Received: 28 June 2023 | Accepted: 27 September 2023

DOI: 10.1002/pamm.202300060

RESEARCH ARTICLE

Towards intelligent design assistants for planar multibody mechanisms

Benedict Röder | **Henrik Ebel** | **Peter Eberhard**

Institute of Engineering and Computational Mechanics, University of Stuttgart, Pfaffenwaldring 9, Stuttgart, Germany

Correspondence
Benedict Röder, Institute of Engineering and Computational Mechanics, University of Stuttgart, Pfaffenwaldring 9, 70569 Stuttgart, Germany.
Email: benedict.roeder@itm.uni-stuttgart.de

Funding information
Deutsche Forschungsgemeinschaft, Grant/Award Numbers: 501890093, 501840485

Abstract
General-purpose mechanisms can perform a broad range of tasks but are usually rather heavy and expensive. If only particular movements need to be executed, more efficient special-purpose mechanisms can be employed. However, they typically require an expert to design the system based on manual inspection of simulations and experimental results. This procedure is not only time-consuming, but the outcome also depends on the expert's experience. Hence, the design process stems from subjective criteria while only a limited number of structurally different mechanisms can be considered. In contrast, a design assistant can consider a broad range of mechanisms and leverage multi-objective optimization to retrieve optimal designs for the given task. Due to the systems being synthesized based on mathematical functions rather than individual experience, the assistant allows a more transparent development of optimal problem-specific mechanisms compared to the conventional process. Experts can then fine-tune and analyze the proposed designs to compose the final system. In recent years, neural networks have been utilized to directly learn the inverse mapping from a trajectory to a mechanism design. This requires some parameterization of the trajectory to be fed into the network. In this work, we evaluate various preprocessing methods for the trajectory on a simple mechanism design model problem. We assess multiple configurations such as different neural network sizes, applying input-output normalization, and varying the number of features. Consequently, we investigate and compare the trends and robustness of the implemented methods.

1 | INTRODUCTION

General-purpose robots can perform a variety of tasks and are widely employed. For example, a 3R manipulator (e.g., a robotic arm) can follow a variety of trajectories with its end effector within its workspace. However, to be precise and robust against low-frequency vibrations they are typically constructed in a stiff and heavy manner. This not only requires the use of expensive actuators but those robots are also more dangerous in shared human-robot work environments.

Yet, a lot of tasks only require the execution of a certain trajectory. Here, special-purpose robots can be tailored to the specific problem. Hence, they can be designed to withstand particular vibrations and be made lighter and cheaper. Due to a specialized design, these robots may also be easier to control. However, nowadays, this requires a domain expert to design the system which makes this process expensive and the outcome is based on the expert's experience. Automated design assistants could bridge this gap and consider a vastly broader range of mechanisms while being based on objective and transparent performance criteria. Ideally, an intelligent design assistant could already incorporate control properties in the design phase.

A core challenge when designing a special-purpose robot is the design synthesis. This includes how many joints should be used, what types of joints, the number of degrees of freedom (DOFs), and much more. Finding optimal designs constitutes a discrete optimization problem that exhibits a non-smooth objective function and does not offer gradient information.

In recent years, neural networks have been utilized to directly learn the inverse task of proposing a mechanism design given a specified trajectory. The first work to simultaneously optimize the mechanism design and the dimensional synthesis using neural networks is by Yim et al. [1]. In their work, a neural network predicts the parameters of a spring-block model given a preprocessed trajectory. As input features, they use the coefficients of a Fourier series which is applied to a centroid distance function of the trajectory. The same procedure is also used in the literature [2–6]. In general, data preprocessing (i.e., the way how the data is presented to a neural network) is a crucial aspect of accurate and robust neural network performance. While their work produced satisfactory results, there are several other potential procedures how to extract features from a trajectory. However, there exists no extensive evaluation comparing different preprocessing methods in the context of automated design assistants for mechanism design using neural networks.

In this work, we introduce various approaches how to extract features from a trajectory which can then be used as inputs for a neural network. To this end, the neural networks learn the inverse problem of mapping a (preprocessed) trajectory to mechanism parameters. We investigate the different procedures and evaluate their performance on a simple model problem. Additionally, different network sizes, the effect of normalization, and varying the feature vector length are investigated. The idea is to outline methods that exhibit decent and robust performance without the need for expert-guided hyperparameter tuning.

