Learning Task-Space Tracking Control with Kernels

Duy Nguyen-Tuong[1], Jan Peters[2]
Max Planck Institute for Intelligent Systems, Spemannstraße 38, 72076 Tübingen[1,2]
Universität Darmstadt, Intelligent Autonomous Systems, Hochschulstraße 10, 64289 Darmstadt[2]

***Abstract*— Task-space tracking control is essential for robot manipulation. In practice, task-space control of redundant robot systems is known to be susceptive to modeling errors. Here, data driven learning methods may present an interesting alternative approach. However, learning models for task-space tracking control from sampled data is an ill-posed problem. In particular, the same input data point can yield many different output values which can form a non-convex solution space. Because the problem is ill-posed, models cannot be learned from such data using common regression methods. While learning of task-space control mappings is *globally* ill-posed, it has been shown in recent work that it is *locally* a well-defined problem. In this paper, we use this insight to formulate a local kernel-based learning approach for online model learning for task-space tracking control. For evaluations, we show in simulation the ability of the method for online model learning for task-space tracking control of redundant robots.**

I. Introduction

Control of redundant robots in operational space, especially task-space tracking control, is an essential ability needed in robotics [1], [2]. Here, the robot's end-effector follows a desired trajectory in task-space, while distributing the resulting forces onto the robot's joints. Analytical formulation of task-space control laws requires given kinematic and dynamic models of the robot. However, modeling the kinematics and dynamics is susceptive to errors. For example, accurate analytical dynamic models are hard to obtain for complex robot systems, due to many unknown nonlinearities resulting from friction or actuator dynamics [3]. One promising possibility to overcome such inaccurate hand-crafted models is to learn them from data. From a machine learning point of view, learning of such models can be understood as a regression problem. Given input and output data, the task is to learn a model describing the input to output mapping.

Using standard regression techniques, such as Gaussian process regression [4], support vector regression [5] or locally weighted regression [6], a model can be approximated to describe a *single-valued* mapping (i.e., one-to-one) between the input and output data. The single-valued property requires that the same input point should always yield the same single output value, resulting in a well-defined learning problem. However, this situation changes when learning a torque prediction model for task-space tracking control of redundant robots. Here, we are confronted with the problem of learning *multi-valued* or one-to-many mappings. In this case, standard regression techniques can not be applied. Naively learning such multi-valued mappings from sampled data using standard regression will average over multiple output values in a potentially non-convex solution space [7]. Thus, the resulting model will output degenerate predictions, which lead to poor control performance and may cause damage to the redundant robot.

However, despite being a globally ill-posed problem, learning such task-space control mapping is locally well-defined [7], [8]. In this paper, we employ this insight to formulate an online local learning approach, appropriate for learning models that allow prediction with such multi-valued mappings. The key idea is to localize a *single* model in configuration space, while continuously updating this model online by including new data points and, eventually, removing old points. Here, local data points are inserted or removed based on a kernel distance measure. Due to the local consistency, a prediction model can be learned. The proposed model parametrization allows us to apply the kernel-trick and, therefore, enables a formulation within the kernel learning framework [5]. Kernel methods have been shown to be a flexible and powerful tool for learning general nonlinear models. In task-space tracking control of redundant robots, the model parametrization enables a projection of the joint-space stabilization torques into the task's null-space.

The remainder of the paper will be organized as follows: first, we give a brief overview of task-space control and provide a review of related work. In Section II, we describe our approach to learn task-space tracking control. The proposed method will be evaluated on redundant robot systems in simulation, e.g., a simulated 3-DoF robot and 7-DoF Barrett WAM, for task-space tracking control in Section III. The most important lessons from this research project will be summarized in Section IV.

A. Problem Statement

To obtain an analytical task-space control law, we first need to model the robot's kinematics [1]. The relationship between the task-space and the joint-space of the robot is usually given by the forward kinematic model $\boldsymbol{x} = f(\boldsymbol{q})$. Here, $\boldsymbol{q} \in \mathbb{R}^m$ denotes the robot's configuration in the joint-space and $\boldsymbol{x} \in \mathbb{R}^d$ represents the task-space position and orientation. For redundant robot systems, it is necessary that $m > d$. The task-space velocity and acceleration are $\dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{x}} = \dot{\mathbf{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \mathbf{J}(\boldsymbol{q})\ddot{\boldsymbol{q}}$, where $\mathbf{J}(\boldsymbol{q}) = \partial f / \partial \boldsymbol{q}$ is the Jacobian. For computing the joint torques necessary for the robot to follow the task-space trajectory, a model of the robot dynamics is required. A typical dynamic model can be given in the form of $\boldsymbol{u} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, see [9] for more details. Here, $\boldsymbol{u}$ denotes the joint torque, $\mathbf{M}(\mathbf{q})$

