End-to-End Learning of Hybrid Inverse Dynamics Models for Precise and Compliant Impedance Control

Moritz Reuss[*‡], Niels van Duijkeren[*], Robert Krug[*], Philipp Becker[§], Vaisakh Shaj[§†] and Gerhard Neumann[§]

[*]Bosch Corporate Research, Renningen, Germany
[†]LCAS, University Of Lincoln, UK
[‡]Intuitive Robots Lab, Karlsruhe Institute of Technology, Germany
[§]Autonomous Learning Robots, Karlsruhe Institute of Technology, Germany

Abstract—It is well-known that inverse dynamics models can improve tracking performance in robot control. These models need to precisely capture the robot dynamics, which consist of well-understood components, e.g., rigid body dynamics, and effects that remain challenging to capture, e.g., stick-slip friction and mechanical flexibilities. Such effects exhibit hysteresis and partial observability, rendering them, particularly challenging to model. Hence, hybrid models, which combine a physical prior with data-driven approaches are especially well-suited in this setting. We present a novel hybrid model formulation that enables us to identify fully physically consistent inertial parameters of a rigid body dynamics model which is paired with a recurrent neural network architecture, allowing us to capture unmodeled partially observable effects using the network memory. We compare our approach against state-of-the-art inverse dynamics models on a 7 degree of freedom manipulator. Using data sets obtained through an optimal experiment design approach, we study the accuracy of offline torque prediction and generalization capabilities of joint learning methods. In control experiments on the real system, we evaluate the model as a feed-forward term for impedance control and show the feedback gains can be drastically reduced to achieve a given tracking accuracy.

I. INTRODUCTION

Automating industrial tasks such as shaft insertion or cable plugging in semi-structured environments remains challenging for industrial robots. Simultaneous compliant and precise motion tracking is required to prevent the damage of fragile parts in assembly steps while maintaining precise movements. However, achieving compliant behavior and precise motion are typically conflicting objectives in feedback control. Feedback controllers can provide precise movements with high feedback gains but this renders the manipulator stiff and therefore non-compliant. By using a precise inverse model of the robot dynamics the reliance on the feedback controller for tracking accuracy can be reduced. Using such a feed-forward compensator enables simultaneous compliant and precise tracking. Identification of a precise inverse dynamics model of an industrial robot is a key research area in precise and compliant control. A classical approach to derive the equations of motion of the rigid body dynamics is the Recursive Newton-Euler Algorithm. These models have the advantage of being valid in the whole state space while having low computational

complexity and are used widely in robotics. However, they are unable to capture non-rigid and temporal effects. Black-box models such as neural networks offer efficient data-driven optimization with a low bias. However, such purely data-driven models are typically data inefficient, lack any stability guarantees for extrapolation, and only achieve good performance in the vicinity of the training data. These drawbacks disallow their usage for safety-critical applications in semi-structured environments [23].

In the last few years structured models, grey-box models, and hybrid models have gained popularity in the research community [4]. They combine the physical prior of rigid-body models with efficient data-driven optimization to complement their individual strengths. Combining the generalization of rigid-body dynamics models with additional data-driven terms to further optimize task-specific accuracy makes residual hybrid models ideal for compliant and precise motion tracking.

Previous residual hybrid model implementations rely on stateless feed-forward neural networks or Gaussian Processes to learn the error of the rigid body dynamics model. Stateless models are limited in their ability to capture partially observable dynamic effects, such as hysteresis and body elasticities. Utilizing recurrent neural networks such as Long Short-Term Memory networks (LSTM) [8] has the potential to capture these effects and improve the model accuracy.

Hybrid robot models that combine rigid-body dynamics, parameterized in the inertial properties of the kinematic chain, with a residual DNN are most easily obtained in a sequential identification process. Using the established toolchains for each step, the inertial parameters are matched with the collected data first, then the residual DNN is trained on the mismatch. Yet, combining the learning of the inertia parameters and the residual DNN in one step can potentially improve the obtained hybrid model. As the residual DNN captures the non-rigid-body dynamics effects increasingly well, the inertia parameters will fit on a less biased residual error [4].

Recently, a differentiable version of the Recursive-Newton-Euler algorithm (DiffNEA) was introduced in [26, 14]. DiffNEA is optimized using gradient descent methods to find valid inertial parameters, which can be used for end-to-

end learning of inverse dynamics. First results in end-to-end learning of forward dynamics show the potential of applying it to inverse dynamics learning [14].

In this work, we extend previous ideas of end-to-end learning to inverse dynamics learning of hybrid models and target the challenges of the system identification process. We propose a new residual hybrid model architecture consisting of a rigid-body model with a simplified Coulomb friction model and an LSTM as its residual model. We evaluate the performance of the joint learned residual hybrid models to accurately predict the inverse dynamics of a Franka Emika Panda Robot from offline data. Further, we introduce an alternative formulation of DiffNEA as an unconstrained nonlinear regression problem to identify fully physically consistent link frame inertial parameters, which we name "DiffBary". Both methods will be used for end-to-end learning of hybrid inverse dynamics models. We compare our model against recent state-of-the-art structured and unstructured model architectures in terms of their offline torque prediction accuracy and generalization capabilities and show superior performance by our hybrid architecture. Motion tracking experiments are conducted with a real robot arm to evaluate different models inside a joint state impedance controller. The effectiveness of the joint hybrid model training is validated through observed performance improvements of an impedance controller for a 7-DoF robot arm setup and offline evaluation of experiment data. Our results show that using our residual hybrid model enables simultaneous compliant and precise motion tracking.

II. BACKGROUND AND RELATED WORK

The goal of inverse dynamics modeling is to find the function f_θ,

$$\boldsymbol{\tau}_{\text{ref}}^t = f_\theta(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}_{\text{des}}^t) \tag{1}$$

that best maps the current robot joint angles $\boldsymbol{q}^t$, velocities $\dot{\boldsymbol{q}}^t$ and desired accelerations $\ddot{\boldsymbol{q}}_{\text{des}}^t$ to the torque necessary to achieve the desired acceleration. The model is optimized to minimize the mean squared error (MSE) between the torques predicted by the model and the ground truth from a data set consisting of numerous trajectories $\mathcal{D} = \{\{(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}_{\text{des}}^t), \boldsymbol{\tau}_{ref}^t\}_{t=1}^T\}_{n=1}^N$, where T refers to the number of time steps and N to the number of trajectories. For a single trajectory the loss is defined as:

$$\mathcal{L}_{MSE} = \frac{1}{T} \sum_{t=1}^T \left\| \boldsymbol{\tau}_{\text{ref}}^t - f_\theta(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}_{\text{des}}^t, \boldsymbol{\theta}) \right\|^2 \tag{2}$$

In this work, we consider three different model types for f_θ: classical rigid-body dynamics models (RBD), black-box models, and hybrid models that combine rigid-body dynamics with black-box models.

A. Rigid-body Dynamics Models

The robot manipulator is described as a chain of rigid bodies, their equations of motion (EOMs) in generalized coordinates can be written as

$$\boldsymbol{M}(\boldsymbol{q}^t)\ddot{\boldsymbol{q}}^t + \boldsymbol{V}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t) + \boldsymbol{G}(\boldsymbol{q}^t) = \boldsymbol{\tau}_{\text{ext}}^t + \boldsymbol{\tau}_{\text{ref}}^t \tag{3}$$

where $\boldsymbol{M}(\boldsymbol{q})$ denotes the mass matrix, $\boldsymbol{V}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ models the Coriolis and centrifugal effects, $\boldsymbol{G}(\boldsymbol{q})$ models the effect of gravity, $\boldsymbol{\tau}_{\text{ext}}^t$ denotes the external (exogenous) torques, and $\boldsymbol{\tau}_{\text{ref}}^t$ the actuation torques. For free space motion, in absence of contact, it can be considered that $\boldsymbol{\tau}_{\text{ext}}^t = \boldsymbol{0}$. For inverse dynamics identification, the linearity of the EOMs in the so-called barycentric parameters is commonly exploited [24]:

$$\boldsymbol{\tau}_{\text{ref}}^t = f_{\text{RBD}}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}^t) = \boldsymbol{Y}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}^t)\boldsymbol{\theta} \tag{4}$$

where the matrix $\boldsymbol{Y}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}^t)$ only depends on the robot kinematic properties and $\boldsymbol{\theta} \in \mathbf{R}^{10k}$ is a parameter vector that contains the values $\boldsymbol{\theta}_k = [m_k, l_{k,x}, l_{k,y}, l_{k,z}, L_{k,xx}, L_{k,xy}, L_{k,xz}, L_{k,yy}, L_{k,yz}, L_{k,zz}]^\mathsf{T}$ for each link k. The set of parameters consists of the mass m_k, the first mass moment $\boldsymbol{l}_k$ and the six elements characterizing the symmetric inertia tensor $\boldsymbol{L}_k$; where the latter is the inertia expressed in the link frame, which is in the joint connecting link k to link $k-1$ [1]. The inertia tensor in the center of mass $\boldsymbol{I}_k$ and the link frame inertia tensor $\boldsymbol{L}_k$ are related by the parallel axis theorem

