Learning Robot Dynamics with Kinematic Bézier Maps

Stefan Ulbrich, Michael Bechtel, Tamim Asfour and Rüdiger Dillmann

***Abstract*— The previously presented Kinematic Bézier Maps (KBM) are a machine learning algorithm that has been tailored to efficiently learn the kinematics of redundant robots. This algorithms relies upon a representation based on projective geometry that uses a special set of polynomial functions borrowed from the field of Computer Aided Geometric Design (CAGD). So far, it has only been possible to learn a model of the forward kinematics function. In this paper, we show how the KBM algorithm can be modified to learn the robot's equation of motion and, hence, its *inverse dynamic model*. Results from experiments with a simulated serial robot manipulator are presented that clearly show the advantages of our approach compared to general function approximation methods.**

I. Introduction

The body schema in humans is a non-static and adaptive cerebral representation of the current state of the body which links several sensor modalities such as *proprioception* (i.e., self perception) and the perceived visual shape of the body together. It is, for instance, held responsible for the ability of using tools, a trait that can be found (among several others) in humans and higher primates. Currently, research begins to focus on the reproduction of this trait on robotic platforms as can be seen by the list of surveys that appeared recently in [5], [9], [11]. From a machine learning point of view, many algorithms seem suitable to learn kinematics models—one aspect of building a robot body schema—among which the most prominent are *Locally Weighted Projection Regression (LWPR)* [12], [4], *Parameterized Self-organizing Maps (PSOM)* [17], [7], *Gaussian Mixture Models (GMM) and Regression (GMR)* [1], [2], and, finally, the *Kinematics Bézier Maps (KBM)* which we presented previously in [16][1]. The latter uses a special geometrical model and functions borrowed from the field of computer aided geometry design (CAGD) to learn the relation between joint encoders and the self-perceived shape of the body. Its design restricts learning to modeling the forward kinematics. The creation of a more complete version of the robot body schema, however, requires the integration of more sensor modalities. In this paper, we therefore present a variation of the KBM learning that allows us to include a dynamic model of the robot by learning the *Inverse Dynamics* (i.e., its equation of motion). The machine learning approach to the Inverse Dynamics is not novel. Various techniques to estimate the multivariate function describing the mapping from motion parameters to applied torques have been proposed over the years. They can be divided into local learning, which quickly produces models which are only valid in the vicinity of previously presented data, and global learning that creates a more complete model of the robot dynamics but is notoriously slower. *Locally Weighted Projection Regression (LWPR)* [12] is a local learning method that can be used to approximate the dynamics equation with multiple locally linear models which are weighted individually to limit their respective region of validity. While being capable of online, real-time learning at low computational costs, this algorithm performs poorly in regions not previously seen in training.

Another, global, learning technique is *Gaussian Process Regression (GPR)* which assumes samples from the dynamics equation to be distributed according to a multivariate Gauss distribution and thus is approximated by a Gaussian process—hence the name. It yields higher accuracy in estimating the Inverse Dynamics than LWPR does but its high computational complexity prohibits online learning. An adapted version of GPR [8] overcomes this problem by regressing the dynamics equation locally, similarly to LWPR [10].

However, to our knowledge, there does not exist any technique that takes the underlying nature of the target function into account and is, consequently, suited to present a numerically exact encoding of the inverse dynamics. The approach presented here accomplishes this under certain conditions—namely if there is no noise included in the perceived sensor data and a large amount of training data can be provided. In this case, the algorithm uses *batch learning* and generalizes perfectly—it even allows to predict the inverse dynamics exactly for parts of the configuration space that have not been previously explored during training *(i.e., extrapolation)*. A fair amount of noise in the data can be compensated by increasing the number of training observations. Thanks to the underlying model, the prediction can be even more reliable than the sensor readings. The biggest drawback of batch learning, however, is the quantity of observations required. It renders the application for robots with many degrees of freedom (DoF) impossible. For those cases, the algorithm relies on an *incremental learning* strategy that quickly adapts to real data using a model previously created in simulation. This learning affects the model only locally and hence offers a performance comparable to those of pure local learners. We evaluate the presented theory in simulation using the *Robotics Toolbox for Matlab* [3].

The document is structured as follows: The next section

S. Ulbrich, M. Bechtel, T. Asfour and R. Dillmann are at the Humanoids and Intelligence Systems Lab, Institute for Anthropomatics at the Karlsruhe Institute for Technology, Germany stefan.ulbrich,asfour,dillmann@kit.edu and michael.bechtel2@student.kit.edu

[1] **Important:** At the time of submission, this paper has been accepted (in January), modified by the editors, and the is due to publication within the next few weeks. In the meanwhile, [14] documents an earlier development stage.

will provide a short introduction to the physical background and underlying theories—including an overview over machine learning for kinematics and a detailed introduction to the KBM algorithm. Afterwards, we show how this algorithm can be modified to also learn the inverse dynamics by first examining the individual components of the equation of motion and later the complete formula. In section three, we present the results from experiments performed in simulation using a dynamic model of an industrial robot with a reduced number of DoF. Finally, the document concludes with a discussion about the presented theory and closes with a perspective on further research.

