Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 127 (2024) 116–121

www.elsevier.com/locate/procedia

10th CIRP Conference on Assembly Technology and Systems (CIRP CATS 2024)

Jacobian-Sensitivity Approach for Identifying Machine Dynamic Model Parameters of Robots with Flexible Joints

Florian Oexle[a,*], Achim Benfer[a], Alexander Puchta[a], Jürgen Fleischer[a]

[a]*wbk Institute of Production Science, Karlsruher Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*
* Corresponding author. Tel.: +49-174-330-2745 ; *E-mail address:* florian.oexle@kit.edu

Abstract

The versatility and large work envelope have made robots a fixture in the field of assembly for years. However, their lower stiffness and pose dependency require robust models to find optimal trajectories even for high accuracy applications. A significant obstacle in this domain is parametrizing such models of compliant robots during operation. Addressing this gap, and considering the trend of robots performing manufacturing tasks in parallel with assembly, we present an automated identification process to estimate the stiffness and damping parameters of robot joints within a milling process. This method relies solely on universally accessible kinematic chain data and force and acceleration measurements at the tool center point, eliminating the need for specialized equipment. The approach is based on a multi-body simulation, which includes flexible 6-DOF bushing joints. Key to our approach is using Jacobian-based sensitivities inside a Random Search (RS) algorithm to navigate the complexities of a sparse multi-dimensional parameter space. Our approach is versatile enough to accommodate various parameter types. We test our approach on a simulated 3-joint robot with 6 DOF per joint. By pairing the Jacobian-based sensitivities with adaptions made to the RS algorithm, we obtain accurate predictions for unknown input data with a mean relative displacement error of 2%.

Keywords: robot; simulation; optimization; machine tool; milling

1. Introduction

Industrial robots have seen a consistent surge in sales over recent decades. However, despite their lower cost and greater operation reach [1], they are not yet a common alternative for high-precision machinery in assembly and manufacturing. One salient limitation is their open kinematic chain, leading to significantly reduced stiffness compared to alternative machines. This results in positioning inaccuracies and low-frequency mode coupling chatter, which reduces the precision of assembly and machining processes and shortens the lifespan of tools and robots. [2, 3]. Significant reductions in positioning errors and chattering can be achieved by systematically selecting trajectories [4, 5] and feed rates [6].

To determine optimal machining paths and feed rates, a model that accurately predicts the robot's dynamic behavior under load is essential. While many dynamic robot models exist, the difficulty lies in parametrizing those models, as robot manufacturers usually withhold dynamic parameters such as stiffness and damping values. Current literature has yielded many solutions to identify the stiffness and damping parameters of robots [1, 7, 8]. However, the proposed solutions are not fully automated and rely on user expertise, making them impractical for non-experts.

This paper investigates methods to identify model parameters during a milling process. Navigating the high-dimensional solution space is challenging, so we investigate multiple methods for effective parameter identification. The presented approach is based on only widely available data like kinematic chain data and operational robot data. Through this, we aim to eliminate human influences and make precision improvements available for every robot, irrespective of the availability of detailed machine models or user expert knowledge. The parametrized model is valid across multiple domains, such as machining, fabrication, and assembly, and is applicable to individual robots over their lifetime.

10.1016/j.procir.2024.07.021

2. Related Work

Identifying dynamic model parameters is intrinsically tied to the model being parameterized. Dynamic robot models are typically classified as either experimental or theoretical models. *Experimental models* function by learning transfer functions to mimic input-output behavior. They are based on fundamental building blocks that simplify the parameterization process due to their universality. However, experimental models often have limitations: they are typically valid only within a confined working space or specific robot configurations. Furthermore, their parameters have no physical meaning and cannot be transferred to other models.

