Dataset and Baseline for an Industrial Robot Identification Benchmark

Jonas Weigand[a,c] Julian Götz[a,c] Jonas Ulmen[b] Martin Ruskowski[a,c]

[a]Chair of Machine Tools and Control Systems, Technical University Kaiserslautern, Germany [b]Chair of Control Systems, Technical University Kaiserslautern, Germany [c]German Research Center for Artificial Intelligence, Kaiserslautern, Germany

jonas.weigand@mv.uni-kl.de, ORCID 0000-0001-5835-3106

julian.goetz@mv.uni-kl.de, ORCID 0000-0002-8126-9962

ulmen@eit.de, ORCID 0000-0003-1597-1523

martin.ruskowski@mv.uni-kl.de, ORCID 0000-0002-6534-9057

Date: April 9, 2023

Version 2.0. Changed the linear baseline model. Corrected the calculation of the coefficient of determination.

1 Introduction

We present an identification benchmark dataset for a full robot movement with a *KUKA KR300 R2500 ultra SE* industrial robot (Fig. 1, left side). It is a robot with a nominal payload capacity of $300\,\mathrm{kg}$, a weight of $1120\,\mathrm{kg}$ and a reach of $2500\,\mathrm{mm}$. It exhibits 12 states accounting for position and velocity for each of the 6 joints. The robot encounters backlash in all joints, pose-dependent inertia, pose-dependent gravitational loads, pose-dependent hydraulic forces, pose- and velocity-dependent centripetal and Coriolis forces as well as nonlinear friction, which is temperature-dependent and therefore potentially time-varying. As a reference to the nonlinear effects to be expected for this benchmark, a nonlinear robot model is presented in appendix A.

Although the robot mechanics are unmodified, the robot is driven by a custom-made controller based on hardware by the industrial supplier *B&R automation* [3–5] (Fig. 1, right side). This allows us comprehensive system access. Therefore, we are able to provide additional information about the dynamical behavior of the robot and the controller implementation. Hardware configuration and basic functionality were quickly realized. However, robot modelling, more advanced control approaches and network distributed computing were enabled by a joint collaboration of 25 researchers, technicians and students. Details on the contributions of each person are given in the appendix B. The robot configuration as well as an additional explanation of hardware and software is specified in the appendix C.

This work is structured as follows. The design of experiments and the data is presented in section 2. The applications related to forward and inverse identification are discussed in section 3. For comparable results on this benchmark, instructions on the metrics and figures of merit are given in section 4. In section 5 the figures of merit are given for a linear baseline trained on the data.

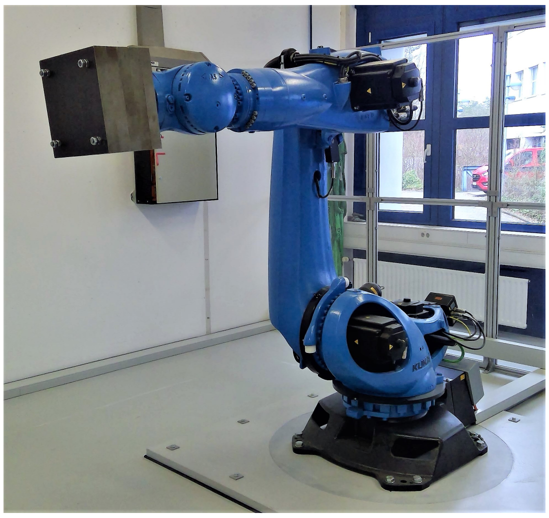
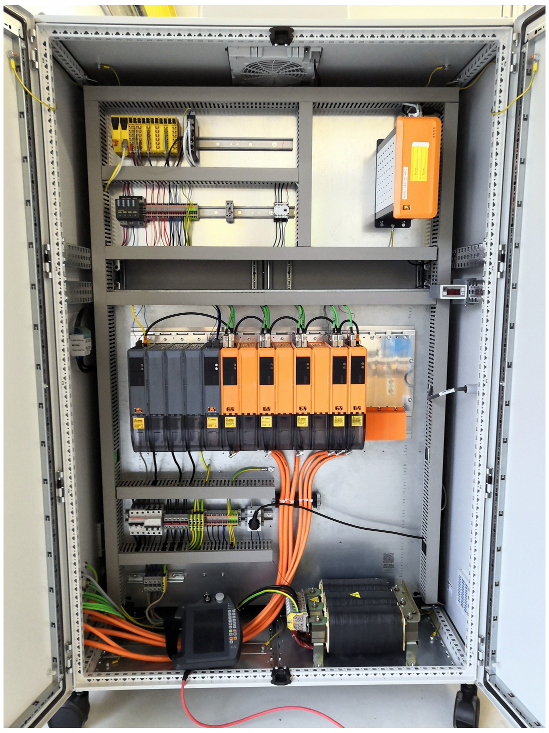

Figure 1: Industrial robot *KUKA KR300 R2500 ultra SE* with custom-made controller based on B&R components.

Table 1: File Overview

File Name	Type	Content
Robot_Identification_Benchmark_Description	pdf	This text document.
Robot_Identification_Benchmark_Without_Raw_Data	rar	Two mat-files, which contain the training and test data for forward and inverse black-box identification.
Robot_Identification_Benchmark_With_Raw_Data	rar	Mat-files with additional data of all experiments in high-frequency. MATLAB scripts used for data post-processing and baseline estimation.
Robot_Identification_Benchmark_Videos	rar	Videos of every experiment. Video timestamp corresponds to the raw data timestamp.

2 Dataset

2.1 Design of Experiments

The experiments are designed as 36 different trajectories with a duration of $60.6\,\mathrm{s}$ each. All movements start and end in homing position (see Fig. 1), so the data can be combined and shuffled in any order. Each trajectory is executed twice so that repeatability is explicitly tested. All in all, we have $2 \cdot 36 \cdot 60.6\,\mathrm{s} \approx 73\,\mathrm{min}$ of data, $67\,\mathrm{min}$ for training and $6\,\mathrm{min}$ for testing. In addition to the data, each experiment is recorded as a video.

The design of experiments for each trajectory is formulated as an optimization problem to incorporate all physical constraints such as position, velocity, acceleration, and jerk limits. We design a joint position reference trajectory, based on which the input, i.e., the motor feedforward and feedback torque, is computed at run-time. Each trajectory is formulated as the sum of sine and cosine functions to ensure continuous differentiability. Furthermore, optimizing for the sine and cosine coefficients $\mathbf{A} \in \mathbb{R}^{V \times N}$, $\mathbf{B} \in \mathbb{R}^{V \times N}$ reduces the number of optimization variables compared to a discrete-time formulation by multiple orders of magnitude. The sum of sine and cosine functions is defined as

$$q_n(k) = \sum_{v=1}^{V} \left(\frac{A_{v,n}}{\omega v} sin(\omega v k) - \frac{B_{v,n}}{\omega v} cos(\omega v k) \right) \tag{1}$$

with a base frequency $\omega \in \mathbb{R}$, a number of $v \in [1, .., V]$ sine and cosine functions, and $k \in [1, .., K]$ discrete time steps for each joint $n \in [1, ..., N]$.

In contrast to previous works [12], we do not optimize the experiment with respect to the condition number. Although this criterion greatly improves the measurement information quality, i.e. by ensuring that all model parameters are excited sufficiently, the condition number is still a model-dependent criterion. As this is a benchmark problem for nonlinear black-box identification, we decided against using model knowledge for the design of experiments. Instead, we rely on general design criteria, such as coverage of state-space, compliance with input and output limitations, and consideration of the frequency spectrum.

We set $V = 10$ coefficients for each frequency and $K = 60$ equally spaced time steps for computing the sine and cosine coefficients. The frequency grid in the design of experiments ranges from the base frequency $\omega = 1/60 \approx 0.0166\,\mathrm{Hz}$ to a maximum frequency of $1\,\mathrm{Hz}$ (compare with Fig. 6). The sample rate $\Delta t = t_{k+1} - t_k$ and therefore the number of time steps K vary depending on the task, as the trajectory time remains unchanged at $t_K = 60.6\,\mathrm{s}$. For estimating the sine and cosine coefficients, we set $\Delta t \approx 1\,\mathrm{s}$, $K = 60$ for a fast computation. For the high-frequency reference trajectory we set $\Delta t = 4\,\mathrm{ms}$, $K = 15147$ to match the feedback controller.

The optimization problem is implemented in MATLAB using CasADi with the IPopt solver [2, 10]. The optimization problem is given by

$$\mathbf{A}^*, \mathbf{B}^* = \underset{\mathbf{A},\mathbf{B}}{\text{argmin}} \quad J(\mathbf{q_k}) \tag{2a}$$

$$\text{s.t.} \quad \mathbf{q^{(m)}_{lb,k}} \leq \mathbf{q^{(m)}_{k}} \leq \mathbf{q^{(m)}_{ub,k}} \qquad \forall k \in [1, 2, ..., K], \forall m \in \{0, 1, 2, 3\} \tag{2b}$$

$$\mathbf{q^{(m)}_{k}} = \mathbf{0}, \qquad \forall k \in \{1, K\}, \forall m \in \{0, 1, 2, 3\} \tag{2c}$$

with the joint upper and lower joints limits $\mathbf{q^{(m)}_{ub,k}}$, $\mathbf{q^{(m)}_{lb,k}}$ for each time step k and each derivative m. The applied objective criterion is empty, i.e. $J(\cdot) = 0$. The optimizer mainly ensures that all trajectory derivatives start and end at zero (2c). Each new trajectory is different as the initial values of the sine and cosine coefficients are created randomly.[1]

The first constraint (2b) ensures that limits of the trajectory including the first 3 derivatives (velocity, acceleration, and jerk) are ensured for all time steps k. The second constraint (2c) ensures that all derivatives are zero at the start and the end of each trajectory. Choosing the same initial and final pose for all experiments is optional, yet it ensures that the data can be shuffled and combined without the loss of physical feasibility.