II. Fundamentals

In this section, we will present the basic definitions/fundamentals necessary to understand the dynamics of robots and the machine learning algorithm presented. We will focus completely on serial robot manipulators with revolute joints only. The following conventions apply to the formulas in this paper:

- t, θ_1 Small letters refer to scalars or—with indices—to components of a vector or matrix with similar name,
- $\boldsymbol{b}$, $\boldsymbol{\theta}$ bold letters denote vectors,
- α, θ_1 small greek letters typically are angles,
- Γ, M and capital letters represent matrices
- Q, L, T with the exception of forces, depending on the context.

A. Robot Dynamics

In order to warrant precise control of robotic systems, it is crucial to have exact knowledge of the robot's dynamical behavior. In this section, we provide a brief introduction to this subject. For more detailed information, please refer to a more comprehensive introduction as such provided in [3] for instance.

The inverse dynamics equation describes the relationship between a manipulator's motion defined by *generalized coordinates* (that is, angular position $\boldsymbol{\theta}$, speed $\dot{\boldsymbol{\theta}}$ and acceleration $\ddot{\boldsymbol{\theta}}$ of the robot's joints) and the generalized forces Q required to achieve this motion(—hence the label 'inverse'). Solving the Lagrangian equation

$$Q_i = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i}$$

where

$$L = T - V$$

with T being the kinetic and V being the potential energy of the robotic system, both represented as functions of the n generalized coordinates $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}$ of the system, yields

$$Q = M(\boldsymbol{\theta}) \cdot \ddot{\boldsymbol{\theta}} + C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \cdot \dot{\boldsymbol{\theta}} + \boldsymbol{h}(\boldsymbol{\theta}) + \boldsymbol{\epsilon}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}) \qquad (1)$$

where

- $M(\boldsymbol{\theta})$ denotes the symmetrical and positive definite $n \times n$ matrix of the robot's inertia,
- $C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the $n \times n$ centripetal and Coriolis force coupling matrix, and
- $\boldsymbol{h}(\boldsymbol{\theta})$ refers to the $n \times 1$ vector that describes the gravitational influence on the system.

Additionally, effects on the dynamic behavior such as backlash or friction, which are non-linear and not well understood, are symbolized by $\boldsymbol{\epsilon}(\theta, \dot{\theta}, \ddot{\theta})$. Finally, the generalized forces are expressed by the $n \times 1$ vector Q.

B. Learning Kinematics

Learning the inverse dynamics of a serial robot manipulator with rotational joints is closely related to learning its forward kinematic model. The forward kinematics can be understood as a mapping from the angular configuration space of the robot into the cartesian workspace. The more complex a robot becomes, the more difficult it gets to derive this function accurately from a CAD model and the more expensive (with respect to time consumption and hardware) is the calibration of the robot. A possible solution to this problem, which recently received increasing attention, is to obtain an approximation to this function by machine learning. Such algorithms extract the relation between joint angles and cartesian positions from example configurations mostly obtained by self observation. See [5], [9], [11] for an overview. However, no algorithm can overcome the limitation that the number of observations required to learn a model grows exponentially with the length of a kinematic chain (i.e., the number of DoF). The problem can be avoided, however, by decomposing the kinematic chain either by observing more elements of the robot or more complex exploration strategies [15].

C. The Kinematic Bézier Maps

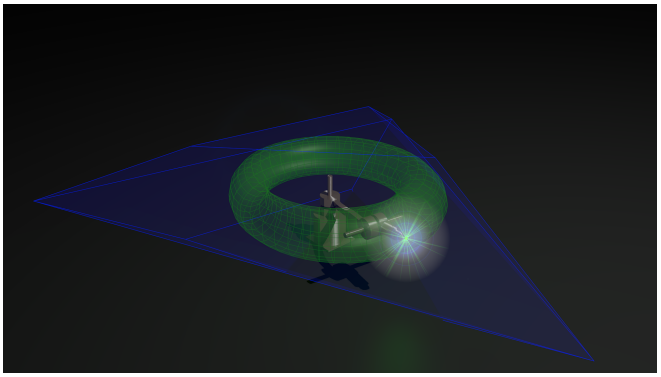


Fig. 1. The geometric representation used with the KBM as a model of a mechanism with two orthogonal DoF. The manifold of reachable positions is enclosed by a net of vertices that compose the learned model.

