Identification of Friction Models for MPC-based Control of a PowerCube Serial Robot

Jörg Fehr*, Arnim Kargl* and Hannes Eschmann*
*Institute of Engineering and Computational Mechanics, University of Stuttgart, Germany

Summary. For model-based control, an accurate and in its complexity suitable representation of the real system is a decisive prerequisite for high and robust control quality. In a structured step-by-step procedure, a model predictive control (MPC) scheme for a Schunk PowerCube robot is derived. Neweul-M^2 provides the necessary nonlinear model in symbolical and numerical form. To handle the heavy online computational burden involved with the derived nonlinear model, a linear time-varying MPC scheme is developed based on linearizing the nonlinear system concerning the desired trajectory and the a priori known corresponding feed-forward controller. To improve the identification of the nonlinear friction models of the joints, a nonlinear regression method and the Sparse Identification of Nonlinear Dynamics (SINDy) are compared with each other concerning robustness, online adaptivity, and necessary preprocessing of the input data. Everything is implemented on a slim, low-cost control system with a standard laptop PC.

Introduction

The use of robots to increase work performance or human-machine interaction for rehabilitation are topics of high topicality. For the performance of robotic manipulators, modeling, control, and sensing play an essential role. Modeling plays an important role in the development process as well as for controlling the robot, e. g., for model-based control concepts like model predictive control (MPC). Model-based control schemes offer many advantages in comparison to individual joint control. The benefits are: (i) operation as a centralized control scheme – the highly nonlinear behavior of such a system is considered; (ii) intuitive parameter tuning – an elaborate and time-consuming parameter tuning necessary for PI or PID-based individual joint control is unnecessary; (iii) constraint considerations - actuator and state limitations are already considered in the control design.

In [1] a model predictive controller (MPC) was derived and implemented for a modular 6-axis Schunk PowerCube robot, see Fig. 4. The general MPC algorithm, see, e. g., [2, 3] is the following: (i) obtain a measurement/estimate of the state; (ii) obtain an optimal input sequence by solving online an open-loop optimal control problem (OCP) over a finite horizon subject to system dynamics and constraints; (iii) apply the first input of the optimal input sequence to the plant/robot; (iv) continue with the first step. Challenges for the application are the highly nonlinear system dynamics as well as the limited computation capacity. Everything should run on a slim low-cost setup, i. e., a standard laptop PC.

These challenges are met by a linear time-variant two-step approach, see Fig. 1. In a first step the nonlinear system dynamics are approximated with a linear time-varying model [4] around the desired trajectory –using an inverse dynamic approach / computed torque approach [5]. In the second step sophisticated techniques for linear MPC are exploited, i. e., the open-source quadratic programming solver qpOASES [6] is used.

The proposed LTV MPC control of the robotic system is able to perform complex trajectories, i. e., motion reversal and zero crossing [1] [1]. Nevertheless, the noticeable difference between calculated feed forward and MPC output, see Fig. 2 along the trajectory implies model inaccuracies. These findings imply that the friction model of the robot's joints still has some weaknesses which limits the performance of the overall control system. Therefore, more effort is needed to identify the friction properties of the joints. Prior to this work, a classical system identification method, i. e., data pre-processing in combination with nonlinear regression, was used to identify the friction properties of a single disassembled module.

Therefore, aim of this work is to evaluate how the large amount of data from different sources, i. e., Artifical Intelligence will facilitate the complex and challenging modeling and system identification of an assembled system – e. g., can the large amount of data be used to identify the effects in the assembled state and/or difference between individual products due to production tolerances.

More specifically, we want to answer the question of whether the Sparse Identification of Nonlinear Dynamics (SINDy) method [7, 8] improve the friction identification. In contrast to black-box AI methods, like Neural Networks or Gaussian Processes – which try to approximate the data by adjusting some weights of a topological system, the SINDy method tries to identify the governing equations from data. It approximates an unknown function $\boldsymbol{f}$ with a library $\boldsymbol{\Theta}(\boldsymbol{X})$ of r potential (nonlinear) terms. The SINDy approach is a parametric approach that, compared to NNs, works without massive amounts of data. The approach allows for on-the-fly model adaptation due to its low computational complexity.

Let us recap the overall goal: "Improve the model-based control performance of a robotic manipulator by improving friction identification using the SINDy approach to identify the governing equation from data." The main features are: (i) the robustness of the approach and (ii) only the friction characteristic is identified – other well-identified or known terms of the system are incorporated as prior knowledge.

In the next section, we describe the model and the methodology in more detail: (i) the robot which serves as an example; (ii) the process control framework; (iii) the derivation of the equation of motion of the rigid multibody system with Neweul-M^2 [9] (iv) the existing friction model, (v) the SINDy concept and (vi) the recording of measurement data. In Section Friction Identification the results of the approach are presented. Finally, in the Conclusion the overall control performance of the system with the improved friction model is discussed.

[1] The complete motion of the manipulator can be seen in the deposited video `https://www.itm.uni-stuttgart.de/en/research/vision-based-control-of-a-powercube-robot/`.

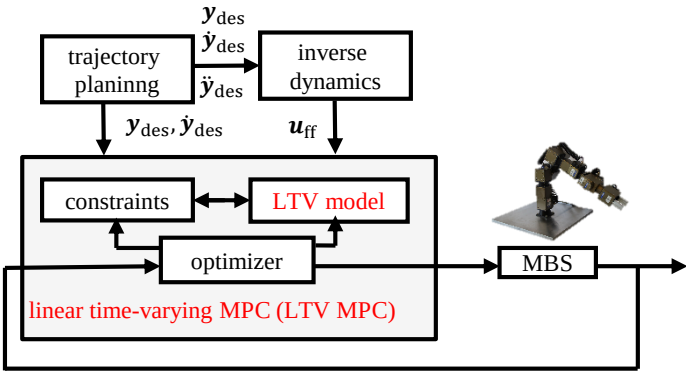

Figure 1: Topologic structure of the two-step control loop with the feed-forward part $\boldsymbol{u}_{\text{ff}}$ and the LTV-MPC part.