This work is structured as follows; In Section 2, the model problem is introduced alongside ways to parameterize different trajectory representations. Additionally, details about the neural network setup are given. In Section 3, the results of some analyses are presented. Section 4 discusses these results and Section 5 concludes the work.

2 | METHODS

In this section, the model problem is introduced that we use to evaluate the different preprocessing methods. We then detail various representations of a trajectory and ways to parameterize them to feed them into a neural network. Lastly, details about the neural network setup are presented.

2.1 | Model problem

For the comparison of preprocessing methods, we use the four-bar linkage as a model problem as shown in Figure 1 (left). We keep all parameters fixed and only vary the link length r_3 (blue) of the mechanism. The revolute joint O is actuated with a constant velocity and for every degree the location of the coupler point C is measured. Hence, 360 discrete points are sampled from the trajectory. Examples of trajectories for different values of r_3 can be seen in Figure 1 (right). The task of the neural networks is the inverse problem of predicting the value r_3 given a (preprocessed) trajectory of the coupler point C. To be able to feed a trajectory into a neural network we have to extract features from it. In the following, we will look at different representations of a trajectory and ways to parameterize them.

2.2 | Representation of trajectories

Subsequently, the different approaches examined in this paper for representing the coupler point trajectories are introduced. Later, when comparing all approaches, we use the same number of features n_f for all approaches, which specifies the length of the input vector to the neural networks.

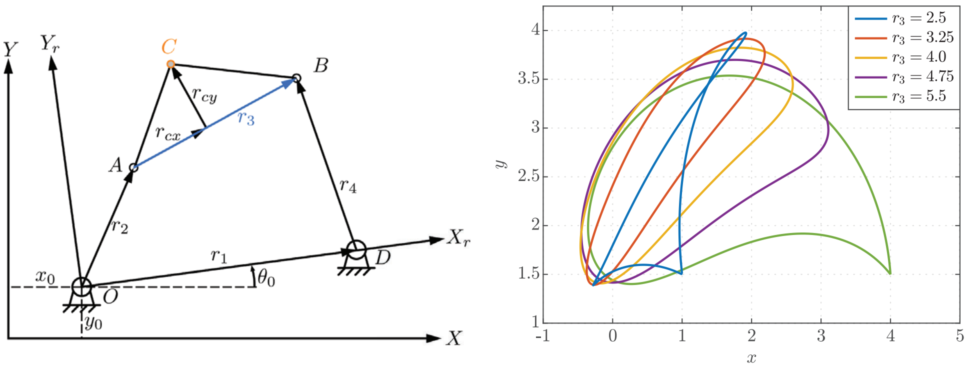

FIGURE 1 Left: Four-bar linkage where all parameters are fixed apart from r_3 (blue). Trajectories of the coupler point C are computed for one mechanism revolution. Right: Trajectories of the coupler point C for different values of r_3.

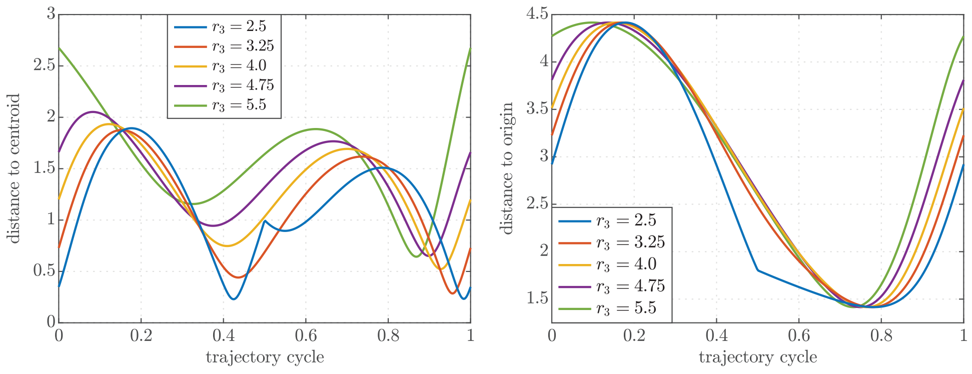

FIGURE 2 Distance functions for different values of r_3. Left: centroid distance function. Right: origin distance function.