is the generalized inertia matrix of the robot, and $\mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a vector containing forces, such as gravity, centripetal and Coriolis forces. Combining the dynamic model with the kinematic model yields one possible operational space control law

$$\boldsymbol{u} = \mathbf{M}\mathbf{J}_w^{\dagger}(\ddot{\boldsymbol{x}}_{\mathrm{ref}} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \mathbf{F}\,, \tag{1}$$

where $\mathbf{J}_w^{\dagger}$ denotes the weighted pseudo-inverse of $\mathbf{J}$, as described in [3]. In Equation (1), a task-space attractor $\ddot{\boldsymbol{x}}_{\mathrm{ref}}$ is employed for tracking the actual task-space acceleration $\ddot{\boldsymbol{x}}$ [3]. Here, the task-space attractor is formulated as $\ddot{\boldsymbol{x}}_{\mathrm{ref}} = \ddot{\boldsymbol{x}}_{\mathrm{des}} + \mathbf{G}_{vv}(\dot{\boldsymbol{x}}_{\mathrm{des}} - \dot{\boldsymbol{x}}) + \mathbf{G}_{pp}(\boldsymbol{x}_{\mathrm{des}} - \boldsymbol{x})$, where $\boldsymbol{x}_{\mathrm{des}}$, $\dot{\boldsymbol{x}}_{\mathrm{des}}$ and $\ddot{\boldsymbol{x}}_{\mathrm{des}}$ denote the desired task-space trajectory. $\mathbf{G}_{vv}$ and $\mathbf{G}_{pp}$ are positive task-space gain matrices.

To ensure stable tracking in the robot's joint-space, the controller command $\boldsymbol{u}$ in Equation (1) is usually extended by a null-space controller term $\boldsymbol{u}_0$. Thus, the total joint torque command $\boldsymbol{u}_{\mathrm{joint}}$ is given as

$$\boldsymbol{u}_{\mathrm{joint}} = \boldsymbol{u} + \left(\mathbf{I} - \mathbf{J}_w^{\dagger}\mathbf{J}\right)\boldsymbol{u}_0\,. \tag{2}$$

The term $\boldsymbol{u}_0$ can be interpreted as joint-space stabilizing torques which are only effective in the task's null-space and, thus, do not interfere with the task achievement [3]. The null-space controller command $\boldsymbol{u}_0$ can be chosen such that the redundant robot is pulled towards a desired rest posture $\boldsymbol{q}_{\mathrm{rest}}$, i.e., $\boldsymbol{u}_0 = -\mathbf{G}_v\dot{\boldsymbol{q}} - \mathbf{G}_p(\boldsymbol{q} - \boldsymbol{q}_{\mathrm{rest}})$, where $\mathbf{G}_p$ and $\mathbf{G}_v$ are positive joint-space gain matrices.

As indicated by Equations (1, 2), an analytical formulation for task-space control requires given analytical kinematic and dynamic models. As modeling these relationships can be inaccurate in practice, model learning presents a promising alternative. In the task-space tracking problem shown in Equation (1), we want to learn mappings from inputs $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{x}})$ to targets $\boldsymbol{u}$. However, this mapping is one-to-many [7], [8], as there can be many torques $\boldsymbol{u}$ which correspond to the same task-space acceleration $\ddot{\boldsymbol{x}}$ given $\boldsymbol{q}, \dot{\boldsymbol{q}}$. Thus, naively learning a task-space control model for redundant robots from sampled data may result in a degenerate mapping. In practice, such degenerate models will provide inconsistent torque predictions.

B. Related Work

Learning multi-valued mappings has previously been investigated in the field of neural motor control [10], [11] and learning inverse kinematics [8], [12]. In [10], the multi-valued relationship is resolved for a particular output solution by jointly approximating the forward and inverse mapping. Originally, the introduced forward-inverse learning principle has been formulated in the framework of neural networks [10]. In a neural networks based implementation, the forward model is chained with the multi-valued inverse model, where the prediction errors made by the forward model are used to adapt the weight values of the inverse model for a given output solution. However, training such neural networks is well-known to be problematic due to local minima, instability and difficulties in selecting the network structures. Nevertheless, this framework of learning forward and inverse models initiated a number of follow-up research projects, such as [11], [13]. For example, in [13] considerable evidence was presented indicating that the forward-inverse models approach may explain human motor control. In [11], the authors approximate pairwise forward-inverse models for different motor control tasks and, subsequently, combine them for prediction.