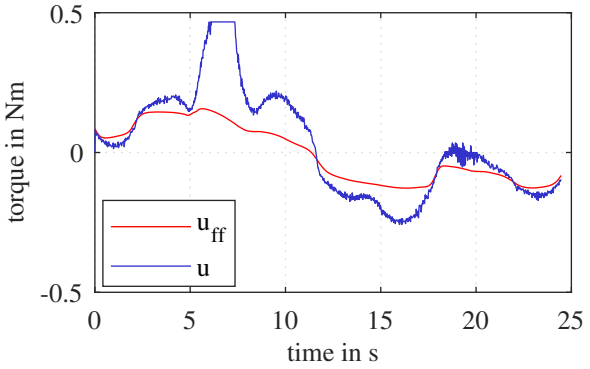

Figure 2: Feed forward torque of the old robot model compared to the MPC output torque applied at joint β during a trajectory with motion reversal.

Model and Methodology

The performance of the SINDy approach is evaluated on a modular 6-axis *Schunk PowerCube Robot*, see Fig. 4. For the considered modular robot a *Process Control Framework* in Matlab Simulink and a *Simulation Model* in Neweul-M^2 is available. We focus on the friction identification of the first three joints of the robot, which are each a rotary unit PR 90 with a Harmonic Drive gear to see the influence of assembly and production differences of the same product. Therefore measurement data is gathered during experiments followed by offline identification of the friction characteristics with nonlinear regression and the proposed *SINDy Framework*, relying on *Sparse Linear Regression*. In the following, we explain the single building blocks for model, *Experimental Measurements* and mathematical methodology.

Schunk PowerCube Robot

The properties of the various links are described in Tab. 1. For the design of the robot a "divide et impera" approach is used. For the first three links the same rotary module (PR 90) is used. The fourth link, is the smaller version(PR 70) within the model series. The fifth and six links of the robot consist of a pan-tilt unit in combination with an anthropomorphic gripper with a spherical wrist, is constructed from the single modules. The robotic manipulator is designed in such a way that the arm has an analytically calculable inverse kinematics.
Each rotary module, consists of a brushless DC-motor which drives a Harmonic Drive gear, which provides torque at each degree of freedom (DOF) based on the defined motor current. The control and power electronics are integrated, and an incremental encoder is used for position and speed evaluation. Furthermore, a brake is incorporated in case of shutdown or power failure.

Process Control Framework

The process control framework is depicted in Fig. 3 , consists of a Microsoft Windows laptop PC with Matlab R2014b and Simulink, including the additional toolboxes Real-Time Windows Target and Simulink Coder. The laptop PC is equipped with an Intel Core i5-3210M CPU (2x2.5 GHz, 8 GB DDR3). The communication between the Simulink model and the hardware is based on an USB-CAN bus interface, which is embedded into Simulink via S-Functions and a communication library from Schunk. A sampling rate of 50 Hz, corresponding to a sampling interval of 20 ms, is used. An external power supply with constant voltage of 24 V in combination with the integrated control and power electronics of modules ensures the necessary torques at the links.
Real-Time Windows Target [10] realizes a real-time engine for Simulink models on a Microsoft Windows PC and offers the capability to run hardware-in-the-loop simulations in real-time. It is a lean solution for rapid prototyping and provides an environment in which a single computer can be used as a host and target computer. Consequently, real-time simulations are executed in Simulink without an external target machine.

Simulation

The robotic manipulator is modeled as rigid multibody system with joint friction. In a first step Neweul-M^2 [9] aids calculating a rigid body model *without friction* with the advantage of generating equations of motion in symbolic and numerical form. A natural choice for generalized coordinates $\boldsymbol{y}$ are the joint coordinates $\boldsymbol{y} = [\beta, \gamma]^{\mathrm{T}}$. Therefore the resulting equation of motion (*without friction*) in minimal form can be denoted as

$$\boldsymbol{M}(\boldsymbol{y})\ddot{\boldsymbol{y}} + \boldsymbol{k}(\boldsymbol{y}, \dot{\boldsymbol{y}}) = \tilde{\boldsymbol{q}}(\boldsymbol{y}, \dot{\boldsymbol{y}}) + \boldsymbol{B}\boldsymbol{u} \tag{1}$$

with the positive definite mass matrix $\boldsymbol{M}$, the vector of generalized Coriolis, centrifugal and gyroscopic forces $\boldsymbol{k}$ and the vector of generalized forces without friction $\tilde{\boldsymbol{q}}$. The system input is the vector $\boldsymbol{u} = [T_2, T_3]^{\mathrm{T}}$ which consists of the applied