In contrast, *theoretical models* are grounded in tangible physical principles. These physical correlations not only offer interpretability but also facilitate transferability across models. However, accurately identifying parameters for those models proves challenging. [9]

Distributed element models, such as Finite Element (FE) models, represent one category of theoretical models. To perform parameter identification on them, which is very resource intensive, they are usually simplified using, for instance, model order reduction or by creating a meta-model [10]. Distributed element models require a detailed 3D model of the robot, often unavailable to robot users. An alternative lies in *multi-body models*, which rely on the more commonly accessible kinematic chain data. Siciliano and Khatib describe such a robot model with rigid links connected by joints modeled as spring and damper systems in the freely movable axis [11]. Yet, Moberg et al. contend that a single DOF per joint might not capture the entire robot dynamics [12, 13]. They suggest an extended flexible robot model with three DOFs per joint - a viewpoint validated by Öhr et al., who further recommend adding translational DOFs as required [14].

Once the model is defined, a method must be developed to estimate the model's parameters. Those parameter identification methods can be divided into direct and iterative methods. *Direct methods* like in [7] derive the equation of motion (EOM) of a multi-body model. To solve the EOM for the stiffness parameters, they simplify and linearize it and perform linear regression. A contrasting approach is seen in *iterative methods* such as those presented by Ellinger and Zaeh [8]. Here, parameters are optimized iteratively to minimize the difference between simulated and real-world robot behavior. They don't require a simplified model but must manage local optima and a sparse high-dimensional input space.

Several methods aim at reducing the dimensionality of the input space to reduce the complexity of parameter identification. Niehues and Semm et al. sequentially assemble the robot from the base to its TCP, conducting experiments after every assembly step [15, 16]. This task requires heavy manual input and expertise. Ellinger et al. use a global sensitivity analysis (GSA) and only optimizes the parameters above a certain threshold [17]. This is applicable when parameter bounds are narrowly defined. However, when they are not, the GSA performs poorly. Additionally, a GSA requires high computing time.

3. Dynamic Model Parameter Identification

Our approach differs from the state of the art in that it does not require the following:

1. Any model data that is not widely available or automatically retrievable. Our approach builds the model using only kinematic chains - such as DH-Parameters - known to the robot, as well as mass and inertia data that can be acquired through automated test runs. [18].
2. Special measurements that cannot be automated. I.e., no impact hammer testing, no sequential assembly tests. We use acceleration and force data recorded during operation at the robot's TCP.
3. User expert knowledge, e.g., manual modeling or narrow parameter bounds. We use broadly defined parameter bounds for practical reasons.

Since finding a suitable optimization method requires testing many different algorithms and parameters, we use a simulated model as reference, subsequently called "reference model", instead of measured data of a real robot. We aim to identify the stiffness and damping parameters of a flexible joint multi-body model in the following section.

3.1. Structure of the multi-body model

The kinematic chain data enables constructing a multi-body model. More complex models like FE or CAD models require data that is not widely available and can, therefore, not be considered. Within our multi-body model, we incorporate six DOFs at each joint: three rotational and three translational. Öhr et al. validated that three rotational DOFs are sufficient to accurately model a robot's displacement [14]. We follow their suggestion to add translational DOFs, ensuring the model's suitability for all kinematics. Each DOF is modeled by a linear spring and a viscous damper. Each link is assumed to be a rigid body. Masses and inertias are assumed to be given, and the robot's pose is assumed to be static during a measurement except for any displacement caused by the robot's flexibility. The behavior of the multi-body model is fully described by its equation of motion (EOM) [11]:

$$\boldsymbol{M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + D\dot{q} + Kq = \tau}$$

With inertia matrix $\mathbf{M}$, Coriolis and centrifugal forces $\mathbf{C}$, gravitational forces $\mathbf{G}$, joint damping $\mathbf{D}$, joint stiffness $\mathbf{K}$, external forces/torques $\boldsymbol{\tau}$, and generalized coordinates $\mathbf{q}$. We use a MATLAB Simscape multi-body model, automatically created from a urdf file, to build and run the simulation, as it ensures fast and reliable differential equation integration. An illustration of our 3-joint robot model can be seen in Fig. 1.

