Computed Torque Control with Nonparametric Regression Models

Duy Nguyen-Tuong, Matthias Seeger, Jan Peters
Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen

***Abstract*— Computed torque control allows the design of considerably more precise, energy-efficient and compliant controls for robots. However, the major obstacle is the requirement of an accurate model for torque generation, which cannot be obtained in some cases using rigid-body formulations due to unmodeled nonlinearities, such as complex friction or actuator dynamics. In such cases, models approximated from robot data present an appealing alternative. In this paper, we compare two nonparametric regression methods for model approximation, i.e., locally weighted projection regression (LWPR) and Gaussian process regression (GPR). While locally weighted regression was employed for real-time model estimation in learning adaptive control, Gaussian process regression has not been used in control to-date due to high computational requirements. The comparison includes the assessment of model approximation for both regression methods using data originated from SARCOS robot arm, as well as an evaluation of the robot tracking performance in computed torque control employing the approximated models. Our results show that GPR can be applied for real-time control achieving higher accuracy. However, for the online learning LWPR is superior by reason of lower computational requirements.**

I. Introduction

Computed torque control offers a large variety of advantages over model-free methods which are only based on model-free linear feedback control, e.g., potentially higher tracking accuracy, lower feedback gains, higher suitability for compliant control, lower energy consumption, etc. However, a major drawback is the requirement of precise analytical models in order to predict the feed-forward torques required for the execution of the trajectory. From our experience [1], [2], such precise models cannot be obtained for many robot systems using standard rigid body formulations due to model errors, actuator dynamics and unmodeled nonlinearities. Thus, the obvious question is how well can our system perform if we use the best currently known regression techniques instead of the analytical model?

While this goal has been considered in the past [3]–[7], the progress in machine learning has long outpaced learning in robot control and it is about time that we reevaluate this issue using state-of-the-art regression techniques. For doing so, we compare several methods for approximating the dynamics model of the system in the setting of robot control. This comparison includes both the currently best method for real-time regression, locally weighted projection regression (LWPR) [8], [9] and the analytical model obtained through rigid body dynamics with estimated parameters [10], [11] against the modern regression technique, Gaussian process regression (GPR) [12], [13]. Two different comparisons are considered, i.e., (i) the function approximation comparison where we evaluate the learned function on data generated from a seven degrees of freedom (DoF) anthropomorphic SARCOS robot arm as shown in Figure 1, (ii) the tracking control performance using feedforward and inverse dynamics model approaches [11].

Fig. 1: Anthropomorphic SARCOS robot arm.

The remainder of this paper proceeds as follows: firstly, we discuss three different ways how such models can be estimated. Subsequently, we discuss how these can be used in robot control explaining our setup and, finally, we will show how these perform in a real-time control setup.

II. Model Estimation with Nonparametric Regression Techniques

Given the input $\mathbf{x} \in \mathbb{R}^n$ and the target $\mathbf{y} \in \mathbb{R}^n$, the task of regression algorithms is to learn the mapping describing the relationship from input to target using samples. Using this learned function, target values can be predicted for query input points.

In this paper we will consider the locally weighted projection regression and the Gaussian process regression. Locally-weighted projection regression is currently the standard learning method in robot control applications and has been shown to scale into very high-dimensional domains [8], [9]. However, it also requires skillful tuning of the meta parameters for the learning process in order to achieve competitive performance. Gaussian process regression [12], [13] on the other hand achieves this higher performance with very little tuning but has significantly higher computational requirements which increase with the number of data points.

A. Locally Weighted Projection Regression (LWPR)

In LWPR, the function values are approximated by a combination of N individually weighted locally linear models. In

addition, these values are normalized by the sum of all weights [1], [8]. Thus, given a data point $\mathbf{x}$ the predicted value $\hat{y}$ is

$$\hat{y} = \frac{\sum_{k=1}^{N} w_k \bar{y}_k}{\sum_{k=1}^{N} w_k}\,, \tag{1}$$

with $\bar{y}_k = \bar{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_k$ and $\bar{\mathbf{x}}_k = [(\mathbf{x} - \mathbf{c}_k)^T, 1]^T$, where w_k is the weight, $\hat{\boldsymbol{\theta}}_k$ contains the regression parameter and $\mathbf{c}_k$ is the center of the k-th linear model. The weight w_k is a measure of how much a data point $\mathbf{x}$ falls into the region of validity of each linear model and is characterized by a kernel function, for which a Gaussian kernel is often used

$$w_k = \exp\left(-\frac{1}{2}\left(\mathbf{x} - \mathbf{c_k}\right)^T \mathbf{D_k} \left(\mathbf{x} - \mathbf{c_k}\right)\right), \tag{2}$$

where $\mathbf{D_k}$ is a positive definite matrix which is called distance matrix. During the learning process the main purpose is to adjust $\mathbf{D_k}$ and $\hat{\boldsymbol{\theta}}_k$, so that the error between the predicted values and the targets are minimal.