In the broader sense, the pairwise forward-inverse model approach [11] can be understood as a *local* learning method, where the data is first partitioned into local regions for which local forward-inverse model pairs are subsequently approximated. A local learning approach is also employed in [14] and [8] to learn models for robot inverse kinematics, where locally weighted regression techniques are used. The locally weighted regression approach has been further extended for learning operational space robot control [7]. While Peters et al. [7] attempt to learn a direct mapping for predicting the joint torques for control, Salaun et al. [15] first learn a forward kinematic model and invert the learned model afterwards. Subsequently, they combine it with an inverse dynamic model to generate the required torque command.

Compared to previous local learning approaches, we attempt to learn a *single* localized model, while continuously updating this local model depending on the robot's current configuration. Due to the local consistency, the model learning problem is well-defined. We propose a model parameterization which enables kernel-based learning of torque prediction models for task-space tracking control. The model parametrization also allows a null-space projection, which is necessary to stabilize the robot in the joint-space without interfering with the task-space performance.

II. Learning Task-Space Tracking with Kernels

In this paper, we want to learn the mapping from inputs $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{x}})$ to outputs $\boldsymbol{u}$, similar to the one described by Equation (1). This mapping is subsequently used for predicting the outputs for given query points. As such one-to-many mappings are locally well-defined [7], [8], they can be approximated with a local kernel learning approach. Here, our model will be localized in the robot's joint position space. The local data is incrementally updated, as the robot moves to new state space regions. Every local data point is weighted by its distance to the most recent joint position. Thereby, we ensure that the local data points form a well-defined set that is appropriate for model learning. Using the weighted local data points, the model's parameters can be obtained by minimizing a cost function. To place the model into the kernel learning framework, we propose a model parametrization appropriate for the application of the kernel-trick. The parametrization is also suitable for robot tracking control in the task-space.

In the following sections, we will describe how the model is localized and updated in an online setting. We present the parametrization of the local model and show how the corresponding parameters can be obtained from data. Subsequently, we show how the learned local model can be used in online learning for task-space robot control.

A. Model Localization

For learning task-space tracking, we use a *single* local model for torque prediction, where the model is localized in the robot's joint position space. This local data set needs to be continuously updated in the online setting, as the robot frequently moves to new state space regions. In this section, we describe the measures needed to localize and update the model during online learning. The procedure includes insertion of new data points into the local data set and removal of old ones.

1) Insertion of New Data Points: For deciding whether to insert a new data point into the local data set, we consider the distance measure δ as proposed in [16] and [17]. This measure is defined by

$$\delta(\boldsymbol{q}^*) = k(\boldsymbol{q}^*, \boldsymbol{q}^*) - \boldsymbol{k}^T \boldsymbol{K}_a^{-1} \boldsymbol{k}\,. \tag{3}$$

where $k(\cdot,\cdot)$ denotes a kernel, $\boldsymbol{K}_a = k(\boldsymbol{L}, \boldsymbol{L})$ is the kernel matrix evaluated for the local joint positions $\boldsymbol{L} = \{\boldsymbol{q}_i\}_{i=1}^N$ and $\boldsymbol{k} = k(\boldsymbol{L}, \boldsymbol{q}^*)$ is the kernel vector [17]. The value δ describes the distance of a point $\boldsymbol{q}^*$ to the surface defined by $\boldsymbol{L}$ in the joint-space. This value increases with the distance of $\boldsymbol{q}^*$ from the surface $\boldsymbol{L}$ [17].

Using Equation (3), we can make decisions for inserting new data points. If the δ values of new data points exceed a given threshold η, we will insert these points into the local model. The employed measure δ ensures that new data points will be included into the local set, when the robot moves to new joint-space regions.

2) Removal of Old Data Points: For removing data points from the local set, we select the point which is the *farthest* from the most recent joint position $\boldsymbol{q}$. Here, we employ a Gaussian kernel as a distance measure between $\boldsymbol{q}$ and other local data points $\boldsymbol{q}_i$

$$k\left(\boldsymbol{q}, \boldsymbol{q}_i\right) = \exp\left(-\frac{1}{2}(\boldsymbol{q}-\boldsymbol{q}_i)^T \mathbf{W} (\boldsymbol{q}-\boldsymbol{q}_i)\right)\,, \tag{4}$$

where $\mathbf{W}$ denotes the kernel width. Removing the farthest local data point implies that its kernel measure $k(\cdot,\cdot)$ is the smallest. By continuously inserting and removing local data points, we make sure that the local data set is suitable for the current region of the state space.