2.2.1 | Coordinate function (*CF*)

One possibility is to represent each coordinate dimension as a separate function, which we will subsequently refer to as coordinate functions (*CFs*). Thus, we have a periodic function for $x(t)$ and for $y(t)$, where $t \in [0, 1]$ describes one mechanism revolution around joint O. However, when we parameterize the *CFs*, we may only use $n_f/2$ features to represent each *CF*.

2.2.2 | Centroid/origin distance functions (*CDF, ODF*)

The distance functions compute the distance of each two-dimensional point on the trajectory to a reference point. The periodic distance function is given by $d(t) = \sqrt{(x(t) - x_c)^2 + (y(t) - y_c)^2}$, where for the centroid distance function (*CDF*), we set x_c and y_c as the centroid of the contour, that is, the mean of the *CFs*. For the origin distance function (*ODF*), we set $x_c = y_c = 0$. A visualization for different values of r_3 can be seen in Figure 2. Both of these distance functions condense the two-dimensional paths into one-dimensional periodic functions. Hence, they essentially reduce the dimensionality of the trajectory information. Compared to *CF* we can use twice as many features to approximate the function. However, there is a loss of information due to the distance computation. In conclusion, there is a trade-off between the accuracy to which one can recover the original function and the information contained within that signal.

2.3 | Parameterization of trajectories

Next, we describe various approaches how features can be extracted from the trajectory representations such that they can be used as input vectors for a neural network.

2.3.1 | Sparse data points (*Coarse*)

This method extracts a set of points from the trajectory so that they are equally spaced in the rotation angle of one revolution. This implicitly contains information about the speed of the end effector as points further away from each other correspond to regions with higher velocity while points closer together indicate regions of slower movement. However, this approach does not include any further preprocessing as it uses raw data points from the trajectory.

2.3.2 | Fourier coefficients (*FC*)

A Fourier series can approximate periodic functions by summation of sine and cosine functions of increasing frequency. The series is given by $f(x) = a_0 + \sum_{n=1}^{\infty}(a_n \cos(\frac{2\pi nx}{L}) + b_n \sin(\frac{2\pi nx}{L}))$, where L is the period of the function. The fitted coefficients a_0, a_i, and b_i for $i \in \{1, \dots, n\}$ can then be used to describe the approximated function, where n is the n-th harmonic. We apply a Fourier series to the centroid distance function (*CDF-FC*), the origin distance function (*ODF-FC*), and the coordinate functions (*CF-FC*). The *CDF-FC* representation has previously been used in the literature [1–6] for the description of closed-loop trajectories.

2.3.3 | Elliptical Fourier descriptors (*EFD*)

This method is based on the elliptical Fourier analysis method, see [7]. It is an extension of Fourier series to two-dimensional closed contour lines. Hence, this method is closely related to *CF-FC* and the major differences are within the implementation details. Additionally, the elliptical Fourier descriptors (*EFD*) can be normalized in the calculation and adapted such that they parameterize the shape in a manner invariant to orientation, scaling, and translation. The implementation used in this work is provided in the literature [8].

2.3.4 | Polynomial regression coefficients (*Poly*)

We fit a polynomial with monomial basis of degree d to the data points from the (preprocessed) trajectory. The polynomial is given by $f(x) = \beta_0 + \sum_{i=0}^{d} \beta_i \, x^i$, where the regression coefficients β_i are obtained by least-squares optimization. The regression is applied on the centroid distance function (*CDF-Poly*), the origin distance function (*ODF-Poly*), and the coordinate functions (*CF-Poly*). In the case of the coordinate functions, one polynomial is fitted for each coordinate function with degree $d/2$.