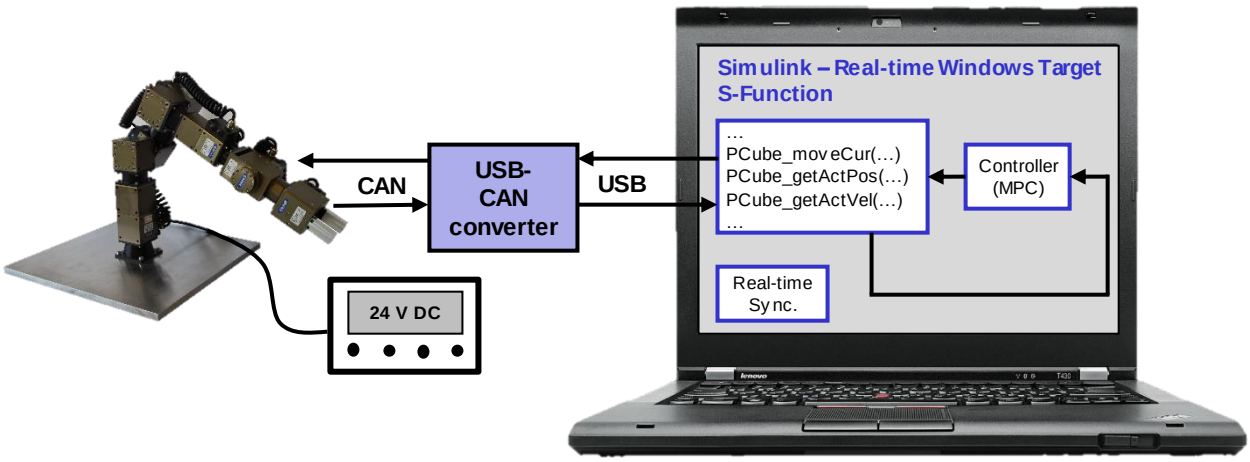

Figure 3: Slim process control framework of the robotic manipulator

motor torque at each joint. Those motor torques are scaled accordingly with gear ratios included in $\boldsymbol{B}$. A linear model $\boldsymbol{u} = K_{\mathrm{M}}\boldsymbol{i}$ describing the relation between motor torque and motor current is assumed. The motor constant K_{M} has been identified manually at the hardware. System parameters, such as masses, inertias, link lengths and gear ratios are taken from CAD data and datasheets, supplied by the manufacturers.
In a second step friction torques $\boldsymbol{\tau}(\dot{\boldsymbol{y}})$ are included into the model without friction, resulting in the vector of generalized forces

$$\boldsymbol{q}(\boldsymbol{y}, \dot{\boldsymbol{y}}) = \tilde{\boldsymbol{q}}(\boldsymbol{y}, \dot{\boldsymbol{y}}) - \boldsymbol{\tau}(\dot{\boldsymbol{y}}). \tag{2}$$

For a more intuitive understanding $\boldsymbol{\tau}$ is subtracted from $\tilde{\boldsymbol{q}}$, since positive friction works against the current movement. The equation of motion *with friction* can be gained by substitution of $\tilde{\boldsymbol{q}}$ with $\boldsymbol{q}$ in eq. (1). Joint friction models of the form

$$\tau_i(\dot{y}_i) = a_1\dot{y}_i + a_2\tanh(a_3\dot{y}_i) + a_4\exp(-a_5|\dot{y}_i|)\tanh(3a_3\dot{y}_i), \tag{3}$$

derived in [11] are used in a first application which results in the stated mismatch between feed forward and MPC output. This form of friction model with parameter vector $\boldsymbol{a}$ includes the share of viscous friction corresponding to $a_1\dot{y}_i$, the share of Coulomb friction corresponding to $a_2\tanh(a_3\dot{y}_i)$ as well as the superelevation of the Stribeck curve which is characterized by $a_4\exp(-a_5|\dot{y}_i|)\tanh(3a_3\dot{y}_i)$. The smooth *tanh*-function replaces the *sgn*-function to avoid discontinuities within the model.
Focus lies on identifying more accurate friction models for both joints with either the SINDy method or general nonlinear regression. Therefore data of $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, $\ddot{\boldsymbol{y}}$ and $\boldsymbol{u}$ has to be collected during experiments or derived after running experiments respectively, such that

$$\boldsymbol{\tau}(\dot{\boldsymbol{y}}) = \tilde{\boldsymbol{q}}(\boldsymbol{y}, \dot{\boldsymbol{y}}) + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{M}(\boldsymbol{y})\ddot{\boldsymbol{y}} - \boldsymbol{k}(\boldsymbol{y}, \dot{\boldsymbol{y}}) \tag{4}$$

can be calculated. Stated methods can then be applied. The resulting values for $\boldsymbol{\tau}$ obviously are dependent on all stated variables $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, $\ddot{\boldsymbol{y}}$ and $\boldsymbol{u}$. The notation $\boldsymbol{\tau}(\dot{\boldsymbol{y}})$ is chosen due to the assumption that friction effects only depend on joint velocities.

SINDy Concept

The concept of **S**parse **I**dentification of **N**onlinear **Dy**namics (SINDy) founds in the field of applied mathematics. It represents a modern method to gain nonlinear models based on experiment data, a long known challenge in system theory. The main concept behind SINDy can be described as reducing the nonlinear fit to a collection of function candidates to a (sparse) linear regression, which can then be effectively solved with state of the art algorithms, providing robust and efficient solutions.
The SINDy setup consists of a standard representation of a nonlinear system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{5}$$

with state vector $\boldsymbol{x} \in \mathbb{R}^n$, possible input $\boldsymbol{u} \in \mathbb{R}^q$ and the unknown vector field $\boldsymbol{f}$. Starting out simple, assume the state $\boldsymbol{x}$, and its time derivative $\dot{\boldsymbol{x}}$ and the system input $\boldsymbol{u}$ to be known for m unique time instances. The data can then be rearranged into three matrices $\boldsymbol{X}$, $\dot{\boldsymbol{X}}$ and $\boldsymbol{U}$ as follows

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}}(t_1) \\ \boldsymbol{x}^{\mathrm{T}}(t_2) \\ \vdots \\ \boldsymbol{x}^{\mathrm{T}}(t_m) \end{bmatrix} \in \mathbb{R}^{m\times n}, \quad \dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\boldsymbol{x}}^{\mathrm{T}}(t_1) \\ \dot{\boldsymbol{x}}^{\mathrm{T}}(t_2) \\ \vdots \\ \dot{\boldsymbol{x}}^{\mathrm{T}}(t_m) \end{bmatrix} \in \mathbb{R}^{m\times n}, \quad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}^{\mathrm{T}}(t_1) \\ \boldsymbol{u}^{\mathrm{T}}(t_2) \\ \vdots \\ \boldsymbol{u}^{\mathrm{T}}(t_m) \end{bmatrix} \in \mathbb{R}^{m\times q}. \tag{6}$$