$$\boldsymbol{L}_k = \boldsymbol{I}_k - m_k \boldsymbol{S}(\boldsymbol{r}_k)\boldsymbol{S}(\boldsymbol{r}_k) = \boldsymbol{I}_k + \frac{1}{m_k}\boldsymbol{S}(\boldsymbol{l}_k)^\mathsf{T}\boldsymbol{S}(\boldsymbol{l}_k), \tag{5}$$

where $\boldsymbol{l}_k = m_k \boldsymbol{r}_k$, with $\boldsymbol{r}_k$ the position of the center of mass of link k in the link frame, and $\boldsymbol{S} \colon \mathbf{R}^3 \to \mathbf{R}^{3\times3}$ the skew symmetric matrix operator such that $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{S}(\boldsymbol{a})\boldsymbol{b}$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbf{R}^3$. By linearity of Eq. (4), one can estimate the inertial parameters using ordinary least square methods [1]. Despite nonlinearity of the matrix $\boldsymbol{Y}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}^t)$ from Eq. (4) in its arguments, its columns are not generally linearly independent. In order to ensure Eq. (2) has a unique minimizer, the estimation usually aims at finding a set of "base parameters" [15] instead.

Additional constraints restrict the values of the inertial parameters. Yoshida and Khalil [30] first introduced the conditions for physically consistent inertial parameters,

$$\begin{cases} m_k > 0 \\ \boldsymbol{I}_k \succ \boldsymbol{0} \end{cases} \tag{6}$$

which states, that all masses need to be positive and the inertia tensors positive definite.

Traversaro et al. [28] highlighted, that those two conditions are not sufficient to guarantee fully physically consistent inertial parameters. The eigenvalues of the inertia tensor must fulfill the so-called triangle inequalities. The inertia tensor can be expressed by the principal axes of rotation and the according principal moments of inertia Y_X, Y_Y, Y_Z. One can decompose the inertia tensor,

$$\boldsymbol{I}_k = \boldsymbol{R}\boldsymbol{Y}_k\boldsymbol{R}^\mathsf{T} \tag{7}$$

in a rotation matrix $\boldsymbol{R}$ and a diagonal matrix $\boldsymbol{Y}_k$ with the principal moments of inertia on the diagonal. These must be positive and must fulfil the following inequalities

$$\begin{cases} Y_X + Y_Y > Y_Z \\ Y_Y + Y_Z > Y_X \\ Y_X + Y_Z > Y_Y \end{cases} \tag{8}$$

to enforce a positive mass density. A set of inertial parameters is only fully physically consistent, if all three conditions in Eq. (6) and Eq. (8) are satisfied. In [28] a manifold optimization method is presented to estimate fully physically consistent parameters.

Wensing et al. [29] have formulated the linear inequalities (LMI) in dependency of the rotational inertia matrix $\boldsymbol{I}_C$, which is required to have a positive-definite covariance of its mass distribution.

$$\Sigma_C = \frac{1}{2}\operatorname{tr}(\boldsymbol{I}_C)\mathbf{1}_3 - \boldsymbol{I}_C \succ \mathbf{0} \tag{9}$$

$\mathbf{1}_3$ denotes a 3x3 identity matrix and tr the trace operator, which takes the sum of the diagonal elements of a matrix. The above linear matrix inequality is shown to be equivalent to the inequalities in Eq. (8).

Wensing et al. [29] and Sousa and Cortesao [25] independently showed that all three conditions in Eq. (7) and Eq. (8) can be combined into a single LMI per robot link.

$$\boldsymbol{S}_k = \begin{bmatrix} \frac{\operatorname{tr}(\boldsymbol{L}_k)}{2}\mathbf{1}_3 - \boldsymbol{L}_k & \boldsymbol{l}_k \\ \boldsymbol{l}_k^\mathsf{T} & m_k \end{bmatrix} \succ \mathbf{0}. \tag{10}$$

The LMIs can be incorporated in a semi-definite programming (SDP) problem that minimizes Eq. (2).

B. Hybrid Inverse Dynamics Models

Several types of hybrid or grey-box models have been introduced in recent years. Physics-inspired neural networks such as Deep Lagriangen Networks (DeLaN) [13] or Hamiltonian networks [5] guarantee physically plausible dynamic models, which conserve energy. They incorporate structure of the system dynamics into the neural network architecture, are limited to model conservative forces, and use additional neural networks to capture all non-conservative effects. Residual hybrid models combine a rigid-body model with an additional error model to learn the error of the RBD model part [6]. For the residual model one can use either Gaussian processes (GPs) or neural networks (NNs). Nguyen-Tuong and Peters [17] and [16] used GP models to learn the error of an inverse dynamics model. Several recent works combine a standard Multilayer perceptron (MLP) neural network with a rigid-body model [7, 10, 14]. A similar approach has been applied to physics-inspired neural networks to capture the non-conservative forces [6]. Previous work successfully applied different recurrent neural networks for black-box inverse dynamics models [20, 21]. However, to the best of our knowledge, there is no previous work in applying recurrent models as residual models in inverse dynamics.

The Differentiable Newton Euler Algorithm (DiffNEA) implements a recursive Newton-Euler algorithm that uses virtual parameters to guarantee physical consistency of the inertial parameters during training [26, 14]. A similar implementation has been presented in [12]. The structure of the fully physically conditions from Eq. (6) and Eq. (9) are incorporated by using a set of unbounded virtual parameters to learn physically consistent inertial parameters with gradient descent. Lutter et al. [14] applied the DiffNEA algorithm with additional
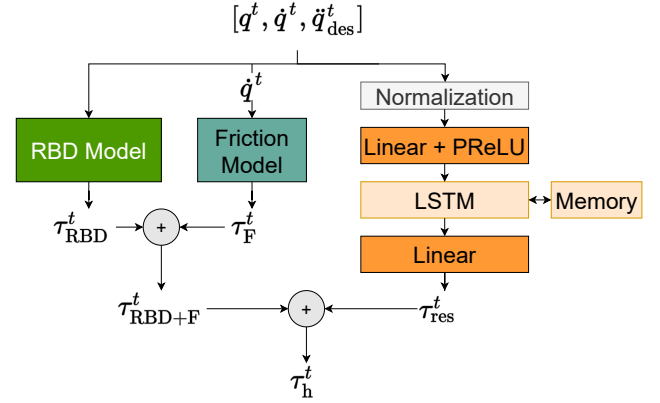

Fig. 1. Schematic of the residual hybrid model architecture proposed in section III consisting of the rigid-body, the sign friction function (Eq. (13)) and the LSTM error model, which has an additional input normalization layer. The final output torque sum τ_H^t is normalized for training.

actuator models in forward dynamics learning to capture dissipating forces. They trained the parameters jointly and achieved similar results to models learned in separate optimizations for systems up to 4 degrees of freedom. Our work is inspired by the method of Sutanto et al. [26] to incorporate the physical conditions into the optimization formulation. We introduce an alternative formulation, which uses a set of unbounded virtual parameters to directly learn a set of fully physically consistent link frame inertial parameters for Eq. (4). We show that both formulations are equally suited for joint residual hybrid model learning.

III. END-TO-END LEARNING OF HYBRID INVERSE DYNAMICS MODELS

Our hybrid architecture pairs a rigid body dynamics model with a recurrent neural network (RNN) which models the residuals of the RBD model and accounts for uncaptured effects such as the flexibility of the robot. Our proposed architecture consists of three components,

$$\boldsymbol{\tau}_{\text{ref}}^t = f_{\text{RBD}}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}_{\text{des}}^t) + f_{\text{F}}(\dot{\boldsymbol{q}}^t) + f_{\text{RNN}}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t, \ddot{\boldsymbol{q}}_{\text{des}}^t, \boldsymbol{\theta}_{\text{RNN}}), \tag{11}$$

where f_{RBD} is a traditional rigid-body model using the Equations of Motion (EOMs) from Eq. (4), f_{F} is a friction model, and f_{RNN} an RNN which can capture partially observable effects such as joint and link flexibilities and more complex friction effects. A general overview of the structured model is visualized in Figure 1.

