Universal identification and control of industrial manufacturing equipment as a service

V. Tessaro[1], A. Vick[2] and J. Krüger[3]
[1] Technical University Berlin, Berlin, Germany
[2] Process Automation and Robotics, Fraunhofer IPK, Berlin, Germany
[3] Industrial Automation Technology, Technical University Berlin, Berlin, Germany
E-mail: axel.vick@ipk.fraunhofer.de

Abstract. Rapid prototyping as well as retrofitting and digitization of legacy manufacturing equipment often needs design and application of closed loop controllers. The analysis and modeling for such systems like energy-conversion or material transport devices is labor-intensive and needs process understanding. This paper presents a universal approach of identification and closed loop control of arbitrary linear systems delivered through web services using OPC UA as a standardized industrial communication interface. The identification service was used to model the dynamics of a 6-DOF industrial robot and a laboratory-scale water plant containing two separately controllable pumps. The control service successfully controlled the robot’s linearizable axes and the water plant’s pumps by using their respective identified state-space models. To evaluate the performance of the controllers in terms of stability, accuracy, and time response, target trajectories and disturbances like signal noise and latency in communication were introduced. Simulation and laboratory experiments show promising results for control of diverse systems with varying time-constants and imply broad applicability. So, this paper brings a guideline to efficiently implement model predictive control in manufacturing.

1. Introduction
With the transformation to industry 4.0 the automation and digitization of factories and production systems is increasing constantly. The development turning away from monolithic control structures and towards high connectivity in distributed systems opens up new opportunities of using cloud services. Aside from computing and storage capacities, cloud services can also offer complex functionalities like the control of industrial robots and other equipment [1].

Cloud service based control structures in conjunction with services for system identification have the potential to make the integration of conventional industrial equipment – independent of type or brand – and new prototypes into a factory cloud possible or easier. Due to the cloud services’ universal applicability labor-intensive and knowledge requiring model synthesis and optimization of control algorithms can be omitted. Thus, research potential arises concerning the feasibility of the universal approach as well as the applicability to unknown dynamic systems. In this paper we focus on linear or linearizable systems for a start.

2. Related works
The outsourcing of computationally demanding functionalities to a cloud can be attractive and is researched intensively. Such a task with a high requirement for computing capacity is image
processing. Rudorfer et al. [2] designed a cloud-service based system architecture within which image processing is offered as a web service. From their experiments they conclude the possibility of fast development of new image processing applications. Due to the usage of the communication protocol OPC UA different modular services from diverse sources can be employed. Though, the authors found the services to lack real-time capability.

Wassermann et al. [3] also designed a cloud-service based system architecture with the objective to offer path planning, robot control, virtualization and hardware services as modular, independent, relocatable web services. They demonstrated their approach with a pick-&-place task in an environment containing obstacles. The object recognition and the path planning were executed correctly and the authors found the processing time to be acceptable. For providing universal applicability they proposed the usage of OPC UA as a communication interface.

In a network environment with varying latency and bandwidth Kilinc et al. [4] compared the performance of two cloud-service based controllers, a model predictive and a P-PI-controller, when controlling a 6-DOF industrial robot. Therefor, the model predictive control (MPC) was modified so that a once computed input sequence could be further used in the case of connection failures. The MPC was based on a state-space model that was derived analytically and the controllers’ parameters were tuned manually. In experiments the MPC turned out to be more stable and faster for increasing latencies, tested up to 160 ms. Additionally, it showed less overshoot and a smaller steady-state error than the P-PI-controller.

Instead of the derivation of a robot’s state-space model Briese et al. [5] investigated the possibility to use a non-model based approach, the modified Switching Active Disturbance Rejection Control, for a cloud-service based, distributed robot control in a network with jitter. In experiments they achieved the stabilization of all six robot axes for cycle times up to 100 ms but physical disturbances could not be compensated completely and the stabilization at high cycle times could only be achieved with loss of performance. Though, one of the advantages of the authors’ approach is the small number of parameters to tune and therefore an easier model synthesis than when using first-principle methods.

### 3. Methods for universal identification and control

#### 3.1. OPC UA

For the communication between servers and clients we chose the OPC UA standard. OPC UA is a TCP/IP-based machine-to-machine protocol for industrial server-client applications and it offers access options for data and remote procedure calls on servers. We used the binary protocol version because of its smaller overhead and therefore higher speed and lower resource consumption.