3.2. Design and Choice of the Optimization Algorithm

After building the multi-body model, we aim to identify the values of stiffness $\mathbf{K}$ and damping $\mathbf{D}$. Direct solutions for $\mathbf{K}$ and $\mathbf{D}$ in the EOM are infeasible due to numerical limitations. While linearization is an option, our preferred approach uses iterative methods to minimize simulation discrepancies.

Given input forces, joint configurations, and stiffness and

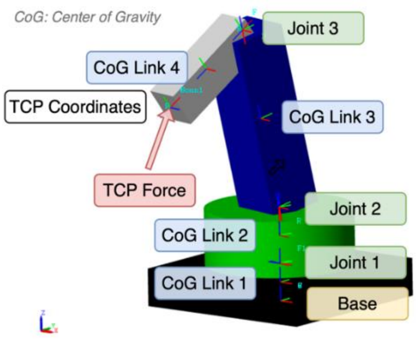

Fig. 1. Visualization of the multi-body model with three joints.

damping parameters, our multi-body model simulates TCP displacement in three dimensions. We evaluate the parameter guess using a loss function that compares TCP displacements of our model to the displacements of the reference model. The optimizer iteratively refines parameter guesses based on this loss function. One parameter guess is called a sample. This approach can be classified as a curve-fitting method, displayed in Fig. 2.

Our model contains 12 parameters (6 stiffness and 6 damping parameters) per joint, which are coupled. The loss function, when mapped against the parameter space, is non-convex. Therefore, an algorithm suitable for high-dimensional, non-convex optimization tasks is required. Deterministic methods like gradient descent risk local optima. We tested various stochastic algorithms, including Genetic Algorithms and Particle Swarm Optimization. Random Search (RS) algorithms outperformed others, especially when combined with annealing, making them our chosen optimization technique. The general sequence of an RS with annealing involves five key steps:

1. Sample N parameter sets within a hypersphere.
2. Compute the loss function for each sample.
3. Recenter the hypersphere at the best sample.
4. Adjust the hypersphere radius.
5. Continue steps 1-4 until a stopping criterion is met.

Fig. 3 illustrates this process. The orange triangles are the initial samples. The best sample is highlighted with a white circle, around which a new hypersphere (blue circle) is established for subsequent sampling. This iterative process refines the hypersphere and converges to an optimum.

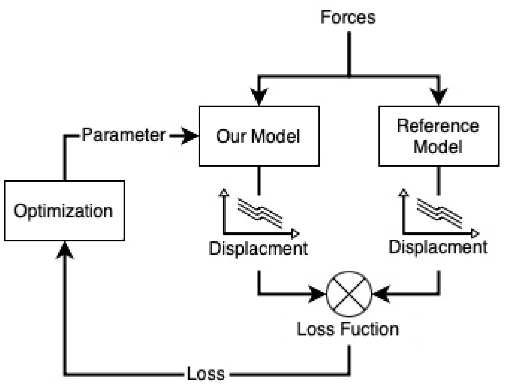

Fig. 2. flowchart of the curve fitting optimization process.

In our application, each sample is evaluated across multiple joint configurations and input force profiles, called "experiments". We evaluate each sample through three experiments. Each experiment simulates a real-world machining step, like a milling path or drilling a hole. Using multiple experiments enhances robustness by mitigating ambiguities and exciting more DOFs.

We performed optimizations using a random search algorithm with annealing [19]. Although this algorithm is superior to the alternatives mentioned, it encounters local optima. It struggles to achieve parameter deviations of less than 25% with a mean absolute error (MAE) loss function and scheduled bound shrinking. To address this issue, we adapt the algorithm as follows. We call the unadapted algorithm "baseline". Each adaptation is tested separately against the baseline, as described in the next section. All results are shown in Table 1.