A. Recurrent Residual Models

While the learned rigid-body dynamics model is globally valid, it is only able to capture rigid and stateless effects. By extending the RBD model with an additional error model, we can increase the accuracy of the model and also capture unmodeled effects such as slip-stick friction. While previous approaches rely on standard MLPs to capture the residual of the RBD model, we argue that recurrent models are better

suited to capture these unmodeled effects as most of these effects have an internal, unobserved state that needs to be inferred [9]. For our implementation we use Long Short-Term Memory networks (LSTM) [8]. LSTMs are commonly used in dynamic system modeling and have been successfully applied to black-box inverse dynamics model learning of a 7-DOF robot arm in [20] using offline data. Yet, as we are also interested in deploying our architecture on the real robot in real-time (i.e., 1kHz), we have to comply with tight computation time constraints and therefore resort to rather small network architectures. Our proposed residual network architecture consists of three layers: a single linear layer with PReLU activation functions, an LSTM layer, and an output layer consisting of seven neurons without any activation function to predict the seven individual residual joint torque values. We introduce additional normalization layers to normalize the input data for the neural network with predefined values. In between the model layers, we apply the layer normalization from [2]. Layer normalization is known to reduce the co-variance shift between train and test set and boosts training speed and generalization. For comparison, we also used a 3-layer MLP network to show the effectiveness of the LSTM at capturing temporal effects in the inverse dynamics data. Together with the fully differentiable DiffBary and DiffNEA methods (see next sub-section), we learn the parameters of the residual LSTM alongside the rigid body dynamics parameters and the parameters of the friction model.

B. Learning Physically Consistent Barycentric Parameters

To allow gradient-based end-to-end learning, we require an unconstrained parameterization of the rigid body dynamics model that always yields physically consistent parameters. To this end, we introduce an alternative formulation to DiffNEA [26] called "Differentiable Barycentric" (DiffBary). The method also uses a set of virtual unbounded parameters and computes the inertial parameters from the virtual parameters. Further, we apply this idea directly to the linear equation of motion with the barycentric parameters. We construct the LMI matrix S_k of Eq. (10), which expresses the conditions for the physical consistency from the virtual unbounded parameters and afterward retrieve the inertial parameters from it. By ensuring the positive definiteness of the matrix S_k of Eq. (10), we can guarantee that all physical conditions of the inertial parameters for the link k, Eq. (6) and Eq. (8), are satisfied. Inspired by the idea of DiffNEA, we learn the terms of the Cholesky decomposition of S_k and thereby enforce positive-definiteness of the matrix itself, i.e., for every generic link frame k, we introduce a lower triangular matrix A_k with 10 unbounded virtual parameters. The product $A_k A_k^\mathsf{T}$ together with a small positive bias $r = 10^{-8}$ on the diagonal entries guarantees strict positive-definiteness:

$$S_k = A_k A_k^\mathsf{T} + r \mathbf{1}_4. \tag{12}$$

Now, all inertial parameters are reconstructed from this matrix S_k. We refer to Eq. (19) and Eq. (20) in the appendix for detailed information on how to retrieve the individual

parameters. Next, we can directly use Eq. (4) to calculate the torques depending on the current system state.

As we can now guarantee fully physically consistent inertial parameters without additional constraints, this method can be straightforwardly applied for end-to-end learning of residual hybrid models or we can use it separately to identify inertial parameters. Our parameterization of the rigid body dynamics model allows us to learn physically consistent inertial parameters alongside the neural network weights using stochastic gradient descent algorithms such as Adam [11].

C. Friction Model

For the friction model, we employ a modified Coulomb model [18], which is only dependent on the joint velocity,

$$f_{\mathrm{F,k}}(\dot{q}_k^t) = \begin{cases} \sigma_{C,k} \mathrm{sign}(\dot{q}_k^t) & \text{if } |\dot{q}_k^t| > \dot{q}_{\lim,k}^t \\ \sigma_{C,k} \dot{q}_k^t / \dot{q}_{\lim,k}^t & \text{if } |\dot{q}_k^t| \le \dot{q}_{\lim,k}^t \end{cases} \tag{13}$$

where $\sigma_{C,k}$ is a friction coefficient and $\dot{q}_{\lim,k}^t = 0.02$ for every degree of freedom k. The friction terms appear linearly in the joint torques, we therefore extend our initial formulation of the EOMs in Eq. (4), to include the friction functions and add the parameters to our link parameter vector $\theta_k = [m_k, l_k, L_k, \sigma_{C,k}]^\mathsf{T} \in \mathbf{R}^{k \times 11}$

IV. OFFLINE EVALUATION

We evaluate our end-to-end learned structured inverse dynamics model shown in Figure 1 on a Franka Emika Panda robot and compare it to several rigid-body dynamics, black-box, and hybrid baseline models. First, we evaluate the approaches on their offline prediction performance. We compare a stand-alone version of DiffBary to other approaches for rigid body dynamics identification. Further, we analyze how the joint training of our residual hybrid architecture disentangles the RBD, friction, and residual components and investigate the generalization capabilities of the different models by explicit evaluation on out-of-distribution test data. Subsequently, we deploy the approaches on the real robot and evaluate their performance in motion tracking experiments using real-time impedance control on the real robot. We test the joint angle tracking accuracy of the different approaches using different controller feedback gains and show that structured models are suited for compliant and precise motion control.

A. Baselines

We compare our proposed hybrid model architecture with the following baselines.

Rigid Body Dynamics Baselines: We use a classical rigid-body dynamics baseline with fully physically consistent inertial parameters obtained using SDP [25] from system identification literature.

Hybrid Model Baselines: We use the DiffNEA algorithm [26] and the DeLaN model [13] as our hybrid physics-inspired neural network baselines. Further, we use a residual hybrid model with an MLP as a baseline for the residual LSTM.

Black-box Baselines: We also compare it with a black-box LSTM model. The black-box LSTM has the same architecture
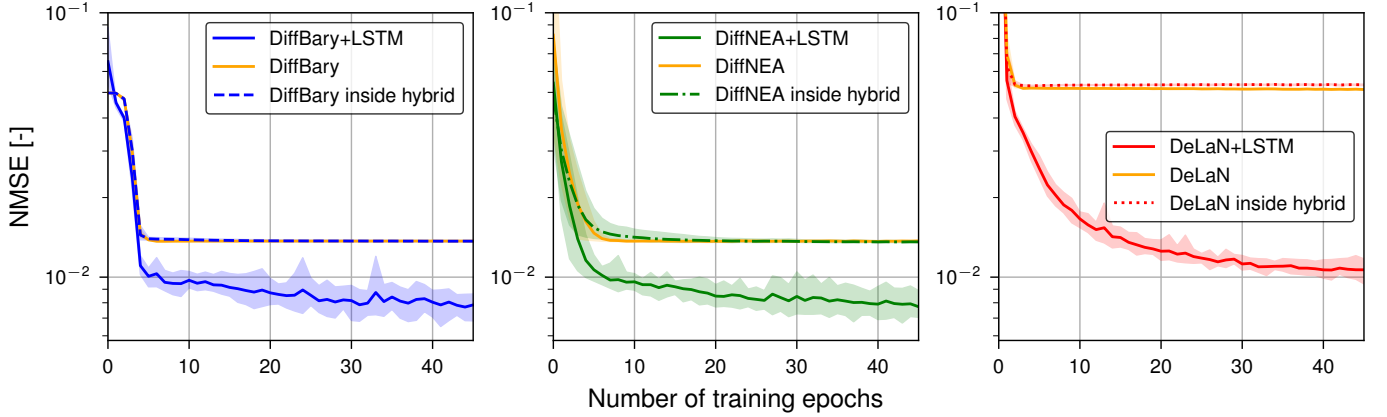
Fig. 2. Comparison of end-to-end inverse dynamics learning on a small system identification data set. All 3 rigid-body models are trained individually and combined with a residual LSTM as a hybrid model. The validation loss of the rigid-body models inside the hybrid model is monitored and visualized to compare the end-to-end learning efficiency of each model type. DiffNEA and DiffBary inside the hybrid model converge to the same solution after a few epochs for all 10 different seeds. The DeLaN model can also be learned jointly with an LSTM. All 3 models achieve a similar loss value if learned independently or in an end-to-end setting with the residual LSTM.

as our residual LSTM, consisting of one LSTM layer between two linear layers.