The regression parameter $\hat{\boldsymbol{\theta}}_k$ can be calculated recursively using least squared method (RLS) [1]. Thus, $\hat{\boldsymbol{\theta}}_k^{n+1} = \hat{\boldsymbol{\theta}}_k^n + w_k \mathbf{P}_k^{n+1} \bar{\mathbf{x}}_k e_k$, where $\mathbf{P}$ is called covariance matrix and e_k is the approximation error. In new developments RLS can be substituted by partial least square (PLS) [8], [9]. Using PLS algorithm the regression is not done over the whole input space as in case of RLS but only in a subspace which is determined by principal components of the input. The key ingredient of the algorithm consists in choosing the principal components which are most relevant for the target spaces. Thus, the computing cost for the regression is reduced through a dimensionality reduction. The computational complexity is then linear in the number of inputs [8].

The distance matrix $\mathbf{D_k}$ determines the size and shape of each local model which can be updated individually by the incremental gradient descent method. The update rule is given by $\mathbf{D}_k = \mathbf{M}_k^T \mathbf{M}_k$ with $\mathbf{M}_k^{n+1} = \mathbf{M}_k^n - \alpha \delta J_k / \delta \mathbf{M}_k^n$, where J_k is a given cost function. Obtaning $\mathbf{D}_k$ the task is to minimize the penalized weighted mean squared error expressed in J_k [8].

Regarding the learning process for function approximation, no local models are initialized [9]. Whenever a training point $\mathbf{x}$ does not fall in any validity region of each linear model characterized by a constant w_{gen}, a new model is created with a center $\mathbf{c}_k = \mathbf{x}$. Thus, the number of local models increases with the complexity of the input. If a training point falls into the validity region of a model, its own distance matrix and regression parameters will be updated. Furthermore, the update of each model is computed completely independently of all other models.

B. Gaussian Process Regression (GPR)

As opposed to LWPR, GPR is performed with a Gaussian model completely described by its mean function and covariance function [13]. There are two equivalent ways to interpret the Gaussian regression, the function-space view and the weight-space view [12]. In order to keep the parallelism to LWPR method, the weight-space view of GPR will be introduced here. The simplest Gaussian model which is also useful for regression estimation is given by the linear model [12]

$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T \mathbf{w}\,, \quad y = f(\mathbf{x}) + \epsilon\,, \tag{3}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is the weight vector. It is further assumed that the observed value y is corrupted by additive noise ϵ, which is independent, Gaussian distributed with zero mean and variance σ_n^2. As seen in Equation (3), the linear computation is done after transforming the input $\mathbf{x}$ with a kernel function $\boldsymbol{\phi}(\bullet)$. The aim behind this action is to project $\mathbf{x}$ into a space of higher order called feature space, where the regression problem can be solved with a linear approach [14]. For this, different functions can be taken such as the Gaussian kernel as shown in (2). Other kernel functions can be found in [12], [14].

With the noise assumption and the model in (3) the likelihood, i.e., probability density of the observations y given the parameters $\boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{w}$, can be written as

$$p(\mathbf{y}|\boldsymbol{\Phi}, \mathbf{w}) \propto \exp(-0.5\,\sigma_n^{-2} |\mathbf{y} - \boldsymbol{\Phi}^T \mathbf{w}|^2) = \mathcal{N}(\boldsymbol{\Phi}^T \mathbf{w}, \sigma_n^2 \mathbf{I})\,,$$

where the vectors $\mathbf{y}$ and $\mathbf{w}$ contain all target and weight values, respectively, the matrix $\boldsymbol{\Phi}$ denotes the aggregations of columns $\boldsymbol{\phi}(\mathbf{x})$ for all cases in the training set. It's further assumed that the weights are Gaussian distributed with zero mean and variance $\boldsymbol{\Sigma}_p$ [12], [13]. Thus, the prior, i.e., probability density of $\mathbf{w}$, is given by

$$p(\mathbf{w}) \propto \exp\left(-0.5\,\mathbf{w}^T \boldsymbol{\Sigma}_p^{-1} \mathbf{w}\right) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)\,.$$