3.3. Adaptions to the Baseline

First, various loss functions are examined to determine their effectiveness in minimizing the parameter deviation. We review the mean absolute error (MAE), the mean squared error (MSE), and the Surface Similarity Parameter (SSP) as candidate loss functions:

- $\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i|$
- $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$
- $\mathrm{SSP} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathcal{F}(x)_i - \mathcal{F}(\hat{x})_i|_2^2}{|\mathcal{F}(x)_i|_2^2 + |\mathcal{F}(\hat{x})_i|_2^2}$

With displacement of the reference model x_i, and of our model $\hat{x}_i$ and the Fourier-Operator $\mathcal{F}$. In our experiments, we find that using the SSP proposed by Wedler [20] as a loss function, the parameter deviation is reduced from an average of 26.57% after 500 iterations to 25.17% (see Table 1).

Next, we address balancing exploration and exploitation in the optimization process. We call the method "Look Outside." It randomly selects a single parameter for evaluation and samples across its entire bound range while the other parameters remain constant. This targeted exploration mitigates the risk of local optima in high-dimensional, non-

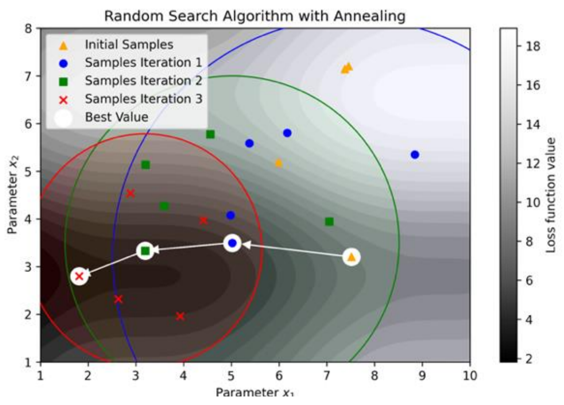

Fig. 3. simple 2D random search algorithm with annealing.

convex optimization tasks and serves as an exploration technique. The method reduces the average parameter deviation from 26.57% to 24.41% (see Table 1).

Finally, we compare different bound shrinking methods. The bounds define the parameter sample region of the next iteration. In detail, we investigate a scheduled, a global adaptive, and a local adaptive bound shrinking method. The scheduled approach shrinks the bound range by a factor of < 1 after each iteration. The adaptive methods adapt the bound range based on the loss instead of the iteration. While the global approach defines one bound range for all parameters, the local defines the bound range for each parameter individually. Testing showed no advantage in using adaptive bounds against scheduled bound shrinking in the baseline.

The results clearly show the importance of choosing a suitable loss function and defining exploration methods. Rather surprisingly, more advanced bound shrinking methods yield no improvements to the optimization.

Table 1: Results from adaptions of Section 3

Method	Mean Parameter Deviation	Delta Baseline
Baseline	26.57 %	
Mean Squared Error (MSE)	25.20 %	- 1.37 %
Surface Similarity Parameter (SSP)	25.17 %	- 1.40 %
Look Outside (LO)	24.41 %	- 2.16 %
Adaptive Bounds Global (ABG)	27.66 %	+ 1.09 %
Adaptive Bound Local (ABL)	27.05 %	+ 0.48%

4. Sensitivity Based Optimization

The individual adaptions improve the algorithm's performance (see Table 1), yet the parameter deviations remain large in the first stage of the optimization compared to optimizing a 1 DOF-robot. This might be due to the high dimensionality of the parameter space and the large span of parameter sensitivities. It is worth noting that this issue and the approach we present to mitigate it are independent of the optimization algorithm used. One approach to tackle this problem is proposed by Ellinger et al., who use a global sensitivity analysis (GSA) and only optimize influential parameters while leaving the others constant [17]. This drastically reduces the parameter space.