2.2 Post Processing

For the identification benchmark, we summarized the trajectories to a single input/output dataset. The original data is still provided for researchers, containing unfiltered, high-frequency measurements in a sample time of $4\,\mathrm{ms}$ with additional data, such as the reference trajectory and velocity measurements. In order to keep the benchmark comparable, please state clearly if this additional information is accessed for model training.

As post-processing the original measurements are filtered and re-sampled to $100\,\mathrm{ms}$ as presented in listing 1. The combined training data consists of 33 different trajectories, each executed twice, connected to $67\,\mathrm{min}$ and 39988 time steps. The test trajectory is independent and shorter, i.e. 3 trajectories, executed twice, $6\,\mathrm{min}$, 3636 time steps so that more details are visible in time series test figures.

Listing 1: Data Filtering.

```matlab
% define filter
dt_data       = 0.004;
filter_design = designfilt('lowpassiir', 'FilterOrder', 4, ...
    'PassbandFrequency', 2, 'PassbandRipple', 0.2,...
    'SampleRate', 1/dt_data);

% apply filtering
u      = filtfilt(filter_design, u')';
y      = filtfilt(filter_design, y')';

% reduce from 4 ms to 100 ms
n_step = 25;
u      = u(:, 1:n_step:end);
y      = y(:, 1:n_step:end);
```

The measured post-processed motor torques in Nm (including both feedforward and feedback control) are presented in Fig. 2 for the test dataset. Each row corresponds to a single joint. More precisely, we measure the actual motor winding currents and estimate the corresponding torques using the motor torque constant. The joint positions in deg are shown in Fig. 3 for the test data. The training data is presented in Fig. 4 and Fig. 5 respectively. Layout and units are analog to the test data.

[1]We tested different objective criteria, such as maximizing velocity or acceleration, i.e. $J(\mathbf{q_k}) = \sum_{k=1}^{K} \dot{\mathbf{q}}_k^2$, or a following discontinuous reference, i.e. $J(\mathbf{q_k}) = \sum_{k=1}^{K} (\mathbf{q}_k - \mathbf{q}_{R,k})^2$. It is actually easier, to identify a robot model in the high-velocity range only, as some stationary nonlinear effects can almost be omitted. Preliminary examinations revealed that the empty objective function captures both stationary and high-velocity effects well.

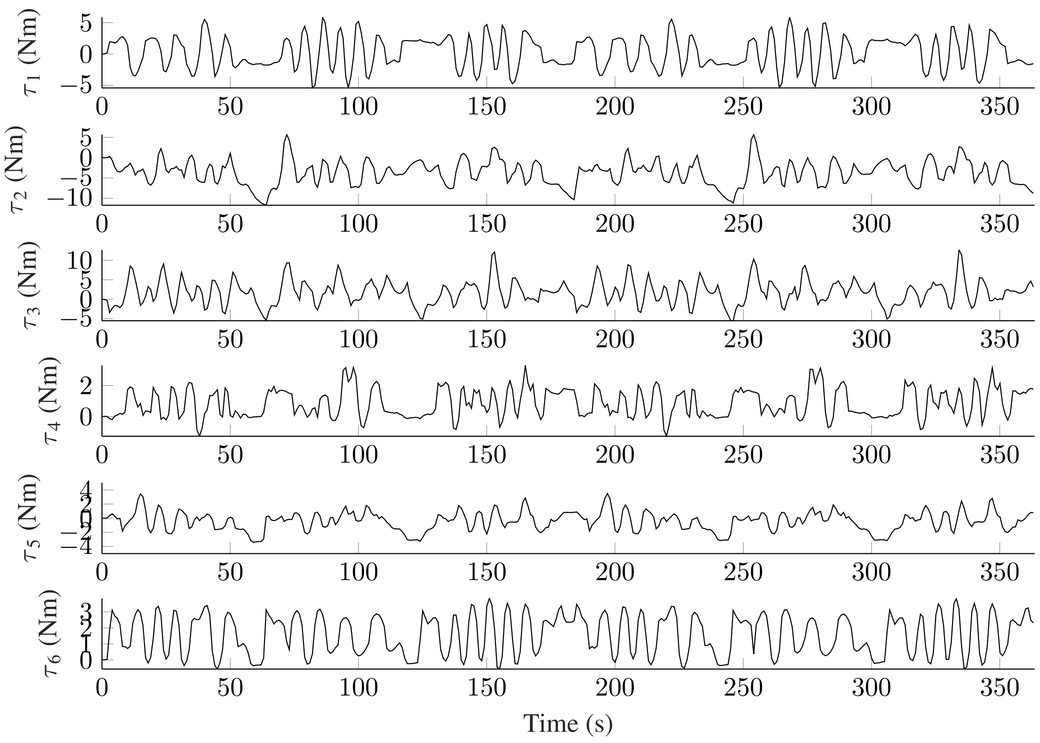

Figure 2: Measured motor torques in Nm as test data. Each row corresponds to one joint.

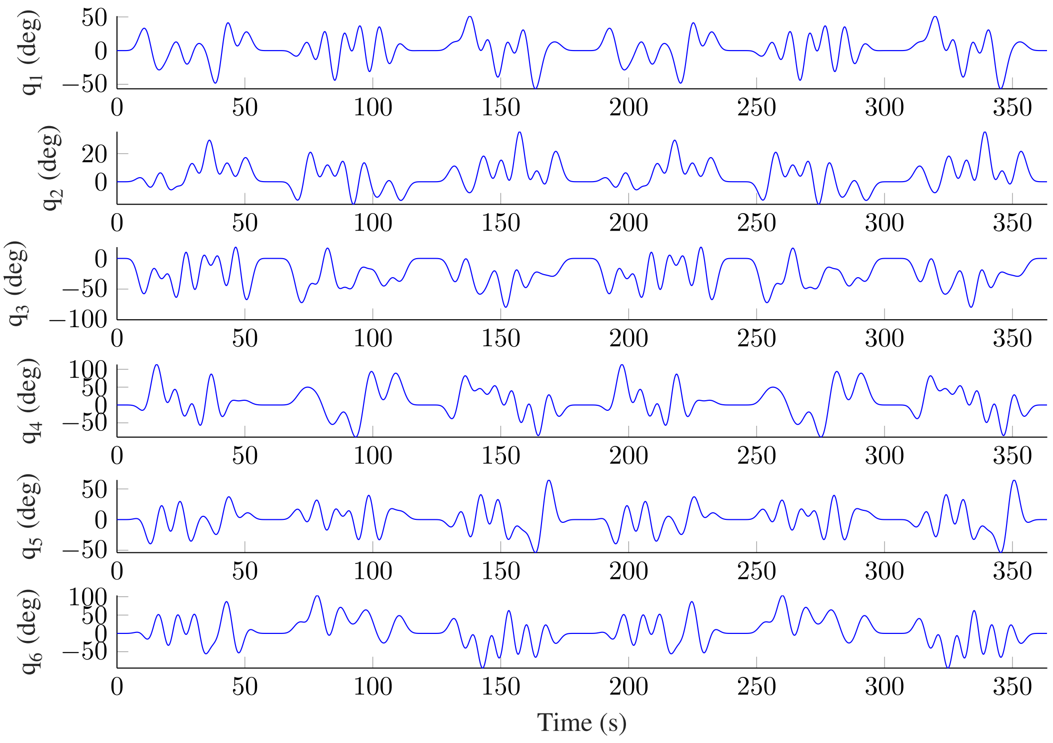

Figure 3: Measured joint positions in deg as test data. Each row corresponds to one joint.