Goal is to approximate the vector field $\boldsymbol{f}$ by a library of r candidate functions $\Theta(\boldsymbol{X}, \boldsymbol{U}) \in \mathbb{R}^{m \times r}$ which are weighed by coefficients $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_r] \in \mathbb{R}^{n \times r}$ as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \approx \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{\Xi}, \tag{7}$$

where the columns of the library of candidate functions can contain all kinds of imaginable nonlinear terms. Those can be polynomials in $\boldsymbol{x}$, $\boldsymbol{u}$ or combinations of both or other nonlinear terms like trigonometric functions. Filling this library with appropriate terms requires intuition on how the solution could look and is considered one critical point concerning the success of the SINDy method.
The solution to the linear regression problem introduced in eq. (7) may be calculated with aid of various solvers, e. g., a standard least squares solver

$$\boldsymbol{\xi}_k = \arg\min_{\boldsymbol{w}} \left\| \dot{\boldsymbol{X}}_k - \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{w} \right\|_2^2. \tag{8}$$

Here $\boldsymbol{\xi}_k$ denotes the k-th column of $\boldsymbol{\Xi}$ while $\dot{\boldsymbol{X}}_k$ denotes the k-th column of $\dot{\boldsymbol{X}}$.
Doing so does result in a reasonable model but at the same time leads to very complex and detailed models with many active function terms. Key feature of the SINDy concept is solving the linear regression problem eq. (7) promoting sparsity within the solution vectors $\boldsymbol{\xi}_k$, which corresponds to the fact that most systems of interest can be described by only a few active terms in $\boldsymbol{f}$. Therefore solution techniques for sparse linear regression are applied to eq. (7). A schematic overview and example for a system with two states is given with Fig. 5.
The overall concept appears to be very flexible regarding to all kinds of dynamical systems. There exists a wide range of extensions which , e. g., allow SINDy to perform in discrete time or even to identify PDEs [7, 12].

The SINDy concept, in general, brings some advantages and some disadvantages compared to other machine learning concepts, e. g., an artificial neural network (NN). Such a NN may be used to describe the unknown vector field $\boldsymbol{f}$, which also gives good results. One general disadvantage of a NN is the training process, which requires large amounts of measureed data to obtain an suitable approximation for $\boldsymbol{f}$. SINDy on the other hand can work with one single short experimental trajectory as in Fig. 2. The SINDy method results in a set of nonlinear differential equations which may even be physically interpretable where a NN just results in some non-interpretable matrix layers. In addition to that, the SINDy results, being differential equations, can be evaluated in their whole domain and possibly even be extrapolated. This is a major advantage over neural networks, which often provide unsatisfying approximation results outside of the domain of the training data. Training data of course, has to cover all important system characteristics for both approaches. Otherwise, some system aspects and properties will be left out in the identified system model.
A crucial drawback of the SINDy concept lies in the function library $\boldsymbol{\Theta}$ and in the fact that only linear combinations of these functions describe the resulting dynamics $\boldsymbol{f}$ of the system. One critical disadvantage of the SINDy concept lies in its function library $\boldsymbol{\Theta}$ and that only linear combinations of those functions can be described in the resulting $\boldsymbol{f}$. Looking at eq. (3), one cannot directly identify a nested term like $a_2 \tanh(a_3 \dot{y}_i)$ with SINDy. Having a look back at eq. (3) one cannot directly identify a term like $a_2 \tanh(a_3 \dot{y}_i)$ with SINDy. The problem is that the parameter a_3 is embedded inside a nonlinear term. We will later avoid this by including multiple terms $\tanh(a_3 \dot{y}_i)$ with different fix parameters a_3 into the library $\boldsymbol{\Theta}$.
The main advantages and problems of the SINDy concept have been briefly touched upon here. In [8], however, a much more detailed comparison of SINDy with artificial neural networks when used in model predictive control can be found.

Sparse Linear Regression

Two algorithms out of the wide variety of sparse linear regression methods were tested and could be applied to the setup. One of them is the so called Least Absolute Shrinkage and Selection Operator (LASSO), which is well known from the

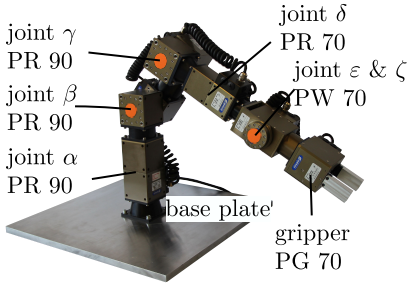

Figure 4: Experimental setup and configuration of the six degrees of freedom Schunk modular manipulator. The control is based on Real-Time Windows Target and Simulink Coder.

Table 1: Technical properties of the Schunk modules.

module name	nom. torque/force	max. velocity
PR 90 (rotary) joint α & β & γ	44.8 Nm	$\omega_{\max} = 25$ rpm
PR 70 (rotary) joint δ	10.0 Nm	$\omega_{\max} = 25$ rpm
PW 70 (pan-tilt) joint ε & ζ	12.0 Nm & 2.0 Nm	$\omega_{\max} = 25$ rpm
PG 70 (gripper)	200 N	$v_{\max} = 82$ mm/s

field of statistics and often used in machine learning applications. The LASSO can be described as ℓ_1 regularized version of the standard least squares linear regression setup and can be written as

$$\boldsymbol{\xi}_k = \arg\min_{\boldsymbol{w}} \left\| \dot{\boldsymbol{X}}_k - \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})\boldsymbol{w} \right\|_2^2 + \lambda \left\| \boldsymbol{w} \right\|_1. \tag{9}$$

The parameter $\lambda \geq 0$ adjusts the influence of the penalty term $\|\boldsymbol{w}\|_1$ and therefore determines how sparse the solution vector $\boldsymbol{\xi}_k$ turns out. With $\lambda = 0$ the solution is equal to the standard least squares problem's solution stated in eq. (8).
Another sparsity promoting linear regression approach is given by the Sequential Thresholded Least Squares algorithm (STLS) introduced in [7]. The STLS algorithm is an iterative procedure based on standard least squares solutions and can be described as follows.

