will use Marginal Likelihood maximization Rasmussen and Williams (2006), by adopting a reparametrization. In fact, since (6.24) is the product of i equal blocks, optimizing directly the $\sigma_\tau^{(p)}$ could lead to undesired behaviors due to the presence of several local maxima. In particular, from the structure (6.24), it is immediate to see that any permutation of the i vectors $\boldsymbol{\sigma}_p$ that represent the diagonals of the matrices Σ_p defines the same $k_{mpk}^{(i)}$.
To avoid this situation, we propose a hierarchical parametrization of the $\boldsymbol{\sigma}_p$ vectors, w.r.t. a set of vectors $\{\boldsymbol{a}_p, p = 1, \dots, i\}$, where $\boldsymbol{a}_p = \left[a_0^{(p)}, \dots, a_m^{(p)}\right]$ with all non negative entries. More specifically, the $\boldsymbol{\sigma}_p$ vectors are defined iteratively by a backward iteration as follows,

$$\boldsymbol{\sigma}_i = \boldsymbol{a}_i, \tag{6.28a}$$

$$\boldsymbol{\sigma}_p = \boldsymbol{\sigma}_{p+1} + \boldsymbol{a}_p, \tag{6.28b}$$

with $p = i-1, \dots, 1$.

Beyond reducing the presence of possible local maxima, this parametrization has an intuitive interpretation also in terms of the penalties and the relative degree of each $u_{k-\tau}$. Indeed, from (6.28) we see that increasing $a_\tau^{(p)}$ means increasing all the $\sigma_\tau^{(d)}$ with $d \leq p$ and then, recalling (6.27), this promotes all the monomials in which $u_{k-\tau}$ has degree at least equal to p. Moreover, if $a_\tau^{(d)}$ is close to zero for $d > p$, then the monomials with $u_{k-\tau}$ of degree larger than p are highly penalized. In this way, by tuning opportunely the hyperparameters it is possible to control the maximum degree of each lagged input.

6.4 Smooth exponentially decaying MPK

In this section, we extend the framework introduced in the previous subsection by designing a novel Smooth Exponentially Decaying MPK (SED-MPK). The model incorporates information on smooth exponential decay of Volterra maps coefficients. The kernel has the same structure of the MPK (6.24), except for the positive definite matrices Σ_p which are no more restricted to be diagonal. In fact, we define the basic building kernel as

$$k_{sed}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \prod_{p=1}^{i} \boldsymbol{u}_t^T \Sigma_p \boldsymbol{u}_j \tag{6.29a}$$

$$\Sigma_p(i,j) = e^{-\alpha_p|i-j|} e^{-\beta_p|i-1+j-1|} \tag{6.29b}$$

where $\Sigma_p(i,j)$ is the (i,j)-entry of Σ_p, and the hyperparameters α_p and β_p regulate the smooth exponential decaying behavior. One can thus notice that while in (6.24) each Σ_p is diagonal with the hyperparameters corresponding to the $m+1$ diagonal elements, now the matrix is full but depends only on α_p and β_p. Similarly to the MPK, to limit the presence of local maxima, α_p and β_p can be parametrized hierarchically.
Then, similarly to the MPK, the SED-MPK of order r is the sum of kernels (6.29) of increasing order, i.e.,

$$K_{sed}^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{i=1}^{r} \rho_i k_{sed}^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) + \rho_0. \tag{6.30}$$

The regularization matrix in (6.29b) has clear analogies with the regularizer (6.8) introduced in Birpoutsoukis et al. (2017a). But a fundamental difference holds: while in Birpoutsoukis et al. (2017a) it is used to regularize the Volterra coefficients h_i in an explicit way, i.e., writing down the model in terms of basis functions, the penalty is here embedded in a kernel. This point is crucial, and the rest of the subsection is devoted to illustrate it by building a kernel that generalizes both MPK and SED-MPK.

As said, once an order i is fixed, the $h_i(\boldsymbol{\tau})$ is the coefficient of a monomial of order i defined by the vector $\boldsymbol{\tau} = [\tau_1, \dots, \tau_i]$ containing the input lags. Recall that $H_i(\boldsymbol{u}_k)$ in (6.2) is the sum of products between h_i and the possible monomials of degree i built with $\boldsymbol{u}_k$. In other words, this is the part of the system output due only to the monomials of order i. With this in mind, let $\mathcal{P}_i(\boldsymbol{\tau}, \boldsymbol{\tau}')$ be a kernel that measures the similarity between two lags vectors. Then, we define

$$k^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{\boldsymbol{\tau}} \sum_{\boldsymbol{\tau}'} \mathcal{P}_i(\boldsymbol{\tau}, \boldsymbol{\tau}') \prod_{p=1}^{i} u_{t-\tau_p} u_{j-\tau_p'} \tag{6.31}$$

as the similarity measure between the system outputs $H_i(\boldsymbol{u}_t)$ and $H_i(\boldsymbol{u}_j)$ obtained using two different inputs $\boldsymbol{u}_t$ and $\boldsymbol{u}_j$. Note that (6.31) accounts for the effect of all the monomials of order i built with $\boldsymbol{u}_t$ and $\boldsymbol{u}_j$ by summing over all the possible couples of vector lags. As a final step, the global kernel that accounts for all the monomials of degree up to r is defined as

$$K^{(r)}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \sum_{i=0}^{r} k^{(i)}(\boldsymbol{u}_t, \boldsymbol{u}_j). \tag{6.32}$$

Now, consider (6.29a) but without assigning any particular structure to Σ_p. In particular, for t, j in $\{0, 1, \dots, m\}$ let $\sigma_{tj}^{(p)}$ be the $(t+1, j+1)$-entry of Σ_p. Then, it is easy to prove that (6.29a) generalizes into

$$\sum_{\boldsymbol{\tau}} \sum_{\boldsymbol{\tau}'} \prod_{p=1}^{i} \sigma^{(p)}_{\tau_p \tau'_p} u_{t-\tau_p} u_{j-\tau'_p}, \tag{6.33}$$

where we are summing over all the possible couples of lags vectors

$$\boldsymbol{\tau} = [\tau_1, \dots, \tau_i], \quad \boldsymbol{\tau}' = [\tau'_1, \dots, \tau'_i].$$

Equating the right hand sides of (6.33) and (6.31), we obtain

$$\mathcal{P}_i(\boldsymbol{\tau}, \boldsymbol{\tau}') = \prod_{p=1}^{i} \sigma^{(p)}_{\tau_p \tau'_p}. \tag{6.34}$$

The equality (6.34) is important because it shows the correspondence between the similarity measure assigned to two lags vectors (and, hence, also to two Volterra map coefficients) and the elements of the Σ_p. In particular, (6.34) says that the similarity between $h_i(\boldsymbol{\tau})$ and $h_i(\boldsymbol{\tau}')$ is factorized by i elements, corresponding to the entries of the Σ_p matrices identified by $\boldsymbol{\tau}$ and $\boldsymbol{\tau}'$. Without parametrizing the Σ_p structure, the model complexity could become relevant, in particular when the system memory m and the Volterra order i are large. MPK reduces complexity by using diagonal matrices exploiting (6.24b). Instead, inspired by Pillonetto and De Nicolao (2010) and Birpoutsoukis et al. (2017a), SED-MPK uses the option (6.29b) to limit the number of parameters and enforce smooth exponential decay of the $h_i(\boldsymbol{\tau})$. This feature is so encoded in the kernel, providing information that monomials built with large input lags values should have less influence on output prediction. To clarify this concept, we provide the explicit expression of the $\mathcal{P}(\boldsymbol{\tau}, \boldsymbol{\tau}')$ obtained with the kernel in (6.29), in the particular case where all the α_p (resp. β_p) are equal to α (resp. β). By (6.33), we have

$$\mathcal{P}(\boldsymbol{\tau}, \boldsymbol{\tau}') = e^{-\alpha\|\boldsymbol{\tau}-\boldsymbol{\tau}'\|_1} e^{-\beta\|\boldsymbol{\tau}+\boldsymbol{\tau}'\|_1}. \tag{6.35}$$

The fundamental point is that, thanks to our kernel-based framework, the SED-MPK computational complexity scales with the number of samples, instead that with the number of coefficients of the Volterra maps as in Birpoutsoukis et al. (2017a). This allows to model also high-order Volterra series. As proven by numerical results reported in the next section, solutions based on SED-MPK are effective in identifying also Volterra series up to order 5.

6.5 Experimental results

We tested the proposed kernel functions in several experiments. Firstly, we compare the performance of the MPK with that of the standard polynomial kernels. In particular, we considered the benchmark system introduced in Spinelli, Piroddi, and Lovera (2005), described by a Volterra series with order $r = 3$ and memory $m = 6$. Finally, in the last part of the section, to evaluate the advantages of the SED-MPK, we use the two proposed kernels to identify a more challenging system, proposed in Stoddard and Welsh (2018), and modeled as Volterra series with $r = 3$ and $m = 70$.

6.5.1 Experiment 1

The system considered in this experiment is described by the following equation

$$\begin{aligned} z_k =& u_k + 0.6u_{k-1} + 0.35(u_{k-2} + u_{k-4}) - 0.25u_{k-3}^2 \\ &+ 0.2(u_{k-5} + u_{k-6}) + 0.9u_{k-3} \\ &+ 0.25u_k u_{k-1} + 0.75u_{k-2}^3 - u_{k-1}u_{k-2} \\ &+ 0.5(u_k^2 + u_k u_{k-2} + u_{k-1}u_{k-3}). \end{aligned} \tag{6.36}$$

The estimator based on the MPK is compared with the ones based on the homogeneous polynomial kernel (PK) and the inhomogeneous polynomial kernel (IPK), expressed, respectively, by (6.18) and (6.23). The input signals are 1000 samples obtained from a realization of Gaussian noise. Concerning m_u^{tr}, m_u^{ts}, σ_u^{tr} and σ_u^{ts}, which are, respectively the input mean and standard deviation of the training and test samples, we considered two different scenarios :

- *Setup 1*: $m_u^{tr} = m_u^{ts} = 0$, $\sigma_u^{tr} = \sigma_u^{ts} = 4$;
- *Setup 2*: $m_u^{tr} = m_u^{ts} = 0$, $\sigma_u^{tr} = 1$, $\sigma_u^{ts} = 4$.