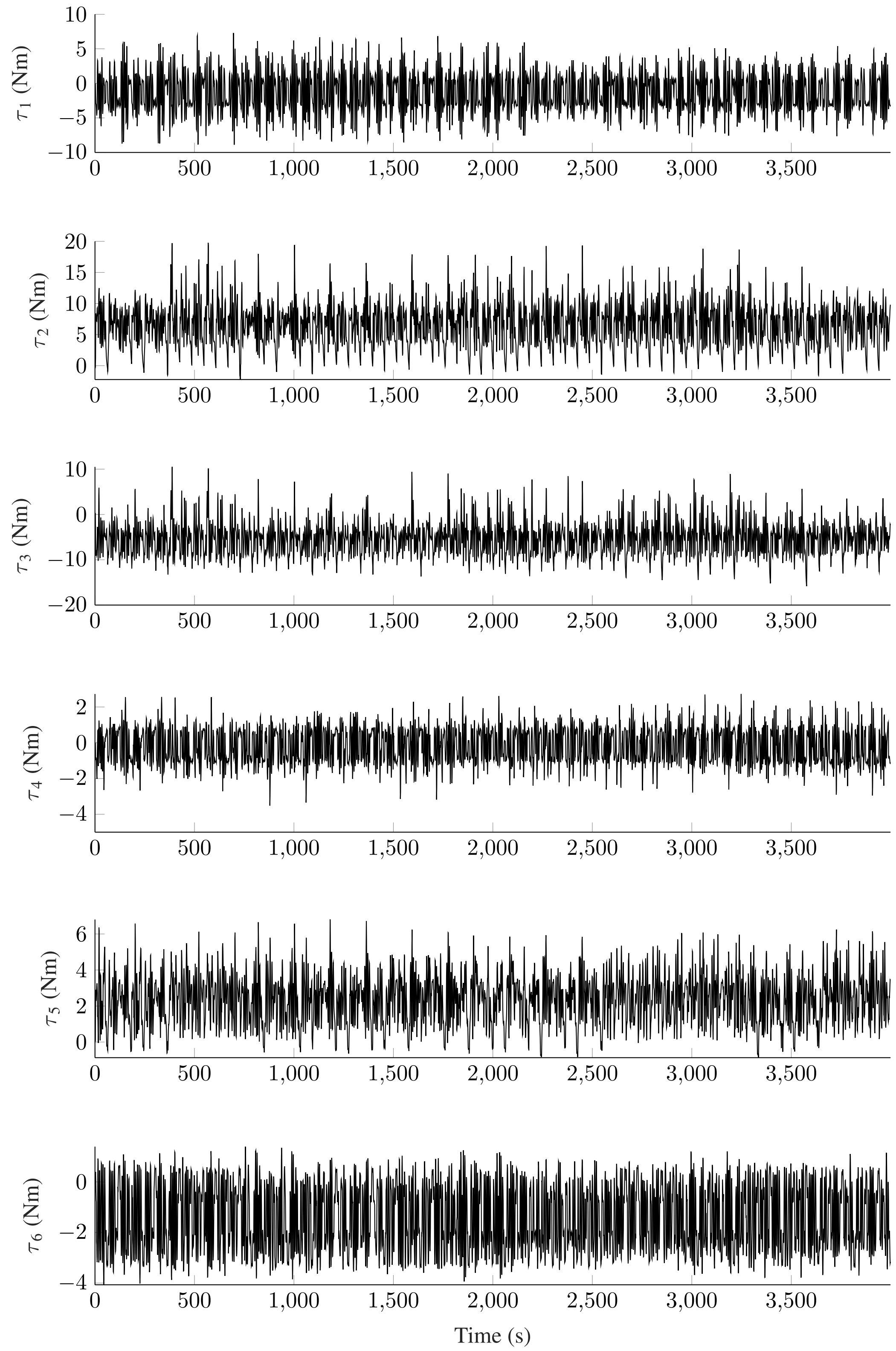

Figure 4: Measured motor torques in Nm as training data. Each row corresponds to one joint.

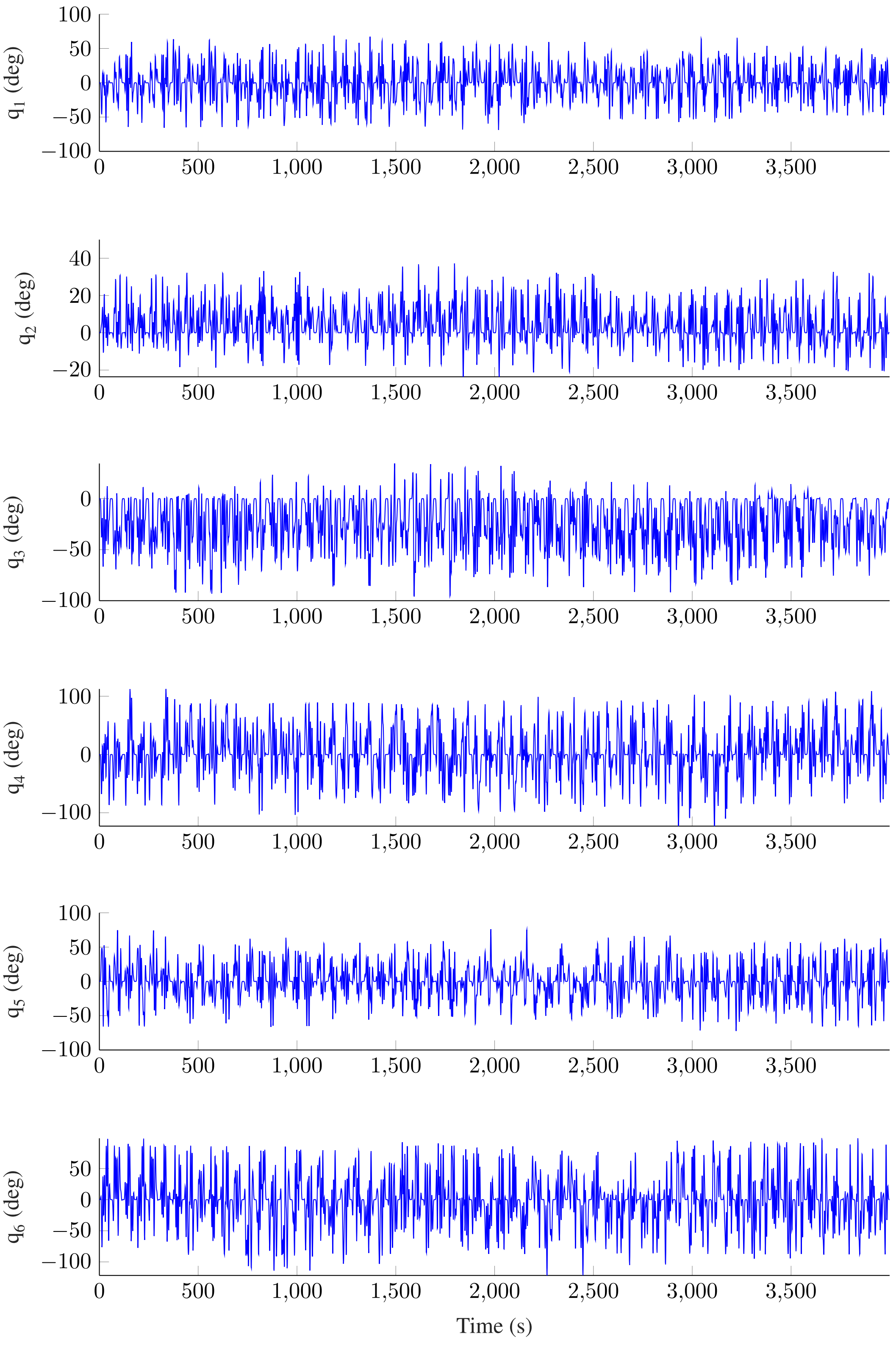

Figure 5: Measured joint positions in deg as training data. Each row corresponds to one joint.

The robot is not built for high-frequency applications. Although the robot feedback controllers react very fast (position control 1250 Hz, velocity control 2500 Hz and current control 20 kHz), high-frequency trajectories would be absorbed by gearbox effects, such as high gearbox ratios, considerable elasticity, and backlash. In other words, as the electrical input energy is limited, we can either generate high amplitudes (which is done) or high frequencies (which would be internally absorbed anyway). Due to these actuation limitations, we restricted the relevant joint position frequency spectrum to less than $0.3\,\mathrm{Hz}$ to enable a sufficient state-space coverage. As a result, the data can be low pass filtered with a cutoff frequency of $2\,\mathrm{Hz}$ (see listing 1). The frequency spectrum of the training input and training output data is presented in Fig. 6.

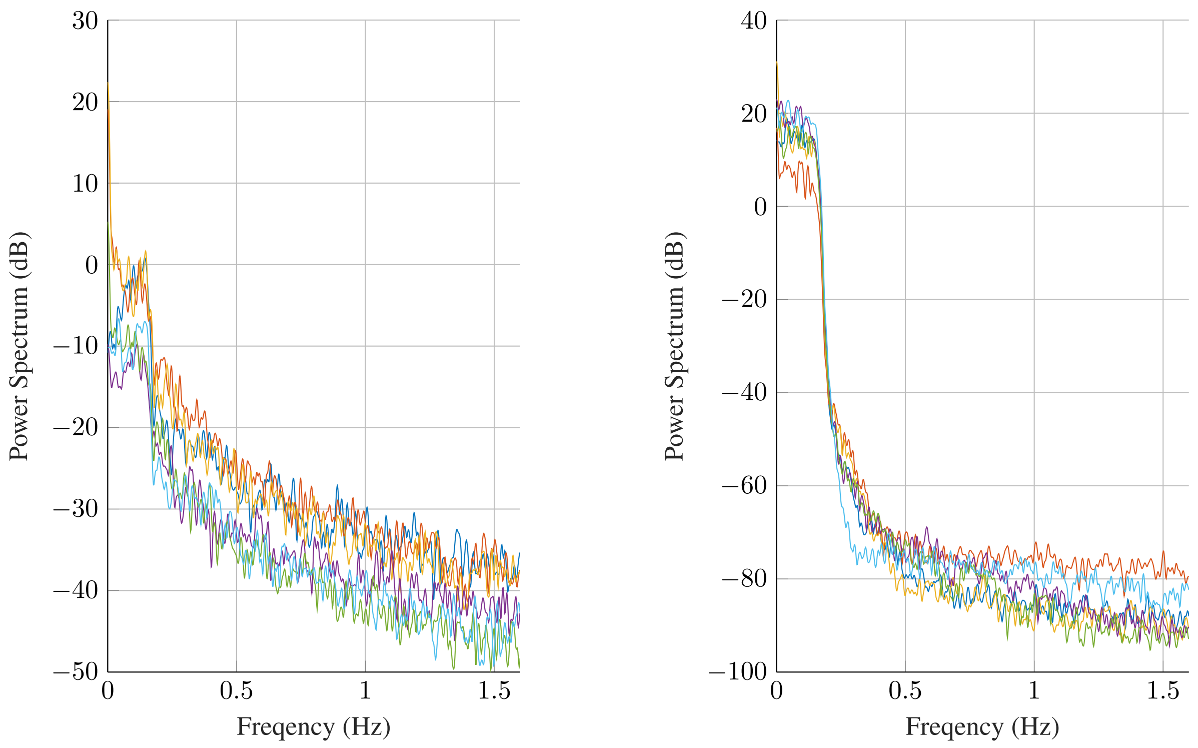

Figure 6: Frequency spectrum of the motor torque training data (left side) and the joint position training data (right side) for all joints. Black: Joint 1. Red: Joint 2. Purple: Joint 3. Green: Joint 4. Blue: Joint 5. Brown: Joint 6.