The *Kinematic Bézier Maps* are a novel machine learning approach presented by Ulbrich et al. [16]. It differs from most common machine learners, which are successfully used for learning kinematics, in terms of being specialized to learning combinations of trigonometric functions. This way, a global model can be obtained much faster and more accurately than it is possible by more general function approximation

methods such as LWPR [12]. The algorithm offers two learning modes. *Batch learning* allows us to learn a global model of the FK with a relatively small number of training data. Without noise, a model that exactly encodes the FK can be learned from only 3^n random training samples. If there is noise in the input data, it can efficiently be compensated by increasing the amount of training data. The second mode, *Incremental learning*, is capable of quickly improving an existing inaccurate model locally. Hence, the KBM are combining the speed of local learners and the good generalization and extrapolation of global learners. The techniques the KBM relies on originate from the field of computer aided geometrical design. Following the observation that the end-effector of a robot (with revolute joints) moves along a circular trajectory when fixing all but one joints, the manifold representing its workspace is a product space of rotations. For a robot with two orthogonal axes for instance, this manifold has the shape of a torus (see Fig. 1). In [16], it is shown that such manifolds can be parameterized by a *tensor product* of *rational quadratic Bézier functions*. A kinematic model $\boldsymbol{f}$ with a matrix of learned parameters G at a given joint configuration $\boldsymbol{\theta}$ can be expressed as

$$\boldsymbol{f}(\boldsymbol{\theta}; G) = \frac{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot \boldsymbol{b}_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau_{\bar{\gamma}}(\boldsymbol{\theta}))}{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau_{\bar{\gamma}}(\boldsymbol{\theta}))}, \qquad \text{where} \qquad (2)$$

- $\boldsymbol{i}$ is an n-dimensional index vector referring to three elements per DoF,
- $\boldsymbol{b_i}$ and $\gamma_{\boldsymbol{i}}$ are the 3^n control points that form the columns of the parameter matrix $\boldsymbol{G}$ and their predestined weights, respectively,
- $B^2_{\boldsymbol{i}}$ refers to a combination of Bernstein polynomials,
- $\tau_{\bar{\gamma}}(\boldsymbol{\theta})$ is a nonlinear parameter transformation that maps a joint configuration $\boldsymbol{\theta}$ to the parameters required by the Bernstein polynomials.

In order to model the forward kinematics, a priori model knowledge has been encoded within this formula, namely inside the parameter transformation $\tau_{\bar{\gamma}}$ and in the choice of the weights $\gamma_{\boldsymbol{i}}$

$$\gamma_{\boldsymbol{i}} = \bar{\gamma}^{\mathrm{count}_1(\boldsymbol{i})}, \quad \bar{\gamma} < 1$$

where the function $\mathrm{count}_n(\boldsymbol{i})$ returns the number of occurrences of the number n in an index vector $\boldsymbol{i}$. In brief, both the weights and the transformation depend on a parameter $\bar{\gamma}$ that is related to size of the part of the configuration space in which the KBM are completely numerically stable. For more details about this parameter please refer to [16].

With this embedded knowledge, the equation above becomes a linear weighted sum in which the 3^n control points $\boldsymbol{b_i}$ are the only unknown variables. Consequently, learning has been reduced to solving the linear equation derived from (2) for a given set of m joint configurations $\Theta = (\boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_m)$ and associated cartesian coordinates $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$:

$$G \cdot B(\Theta) = X \qquad (3)$$

where the dimension of the *coefficient matrix* $B(\Theta)$ that contains the necessary combinations of the $B_{\boldsymbol{i}}(\boldsymbol{\theta}_j)$ is $3^n \times m$. The minimum number of samples required to find a solution consequently matches the quantity of control points. For higher numbers of observations, a least squares solution can be found which is useful for canceling potential noise in the input data. Another property crucial for learning dynamic models is the ability to learn the partial derivatives of a FK in a similar way. Usually, the parameters for models of the derivatives are derived from the FK model by geometrical means though.

The linear characteristic of (2) also facilitates the application of fast-adapting incremental and online-applicable algorithms such as *Normalized Linear Mean Squares* (also known as the δ-Rule).

In the remainder of this paper, we will present the modifications of the KBM algorithms necessary in order to be able to also learn the inverse dynamic model of serial robot manipulators with revolute joints.

III. Learning Robot Dynamics

A. Decomposition of the Equation of Motions

According to Hollerbach [6], the Lagrangian formulation in (1) can be expressed as

$$\begin{aligned} Q_i = \sum_{j=1}^{n} \Bigg[& \sum_{k=1}^{j} \left(\mathrm{tr} \left(\frac{\partial W_j}{\partial \theta_i} J_j \frac{\partial W_j}{\partial \theta_k}^T \right) \ddot{\theta} \right) \\ & + \sum_{k=1}^{j} \sum_{l=1}^{j} \left(\mathrm{tr} \left(\frac{\partial W_j}{\partial \theta_i} J_j \frac{\partial^2 W_j}{\partial \theta_k \partial \theta_l}^T \right) \dot{\theta}_k \dot{\theta}_l \right) \\ & - m_j \boldsymbol{g}^T \frac{\partial W_j}{\partial \theta_i} \boldsymbol{c}_j \Bigg] \end{aligned} \qquad (4)$$

where

- $W_j \in SO_3$ represent the FK and are homogeneous matrices describing the coordinate system of the j-th joint,
- $J_j \in \Re^{4\times4}$ denotes the inertia tensor of the j-th link with respect to W_j,
- $m_j \cdot \boldsymbol{g}^T$ is the weight force of the link j, and
- $\boldsymbol{c}_j$ are the (homogeneous) coordinates of the center of mass of link j, again with respect to W_j.