- An initial solution $\boldsymbol{\xi}_k^{(0)}$ is calculated as least squares solution with the complete library matrix $\boldsymbol{\Theta}^{(0)} = \boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{U})$;
- In each iteration the regression setup reduces by
  - setting entries in $\boldsymbol{\xi}_k^{(r)}$ with absolute value less than a threshold parameter λ to zero;
  - deleting the corresponding columns in $\boldsymbol{\Theta}^{(r)}$, which would then be multiplied by zero, which leads to $\boldsymbol{\Theta}^{(r+1)}$;

  Solving the reduced least squares problem with $\boldsymbol{\Theta}^{(r+1)}$ leads to a new solution vector $\boldsymbol{\xi}_k^{(r+1)}$ which includes the remaining entries of $\boldsymbol{\xi}_k^{(r)}$.
- Iteration ends if no more entries in $\boldsymbol{\xi}_k^{(r)}$ fulfill the threshold condition.

Again, the parameter $\lambda > 0$ determines the sparsity of the solution. One iteration step of the STLS algorithm is illustrated in Fig. 6.
The critical point with both algorithms is choosing an appropriate value for the hyper parameter λ. In practice models for a broad range of parameters λ are calculated which allows finding a good compromise between sparsity of the resulting model and model error along a pareto front. The LASSO combined with k-fold cross validation is able to find this compromise by its own. With the STLS approach as it is described above one has to choose a model by hand. More sparse solutions on the one hand allow for some form of physical interpretation of the result but on the other hand may show a greater model error compared to other more flexible nonlinear regression techniques.

The whole SINDy setup was implemented in Matlab [13]. The STLS algorithm was implemented manually while a LASSO implementation already exists within the Statistics and Machine Learning Toolbox.

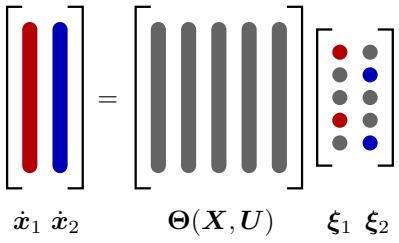

Figure 5: Application of the SINDy concept with sparse linear regression to a system with two states. Grey entries in $\boldsymbol{\xi}_k$ mark entries which are zero.

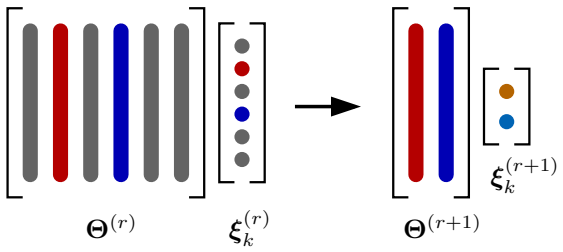

Figure 6: Illustration of the STLS algorithm. Grey entries in $\boldsymbol{\xi}_k^{(r)}$ are set to zero. Corresponding columns in $\boldsymbol{\Theta}^{(r)}$ are removed. $\boldsymbol{\xi}_k^{(r+1)}$ is the new least squares solutions with slightly different entries.

Experimental Measurements

Before taking measurements, a suitable class of trajectories has to be defined used for friction identification. We chose sine-shaped trajectories with variable frequency within this study since they can be constructed relatively easily. Being at least two times continuous differentiable sine trajectories brings smooth acceleration and deceleration, which is crucial for not exceeding joint limitations. Furthermore, differentiation and integration can be done analytically. The sine part of the trajectory can be described as

$$y(t) = \hat{a}\ \sin(\omega(t)t) \tag{10}$$

with a constant amplitude $\hat{a}$. Polynomial acceleration and deceleration phases around the sine trajectory part are needed since the robot starts and ends in a static pose where velocities and accelerations must be zero. Part of such a trajectory can be seen in Fig. 7.
The figure also points out a difficulty with measurement data. The robot's joint positions are measured by absolute angle encoders within each joint, which results in a non-smooth velocity measurement with coarse resolution. Therefore a

suitable differentiation method is needed to calculate joint velocities and accelerations from joint position data. In [8] the use of Total Variation Regularization Differentiation (TVDiff) is recommended. The TVDiff algorithm is introduced in [14]. The paper and Matlab implementations of the algorithm for one- and two-dimensional data, can be found on the author's web page. The idea of TVDiff originates in the Tikhonov regularization where the energy of a signal is minimized according to an energy functional without influencing the signal in a way a low pass filter would do. Variational methods are quite popular in the field of imaging science for efficiently denoising images. TVDiff results for trajectory measurement data are presented in Fig. 7. Figure. 8 demonstrates the advantages of TVDiff over finite differences with a therefore synthesized signal with added white noise. With higher frequencies a slight low pass effect is visible but the results are sufficient for the application with SINDy.

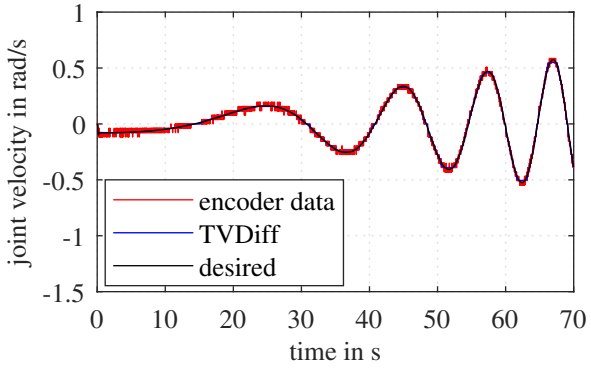

Figure 7: Measured encoder velocity compared with the filtered velocity from position measurements via TVDiff and the desired velocity.

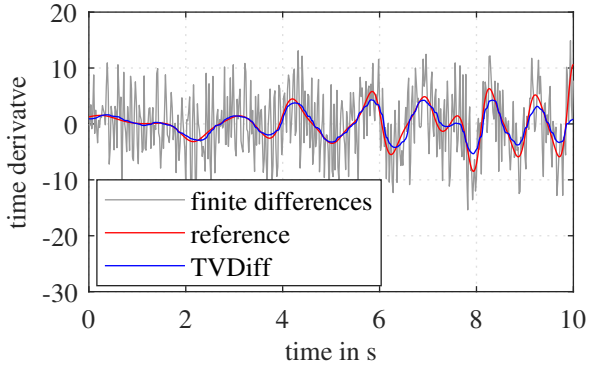

Figure 8: Synthesized example to demonstrate TVDiff advantages over finite differences with comparison to the analytically calculated reference signal.