3 Forward And Inverse Model Identification

For robot simulation and model-predictive-control (MPC), a forward model of the robot is required which applies the motor torque τ as input and outputs the robot position $\mathbf{q}$. For robot feedforward controllers, such as computed torque control (CTC), an inverse model of the robot is required which takes the desired position $\mathbf{q_R}$ as input and outputs the feedforward torque $\tau_{\mathbf{FF}}$. Usually, the simulation task requires a multi-step-ahead prediction setting and the control task requires a one-step-ahead prediction. These experiments offer the possibility for both tasks. However, we created two separate datasets from the same measurements (see Tab. 2).

For the simulation task, the data can be applied as presented in Fig. 2, 3, 4 and 5 without modification. The corresponding nonlinear state-space robot model is given by

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1} \left(\mathbf{U}\tau_{\mathbf{M}} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) - \tau_{\mathbf{F}}(\dot{\mathbf{q}}) - \tau_{\mathbf{H}}(\mathbf{q}) \right) \tag{3}$$

as explained in the appendix A. Equation (3) shows the the acceleration $\ddot{\mathbf{q}}$ is expected to be a function of robot position $\mathbf{q}$, velocity $\dot{\mathbf{q}}$ and motor torque $\tau_{\mathbf{M}}$. Therefore, the mapping from motor torque as input to robot position as output, with optionally the velocity as an internal state, is well defined.

For the control task, some modifications are required. The inverse robot model is

$$\tau_{\mathbf{FF}} = \mathbf{U}^{-1} \left(\mathbf{M}(\mathbf{q_R})\ddot{\mathbf{q}}_{\mathbf{R}} + \mathbf{C}(\mathbf{q_R}, \dot{\mathbf{q}}_{\mathbf{R}})\dot{\mathbf{q}}_{\mathbf{R}} + \mathbf{g}(\dot{\mathbf{q}}_{\mathbf{R}}) + \tau_{\mathbf{F}}(\dot{\mathbf{q}}_{\mathbf{R}}) + \tau_{\mathbf{H}}(\dot{\mathbf{q}}_{\mathbf{R}}) \right) \tag{4}$$

which is a constant mapping, in the sense that the output $\tau_{\mathbf{FF}}$ does not rely on any previous motor torques.[2]

To fit the feedforward control as well as possible to the real system, it is trained on the time-shifted measured position rather than on the reference position and on the measured torques rather than the computed torques. A time-shift of $4\,\mathrm{ms}$ is required to compensate for the feedback controller computation time. Additional input data is provided for this challenging task. The input consists of the time-shifted position, velocity, and acceleration, and the outputs are the measured motor torques, as summarized in Tab. 2.

Table 2: Summary for the separate forward and inverse model identification datasets.

	Forward Model	Inverse Model
Input	Motor torque τ (Nm) 6 Channels	Joint Position $\mathbf{q}$ (deg) Joint Velocity $\dot{\mathbf{q}}$ $(\mathrm{deg/s})$ Joint Acceleration $\ddot{\mathbf{q}}$ $(\mathrm{deg/s^2})$ (All time-shifted) 18 Channels
Output	Joint Position $\mathbf{q}$ (deg) 6 Channels	Motor torque τ (Nm) 6 Channels
	Simulation-mode recommended	Prediction-mode recommended

In addition to the data prepared for the identification benchmark, the dataset includes raw measurements of every single experiment, as presented in Tab. 3. The raw data is unfiltered and given at $250\,\mathrm{Hz}$ in contrast to the filtered, $10\,\mathrm{Hz}$ data prepared for the benchmark. Please state clearly if any of the following data is accessed for the benchmark.

Table 3: Raw data for every single experiment. All data is given in high-frequency $250\,\mathrm{Hz}$

Raw Data
Joint Reference Position $\mathbf{q_R}$ (deg)
Joint Reference Velocity $\dot{\mathbf{q}}_\mathbf{R}$ $(\mathrm{deg/s})$
Joint Position $\mathbf{q}$ (deg) measured by motor resolver
Joint Position $\mathbf{q}$ (deg) measured by secondary encoder
Joint Velocity $\dot{\mathbf{q}}$ $(\mathrm{deg/s})$ measured by motor resolver
Joint Velocity $\dot{\mathbf{q}}$ $(\mathrm{deg/s})$ measured by secondary encoder
Absolute motor torque τ (Nm) $(\tau = \tau_{\mathbf{FF}} + \tau_{\mathbf{FB}})$
Model-based feedforward motor torque $\tau_{\mathbf{FF}}$ (Nm)
Feedback controller motor torque $\tau_{\mathbf{FB}}$ (Nm)

[2]Considering the real control architecture, the motor torque is naturally not completely discontinuous.

4 Metrics and Figures of Merit

4.1 Forward Model Metrics

Please report the model performance on the test dataset, similar to the baseline in Fig. 7, with the measured and estimated joint positions. Comment explicitly if prediction or simulation mode is chosen. Report all angles in deg and torques in Nm. Exemplary figures of merit are given by Fig. 7 and Fig. 8. As for performance metrics, apply the normalized-root-mean-squared-error (NRMSE) (5) and the R^2 norm (6) and average these metrics over all joints (9), (10). The number of joints is $N = 6$, the number of time steps is $K = 3636$ for the test data, $K = 39988$ for the training data. The standard deviation regarding measured torques for each joint n is $\sigma_{\tau,n}$ and regarding position is $\sigma_{q,n}$.

$$\mathbf{NRMSE}_n = \sqrt{\frac{1}{K\sigma_{q,n}^2}\sum_{k=1}^{K}(q_{meas,n,k} - q_{est,n,k})^2}, \tag{5}$$

$$\mathbf{R}_n^2 = 100\left(1 - \frac{\sum_{k=1}^{K}(q_{meas,n,k} - q_{est,n,k})^2}{\sum_{k=1}^{K}(q_{meas,n,k} - (\frac{1}{K}\sum_{j=1}^{K} q_{meas,n,j}))^2}\right) \tag{6}$$

4.2 Inverse Model Metrics

For the inverse model identification task, the metrics are similar, except motor torque is applied instead of the position as output.

$$\mathbf{NRMSE}_n = \sqrt{\frac{1}{K\sigma_{\tau,n}^2}\sum_{k=1}^{K}(\tau_{meas,n,k} - \tau_{est,n,k})^2}, \tag{7}$$

$$\mathbf{R}_n^2 = 100\left(1 - \frac{\sum_{k=1}^{K}(\tau_{meas,n,k} - \tau_{est,n,k})^2}{\sum_{k=1}^{K}(\tau_{meas,n,k} - (\frac{1}{K}\sum_{j=1}^{K} \tau_{meas,n,j}))^2}\right) \tag{8}$$

4.3 Averaged Metrics

In both cases, inverse and forward identification, report the over all joints averaged NRMSE and R^2 norm.

$$\textbf{average NRMSE} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{NRMSE}_n, \tag{9}$$

$$\textbf{average } \mathbf{R}^2 = \frac{1}{N}\sum_{n=1}^{N}\mathbf{R}_n^2 \tag{10}$$

5 Figures of Merit and Linear Identification Baseline

5.1 Forward Model Baseline

We apply a linear state-space model as a baseline like presented in listing 2 using the *MATLAB System Identification Toolbox* [10]. The data is normalized. Furthermore, the prior knowledge is applied that the robot obtains 12 states and set the initial state to zero. The performance of the test data is presented in Fig. 7. Performance metrics for each joint and on average are given in Tab. 4.

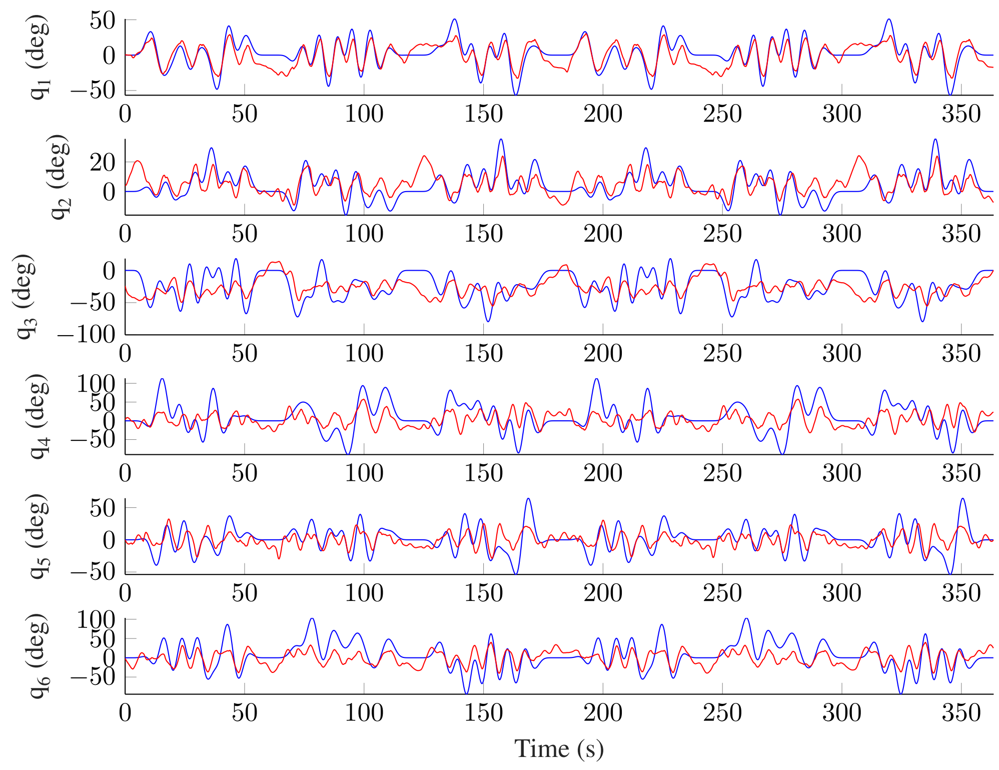

Figure 7: Performance of the linear model on the test data in simulation mode. Each row corresponds to one joint. Blue: Measured joint positions. Red: Estimated joint positions.