B. Data Collection

We collected training data for the offline experiments using a Franka Emika Panda robot. The robot joint angles and velocities are measured directly from the robot. Further, we approximate the acceleration of the robot using the filtered velocity signal. For the filtering, we use a non-causal filter, by applying a causal Butterworth filter once forward and then backward to cancel the phase shift. The filter has a cutoff frequency of 5 Hz. For the reference signal, we use the desired torque τ_{ref} computed by our stiff impedance controller; as opposed to the measured torque signal from the Panda robot, which internally performs friction compensation.

$$\tau_{\text{ref}} = \tau_{\text{fb}} + \tau_{\text{ff}} \qquad (14)$$

For the data collection, we use an inverse dynamics PD controller with proportional gains of $[400, 400, 200, 200, 200, 200]$ and damping gains tuned for critical damping. The controller uses a rigid-body inverse dynamics model, which has been optimized with SDP. The training data set is obtained from optimal experiment design [27], where trajectories designed to maximize the information gain and numerical conditioning of the regression problem obtained by stacking Eq. (4) for many measurement samples. The data collection procedure is commonly used for system identification of rigid-body dynamics models and hence we name this the system identification data. These trajectories have several advantages for system identification, such as time-domain averaging and avoiding frequency leakage errors. They try to cover large parts of the state-space by moving the robot with sine and cosine movements. The trajectories for the joint angles are described by

$$q_i(t) = \sum_{l=1}^{N} \frac{a_l^i}{\omega_f l} \sin(\omega_f l t) - \frac{b_l^i}{\omega_f l} \cos(\omega_f l t) + q_0 \qquad (15)$$

with ω_f the fundamental pulsation [27]. Before optimizing the trajectories, the initial values for the parameters a_l^i, b_l^i, and q_0 are chosen randomly. Furthermore, ω_f is chosen sufficiently low, such that a large amplitudes can be achieved without violating velocity and acceleration bounds. On the other hand, N is selected such that excitation of high-frequency modes is avoided. Each of the 20 trajectories generated are executed in two manners. Once with a maximum velocity $0.1\dot{q}_{\text{max}}$ and once faster with a maximum velocity of $0.4\dot{q}_{\text{max}}$, where $\dot{q}_{\text{max}}$ is as provided by the robot manufacturer. All the signals are collected at 1000 Hz. We generate 26 different trajectories by varying the seed of the random number generator used to initialize the optimizer and execute them in a slow and fast manner. In total, we collected around 5 hours of data, i.e., about 17.5 million data points. The recorded data was then subsequently split into a separate training, validation, and test set containing 42, 6, and 4 recorded movements, respectively. The test set consists of two trajectories executed in a slow and fast manner, which are not in the training or validation data. We stop training by monitoring the validation loss of the model. As a consequence, we decided to design the validation data to consist of two trajectories executed with different settings unseen in the training, and two trajectories which are included with a different execution speed in the training data.

C. Offline Data Experiments

To compare the offline model learning performance, all models are implemented in Pytorch [19] and trained using the mean squared error (MSE) loss for the normalized torque predictions. We use the same normalization for all torque values to guarantee a fair comparison of the results: $\tau_{\text{scaled},k} = \frac{\tau_k - \tau_{\min,k}}{\tau_{\max,k} - \tau_{\min,k}}$, where $\tau_{\min,k}$ and $\tau_{\max,k}$ respectively denote the minimum and maximum observed torque values across all recorded training data for joint index k. The values for all $\tau_{\min,k}$ and $\tau_{\max,k}$ are summarized in Table B. The offline ex-

	Training Loss	Validation Loss	Test Loss
SDP	**0.0070**±(0.0000)	**0.0074**±(0.0000)	**0.0085**±(0.0000)
DiffBary	0.0071±(10^{-4})	0.0075±(10^{-4})	0.0086±(10^{-4})
DiffNEA	0.0071±(10^{-4})	0.0075±(10^{-4})	0.0087±(10^{-4})
DeLaN	0.0081±(3×10^{-4})	0.13066±(0.0834)	0.2718±(0.2259)

TABLE I. Comparison of the average normalized torque prediction NMSE [−] and its standard deviation of inertial parameters retrieved using SDP, DiffNEA, and DiffBary method and DeLaN on a small training data set and two independent validation and test sets. The inertial parameters identified with SDP have the lowest average error. DiffNEA and DiffBary have similar performances. DeLaN has a similar training loss to the other models. However, it is not able to generalize its performance to unseen data.

periments are split into 3 sections: first, the different methods for identifying inertia parameters are evaluated against each other. Second, we test different models for their ability to be learned in an end-to-end setting together with a residual neural network. Finally, we evaluate our proposed hybrid model architecture against the current baseline models introduced in Section IV-A.

D. Inertial Parameter Identification

We compare the performance of the DiffBary method against rigid body dynamics baselines [25] and other hybrid baselines [26, 13] to identify a set of feasible inertial parameters. To do so, we train all models on a small data set consisting of 2 system identification trajectories, without the friction and residual components. These trajectories follow the same path but are executed in a slow and fast manner.

The DiffBary, DiffNEA, and DeLaN models are trained in Pytorch using Adam optimizer with a batch size of 1000 and with a learning rate of 0.004 (0.001 for DeLaN). The DeLaN model is trained with two layers of 200 neurons. We stop the training of all models when no improvement in loss is observed after 10 consecutive epochs. We evaluate the models on the validation and test data set to analyze their generalization capabilities. The results of the torque prediction on the training and test datasets are shown in Table I. The hybrid formulations of DiffBary and DiffNEA achieve similar torque prediction errors as that of classical RBD models obtained by SDP, on both training and test datasets. Thus we conclude that the hybrid formulations of DiffBary and DiffNEA are reliable models for finding a set of feasible inertial parameters. We investigate this claim further using model control experiments in Section V-B. As seen in Table I the DeLaN model fails at generalizing to unseen data, though it achieves a similar prediction error on the training set. Interestingly the methods to identify inertial parameters do not converge to the same solutions. We hypothesize, that this is explained by the different solvers that are used and to what accuracy they solve the regression problems. That is, the stochastic Adam optimizer in a mini-batch optimization scheme for the PyTorch model versus a (deterministic) interior-point method for the SDP. An overview of the resulting base parameters of DiffBary and SDP method is summarized in Table B of the Appendix.

E. Joint Structured Model Learning

We now compare the performance of different rigid-body dynamics formulations when learned jointly with the residual neural network weights as in Figure 1, using gradient descent methods. Therefore, DiffBary is tested against the DiffNEA and the DeLaN model in an end-to-end learning setting with the residual LSTM. We investigate if the joint learning correctly identifies the inertial parameters of the RBD model inside the hybrid model shown in Figure 1 without the friction part. To do so, we record the torque prediction error by the RBD model component in addition to the torque prediction error from the hybrid model on the validation data set. As a baseline, we train the individual rigid-body dynamic models separately on the same data set without the residual part.

We deploy the Adam optimizer with a learning rate of 0.004 (0.001 for DeLaN) with a batch size of 1000 and train all models on 10 random seeds to track the average performance. All residual hybrid models with LSTMs are trained using small sequences of data with 100 time steps each. The LSTM has an internal hidden state, which is updated during each time step in a sequence. We set the hidden state to zero at the beginning of every sequence. Training the models on the smaller parts of the trajectories instead of the complete ones makes the training more stable. The residual LSTM inside each hybrid model has the hyperparameters listed in Table B of the appendix. The results of our experiment are visualized in Figure 2. The average validation loss of all model types is shown together with the 5 percentile as the lower bound and the 95 percentile as the upper bound. As seen in Figure 2 for both DiffNEA and our method, the joint learning of the RBD model with a residual LSTM does not affect the correct identification of a valid set of inertial parameters for the RBD model. For both DiffNEA and DiffBary, the RBD model converges to an optimal value quickly in a few epochs. After the first few epochs, the model learns to capture the non-rigid-body dynamics effects using the residual LSTM resulting in a further decrease in the hybrid loss. Thus our architecture correctly disentangles the RBD and residual effects in an interpretable manner. We make a similar observation for DeLaN but with a much higher torque prediction error from both RBD and residual model components. We conclude that both DiffNEA and DiffBary are suited for joint end-to-end learning of a hybrid residual model. In the following experiments, we choose the DiffBary method, however, the DiffNEA method would perform equally well.

F. Inverse Dynamics Learning Comparison

Now, we aim to learn a highly precise inverse dynamics model applicable for motion tracking with the real robot. We study different inverse dynamics models trained on the complete dataset and evaluate their offline torque prediction errors on a separate test data set detailed in Section IV-B. For all models, we apply early stopping, where the training is interrupted when the validation loss does not improve after 30 consecutive epochs. A learning rate scheduler is deployed to lower the learning rate by the factor of 0.1 after 30 epochs. All

Model Type	Optimization	Evaluation		
		Train Loss (NMSE)	Val Loss (NMSE)	Test Loss (NMSE)
RBD+Sign+LSTM	E2E	$0.0021\pm(6\times10^{-4})$	$0.0025\pm(10^{-4})$	$\mathbf{0.0033}\pm(4\times10^{-4})$
	2S	$0.0013\pm(5\times10^{-4})$	$\mathbf{0.0022}\pm(2\times10^{-4})$	$0.0035\pm(3\times10^{-4})$
RBD+Sign+MLP	2S	$\mathbf{0.0004}\pm(10^{-4})$	$0.0030\pm(3\times10^{-4})$	$0.0065\pm(4\times10^{-4})$
RBD+Sign	E2E	$0.0038\pm(10^{-4})$	$0.0036\pm(10^{-4})$	$0.0037\pm(10^{-4})$
	2S	$0.0035\pm(0.0000)$	$0.0036\pm(0.0000)$	$0.0035\pm(0.0000)$
RBD	E2E	$0.0082\pm(2\times10^{-4})$	$0.0076\pm(10^{-4})$	$0.0082\pm(10^{-4})$
	2S	$0.0080\pm(0.0000)$	$0.0075\pm(0.0000)$	$0.0081\pm(0.0000)$
LSTM	E2E	$0.0036\pm(0.0011)$	$0.0056\pm(3\times10^{-4})$	$0.0089\pm(9\times10^{-4})$