Friction Identification

With previously described methods almost all infrastructure needed to identify friction models is present. Friction characteristics for robot joints B and C which correspond to the joint coordinates β and γ are displayed in Fig. 9 and Fig. 10. The resulting friction characteristics of both joints, if moved individually, will be dependent on the joint angle. These effects arise from model inaccuracies within the rigid multibody dynamics, e. g., incorrect inertia values or not modeling cables etc. The influence of gravity at non-zero joint angles makes the resulting friction characteristics dependent on the joint angle. This is against the assumption of friction only depending on joint velocity. Therefore for joint B the gained data is preprocessed by selecting data points with $|\beta| < 0.1$ where the influence of gravity is negligible. The filtered data is plotted in Fig. 12. Since many data points remain unused, a different strategy is chosen for joint C. Since it is the upper body of two joints, the influence of gravity can be easily compensated by moving both joints in opposite directions, keeping the upper part of the robot pointing straight upwards. Figure 11 illustrates half of a period of the particular periodic trajectory. Identification with the gravity compensated trajectory leads to the friction characteristics for joint C shown in Fig.10. As stated earlier the SINDy results are compared to a model calculated by nonlinear regression. Eq. 3 therefore functions as template for the nonlinear regression with its parameters a_1 to a_5. The nonlinear regression algorithm used is the `nlinfit` Matlab function which is included in the Statistics and Machine Learning Toolbox as well.

The application of the SINDy concept still requires us to specify a library of function terms. Over time and within many

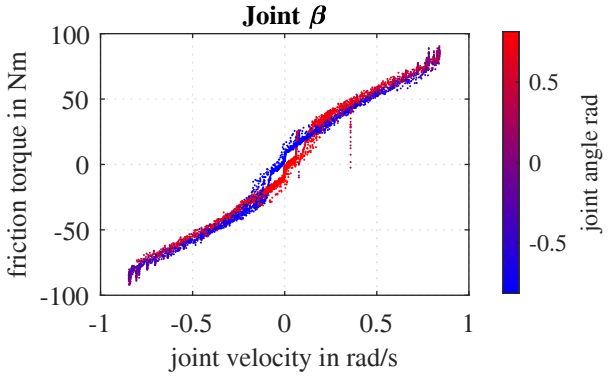

Figure 9: Friction characteristic for joint B measured with a sine trajectory. Only joint B was moved for these measurements.

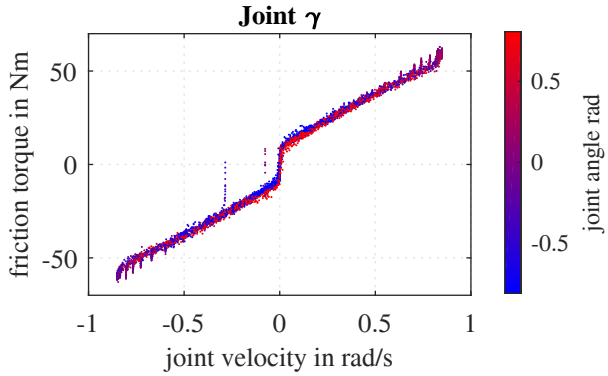

Figure 10: Friction characteristic for joint C measured with a sine trajectory. Joint B and joint C were moved equally in opposite directions to compensate gravity.

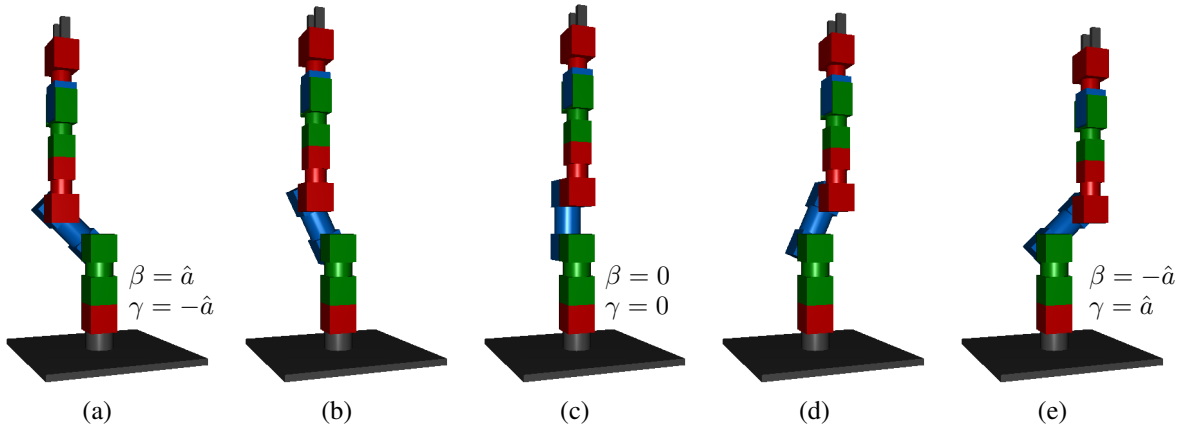

Figure 11: Periodic trajectory for friction identification in joint C. Shown is half a period starting at the left turning point (a), going through the robot's zero position (c) and ending at the right turning point (e). Frames (b) and (d) display intermediate points.

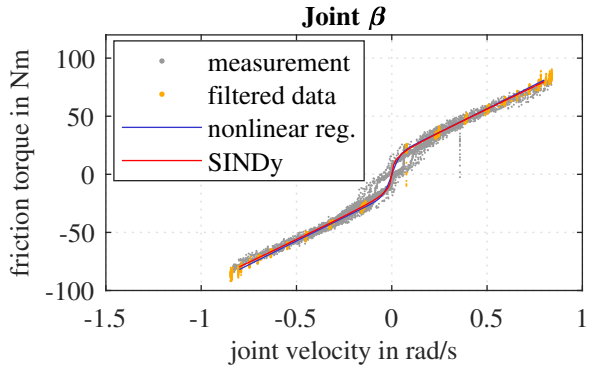

Figure 12: Fitted friction models for joint B after preprocessing measurement data.