Based on the principle presented by Vick at al. [1], we introduced an additional layer between logical functionalities and communication. We built all services as automation services that inherit from an OPC UA adapter class and contain an automation class object. The adapter class contains methods for communication whereas the automation class provides the more complex functionalities. Thus, we achieved a separation of these two functionalities which allows easier reuse of both. We built these classes using the python-opcua [6] module for our servers implemented in Python and using the open62541 library [7] for our servers in C++.

#### 3.2. Subspace Identification

A state-space model is a transfer system’s representation containing the relations of input, state, and output variables as first order differential equations. Its system specific matrices are obtained by performing a subspace identification on an input-output data set. With this data subspace identification methods try to find a subspace spanned by the columns of an extended observability matrix. One well known method is Multivariable Output Error State Space (MOESP) which is used in this paper.
Figure 1. System architecture with communication between servers and clients.

The Python module SIPPY [8] provides MOESP and other methods for identifying dynamic SISO and MIMO systems based on their input-output data. If the system to be identified is a robot-like system, e.g. if the respective numbers of first and second order states and the number of inputs are equal, then we assume one independent system per axis, identify these systems individually and eventually combine them to the overall state-space matrices. Otherwise, if we cannot make an assumption about the system then the system order and the model structure are chosen automatically by the means of different models’ Bayes Information Criterion. Subsequently, we transform the matrices so that the output matrix is an identity matrix and we discretize the state-space model.

3.3. Model Predictive Control
A discrete state-space model can be used in model predictive control (MPC) of processes for estimating future outputs. These future outputs depend on a sequence of future and past inputs. The deviation of the estimated outputs from a target trajectory and the control effort up to a control horizon are weighted in an objective function. This function is then minimized by iteratively modifying the input sequence with respect to the input limitations. Only the first signal of the input sequence is used to control the system and all other inputs are discarded because finally the control horizon is shifted one step to the future and the control cycle restarts.

Vick et al. [9] extended the MPC principle considering distributed systems with varying latencies in communication. Instead of discarding all input signals except the first, all inputs are buffered. This allows to use a shifted input sequence to compensate for the transfer time from the controller to the system as well as to reuse a once computed input sequence in case of disturbances of the cycle time or communication failures.

4. System Architecture
We designed a cloud-based system architecture where every service’s task is processed independently of other services. The orchestration client[2] cycles through the process logic and forwards service’s outputs to other services and thereby triggers server-side actions. The services and the orchestration client as well as their exchanged data are depicted in figure 1.

The system service is closest to the controlled system and provides access to its hardware e.g. actuators controlled by inputs and sensors outputting the current state. Additionally, there is a simulation service that can simulate such a system and its reactions to inputs. The identification service identifies a state-space model based on input and output data. The simulation service and the MPC service have to be initialized with this model. When provided the current state of the system, the MPC server can then compute an input sequence to reach a target state. Target states, trajectories and input sequences can be generated by the target client.
Table 1. Average steady-state errors, overshoots and rise times of the robot axes 1, 4, 5, and 6 and the water plant (W).

| Achse | 1     | 4     | 5     | 6     | W     |
|-------|-------|-------|-------|-------|-------|
| Steady-state error | -0.4% | -3.9% | -0.9% | -2.1% | 0.2%  |
| Overshoot | 5.4%  | 1.9%  | 14.5% | 9.4%  | 31%   |
| Rise time | 1.0 s  | 2.4 s  | 0.5 s  | 0.4 s  | 0.6%  |

5. Experimental setup and plan
We tested our approach using a 6-DOF robot and its simulation of which the axis positions and velocities are controlled through the axis motors’ currents and a laboratory-scale water plant with two pumps that can be used to control the pressure difference from the system normal. Both systems provide access to their actuators and state variable measurements with the means of OPC UA servers. Firstly, random input sequences are used to record input-output data sets of the systems. Then, based on this data we identify one state-space model per data set. Finally, we use each of these models to control the systems on a target trajectory.

In the simulator experiments, we use two random trajectories, one for identification and one for validation. The identification trajectory is re-recorded with an increasing standard deviation of Gaussian noise ranging from 0 degrees to 3 degrees and with increasing latencies of 7 ms, 14 ms, 35 ms, 70 ms, 105 ms, and 140 ms. We check for model accuracy by simulating the validation trajectory with the identified models and calculating the normalized root mean square error (NRMSE) as a means of comparison of modeled and true state variables.