Notice that in *Setup 1* the distribution of the training and test inputs is the same, while in *Setup 2* is different. In particular, we limited the variability of the training samples, increasing the probability of generating test samples that are significantly distant from the training distribution. Consequently, the second scenario is more challenging. In all the experiments the noise standard deviation is $\sigma_n = 4$. The hyperparameters of the kernels have been trained maximizing the marginal likelihood in the training samples. As concerns the optimization, we used standard gradient descent algorithm, with adaptive learning rate. The algorithms are implemented in Pytorch Paszke et al. (2017), to exploit its automatic differentiation capabilities for computing the gradient.

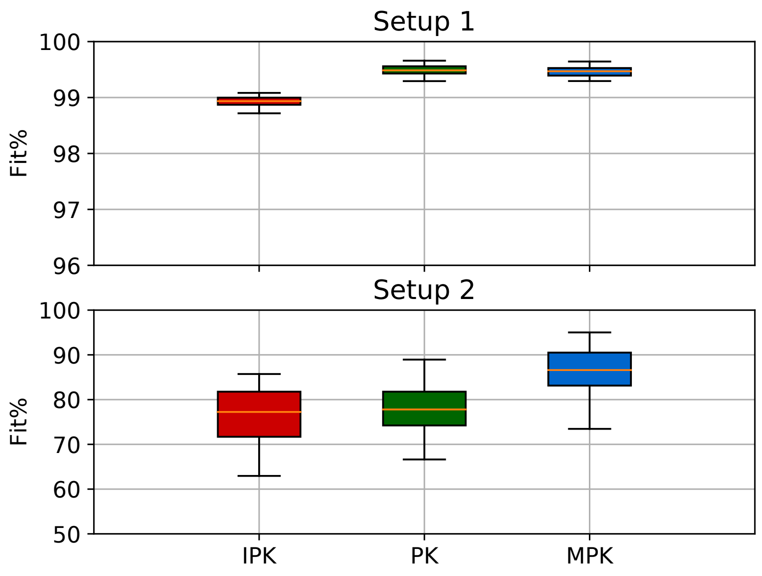

Figure 6.1: Boxplot of the Fit obtained with a Monte Carlo simulation composed of 100 simulations of the system described by (6.36). IPK, PK, and MPK correspond to the estimators based on (6.23), (6.18), and (6.24), respectively.

For each scenario we performed a Monte Carlo of 100 simulations. In each simulation the same training and test dataset have been used to train and test the three estimators. Performance is measured by the percentage fit (Fit%), defined as

$$100\% \left(1 - \frac{\|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_1}{\|\boldsymbol{z} - \bar{z}\|_1}\right),$$

where $\boldsymbol{z}$ is a vector collecting the noiseless system outputs, $\hat{\boldsymbol{z}}$ is the vector of the estimated outputs, and, finally, $\bar{z}$ is the mean of $\boldsymbol{z}$. Results are reported in Figure 6.1. As far as *Setup 1* is concerned, all the three estimators perform well. This is due to the fact that the input distribution does not vary between training and test, combined with the low value of m. Instead, performance vary significantly in the second scenario, where the MPK-based estimator outperforms the other estimators; the MPK Fit is 10% higher than the IPK and PK Fit. This fact shows that, thanks to the higher flexibility provided by the additional parameters, the MPK extracts more information from the training data, obtaining better out of sample performance.

To closely inspect about the regularization properties of the three kernels, we analyze the penalties assigned to the different monomials. In particular, we focus on the monomials with degree two, and we compute the penalty assigned to the h_2 map. We recall that $\lambda_q^{(i)}$, the scalar defining the penalty assigned to monomial $\phi_q^{(i)}$, is equally divided among the

Volterra map coefficients associated to $\phi_q^{(i)}$. Then, given a vector of lags $\boldsymbol{\tau} = [\tau_1 \, \dots \, \tau_i]$, with $\phi_q^{(i)} = \prod_{\tau \in \boldsymbol{\tau}} u_{k-\tau}$, we have that $\lambda_{h_i(\boldsymbol{\tau})}$, the portion of λ_q assigned to the $h_i(\boldsymbol{\tau})$ coefficient, is given by

$$\lambda_{h_i(\boldsymbol{\tau})} = \frac{\lambda_q^{(i)}}{\binom{i}{\boldsymbol{d}_q^{(i)}}}. \tag{6.37}$$

In Figure 6.2, we compare the magnitude of the h_2 coefficients of the system in (6.36) with the λ_{h_2} computed by the IPK, PK and MPK estimators after tuning the hyperparameters by marginal likelihood maximization. We recall that small values of λ entail high penalization. Results confirm the consideration done in Section 6.2 and 6.3. Notice that the IPK and PK estimators are not able to penalize properly the different monomials. In particular, IPK promotes significantly only the monomial u_{k-2}^2. This is probably due to the constraints between penalties assigned to the different orders. Indeed, notice that in (6.36) the only monomial with degree three is u_{k-2}^3. Instead, PK is not able to select the relative degree of each monomial; for example, u_{k-1}^2 is promoted despite its maximum relative degree in (6.36) is one. Finally, results confirm that MPK can promote monomials depending on the relative degrees. Indeed, the monomial $u_{k-2}u_{k-3}$ is promoted without promoting u_{k-2}^2 and u_{k-3}^2, contrary to PK, and in accordance with (6.36).

6.5.2 Experiment 2

In this set of experiments we compare the performance of the estimators based on MPK and SE-MPK, simulating a more complex system. We considered the Wiener system proposed in Stoddard and Welsh (2018), and described by the following equation,

$$z_k = \sum_{i=1}^{r} \left(\frac{10q^{-1}}{A(q)} u_k \right)^i, \tag{6.38}$$

where q is the forward shift operator, and $A(q) = 1 - 1.8036q^{-1} + 0.8338q^{-2}$. To analyze the robustness of the two kernels w.r.t. measurement noise, we performed simulations with different noise levels. Specifically, we varied σ_n obtaining simulations with signal to noise ratio (SNR) approximately equal to $5dB$ and $20dB$. Concerning the Volterra orders, we considered r = 2,...,5. As for the system memory m, it was set to 70. For each Volterra order we performed a Monte Carlo study composed of 40 simulations. As done in Stoddard and Welsh (2018), in each simulation the training and test inputs are the collection of 3500 samples, generated from a Gaussian distribution with zero mean and unitary standard deviation. Notice that the number of samples is approximately

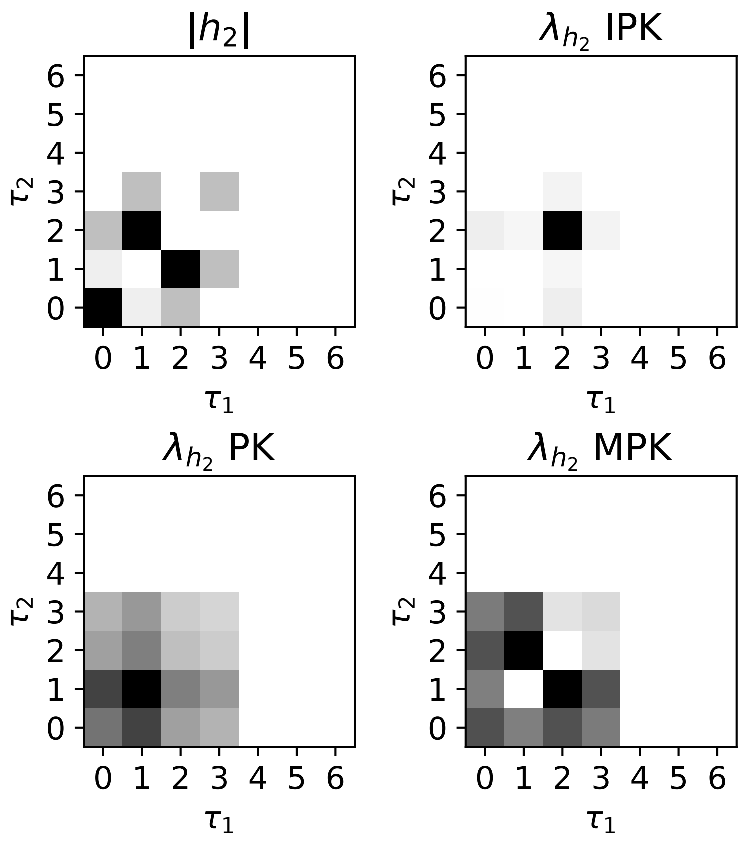

Figure 6.2: Visualization of h_2 coefficients' magnitude associated to the system in (6.36), compared with the penalties assigned to the h_2 coefficients by IPK, PK, MPK. Penalties have been computed with the expression in the left-hand-side of (6.37). Dark squares represent higher values. For each matrix the values have been normalized w.r.t. the maximum value, i.e., black squares corresponds to 1, while white squares to zero.

equal to the number of distinct coefficients of a Volterra map with $r = 2$ and $m = 70$, and significantly lower than the number monomials in $\boldsymbol{\phi}_i$ with $i > 2$. In Figure 6.3 we visualized the Fit obtained for $r = 2$ and $r = 3$, after training the hyperparameters by marginal likelihood maximization. As concerns the SED-MPK, we considered the special case reported in (6.35), with all the α_p and β_p equal. Results show that, compared to SED-MPK, the performance of the MPK decreases faster with the increasing of r, as well as being more sensitive to measurement noise. Due to the high number of parameters, and the small number of samples, the MPK is not able to identify a convenient set of penalties capable of providing generalization and robustness to measurement noise. Instead, results show the effectiveness of the regularization strategy implemented by the SED-MPK. Notice that the SED-MPK based estimator provides accurate estimates of the system output also with $r = 3$, as well as being robust to the presence of noise. This trend is confirmed also by results reported in Figure 6.4, where we plotted the SED-MPK Fit obtained with $r = 3, 4, 5$. Despite the number of samples is orders of magnitude smaller than the number of monomials in $\boldsymbol{\phi}_5$, the average Fit is higher that 80%. Finally, notice that effects of noise become particularly relevant only with $r = 5$, where there have been several outliers. However, we would like to stress that the considered configuration is particularly challenging.