B. Model Parametrization

The described insertion and removal operations in preceding section result in a data set localized in the joint position space. Due to the local consistency, model learning using this data is well-defined. Given the sampled local data set $\boldsymbol{D} = \{\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{x}}_i, \boldsymbol{u}_i\}_{i=1}^N$, we can now learn a model for torque prediction for task-space control.

From Equation (1), we can see that the joint torque $\boldsymbol{u}$ is *linear* in the task-space acceleration $\ddot{\boldsymbol{x}}$, while it is *nonlinear* in the joint position $\boldsymbol{q}$ and velocity $\dot{\boldsymbol{q}}$. Using this insight, we propose the following parametrization for the local model

$$\boldsymbol{u} = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\theta}_0^T \ddot{\boldsymbol{x}}\,, \tag{5}$$

where $\boldsymbol{\phi}$ is a vector containing nonlinear functions projecting $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$ into some high-dimensional spaces. Generally, $\ddot{\boldsymbol{x}}$ can have d dimensions and $\boldsymbol{u}$ is a m dimensional vector. Following the representer theorem [5], the coefficients $\boldsymbol{\theta}, \boldsymbol{\theta}_0$ in Equation (5) can be expanded in term of N local data points. Hence, we have

$$\boldsymbol{\theta} = \textstyle\sum_{i=1}^N \alpha_i \boldsymbol{\phi}(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)\,, \quad \boldsymbol{\theta}_0 = \textstyle\sum_{i=1}^N \alpha_0^i \ddot{\boldsymbol{x}}_i\,,$$

where α_i, α_0^i are the corresponding linear expansion coefficients. Inserting the linear expansions into Equation (5) and re-writing it in term of N sample data points yields

$$\boldsymbol{U} = \boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{P}\boldsymbol{\alpha}_0\,. \tag{6}$$

Here, the elements $[\boldsymbol{K}]_{ij} = \langle \boldsymbol{\phi}(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i), \boldsymbol{\phi}(\boldsymbol{q}_j, \dot{\boldsymbol{q}}_j) \rangle$ are the pairwise inner-products of the feature vectors. Thus, $[\boldsymbol{K}]_{ij}$ can be represented with kernels [5], i.e., $[\boldsymbol{K}]_{ij} = \tilde{k}([\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i], [\boldsymbol{q}_j, \dot{\boldsymbol{q}}_j])$. The matrix $\boldsymbol{K}$ is thus a kernel matrix evaluated at the joint position and velocity employing the kernel $\tilde{k}(\cdot,\cdot)$. Using this so-called kernel-trick, only the kernel function $\tilde{k}$ needs to be determined instead of an explicit feature mapping $\boldsymbol{\phi}$ [5]. Similarly, the elements $[\boldsymbol{P}]_{ij} = \langle \ddot{\boldsymbol{x}}_i, \ddot{\boldsymbol{x}}_j \rangle$ represent the pairwise inner-products of the task-space acceleration $\ddot{\boldsymbol{x}}$. Thus, $\boldsymbol{P}$ can be understood as a kernel matrix where linear kernels are applied. In Equation (6), the matrix $\boldsymbol{U}$ is given by $\boldsymbol{U} = \{\boldsymbol{u}_i\}_{i=1}^N$.

C. Online Learning of Local Model

Learning requires the estimation of the expansion parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}_0$ in Equation (6) from the local data set. Employing the learned model, we can predict the output for a query point. In particular, for online learning the expansion parameters have to be estimated incrementally, as the data arrives as a stream over time.

a) Estimation of Model Parameters.: Using the model parametrization in Section II-B, the expansion parameters can be estimated from data by minimizing an appropriate cost function $\mathcal{L}$ given by

$$\begin{aligned}\mathcal{L} = &\frac{\gamma}{2}\left(\boldsymbol{\alpha}^T \boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{\alpha}_0^T \boldsymbol{P}\boldsymbol{\alpha}_0\right) \\ &+ \frac{1}{2}\left(\boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{P}\boldsymbol{\alpha}_0 - \boldsymbol{U}\right)^T \mathbf{N}\left(\boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{P}\boldsymbol{\alpha}_0 - \boldsymbol{U}\right).\end{aligned} \tag{7}$$

The first term in Equation (7) acts as regularization, while the second term represents a squared loss based data-fit. In Equation (7), the parameter γ controls the regularization and the diagonal matrix $\mathbf{N}$ denotes the *weight* for each data point in the local set. The minimization of $\mathcal{L}$ w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}_0$ yields the analytical solution

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}_0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} + \gamma \mathbf{N}^{-1} & \boldsymbol{P} \\ \boldsymbol{K} & \boldsymbol{P} + \gamma \mathbf{N}^{-1} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{U} \end{bmatrix}. \tag{8}$$

The weighting metric $\mathbf{N}$ incorporates a distance measure of each local data point to the most recent point in the local set. Here, we employ a kernel distance measure in the joint position space, as given in Equation (4). The weighting metric $\mathbf{N}$ ensures that the local data will form a well-defined set appropriate for the model learning step. It should be noted that the Equation (7) is computed using *all* local data points.