2.4 | Training setup

For the data set, 201 trajectories are generated with $r_3 \in [2.5, 5.5]$ equally spaced in the interval, using a train-validation-test split of $[0.7, 0.2, 0.1]$. Each trajectory is preprocessed by one of the presented methods while varying the number of features (number of extracted points, number of Fourier coefficients, number of regression coefficients) that are fed into the neural network. For a Fourier series this equates to adding more sine and cosine terms and for polynomials it constitutes using monomials of higher order. We use fully connected feed-forward neural networks with ReLU activation in all hidden layers. Different network sizes are examined (e.g., one hidden layer, two hidden layers, and three hidden layers), where each hidden layer contains 40 neurons, and the analysis is conducted without normalization and with unit normalization. When applying unit normalization, each input feature is normalized individually to the $[-1, 1]$ range using the entire data set. The same procedure is applied to the output r_3 values. For each configuration (e.g., preprocessing method, number of input features, neural network size, normalization type), 20 independent neural networks are trained using randomly sampled training data sets. Overall, this amounts to training 12360 individual neural networks. The Adam optimizer is used with a learning rate of 0.001. The mean performance of the networks is reported in the following.

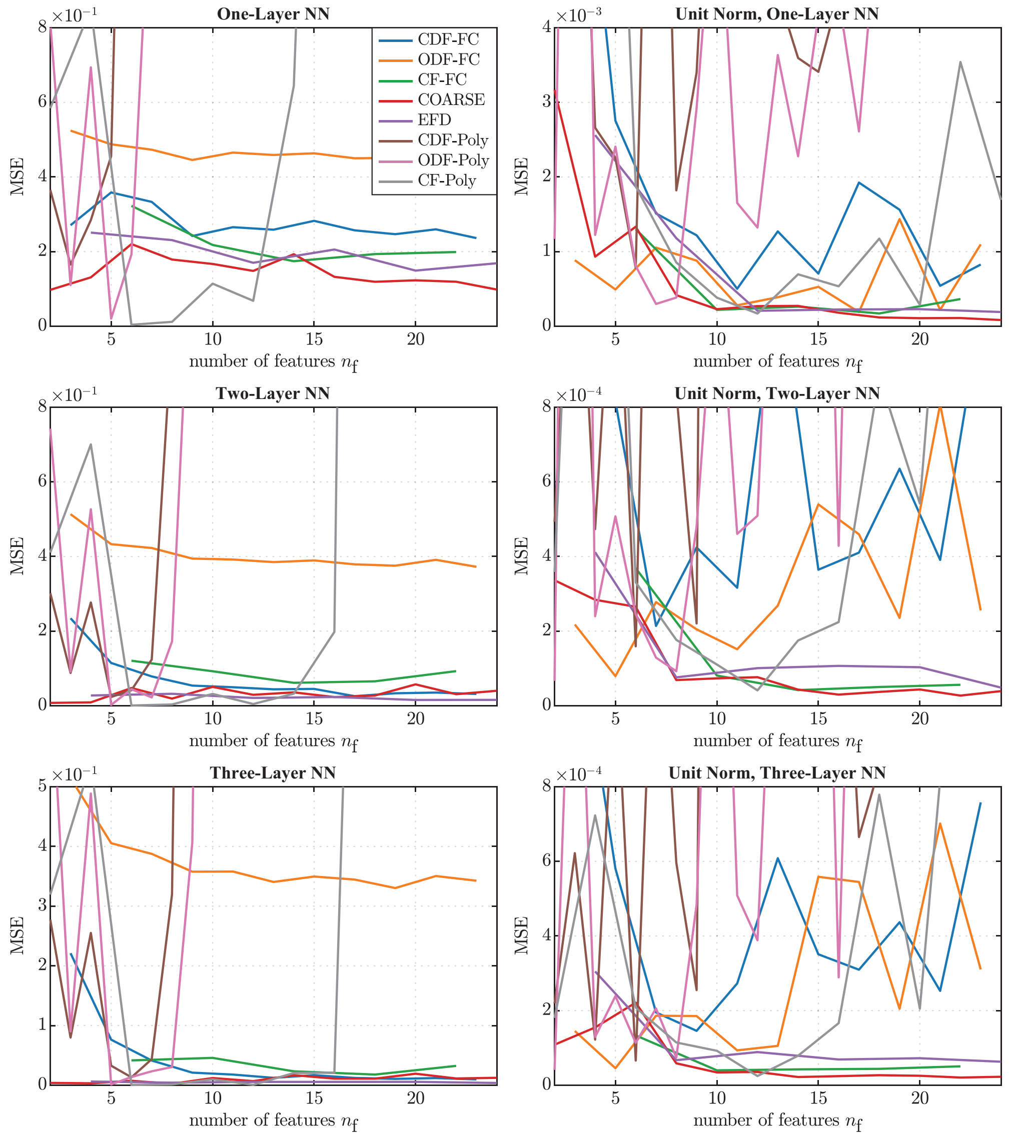

FIGURE 3 Mean squared errors for the prediction of r_3 values on unseen test data for different neural network sizes. Left column: neural networks without normalization. Right column: applying unit normalization on the inputs and outputs.

3 | RESULTS

For a fixed number of features n_f, one of the described preprocessing methods was run and 20 individual neural networks of the same size have been trained on randomly sampled data. These networks were tested on unseen test data and had to predict the r_3 value given preprocessed trajectories. Figure 3 shows the mean squared error (MSE) between the r_3 prediction and the ground truth value, averaged across the 20 trials. The left column of the figure refers to using neural networks with one, two, and three hidden layers. The plots in the right column show the same setup but applying unit normalization on the inputs and outputs. Notice that the preprocessing methods which used polynomials did not work robustly across every evaluated feature vector length (i.e., the brown, pink and grey lines). The plots also show that unit normalization has a significant impact on the prediction accuracy of the networks. It improves the prediction error by almost two orders of magnitude as can be seen on the vertical scale. We also see that no method consistently outperforms all others. However, *Coarse*, *EFD*, and *CF-FC* perform robustly across all feature lengths and neural network configurations. *CDF-FC* performs robustly over all input vector sizes but when applying unit normalization it does not benefit as much as the previously mentioned methods. The same plots have been generated for the standard deviation across the 20 trials for each configuration. They show a similar behaviour and, thus, are omitted here.