The sensitivities of a GSA represent the impact of parameter deviations on displacement errors. They are a combination of the following three influences:

1. The strength of a specific parameter. For example, a joint with a lower flexibility has a larger influence on the displacement than a stiff one, as it leads to greater displacements. We call this *intrinsic sensitivity*.
2. A parameter's influence on a specific displacement direction. For example, the stiffness of the first joint of a robot, typically the rotation around the global z-axis, has no influence on the TCP displacement in z-direction. We call this *directional sensitivity*.
3. A parameter's influence on a given joint configuration. For example, a joint could be in an orientation where a DOF is not excited by a TCP force. Thereby, the parameters of this DOF do not influence the TCP displacement. We call this *configuration sensitivity.*

The GSA is suitable for cases where parameter bounds are narrowly defined. Since we do not require user expert knowledge or manual modeling, we assume minimal knowledge of bounds. In such scenarios, the GSA may not accurately capture the intrinsic sensitivity, which describes the strength of a parameter, because the parameter deviation is not linearly correlated with the loss function. Evaluating the sensitivity of one parameter while other parameters deviate significantly from their actual values can lead to incorrect intrinsic sensitivities. To address this potential error, we propose a method that provides information solely on directional and configuration sensitivity without requiring knowledge of parameter bounds. This approach is based on the Jacobian, offering the added advantage of resource efficiency, as calculating the Jacobian is standard practice. Subsequently, we present the calculation of the Jacobian and its integration into the optimization algorithm.

4.1. Calculation of Jacobian Sensitivities

The Jacobian can be used to relate DOF velocities to TCP velocities. Using the Jacobian, we aim to partially separate the effects of parameters on the displacement to reduce the dimensionality of the optimization. Since we only measure the translation and not the rotation of the TCP, we only use the translational components of the Jacobian, defined by:

$$\boldsymbol{J}_{\boldsymbol{C}\times\boldsymbol{D}} = \left(\frac{\partial f_i}{\partial q_j}\right)_{\boldsymbol{i=x,y,z;\, j=1,\dots,D}} \qquad (1)$$

Here f_i denotes the TCP displacement in TCP coordinate directions. q_1 to q_D are the generalized coordinates of the DOFs with D, the total number of degrees of freedom (18 in our case). A large entry in (1) means the DOF significantly influences the movement of the TCP, meaning that the effect of the stiffness and damping parameter of the DOF is well observable. In contrast, a null entry means the parameters of the specific DOF do not influence the movement of the TCP in that direction. This is directional sensitivity.

To estimate the configuration sensitivity, we extend the Jacobian for all our experiments. We obtain a matrix with $C \cdot E$ rows and D columns. C is the number of coordinate directions (usually 3), and E *is* the number of experiments (3 in our case). Resulting in the formula:

$$\boldsymbol{J}_{\boldsymbol{C\cdot E}\times\boldsymbol{D}} = \left(\frac{\partial f_i}{\partial q_j}\right)_{\boldsymbol{i=x1,y1,z1,\dots,xE,yE,zE;\, j=1,\dots,D}}$$

As the Jacobian is equal for damping and stiffness values within a DOF, we apply the same matrix to our stiffness and damping parameters.

4.2. Integration into the Optimization Algorithm

Including the Jacobian in the optimization process is an important step. After every simulation, we obtain $C \cdot E$ loss values. Until now, we took the mean of them to assign one loss value to every parameter set. Now, we calculate a custom loss value for every parameter of every parameter set. To do this, the Jacobian is normalized column-wise, i.e., for the same parameter, the sum of all the entries is one. From now on, the normalized entries of the Jacobian are referred to as "sensitivities". Then, a weighted sum of the $C \cdot E$ loss values with the sensitivities as weights is performed. The weighted parameter loss for one sample is calculated by

$$\boldsymbol{L}_{1 \times P} = \boldsymbol{L}_{1 \times C \cdot E} \cdot \boldsymbol{S}_{C \cdot E \times P}$$

with loss

$$\boldsymbol{L}_{1 \times C \cdot E} = \begin{pmatrix} l_{c_x e_1} & l_{c_y e_1} & l_{c_z e_1} & \cdots & l_{c_z e_E} \end{pmatrix}$$

and sensitivities

$$\boldsymbol{S}_{C \cdot E \times P} = \begin{pmatrix} s_{c_x e_1 p_1} & s_{c_x e_1 p_2} & \cdots & s_{c_x e_1 p_D} \\ s_{c_y e_1 p_1} & s_{c_y e_1 p_2} & \cdots & s_{c_y e_1 p_D} \\ s_{c_z e_1 p_1} & s_{c_z e_1 p_2} & \cdots & s_{c_z e_1 p_D} \\ \cdots & \cdots & \cdots & \cdots \\ s_{c_z e_E p_1} & s_{c_z e_E p_2} & \cdots & s_{c_z e_E p_D} \end{pmatrix}$$