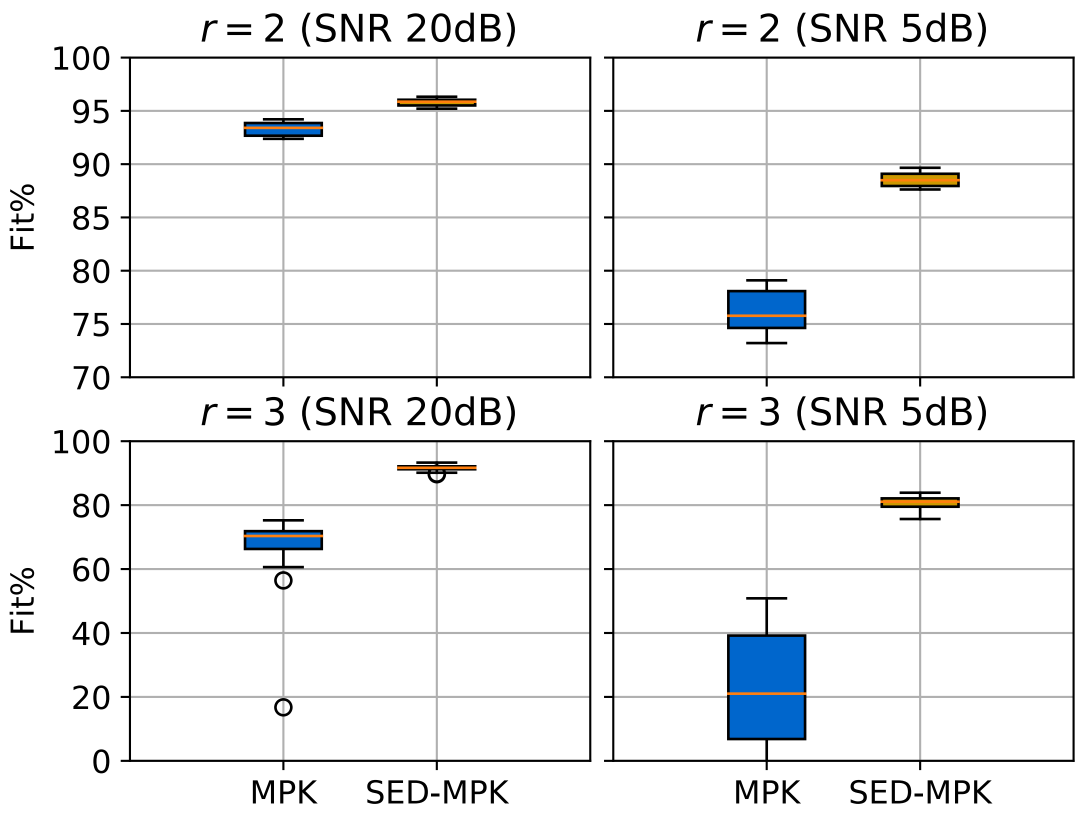

Figure 6.3: Comparison of the Fit obtained by MPK and SED-MPK in a Monte Carlo study composed of 40 simulations of the system in (6.38).

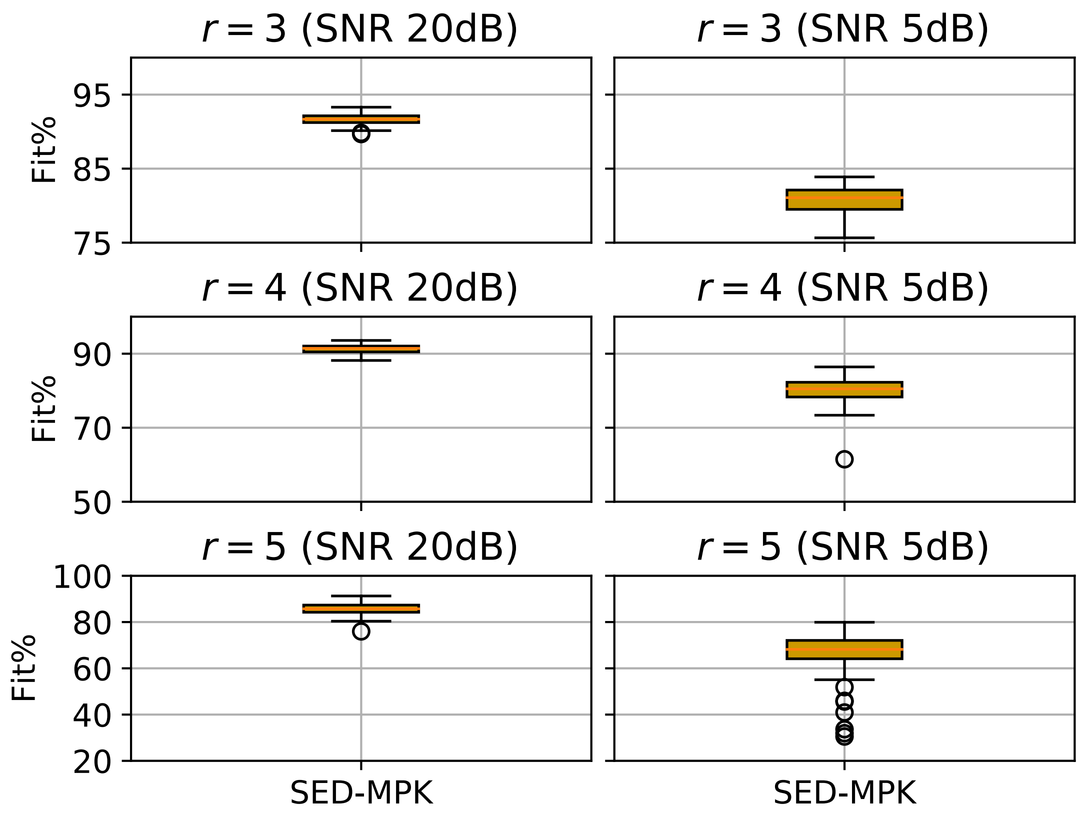

Figure 6.4: Boxplot of the Fit obtained by the SED-MPK based estimator in a Monte Carlo study composed of 40 simulations of the system in (6.38), with $r = 3, 4, 5$.

6.6 Conclusion

In this chapter, we introduce two new kernels to deal with the Volterra series identification. The two new kernels are named *multiplicative polynomial kernel* and *smooth exponentially decaying multiplicative polynomial kernel*. The MPK of degree i is defined as the product of i basic building blocks, represented by distinct linear kernels. Compared to standard polynomial kernels, the MPK identify the same RKHS, but it is equipped with a richer set of hyperparameters that allows for a flexible regularization, allowing to penalize monomials w.r.t. their maximum relative degree. Similarly to the MPK, the SED-MPK is defined by the product of distinct linear kernels. However, by defining parameters of each linear kernel w.r.t. a proper function, the SED-MPK encodes the sooth exponentially decaying of the Volterra maps coefficients. In this way, we combine encoding properties of the polynomial kernel with explicit regularization of the Volterra maps, increasing significantly regularization and data-efficiency. Experimental results obtained in two benchmark systems confirm the effectiveness of the proposed solutions, and validate the possibility of identifying high order Volterra series using a limited number of samples.

7 Conclusions

In this manuscript, we presented different learning algorithms for robotics systems. In the first part, we introduced two data-driven algorithms for deriving the kinematics and dynamics models of the robot, while in the last part we addressed the collision detection problem, as well as proposing a model-based Reinforcement Learning algorithm for learning control.

Specifically, the kinematics identification algorithm proposed in Chapter 2 derives the geometrical model of the robot combining proprioceptive measures with data acquired with a 2D camera. No prior knowledge about the robot geometry is assumed. Hence, the algorithm is particularly useful in the context of Modular Robotics, as well as in applications of automatic adaptation to hardware failure. The effectiveness of the solution is shown with tests performed on a real setup, where we validate also the implementation of a simple kinematic controller based on the model learned.

In Chapter 3, we derived the GIP kernel, a data-efficient kernel for robot inverse dynamics identification. Experimental results showed that, compared to other data-driven solutions, the GIP kernel achieves better performance, both in terms of accuracy and data efficiency. Despite it requires minimal prior information about the robot, the GIP kernel-based estimator behaves similarly to model-based estimators. However, with respect to model-based solutions, our algorithm is not affected by model bias, and it is not platform-dependent, possibly leading to reductions in costs and time design.

In Chapter 4, we validated the possibility of using GPR for deriving a proprioceptive collision detection algorithm. Moreover, we focused the attention on behaviors happening in quasi-static configurations, where friction forces become particularly non-linear and unpredictable. Besides highlighting the consequences of not considering these behaviors, we modified the standard GPR models used to identify the inverse dynamics, with the aim of modeling effects due to frictions and improving the collision detection performance. Tests performed with a UR10 robot confirmed the effectiveness of the proposed solution.

In Chapter 5, we introduced MC-PILCO, a model-based Reinforcement Learning algorithm inspired by PILCO. By approximating the expected cost relying on Monte Carlo particle-based methods, MC-PILCO addresses two main limitations of the original algorithm, namely, the unimodality of the predicted state distributions, and the mandatory use of RBF kernel to model the system evolution. A comparison of the performance obtained in the learning of a swing-up task shows consistent improvements, both in terms of speed and success rate.