4 | DISCUSSION

In general, training neural networks exhibits a large number of different hyperparameters such as network architecture, optimizer choice, learning rate, initialization strategy, activation function and more. For this evaluation, no hyperparameters have been extensively tuned. This means that for all methods there is possibly margin for improvement. Hence, the error plots should not be taken as absolute performance measures but only as an indication of the robustness and relative performance of the preprocessing methods. Generally, for the usage in automated design assistants, preprocessing methods with decent out-of-the-box performance should be preferred over methods where the performance is highly sensitive to the hyperparameters. After all, expert-guided hyperparameter tuning conflicts with the idea of automated design assistants.

In the plots of Figure 3, the number of features n_f are varied along the x-axis. This corresponds to varying the length of the input vectors and as such the number of features used to describe the trajectories. A suitable prepocessing methods should exhibit low MSEs independent of the specific choice of n_f. Hence, preprocessing methods that consistently achieve a low MSE across the entire x-axis in all plots can be regarded as robust. These methods are preferred in the use of an automated design assistant.

The *Coarse* method performed robustly across different numbers of extracted points and led to good results. This is surprising since it does not compress information from the entire trajectory. Especially for very few extracted points the complete trajectory information is rather scarce. One possible explanation could be that, since we only vary one parameter, the corresponding trajectories change rather smoothly as can be seen in Figure 1 (right). Thus, we have a bijective mapping from extracted points to the value of r_3, which seems to give a very clear learning signal. However, this only holds due to the simplicity of the problem. For more complex problems like varying multiple parameters at once, different trajectories could have very similar points extracted. In these instances, the neural networks could fail to recover the ground truth parameters and the performance of *Coarse* could worsen.

On the other hand, *EFD* seems to be particularly suitable for representing closed-loop trajectories due to the fact that the coefficients are invariant to translation, rotation, and dilation of the contour. This is a useful property as these variations do not correspond to structurally different mechanisms but only to a different mounting and sizing of the same mechanism. Hence, it makes the inverse learning task easier as it removes unnecessary complexity.