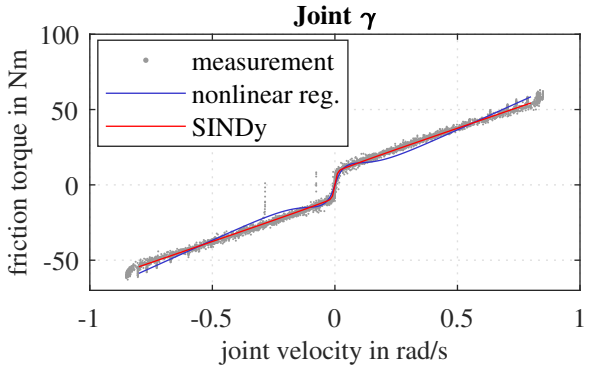

Figure 13: Fitted friction models for joint C. No preprocessing is needed.

experiments the library

$$\boldsymbol{\Theta}(\dot{\boldsymbol{y}}) = \begin{bmatrix} \mathbf{1} & \dot{\boldsymbol{y}} & \operatorname{sgn}(\dot{\boldsymbol{y}}) & \tanh(5\dot{\boldsymbol{y}}) & \tanh(10\dot{\boldsymbol{y}}) & \tanh(20\dot{\boldsymbol{y}}) & \tanh(100\dot{\boldsymbol{y}}) \end{bmatrix} \tag{11}$$

lead to promising results. The library contains functional terms for describing friction characteristics and at the same time, leaves enough options for the sparse linear regression to cancel out function terms. Promoting sparsity without leaving enough options for the algorithm would not lead to the expected results. A finer sampling of the terms $\tanh(a\dot{\boldsymbol{y}})$ was tested but results in high correlations between these functions, ultimately leading to difficulties in sparse linear regression. The library does not contain terms describing the superelevation of the Stribeck curve. This will only be a minor disadvantage since such a superelevation is not noticeable within the measurement data.

Applying SINDy and nonlinear regression to the measurement data together with the earlier discussed preprocessing leads to the the friction models shown in Fig. 12 and Fig. 13. Both regression concepts deliver quite similar results. SINDy outperforms nonlinear regression in the case of joint C, having the ability to choose freely from its function library while the nonlinear regression always has to fit to its function template. While nonlinear regression for the data of joint B would not work properly without preprocessing, the SINDy method would still lead to reasonable results. Although the preprocessed data gives a better friction model and shows that SINDy can even work with a percentage of the available data, making it robust to the amount of available measurement data.

An additional friction identification was performed for link A (corresponding to α) for better comparison of the joint friction characteristics and models. Therefore a linear quadratic regulator was implemented, since there doesn't exist an MPC scheme for the rigid body model with $\boldsymbol{y} = \alpha$. The resulting friction models for link A are displayed in Fig. 14. Again, nonlinear regression as well as the SINDy method perform well, although the measurement data is spread quite heavily. Preprocessing the data is not necessary for either method.

Comparing the newly identified joint friction models to the previously used model from [11] points out two major aspects. At first, friction models differ from joint to joint, although being the same kind of link module (PR90). Secondly, whether the joint friction is identified for an isolated joint or at an assembled robot makes a big difference. Therefore the two newly identified joint friction models differ heavily from the model identified in [11]. While in [11] single isolated joint modules were researched, the additional weight of our robot above joints B and C amplifies friction at the joints' axles. Fig. 15 shows the comparison of the different models. Compared are the SINDy results identified above.

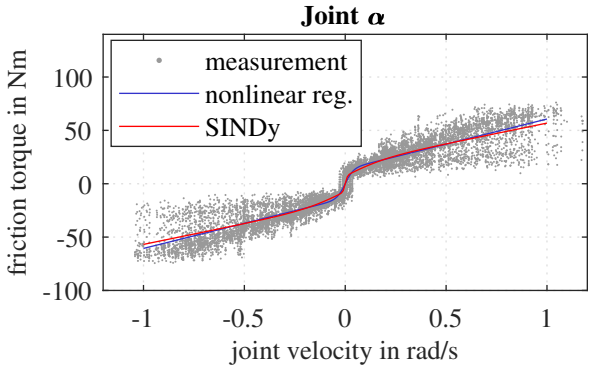

Figure 14: Fitted friction models for joint A. No preprocessing is needed.

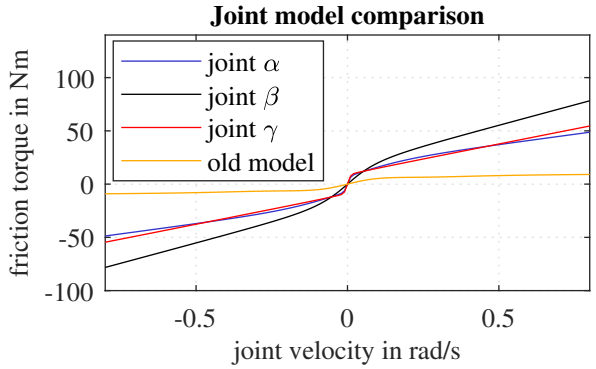

Figure 15: Comparison of the three new joint friction models with the old model from [11]. Visualized are the identified SINDy models.