Table 4: Performance metrics of the linear model on the forward identification test data in simulation mode.

Joint	Training Data		Test Data	
	NRMSE	R^2	NRMSE	R^2
		%		%
q_1	0.635	59.733	0.638	59.236
q_2	0.717	48.603	0.829	31.293
q_3	0.726	47.323	0.876	23.179
q_4	0.866	25.083	0.894	20.098
q_5	0.865	25.099	0.869	24.534
q_6	0.838	29.852	0.845	28.513
All Joints	0.7745	39.2822	0.82517	31.1422

5.2 Inverse Model Baseline

As (4) is a constant mapping in a one-step-ahead prediction setting, the baseline can be formulated as a single, linear matrix $\mathbf{A} \in \mathbb{R}^{N \times 3N}$. The matrix coefficients are estimated using the *MATLAB Optimization Toolbox* [10] and the code is given in listing 3. Results on the test data are presented in Fig. 8 and metrics are given in Tab. 5.

The dominant effects for each joint significantly differ. Gravitational loads act primarily on joints 2 and 3. Hydraulic loads occur only on joint 2. Joints are equipped with significantly different motor types (see appendix C). The maximum torque input of joints 1, 2, and 3 amounts to triple of the joints 4, 5, and 6 (see Fig. 2). The motor power supply ranges from 5.5 kW to 16 kW. Simultaneously, because the load is considerably reduced, the amplitude of motion joints 4, 5, and 6 is a multiple of the joints 1, 2, and 3 (see Fig. 3).

$$\tau = \mathbf{A} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} \tag{11}$$

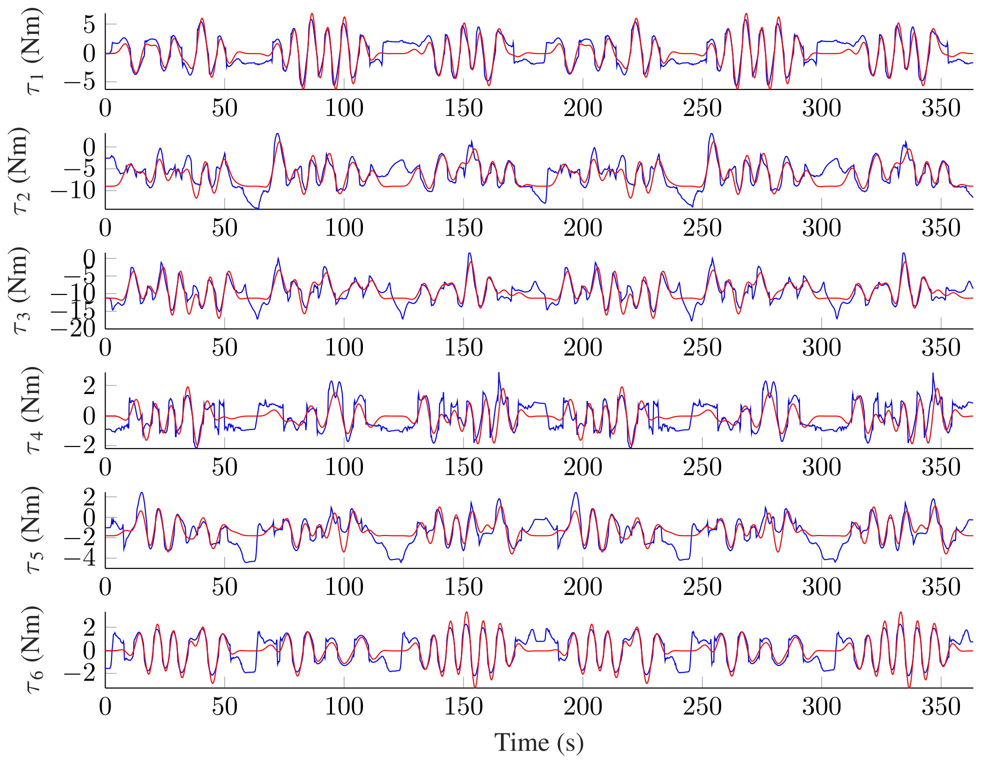

Figure 8: Performance for the inverse benchmark on the test data in prediction mode. Each row corresponds to one joint. Blue: Measured motor torques. Red: Estimated motor torques.

Table 5: Performance metrics of the linear model on the inverse identification test data in prediction mode.

Joint	Training Data		Test Data	
	NRMSE	R^2 %	NRMSE	R^2 %
τ_1	0.418	82.502	0.447	79.982
τ_2	0.595	64.615	0.681	53.607
τ_3	0.555	69.228	0.549	69.841
τ_4	0.673	54.72	0.705	50.253
τ_5	0.679	53.902	0.71	49.615
τ_6	0.551	69.635	0.559	68.723
All Joints	0.5785	65.767	0.6085	62.0035

5.3 Discussion of Standstill and Low-Velocity Phases

In the first version of this benchmark, we applied the *ssest()* function instead of the *n4sid()* function of the *MATLAB System Identification Toolbox* for the forward baseline. The results differ significantly[3]. For reference, see Fig. 9 for the first version.

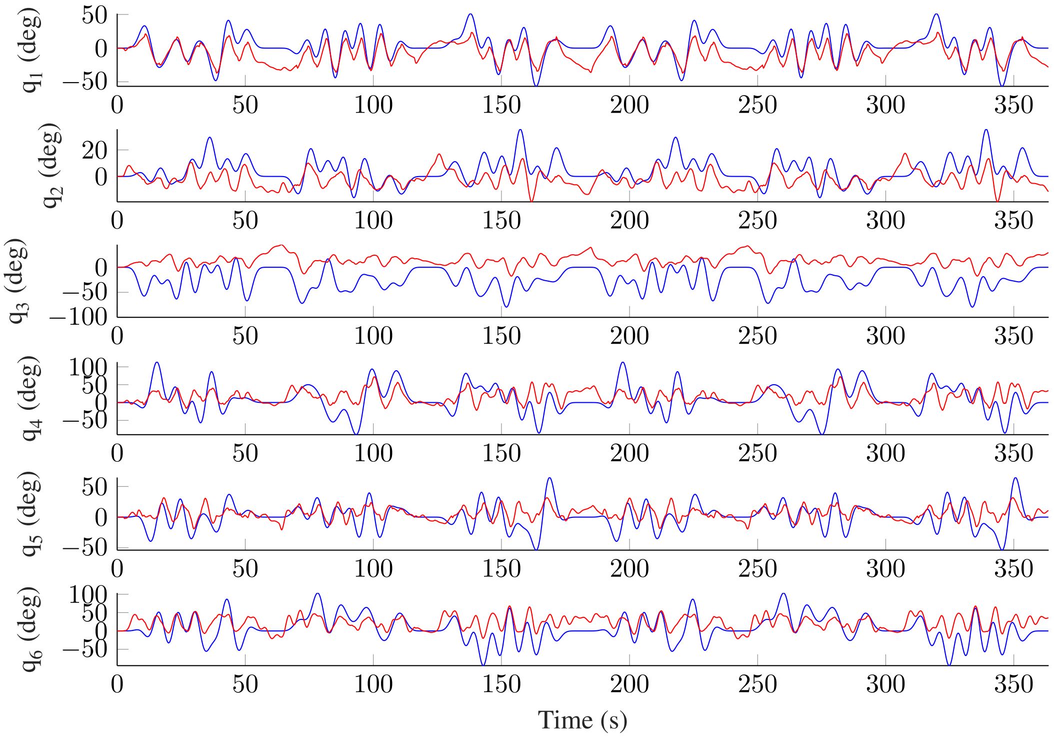

Figure 9: Performance of the linear model on the test data in simulation mode. Each row corresponds to one joint. Blue: Measured joint positions. Red: Estimated joint positions. First version of the baseline using the *MATLAB ssest()* function instead of the current version using the *MATLAB n4sid()* function. *This figure represents the legacy baseline for comparison. The current results are presented in Fig. 7.*

A major challenge of the benchmark are nonlinear effects which occur during standstill or low-velocity phases of the experiment. In these phases, the robot position output signal is almost constant, while the motor torque input changes significantly to overcome static friction. Even worse, different torque input signals lead to the same, almost constant position output. From an application and control point of view, these phases are not relevant, as almost no robot movement is present. Still, these phases are a significant challenge for black-box models. The low-velocity phases can be accurately identified using the measured velocity signal. As a rough estimate, the low-velocity phases occur in the first 5 seconds and last 5 seconds of each trajectory. Low-velocity phases occur approximately from $0-5$ seconds, $55-65$ seconds, $115-125$ seconds, and so forth.