The evaluation showed that the input-output normalization to the $[-1, 1]$ range had significant impact on the prediction accuracy. The overall MSE was improved by almost two orders of magnitude. However, to be able to normalize the inputs and outputs, the minimal and maximal values have to be computed using the entire preprocessed data set. This might not always be possible, especially in scenarios when the test set is not previously known. In these cases, lower and upper boundaries can be estimated and used for normalization. Another approach would be to only use the training set for normalization meaning that the networks would potentially have to extrapolate on some of the unseen test data.

It is important to note that the studied model problem is rather simple since it consists of just predicting a single mechanism parameter given a (preprocessed) trajectory. Future work will investigate how the preprocessing methods compare when being applied to more complex problems. This can be the prediction of multiple parameters at once for a given mechanism or predicting coefficients of a meta model that can capture different mechanism designs simultaneously. In these instances, a good representation of the trajectory will be crucial as the design space is significantly larger and data points may only be sparsely distributed. It will be interesting to see if the difference between the preprocessing methods becomes more apparent.

5 | CONCLUSION

In this work, an evaluation of different preprocessing methods for the inverse learning task of predicting a mechanism parameter given data from its trajectory is presented. Different representations and parameterizations for the trajectories are shown. Furthermore, different neural network sizes and the influence of normalization are examined. We showed that no method signficantly outperforms all other methods across all configurations. The *CDF-FC* method, which has been used widely in the literature, performs robustly across all feature lengths but methods like *EFD* and *Coarse* robustly lead to better results. The analysis indicated that polynomials are not suitable to parameterize trajectories in the context of an automated design assistant as they are sensible to the choice of the regression order, where the optimal regression order for a given trajectory is generally not known a priori. Normalizing the inputs and outputs to the $[-1, 1]$ range significantly improves the prediction accuracy on the test data by almost two orders of magnitude.

ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grants 501890093 (SPP 2353, Coordination) and 501840485 (SPP 2353, Data- and Response-Surface-Driven Design Assistant for Controlled Flexible Multibody Systems). This support is highly appreciated.

Open access funding enabled and organized by Projekt DEAL.

ORCID

Benedict Röder https://orcid.org/0009-0006-0981-4395
Henrik Ebel https://orcid.org/0000-0002-2632-6960
Peter Eberhard https://orcid.org/0000-0003-1809-4407

REFERENCES

1. Yim, N. H., Lee, J., Kim, J., & Kim, Y. Y. (2021). Big data approach for the simultaneous determination of the topology and end-effector location of a planar linkage mechanism. *Mechanism and Machine Theory*, *163*, 104375.
2. Ullah, I., & Kota, S. (1997). Optimal synthesis of mechanisms for path generation using Fourier descriptors and global search methods. *Journal of Mechanical Design*, *119*(4), 504–510.
3. Buśkiewicz, J., Starosta, R., & Walczak, T. (2009). On the application of the curve curvature in path synthesis. *Mechanism and Machine Theory*, *44*(6), 1223–1239.
4. Buśkiewicz, J. (2010). Use of shape invariants in optimal synthesis of geared five-bar linkage. *Mechanism and Machine Theory*, *45*(2), 273–290.
5. Wu, J., Ge, Q. J., Gao, F., & Guo, W. Z. (2011). On the extension of a Fourier descriptor based method for planar four-bar linkage synthesis for generation of open and closed paths. *Journal of Mechanisms and Robotics*, *3*(3), 031002.
6. Li, X., Wei, S., Liao, Q., & Zhang, Y. (2020). A novel analytical method for four-bar path generation synthesis based on Fourier series. *Mechanism and Machine Theory*, *144*, 103671.
7. Kuhl, F. P., & Giardina, C. R. (1982). Elliptic Fourier features of a closed contour. *Computer Graphics and Image Processing*, *18*(3), 236–258.
8. Grieve, S. W. D. (2017). spatial-efd: A spatial-aware implementation of elliptical Fourier analysis. *Journal of Open Source Software*, *2*(11), 189.

How to cite this article: Röder, B., Ebel, H., & Eberhard, P. (2023). Towards intelligent design assistants for planar multibody mechanisms. *Proceedings in Applied Mathematics and Mechanics*, *23,* e202300060. https://doi.org/10.1002/pamm.202300060