Finally, in the last chapter, we analyzed in-depth the properties of the MPK, a refinement of the standard polynomial kernel used by the GIP kernel. In particular, we applied the proposed kernel to the identification of Volterra series, a class of models particularly useful to describe non-linear systems. Results confirm that, compared to the standard polynomial kernel, the MPK exhibits better regularization properties and, consequently, higher out of sample accuracy.

A

Appendix

A.1 Proof of Proposition 2.3.5

To prove Proposition 2.3.5, we introduce the relative transformations between two links L_{j_1} and L_{j_2} which are, in general, not subsequent. The relative orientation between L_{j_1} and L_{j_2}, denoted by $R^{L_{j_1}}_{L_{j_2}}$, is derived iterating (2.2) along the kinematic chain, namely considering all transformations due to the joints between L_{j_1} and L_{j_2}, that is,

$$R^{L_{j_1}}_{L_{j_2}} = \prod_{m=j_1+1}^{m=j_2} \left(R_z(\theta_m) R_x(\alpha_m) \right), \tag{A.1}$$

where θ_m is a function of q_{k_m} if the joint is revolute. Instead, as regards $\boldsymbol{l}^{L_{j_1}}_{L_{j_2}}$, i.e., the position of L_{j_2} w.r.t. L_{j_1}, we have

$$\boldsymbol{l}^{L_{j_1}}_{L_{j_2}} = \sum_{m=j_1+1}^{m=j_2} \left(R^{L_{j_1}}_{L_{m-1}} \boldsymbol{l}^{L_{m-1}}_{L_m} \right). \tag{A.2}$$

Proposition 2.3.5 states that, when excitation trajectories are in accordance with Definition 2.3.4, if $(M_{i_1} M_{i_2}, q_k)$ verifies equations of Proposition 2.3.1 for the minimal set of observations $\bar{\mathcal{D}}$, then L_{j_1} and L_{j_2}, that is, the links to which M_{i_1} and M_{i_2} are attached, are subsequent in the kinematic chain and connected through the prismatic joint q_k.

We prove Proposition 2.3.5 by contradiction. First, we consider the case in which there is at least one revolute joint between L_{j_1} and L_{j_2}. Without loss of generality assume that j_3 is the index of the link coming after the revolute joint. The relative orientation between the markers is $R^{M_{i_1}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_1}} R^{L_{j_1}}_{L_{j_3}} R^{L_{j_3}}_{L_{j_2}} R^{L_{j_2}}_{M_{i_2}}$, which, exploiting (A.1), can be rewritten as

$$\begin{aligned} R^{M_{i_1}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_1}} &\left(\prod_{m=j_1+1}^{m=j_3-1} R_z(\theta_m) R_x(\alpha_m) \right) \times \\ &R_z(\theta_{j_3}) R_x(\alpha_{j_3}) \left(\prod_{\bar{m}=j_3+1}^{\bar{m}=j_2} R_z(\theta_{\bar{m}}) R_x(\alpha_{\bar{m}}) \right) R^{L_{j_2}}_{M_{i_2}} \end{aligned} \tag{A.3}$$

Now let $\theta_{j_3} = q_{k_{j_3}} + \theta^0_{j_3}$, where $\theta^0_{j_3}$ is the initial value and $q_{k_{j_3}}$ the joint variable. Moreover let us define

$$R^{\bar{L}_{j_3}}_{M_{i_2}} := R_z(\theta^0_{j_3}) R_x(\alpha_{j_3}) \left(\prod_{\bar{m}=j_3+1}^{\bar{m}=j_2} R_z(\theta_{\bar{m}}) R_x(\alpha_{\bar{m}}) \right) R^{L_{j_2}}_{M_{i_2}}.$$

Then it follows

$$R^{M_{i_1}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_3-1}} R_z(q_{k_{j_3}}) R^{\bar{L}_{j_3}}_{M_{i_2}}.$$

If conditions of Proposition 2.3.1 are verified, then $R^{M_{i_1}}_{M_{i_2}}$ is constant, and, in turn, when considering two different time instants t_1 and t_2, we have $R^{M_{i_1}}_{M_{i_2}}(t_1) = R^{M_{i_1}}_{M_{i_2}}(t_2)$ implying

$$R^{M_{i_1}}_{L_{j_3-1}}(t_1) R_z(q_{k_{j_3}}(t_1)) R^{\bar{L}_{j_3}}_{M_{i_2}}(t_1) = R^{M_{i_1}}_{L_{j_3-1}}(t_2) R_z(q_{k_{j_3}}(t_2)) R^{\bar{L}_{j_3}}_{M_{i_2}}(t_2). \tag{A.4}$$

Now, consider the set of observation $\bar{\mathcal{D}}$ introduced in Definition 2.3.4 and, in particular, let us select two input locations belonging to the subset of trajectories with $mod\left(q_{k_{j_3}}(t_1)\right) \neq mod\left(q_{k_{j_3}}(t_2)\right)$ and $q_w(t_1) = q_w(t_2)$ if $w \neq k_{j_3}$. Then $R^{M_{i_1}}_{L_{j_3-1}}(t_1) = R^{M_{i_1}}_{L_{j_3-1}}(t_2)$ and $R^{\bar{L}_{j_3}}_{M_{i_2}}(t_1) = R^{\bar{L}_{j_3}}_{M_{i_2}}(t_2)$, but for the uniqueness of the Euler angles $R_z(q_{k_{j_3}}(t_1)) \neq R_z(q_{k_{j_3}}(t_2))$, thus contradicting (A.4).

To conclude the proof, we consider the case in which there is at least one prismatic joint between L_{j_1} and L_{j_2}. Under this assumption the relative translation between markers is

$$\begin{aligned}
\boldsymbol{l}_{M_{i_2}}^{M_{i_1}} =& \boldsymbol{l}_{L_{j_1}}^{M_{i_1}} + R_{L_{j_1}}^{M_{i_1}} \boldsymbol{l}_{L_{j_2}}^{L_{j_1}} + R_{M_{i_2}}^{M_{i_1}} \left(-\boldsymbol{l}_{L_{j_2}}^{M_{i_2}}\right) \qquad (A.5)\\
=& \sum_{m=j_1+1}^{m=j_2} \left(R_{L_{m-1}}^{M_{i_1}} \left(\begin{bmatrix} 0 \\ 0 \\ d_m \end{bmatrix} + R_z(\theta_m) \begin{bmatrix} a_m \\ 0 \\ 0 \end{bmatrix} \right) \right) + \\
& \boldsymbol{l}_{L_{j_1}}^{M_{i_1}} + R_{M_{i_2}}^{M_{i_1}} \left(-\boldsymbol{l}_{L_{j_2}}^{M_{i_2}}\right) + \sum_{m=j_1+1}^{m=j_2} \left(R_{L_{m-1}}^{M_{i_1}} \begin{bmatrix} 0 \\ 0 \\ q_{k_m} \end{bmatrix} \right) \\
=& \boldsymbol{l}_{i_1*}^{i_2*} + \sum_{m=j_1+1}^{m=j_2} \left(R_{L_{m-1}}^{M_{i_1}} \begin{bmatrix} 0 \\ 0 \\ q_{k_m} \end{bmatrix} \right), \qquad (A.6)
\end{aligned}$$

where in $\boldsymbol{l}_{i_1*}^{i_2*}$ we have collected all the terms which are constant, while the terms in the last sum depend on the joint signals. Now, similarly to (2.7), let us define the parameters vector $\boldsymbol{b}_{i_1*}^{i_2*} = [\boldsymbol{l}_{i_1*}^{i_2*} \ (\boldsymbol{z}_{L_{j_1}}^{M_{i_1}})^T \ \ldots \ (\boldsymbol{z}_{L_{j_2-1}}^{M_{i_1}})^T]$, and the matrix $A_* = \left[I_3 \ \ q_{k_{j_1}} I_3 \ \ldots \ q_{k_{j_2}} I_3\right]$ where the $\boldsymbol{z}_{L_{m-1}}^{M_m}$ vectors have unitary norm. Accordingly, let $\mathcal{A}_*(\bar{\mathcal{D}})$ the matrix built stacking the matrices $A_*(t_{j,h})$, $j = 1, 2$, $h = 1, \ldots, n-1$, on top of one another.

If equations of Proposition 2.3.1 are verified for the set of observations $\bar{\mathcal{D}}$, then, from (2.9) it follows that

$$\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right) = \mathcal{A}(\boldsymbol{q}_k)\boldsymbol{b}_{i_1}^{i_2},$$

but, also,

$$\boldsymbol{l}\left(P_{i_1}, P_{i_2}\right) = \mathcal{A}_*(\bar{\mathcal{D}})\boldsymbol{b}_{i_1*}^{i_2*}.$$

However, when joints trajectories belong to the class defined in Definition 2.3.4, $\text{rank}\left(\mathcal{A}(\boldsymbol{q}_k)\right) = 6$ and $\text{rank}\left(\mathcal{A}^*(\bar{\mathcal{D}})\right) = 3 + 3(j_2 - j_1)$, and, given the constraints on $\boldsymbol{b}_{i_1}^{i_2}$ and $\boldsymbol{b}_{i_1*}^{i_2*}$, $\text{rank}\left(A(\bar{\mathcal{D}})\boldsymbol{b}_{i_1}^{i_2}\right) \leq \text{rank}\left(A^*(\bar{\mathcal{D}})\boldsymbol{b}_{i_1*}^{i_2*}\right)$. Then, it must hold $j_2 = j_1 + 1$. Finally, since $\mathit{span}\left([q_{k_i}(t_1) \ldots q_{k_i}(t_{n-1})]\right) = \mathit{span}\left(\left[q_{k_j}(t_1) \ldots q_{k_j}(t_{n-1})\right]\right)$ if and only of $k_i = k_j$, the equivalence in (A.1) holds only when considering the joint input signal q_k. This concludes the proof of Proposition 2.3.5.