Algorithm 1 Online learning of the local model.

Given: local data set $\boldsymbol{D}=\{\boldsymbol{q}_i,\dot{\boldsymbol{q}}_i,\ddot{\boldsymbol{x}}_i,\boldsymbol{u}_i\}_{i=1}^N$, $N_{\max}$, threshold value η.
Input: new input $\{\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{x}}\}$ and output $\boldsymbol{u}$.

Evaluate the distance of $\boldsymbol{q}$ to the surface defined by $\boldsymbol{L}=\{\boldsymbol{q}_i\}_{i=1}^N$ based on the measure $\delta(\boldsymbol{q})$ from Equation (3).
if $\delta(\boldsymbol{q})>\eta$ **then**
  for $i=1$ **to** N **do**
    Compute: $\mathbf{N}(i,i)=k(\boldsymbol{q},\boldsymbol{q}_i)$ using Equation (4).
  end for
  if $N<N_{\max}$ **then**
    Include the new point: $\boldsymbol{D}_{N+1}=\{\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{x}},\boldsymbol{u}\}$.
  else
    Find the farthest point: $j=\min_i \mathbf{N}(i,i)$.
    Replace the j-th data point by the query point: $\boldsymbol{D}_j=\{\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{x}},\boldsymbol{u}\}$.
  end if
  Update the expansion parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}_0$ incrementally using Equation (8), while re-weighting every local data point with the new distance metric $\mathbf{N}$.
end if

b) Online Model Learning.: As the data arrives continuously in the online setting, Equation (8) has to be updated incrementally. Such incremental updates require adjusting the corresponding row and column of the inverse matrix, i.e., a rank-one update of the inverse matrix [18], [19]. Additionally, every data point in the local set has to be re-weighted by its distance to the most current point after every insertion and removal step. In practice, we initialize the inverse matrix in Equation (8) as a diagonal matrix, where the number $N_{\max}$ of local data points is fixed. During online learning, the inverse matrix is first updated $N_{\max}$ times while filling up the local data set. Subsequently, old data points have to be removed when new points are inserted. The complete procedure for learning the local model is summarized in the Algorithm 1.

c) Prediction.: With the optimization results from Equation (8), the prediction $\hat{\boldsymbol{u}}$ for a query point $[\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{x}}_{\mathrm{ref}}]$ can be computed as

$$\hat{\boldsymbol{u}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{x}}_{\mathrm{ref}})=\boldsymbol{\alpha}^T\tilde{k}(\boldsymbol{Q},[\boldsymbol{q},\dot{\boldsymbol{q}}])+\boldsymbol{\alpha}_0^T\langle\ddot{\boldsymbol{X}},\ddot{\boldsymbol{x}}_{\mathrm{ref}}\rangle\,, \quad (9)$$

where $\ddot{\boldsymbol{X}}=\{\ddot{\boldsymbol{x}}_i\}_{i=1}^N$ and $\boldsymbol{Q}=\{\boldsymbol{q}_i,\dot{\boldsymbol{q}}_i\}_{i=1}^N$.

D. Using Local Model for Task-Space Control

Up to now, we have learned a well-defined local model to predict the joint torques required to drive the robot along a desired task-space trajectory. To ensure the local consistency, this local model is continuously updated depending on the current robot's configuration. However, even after obtaining a perfect prediction of the necessary torques, it is not clear whether the robot will be stable in the joint-space. Thus, we need to explore ways to stabilize the robot in the joint-space without interfering the task-space performance, as done in analytical task-space control (see Equation (2)). Here, the key idea is to *project* the stabilizing torques $\boldsymbol{u}_0$ into the null-space of the "task relevant" part.