For the robot and water plant, we record six identification trajectories and one validation trajectory, respectively. For each model parameter, we calculate the parameter mean and certainty. Then we check again for model accuracy by comparing the validation trajectory simulated with the identified models to the real validation trajectory.

In order to test for controller performance, we control the all systems on a target trajectory and calculate the mean steady-state errors, overshoots and rise times.

6. Results and Discussion
In the simulations, the model accuracy for the axis velocities was higher than for their positions and decreased with increasing latency from a mean NRMSE of 0.73 to 0.025 and remains constant for increasing noise. As does the steady-state error which stays in the range from -1.5% to 1% for all tested noise levels and latencies. The overshoot increases from 0% to an average of 2% for growing latencies and remains constant for increasing noise. Rise times and overshoots depend significantly more on the MPC weights than on noise or latencies.

In the robot experiments the axes 1, 4, 6, and 5 turned out sufficiently linear so that a linear state-space model could be identified. Therefore, only those axes were considered in the experiments. At a mean latency of 70 ms, the NRMSE of the velocities was again better than the NRMSE of the positions with mean values of 0.83 and 0.23, respectively. Small-scale bootstrapping showed certainty of the model parameters with all certainties at least one magnitude smaller than their respective parameter’s mean except for the parameters that were identified as zero. The steady-state error for axes 1 and 5 was lesser than 1% and for axes 4 and 6 lesser than 4%. The tuning of the MPC weights was more difficult for the real robot’s axes leading to higher average values of overshoots as shown in table 1.

The water plant showed to be sufficiently linear in a range from 60% to 120% of pump power. At a mean cycle time of 300 ms the NRMSE of the modeled system pressure was 0.78 with a high certainty of the models’ parameters. The steady-state error (see table 1) is lower than the actual value because the it is negative if the pressure differences are lesser than 0.08 Pa and positive if they are greater than 0.08 Pa (see figure 2). Thus, the error values cancel each other
out on average. After removing outliers considered to be singular non-linearities the average overshoot is still high with 31%. This is because no negative inputs are possible for the water plant and therefore the MPC weights should have been tuned to punish the control effort more.

7. Conclusion

For the MPC applies: the greater the discrepancy between a system and its model the less performant the MPC will be that is based on such model. Yet, the adequate tuning of the MPC weights has a significant impact on the control performance too which we found to exceed the influence of fluctuations in model quality caused by disturbances like noise or latency of the tested magnitudes. We identified the models of structurally and physically diverse systems and successfully controlled them by using these models in an MPC, thereby implying the feasibility of our approach. However, further research is needed to identify universal and automatable methods for tuning MPC weights so that a consistently high control quality for a multitude of systems can be achieved. Likewise, the extension of our linear modeling approach by a non-linear portion would allow for a greater variety of controllable systems.

References

[1] Vick A and Krüger J 2018 Using opc ua for distributed industrial robot control ISR 2018; 50th Int. Symp. on Robotics (VDE) pp 1–6
[2] Rudorfer M and Krüger J 2018 Procedia CIRP 76 144–8
[3] Wassermann J, Vonašek V and Vick A 2018 Distributed industrial robot control using environment perception and parallel path planning cloud services 2018 IEEE 23rd Int. Conf. on Emerging Technologies and Factory Automation (ETFA) vol 1 (IEEE) pp 1263–6
[4] Vick A, Kilinc A and Guhl J 2019 Variable-latency networked p-pi and mpc controller performance for industrial robots 2019 IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA) (IEEE) pp 1501–4
[5] Briese C, Vick A and Krüger J 2018 Cloud-based active disturbance rejection control for industrial robots 2018 IEEE 23rd Int. Conf. on Emerging Technologies and Factory Automation (ETFA) vol 1 (IEEE) pp 559–65
[6] Free OPC-UA Library python-opcua https://github.com/FreeOpcUa/python-opcua [Accessed: 23 Jan 2021]
[7] Pfrommer J, Profanter S, Iatrou C P, Palm F and the open62541 community open62541 https://open62541.org/ [Accessed: 23 Jan 2021]
[8] Armenise G, Vaccari M, Di Capaci R B and Pannocchia G 2018 An open-source system identification package for multivariable processes 2018 UKACC 12th International Conference on Control (CONTROL) (IEEE) pp 152–157
[9] Vick A, Guhl J and Krüger J 2016 Model predictive control as a service – concept and architecture for use in cloud-based robot control 2016 21st Int. Conf. on Methods and Models in Automation and Robotics (MMAR) (IEEE) pp 607–12