[3]We thank Alberto Bemporad for the helpful advice to improve the results.

Appendix

A Industrial Robot Model

To focus this contribution, we will only briefly summarize the model. The well-known robot model

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \tau_{\mathbf{F}}(\dot{\mathbf{q}}) + \tau_{\mathbf{H}}(\mathbf{q}) = \mathbf{U}\tau_{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}), \tag{12}$$

consists of the pose-dependent inertia matrix $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{N\times N}$, the centrifugal and Coriolis matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{N\times N}$, the vector of gravitational torques $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{N}$, the nonlinear friction torque vector $\tau_{\mathbf{F}}(\dot{\mathbf{q}}) \in \mathbb{R}^{N}$, the torque vector resulting from the Hydraulic Weight Counterbalance (HWC) $\tau_{\mathbf{H}}(\mathbf{q}) \in \mathbb{R}^{N}$, the diagonal gearbox transmission matrix $\mathbf{U} \in \mathbb{R}^{N\times N}, \mathbf{U} = diag(u_1, u_2, \cdots, u_N)$ and the input torque transmitted from the electrical motors $\tau_{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{N}$. The model is based on generalized coordinates $\mathbf{q} \in \mathbb{R}^{N}$, where N stands for the number of degrees of freedom, which are 6 for the given demonstrator. The estimation of $\mathbf{M}(\mathbf{q}), \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{g}(\mathbf{q})$ is given in [6]. Details on the robot and control are given previous works [13]. A flatness-based feedforward control is not applied, rather a model-based feedforward control based on equation (12).

A.1 Controller Design

For high accuracy applications, the robot is on the three main joints equipped with link-side angular measurements (secondary encoders). The motor angle $\boldsymbol{\Theta}$ actuates the gearbox, and the gearbox output angle (link side) $\mathbf{q}$ is measured additionally. So gearbox deformations can be measured directly and can be compensated by the feedback position control. The velocity control, however, is applied on the motor side for an improved system behavior in all cases. For details, we refer to [13]. In this work, we set for simplicity $\mathbf{q} \approx \mathbf{U}\boldsymbol{\Theta}$. In the dataset, the link side sensor is employed whenever possible. This implies that gearbox deformation and backlash are included in the measurements for the first three joints. As the hand joints are not equipped with secondary encoders, the motor sensors are utilized.

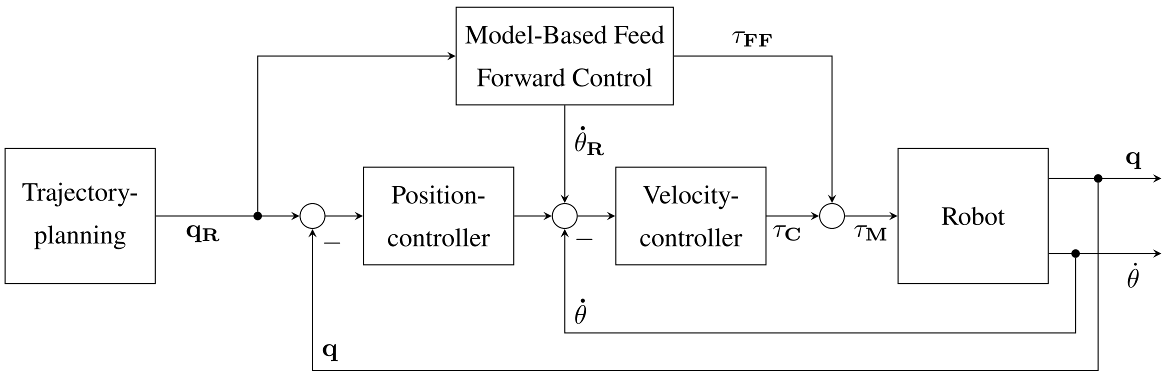

Figure 10: Controller design with feedforward and feedback control

The commanded motor torque is given by a PD-controller

$$\tau_{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) = \tau_{\mathbf{FF}} + \mathbf{P}\left(\mathbf{q}_{\mathbf{R}} - \mathbf{q}\right) + \mathbf{D}\left(\dot{\mathbf{q}}_{\mathbf{R}} - \dot{\mathbf{q}}\right), \tag{13}$$

with the feedforward torque $\tau_{\mathbf{FF}}$, the reference trajectory $\dot{\mathbf{q}}_{\mathbf{R}}$ and the proportional and derivative diagonal gain controller matrices $\mathbf{P}$ and $\mathbf{D}$ respectively.

A.2 Hydraulic Weight Counterbalance

We model the HWC as a static hydraulic torque $\tau_{H,S}$ which acts only on joint 2,

$$\tau_{\mathbf{H}}(\mathbf{q}) = [0,\ \tau_{H,S},\ 0,\ 0,\ 0,\ 0]^T\,. \tag{14}$$

The static hydraulic torque $\tau_{H,S}$ is generated by a hydraulic force $F_{H,S}$ and an orthogonal height $h(q_2)$. The height $h(q_2)$ can be derived from geometry, as depicted in Fig. 11, which results in

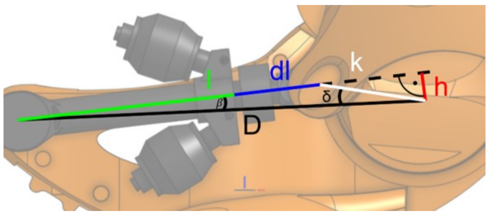

Figure 11: The geometry of the Hydraulic Weight Counterbalance of *KUKA KR300 R2500 ultra SE*.

$$dl(q_2) = \sqrt{D^2 + k^2 - 2Dk\cos(q_2)} - l\,, \tag{15}$$

$$h(q_2) = \frac{Dk\sin(q_2)}{l + dl(q_2)}\,, \tag{16}$$

with the distance D from joint center to HWC attachment, HWC attachment radius k, base length $l = D - k$ and extension length $dl(q_2)$. The hydraulic pressure p is generated by and equal to the nitrogen gas pressure in the blatter accumulators. We apply the ideal gas law, include a compressibility factor Z, and get

$$p(q_2) = \frac{m_N Z R_S T_N}{V_{max} - dl(q_2)A}\,, \tag{17}$$

$$\tau_{H,S} = h(q_2)\tau_{S,F} = h(q_2)\cdot p(q_2)A, \tag{18}$$

with the specific gas constant R_S, the nitrogen mass m_N, the nitrogen temperature T_N, the maximum bladder volume V_{max} and the effective piston area A. All geometric parameters are given in the CAD model. We assume the temperature and the compressibility factor to be constant. The nitrogen mass and the bladder volume are identified using pressure measurements in several static joint positions.

A.3 Friction Model

We apply a nonlinear friction model based on [7, 8]. The friction model exploits an asymmetrical, viscous, Coulomb, and degressive friction term

$$\tau_{\mathbf{F}}(\dot{\mathbf{q}}) = \mathbf{f_{asym}} + \mathbf{f_v}\dot{\mathbf{q}} + \mathbf{f_c}\tanh(\mathbf{s_f}\,\dot{\mathbf{q}}) + \mathbf{f_a}\tanh(\mathbf{f_b}\,\dot{\mathbf{q}}),$$

with the zero drift error of friction torque $\mathbf{f_{asym}} \in \mathbb{R}^N$, the viscous friction coefficient $\mathbf{f_v} \in \mathbb{R}^N$, the Coulomb friction coefficient $\mathbf{f_c} \in \mathbb{R}^N$, the sign smoothness factor $\mathbf{s_f} \in \mathbb{R}^N$ and degressive friction coefficients $\mathbf{f_a} \in \mathbb{R}^N$ and $\mathbf{f_b} \in \mathbb{R}^N$. The degressive friction torque routes back to [8] and encompasses a saturation of the friction torque in the high velocity range. Modeling this degressive friction behavior matches various measurements. Furthermore, the Coulomb friction is approximated by $sign(\cdot) \approx tanh(\mathbf{s_f}\,(\cdot))$ for a differentiable friction in the trajectory and identification optimization algorithm. In contrast to $\mathbf{f_b}$, the smoothness factor $\mathbf{s_f}$ is not a degree of freedom in the model and we ensure $\mathbf{s_f} \gg \mathbf{f_b}$.

B Acknowledgments

This work is only possible due to the outstanding work of many collaborators in the last 4 years, developing the complete robot and cell control, both hardware configuration and software. Furthermore, we thank Ethan Deng and Dongsheng Deng for providing the paper template. https://github.com/ElegantLaTeX

Table 6: Detailed contributions of all collaborators.