From Equation (9), it can be seen that the second term is the task relevant part, as this term explicitly depends on $\ddot{\boldsymbol{x}}_{\mathrm{ref}}$. Therefore, for the robot joint-space stabilization, we can project the stabilization torques $\boldsymbol{u}_0$ into the null-space of this term. Hence, the total joint torque controller command $\boldsymbol{u}_{\mathrm{joint}}$ can be computed as

$$\begin{aligned}\boldsymbol{u}_{\mathrm{joint}}=\boldsymbol{\alpha}^T\tilde{k}(\boldsymbol{Q},[\boldsymbol{q},\dot{\boldsymbol{q}}])+\boldsymbol{\alpha}_0^T\langle\ddot{\boldsymbol{X}},\ddot{\boldsymbol{x}}_{\mathrm{ref}}\rangle \\ +(\mathbf{I}-\boldsymbol{H}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T)\boldsymbol{u}_0\,.\end{aligned} \quad (10)$$

The null-space projection is then given by the matrix $\boldsymbol{H}=\boldsymbol{\alpha}_0^T\ddot{\boldsymbol{X}}$. The resulting null-space projection allows joint-space stabilization based on $\boldsymbol{u}_0$ without interfering the task performance. The procedure for online torque prediction in task-space tracking control is summarized in the Algorithm 2.

Algorithm 2 Online prediction for task-space control.

Given: a rest posture $\boldsymbol{q}_{\mathrm{rest}}$, local data $\ddot{\boldsymbol{X}}=\{\ddot{\boldsymbol{x}}_i\}_{i=1}^N$ and $\boldsymbol{Q}=\{\boldsymbol{q}_i,\dot{\boldsymbol{q}}_i\}_{i=1}^N$, expansion parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}_0$.
Input: query point $\{\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{x}_{\mathrm{des}},\dot{\boldsymbol{x}}_{\mathrm{des}},\ddot{\boldsymbol{x}}_{\mathrm{des}}\}$.

Compute null-space control torque $\boldsymbol{u}_0$.
Compute null-space projection matrix $\boldsymbol{H}=\boldsymbol{\alpha}_0^T\ddot{\boldsymbol{X}}$.
Compute task-space attractor $\ddot{\boldsymbol{x}}_{\mathrm{ref}}$.
Compute joint torque control $\boldsymbol{u}_{\mathrm{joint}}$ as given in Equation (10).

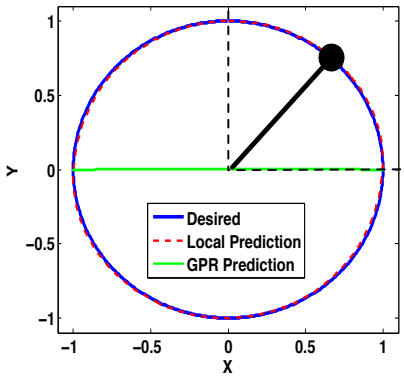
III. Evaluations

In this section, we evaluate the proposed approach for learning task-space control, as described in Section II. First, we show for a toy example how a non-unique function can be learned in the online setting using this local learning approach. This example further illustrates the basic idea behind the local learning principle when used for approximating a multi-valued mapping. Subsequently, we show the ability of our method in learning torque prediction models for task-space tracking control of redundant robot systems in simulation. The control experiments are performed with both simulated 3-DoF robot and 7-DoF anthropomorphic Barrett arm.

A. Online Learning of a Non-unique Function

As benchmark example, we create a one-dimensional non-unique function shown in Figure 1. In this example, there is a pendulum that can rotate in the $\boldsymbol{x}-\boldsymbol{y}$ plane. For a circular rotation, the trajectory of $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by $x_i=\sin(t_i)$ and $y_i=\cos(t_i)$ for t_i ranging from 0 to 2π. For the experiment, we sample 500 data points from the generated trajectory. If we employ $\boldsymbol{x}$ as input and $\boldsymbol{y}$ as the target output, we will have a non-unique prediction problem.

In this example, the parametrization of the local model is given by $\boldsymbol{y}=\boldsymbol{\theta}^T\boldsymbol{\phi}(\boldsymbol{x})$. While the model is localized in the $\boldsymbol{x}$ space, we update the local data set and learn the model in the online setting, as described in Section II. For online model learning, we incrementally feed the data to the algorithm. Figure 1 shows the results after one sweep through the data set. To highlight the difficulty in learning such multi-valued mappings from data, the well-known Gaussian process regression [4] is employed to globally approximate



(a) **Trajectory of a pendulum in the $x-y$ space**

Fig. 1: An example of learning a non-unique function. For the pendulum, we have for each input position x two possible output values. Naively learning a global mapping $\boldsymbol{x} \rightarrow \boldsymbol{y}$ using GPR [4] results in an average over multiple output solutions. However, when the mapping is learned locally within the vicinity of the query point in an online setting, the model learning problem is well-defined resulting in a proper prediction.

the mapping $\boldsymbol{x} \rightarrow \boldsymbol{y}$. The comparison between the two methods is given in Figure 1.