A.2 Proof of Proposition 2.3.6

We prove the proposition by contradiction. Assume that systems defined by Proposition 2.3.2 and Proposition 2.3.3 admit a solution for the set of observations $\bar{\mathcal{D}}$. Consider the case where L_{j_1} and L_{j_2} are not consecutive. Let N_{rev} be the number of revolute joints present in the chain between the two links, and let $k_{j_3} \dots k_{j_3+N_{rev}}$ be the corresponding indexes. The relative orientation between M_{i_2} and M_{i_1} is $R^{M_{i_1}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_1}} R^{L_{j_1}}_{L_{j_2}} R^{L_{j_2}}_{M_{i_2}}$, with $R^{L_{j_1}}_{L_{j_2}}$ as in (A.1). We rewrite $R^{M_{i_1}}_{M_{i_2}}$ highlighting the contributions of a particular revolute joint, for example joint j_3, obtaining $R^{M_{i_1}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_3-1}} R_z(q_{k_{j_3}}) R^{\bar{L}_{j_3}}_{M_{i_2}}$. Since the system defined by Proposition 2.3.3 holds true, we have $R^{M_{i_1}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_1}} R_z(q_k) R^{\bar{L}_{j_2}}_{M_{i_2}}$, with $R^{M_{i_1}}_{L_{j_1}}$ and $R^{\bar{L}_{j_2}}_{M_{i_2}}$ corresponding to the solution of the non linear system. Then it follows

$$R^{M_{i_1}}_{L_{j_1}} R_z(q_k) R^{\bar{L}_{j_2}}_{M_{i_2}} = R^{M_{i_1}}_{L_{j_3-1}} R_z(q_{k_{j_3}}) R^{\bar{L}_{j_3}}_{M_{i_2}}.$$

Exploiting the standard properties of the rotation matrices and of the Euler angles, it holds $R_z(q_k) = R^{L_{j_1}}_{M_{i_1}} R^{M_{i_1}}_{L_{j_3-1}} R_z(q_{k_{j_3}}) R^{\bar{L}_{j_3}}_{M_{i_2}} R^{M_{i_2}}_{\bar{L}_{j_2}}$. Rewriting $R^{L_{j_1}}_{M_{i_1}} R^{M_{i_1}}_{L_{j_3-1}}$ and $R^{\bar{L}_{j_3}}_{M_{i_2}} R^{M_{i_2}}_{\bar{L}_{j_2}}$ w.r.t. their Euler angles, and adopting the zxz convention, we obtain

$$R_z(q_k) = R_z(\alpha_1) R_x(\beta_1) R_z(\gamma_1) R_z(q_{k_{j_3}}) R_z(\alpha_2) R_x(\beta_2) R_z(\gamma_2),$$

where $(\alpha_1, \beta_1, \gamma_1)$ refers to $R^{L_{j_1}}_{M_{i_1}} R^{M_{i_1}}_{L_{j_3-1}}$ and $(\alpha_2, \beta_2, \gamma_2)$ refers to $R^{\bar{L}_{j_3}}_{M_{i_2}} R^{M_{i_2}}_{\bar{L}_{j_2}}$. Then

$$R_z(q_k - \alpha_1 - \gamma_2) = R_x(\beta_1) R_z(q_{k_{j_3}} + \gamma_1 + \alpha_2) R_x(\beta_2).$$

Given the uniqueness of the Euler angles, the last equation holds true if and only if β_1 and β_2 are null, and $q_k - \alpha_1 - \gamma_2 = q_{k_{j_3}} + \gamma_1 + \alpha_2$, namely $q_k - q_{k_{j_3}} = \alpha_1 + \gamma_2 + \gamma_1 + \alpha_2$. Now, consider two observations at time t_1 and t_2, selected from $\bar{\mathcal{D}}$, satisfying $mod\left(q_{k_{j_3}}(t_1)\right) \neq mod\left(q_{k_{j_3}}(t_2)\right)$ and $q_w(t_1) = q_w(t_2)$ if $w \neq k_{j_3}$. With this type of trajectory α_1, β_1, γ_1, α_2, β_2 and γ_2 are constant when t is equal to t_1 and t_2. Then, it follows

$$q_k(t_1) - q_{k_{j_3}}(t_1) = q_k(t_2) - q_{k_{j_3}}(t_2).$$

Under the excitation assumptions in Definition 2.3.4, the last equation holds if and only if $k = k_{j_3}$. Similar considerations can be done when considering the other revolute joints, thus implying that only one revolute joint can be present between L_{j_1} and L_{j_2}, and that the corresponding input must be q_k.

To conclude the proof we need to consider the case in which between L_{j_1} and L_{j_2} there is at least one prismatic joint, in addition to the revolute joint q_k. Let j_3 be the index of the link after the revolute joint, that is, the revolute joint connects L_{j_3-1} and L_{j_3}. Consider the expression of $\boldsymbol{l}^{M_{i_1}}_{M_{i_2}}$ in (2.4), and let us expand $\boldsymbol{l}^{L_{j_1}}_{L_{j_2}}$ highlighting the prismatic transformations that occurs before, or after, the revolute joint. We have

$$
\begin{aligned}
\boldsymbol{l}_{M_{i_2}}^{M_{i_1}} =& \boldsymbol{l}_{L_{j_1}}^{M_{i_1}} + R_{M_{i_2}}^{M_{i_1}}(-\boldsymbol{l}_{L_{j_2}}^{M_{i_2}}) + \sum_{m=j_1+1}^{m=j_3-1} R_{L_{jm}}^{M_{i_1}} \boldsymbol{l}_{L_m}^{L_{m-1}} \\
&+ R_{Lj_3-1}^{M_{i_1}} \boldsymbol{l}_{L_{j_3}}^{L_{j_3-1}} + R_{M_{i_2}}^{M_{i_1}} \sum_{m=j_3+1}^{m=j_2} R_{L_{j_{m-1}}}^{M_{i_2}} \boldsymbol{l}_{L_m}^{L_{m-1}},
\end{aligned} \tag{A.7}
$$

where the $R_{L_{jm}}^{M_{i_1}}$, $R_{L_{j_3-1}}^{M_{i_1}}$ and $R_{L_{j_{m-1}}}^{M_{i_2}}$ matrices are constant. The two sums account for the prismatic transformation happening, respectively, before and after the revolute joint; in each sum, the $\boldsymbol{l}_{L_m}^{L_{m-1}}$ expression is defined in (2.3), where d_j is a function of a suitable joint signal. The translation due to the revolute joint instead, is represented by $R_{Lj_3-1}^{M_{i_1}} \boldsymbol{l}_{L_{j_3}}^{L_{j_3-1}}$, where $\boldsymbol{l}_{L_{j_3}}^{L_{j_3-1}}$ is defined as in (2.3), with d_{j_3} constant and θ_{j_3} function of q_k. Equation (A.7) can be rewritten as a linear expression w.r.t. a suitable vector $\boldsymbol{b}_{i_1*}^{i_2*}$ and the following matrix of coefficients

$$
A_* = \left[I_3 \;\; q_{k_{j_1}} I_3 \;\; \dots \;\; q_{k_{j_3}} I_3 \;\; q_{k_{j_3+1}} R_{M_2}^{M_1} \;\; \dots \;\; q_{k_{j_2}} R_{M_2}^{M_1} \;\; R_z(q_k) \right]
$$

Moreover, recall that if conditions of Proposition 2.3.2 are verified $\boldsymbol{l}\,(P_{i_1}, P_{i_2}) = \bar{\mathcal{A}}(P_{i_1}, P_{i_2})\bar{\boldsymbol{b}}_{i_1}^{i_2}$, and then

$$
\boldsymbol{l}\,(P_{i_1}, P_{i_2}) = \bar{\mathcal{A}}(P_{i_1}, P_{i_2})\bar{\boldsymbol{b}}_{i_1}^{i_2} = \mathcal{A}_*(\bar{\mathcal{D}})\boldsymbol{b}_{i_1*}^{i_2*}.
$$

If the input trajectories are in accordance with Definition 2.3.4 we have that rank $\left(\bar{\mathcal{A}}(P_{i_1}, P_{i_2})\right) =$ 6 while rank $\left(\mathcal{A}_*(\bar{\mathcal{D}})\right) = 6 + N_p$, where N_p is the number of prismatic joints. With considerations similar to the ones done in the proof of the previous proposition, we obtain that $N_p = 0$. Then we have proved that conditions of Proposition 2.3.6 are verified if and only if the triplet (M_{i_1}, M_{i_2}, q_k) is connected and q_k is revolute.