The random search algorithm draws new samples on each iteration in a hypersphere defined by a center point and a radius. In the former approach, that center point would be the parameter value of the best sample of an iteration. Therefore, only one sample of all drawn samples would be considered. Now, parameters can be chosen from individual samples because the loss value is specific to every parameter and not the whole sample, providing more specific results.

The same integration can be applied to GSA-based sensitivities. Using the GSA is compelling because it provides not only information about the directional and configuration sensitivity like the Jacobian but also intrinsic sensitivity. On the other hand, the Jacobian-based method does not require parameter bound knowledge and computes faster. To evaluate the performance of either method combination, we test them in the following section.

5. Results and Discussion

To assess the adaptions made in Sections 3 and 4, we evaluate the parameter deviations of the different method combinations during the simulation in Section 5.1 and perform a validation test in Section 5.2. We perform the tests on each of the following method combinations:

1. *Baseline*: Same as Baseline of Section 3
2. *Baseline adapted*: Baseline with adaptions from Section 3
3. *GSA*: Baseline adapted with GSA-sensitivity-based optimization
4. *Jacobian*: Baseline adapted with Jacobian-sensitivity-based optimization

5.1. Parameter Deviations

We investigate the first stage of the algorithm, in which we aim for a fast reduction of the parameter deviation. Parameter deviation is defined as the mean absolute error between actual and identified parameters divided by the bound range:

$$E = \frac{1}{n \cdot R} \sum_{i=1}^{n} \left| P_i - \widehat{P_i} \right|$$

With parameter deviation E, number of parameters n, bound Range R, actual parameter P_i, and identified parameter $\widehat{P}_i$. The mean parameter deviation over ten simulations is plotted against the iteration in Fig. 4 and the results after 500 iterations are listed in Table 2. The baseline performs the poorest throughout the optimization. The adapted baseline with SSP-Loss and the additional exploration method outperforms the baseline by 2.89%. Using sensitivities improves the optimization even further by another 2.02% for GSA and 2.44% for Jacobian sensitivities. While GSA-sensitivities show faster parameter deviation reductions, it is outperformed by the Jacobian sensitivities after 320 iterations by a small margin.

Table 2: Results from sensitivity-based optimization

Method Combination	Mean Parameter Deviation	Absolute Change Baseline
Baseline	24.60%	-
Baseline adapted	21.71%	- 2.89%
Global Sensitivity Analysis (GSA)	19.69%	- 4.91%
Jacobian Sensitivity (Jacobian)	**19.27%**	**- 5.33%**

Even after optimization, the parameter deviations remain high. This can be explained by the sensitivity range of the parameters. Some parameters have a diminishing influence on the TCP displacement. As a result, the optimizer, which minimizes the TCP displacement, sets their value rather arbitrarily, deteriorating the parameter deviation significantly.

5.2. Validation Tests

To validate this claim and to test if the fitted model can reproduce the behavior of the reference model, we compare the behavior in experiments unknown to the optimizer. By this, we can evaluate whether the fitted model is resilient to joint configuration changes or just overfitted the presented joint configurations and forces. One exemplary test can be seen in Fig. 5. It shows that the TCP displacement of the reference model in the x-direction and the displacement simulated with