Name	Contribution
MAIN COLLABORATORS	
Jonas Weigand	project lead, integration of all collaborators, supervision of all students, software development and validation, writing this paper
Julian Götz	several student thesis, major contribution to the hardware configuration and software setup
Jonas Ulmen	robot modelling, design of experiments
Martin Ruskowski	project supervision and guidance, funding
SCIENTIFIC EMPLOYEES	
Achim Wagner	scientific consulting
Andreas Wagner	electronic configuration and cell development
Aleksander 'Sasha' Sidorenko	Matlab and B&R connection
Alexandre Janot	robot model and identification
Magnus Volkmann	electronic configuration and cell development
Nigora Gafur	development of the robot controller
Simon Lamoth	B&R development, code validation
William Motsch	mySQL database integration
TECHNICAL EMPLOYEES	
Bendof 'Rouven' Chazkelewitz	electronic calbeling and cell construction
Rüdiger Ruppert	electronic calbeling and cell construction
STUDENTS	
Björn Dietrichs	controller development: robustness
Christopher Lippert	sensor integration
Florian Wendling	controller development: stability
Gajanan Kanagalingam	kinematic model, trajectory generation
Leon Hensel	controller development: stability
Niklas Hagen	overall system validation
Patrick Kremser	OPC-UA development, sensor integration
Sebastian Harttig	CAD model, kinematic and dynamic model
Sergej Gertje	controller development: MPC
Stephan Belz	controller development: dynamic model, flatness-based feedforward control
Xiaohai Wang	controller development: nonlinear optimization
INDUSTRIAL SUPPORT	
Joachim Spangenberg	technical support B&R Automation
Marco Sprenger	technical support B&R Automation
Sven Varelmann	technical support B&R Automation

C Technical Details

C.1 Dimensions and Limits

The robot workspace is presented in Fig. 12. All lengths are given in mm and angles are given in deg. Please note that Fig. 12 applies a different zero-pose notation than the measurements. The angular difference is given by

$$\mathbf{q} = [0, -90, 90, 0, 0, 0] \quad \mathrm{deg}. \tag{19}$$

The position and velocity limits are presented in Tab. 7, using the measurement position convention. As the velocity limits are all symmetrical, we only present the upper limits, as $\dot{q}_{lb,n} = -\dot{q}_{ub,n}$ holds.

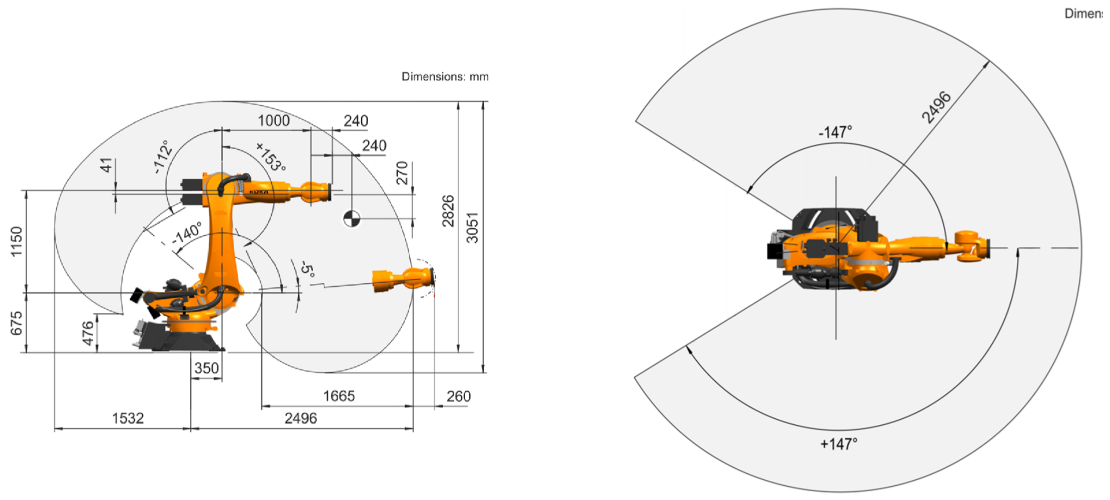

Figure 12: *KUKA KR300 R2500 ultra SE* dimensions in mm and absolute angular limits in deg. Different zero angle definitions. Copyright by [9].

Table 7: Joint position and velocity limits. Velocity limits are symmetrical. Absolute limits are caused by electrical and mechanical constraints [9]. In the experiments, we applied a reduced subset.

	Absolute Limits			Applied Limits		
	q_{lb}	q_{ub}	$\dot{q}_{ub}$	q_{lb}	q_{ub}	$\dot{q}_{ub}$
	deg	deg	deg/s	deg	deg	deg/s
Joint 1	-147	147	84.6	-90	90	63.4
Joint 2	-50	85	82.3	-30	40	61.7
Joint 3	-202	63	79.3	-110	40	59.5
Joint 4	-350	350	122	-180	180	91.5
Joint 5	-122.5	122.5	113	-90	90	84.8
Joint 6	-350	350	175	-180	180	131.3

C.2 Hardware Layout

Figure 13 presents the basic hardware layout of the controller. Starting at the top right, an industrial real-time computer performs the main robot tasks: the process state machine, the measurement handling, the diagnose system, the web-based human-machine-interface, the hand-held operating device, parts of the control loop, as well as error-detection and machine supervision. The real-time computer hosts an OPC-UA server, which communicates bidirectionally with a Matlab OPC-UA client (light blue line). The Matlab code performs non-real-time tasks such as trajectory generation, robot model identification, or machine learning applications. Matlab is also the interface for measurement post-processing and file management. The industrial computer also hosts a second non-real-time operating system, Windows 10, on which the programming and diagnosis can be done. Other than for diagnosis, the Windows-based operating system does not interfere with the run-time tasks of the robot. Cycle time of main tasks is given in Tab. 8. The main components presented in Fig. 13 are listed in Tab. 9.

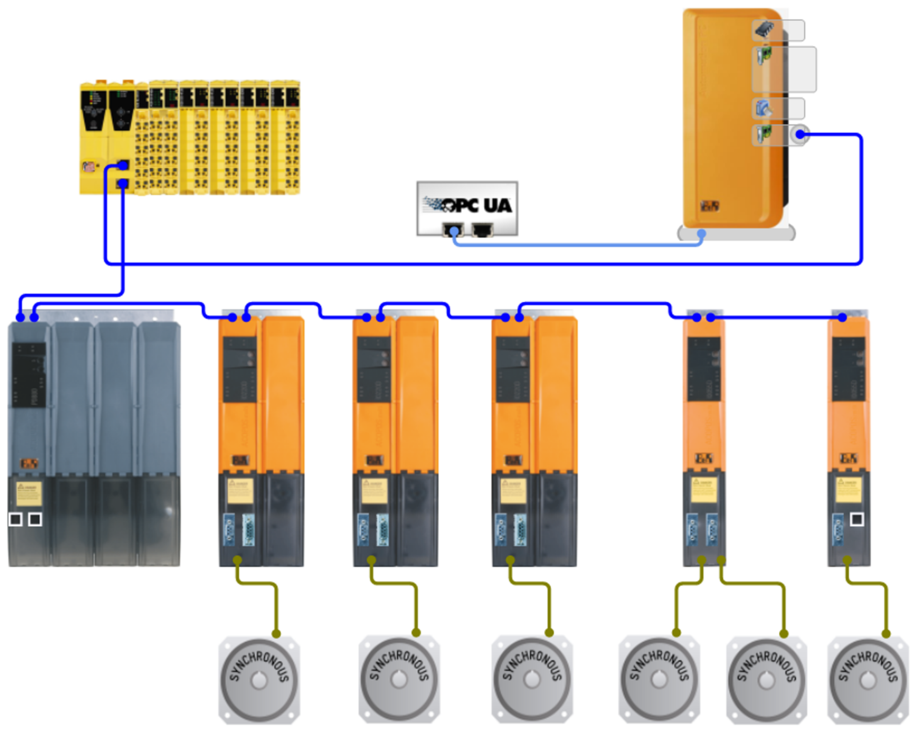

Figure 13: Hardware configuration. The corresponding picture is given in Fig. 1, right side. Copyright by [5].

The dark blue lines in Fig. 13 represent the industrial bus protocol, called POWERLINK by B&R Automation, over which all run-time information is communicated. The green lines on the bottom represent the motor signals, such as motor temperature sensor, motor brake control, and motor position and velocity signal. Not displayed in Fig. 13 are three components: The signals from the secondary encoders (link side position sensors) to the motor power inverters, the 3-phase power connection from the inverters to the motors, and the main power cables connecting inverters and power supply.

Following the industrial bus from the real-time computer (dark blue line, top), a safety programmable logic controller (PLC) is connected. This PLC is safety-certified, independent of all other hard- and software, and can shut down the main power supply in all cases. It explicitly handles hardware emergency stops and supervises the robot cell. It is the only component that cannot be reconfigured without advanced authorization.

The bus signal is sent from the safety PLC to the main power supply (left side, grey), which transforms the laboratory power supply corresponding to the motor inverter requirements. Parallel to the main power, the industrial bus signal is sent to all motor inverters, which locally compute the position, velocity, and current feedback control, record measurements, and read the secondary encoders. The code of the motor inverters, as well as the information which is sent over the bus system, can be adopted.

All motor inverters are designed for double-axis. For joints 1, 2, and 3, only one high-power output is used. The other side of the double-axis is required for additional sensor integration and applied for software computation directly on the inverter. Joints 4, 5 and 6 operate on double-axis, with Joint 4 and 5 on a single device. The other side of the double-axis device for joint 6 is unused.

Table 8: (Deterministic) computation time of main software components [4].