A.3 Proof of Proposition 2.5.1

We prove Proposition 2.5.1 by inspection. Given that Proposition 2.5.1 assumes the knowledge of the kinematic structure, in the following we assume that $\mathcal{Q}_q = \{q_1, \dots, q_n\}$, that is, joints are provided in accordance with the order with which they occur in the kinematic chain. The relative rotation between the marker attached to the last link and the camera frame is given by

$$
R_M^c(\boldsymbol{q}) = R_{L_1}^c R_{L_n}^{L_1}(\boldsymbol{q}) R_M^{L_n},
$$

where $R^c_{L_1}$ and $R^{L_n}_M$ are constant matrices, expressing, respectively, the orientation of the base link RF w.r.t. the camera RF and the relative orientation between the last marker and the RF of the last joint. The expression of the matrix $R^{L_1}_{L_n}(\boldsymbol{q})$ instead, is reported in (A.1), namely, it is factorized by the matrices $R^{L_{m-1}}_{L_m}$, $m = 2, \dots, n$, defined as in (2.2). Observe that the elements of $R^{L_{m-1}}_{L_m}$ are constant if the joint between links L_m and L_{m-1} is prismatic, or linear functions of $\cos(q_{m-1})$ and $\sin(q_{m-1})$ if the joint is revolute. Consequently, $R^c_M(\boldsymbol{q})$ is a polynomial function in $\boldsymbol{q}_c$ and $\boldsymbol{q}_s$, with maximum degree N_r, where each monomial is subject to

$$deg(q_{c_b}) + deg(q_{s_b}) \leq 1. \tag{A.8}$$

To prove that the relative translation between the marker attached to the last link and the camera RF is a polynomial function in the elements of $\boldsymbol{q}_c$, $\boldsymbol{q}_s$ and $\boldsymbol{q}_p$ we follow the same reasoning. The expression of $\boldsymbol{l}^c_M$ is

$$\boldsymbol{l}^c_M = \boldsymbol{l}^c_{L_1} + R^c_{L_1} \boldsymbol{l}^{L_n}_{L_1}(\boldsymbol{q}) + R^c_{L_n}(\boldsymbol{q}) \boldsymbol{l}^{L_n}_M,$$

where $\boldsymbol{l}^c_{L_1}$ and $\boldsymbol{l}^{L_n}_M$ are the coordinates of the base-link and the marker attached to the last link expressed, respectively, w.r.t. the camera RF and L_n- RF. From (A.2), $\boldsymbol{l}^{L_n}_{L_1}(\boldsymbol{q})$ can be factorized as $R^{L_1}_{L_{m-1}}(\boldsymbol{q}) \boldsymbol{l}^{L_{m-1}}_{L_m}(\boldsymbol{q})$, where $\boldsymbol{l}^{L_{m-1}}_{L_m}$ is defined as in (2.3). Its elements are linear functions of $\cos(q_{m-1})$, $\sin(q_{m-1})$ when q_{m-1} is revolute, or of q_{m-1} if q_{m-1} is prismatic. Then, the elements of $R^{L_1}_{L_{m-1}}(\boldsymbol{q})$ are polynomial functions in q_{c_b}, q_{s_b}, with $b = 1, \dots, m-2$, subject to (A.8). It turns out that $\boldsymbol{l}^c_M$ is a polynomial function in the elements of $\boldsymbol{q}_c$, $\boldsymbol{q}_s$ and $\boldsymbol{q}_p$, where each element appears with maximum degree 1, and each monomial is subject to the constraint in (A.8).

A.4 Proof of Proposition 3.3.1

We prove Proposition 3.3.1 by inspection, analyzing individually all the terms in (3.1), i.e., the $B(\boldsymbol{q})\ddot{\boldsymbol{q}}$ and $C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ contributions and the gravity term $\boldsymbol{g}(\boldsymbol{q})$. Firstly, we provide a characterization of the elements of $B(\boldsymbol{q})$ as polynomials in $\boldsymbol{q}_c$, $\boldsymbol{q}_s$ and $\boldsymbol{q}_p$. The inertia matrix is given by

$$B(\boldsymbol{q}) = \sum_{i=1}^{i=n} m_i J_i^T J_i + J_{\omega_i}^T R^0_i I^i_i R^i_0 J_{\omega_i},$$

where m_i and I^i_i are the i-th link mass and inertia matrix, expressed in a reference frame (RF) solidal with the i-th link. J_i and J_{ω_i} are the linear and angular Jacobians of the i-th RF, i.e., $\dot{\boldsymbol{c}}_i = J_i \dot{\boldsymbol{q}}$ and $\boldsymbol{\omega}_i = J_{\omega_i} \dot{\boldsymbol{q}}$, where $\boldsymbol{c}_i$ is the position of the center of mass of the i-th link, while $\boldsymbol{\omega}_i$ is the angular velocity of the i-th RF. To expand the J_i and J_{ω_i} expressions, we introduce some notions regarding the kinematics. Adopting the Denavit-Hartenberg (DH) convention, the R_i^{i-1} and $\boldsymbol{l}_i^{i-1}$ variables, which denote the i-th RF orientation and translation with the respect to the previous RF, are

$$\begin{aligned} R_i^{i-1} &= R_z(\theta_i)\, R_x(\alpha_i)\,, \\ \boldsymbol{l}_i^{i-1} &= [0\,,\,0\,,\,d_i]^T + R_z(\theta_i)\,[a_i\,,\,0\,,\,0]^T\,, \end{aligned}$$

where R_x and R_z are the elementary rotation matrices around the x and z axis, while a_i and α_i are two constant geometrical parameters, see Siciliano et al. (2009). The definitions of d_i and θ_i depend on the joint interconnecting the i-th link with the previous link. When the joint is revolute, d_i is constant and $\theta_i = \theta_{0_i} + q_i$, and the only terms that depend on $\boldsymbol{q}$ are $\cos(q_i)$ and $\sin(q_i)$ contained in R_i^{i-1}. Referring to the polynomial notation previously introduced, we can write that the elements of R_i^{i-1} are functions in $\mathbb{P}_{(1)}\left(\cos(q_i)_{(1)}\,, \sin(q_i)_{(1)}\right)$. In case the joint is prismatic, θ_i is constant, and $d_i = d_{0_i} + q_i$. Consequently the only $\boldsymbol{q}$ dependent terms are in $\boldsymbol{l}_i^{i-1}$. In particular the elements of $\boldsymbol{l}_i^{i-1}$ belong to $\mathbb{P}_{(1)}\left(q_{i_{(1)}}\right)$.

The J_{ω_i} matrix relates $\dot{\boldsymbol{q}}$ with the angular velocity of the i-th link. Adopting the DH convention, $\boldsymbol{\omega}_i^{i-1} = \lambda_i\,[0\,,\,0\,,\,\dot{q}_i]^T$, 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 $R_j^0 = \prod_{b=j}^{b=0} R_b^{b-1}$ matrices, and remarking that $\boldsymbol{\omega}_i = \sum_{j=1}^{j=i} \lambda_j R_{j-1}^0 \boldsymbol{\omega}_j^{j-1}$, we obtain

$$\boldsymbol{\omega}_i \;=\; \begin{bmatrix} R_0^0 \begin{bmatrix} 0 \\ 0 \\ \lambda_1 \end{bmatrix} , \dots , R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ \lambda_i \end{bmatrix} , \mathbf{0}(3, n-i) \end{bmatrix} \dot{\boldsymbol{q}},$$

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

$$J_{\omega_i} = \begin{bmatrix} R_0^0 \begin{bmatrix} 0 \\ 0 \\ \lambda_1 \end{bmatrix} , \dots , R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ \lambda_i \end{bmatrix} , \mathbf{0}(3, n-i) \end{bmatrix}.$$

Exploiting the properties of the rotation matrices, we obtain

$$R_0^i J_{\omega_i} = \begin{bmatrix} R_0^i \begin{bmatrix} 0 \\ 0 \\ \lambda_1 \end{bmatrix} , \dots , R_{i-1}^i \begin{bmatrix} 0 \\ 0 \\ \lambda_1 \end{bmatrix} , \mathbf{0}(3, n-i) \end{bmatrix} \dot{\boldsymbol{q}}.$$

Let $\{I_r \leq i\}$ be the set of revolute joint indexes lower or equal than i, and let $\boldsymbol{q}_c(\{I_r \leq i\})$

be the corresponding subset. Recalling that the elements of R_i^{i-1} are functions in $\mathbb{P}_{(1)}\left(\cos\left(q_i\right)_{(1)}, \sin\left(q_i\right)_{(1)}\right)$, with maximal degree one, and that $R_j^k = \prod_{b=j}^{b=k} R_b^{b-1}$, with $j > k$, it follows that the $J_{\omega_i}^T R_i^0 I_i^i R_0^i J_{\omega_i}$ elements belong to

$$\mathbb{P}_{(2|\{I_r \leq i\}|)}\left(\boldsymbol{q}_c\left(\{I_r \leq i\}\right)_{(2)}, \boldsymbol{q}_s\left(\{I_r \leq i\}\right)_{(2)}\right),$$

where in each monomial the following constraint holds

$$deg\left(q_{c_b}\right) + deg\left(q_{s_b}\right) \leq 2\,. \tag{A.9}$$

To derive a similar characterization of the J_i elements, we analyze the $\boldsymbol{c}_i$ expression. The position of the *i-th* center of mass in the base frame is $\boldsymbol{c}_i = \sum_{j=1}^{j=i-1} R_{j-1}^0 \boldsymbol{l}_j^{j-1} + R_i^0 \boldsymbol{c}_i^i$. Then the $\boldsymbol{c}_i$ elements are functions in

$$\mathbb{P}_{(i)}\left(\boldsymbol{q}_c\left(\{I_r \leq i\}\right)_{(1)}, \boldsymbol{q}_s\left(\{I_r \leq i\}\right)_{(1)}, \boldsymbol{q}_p\left(\{I_p \leq i\}\right)_{(1)}\right),$$

and in each monomial the following inequality holds

$$deg\left(q_{c_j}\right) + deg\left(q_{s_j}\right) \leq 1. \tag{A.10}$$

Since $\dot{\boldsymbol{c}}_i = J_i\dot{\boldsymbol{q}}$, and since the derivative of $\cos\left(q_j\right)$, $\sin\left(q_j\right)$ and q_j does not increase the degree of these terms when inequality (A.10) holds, it follows that the J_i elements belong to the same functional space of $\boldsymbol{c}_i$. Consequently, the elements of $J_i^T J_i$ are functions in $\mathbb{P}_{(2i)}\left(\boldsymbol{q}_c\left(\{I_r \leq i\}\right)_{(2)}, \boldsymbol{q}_s\left(\{I_r \leq i\}\right)_{(2)}, \boldsymbol{q}_p\left(\{I_p \leq i\}\right)_{(2)}\right)$; as before, in each monomial the q_{c_j} and q_{s_j} degrees are subject to inequality (A.9).