According to Bayes' rule, the posterior, i.e., probability density of the weights given inputs and targets, is proportional to the product of likelihood and prior [12]. Thus,

$$p(\mathbf{w}|\boldsymbol{\Phi}, \mathbf{y}) \propto \exp(-0.5\,(\mathbf{w} - \bar{\mathbf{w}})^T \mathbf{A} (\mathbf{w} - \bar{\mathbf{w}})) = \mathcal{N}(\bar{\mathbf{w}}, \mathbf{A}^{-1}),$$

with $\mathbf{A} = \sigma_n^{-2} \boldsymbol{\Phi} \boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_p^{-1}$ and $\bar{\mathbf{w}} = \sigma_n^{-2} \mathbf{A}^{-1} \boldsymbol{\Phi} \mathbf{y}$. To make a prediction $f(\mathbf{x}_*)$ for a new input $\mathbf{x}_*$ the outputs of all linear models are averaged and additionally weighted by their posterior. In so doing, the predicted mean $\bar{f}_*$ and predicted variance $\mathbf{V}_*$ can be given as follow [12]

$$\begin{aligned} \bar{f}_* &= {\mathbf{k}_*}^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y} = {\mathbf{k}_*}^T \boldsymbol{\zeta}\,, \\ \mathbf{V}_* &= \mathbf{k}(\mathbf{x}_*, \mathbf{x}_*) - {\mathbf{k}_*}^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{k}_*\,, \end{aligned} \tag{4}$$

where $\mathbf{k}_* = {\boldsymbol{\phi}_*}^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi}$, $\mathbf{k}(\mathbf{x}_*, \mathbf{x}_*) = {\boldsymbol{\phi}_*}^T \boldsymbol{\Sigma}_p \boldsymbol{\phi}_*$ and $\mathbf{K} = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi}$. Equation (4) represents the key equation for the Gaussian process, in which $\bar{f}_*$ gives the predicted value for an observed input x_* and $\mathbf{V}_*$ the corresponding uncertainty of the prediction. In opposite of LWPR, GPR is a global method, since every training point is included in the prediction vector $\boldsymbol{\zeta}$ and hence has a 'global' influence on prediction behavior.

III. Computed Torque Control Techniques

The main concept of computed torque control is to compute controller command regarding a priori knowledge about the system expressed in a dynamics model [10], [11], [15]. In robot control two control methods are often used which will

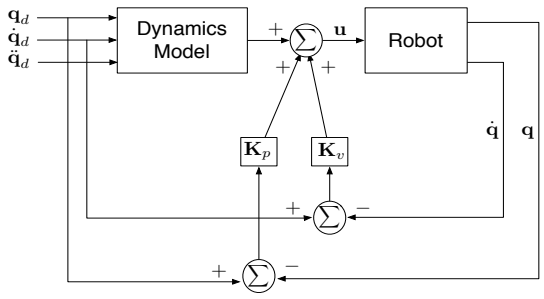

Fig. 2: Feedworward nonlinear control

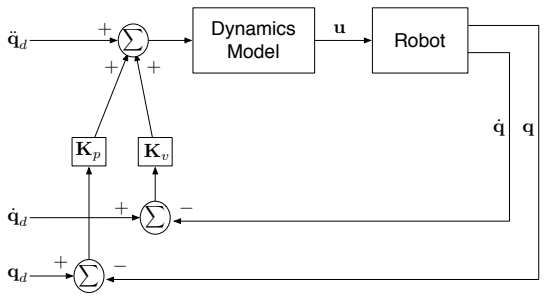

Fig. 3: Inverse dynamics control

be discussed in next sections, i.e., (i) feedforward nonlinear control, (ii) inverse dynamics control [10], [11].