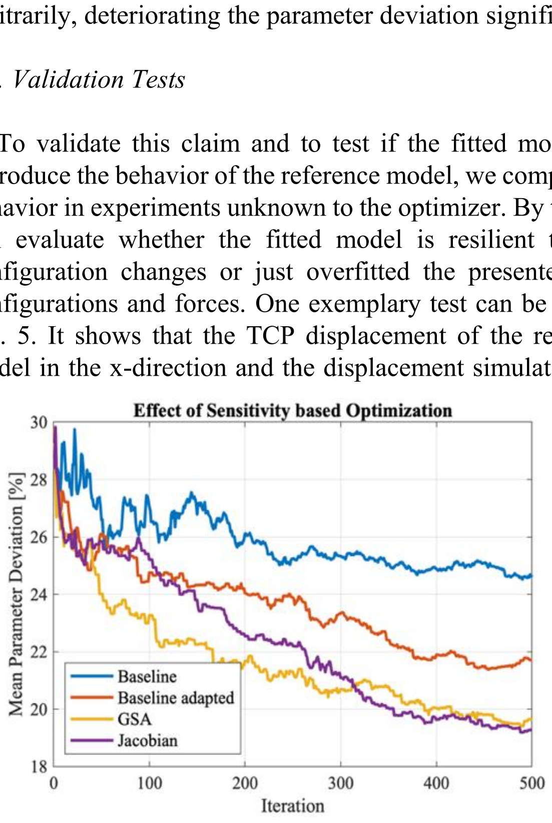

Fig. 4. course of the optimization when using the different method combinations.

the parameters obtained by using the Jacobian-based sensitivities are closely approximated. We calculate the relative error for every direction and experiment to quantify the behavior similarity. Then, we compare the median over all experiments and directions to obtain a single error. We use the median instead of the mean because the relative errors are prone to outliers.

The results can be obtained in Table 3. They support the findings of the parameter deviation analysis. Through the adaptions to the baseline, the median displacement error can be reduced from 3.67% to 2.85%. By incorporating sensitivities into the algorithm, the error can be reduced drastically. In the case of GSA sensitivities, it can be reduced to 2.03% and for Jacobian sensitivities to 2.27%.

Table 3: Validation results

Method Combination	Median relative displacement error	Relative to Baseline
Baseline	0.0367	-
Baseline adapted	0.0285	- 22.36%
Global Sensitivity Analysis (GSA)	0.0203	- 44.69%
Jacobian Sensitivity (Jacobian)	0.0227	- 38.18%

6. Conclusion and Outlook

Our results highlight a promising approach for improving optimization in robotic systems with complex, high-dimensional parameter spaces. The adaptions, particularly when combined with sensitivity-based techniques such as GSA and Jacobian Sensitivities, yield significant reductions in parameter deviations relative to the baseline. Crucially, despite the seemingly high parameter deviations, our validation tests using unknown experimental data demonstrate the model's resilience and precision. After 500 iterations, our sensitivity-based algorithms can predict the robot's behavior with a mean relative displacement error of 2%. Our test shows that using Jacobian-based sensitivities is as good as GSA-based sensitivities with the advantage of no bound range knowledge and less computing time required.

Some aspects remain unresolved, though. While our existing optimization approach often results in a plateau in the optimization progress, introducing alternative strategies in a subsequent phase could further reduce parameter deviations. Additionally, some parameters might be challenging to discern due to their minimal impact. A method to assess the reliability of a parameter estimation is needed. Furthermore, testing this approach on a real robot for practical validation is essential.

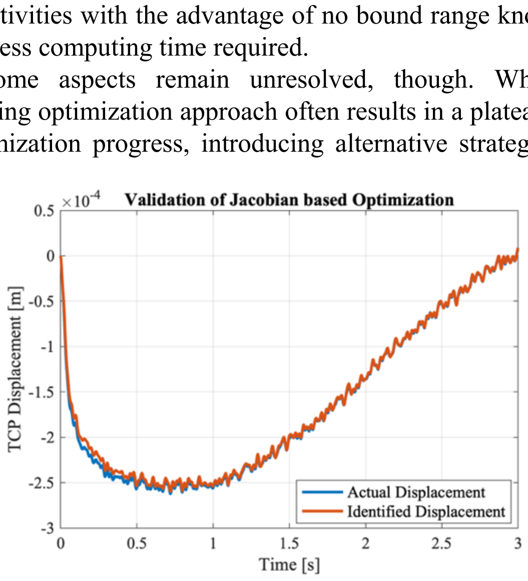

Fig. 5. TCP displacement in the x-direction with parameters obtained by Jacobian-based sensitivities compared to the displacement of the reference model.