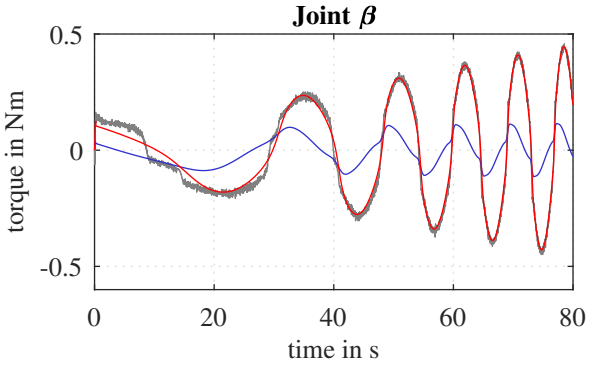

Figure 16: Feed forward torqes of the old and new model compared to the MPC output torque applied at joint β.

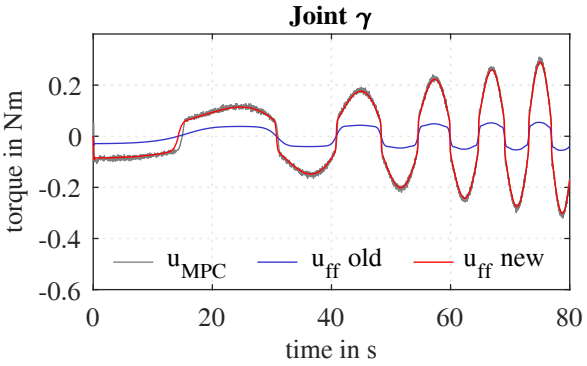

Figure 17: Feed forward torqes of the old and new model compared to the MPC output torque applied at joint γ.

Conclusion

With the methods and results developed throughout this paper, we can finally find the following conclusion. Both, System Identification for Nonlinear Dynamics (SINDy) and nonlinear regression were applied to identify friction models for the relevant joints B and C of the PowerCube robot. Additionally, a friction model for joint A was identified for use with a linear quadratic regulator. The vast improvement of the feed-forward calculations together with the newly identified joint friction models imply a general improvement of the robot model. Although the feed-forward torque almost matches the controller output, slight differences still remain that result from model errors within the rigid body model without friction. However, better trajectory tracking performance, critical for practical usage of the robotic manipulator, could not be improved with the current setup. A first reason lies in using a linearizing MPC approach that does not adequately represent nonlinear system dynamics. Second, the setup underlies limitations in hardware and software. On the hardware side, communication between the robot and the centralized controller in Simulink is limited in speed since the Schunk PowerCube Robot is designed for decentralized joint control schemes such as PID. On the software side, the MPC online optimization cannot be evaluated arbitrarily fast. Both limitations contrast the need for trajectories and linearized dynamics with high resolution in time whenever using LTV MPC.

Further research out of the scope of this publication has shown improvements in trajectory tracking performance with the newly identified friction models when using a truly nonlinear MPC scheme.

When determining the friction, the research has been shown that the friction determination for isolated joint connection modules is not sufficient. Within the robot assembly, the axes of the joints have to bear an additional load, which leads to higher friction.

The SINDy method and its sparse regression algorithms make for a robust and fast method for identifying nonlinear system dynamics. Nonlinear regression in comparison leads to very similar results but needs more computation time due to its nonlinear optimization, making it less capable for real-time applications.

All in all, SINDy's potential to identify parts of unknown system dynamics was successfully presented.

Acknowledgements

Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 - 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). We thank Patrick Schmid and Georg Schneider for the helpful discussions regarding MPC and the experimental setup.

References

[1] Fehr, J.; Schmid, P.; Schneider, G.; Eberhard, P.: Modeling, Simulation, and Vision-/MPC-based Control of a PowerCube Serial Robot. Applied Sciences, Vol. 10, No. 20, id. 7270, 2020.

[2] Rawlings, J.B.; Mayne, D.Q.: Model Predictive Control: Theory and Design. Madison: Nob Hill Publishing, 2009.

[3] Maciejowski, J.M.: Predictive Control with Constraints. Upper Saddle River: Prentice Hall, 2002.

[4] Schnelle, F.; Eberhard, P.: Real-time Model Predictive Control of a Pendulum. PAMM, Vol. 14, No. 1, pp. 907–908, 2014.

[5] Grotjahn, M.; Heimann, B.: Model-based Feedforward Control in Industrial Robotics. The International Journal of Robotics Research, Vol. 21, No. 1, pp. 45–60, 2002.

[6] Ferreau, H.J.; Kirches, C.; Potschka, A.; Bock, H.G.; Diehl, M.: qpOASES: A Parametric Active-Set Algorithm for Quadratic Programming. Mathematical Programming Computation, Vol. 6, No. 4, pp. 327–363, 2014.

[7] Brunton, S.L.; Proctor, J.L.; Kutz, J.N.: Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems. Proceedings of the National Academy of Sciences, Vol. 113, No. 15, pp. 3932–3937, 2016.

[8] Kaiser, E.; Kutz, J.N.; Brunton, S.L.: Sparse Identification of Nonlinear Dynamics for Model Predictive Control in the Low-data Limit. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 474, No. 2219, p. 20180335, 2018.

[9] Kurz, T.; Eberhard, P.; Henninger, C.; Schiehlen, W.: From Neweul to Neweul-M^2: Symbolical Equations of Motion for Multibody System Analysis and Synthesis. Multibody System Dynamics, Vol. 24, No. 1, pp. 25–41, 2010.

[10] The Mathworks, Inc., Natick, Massachusetts: MATLAB and Real-Time Windows Target R2014b, 2014.

[11] Oberhuber, B.: Ein Beitrag zur modularen Modellierung und Regelung redundanter Robotersysteme (in German). Linz: Trauner, 2013.

[12] Rudy, S.H.; Brunton, S.L.; Proctor, J.L.; Kutz, J.N.: Data-driven Discovery of Partial Differential Equations. Science Advances, Vol. 3, No. 4, id. e1602614, 2017.

[13] MathWorks: Documentation for Matlab, Release 2019b, 2019. Accessed: 2022-03-07, Url: `https://www.mathworks.com/help/releases/R2019b/index.html`.

[14] Chartrand, R.: Numerical Differentiation of Noisy, Nonsmooth Data. ISRN Applied Mathematics, Vol. 2011, id. 164567, 2011.