In the experiment, the size of the local data set is chosen to be 10. Here, we first fill the local set up incrementally and, subsequently, update the local model online by insertion and removal. We employ the Gaussian kernel for the localization step, as well as for model learning. The kernel width $\mathbf{W}$ is optimized by cross-validation, and the threshold η is set to be 0.001. As the farthest point in the input space is removed when a new point is inserted, one can observe that the local data set always covers a region in the vicinity of the recent query point. Since the local data set forms a convex solution space, the local model can be learned and results in a proper prediction of the targets $\boldsymbol{y}$, shown in Figure 1 (b).

B. Online Model Learning for Task-Space Tracking Control

In this section, we apply the proposed method to learning torque prediction models for task-space control of a simulated 3-DoF robot and the simulated 7-DoF Barrett WAM. In the experiments, the models are learned online, while the robots are controlled to track a task-space trajectory. Here, the task-space trajectory is given by the positions of the end-effector in Cartesian space. The tracking results in task-space for the 3-DoF robot and the Barrett WAM are shown in Figures 2 and 3, respectively. The figures show the tracking performance during the first 10 seconds.

In the experiment using the 3-DoF robot model shown in Figure 2, we compare the task-space tracking control performance, when employing the online learned model and the perfect analytical model. Using the perfect analytical model knowledge, the joint torques are computed as given in Equation (2). Thus, the robot performs perfect task-space tracking, as shown in Figure 2 (a). In this example, the rest posture is set to be $\boldsymbol{q}_{\text{rest}} = [-\pi/3, \pi/3, \pi/3]^T$. For the online learning of the task-space control model, the torque prediction is computed as given in Equation (10). The size of the local set is determined to be 30 and $\eta = 0.01$. Here, we employ a Gaussian kernel, where the kernel width $\mathbf{W}$ is optimized beforehand. As shown in Figure 2 (b), the predicted joint torques converge to the perfect analytical torques after a short transient phase. As a result, the robot achieves good task-space tracking performance after a few seconds of online model learning. In general, the size of the local model and η should be chosen according to the available computational power. The larger the model size, the more expensive is the incremental learning. Smaller η will lead to frequent updates of the local model which might not be beneficial for fast model learning.

In the next experiment, we employ the proposed approach to control the more complex 7-DoF Barrett WAM in simulation. Similar to the previous experiment, the robot is controlled to follow a figure-8 in task-space while learning the torque prediction model online. Here, the local set consists of 150 data points, $\eta = 0.05$ and a Gaussian kernel is used. During online learning, the model is incrementally updated 300 times. The results for the first 10 seconds are shown in Figure 3. It can be observed that the robot is able to follow the task-space trajectory well, while keeping the joint-space trajectory in the vicinity of the rest posture $\boldsymbol{q}_{\text{rest}}$.

IV. Conclusion

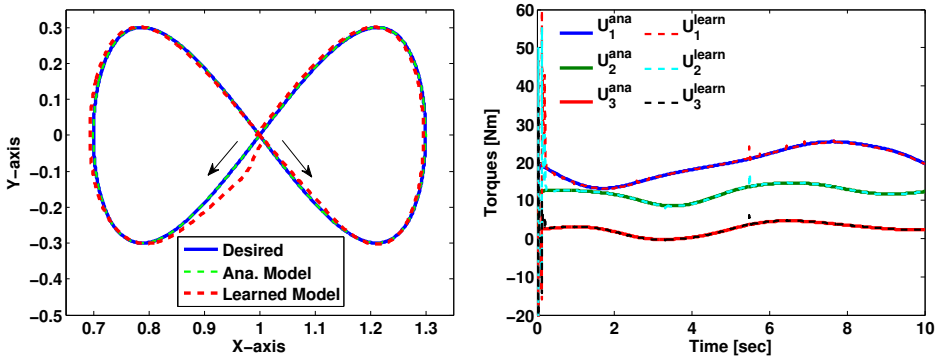
In this paper, we employed local, kernel-based learning for the online approximation of a multi-valued mapping. This approach is based on the key insight that an approximation of such mappings from data is globally an ill-posed problem, while it is locally well-defined. Our proposed method uses an online procedure for updating the local model by inserting and removing data points. We further proposed a parametrization for the local model that allows learning task-space tracking control. The update procedures and the model are formulated in the kernel framework, where the resulting parameters can be incrementally learned online. As evaluation, we showed that the approach was able to learn torque prediction models for task-space tracking of redundant robots in several setups. The experiments are performed both on a simulated 3-DoF robot and the simulated 7-DoF Barrett WAM. The results show that the presented kernel-based approach can be used to approximate multi-valued mappings for task-space tracking control. Future work includes the implementation on real robots and investigation in stability issues. Furthermore, practical problems need to be considered in more details, such as gain tuning and efficient implementation.