Note that W_j depends on the joint positions $(\theta_1, \ldots, \theta_j)$—the parameters are omitted for better readability. This equation directly shows the close coupling of the forward kinematics and the inverse dynamics. In the following, we will investigate how the components of the generalized forces can first be learned individually and then simultaneously.

1) Gravity: The gravity component is by far the component of (4) to be learned the most easily. It is also the most dominant summand in the equation of motion. One can observe that the gravity component accounts to

$$-\sum_{j=1}^{n} \left(m_j \boldsymbol{g}^T \frac{\partial W_j}{\partial \theta_i} \boldsymbol{c}_j \right) \qquad (5)$$

in the sum. The component for the extended force Q_i is a sum of partial derivatives (with respect to joint j) of the FK of all of the robots center of masses. As such it can be learned without modifications to the KBM algorithm with 3^n observations. Similar to (3), we name the new coefficient matrix for learning $B_{\boldsymbol{h}}(\Theta)$.

2) The Intertia Matrix: From (4) follows that, when neglecting the influence of the Coriolis effects and gravitation, the generalized forces Q_i are:

$$Q_i = \sum_{j=1}^{n} \left(\sum_{k=1}^{j} \underbrace{\operatorname{tr}\left(\frac{\partial W_j}{\partial \theta_i} J_j \frac{\partial W_j^T}{\partial \theta_k} \right)}_{=:\mu_{i,j,k}} \ddot{\theta}_k \right)$$

After rearrangement of the sums, we can express the generalized forces Q as a weighted sum of the joint accelerations:

$$Q = \begin{pmatrix} \sum_{i=1}^{n} \mu_{1,i,1} + \sum_{i=2}^{n} \mu_{1,i,2} + \sum_{i=3}^{n} \mu_{1,i,3} + \dots \\ \sum_{i=2}^{n} \mu_{2,i,1} + \sum_{i=2}^{n} \mu_{2,i,2} + \sum_{i=3}^{n} \mu_{2,i,3} + \dots \\ \sum_{i=3}^{n} \mu_{3,i,1} + \sum_{i=3}^{n} \mu_{3,i,2} + \sum_{i=3}^{n} \mu_{3,i,3} + \dots \\ \vdots \end{pmatrix}$$
$$= \left(\sum_{j=1}^{n} \ddot{\theta}_j \cdot \sum_{k=max(j,i)}^{n} \operatorname{tr}\left(\frac{\partial W_k}{\partial \theta_i} J_k \frac{\partial W_k^T}{\partial \theta_j} \right) \right)_i \quad (6)$$

Because of $\mu_{i,j,k} = \mu_{k,j,i}$, one obtains the entries of the symmetric inertia matrix M (for a given robot configuration):

$$M = \left(\sum_{k=max(j,i)}^{n} \operatorname{tr}\left(\frac{\partial W_k}{\partial \theta_i} J_k \frac{\partial W_k^T}{\partial \theta_j} \right) \right)_{i,j} \quad (7)$$

Unlike gravity, the components of the inertia matrix cannot be learned without modification of the KBM algorithm. This results from the occurence of the product of partial derivatives W_k of the FK in the sums. As a consequence, the matrix elements are not only combinations of trigonometric functions but also of their squares. These combinations cannot be represented by (rational) quadratic functions. To solve this problem, the polynomial degree of the Bézier functions has to be raised to four. In (2), this means to redefine the index variable $\boldsymbol{i}$, as well as to adjust the weights $\gamma_{\boldsymbol{i}}$ of the control points of the learned manifold. Remember that—in the one-dimensional case—the weights were $(1, \gamma, 1)^T$. Now assume that we have a rational Bézier curve $\boldsymbol{b}(\tau(\theta))$ with

$$\boldsymbol{b}(t(\theta)) = \sin(\theta)$$

In order to obtain the new weights, we observe the squared function

$$b_2(\underbrace{\tau(\theta)}_{=:t})^2 = sin^2(\theta)$$
$$= \left(1 \cdot t^2 + \bar{\gamma} \cdot 2 \cdot t \cdot (1-t) + 1 \cdot (1-t)^2\right)^2$$
$$=: \sum_{i=1}^{5} \gamma_i \cdot B_i^4(t), \quad \boldsymbol{\gamma} = (1, \bar{\gamma}, 1/3(1 + 2\bar{\gamma}^2), \bar{\gamma}, 1)$$

The $B_i^4(t)$ denote the Bernstein polynomials of degree four. Note that $b(t)$ not only can represent a combination of non squared trigonometric functions but also mixed combinations of squared and non squared trigonometric functions. In the multivariate case, the weights $\gamma_{\boldsymbol{i}}$ can be obtained by the following formula:

$$\gamma_i = \bar{\gamma}^{(\mathrm{count}_1(\boldsymbol{i}) + \mathrm{count}_3(\boldsymbol{i}))} \cdot \left(1/3(1 + 2\bar{\gamma}^2)\right)^{\mathrm{count}_3(\boldsymbol{i})}$$

In analogy to (3), we name the coefficient matrix $B^4(\Theta)$ for a given set of robot configurations Θ [see (9)].