The traditional way to obtain a model for the system is to analyze the physical properties and subsequently derive its equation of motion. In so doing, the robot dynamics can be modeled as a rigid body dynamics system

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{u}\,, \tag{5}$$

where $\mathbf{q},\ \dot{\mathbf{q}},\ \ddot{\mathbf{q}}$ are joint angles, velocities and accelerations of the robot, $\mathbf{u}$ denotes the inputs to the system, $\mathbf{M}(\mathbf{q})$ the inertia matrix of the robot, and $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ all the forces acting on the system (e.g., Coriolis forces, centripetal force, gravity, friction, etc.). The system input $\mathbf{u}$ is in our case joint torques given as motor command which can be computed in various ways depending on control method. Usually, $\mathbf{u}$ is computed in such a way that the robot is able to track some desired trajectories. It should be noted that in this case we are concerned with control problem in joint space. That means, the desired trajectories will be given as desired joint angles, velocities and accelerations.

A. Feedforward Nonlinear Control

Regarding the dynamics described in Equation (5), the intention of feedforward is eliminating the nonlinearity in the dynamics as shown in Figure 2. With the linearized system the tracking task can be conducted by a simple PD controller. For this, the controller command $\mathbf{u}$ is chosen as [11]

$$\mathbf{u} = \mathbf{M}(\mathbf{q}_d)\,\ddot{\mathbf{q}}_d + \mathbf{F}(\mathbf{q}_d, \dot{\mathbf{q}}_d) + \mathbf{K}_p\mathbf{e} + \mathbf{K}_v\dot{\mathbf{e}}\,, \tag{6}$$

where $\mathbf{q}_d,\ \dot{\mathbf{q}}_d,\ \ddot{\mathbf{q}}_d$ denote desired joint angles, velocities and accelerations. Combining (6) with the rigid dynamics (5), we have an error equation for the closed-loop: $\ddot{\mathbf{e}} + \mathbf{M}^{-1}\mathbf{K}_v\dot{\mathbf{e}} + \mathbf{M}^{-1}\mathbf{K}_p\mathbf{e} = \mathbf{0}$, where $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$. The feedback gains $\mathbf{K}_p$ and $\mathbf{K}_v$ can now be chosen, so that the error equation is stable [11]. In case of using regression technique for model approximation, i.e., the dynamics model is function of $\mathbf{q}_d,\ \dot{\mathbf{q}}_d,\ \ddot{\mathbf{q}}_d$, the Equation (6) can generally be written as

$$\mathbf{u} = \mathbf{f}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) + \mathbf{K}_p\mathbf{e} + \mathbf{K}_v\dot{\mathbf{e}}\,, \tag{7}$$

where $\mathbf{f}(\bullet)$ is approximated by an appropriate regression method.

B. Inverse Dynamics Control

Another method for cancelling nonlinearities in the dynamics is applying inverse dynamics approach which is illustrated in Figure 3. In this case, the controller command $\mathbf{u}$ is given by [11]

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}}_{\text{ref}} + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})\,, \tag{8}$$

with $\ddot{\mathbf{q}}_{\text{ref}} = \ddot{\mathbf{q}}_d + \mathbf{K}_p\mathbf{e} + \mathbf{K}_v\dot{\mathbf{e}}$. Considering the Equations (5) and (8), the closed-lood error results in: $\ddot{\mathbf{e}} + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e} = \mathbf{0}$, which is stable with an appropriate choice of $\mathbf{K}_v$ and $\mathbf{K}_p$ [10], [11]. For regression case, the controller command in (8) is computed as follow

$$\mathbf{u} = \mathbf{f}\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_{\text{ref}}\right)\,. \tag{9}$$

The function $\mathbf{f}(\bullet)$ can be learned online or offline. In case of online approximation we have an adaptive dynamics model, with which the time dependency of the system can be regarded.

IV. Model Approximation Comparison

In this section, we will compare the approximation performance of GPR and LWPR using (i) simulation data and (ii) real SARCOS robot data. Generating the simulation data, we use an analytical model of the 7-DoF SARCOS robot arm modeled with the SL-software package [16]. In so doing, we produce for each joint the corresponding torque given joint angles, velocities and accelerations.

A. Approximation of Simulation Data

For the simulation we choose the joint angles q_i of the robot such that the relationship between the inputs, i.e., joint angles, velocities and accelerations, and the joint torques are sufficiently nonlinear. In this example, q_i consists of two sinusoids which have different frequencies and amplitudes for each joint: $q_i(t) = A\sin(2\pi f_{1i}t) + A/3\sin(2\pi f_{2i}t)$.