References

[1] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Journal of Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

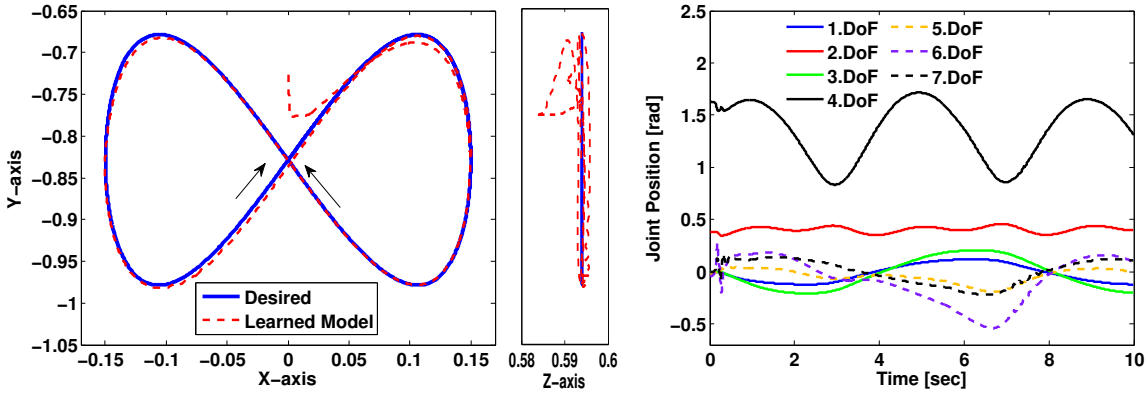
[2] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[3] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: a theoretical and emprical comparison," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.



(a) **Task-space tracking** (b) **Control torque for each DoF**

Fig. 2: Task-space tracking by a 3-DoF robot. Here, we compared task-space tracking using perfect analytical model with one that was learned online. As a perfect model is used, the analytical task-space tracking control yields a perfect tracking, as shown in (a). During online learning, the learned task-space controller continuously improves the task-space tracking performance. As shown in (b), the learned task-space control torques $\boldsymbol{u}_{\text{joint}}$ converge to the perfect analytical torques after a short transient phase.



(a) **task-space tracking performance** (b) **joint-space trajectory**

Fig. 3: Task-space tracking control of a simulated 7-DoF Barrett WAM with online learned model. (a) During online model learning, the task-space controller is able to compute the required torques to follow the task-space trajectory. (b) joint-space trajectory during the online learning. Here, the rest posture is given by $\boldsymbol{q}_{\text{rest}} = [0.0, 0.5, 0.0, 1.9, 0.0, 0.0, 0.0]^T$.

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

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

[6] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1–5, pp. 11–73, 1997.

[7] J. Peters and S. Schaal, "Learning to control in operational space," *International Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.

[8] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *IEEE International Conference on Intelligent Robots and Systems*, 2001.

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

[10] I. Jordan and D. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science*, vol. 16, pp. 307–354, 1992.

[11] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, pp. 1317–1329, 1998.

[12] M. Lopes and B. Damas, "A learning framework for generic sensory-motor maps," in *International Conference on Intelligent Robots and Systems*, San Diego, CA, 2007.

[13] N. Bhushan and R. Shadmehr, "Evidence for a forward dynamics model in human adaptive motor control," *Advances in Neural Information Processing Systems*, 1999.

[14] G. Tevatia and S. Schaal, "Efficient inverse kinematics algorithms for high-dimensional movement systems," *University of Southern California*, 2008.

[15] C. Salaun, V. Padois, and O. Sigaud, "Control of redundant robots using learned models: an operational space control approach," in *Proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems*, 2009.

[16] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1000–1017, 1999.

[17] D. Nguyen-Tuong and J. Peters, "Incremental sparsification for real-time online model learning," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

[18] M. Seeger, "Low rank update for the cholesky decomposition," University of California at Berkeley, Tech. Rep., 2007.

[19] D. Nguyen-Tuong and J. Peters, "Model learning with local gaussian process regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.