These modifications are sufficient to learn the elements of the matrix M. However, this ability comes at a high cost as the minimal number of observations during training is increased drastically. Fortunately, it can be logically concluded that the generalized force in the first joint is not influenced by its joint position—of course only if all components but the inertia are neglected. The same accounts for all other joints so the total number of of training observations required is raised only to 5^{n-1}.

3) Coriolis and centripetal coupling effects: Now we consider the influence of the Coriolis and centripetal coupling effects in (4).

$$Q_i = \sum_{j=i}^{n} \sum_{k=1}^{j} \sum_{l=1}^{j} \underbrace{\left(tr\left(\frac{\partial W_j}{\partial \theta_i} J_j \frac{\partial^2 W_j^T}{\partial \theta_k \partial \theta_l} \right) \dot{\theta}_k \dot{\theta}_l \right)}_{\gamma_{i,j,k,l}}$$

In analogy to the previous section, we can write this formula as

$$Q = \left(\sum_{j=1}^{n} \sum_{k=1}^{n} \dot{\theta}_j \dot{\theta}_k \cdot \sum_{l=max(i,j,k)}^{n} tr\left(\frac{\partial W_l}{\partial \theta_i} J_l \frac{\partial^2 W_l^T}{\partial \theta_j \partial_k} \right) \right)_i$$
$$=: \Gamma \cdot \left[\dot{\theta}_1\dot{\theta}_1, \dot{\theta}_1\dot{\theta}_2, \dots, \dot{\theta}_2\dot{\theta}_1, \dot{\theta}_2\dot{\theta}_2, \dots \right]^T, \quad (8)$$

a matrix multiplication with vector containing all n^2 possible products of the joint velocities $\dot{\theta}_i$. As previously with the inertia matrix M, the elements of Γ again contain products of partial derivatives of the FK and therefore require the same degree elevation of the rational Bézier tensor product. Also the Coriolis matrix Γ is independent of the first joint position θ_1 as well, so that the number of observations to train a single element of the matrix is again 5^{n-1}. The number of columns in Γ can also be decreased, as redundancy can be eliminated by combining all elements that have identical joint velocities (i.e., $\dot{\theta}_i\dot{\theta}_j$ and $\dot{\theta}_j\dot{\theta}_i$ if $i \neq j$). That way, the number of elements in this vector is reduced to $n \cdot \frac{(n+1)}{2}$. If the matrix elements are to be learned, the Bernstein polynomials are included in the coefficient matrix $B_\Gamma(\Theta)$ given a set of robot configurations Θ.

4) Computed Torque Control: In order to allow for stable, precise trajectory following abilities and to cancel nonlinearities $\boldsymbol{\epsilon}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$ in (1) of real robotic systems such as Coulomb friction or backlash, model-based feed forward controllers such as the computed torque control approach

have been established [12], [8]. Typically, the motor command signal u is defined by

$$u = u_{CT} + u_{FB}$$

which linearly decouples the control signal in u_{CT}, which is the torque determined directly from the learned inverse dynamics equation, and a feedback term u_{FB}, which is used to stabilize the following of a given trajectory $\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d$.
The feedback term u_{FB} is presented by a PD control law[2]

$$u_{FB} = K_p e + K_d \dot{e}$$

where

$$e = q - q_d$$

is the tracking error and K_p and K_d are proportional and derivative gain matrices, respectively.
If an exact model for the inverse Dynamics equation of the robot is known, the feedback loop is able to overcome nonlinearities of the system, yielding an appropriate motor signals for compliant robot control. Note that the application of computed torque control might be necessary in real-life experiments. In our experiments to verify learning, however, we use a dynamic simulator and cancelled all non-linearities subsumed in the term $\boldsymbol{\epsilon}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$ in order to show the ability to exactly learn all other terms in the absence of noise.

B. Learning the closed-form

1) Composition: If—as shown in the previous section—the elements of the inertia matrix can be expressed by a KBM each, a simplified version of the equation of motion consisting only of the inertial moments can be written as follows:

$$\begin{aligned} Q &= M \cdot \ddot{\boldsymbol{\theta}} \\ &= \begin{pmatrix} \boldsymbol{m}_{11} \cdot B^4(\boldsymbol{\theta}) & \cdots & \boldsymbol{m}_{1n} \cdot B^4(\boldsymbol{\theta}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{m}_{n1} \cdot B^4(\boldsymbol{\theta}) & \cdots & \boldsymbol{m}_{nn} \cdot B^4(\boldsymbol{\theta}) \end{pmatrix} \cdot \begin{pmatrix} \ddot{\theta}_1 \\ \vdots \\ \ddot{\theta}_n \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{m}_{11} \cdots \boldsymbol{m}_{1n} \\ \vdots \\ \boldsymbol{m}_{n1} \cdots \boldsymbol{m}_{nn} \end{pmatrix} \cdot \begin{pmatrix} B^4(\boldsymbol{\theta})^T \cdot \ddot{\theta}_1 \\ \vdots \\ B^4(\boldsymbol{\theta})^T \cdot \ddot{\theta}_n \end{pmatrix} \\ &=: G \cdot B_M(\boldsymbol{\theta}, \ddot{\boldsymbol{\theta}}), \end{aligned} \tag{9}$$