The Table V in the appendix shows the values taken for q_i. In so doing, a training set and a test set with 21 inputs and 7 targets are generated which consist of 14094 examples for training and 5560 for testing. The training takes place for each DoF separately employing GPR and LWPR. Table I gives the normalized mean squared error (nMSE) in percent of the evaluation on the test set, where the normalized mean squared error is defined as: nMSE $=$ Mean squared error/Variance of target.

It can be seen that GPR gives better approximation compared to LWPR, since GPR is a global method. A further

nMSE [%]	Joint [i]						
	1	2	3	4	5	6	7
LWPR	3.9	1.6	2.1	3.1	1.7	2.1	3.1
GPR	0.7	0.2	0.1	0.5	0.1	0.4	0.6

TABLE I: Approximation error for each DoF using simulation data. GPR provides a better approximation resulting in smaller nMSE.

nMSE [%]	Joint [i]						
	1	2	3	4	5	6	7
LWPR	1.7	2.1	2.0	0.5	2.5	2.4	0.7
GPR	0.5	0.3	0.1	0.1	1.5	1.2	0.2
Ana. model	5.9	226.3	111.3	3.4	2.7	1.3	1.4

TABLE II: Approximation error for each DoF using real SARCOS data. LWPR as well as GPR show a good learning performance with real data. nMSE of the analytical model determined by linear regression is large for 2. and 3. DoF, which indicates that for these DoF the analytical model fails to explain the data.

advantage of GPR is that there are only some hyperparameters to be determined. As our input dimension is 21 and the Gaussian kernel is used, there are 23 hyperparameters necessary for each DoF which can be calculated using optimization algorithms [12], [13]. Thus, GPR approximation can be mostly automated. However, the main drawback is the computational cost. In general, the training time for GPR is about 2 time longer compared to LWPR.

The advantage of LWPR is the fast computation, since the model update is done locally. Thus, LWPR is more capable for a real-time application. But due to many meta parameters which have to be set manually for the training, it is fairly tedious to find an optimal setting for those by trial-and-error.

B. Approximation of Real Robot Data

The data is taken from the real 7-DoF SARCOS robot arm as shown in Figure 1. Here, we have 13622 examples for the training set and 5500 for the test set. Table II shows the nMSE in percent after learning with real robot data for each DoF. Additionally, we also calculate the nMSE of a linear regression using the analytical robot model. The resulting error will indicate, how far the analytical model can explain the data.

As seen in Table II, both LWPR as well as GPR show a good ability on approximation with real data considering noisy components which are unavoidable in this case. Compared to LWPR, GPR provides better approximation for every DoF.

Considering the analytical model, the linear regression yields very large approximation error for the 2. and 3. DoF. Apparently, for these DoF the nonlinearities (e.g., hydraulic cables, complex friction) cannot be approximated well using just the rigid body functions. This example shows the difficulty using the analytical model for control in practice. The imprecise dynamics model will result in poor control performance for real system, e.g., large tracking error. This is the main reason why model-based control is not widely used in robotics. Here, the approximated models present a better approach [5].

V. Control Performance Comparison

The dynamics model is learned with trajectories given in Table V in the appendix which are used as desired trajectories during the tracking afterwards. In addition, some test trajectories are also generated in order to compare the generalization ability of each approximation method. For this purpose, the frequency f_2, the amplitudes A are modified and a phase ψ is accessorily added to some sinussoids, as shown in the Table VI.

nMSE [%]	Analy. model		GPR model		LWPR model		PD with
Joint [i]	FF	IDM	FF	IDM	FF	IDM	g. comp.
1	0.17	0.43	0.16	0.12	0.37	0.38	10.1
2	0.69	0.86	0.73	0.74	0.74	0.73	19.7
3	0.24	0.45	0.18	0.31	0.29	0.56	8.48
4	0.55	0.10	0.53	0.77	0.65	0.99	7.14
5	0.10	0.23	0.07	0.17	0.16	0.64	3.91
6	0.42	1.06	0.23	0.56	0.40	1.11	17.3
7	0.35	2.72	0.34	2.01	0.42	2.94	4.24

TABLE III: Tracking error as nMSE for each DoF using training trajectories. The control performance using analytical model yields very good results, since the tracking is done in simulation and thus has an *exact* analytical model. The tracking error in case of GPR model is mostly smaller than LWPR due to better model approximation. It can also be seen that controllers with learned model generally provide better results compared to traditional PD approach with gravity compensation.