Task	Deterministic	Time
OPC-UA server	no	between $4\,\mathrm{ms}$ and $10\,\mathrm{s}$, depending on task requirements and data volume. Bidirectional.
State machine	yes	$4\,\mathrm{ms}$
Diagnose system	yes	$4\,\mathrm{ms}$
Measurement handling	yes	$4\,\mathrm{ms}$, can be set to $0.8\,\mathrm{ms}$
Position feedback control	yes	$0.8\,\mathrm{ms}$
Velocity feedback control	yes	$0.2\,\mathrm{ms}$
Motor current feedback control	yes	$0.05\,\mathrm{ms}$

Table 9: Main components of the robot control [3, 4]

Component	Description
1x Real-Time Computer	Automation PC 910 series with Intel Core i7 6820EQ, QM170 chipset, 128 GB CFast memory, 8 GB DDR4 RAM, POWERLINK managing node, 2 MB SRAM battery buffered
1x Power Supply	ACOPOSmulti series with 3x 400 VAC input voltage, 750 V_{DC} link voltage, 60 kW continuous power
3x Inverter Joint 1, 2, 3	ACOPOSmulti3 series dual-axis module with 750 V_{DC} bus voltage, 55 A peak current, 22 A continuous current, 16 kW continuous power, 24 V_{DC} and 2.1 A for holding brake
2x Inverter Joint 4, 5, 6	ACOPOSmulti3 series dual-axis module with 750 V_{DC} bus voltage, 18.9 A peak current, 7.6 A continuous current, 5.5 kW continuous power, 24 V_{DC} and 1.1 A for holding brake

C.3 Sensor Specification

Table 10 shows the sensor types and corresponding resolutions for all joints. We refer to the software resolution as the smallest incremental change that can be detected in the software. The real sensor uncertainty is greater: First of all, the sensor head, the measurement ring, and the resolver depend on manufacturing tolerances. Considering the manufacturer's specifications, however, these uncertainties are not significant. Second, for both sensors, we define the resolution on the link side. This implies for the motor-side mounted resolvers, that the effective resolution depends on the gearbox factors. So this estimated effective resolution is corrupted by gearbox deformation and backlash, which sums up to an approximate error range of $0.01\,\mathrm{deg}$ to $0.05\,\mathrm{deg}$ (depending on the trajectory and payload, see [13, 14]). All sensors are low pass filtered in first order, with a cutoff frequency of $1000\,\mathrm{Hz}$ for both the secondary encoders and for the motor resolves.

Table 10: Secondary encoders (SE) [1] and motor resolvers (RE) [11]. All resolutions are given for the link side. Software resolution is the smallest incremental change that can be detected. Effective resolution accounts for additional uncertainties, see main text.

Sensor	Type	Software Resolution deg	Effective Resolution deg
SE Joint 1	WMR-301-0507-01-S03 (Ring)	$4.33 \cdot 10^{-6}$	$5 \cdot 10^{-6}$
	WMR-301.12-0507-0.20-9-S01 (Head)		
SE Joint 2	WMR-301-0413-01-S03 (Ring)	$5.32 \cdot 10^{-6}$	$6 \cdot 10^{-6}$
	WMR-301.12-0413-0.10-9-S01 (Head)		
SE Joint 3	WMR-301-0339-01-S03 (Ring), WMR-301.12-0339-0.125-9-S01 (Head)	$6.48 \cdot 10^{-6}$	$7 \cdot 10^{-6}$
	WMR-301.12-0339-0.125-9-S01 (Head)		
RE Joint 1	1FK7-101-5AY71-1SY3-Z	$85.5 \cdot 10^{-6}$	$5 \cdot 10^{-2}$
RE Joint 2	1FK7-103-5AY71-1SY3-Z	$82.2 \cdot 10^{-6}$	$5 \cdot 10^{-2}$
RE Joint 3	1FK7-103-5AY71-1SY3-Z	$87.1 \cdot 10^{-6}$	$5 \cdot 10^{-2}$
RE Joint 4	1FK7-063-5AF71-1SY3-Z	$99.4 \cdot 10^{-6}$	$5 \cdot 10^{-2}$
RE Joint 5	1FK7-063-5AF71-1SY3-Z	$91.7 \cdot 10^{-6}$	$5 \cdot 10^{-2}$
RE Joint 6	1FK7-063-5AF71-1SY3-Z	$240 \cdot 10^{-6}$	$5 \cdot 10^{-2}$

D MATLAB Model Code

Listing 2: Linear Forward Simulation Model Baseline.

```
% normalize the data
u_std = std(u_train,0,2); y_std = std(y_train,0,2);
u_mean = mean(u_train,2); y_mean = mean(y_train,2);

for k_ax = 1:size(y_train,1)
    y_train(k_ax,:) = (y_train(k_ax,:) - y_mean(k_ax)) / y_std(k_ax);
    u_train(k_ax,:) = (u_train(k_ax,:) - u_mean(k_ax)) / u_std(k_ax);
    u_test(k_ax,:) = (u_test(k_ax,:) - u_mean(k_ax)) / u_std(k_ax);
end

% define id data
dt                   = 0.1;
id_data              = iddata(y_train', u_train', dt);

% identify model
opt_n4sid            = n4sidOptions();
opt_n4sid.Focus      = 'simulation';
n_states             = 12;
robot_model          = n4sid(id_data, n_states, opt_n4sid);

% simulate on test data
y_sim                = lsim(robot_model, u_test, time_true)';

% denormalize
for k_ax = 1:size(y_sim,1)
    y_sim(k_ax,:) = y_sim(k_ax,:) * y_std(k_ax) + y_mean(k_ax);
end
```

Listing 3: Linear Inverse Prediction Model Baseline.

```
% normalize the data
u_std = std(u_train,0,2); y_std = std(y_train,0,2);
u_mean = mean(u_train,2); y_mean = mean(y_train,2);

for k_ax = 1:size(y_train,1)
    y_train(k_ax,:) = (y_train(k_ax,:) - y_mean(k_ax)) / y_std(k_ax);
end
for k_ax = 1:size(u_train,1)
    u_train(k_ax,:) = (u_train(k_ax,:) - u_mean(k_ax)) / u_std(k_ax);
    u_test(k_ax,:) = (u_test(k_ax,:) - u_mean(k_ax)) / u_std(k_ax);
end

% define id data
dt                                = 0.1;
id_data                           = iddata(y_train', u_train', dt);

% identify model
opt_fminunc                       = optimoptions('fminunc');
opt_fminunc.MaxFunctionEvaluations  = 1e4;
opt_fminunc.MaxIterations         = 1e3;

```

```
rng(1234);
n_outputs                       = size(y_train, 1);
n_inputs                        = size(u_train, 1);
inverse_lin_model               = zeros(n_outputs, n_inputs);
n_opt                           = numel(inverse_lin_model);
x0                              = randn(n_opt, 1);

[x_sol, ~] = fminunc(@cost_fun_inverse_lin_model, x0, opt_fminunc);
inverse_lin_model(:) = x_sol;

% simulate on test data
y_sim       = inverse_lin_model * u_test;

% denormalize
for k_ax = 1:size(y_sim,1)
    y_sim(k_ax,:) = y_sim(k_ax,:) * y_std(k_ax) + y_mean(k_ax);
end

% nested cost function
    function cost = cost_fun_inverse_lin_model(x)
        % compute the sum of squred errors
        inverse_lin_model(:)    = x;
        error                   = (y_train - inverse_lin_model * u_train);
        cost                    = sum(error.^2, 'all');
    end
```

References

[1] AMO Automatisierung Messtechnik Optik, St. Peter am Hart, Austria (2021). Modulare Winkelmessgeräte nach dem induktiven AMOSINő Messprinzip (German), https://www.amo-gmbh.com/.

[2] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36.

[3] BR Automation, Bad Homburg, Germany (2021a). ACOPOSmulti user's manual (English, German), https://www.br-automation.com/en/products/motion-control/acoposmulti/.

[4] BR Automation, Bad Homburg, Germany (2021b). APC910 user's manual (English, German, Russian), https://www.br-automation.com/en/products/industrial-pcs/automation-pc-910/.

[5] BR Automation, Bad Homburg, Germany (2021c). Automation Studio Software (English, German), https://www.br-automation.com/en/products/software/automation-software/automation-studio/.

[6] Corke, P. (2017). *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised, https://peter-corke.com/toolboxes/robotics-toolbox/*, volume 118. Springer.

[7] Ding, L., Li, X., Li, Q., and Chao, Y. (2018). Nonlinear friction and dynamical identification for a robot manipulator with improved cuckoo search algorithm. *Journal of Robotics*, 2018:1–10.

[8] Grotjahn, M., Daemi, M., and Heimann, B. (2001). Friction and rigid body identification of robot dynamics. *International journal of solids and structures*, 38(10-13):1889–1902.

[9] KUKA Aktiengesellschaft, Augsburg, Germany (2021). KR QUANTEC KR300 R2500 ultra SE Operating Manual, https://www.kuka.com/en-de.

[10] MATLAB (2020). *Version 9.8.0.1380330 (R2020a) Update 2, https://de.mathworks.com/products/matlab.html*. The MathWorks Inc., Natick, Massachusetts.

[11] SIEMENS AG, München, Germany (2006). Projektierungshandbuch Ausgabe 12/2006, Synchronmotoren 1FK7, SINAMICS S120 (German), https://support.industry.siemens.com/cs/document/28683106/sinamics-s120-synchronmotoren-1fk7-?dti=0&lc=de-WW.

[12] Tika, A., Ulmen, J., and Bajcinca, N. (2020). Dynamic Parameter Estimation Utilizing Optimized Trajectories. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7300–7307. IEEE.

[13] Weigand, J., Gafur, N., and Ruskowski, M. (2020). Flatness Based Control of an Industrial Robot Joint Using Secondary Encoders. *Robotics and Computer-Integrated Manufacturing*.

[14] Weigand, J., Volkmann, M., and Ruskowski, M. (2019). Neural Adaptive Control of a Robot Joint Using Secondary Encoders. In *International Conference on Robotics in Alpe-Adria Danube Region*, pages 153–161. Springer.