where,

- $\boldsymbol{m}_{ij}$ are the control points ($1 \times 5^{n-1}$) that encode the matrix elements' dependence on $\boldsymbol{\theta}$,
- $B^4(\boldsymbol{\theta})$ is a vector that contains the 5^{n-1} combinations of Bernstein polynomials of degree four,
- G is a $n \times n \cdot 5^{n-1}$ matrix that yields all parameters of the simplified model, and
- $B_M(\boldsymbol{\theta}, \ddot{\boldsymbol{\theta}})$ is the combination of the B^4 and $\dot{\boldsymbol{\theta}}$ (with the dimensions $n \cdot 5^{n-1} \times 1$).

This means that this simplified equation of motion can be easily learned if the basis polynomials in the KBM are changed to $B_M(\boldsymbol{\theta}, \ddot{\boldsymbol{\theta}})$. Note that this learning is expensive as the minimum required number of training instances is increased n-fold to $n \cdot 5^{n-1}$. In the context of the system of linear equations in (3), we name the coefficient matrix $B_M(\Theta, \ddot{\Theta})$ for a set of robot configurations.
Now, we also include the gravity component $\boldsymbol{h}$ into the equation. Since it is impossible to measure its influence separately from the inertia component (in fact, none of the components can be observed individually), they have to be learned simultaneously. This increases the number of unknown variables in this equation. Mathematically, this can be achieved by stacking the coefficient matrices $B_M(\Theta, \ddot{\Theta})$ and $B_h(\Theta)$ on top of each other.

$$B_{M,\boldsymbol{h}}(\Theta, \ddot{\Theta}) := (B_M B_{\boldsymbol{h}})^T$$

The resulting matrix has $n \cdot 5^{n-1} + 3^n$ rows. Finally, the components of the Coriolis matrix Γ have to be included as well in the same manner. This time, the coefficient matrix B_Γ is combined with joint velocities resulting in a new matrix $B_C(\Theta, \dot{\Theta})$ which is then included in the final coefficient matrix

$$B_{M,C,\boldsymbol{h}}(\Theta, \dot{\Theta}, \ddot{\Theta}) := (B_M B_C B_{\boldsymbol{h}})^T .$$

The resulting number of rows is raised to its final number of $n\frac{(n+3)}{2}5^{(n-1)} + 3^n$. Learning the complete equation of motion means then solving the following system of linear equation

$$G_{M,C,\boldsymbol{h}} \cdot B_{M,C,\boldsymbol{h}} = X, \qquad X = (Q_1, \ldots, Q_m).$$

After learning, the same equation is used to predict joint torques values for a given set of generalized coordinates, that is, the matrix $B_{M,C,\boldsymbol{h}}$ has to be computed first. Compared to the 3^n observations required to learn the FK or the gravity component, the number of unknowns in the model increased drastically. One must not forget, however, that this is an exact model when neglecting friction. The problem itself (i.e., deriving the parameters of physical systems), however, has recently been proven to be NP-hard [13]. Similarly to learning the FK, the system of linear equations can be learned directly when doing enough observations *(batch learning)* or continuously using the *Normalized Least Squares (NLMS)* algorithm *(incremental learning)*. [Absatz zu viel Information zu dicht]

IV. Experiments

This section presents the experiments that demonstrate the new approach. We did all experiments in simulation using the *Robotics Toolbox for MATLAB* [3]. In this experiments, we use a dynamic model of a *PUMA-560 (Programmable Universal Machine for Assembly)* robot with a reduced number of DoF (i.e., ignoring the most distal joints). We use only four DoF in order to be able to work with a reasonable number of training samples.

A. Batch Learning

1) Noise Compensation: In order to demonstrate the batch-learning algorithm's capability of dealing with noisy training data, two experiments have been carried out. Firstly,

[2]Proportional and derivative controller

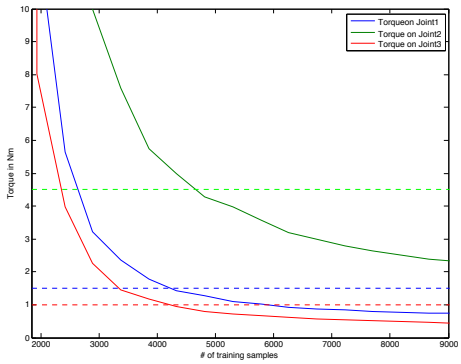


Fig. 2. The learning curves of the first three joint torques in dependence of the number of training samples. The dashed lines indicate mean noise in the torque sensors.

it has been examined how many training samples are needed to compensate a fixed noise on the torque samples used for training. Therefore, an artificial, normally distributed noise with 5% standard deviation with respect to the value range of the individual torques has been added to the torque values computed for a set of randomly created generated generalized coordinates. The prediction error of the algorithm is evaluated on a separated, noise-free set of test samples. Multiple models are learned subsequently with increasing numbers of training samples until the noise is compensated. The results of this experiment are displayed in Fig. 2.

This experiment shows that for a sufficiently high number of training samples, the algorithm is able to cancel out errors arising from noisy training data given enough redundant training samples. Additionally, it manages to yield higher accurate predictions than the (noisy) measurement of the torques could (as it can be seen from the predicted torque values dropping below the mean sensor noise), that is, because of the model-based approach it becomes possible to create a more accurate approximation than with a model-less method.

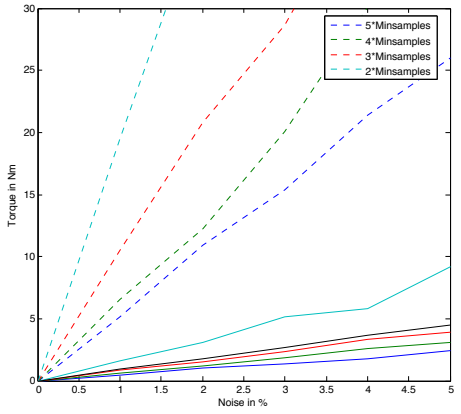


Fig. 3. Mean errors for fixed numbers of training samples in dependence of the degree of noise in the sensor data. Regular curves denote the error for interpolation, the dashed line for extrapolation. The black line represents the mean noise in the combined [(Euclidean distance)] torques.

2) Interpolation and Extrapolation: Secondly, the algorithm's interpolation and extrapolation (i.e., predicting torque values far away from previously observations) abilities in presence of noise have been evaluated. For interpolation, both training and test samples have been created from angles from the same restricted range of the joint space. Test samples, again, remain noiseless. The extrapolation uses test data out of a portion of the configuration space that is 16 times bigger than during training. Please refer to [16] for a more formal definition of these spaces.

As in the previous experiment, models for increasing degree of redundancy in the training samples have been learned subsequently. This time, however, the experiment has been carried out for varying amounts of noise (ranging from 1 to 5 percent). The mean errors are displayed in dependence of the degree of noise in Fig.3.
The results clearly show, that number of training samples required to achieve a certain precision is proportional to the degree of noise. This means, that no matter how much the noise becomes (for still reasonable values) three times the minimal number of training samples is required to surpass the sensor input. Another fact observable in the image is that even for relatively small training data the algorithm expose a reasonably extrapolation not possible to achieve with local learning approaches.

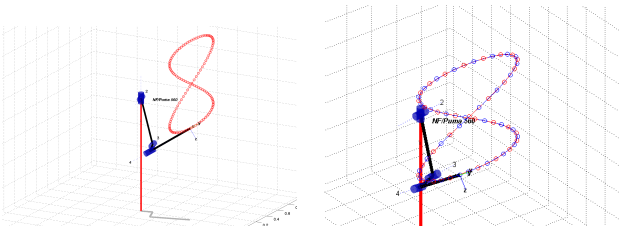
B. Incremental Learning

Now we will investigate how a previously learned model can be improved using the incremental learning algorithm. These experiments simulate real-life scenarios where the real parameter of the robot can only be estimated with the CAD but cannot be directly measured. An equally important and quite similar use-case is model-adaptation when the robot grabs and holds a tool (tool use). In these experiments, the robot moves along a periodical planar trajectory (see Fig. 4(a)) defined by the following formula:

$$\boldsymbol{x}(\alpha) = (0.4 \cdot \sin(\alpha), -0.5, 0.4 \cdot \cos(\alpha/2) + 0.2)^T \quad (10)$$

where α varies equidistantly from -2π to 2π within $10s$. The generalized coordinates and forces are calculated by the dynamics simulation and recorded while the end-effector moves along the trajectory.
The main idea behind this experiment is to simulate that the robot is using a tool. Therefore, we added an object to the end-effector with a mass of $2.5kg$ and its center of mass in distance of $20cm$ along the x-axis of the local coordinate frame. That way, we can simulate a heavy tool—comparable to a hammer for instance—that influences the dynamics of the system heavily.

The robot's inverse dynamics (i.e., without the tool) has been previously learned with the batch learning algorithm. No noise has been added to the training data. Then, the robot moves along the trajectories recording 42 value pairs of generalized forces and coordinates. These are alternately divided into training and test data sets [see Fig. 4(b)]. Afterwards, the model is incrementally adapted to the new dynamics repeatedly learning the complete training data set three times.



(a) The trajectory and the PUMA robot shown in cartesian space.

(b) Every second sample is used for training, the remaining form the test data set.

Fig. 4. The 'figure-8' movement defined in (10) used in the experiments in this section.

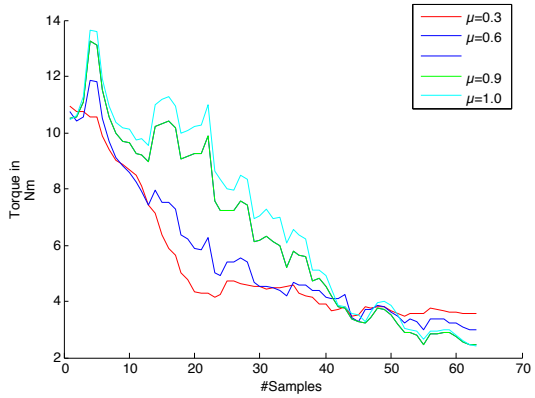


Fig. 5. The result of the incremental learning experiments. The value μ denotes the adaptation rate.

After each new sample, the model is evaluated against the whole test data set. The experiment was repeated multiple times with different adaptation rates. The results are shown in Fig. 5. One can see that the error quickly decreases during learning. With lower adaptation rates, learning is initially faster while it is slower in the long run. This experiment shows clearly that the incremental learning can be very fast and thus is comparable to local learners.

V. Discussion and Conclusion

In this paper, we presented a machine learning technique that is specialized to the underlying functions the inverse dynamics problem consists of. It is an advancement of the previously presented KBM algorithm that successfully learns models of the FK. Under certain conditions, that is, when the training data is not noisy and can be provided with the required quantity, the modified algorithm produces a model that is an exact implicit encoding of the dynamic model. Its greatest drawback is the large amount of training data at least required when performing batch-learning. The number of observations during training is exponential in the number of the robot's DoF. This lies, however, in the nature of the problem as the determination of model parameters is a NP-hard problem. To tackle this problem, an incremental variant of the algorithm is proposed, which quickly learns from real data refining a model previously built in simulation.

Yet there are still many open questions we would like to address in future research. At first, we will investigate if the need for observations can be optimized using well designed exploration strategies. The theory has to be adapted to parallel robots systems. Right now, the algorithm neglects the influence of friction which could be absorbed by learning as well. Finally, we plan on applying the learning on a real humanoid robot.

VI. Acknowledgments

This work was partially conducted in the project "Robots Exploring their Bodies Autonomously (REBA)" within the priority program "Autonomous Learning" founded by the German Research Foundation (DFG).

The authors would like to thank Peter Corke for the *Robotic Toolbox for MATLAB* and for his support that helped us to successfully implement the experiments.

References

[1] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 2, pp. 286–298, 2007.

[2] C. Constantinopoulos and A. Likas, "Unsupervised learning of gaussian mixtures based on variational component splitting," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 745–755, 2007.

[3] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB*, ser. Springer Tracts in Advanced Robotics. Springer, 2011, vol. 73.

[4] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. IEEE/RSJ Intl. Conf. Int. Robots Systems (IROS)*. Piscataway, New Jersey, 2001.

[5] M. Hoffmann, H. G. Marques, A. H. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer, "Body schema in robotics: A review," *IEEE T. Autonomous Mental Development*, vol. 2, no. 4, pp. 304–324, 2010.

[6] J. M. Hollerbach, "A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Trans. Syst., Man, Cybern. B*, vol. 10, no. 11, pp. 730–736, nov. 1980.

[7] S. Klanke and H. J. Ritter, "PSOM+ : Parametrized self-organizing maps for noisy and incomplete data," in *Proc. of the 5th Workshop on Self-Organizing Maps (WSOM-05)*, Paris, France, Sept. 2005.

[8] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *Proc. IEEE/RSJ Intl. Conf. Int. Robots Systems (IROS)*, sept. 2008, pp. 380–385.

[9] ——, "Model learning for robot control: a survey," *Cognitive Processing*, pp. 1–22, 2011.

[10] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf, "Learning inverse dynamics: a comparison," in *ESANN*, 2008, pp. 13–18.

[11] O. Sigaud, C. Salaun, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: a survey," *Robotics and Autonomous Systems*, vol. 59, pp. 1115–1129, 2011.

[12] J. Sun de la Cruz, D. Kulic, and W. Owen, "Learning inverse dynamics for redundant manipulator control," in *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, june 2010, pp. 1–6.

[13] M. W. T. Cubitt, J. Eisert, "Extracting dynamical equations from experimental data is NP hard," *Physical Review Letters*, Accepted in Ferbruary 2012.

[14] S. Ulbrich, V. Ruiz de Angulo, T. Asfour, C. Torras, and R. Dillmann, "Rapid learning of humanoid body schemas with kinematic bézier maps," in *Proc. IEEE/RAS Intl. Conf. Humanoid Robots (Humanoids)*, Paris, France, Dec. 2009, pp. 431–438.

[15] ——, "General robot kinematics decomposition without intermediate markers," *IEEE Trans. Neural Netw., Learning Syst.*, vol. 23, no. 4, pp. 620–630, april 2012.

[16] ——, "Kinematic Bézier Maps," *IEEE Trans. Syst., Man, Cybern. B*, 2. Quartal 2012.

[17] J. Walter and H. Ritter, "Rapid learning with parametrized self-organizing maps," *Neurocomputing*, vol. 12, no. 2-3, pp. 131–153, 1996.