Acknowledgements

We extend our sincere thanks to the German Federal Ministry of Economic Affairs and Climate Action (BMWK) and the European Union (EU) for supporting this research project 13IK001ZF “SoftwareDefined Manufacturing for the automotive and supplying industry” (SDM4FZI).

References

[1] Verl, A., Valente, A., Melkote, S., Brecher, C. *et al.*, 2019. Robots in machining *68*, p. 799.
[2] Pan, Z., Zhang, H., Zhu, Z., Wang, J., 2006. Chatter analysis of robotic machining process *173*, p. 301.
[3] Abele, E., Weigold, M., Rothenbücher, S., 2007. Modeling and Identification of an Industrial Robot for Machining Applications *56*, p. 387.
[4] Tunc, L.T., Stoddart, D., 2017. Tool path pattern and feed direction selection in robotic milling for increased chatter-free material removal rate *89*, p. 2907.
[5] Pan, Z., Zhang, H., 2007. Analysis and suppression of chatter in robotic machining process, p. 595.
[6] Wang, G., Dong, H., Guo, Y., Ke, Y., 2017. Chatter mechanism and stability analysis of robotic boring *91*, p. 411.
[7] Zollo, L., Lopez, E., Spedaliere, L., Garcia Aracil, N. *et al.*, 2015. Identification of Dynamic Parameters for Robots with Elastic Joints *7*, p. 843186.
[8] Ellinger, J., Zaeh, M.F., 2022. Automated Identification of Linear Machine Tool Model Parameters Using Global Sensitivity Analysis *10*, p. 535.
[9] Isermann, R., 1992. *Identifikation dynamischer Systeme,* 2nd edn. Springer, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest.
[10] Hernandez-Vazquez, J.-M., Garitaonandia, I., Fernandes, M.H., Muñoa, J. *et al.*, 2018. A Consistent Procedure Using Response Surface Methodology to Identify Stiffness Properties of Connections in Machine Tools. Materials (Basel) *11*.
[11] Siciliano, B., Khatib, O., 2016. *Springer handbook of robotics,* 2nd edn. Springer, Berlin.
[12] Moberg, S., Hanssen, S. A DAE approach to Feedforward Control of Flexible Manipulators, in p. 3439.
[13] Moberg, S., Wernholt, E., Hanssen, S., Brogårdh, T., 2014. Modeling and Parameter Estimation of Robot Manipulators Using Extended Flexible Joint Models *136*.
[14] Öhr, J., Moberg, S., Wernholt, E., 2006. Identification of Flexibility Parameters of 6-axis Industrial Manipulator Models.
[15] Niehues, K.K. *Identifikation linearer Dämpfungsmodelle für Werkzeugmaschinenstrukturen*. Herbert Utz Verlag, München.
[16] Semm, T., Sellemond, M., Rebelein, C., Zaeh, M.F., 2020. Efficient Dynamic Parameter Identification Framework for Machine Tools *142*.
[17] Ellinger, J., Semm, T., Zaeh, M.F., 2022. Dimensionality Reduction of High-Fidelity Machine Tool Models by Using Global Sensitivity Analysis *144*.
[18] Atkeson, C.G., An, C.H., Hollerbach, J.M., 1986. Estimation of Inertial Parameters of Manipulator Loads and Links *5*, p. 101.
[19] Zabinsky, Z.B., others, 2009. Random search algorithms.
[20] Wedler, M., Stender, M., Klein, M., Ehlers, S. *et al.*, 2022. Surface Similarity Parameter: A New Machine Learning Loss Metric for Oscillatory Spatio-Temporal Data.