nMSE [%]	Analy. model		GPR model		LWPR model		PD with
Joint [i]	FF	IDM	FF	IDM	FF	IDM	g. comp.
1	0.26	0.68	0.78	0.59	1.45	1.55	15.3
2	0.19	0.35	1.05	1.09	0.63	0.63	7.54
3	0.06	0.16	0.24	0.55	0.19	0.37	1.75
4	1.97	3.02	2.42	3.77	3.24	5.99	7.69
5	0.09	0.18	0.23	0.58	0.23	0.61	1.98
6	0.07	0.27	0.31	1.13	0.29	0.85	2.61
7	0.16	1.21	0.23	2.12	0.26	2.71	1.66

TABLE IV: Tracking error as nMSE for each DoF using test trajectories. The errors show that GPR and LWPR are able to generalize the learned model well for trajectories similar to training trajectories. Due to the lower frequency f_2 in this case, i.e., smaller tracking velocity, the analytical model yields smaller errors compared to the training case. Inverse dynamics control gives a larger error compared to feedforward control, since IDM is more sensitive against model inaccuracy.

Computed torque approaches introduced in Section III are applied to control the model of the 7-DoF SARCOS robot arm accomplishing a tracking task. For evaluation, the tracking error of each DoF is calculated. Table III shows the tracking error of each joint as nMSE in percent for the training trajectories. Beside the feedforward (FF) and inverse dynamics control (IDM), a PD controller with gravity compensation is also computed for comparison.

It's necessary to emphasize that the control task is done in real-time where desired trajectories are sampled with 480 Hz. The simulation is conducted on a Macintosh personal computer with a 2.2 GHz Intel processor.

As shown in Table IV, the control performance using analytical model yields very good results as expected, since the analytical model corresponds *exactly* the dynamics model in this case. Comparing control quality using GPR and LWPR models, nMSE of GPR model is mostly smaller due to better model approximation. It can also be seen that controllers with learned model generally provide better results compared to traditional PD approach with gravity compensation

For LWPR we are able to calculate the controller command $\mathbf{u}$ for every sample, since evaluation of the prediction values (1) is quite fast. In contrast, with GPR the controller command is updated at every 4th sampling step due to more involved calculations. Before tracking task, the prediction vector $\boldsymbol{\zeta}$ is loaded from learned GPR model. The torque prediction is then accomplished by online calculation of covariance vector $\mathbf{k}_*$ and subsequently the predicted mean as given in Equation (4). This operation has the order of $\mathcal{O}(mn)$, where m is the number of robot joints and n the number of training points, respectively.

Table IV gives the normalized mean squared error of each joint using test trajectories. It can be seen that the tracking error of GPR and LWPR is slightly larger than in case of training trajectories. This fact shows that the learned models are able to generalize well for trajectories similar to training trajectories. Figures 4 and 5 show the tracking performance of each joint in case of feedforward control with test trajectories.

Considering the different control approaches, the feedforward provides the best tracking performance. A further advantage is that the dynamics model is outside of the loop, and thus enables a fast inner servo-loop and a slower outside-loop for torque prediction. Thus, in case of involved torque calculation the feedforward control is more capable for a real-time application. In contrast, the dynamics model is inside of the loop in case of inverse dynamics control. If the model is exact, the nonlinearities will be canceled out completely. In case of imperfect match between model and system, this setting is sensitive against model error, since the error is not directly corrected by the closed-loop. Furthermore, using inverse dynamics the torque computation has to be as fast as the servo-loop which is prohibitive for time-consuming torque calculation.

As shown in the Figures 4 and 5, the PD controller with gravity compensation cannot provide satisfactory results for several DoFs (e.g., 1. and 2. DoF), even though the gravity compensation is determined using the *exact* dynamics model here. Furthermore, for the PD controller high-gains have to be taken in order to achieve competitive performance. For example, in this case the gains for the PD controller are about 5 times larger than those used for FF and IDM controller. However, high-gain feedback can cause several problems, such as saturation of the actuators, excitation of the unmodeled dynamics, large error in present of noise etc., and thus should be avoided in practice.