Given the characterization of $J_i^T J_i$ and $J_{\omega_i}^T R_i^0 I_i^i R_0^i J_{\omega_i}$, we obtain that the $B\left(\boldsymbol{q}\right)$ elements are functions in $\mathbb{P}_{(2n)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}\right)$, where in each monomial the q_{c_j} and q_{s_j} degrees are subject to inequality (A.9). Then the $B\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}}$ are functions in $\mathbb{P}_{(2n+1)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}, \ddot{\boldsymbol{q}}_{(1)}\right)$.

As reported in Siciliano et al. (2009), the *i-th* element of the $C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}}$ product is equal to

$$\sum_{j=1}^{j=n} c_{ij}\dot{q}_j = \sum_{j=1}^{j=n}\sum_{k=1}^{k=n}\left(\frac{\partial b_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial b_{ik}}{\partial q_i}\right)\dot{q}_k\dot{q}_j.$$

Since the $B\left(\boldsymbol{q}\right)$ elements belong to $\mathbb{P}_{(2n)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}\right)$ and (A.9) holds true, also the b_{ij} partial derivatives belong to $\mathbb{P}_{(2n)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}\right)$, with (A.9) satisfied. Indeed, for each monomial in b_{ij}, the derivation respect to $\boldsymbol{q}_p$ decreases the degree by one, while the derivation respect to $\boldsymbol{q}_c$ or $\boldsymbol{q}_s$ does not alter the monomial degree. Then we obtain that the elements of $C(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \dot{\boldsymbol{q}}$ are functions in $\mathbb{P}_{(2n+1)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}, \dot{\boldsymbol{q}}_{v_{(1)}}\right)$.

Regarding $\boldsymbol{g}(\boldsymbol{q})$, we observe that the *i-th* element is given by $-\partial U / \partial q_i$, where by definition the potential energy $U = \sum_{j=1}^{j=n} \boldsymbol{g}_0^T \boldsymbol{c}_j$, with $\boldsymbol{g}_0$ denoting the vector of the gravitational acceleration. Then, the elements of $\boldsymbol{g}(\boldsymbol{q})$ are functions in the same space of the J_n 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 $\mathbb{P}_{(2n+1)}\left(\boldsymbol{q}_{c_{(2)}}, \boldsymbol{q}_{s_{(2)}}, \boldsymbol{q}_{p_{(2)}}, \dot{\boldsymbol{q}}_{v_{(1)}}, \ddot{\boldsymbol{q}}_{(1)}\right)$, with each monomial satisfying (A.9).

References

Andersen T. T. Optimizing the universal robots ros driver. Technical report, Technical University of Denmark, Department of Electrical Engineering, 2015.

Augugliaro F., Lupashin S., Hamer M., Male C., Hehn M., Mueller M. W., Willmann J. S., Gramazio F., Kohler M., and D'Andrea R. The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine*, 34(4):46–64, Aug 2014.

Berkenkamp F., Turchetta M., Schoellig A. P., and Krause A. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 908–919, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3294771.3294858.

Billings S., Chen A., and Korenberg M. Identification of MIMO non-linear systems using a forward-regression orthogonal algorithm. *Intern. J. of Control*, 49:2157 – 2189, 1989.

Birpoutsoukis G., Marconato A., Lataire J., and Schoukens J. Regularized nonparametric volterra kernel estimation. *Automatica*, 82:324 – 327, 2017a.

Birpoutsoukis G., Csurcsia P., and Schoukens J. Nonparametric volterra series estimate of the cascaded water tanks using multidimensional regularization. *IFAC-PapersOnLine*, 50:476–481, 07 2017b.

Brodbeck L. and Iida F. Enhanced robotic body extension with modular units. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1428–1433, Oct 2012.

Brommer C., Malyuta D., Hentzen D., and Brockers R. Long-duration autonomy for small rotorcraft UAS including recharging. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page arXiv:1810.05683. IEEE, oct 2018. URL https://doi.org/10.1109/iros.2018.8594111.

Caflisch R. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998. ISSN 0962-4929.

Camoriano R., Traversaro S., Rosasco L., Metta G., and Nori F. Incremental semiparametric inverse dynamics learning. In *ICRA*, pages 544–550, May 2016.

Chen S., Billings S. A., and Luo W. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50: 1873–1896, 1989.

Chua K., Calandra R., McAllister R., and Levine S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *CoRR*, abs/1805.12114, 2018. URL http://arxiv.org/abs/1805.12114.

Cirillo A., Ficuciello F., Natale C., Pirozzi S., and Villani L. A conformable force/tactile skin for physical human–robot interaction. *IEEE Robotics and Automation Letters*, 1(1):41–48, Jan 2016. ISSN 2377-3766.

Dalla Libera A. and Carli R. A data-efficient geometrically inspired polynomial kernel for robot inverse dynamic. *IEEE Robotics and Automation Letters*, 5(1):24–31, Jan 2020. ISSN 2377-3774.

Dalla Libera A., Terzi M., Rossi A., Susto G. A., and Carli R. Robot kinematic structure classification from time series of visual data. In *2019 18th European Control Conference (ECC)*, pages 1586–1591, June 2019a.

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

Dalla Libera A., Carli R., and Pillonetto G. A novel multiplicative polynomial kernel for volterra series identification. *arXiv:1905.07960*, 2019c.

Dantone M. Real time detection of human body and face parts. 2014.

De Luca A., Albu-Schaffer A., Haddadin S., and Hirzinger G. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630, Oct 2006.

De Luca A. and Mattone R. Actuator failure detection and isolation using generalized momenta. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 634–639 vol.1, Sep. 2003.

Deisenroth M. P., Fox D., and Rasmussen C. E. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, Feb 2015.

Deisenroth M. P. and Rasmussen C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 465–472, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL `http://dl.acm.org/citation.cfm?id=3104482.3104541`.

Dupont P. E. Friction modeling in dynamic robot simulation. In *Robotics and Automation*, 1990.

Durrant-Whyte H., Roy N., and Abbeel P. *Learning to Control a Low-Cost Manipulator Using Data-Efficient Reinforcement Learning.* MITP, 2012. URL `https://ieeexplore.ieee.org/document/6301026`.

Ebert D. M. and Henrich D. D. Safe human-robot-cooperation: image-based collision detection for industrial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1826–1831 vol.2, Sept 2002.

Efron B., Hastie T., Johnstone L., and Tibshirani R. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Espinoza M., Suykens J. A. K., and De Moor B. Kernel based partially linear models and nonlinear identification. *IEEE Trans. on Automatic Control*, 50(10): 1602–1606, 2005.

Fiala M. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1317–1324, July 2010. ISSN 0162-8828.

Finn C., Yu T., Zhang T., Abbeel P., and Levine S. One-shot visual imitation learning via meta-learning. In **Levine S., Vanhoucke V., and Goldberg K.**, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 357–368. PMLR, 13–15 Nov 2017. URL `http://proceedings.mlr.press/v78/finn17a.html`.

Franz M. and Schölkopf B. A unifying view of Wiener and volterra theory and polynomial kernel regression. *Neural Computation*, 18:3097–3118, 2006.

Frigola R., Lindsten F., Schon T., and Rasmussen C. Bayesian inference and learning in Gaussian process state-space models with particle mcmc. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Frigola R. and Rasmussen C. Integrated pre-processing for Bayesian nonlinear system identification with Gaussian processes. In *Proceedings of the 52nd Annual Conference on Decision and Control (CDC)*, 2013.

Gao G., Wang W., Lin K., and Chen Z. Kinematic calibration for articulated arm coordinate measuring machines base on particle swarm optimization. In *2009 Second International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 189–192, Oct 2009.

Ge S. S. and Hang C. C. Structural network modeling and control of rigid body robots. *IEEE Transactions on Robotics and Automation*, 14(5):823–827, Oct 1998. ISSN 1042-296X.

Gelderblom G. J., Wilt M. D., Cremers G., and Rensma A. Rehabilitation robotics in robotics for healthcare; a roadmap study for the european commission. In *2009 IEEE International Conference on Rehabilitation Robotics*, pages 834–838, June 2009.

Gilpin K. and Rus D. Modular robot systems. *IEEE Robotics Automation Magazine*, 17(3):38–55, Sep. 2010.

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

Guan Y., Jiang L., Zhangy X., Zhang H., and Zhou X. Development of novel robots with modular methodology. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2385–2390, Oct 2009.

Haber R. and Unbehauen H. Structure identification of nonlinear systems-a survey. *Automatica*, 26:651–677, 1990.

Haddadin S., De Luca A., and Albu-Schäffer A. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6): 1292–1312, Dec 2017. ISSN 1552-3098.

Hajjaj S. S. H. and Sahari K. S. M. Review of agriculture robotics: Practicality and feasibility. In *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 194–198, Dec 2016.

Hall J., Rasmussen C., and Maciejowski J. Modelling and control of nonlinear systems using Gaussian processes with partial model information. In *Proceedings of the 51st Annual Conference on Decision and Control (CDC)*, 2012.

Hanjong Joo , ChiSu Son , Kyunghun Kim , Kyunghwan Kim , and Jaejun Kim . A study on the advantages on high-rise building construction which the application of construction robots take (iccas 2007). In *2007 International Conference on Control, Automation and Systems*, pages 1933–1936, Oct 2007.

Hastie T. J., Tibshirani R. J., and Friedman J. *The Elements of Statistical Learning. Data Mining, Inference and Prediction.* Springer, Canada, 2001.

Hersch M., L. Sauser E., and Billard A. Online learning of the body schema. 5: 161–181, 06 2008.

Hewing L., Liniger A., and Zeilinger M. Cautious nmpc with gaussian process dynamics for autonomous miniature race cars. pages 1341–1348, 06 2018.

Hinton G. E., editor. *Neural Network Architectures for Artificial Intelligence.* American Association for Artificial Intelligence, Menlo Park, CA, USA, 1988. ISBN 0-929280-15-6.

Hong X., Mitchell R. J., Chen S., Harris C. J., Li K., and Irwin G. W. Model selection approaches for non-linear system identification: A review. *International Journal of Systems Science*, 39(10):925–946, 2008.

Hornby G. S., Lipson H., and Pollack J. B. Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719, Aug 2003.

Khosla P. and Kanade T. Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators. *IJRR*, 7(1):18 – 28, February 1988.

Kim H., Lee J. Y., Kim J. H., Kim J. B., and Han W. Y. Object recognition and pose estimation using klt. In *2012 12th International Conference on Control, Automation and Systems*, pages 214–217, Oct 2012.

Kim S. and Yoon K. Point density-invariant 3d object detection and pose estimation. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2647–2651, Sept 2017.

Kingma D. P. and Ba J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Kingma D. P. and Welling M. Auto-encoding variational bayes, 2013. URL http://arxiv.org/abs/1312.6114. cite arxiv:1312.6114.

Kurutach T., Clavera I., Duan Y., Tamar A., and Abbeel P. Model-ensemble trust-region policy optimization. *CoRR*, abs/1802.10592, 2018. URL http://arxiv.org/abs/1802.10592.

Landi C. T., Ferraguti F., Costi S., Bonfè M., and Secchi C. Safety barrier functions for human-robot interaction with industrial manipulators. In *2019 18th European Control Conference (ECC)*, pages 2565–2570, June 2019.

Latombe J. *Robot Motion Planning.* The Springer International Series in Engineering and Computer Science. Springer US, 2012. ISBN 9781461540229. URL https://books.google.it/books?id=nQ7aBwAAQBAJ.

Le D. P., Choi J., and Kang S. External force estimation using joint torque sensors and its application to impedance control of a robot manipulator. In *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, pages 1794–1798, Oct 2013.

Levine S. and Abbeel P. Learning neural network policies with guided policy search under unknown dynamics. In **Ghahramani Z., Welling M., Cortes C., Lawrence N. D., and Weinberger K. Q.**, editors, *NIPS 27*, pages 1071–1079. Curran Associates, Inc., 2014.

Li Y., Li L., Su H., and Chun J. Least squares support vector machine based partially linear model identification. In **Huang D.-S., Li K., and Irwin G.**, editors, *Intelligent Computing*, volume 4113 of *Lecture Notes in Computer Science*, pages 775–781. Springer Berlin Heidelberg, 2006.

Lin K., Rojas J., and Guan Y. A vision-based scheme for kinematic model construction of re-configurable modular robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2751–2757, Sep. 2017.

Lin Y., Zhao H., Ye C., and Ding H. A computationally efficient and robust kinematic calibration model for industrial robots with kinematic parallelogram. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1334–1339, Dec 2017.

Lind I. and Ljung L. Regressor selection with the analysis of variance method. *Automatica*, 41(4):693 – 700, 2005.

Lind I. and Ljung L. Regressor and structure selection in NARX models using a structured ANOVA approach. *Automatica*, 44:383–395, 2008.

Ljung L. *System Identification - Theory for the User*. Prentice-Hall, Upper Saddle River, N.J., 2nd edition, 1999.

Longo D. and Muscato G. The alicia/sup 3/ climbing robot: a three-module robot for automatic wall inspection. *IEEE Robotics Automation Magazine*, 13(1):42–50, March 2006.

López-Cerón A. Accuracy analysis of marker-based 3 d visual localization. 2016.

Malyuta D., Brommer C., Hentzen D., Stastny T., Siegwart R., and Brockers R. Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition. *Journal of Field Robotics*, page arXiv:1908.06381, August 2019. URL https://doi.org/10.1002/rob.21898.

Masinga P., Campbell H., and Trimble J. A. A framework for human collaborative robots, operations in south african automotive industry. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1494–1497, Dec 2015.

Mchutchon A. and College C. Nonlinear modelling and control using gaussian processes, 2014.

Nguyen L., Patel R. V., and Khorasani K. Neural network architectures for the forward kinematics problem in robotics. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 393–399 vol.3, June 1990.

Nguyen-Tuong D. and Peters J. Using model knowledge for learning inverse dynamics. In *ICRA*, pages 2677–2682, May 2010.

Nguyen-Tuong D., Seeger M., and Peters J. Local gaussian process regression for real time online model learning and control. In *NIPS*, pages 1193–1200, Red Hook, NY, USA, June 2009. Max-Planck-Gesellschaft, Curran.

Nguyen-Tuong D., Peters J., Seeger M., and Schölkopf B. Learning inverse dynamics: A comparison. pages 13–18, 01 2008.

Olson E. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, May 2011.

Parmas P., Rasmussen C. E., Peters J., and Doya K. PIPPS: Flexible model-based policy search robust to the curse of chaos. In **Dy J. and Krause A.**, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4065–4074, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/parmas18a.html.

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

Pillonetto G. and Chiuso A. Tuning complexity in regularized kernel-based regression and linear system identification: The robustness of the marginal likelihood estimator. *Automatica*, 58:106–117, 2015.

Pillonetto G. and De Nicolao G. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, 2010.

Pillonetto G., Dinuzzo F., Chen T., De Nicolao G., and Ljung L. Kernel methods in system identification, machine learning and function estimation: a survey. *Automatica*, 50(3):657–682, 2014.

Poggio T. and Girosi F. Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990.

Poignet P. and Gautier M. Nonlinear model predictive control of a robot manipulator. In *AMC*, pages 401–406, March 2000.

Polydoros A. S., Nalpantidis L., and Krüger V. Real-time deep learning of robotic manipulator inverse dynamics. In *IROS*, pages 3442–3448, Sep. 2015.

Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., and Ng A. Y. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

Rasmussen C. and Kuss M. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, pages 751–759, Cambridge, MA, USA, June 2004. Max-Planck-Gesellschaft, MIT Press.

Rasmussen C. and Williams C. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Romeres D., Jha D. K., Dalla Libera A., Yerazunis B., and Nikovski D. Semi-parametrical gaussian processes learning of forward dynamical models for navigating in a circular maze. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3195–3202, May 2019.

Romeres D., Zorzi M., Camoriano R., and Chiuso A. Online semi-parametric learning for inverse dynamics modeling. In *CDC*, pages 2945–2950, Dec 2016.

Rueckert E., Nakatenus M., Tosatto S., and Peters J. Learning inverse dynamics models in o(n) time with lstm networks. In *Humanoids*, pages 811–816, Nov 2017.

Rugh W. *Nonlinear System Theory: The Volterra-Wiener Approach.* Johns Hopkins University Press, 1980.

Rühr T., Sturm J., Pangercic D., Beetz M., and Cremers D. A generalized framework for opening doors and drawers in kitchen environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 3852–3858, May 2012.

Schölkopf B., Herbrich R., and Smola A. J. A generalized representer theorem. *Neural Networks and Computational Learning Theory*, 81:416–426, 2001.

Schölkopf B. and Smola A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* (Adaptive Computation and Machine Learning). MIT Press, 2001.

Siciliano B., Sciavicco L., Villani L., and Oriolo G. *Robotics, Modelling, Planning and Control.* 2009.

Siciliano B. and Villani L. *Robot Force Control.* Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 2000. ISBN 0792377338.

Silver D., Hubert T., Schrittwieser J., Antonoglou I., Lai M., Guez A., Lanctot M., Sifre L., Kumaran D., Graepel T., Lillicrap T., Simonyan K., and Hassabis D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. 2018. ISSN 0036-8075.

Sousa C. a. D. and Cortesão R. Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach. *Int. J. Rob. Res.*, 33(6):931–944, May 2014. ISSN 0278-3649. URL http://dx.doi.org/10.1177/0278364913514870.

Spinelli W., Piroddi L., and Lovera M. On the role of prefiltering in nonlinear system identification. *IEEE Transactions on Automatic Control*, 50(10):1597–1602, Oct 2005. ISSN 0018-9286.

Sprowitz A., Pouya S., Bonardi S., Den Kieboom J. V., Mockel R., Billard A., Dillenbourg P., and Ijspeert A. J. Roombots: Reconfigurable robots for adaptive furniture. *IEEE Computational Intelligence Magazine*, 5(3):20–32, Aug 2010. ISSN 1556-603X.

Stoddard J. and Welsh J. Volterra kernel identification using regularized orthonormal basis functions. 04 2018.

Sturm J., Plagemann C., and Burgard W. Unsupervised body scheme learning through self-perception. In *2008 IEEE International Conference on Robotics and Automation*, pages 3328–3333, May 2008.

Tibshirani R. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society, Series B.*, 58:267–288, 1996.

Todorov E., Erez T., and Tassa Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.

Todorov E. and Li W. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC.*, pages 300–306. IEEE, 2005.

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

Villagrossi E., Beschi M., Simoni L., and Pedrocchi N. A virtual force sensor for interaction tasks with conventional industrial robots. In *Mechatronics*, 2018.

Vuong N. D. and Jr M. H. A. Dynamic model identification for industrial robots. In *IEEE Control Systems*, 2007.

Wang J. and Olson E. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, Oct 2016.

Xu Y. and Chen D. Partially-linear least-squares regularized regression for system identification. *IEEE Trans. Automat. Contr.*, 54(11):2637–2641, 2009.

Yim M., Duff D. G., and Roufas K. D. Polybot: a modular reconfigurable robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference* *on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 514–520 vol.1, April 2000.

Yim M., Shen W., Salemi B., Rus D., Moll M., Lipson H., Klavins E., and Chirikjian G. S. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14(1):43–52, March 2007. ISSN 1070-9932.

Yun W., Lee J., Lee J., and Kim J. Object recognition and pose estimation for modular manipulation system: Overview and initial results. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 198–201, June 2017.

Zhou T. and Shi B. E. Simultaneous learning of the structure and kinematic model of an articulated body from point clouds. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 5248–5255, July 2016.