TABLE II. Comparison of different inverse dynamics models average torque prediction accuracies on different data sets. The performance of the end-to-end trained (E2E) hybrid model is disentangled by showing the accuracy of the RBD model part and the RBD model with Sign friction(RBD+Sign) against the performance with the residual LSTM. The hybrid model trained in 2 steps (2S) has the best validation performance, while its version trained end-to-end (E2E) has the lowest average test loss. Extending the rigid-body dynamics model (RBD) with the sign friction function (RBD+Sign) boosts its average performance. The LSTM is not able to generalize to unseen data outside the vicinity of the training distribution.
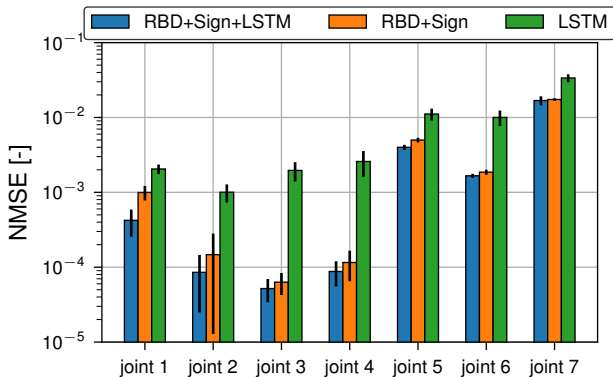

Fig. 3. Comparison of our proposed residual hybrid model trained end-to-end (E2E), its rigid-body dynamics model with sign friction (RBD+Sign), and a black-box LSTM on the test data set consisting of four system identification trajectories. The average individual torque prediction NMSE for all joints is plotted for 10 different seeds. Both, the rigid-body dynamics model with friction, and the hybrid model outperform the black-box LSTM on all joints. The additional residual LSTM in combination with the RBD model boosts the average performance for all joints.

models are trained with a batch size of 1000 samples and the Adam optimizer with a learning rate of 0.004. An overview of the hyperparameters of the training is summarized in Table B of the appendix. For the evaluation, the models predict the whole trajectories in one large sequence. This allows the LSTM models to use their internal memory throughout the whole trajectory.

End-to-End vs 2 Step Learning: First, we analyze the capabilities of end-to-end learning against a separate system identification for the residual hybrid model, which will be referred to as the "2-steps" model. The training of the 2-steps model involves two steps. In the first step, inertial parameters are identified using SDP. In the second step, the residual weights of the LSTM are learned in a second optimization using gradient descent with the RBD model parameters fixed. The performance of both model types is summarized in Table II. The results show that the 2 steps model has a lower training and validation accuracy, while the end-to-end model has a lower test error. The average performance of the RBD model

inside the end-to-end hybrid model (RBD E2E) confirms the results of Section IV-E. Optimizing the barycentric weights alongside the neural network weights achieves similar accuracy as the RBD model optimized by SDP.

Residual LSTM vs MLP: Secondly, we compare two variants of the residual hybrid model, 1) with a residual LSTM and 2) with a residual MLP. We perform this ablation to evaluate the influence of the internal memory on the hybrid model's accuracy. The torque prediction errors of Table II show that the residual MLP achieves a lower torque prediction loss on the training data, while the LSTM outperforms the MLP on the validation and test loss. Our ablation suggests that the LSTM fares better when it comes to generalization capabilities. Thus introducing internal memory into the hybrid model improves the generalization accuracy at the cost of task-specific accuracy.

Hybrid vs Black-Box LSTM: Finally, we compare our hybrid architecture with a black-box LSTM. The individual torque prediction losses on the test data are visualized in Figure 3. The black-box LSTM achieves similar training loss while having a higher validation and test loss. The residual hybrid model with the LSTM performs well on all datasets and is still more accurate than the baseline RBD model on unseen data, except for the last joint. Comparing the bar plots of the hybrid model against the RBD shows, that the hybrid model achieves lower task-specific prediction error than its rigid-body baseline with friction while maintaining the good generalization performance of the RBD model. The general improvement of the hybrid models over the RBD variants appears to be small in Table II and Figure 3 of the offline evaluation. However, the real robot experimental results in Table III show, that these small differences in the average torque prediction lead to large motion tracking errors.

V. MOTION TRACKING EXPERIMENTS

Motion tracking experiments evaluate the performance of inverse dynamics models and test their generalization abilities. We deploy different inverse dynamics models inside a joint-space impedance-controlled Panda robot. The inverse dynamics models are deployed as the feed-forward term inside
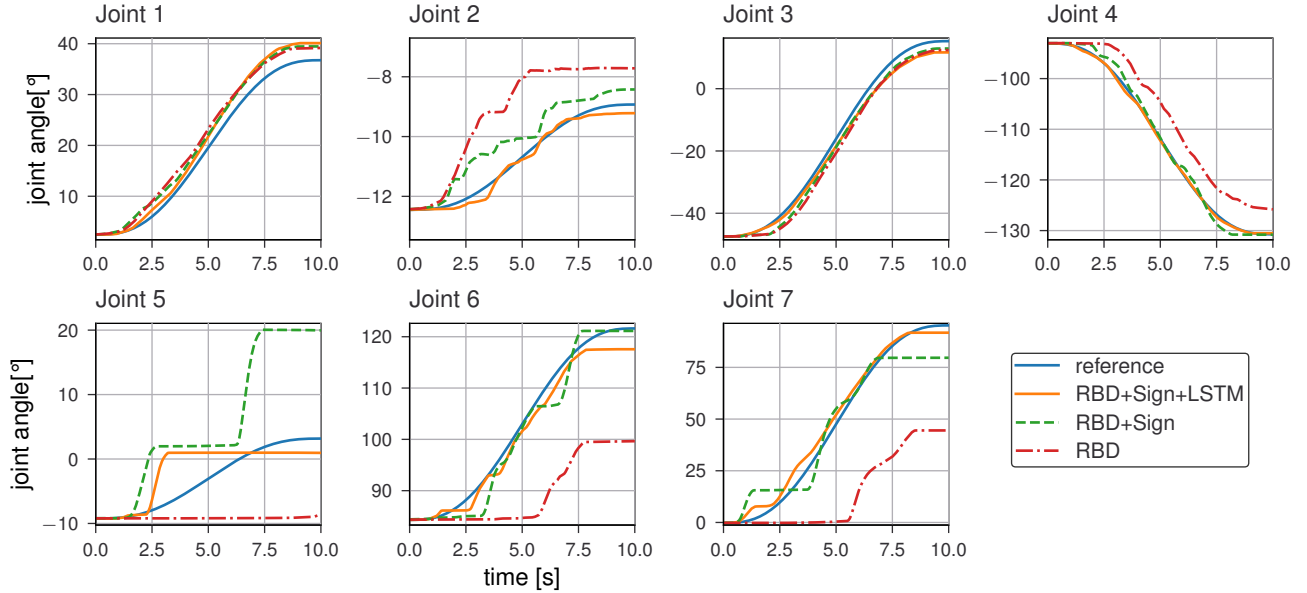
Fig. 4. Motion tracking performance for a minimum-jerk trajectory of the proposed end-to-end learned hybrid model. The tracking performance is disentangled into 3 models by testing the rigid-body dynamics model (RBD) inside the hybrid model independently with sign friction and without. All models are combined with the medium controller settings on a 10 seconds movement. The results show, that the residual hybrid model is able to further increase the tracking accuracy for 5 out of 7 joints.

the controller. The robot consecutively performs 5 minimum-jerk trajectories [3], which are generated from a set of joint state set-points. The impedance controller imposes a desired dynamic behavior of a spring-mass-damper system for each generalized robot coordinate independently and uses the inverse dynamics to compute the corresponding reference torques. A detailed description of the controller is given in Section B of the Appendix.

We evaluate the following models: an end-to-end (E2E) residual hybrid model, a hybrid model trained in 2 steps (2S), a residual hybrid model with an MLP, an RBD model optimized with SDP, and one RBD model identified inside the end-to-end learned hybrid model and a black-box LSTM. All controller variants with an LSTM inside are initialized before starting the experiment. Therefore, the control loop is run 500 iterations to update the internal memory of the LSTM to the current system state.

A. Experiment Details

All trajectories are executed in slow and fast versions of 2 and 10 seconds to test the generalization performance of the different models to different velocities. For the final evaluation we record the joint state desired pose q_{des}^t and current pose q^t during the execution at 1000 Hz. The final position offset at the end of every movement is evaluated in addition to the significance of the final position error for practical applications. We evaluate the performance of our models with three different controller gain settings and thereby illustrate the sensitivity of controller performance to these gains: Soft: $[1, 1, 1, 1, 1, 1, 1]$, Medium: $[50, 50, 50, 50, 50, 50, 50]$ and Stiff: $[400, 400, 200, 200, 200, 200]$ together with damping

constants tuned for critical damping. We remark that the stiff set of gains is the preferred setting for most activities in the lab. In the experiment with the lowest gains (Soft), which are two orders of magnitude lower than the stiffest (Stiff), the tracking performance is dominated by the trained model and the influence of the feedback controller is small. For the evaluation, we separately calculate the RMSE of the joint angles during movement, and the final position offset.

B. Experiment Results

The motion-tracking results of the different models are summarized in Table III. A visualization of the joint angle tracking performance on one sample sequence in Figure 4 further illustrates the performance increase of the hybrid model over its baseline RBD model with sign friction using the example of a trajectory with a medium controller gains.

Contrary to the offline torque prediction results, where the hybrid model trained in 2 steps excels, the end-to-end hybrid model achieved the best performance overall. The results support our hypothesis from the offline experimental results that recurrent models with an internal memory outperform stateless models like MLP. This supports the initial argument, that internal memory is important for the accuracy of the model. Embedding the RBD model into the recurrent model architecture allows the model to compensate for lower feedback gains while still improving the overall tracking performance in comparison to all RBD models. Thus, rendering them ideal for compliant and precise control applications.

We also note that the RBD model from the end-to-end learned hybrid model has a similar tracking performance as the RBD baseline models identified with SDP. Interestingly, the

	Controller gains	RBD+Sign+LSTM		RBD+Sign+MLP	RBD+Sign		RBD		LSTM
		2S	E2E	2S	E2E	2S	E2E	2S	E2E
Average joint angle RMSE[°]	Soft	16.207	**12.799**	22.442	22.892	18.372	19.013	18.867	failed
	Medium	2.599	**2.244**	3.719	2.948	6.764	11.613	12.495	failed
	Stiff	0.763	**0.744**	1.053	2.863	2.781	4.005	7.729	failed
Final joint angle offset[°]	Soft	24.45	**18.401**	34.427	30.053	24.379	29.757	29.733	failed
	Medium	**2.304**	3.125	3.955	3.447	9.075	16.388	18.538	failed
	Stiff	**0.596**	0.992	0.947	1.09	3.21	4.478	9.739	failed

TABLE III. Overview of the average joint angle tracking RMSE and the final joint angles error of different inverse dynamics models inside the impedance controller. All models are evaluated on 5 trajectories in a slow and fast manner. The results show, that the end-to-end hybrid model with the LSTM referred to as (E2E) outperforms all other models tested in the controller at 1000 Hz. Both RBD models identified end-to-end (E2E) with the DiffBary method and in two steps (2S) with the baseline SDP method have similar performance. The black-box LSTM model inside the controller immediately runs into the angle constraints of the joints. Hence we do not report results with black-box LSTM models.

end-to-end learned RBD model with the sign friction function outperforms the SDP variant. These results provide evidence, that the end-to-end learning of hybrid models correctly disentangles forces caused by inertia, friction, and other effects, which boost the individual performance of each model type. Thus, these models achieve the best generalization accuracy.

VI. Conclusion

We propose a new residual hybrid model for inverse dynamics of robot manipulators. It combines a rigid body dynamics model, a simple friction model, and a recurrent neural network. This combination allows the model to rely on strong physical priors for improved generalization and enables it to capture partially observable dynamics effects, e.g., stick-slip friction, and joint and link flexibilities. Furthermore, based on a recent method [25] for optimizing fully physical consist rigid body dynamics parameters and inspired by [26], we propose a model formulation suitable for unconstrained gradient-based optimization. Both methods allow joint, end-to-end training of all parts of the hybrid model, which results in improved online tracking results, compared to a two-step approach of first learning the rigid body dynamics and then fitting the network to the remaining error. Compared to rigid body dynamics, our hybrid approach shows improved accuracy, and compared to black-box methods it demonstrates better generalization capabilities. We deployed our model as the inverse dynamics map for a computed-torque impedance controller, controlling a real robot at a frequency of 1kHz. In this setup, the model allows us to significantly reduce the feedback gains of the controller while mitigating a loss of tracking accuracy, thus achieving more precise and compliant control. In future work, we want to investigate how we can exploit the fast gradient-based adaptation of all parameters in scenarios with changing dynamics, such as robot grasping.

Acknowledgments

References

[1] Christopher G Atkeson, Chae H An, and John M Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *stat*, 1050:21, 2016.

[3] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[4] A René Geist and Sebastian Trimpe. Structured learning of rigid-body dynamics: A survey and unified view from a robotics perspective. *GAMM-Mitteilungen*, 44(2): e202100009, 2021.

[5] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[6] Jayesh K Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. In *Learning for Dynamics and Control*, pages 328–337. PMLR, 2020.

[7] Kevin Hitzler, Franziska Meier, Stefan Schaal, and Tamim Asfour. Learning and adaptation of inverse dynamics models: A comparison. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 491–498. IEEE, 2019.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Karl Johanastrom and Carlos Canudas-De-Wit. Revisiting the lugre friction model. *IEEE Control Systems Magazine*, 28(6):101–114, 2008.

[10] Daniel Kappler, Franziska Meier, Nathan Ratliff, and Stefan Schaal. A new data source for inverse dynamics learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4723–4730. IEEE, 2017.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[12] Fernando Díaz Ledezma and Sami Haddadin. First-order-principles-based constructive network topologies:

An application to robot inverse dynamics. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 438–445. IEEE, 2017.

[13] Michael Lutter, Kim Listmann, and Jan Peters. Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7718–7725. IEEE, 2019.

[14] Michael Lutter, Johannes Silberbauer, Joe Watson, and Jan Peters. A Differentiable Newton-Euler Algorithm for Real-World Robotics. *International Conference on Robotics and Automation (ICRA)*, October 2021.

[15] Hirokazu Mayeda, Koji Yoshida, and Koichi Osuka. Base parameters of manipulator dynamic models. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1367–1372. IEEE, 1988.

[16] Franziska Meier, Daniel Kappler, Nathan Ratliff, and Stefan Schaal. Towards robust online inverse dynamics learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4034–4039. IEEE, 2016.

[17] Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *2010 IEEE international conference on robotics and automation*, pages 2677–2682. IEEE, 2010.

[18] Henrik Olsson, Karl Johan Åström, Carlos Canudas De Wit, Magnus Gäfvert, and Pablo Lischinsky. Friction models and friction compensation. *Eur. J. Control*, 4(3): 176–195, 1998.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[20] Elmar Rueckert, Moritz Nakatenus, Samuele Tosatto, and Jan Peters. Learning inverse dynamics models in o (n) time with lstm networks. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 811–816. IEEE, 2017.

[21] Vaisakh Shaj, Philipp Becker, Dieter Buchler, Harit Pandya, Niels van Duijkeren, C James Taylor, Marc Hanheide, and Gerhard Neumann. Action-conditional recurrent kalman networks for forward and inverse dynamics learning. *Conference on Robot Learning*, 2020.

[22] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 1846286417.

[23] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. Modelling, planning and control. *Advanced Textbooks in Control and Signal Processing. Springer,*, 2009.

[24] Cristóvao D Sousa and Rui Cortesao. Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach. *The International Journal of Robotics Research*, 33(6):931–944, 2014.

[25] Cristóvao D Sousa and Rui Cortesao. Inertia tensor properties in robot dynamics identification: A linear matrix inequality approach. *IEEE/ASME Transactions on Mechatronics*, 24(1):406–411, 2019.

[26] Giovanni Sutanto, Austin Wang, Yixin Lin, Mustafa Mukadam, Gaurav Sukhatme, Akshara Rai, and Franziska Meier. Encoding physical constraints in differentiable newton-euler algorithm. In *Learning for Dynamics and Control*, pages 804–813. PMLR, 2020.

[27] Jan Swevers, Chris Ganseman, D Bilgin Tukel, Joris De Schutter, and Hendrik Van Brussel. Optimal robot excitation and identification. *IEEE transactions on robotics and automation*, 13(5):730–740, 1997.

[28] Silvio Traversaro, Stanislas Brossette, Adrien Escande, and Francesco Nori. Identification of fully physical consistent inertial parameters using optimization on manifolds. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5446–5451. IEEE, 2016.

[29] Patrick M Wensing, Sangbae Kim, and Jean-Jacques E Slotine. Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution. *IEEE Robotics and Automation Letters*, 3(1):60–67, 2017.

[30] Koji Yoshida and Wisama Khalil. Verification of the positive definiteness of the inertial matrix of manipulators using base inertial parameters. *The International Journal of Robotics Research*, 19(5):498–510, 2000.

A. Learning fully physically consistent barycentric parameters using unbounded non-linear regression

The barycentric parameter equation is a linear mapping of the inertia parameters in the joint link frame to the torque.

$$\boldsymbol{\tau}_{\text{ref}} = \boldsymbol{Y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{\theta} \tag{16}$$

This equation can be optimized using simple gradient-descent methods. However, there are no guarantees of the physical consistency of the inertia parameters. Link frame inertia parameters are fully physically consistent if the linear matrix inequality of Eq. (10) is kept for every joint link. Writing out the equation yields

$$\boldsymbol{S}_k =$$
$$\begin{bmatrix} \frac{(L_{yy}+L_{zz}-L_{xx})}{2} & -L_{xy} & -L_{xz} & l_x \\ -L_{xy} & \frac{(L_{xx}+L_{zz}-L_{yy})}{2} & -L_{yz} & l_y \\ -L_{xz} & -L_{yz} & \frac{(L_{xx}+L_{yy}-L_{zz})}{2} l_z \\ l_x & l_y & l_z & m \end{bmatrix} \succ 0. \tag{17}$$

The fully physical consistency inequality for any joint link is defined in dependency of all 10 inertia parameters.

We apply the idea of [26] to use the terms of the Cholesky decomposition to guarantee a positive-definite matrix. Therefore, we introduce a lower triangular matrix $\boldsymbol{A}_k$ of a generic link frame k, which is expressed in dependency of 10 unbounded parameters.

$$\boldsymbol{A}_k = \begin{pmatrix} a_k & 0 & 0 & 0 \\ e_k & b_k & 0 & 0 \\ f_k & g_k & c_k & 0 \\ h_k & i_k & j_k & d_k \end{pmatrix} \tag{18}$$

Now the physical barycentric parameters are constructed from the positive definite matrix product $\boldsymbol{A}_k\boldsymbol{A}_k^\top$ of the lower triangular matrix $\boldsymbol{A}_k$ together with a small position bias r for all diagonals.

$$\boldsymbol{S}_k = \boldsymbol{A}_k\boldsymbol{A}_k^\top + r\boldsymbol{I}_4$$
$$= \begin{bmatrix} \delta_1 & \delta_5 & \delta_6 & \delta_8 \\ \delta_5 & \delta_2 & \delta_7 & \delta_8 \\ \delta_6 & \delta_7 & \delta_3 & \delta_{10} \\ \delta_8 & \delta_9 & \delta_{10} & \delta_4 \end{bmatrix}. \tag{19}$$

where,
$$\delta_1 = a_k^2 + r,$$
$$\delta_2 = e_k^2 + b_k^2 + r,$$
$$\delta_3 = f_k^2 + g_k^2 + c_k^2 + r,$$
$$\delta_4 = h_k^2 + i_k^2 + j_k^2 + d_k^2 + r,$$
$$\delta_5 = a_k e_k,$$
$$\delta_6 = a_k f_k$$
$$\delta_7 = e_k f_k + g_k b_k,$$
$$\delta_8 = a_k h_k,$$
$$\delta_9 = c_k h_k + i_k b_k,$$
$$\delta_{10} = f_k h_k + g_k i_k + c_k j_k$$

Now all the inertial parameters can be retrieved from the positive-definite matrix $\boldsymbol{S}_k$. Obtaining the values for L_{xx}, L_{yy}, L_{zz} requires additional calculations,

$$\begin{aligned} L_{xx} &= \frac{(L_{xx}+L_{zz}-L_{yy})}{2} + \frac{(L_{xx}+L_{yy}-L_{zz})}{2} \\ &= e_k^2 + b_k^2 + f_k^2 + g_k^2 + c_k^2 + 2r \\ L_{yy} &= \frac{(L_{yy}+L_{zz}-L_{xx})}{2} + \frac{(L_{yy}+L_{xx}-L_{zz})}{2} \\ &= a_k^2 + f_k^2 + g_k^2 + c_k^2 + 2r \\ L_{zz} &= \frac{(L_{yy}+L_{zz}-L_{xx})}{2} + \frac{(L_{xx}+L_{zz}-L_{yy})}{2} \\ &= a_k^2 + e_k^2 + b_k^2 + 2r, \end{aligned} \tag{20}$$

while other parameters can directly be reconstructed

$$\begin{aligned} L_{xy} &= -a_k e_k \\ L_{xz} &= -a_k f_k \\ L_{yz} &= -e_k f_k - g_k b_k \\ l_x &= a_k h_k \\ l_y &= c_k h_k + i_k b_k \\ l_z &= f_k h_k + g_k i_k + c_k j_k \\ m &= h_k^2 + i_k^2 + j_k^2 + d_k^2 + r. \end{aligned} \tag{21}$$

For a 7 degree of freedom robot its barycentric parameters are identified from scratch using 7 sets with 10 virtual unbounded parameters $a_k, ..., h_k$, one for every joint k. For our implementation we initialize all unbounded virtual parameters with a mean of 0 and a standard deviation of 0.001.

B. Joint space impedance control

For the motion tracking experiments a classic joint space impedance controller is used [22]. The impedance controller mimics the dynamics of mass-spring-damper system with adjustable parameters according to

$$\boldsymbol{M}_{\text{des}}(\ddot{\boldsymbol{q}}^t - \ddot{\boldsymbol{q}}_{\text{des}}^t) + \boldsymbol{B}_{\text{des}}(\dot{\boldsymbol{q}}^t - \dot{\boldsymbol{q}}_{\text{des}}^t) + \boldsymbol{K}_{\text{des}}(\boldsymbol{q}^t - \boldsymbol{q}_{\text{des}}^t) = \boldsymbol{\tau}_{\text{ext}}^t, \tag{22}$$

where $\boldsymbol{M}_{\text{des}}$ is the desired inertia matrix, $\boldsymbol{B}_{\text{des}}$ the virtual dampening matrix, $\boldsymbol{K}_{\text{des}}$ denotes the spring constant matrix and $\boldsymbol{\tau}_{\text{ext}}$ are the joint torques caused by external forces. We use feedback-linearization in joint acceleration space and insert Eq. (22) into the inverted manipulator dynamics from Eq. (3). This yields the following torque control feedback law

$$\begin{aligned} \boldsymbol{\tau}_{\text{ref}}^t = \ & \boldsymbol{V}(\boldsymbol{q}^t, \dot{\boldsymbol{q}}^t) + \boldsymbol{G}(\boldsymbol{q}^t) - \boldsymbol{\tau}_{\text{ext}}^t \\ &+ \ \boldsymbol{M}(\boldsymbol{q}^t)\boldsymbol{M}_{\text{des}}^{-1}\left(\boldsymbol{\tau}_{\text{ext}}^t - \boldsymbol{B}_{\text{des}}(\dot{\boldsymbol{q}}^t - \dot{\boldsymbol{q}}_{\text{des}}^t) - \boldsymbol{K}_{\text{des}}(\boldsymbol{q}^t - \boldsymbol{q}_{\text{des}}^t)\right) \\ &+ \ \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\text{des}}^t, \end{aligned}$$

with $\boldsymbol{\tau}_{\text{ref}}^t$ being the actuation torque. In case of free space motion, where $\boldsymbol{\tau}_{\text{ext}}^t = \boldsymbol{0}$, the above control law simplifies to classical inverse dynamics PD control.

	Residual LSTM	Residual MLP	Black-box LSTM	DeLaN
Input dimension	21	21	21	21
Output dimension	7	7	7	7
Number LSTM layers	1	-	1	-
LSTM hidden dimension	50	-	50	-
Number linear layers	1	2	1	2
Linear dimension	100	200	100	200
Linear activation	PReLU	PReLU	PReLU	SoftPlus
Number of weights	33358	46008	33358	47629
Time steps	100	1	100	100
Batch size	1000	1000	1000	1000
Learning rate	0.004	0.004	0.004	0.001
Inverse Dynamics Learning Comparison				
Training samples	[135210, 100, 21]			
Validation samples	[25930, 100, 21]			
Min torque values [Nm]	[-10.00, -63.80, -28.24, -24.98, -3.95, -4.21, -1.36]			
Max torque values [Nm]	[9.78, 61.37, 27.64, 27.9, 3.94, 3.84, 1.84]			

TABLE IV. Overview of the optimized hyperparameters of the different residual and black-box LSTMs, the DeLan model and the torque values used for normalization.

Base Parameter	DiffBary	SDP
$L_{1yy} + L_{2zz}$	0.14240	0.05440
$L_{2xx} - L_{2zz} + L_{3zz} + \frac{79}{125}l_{3y} + \frac{372199}{4000000}(m_3 + m_4 + m_5 + m_6 + m_7)$	1.21090	1.71400
L_{2xy}	-0.00560	-0.00040
L_{2xz}	0.02060	0.03450
$L_{2yy} + L_{3zz} + \frac{79}{125}l_{3y} + \frac{372199}{4000000}(m_3 + m_4 + m_5 + m_6 + m_7)$	1.22020	1.72490
L_{2yz}	-0.06220	0.00600
l_{2x}	-0.05690	-0.05410
$l_{2z} + l_{3y} + \frac{79}{250}(m_3 + m_4 + m_5 + m_6 + m_7)$	3.44070	3.42250
$L_{3xx} - L_{3zz} + L_{4zz} + \frac{1089}{160000}m_3$	0.00860	-0.00890
$L_{3xy} - \frac{33}{400}l_{3y}$	-0.01050	-0.14800
L_{3xz}	-0.00010	0.00230
$L_{3yy} + L_{4zz} - \frac{1089}{160000}(-m_3 - 2m_4 - 2m_5 - 2m_6 - 2m_7)$	-0.08700	-0.02990
L_{3yz}	-0.01160	0.00570
$l_{3x} + \frac{33}{400}(m_3 + m_4 + m_5 + m_6 + m_7)$	0.84560	0.81150
$l_{3z} - l_{4y}$	-0.00840	-0.01160
$L_{4xx} - L_{4zz} + L_{5zz} + \frac{6}{125}l_{5y} + \frac{1089}{160000}m_4 + \frac{617049}{4000000}(m_5 + m_6 + m_7)$	0.76810	0.82510
$L_{4xy} + \frac{33}{400}l_{4y}$	0.00090	-0.00370
L_{4xz}	-0.00010	0.01910
$L_{4yy} + L_{5zz} + \frac{96}{125}l_{5y} - \frac{1089}{160000}m_4 + \frac{562599}{4000000}(m_5 + m_6 + m_7)$	0.68210	0.77540
L_{4yz}	-0.00290	0.00110
$l_{4x} - \frac{33}{400}(m_4 + m_5 + m_6 + m_7)$	-0.52140	-0.51670
$l_{4z} + l_{5y} + \frac{48}{125}(m_5 + m_6 + m_7)$	2.05570	2.05900
$L_{5xx} - L_{5zz} + L_{6zz} - \frac{121}{15625}(m_6 + m_7)$	0.03670	-0.00180
L_{5xy}	-0.00090	0.01240
L_{5xz}	0.00040	0.00080
$L_{5yy} + L_{6zz} - \frac{121}{15625}(m_6 + m_7)$	0.03690	0.02670
L_{5yz}	0.00530	0.00040
l_{5x}	0.00710	0.00610
$l_{5z} + l_{6y}$	-0.08870	-0.08830
$L_{6xx} - L_{6zz} + L_{7yy} + \frac{107}{500}l_{7z} + \frac{121}{15625}m_6 + \frac{19193}{1000000}m_7$	0.00990	0.01460
$L_{6xy} - \frac{11}{125}l_{6y}$	-0.01520	-0.00770
L_{6xz}	-0.01760	-0.00030
$L_{6yy} + L_{7yy} + \frac{107}{500}l_{7z} - \frac{121}{15625}m_6 + \frac{741}{200000}m_7$	0.03790	0.01550
L_{6yz}	-0.00680	0.00050
$l_{6x} + \frac{11}{125}(m_6 + m_7)$	0.22110	0.22340
$l_{6z} + l_{7z} + \frac{107}{1000} * m_7$	0.16480	0.16500
$L_{7xx} - L_{7yy}$	0.00020	-0.00090
L_{7xy}	-0.00040	-0.00060
L_{7xz}	-0.00060	0.00180
L_{7yz}	-0.00070	0.00100
L_{7zz}	0.00090	0.00430
l_{7x}	0.00720	0.00860
l_{7y}	0.00950	0.00990

TABLE V. Comparison of the identified Dynamic base parameters identified using SDP [25] and our new proposed DiffBary method

		joint 1	joint 2	joint 3	joint 4	joint 5	joint 6	joint 7
Final Joint Angle Error with Soft Gains								
RBD+LSTM+Sign	2S	12.654	20.557	12.039	8.238	29.701	19.135	68.824
RBD+LSTM+Sign	E2E	**10.245**	10.879	**6.773**	14.452	39.852	11.664	34.94
RBD+MLP+Sign	2S	31.706	31.421	54.849	19.944	26.753	16.283	60.034
RBD+Sign	E2E	29.429	16.983	30.662	13.045	63.4	12.895	43.959
RBD+Sign	2S	31.231	11.852	32.43	**9.26**	49.03	**7.574**	**29.273**
RBD	E2E	25.796	9.814	21.706	23.595	12.334	24.934	90.121
RBD	2S	20.5	**8.435**	28.757	23.062	**12.33**	24.929	90.121
Final Joint Angle Error with Medium Gains								
RBD+LSTM+Sign	2S	**1.145**	1.115	**1.144**	1.13	5.575	2.41	**3.608**
RBD+LSTM+Sign	E2E	1.855	**0.399**	1.82	**0.585**	8.411	**1.184**	7.621
RBD+MLP+Sign	2S	1.338	1.057	1.68	1.062	11.609	4.032	6.906
RBD+Sign	E2E	3.561	0.993	3.81	0.903	**3.049**	1.734	10.077
RBD+Sign	2S	4.409	0.69	4.904	1.101	4.084	1.84	46.497
RBD	E2E	3.701	0.726	3.486	4.02	12.069	22.522	68.194
RBD	2S	3.576	0.697	3.134	3.177	12.303	16.831	90.045
Final Joint Angle Error with Stiff Gains								
RBD+LSTM+Sign	2S	**0.119**	0.323	**0.321**	0.365	1.152	0.66	**1.235**
RBD+LSTM+Sign	E2E	0.245	**0.15**	0.504	**0.212**	3.244	**0.24**	2.351
RBD+LSTM	2S	0.379	0.514	0.34	1.677	6.586	3.857	9.618
RBD+MLP+Sign	2S	0.164	0.193	0.309	0.431	2.62	0.865	2.046
RBD+Sign	E2E	0.434	0.241	0.923	0.244	**0.787**	0.741	4.262
RBD+Sign	2S	0.54	0.177	1.228	0.343	2.379	0.92	16.884
RBD	E2E	0.322	0.18	0.6	1.067	3.76	6.084	19.331
RBD	2S	0.399	0.105	0.867	0.871	4.881	4.437	56.613

TABLE VI. Comparison of final joint angles offset averaged for all motion tracking trajectories for all different hybrid models with three different controller configurations

		joint 1	joint 2	joint 3	joint 4	joint 5	joint 6	joint 7
Average Root Mean Squared Error with Soft Gains								
RBD+LSTM+Sign	2S	9.175	13.139	8.47	**5.904**	17.176	13.621	45.967
RBD+LSTM+Sign	E2E	5.358	**6.347**	**4.024**	10.419	26.368	10.98	**26.096**
RBD+MLP+Sign	2S	19.259	18.881	33.772	12.486	27.59	**9.733**	35.372
RBD+Sign	E2E	19.492	10.012	20.51	9.06	43.468	10.253	47.449
RBD+Sign	2S	20.816	6.578	21.384	6.751	33.836	8.944	30.294]
RBD	E2E	15.483	8.643	14.473	14.75	7.72	15.607	56.413
RBD	2S	**3.012**	6.75	17.876	14.698	**7.718**	15.605	56.413
Average Root Mean Squared Error with Medium Gains								
RBD+LSTM+Sign	2S	**1.367**	0.779	**1.262**	1.105	4.811	2.064	6.802
RBD+LSTM+Sign	E2E	1.45	**0.379**	1.521	**0.621**	5.666	1.668	**4.401**
RBD+MLP+Sign	2S	2.088	0.751	1.905	1.141	10.354	2.822	6.971
RBD+Sign	E2E	3.191	0.928	3.46	0.925	**2.41**	**1.62**	8.102
RBD+Sign	2S	3.991	0.693	4.42	1.068	4.936	1.722	30.518
RBD	E2E	3.656	0.852	3.487	3.031	7.574	14.766	47.923
RBD	2S	3.596	0.653	3.339	2.935	7.702	12.865	56.374
Average Root Mean Squared Error with Stiff Gains								
RBD+LSTM+Sign	2S	**0.194**	0.241	**0.372**	0.398	1.512	0.701	1.923
RBD+LSTM+Sign	E2E	0.293	**0.13**	0.517	**0.187**	2.277	**0.414**	**1.392**
RBD+LSTM	2S	0.539	0.403	0.728	2.008	9.278	6.34	7.109
RBD+MLP+Sign	2S	0.319	0.219	0.544	0.345	3.054	0.711	2.181
RBD+Sign	E2E	0.466	0.272	1.017	0.276	**0.77**	0.643	3.176
RBD+Sign	2S	0.593	0.225	1.354	0.316	2.698	0.757	14.096
RBD	E2E	0.529	0.284	1.006	1.109	2.915	5.575	16.62
RBD	2S	0.545	0.193	1.058	1.027	3.997	4.417	42.864

TABLE VII. Comparison of root mean squared errors (RMSE) for all different hybrid models with all three controller configurations averaged for all motion tracking experiments