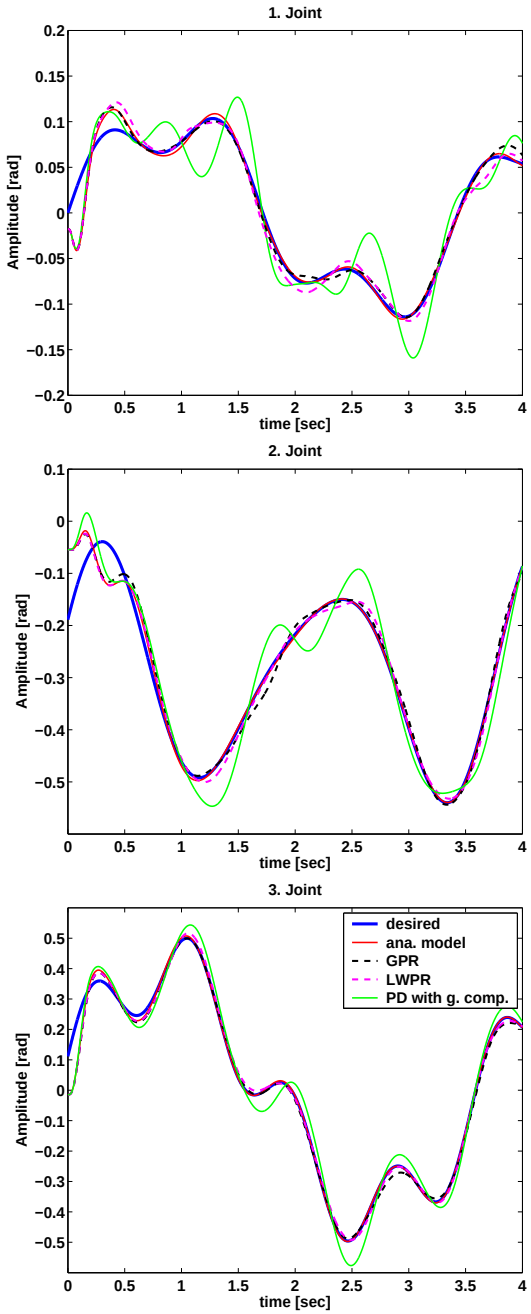

Fig. 4: Tracking performance for joint 1, 2 and 3 with feedforward control using test trajectory.

VI. Conclusion

In this paper, we compare two regression methods, i.e., LWPR and GPR, for approximation of the dynamics model in the setting of computed torque robot control. Hereunto, the dynamics model was learned offline and subsequently applied for an online torque evaluation in real-time control. Two control approaches, i.e., feedforward and inverse dynamics control, were used for real-time tracking of a model of SARCOS robot arm. It can be shown that with the approximated model the control quality can be significantly improved towards a traditional PD controller with gravity compensation.

Our results indicate that GPR can be made to work for control applications in real-time, and that it is easier to apply to learning problems and achieves a slightly higher accuracy compared to LWPR. However, the computational cost is prohibitively high for online learning. Our next step is to modify GPR, so that it can be applied for an online regression and thus is capable for real-time learning. Here, the problem of expensive computation has to be overcome using other techniques, such as sparse or local GPR models.

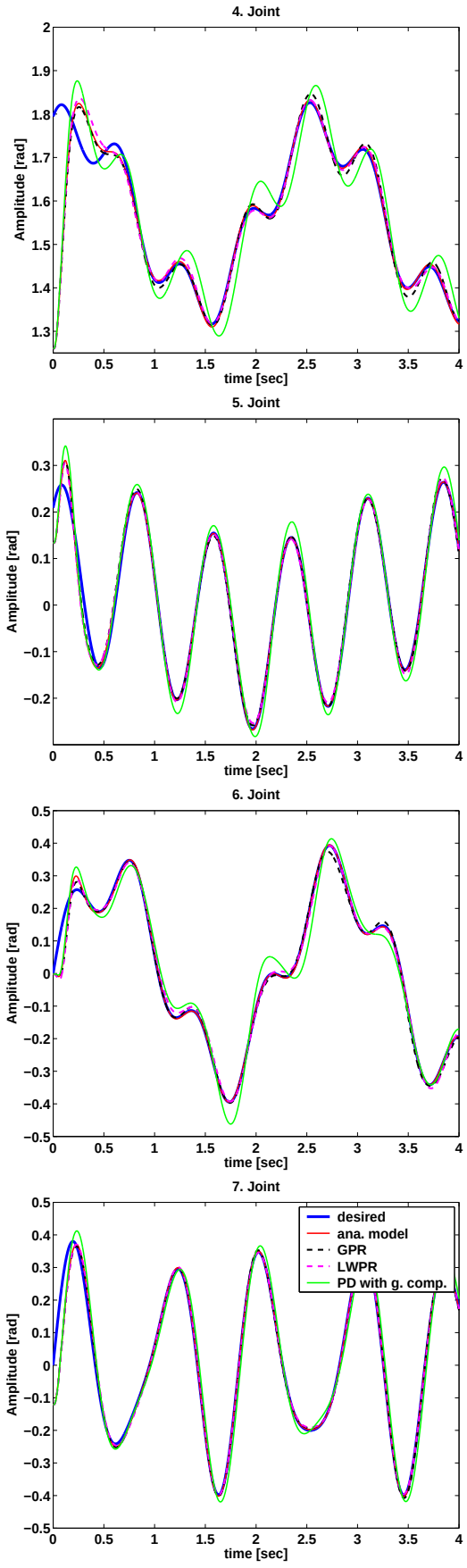

Fig. 5: Tracking performance for joint 4, 5, 6 and 7 with feedforward control using test trajectory.

APPENDIX

Joint [i]	$A\ [rad]$	$f_1\ [Hz]$	$f_2\ [Hz]$
1	0.2	$0.2\sqrt{2}$	1.1
2	0.2	$0.3\sqrt{3}$	2.3
3	0.3	$0.1\sqrt{7}$	2.2
4	0.3	$0.1\sqrt{17}$	1.7
5	0.3	$0.4\sqrt{11}$	1.9
6	0.25	$0.2\sqrt{5}$	2.7
7	0.2	$0.3\sqrt{13}$	2.7

TABLE V: Values taken for function approximation and training trajectories.

Joint [i]	$A\ [rad]$	$f_1\ [Hz]$	$f_2\ [Hz]$	$\psi\ [rad]$
1	0.1	$0.2\sqrt{2}$	0.9	0.0
2	0.2	$0.3\sqrt{3}$	0.77	0.43
3	0.4	$0.1\sqrt{7}$	1.12	0.21
4	0.2	$0.1\sqrt{17}$	1.63	1.0
5	0.2	$0.4\sqrt{11}$	0.3	0.9
6	0.3	$0.2\sqrt{5}$	1.57	0.0
7	0.3	$0.3\sqrt{13}$	1.67	0.0

TABLE VI: Values taken for test trajectories.

REFERENCES

[1] J. Nakanishi, J. A. Farrell, and S. Schaal, "Composite adaptive control with locally weighted statistical learning," *Neural Networks*, 2005.

[2] J. Nakanishi and S. Schaal, "Feedback error learning and nonlinear adaptive control," *Neural Networks*, 2004.

[3] E. Burdet and A. Codourey, "Evaluation of parametric and nonparametric nonlinear adaptive controllers," *Robotica*, vol. 16, no. 1, pp. 59–73, 1998.

[4] E. Burdet, B. Sprenger, and A. Codourey, "Experiments in nonlinear adaptive control," *International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 537–542, 1997.

[5] J. A. Farrell and M. M. Polycarpou, *Adaptive Approximation Based Control*. New Jersey: John Wiley and Sons, 2006.

[6] G. Gregorcic and G. Lightbody, "Internal model control based on a gaussian process prior model," *Proceedings of the American Control Conference*, 2003.

[7] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard, "Gaussian process model based predictive control," *Proceeding of the American Control Conference*, 2004.

[8] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space," *International Conference on Machine Learning, Proceedings of the Sixteenth Conference*, 2000.

[9] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparameteric statistics for real-time robot learning," pp. 49–60, 2002.

[10] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Dynamics and Control*. New York: John Wiley and Sons, 2006.

[11] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Prentice Hall, 2004.

[12] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. Massachusetts Institute of Technology: MIT-Press, 2006.

[13] M. Seeger, "Gaussian processes for machine learning," *International Journal of Neural Systems*, 2004.

[14] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT-Press, 2002.

[15] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. New Jersey: Prentice Hall, 1991.

[16] S. Schaal, "The SL simulation and real-time control software package." University of Southern California. [Online]. Available: http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf