Groensfelder *et al. Adv. Model. and Simul. in Eng. Sci.(2020)7:26*
https://doi.org/10.1186/s40323-020-00163-4

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Application of machine learning procedures for mechanical system modelling: capabilities and caveats to prediction-accuracy

Thomas Groensfelder, Fabian Giebeler, Marco Geupel, David Schneider and Rebecca Jaeger

*Correspondence:
thomas.groensfelder@h-da.de
Dep. of Mechanical and Plastics Engineering, UAS Darmstadt (Germany), Schoefferstr. 3, 64295 Darmstadt, Germany

Abstract

This article presents an investigation about prediction accuracy of multi-parametric models derived from numerical data. Three different mechanical test-cases are used for the generation of the numerical data. From this data, models are derived for the prediction of characteristic variation to arbitrary changes of the input parameters. Different modeling approaches are evaluated regarding their prediction accuracy. Polynomial matrix equations are compared to regression models and neural network models provided by Machine-Learning toolboxes. Similarities and differences of the models are worked out. An exponential matrix-equation-model is proposed to increase accuracy for certain applications. Influences and their causes to the prediction accuracy for the model predictions are evaluated. From this minimum requirements for deriving valuable models are defined. Leading to a comparison of the modelling approaches in relation to physical plausibility and model efficiency. Where efficiency is related to the effort for data creation and training-procedure. For one of the sample cases, a prediction-model is applied to demonstrate the model application and capabilities. The model equation is used to calculate the value of a penalty function in a multi-input/multi-output optimization task. As outcome of the optimization, four natural frequencies are fitted to measured values by updating material parameters. For all other cases sensitivity-studies including verification to numerical results are conducted.

Keywords: System modelling, Machine Learning, Numerical experiment, Parametric Finite Element Analysis, Prediction accuracy, Sensitivity study, Model optimization

Introduction and background

Today's engineering tasks are usually part of multi-parameter, -level and -physics processes. Numerical models and simulations are often applied in development processes. Common to all model-based approaches is a certain calculation procedure that contains one or more variables which are used to create the link between the model and a real counterpart. A drawback of these numerical simulations is also commonly known. They

are only able to deliver "one result" for a given input. A simulation outcome using average material parameters for a nominal geometry represents therefore an *average result*.

Model optimization updates material parameters of an available FE-model to fit better to experimental data, see [1]. But getting closer to one measurement is also not sufficient looking at a series of products. Manufacturing requires tolerances, which lead to inevitable variations of the final geometry. These variations result in a scatter of characteristic properties of the product. They are not deterministic—therefore the resulting scatter also is uncertain. The authors of [2] describe a procedure to predict the frequency scatter for Low-Pressure-Steam-Blades under different operating conditions. The task here is a prediction of a max. deviation from nominal data. Which also requires a model-based optimization method. A third application of models is product optimization. Optimization routines try to find an optimum (nominal) solution. All applications are dealing small changes in input-values to reach a local or global minimum. Direct optimization on numerical simulations is possible, but it may also be expensive in computational effort. To reduce effort and speed up development iterations, model based optimization may be used. Another demand for efficient but accurate predicting models are whole-system models for automation purposes. Full detailed models are not applicable in this scope as numerical effort contradicts real time requirements.

This variety of applications requires models to be established, which are capable to predict the characteristics of the (expensive) full featured-models with reduced effort. As the deviations from the nominal value are relatively small compared to the characteristic values, high accuracy and still physical valid prediction methods are required.

From a system modeling standpoint different approaches are available. This study presents several methods, highlights and discusses their capabilities, efficiency and influences on accuracy.

System modeling

Usually system modeling is based on a presumed structure represented by a certain **model equation** with a **number of model parameters** (n_sys), e.g. [3]. Similar to the material parameters of an FE-model, these model parameters are unknown. Prior to transferring the model to a productive state, they have to be defined appropriately. Engineers often call this approach *grey box* model [4].

As the exact physical functions are most probably not known analytically, they are fitted by a polynomial sum, see [5]. Such a polynomial is written

$$f(x) = \sum_{i=0}^{d} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots \tag{1}$$

as a sum of infinite number of elements. The variable x represents a single system parameter of the system to be modelled. In a real application the sum often does not contain a high number of degrees d. A plot of three generic functions is given in Fig. 1.

To identify the **model parameters** (a_i), experimental data is usually fitted by a *Least Square Fit method (LSF)*, e.g. [6]. With a certain number of measurements, the function values $f(x_{\text{measured,i}})$ at a location $x_{\text{meas,i}}$ are known. For a good fitting, a higher number of measurements than the actual number of model parameters n_{poly} should be used.

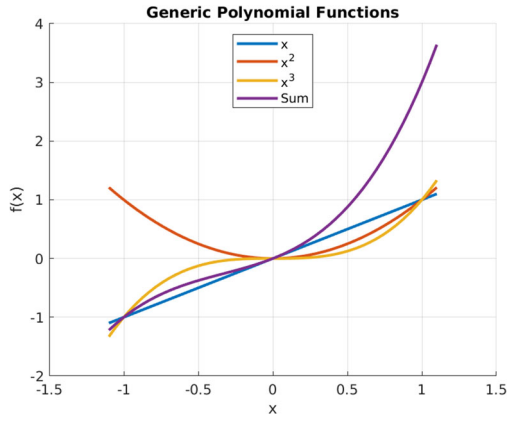

Fig. 1 Scalar function. Plot of generic polynomial function

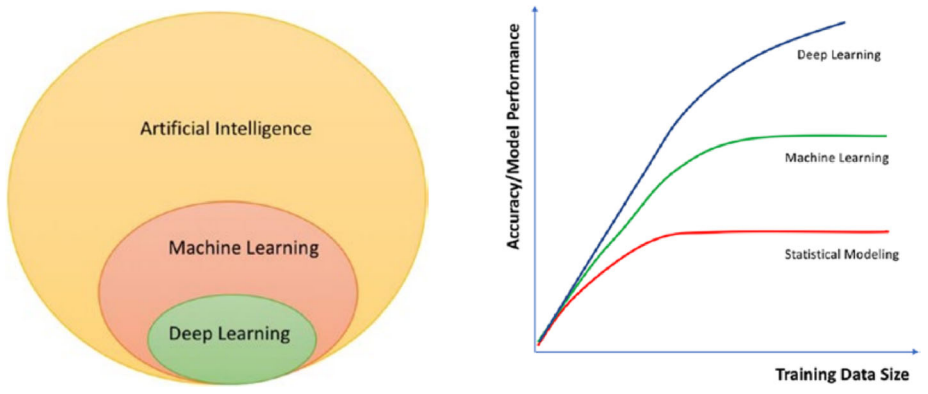

Fig. 2 Artificial Intelligence. Artificial Intelligence, Machine Learning and Deep Learning, see [7]

Machine learning

Artificial Intelligence (AI), *Machine Learning (ML)* and *Deep Learning (DL)* are frequently used words today. Their relation is shown on the left side of Fig. 2 from [7].

AI basically deals with data handling and analysis. The different levels of the models require different amounts of data for application. DL, using neural networks with many layers, requires most data, ML requires less, see Fig. 2 on the right.

ML can be subdivided in *Supervised Learning* respectively *Unsupervised Learning*.

In the scope of this study supervised learning techniques were applied. An outcome of an ML-procedure is a tool which can be applied for making decisions or predictions. At this point we are close to system modeling. Both methods are using an algorithm for making predictions and both use empirical data to define model parameters. In contrast to this modeling approach, *Artificial Neural Networks (ANN)* do not prescribe a certain structure, they are closer to a "black box"-model type.

Methods, procedures and models

This section starts with a description of the applied methods and calculation procedures used in the present study. Then modelling techniques from classical engineering and ML packages are described briefly and compared regarding their capabilities. Finally, three

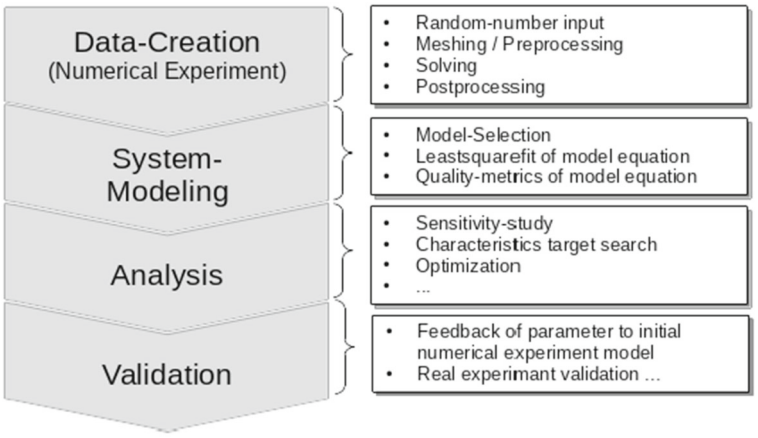

Fig. 3 Calculation procedure. Schematic sketch of calculation procedure, similar to [2]

sample problems are introduced, which are used to demonstrate and discuss influences on model accuracy.

Automated calculation procedure

The automated calculation procedure applied in this study is very similar to the procedure described in [2]. The basic idea is to use data from a numerical experiment during development to derive a simplified model. This model is then used for system optimization or predicting deviations of certain system characteristics. The focus for the newly developed software package is being versatile and not bound on a specific application, underlying software or design target. This is demonstrated here using different FE-software and meshing procedures.

The procedure is divided into four steps

1 Data creation
2 System Modeling
3 Analysis of system characteristics
4 Validation of models and results

which are illustrated in Fig. 3.

The data creation is done by using a *Numerical Experiment*. Required for this is a parametric numerical model, which may either be an FE- or a CFD-analysis. The parametric system includes n_{sys} different **system parameters**. For example geometrical dimensions, material parameter or layer-directions for fibre reinforced plastics. During preprocessing of the numerical experiment, the model is modified by random input data within reasonably defined boundaries. This is leading to n_{calc} different calculation runs for the underlying model. After solving, relevant characteristic data (for example natural frequencies or stress values) is extracted and gathered in adequate data-files.

With the second step the model identification is performed. This is done by combining and least-square-fitting the random-input- in conjunction with the characteristics-output-data. Details and results of feasible accuracy is the main topic of this study and will be described later.

In the third step, the identified system is used to do further analyses. The main advantage to the underlying numerical simulation is a significant reduction of computational effort. Using the simplified model therefore leads to quick and efficient calculations. Optimization, model updating of the numerical simulation to match characteristic-targets or prediction of spread for characteristic values is feasible. The latter application represents the focus of [2].

For the later shown sample problems, a sensitivity study and max. deviation search will be performed.

As an optimization result is highly dependent on the model-accuracy, this study puts a focus on comparing model accuracy of the models discussed. To validate the prediction results, the fourth and final step is done. The optimized system parameters n_{sys} are fed back to the FEA and calculation results are compared.

Modeling approaches

The focus of this study is set on the comparison of different modelling approaches. To understand the differences and similarities, common models are described briefly and a new model equation is proposed.

Vectorial model equation

The polynomial Eq. (1) is only for adequate for fitting a function of a scalar value x. As the parametric model has n_{sys} input parameters, the scalar value x evolves to a vector $\vec{x}$ consisting of n_{sys} entries. With that (1) rewrites to

$$f(\vec{x}) = C_0 + \underline{A} \cdot \vec{x} + \vec{x}^T \underline{B} \cdot \vec{x} + \vec{x}^T \left(\underline{C} \cdot \vec{x}\right) \cdot \vec{x} \tag{2}$$

a vectorial notation. For simpler reading we will only use up to third order terms of the sum. Newly introduced terms are a

- scalar C_0
- row shaped vector $\underline{A}$ with n_{sys} entries
- $n_{\text{sys}} \times n_{\text{sys}}$ square-shaped matrix $\underline{B}$
- $n_{\text{sys}} \times n_{\text{sys}} \times n_{\text{sys}}$ cubic matrix $\underline{C}$

All components of the new variables C_0, $\underline{A}$, $\underline{B}$, $\underline{C}$ are unknown and free parameters of the *model parameters*. For a three-parameter system $n_{\text{sys}} = 3$ this would result in $1 + 3 + 3^2 + 3^3 = 40$ free parameter.

This number can be reduced significantly. For our sample three-parameter problem, the second term of (2) can be rewritten as

$$\begin{aligned} \vec{x}^T \underline{B} \cdot \vec{x} = {} & B_{1,1}\, {x_1}^2 + B_{2,2}\, {x_2}^2 + B_{3,3}\, {x_3}^2 \\ & + \left(B_{1,2} + B_{2,1}\right) x_1 x_2 + \left(B_{1,3} + B_{3,1}\right) x_1 x_3 + \left(B_{2,3} + B_{3,2}\right) x_2 x_3 \end{aligned} \tag{3}$$

containing sums of $\left(B_{i,j} + B_{j,i}\right)$ for the related mixed $x_i \cdot x_j$-products. With this the matrix

$$\underline{B} = \begin{bmatrix} B_{1,1} & B_{1,2} & B_{1,3} \\ 0 & B_{2,2} & B_{2,3} \\ 0 & 0 & B_{3,3} \end{bmatrix} \tag{4}$$

Table 1 Number of model parameters for 5 and 6 system-parameter-problem

Polynomial-degree	Number of model parameters n_{poly} for	
	$n_{sys} = 5$	$n_{sys} = 6$
1	6	7
2	21	28
3	56	84
4	126	210
5	252	462
6	462	924

can be reduced to a upper (respectively lower) tri-diagonal matrix. Having

$$n_B = \frac{1}{2}\left(n_{sys}^2 + n_{sys}\right) \tag{5}$$

parameters in the matrix. With a similar justification the number of parameters

$$n_C = \frac{n_{sys}}{6}\left(n_{sys}^2 + 3n_{sys} + 2\right) \tag{6}$$

for the cubic matrix $\underline{C}$ can be reduced. Generally, the number of parameters in (2) can be calculated to the number of

$$n_{\text{poly}} = \frac{(n_{sys} + d)!}{n_{sys}! \cdot d!} \tag{7}$$

when using d as polynomial degree of the model equation, see [8]. As one can see, this will generate a very high number of model parameters n_{poly}, when the number of numerical system parameters n_{sys} is increased. Table 1 lists an overview on the polynomial model parameters (7) for a five and six variables system.

Meaning that the polynomial order ought to be as low as possible for efficiency reasons. On the other hand, polynomial functions of low order are restricted in the capabilities to reproduce steep gradients.[1] Furthermore, does first order modeling of (2) not seem promising in the desired context of this study, as it is not capable of reproducing cross-interactions of $\vec{x}$-components.

An exponential-function approach

$$g(\vec{x}) = \exp\left(\underline{A}_{\text{exp}} \cdot \vec{x}\right) + \exp\left(\vec{x}^T \underline{B}_{\text{exp}} \cdot \vec{x}\right) + \exp\left(\vec{x}^T \left(\underline{C}_{\text{exp}} \cdot \vec{x}\right) \cdot \vec{x}\right) \tag{8}$$

is proposed here to improve the steep gradient capabilities. The complete model equation

$$f_{\text{model}}(\vec{x}) = f(\vec{x}) + g(\vec{x}) \tag{9}$$

simply sums up (2) and (8). The final number of free parameter in (9) can be calculated by

$$n_{\text{total}} = 2 \cdot n_{\text{poly}} - 1 \tag{10}$$

bypassing the factorial increase of model parameters.

[1] Which is one of the reasons why meshes in FE-Analyses have to be refined at high stress regions. FE-software usually uses second order meshes as efficient tradeoff.

We will call these terms of the exponential equation to be of first to third order, following the definition of the polynomial nomenclature. The currently realized internal software implementation offers arbitrary combinations of first to third order polynomial-equation (2) and none to third order exponential-equation (8) system identification. The identification procedure facilitates SciPy, [6], *leastsq()*-function amongst others and offers CPU-parallelization also for the exponential equation.

Machine-Learning algorithms

From an engineering standpoint it is important to understand that terminologies in ML context are somewhat different from common engineering speech. An engineer would select for example (2) as model equation. This equation would be used to calculate a result, which the engineer is interested in.

ML and its terminology are centered on the data it is based on. In ML-context calculating a result according to the certain model data is called *predicting*. ML-Toolboxes offer a variety of different models to be used. For making predictions the models are called Regressor, as they do regressions to the model data. Finally, the engineering identification of model parameters (e.g. $\underline{A}$, $\underline{B}$ etc.) is called *training* of the regressor. One of the first regressor algorithms introduced in [8] is called *LinearRegressor*, see also [9]. At a first glance, it seems like LinearRegressor is only capable of performing linear, like using only $\underline{A}$ in (2). By expansion of the input vector

$$\vec{q} = \left[1, x_1, x_2, x_3, \ldots, x_1x_2, x_1x_3, x_2x_3, \ldots, x_1^2, x_2^2, x_3^2, \ldots\right]^T \tag{11}$$

containing also the different cross-combinations of $\vec{x}$ the model equation can be rewritten to

$$h(\vec{x}) = \underline{R}_{\text{lin}} \cdot \vec{q} \tag{12}$$

where $\underline{R}_{\text{lin}}$ represents a column vector of linear parameters for the model.

Using this method, a linear system of equations is derived, and the LinearRegressor-function can be utilized for polynomial regression, too. The number of entries in $\underline{R}_{\text{lin}}$ is identical to (7). In practical applications the ML-Toolbox provides a preprocessing-function that automatically transforms each entry in the dataset. A-today minor- drawback is, that datasets using a higher number of input parameters n_{nsys} and calculations n_{calc} dissipate a lot of RAM. This *PolynomialRegression*-called approach is not only identical in theory, the actual implementation in Python also uses a similar *leastsq()*-function, see [8], for training. High polynomial order model selection is quite easy with the provided function. Nevertheless, the number of free parameters in $\underline{R}_{\text{lin}}$ will grow extremely fast. Thus, the reproduction of steep gradients still is limited for the PolynomialRegression.

Unfortunately, switching to (8) in conjunction with *LinearRegressor* is not feasible as rewriting it to a linear notation is not possible. Only a few irrelevant subsets with a single term would deliver linear models.

Several other functions, for example *Ridge-Regression*, are provided by [9], with an excellent introduction to application given in [8]. Ridge-Regression, and others like Lasso-Regression, represent a class of linear regression algorithms. In combination with the data preprocessing by (11) they become capable of polynomial regression. This class of models

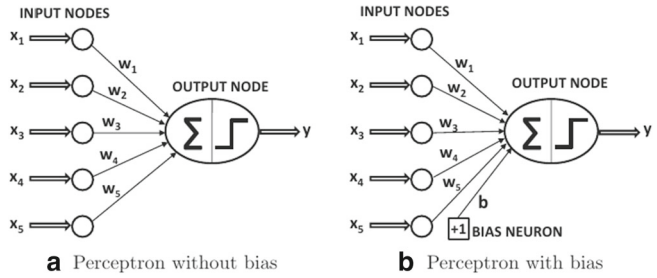

Fig. 4 Single perceptron, from [12] with/without bias and variables

is applied in order to reduce model overfitting by introduction of an additional model parameter. Another class of functions is available with *Support Vector Machines (SVM)* introducing new models. SVM were also tested in the scope of this study. As they also did not perform better than the discussed models and a main reason for overfitting lies in the polynomial preprocessing, the whole class should be represented by the Ridge-Regression algorithm only.

All these functions have in common that they are introducing another model with an additional model-equation, having at least one or two additional model parameters. These model parameters are unknown and must be identified by the user. This identification step of the additional parameters is typically referred to as *model-tuning*. There are no strict rules given for tuning of these additional model parameters. Some helper functions (i. e. *grid_search()*) are provided, which automatize the calculations. In general, ML-functions cannot be rated in *best for all*-categories. It is a known behaviour, that a well performing method for one application fails on another.

A completely different approach to system modelling is taken by *Neural Networks (NN)*. Based on the idea of "learning like a brain", they come closer to a "black box"-method. Theory about NNs can be found in literature, e.g. [8,10], or in documentation for software packages, e.g. [9,11].

Basically NN build models that connect input with output, too. The smallest entity a NN is built of is called *perceptron*, see Fig. 4 from [12]. Each perceptron has as many *weigh-factors* as inputs, marked as w_i in Fig. 4. If required, an *bias*-value can be added. All of these values represent unknown parameters for that perceptron. Finally, each perceptron got an *activation function* which has to be chosen by the user and is therefore considered as known.

In a *Multi-Layer-Network* , these perceptrons are ordered in *hidden layers*, see Fig. 5. In a *feed-forward network*, each perceptron can be connected to every perceptron of the next layer. A NN is completed by an *Input-* and *Output-Layer* to fit the demands of system resp. output parameter. The unknown weigh-factors are identified during a *training* step. This design gives the opportunity to create individual interactions between input and output. For example: Starting from the input layer of Fig. 5 with x_1 and x_2. Putting values to the input weigh-factors $w_1 \neq 0$ and $w_2 \neq 0$ of the first perceptron would lead to

$$w_{inp} = w_1 \cdot x_1 + w_2 \cdot x_2 \tag{13}$$

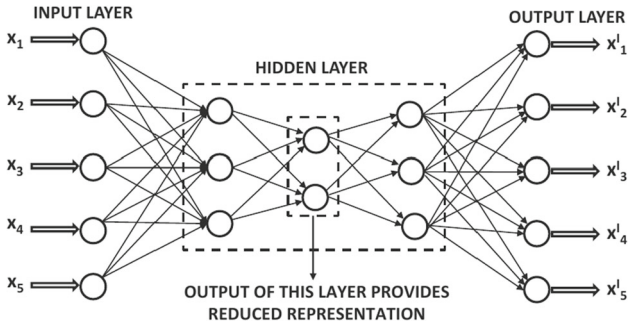

Fig. 5 Multi-Layer-Network, from [12] with input. 3 hidden and output layers

as input to the selected activation function. All weigh-factors and bias-values sum up to a total number n_{total} of trainable variables of the network.

As the weigh-factors also represent the connections between the perceptrons, a NN can be evaluated as a step closer to a "black box" approach. Several training methods for NN are available. A very important detail for training is that the weigh-factors has to be initialized by random-numbers. This is required to enable the applied training algorithms to find a solution.

Model parameter identification and requirements

The described model equations contain n_{poly} resp. n_{total} unknown model parameters. A linear equation requires at least this number of datasets to be prepared for solving. Such a direct solution is usually replaced by a *Least Square Fit (LSF)* procedure to reduce influence of measurement errors or other random effects.

An LSF builds up an over-determined system using more than required n_{total} linear equations. More details on LSF should not discussed here.

Relevant for this study is an *Over-Determination-Factor*

$$odf = \frac{n_{\text{ident}}}{n_{\text{total}}} \tag{14}$$

which relates the numbers of required and actually used n_{ident} datasets for identification. This factor was used to evaluate influences on the accuracy of the model-predictions.

The mathematical requirement of $odf \geq 1$ is not necessarily checked, if you are using external functions. All externally supplied ML implementations ([9,11]) ran also under-determined training without warning. As a result, an effect called *Overfitting* is commonly known in ML application. We will discuss this later using example problems. The total dataset of n_{calc} numerical calculations was therefore split into

- identification-set with n_{ident}
- validation-set consisting of n_{vali}

datasets. An odf-value close to 1 is also not sufficient for identification of physically accurate models, due to noise in the experimental data. Results are shown and and recommendations are derived below.

Quality metrics

The quality of the model output/regression has to be measured by certain quality metrics. Required for calculation of a meaningful metric is a dataset containing a

- number of n_{samples} having measured (=true) values $y_{\text{true},i}$ with
- corresponding number of input datasets $\vec{x}_i$

for each sample. These input-vectors are used for a calculation/prediction of $y_{\text{pred},i}$ output-values. The data used for metric-calculation should, at least partly, deviate from the data used for identification/training, see [8].

First the absolute deviation

$$\Delta y_i = y_{\text{true,i}} - y_{\text{pred,i}} \tag{15}$$

from actual to predicted values can be evaluated. Alternatively written as relative

$$r_i = \frac{\Delta y_i}{y_{\text{true,i}}} \tag{16}$$

deviation. For the relative deviation vector $\vec{r}$ a

$$r_{\text{min}} = \min(\vec{r}) \tag{17}$$

$$r_{\text{max}} = \max(\vec{r}) \tag{18}$$

max. and min deviation can be evaluated. And from this the

$$\Delta_{\text{rel}} = r_{\text{max}} - r_{\text{min}} \tag{19}$$

maximum relative spread (given in % in this study) within the evaluated data.

Very common—and also by ML-Toolboxes provided—is the

$$R^2 = 1 - \frac{u}{v} \tag{20}$$

$$u = \sum_{i=1}^{n_{\text{samples}}} (\Delta y_i)^2 \tag{21}$$

$$v = \sum_{i=1}^{n_{\text{samples}}} \left(y_{\text{true,i}} - \bar{y}_{\text{true}}\right)^2 \tag{22}$$

Root-Mean-Square-Error (RMSE) often only referred as *Score*, see [9]. A perfect fit would deliver a value of $R^2 = 1$, very bad values are negative.

Sample cases

The present study uses sample cases to elaborate the specific qualities of the described modeling approaches. This section describes the basic FE-modeling used for the numerical experiment in the procedure of Fig. 3.

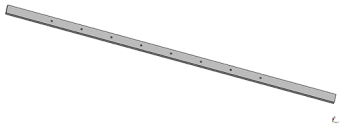

Fig. 6 Beam with holes. Sample beam with 8 holes for natural frequency analysis, geometry created according to [1]

Table 2 Technical data for sample beam

Description	Value (mm)
Length	1100
Width	29.2
Height	9.6
Hole diameter	5.8
Hole distance	100

Table 3 Natural frequencies from [1]

Mode	Measured Hz	Initial calculation Hz
1	41.5	42.3
2	114.5	117.0
3	224.5	227.3
4	371.6	376.9

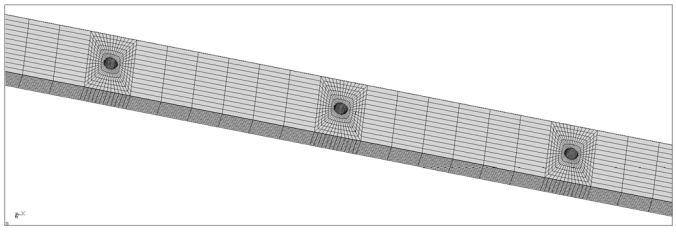

Fig. 7 Meshing at holes. Mesh at holes after mesh independence study

Natural frequencies of a beam with holes

The first case was introduced by Marwala in the 1990's and is constantly referred to in his work, e.g. [1]. The geometry is shown in Fig. 6. The beam is made of aluminum and has a constant thickness, height and cross section. There are eight holes equally spaced on the centerline of the beam. Due to an intended asymmetry, the bearing out at one end of the beam is larger than the other (Table 2). Four natural frequencies were measured by the author of [1] under free suspension condition and were also calculated using a simplified beam model. The frequencies for transversal bending modes are listed in Table 3. These modes were measured by the applied equipment. The *Initial Frequency*-column was calculated using an Elastic-Modulus of $E = 7.0 \cdot 10^{10}\,\mathrm{N/m^2}$.

For the present study, a parametric FE-model reproducing the given geometry was created. As solver the OpenSource FE-Program *Calculix* was selected, [13]. A mesh study delivering a mesh independent solution was conducted. Figure 7 displays a magnification of the final mesh.

Table 4 Comparison of FE-calculated natural frequencies and [1]

Frequency	FE-Analysis Calculix Hz	Marwala measured Hz	Marwala model Hz
F_1	41.14	41.5	42.3
F_2	113.54	114.5	117.0
F_3	126.22	N/A	N/A
F_4	222.75	224.5	227.3
F_5	346.58	N/A	N/A
F_6	368.31	371.6	376.9

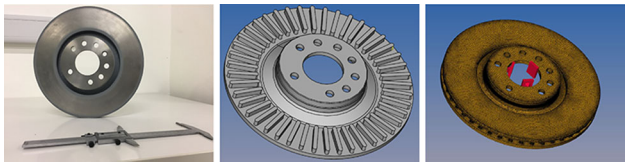

Fig. 8 Sample brake disk photo, geometry (Cut) and meshing of sample brake disk

In order to get accurate FE-results it is kept rather dense in the vicinity of the holes. The FE-results were verified against CAD-geometry simulation using an unstructured mesh.

Table 4 shows a comparison of the frequencies.

All available FE-calculated values are below the measurement data. In contrast to this, lie all frequencies reported by [1] above the actually measured ones. Lines marked with N/A contain lateral bending modes which were not detectable by the applied sensors, these were only calculated by the FE-model in this study.

Five parameters were selected

- Elastic Modulus E,
- Hole diameter d_{hole},
- Hole distance Δ_{hole},
- Total length of the beam l_{beam},
- Width of the beam cross section w_{beam}

to be modified in the numerical experiment of the current study.

Natural frequencies of a brakedisk

Second case is another analysis of natural frequencies, see Fig. 8. The Geometry is a brakedisk taken from a common automobile application. All geometric data was measured from the actual part and a model was created in an OpenSource-CAD software, [14]. The CAD-software modelling is fully parametric in all dimensions and all dimensions can be manipulated via a Python programming interface.

The geometry was manipulated with random input data on five parameters

- thickness of material at contact surfaces (parameter #1 and #5)
- width and length of "blades" in ventilation ducts (parameter #2 and #3)
- angle of inner ventilation ducts (parameter #4)

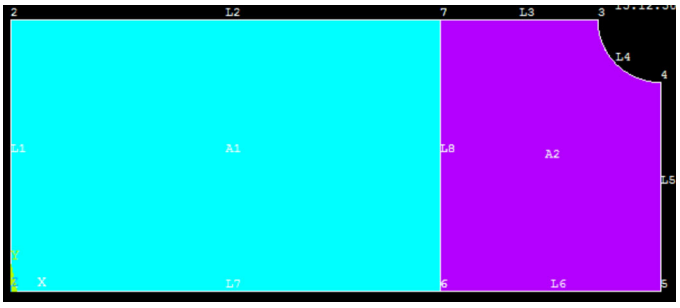

Fig. 9 Sample geometry notched plate. ($\frac{1}{4}$-Model with symmetry). Model and mesh of the notched plate sample problem

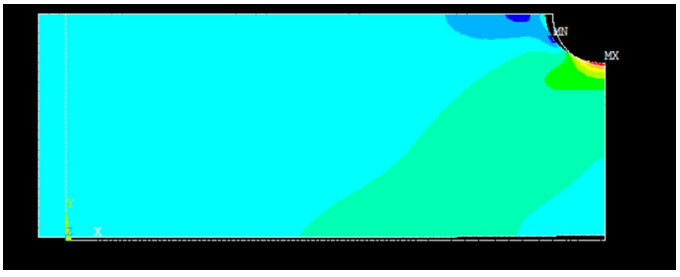

Fig. 10 Result—contour plot. Exemplary result of a notched geometry

and then meshed within the CAD software. The mesh was exported and the subsequent modal analysis delivered six natural frequencies for each geometry. For this FEA the geometry was fixed at the central cylinder, see red markers in Fig. 8 on the right.

This case and FE-workflow were chosen for several reasons. The geometry is more complex than in the first sample. Thus, the influence and potential interactions of the parameters to the natural frequencies are expected to be more complex, too. Another interest was to check out the capabilities and performance of the implemented toolchain, especially the tetrahedral meshing.

Stress calculation in a notched plate

Third sample is a 2D-problem. A stress calculation of a notched plate is performed, assuming a linear-elastic material behavior (Fig. 9). Due to symmetry in the chosen setup just a quarter section of the geometry was modelled in the FEA. The calculation result is well known. A contour plot of the stress distribution is shown in Fig. 10. An initial mesh study was conducted for definition of a reliable unstructured mesh. It was taken care to set a FE-node at the highest loaded location of the notch. The automated calculations for the numerical experiment were performed using a commercial FEA-software, [15].

All models for all test cases were trained on identical data. For this test case, a restriction was introduced to the numerical results. The overall dataset was limited to values being smaller than 150% of the value of the original calculation. The limited data was not evaluated as outlier having poor quality result. From modeling perspective, the restriction is an discrimination of the exponential model equation.

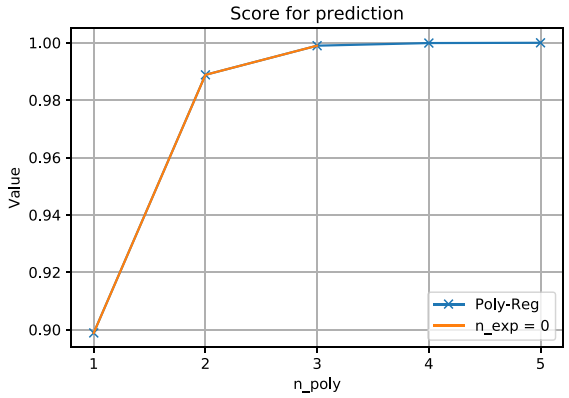

Fig. 11 Identical models. Identical score results for different degrees in (2) and (12)

Results and discussion

The previous section showed that the vectorial Eq. (2) and the polynomial regression equation according to (12) result in identical models, which was verified by comparison of least-square identification model results. This is shown here exemplarily by comparison of the RMSE-Scores for both models given in Fig. 11.

The orange line depicts the vectorial equation, while the polynomial regression model is given in blue with $\times$-markers.

Evaluation of brake-disk

This model uses $n_{sys} = 5$ system parameter. A dataset of 601 randomized calculations was created for this study. The number of polynomial model parameters n_{poly} for different polynomial degrees is therefore given in the second column of Table 1.

The prediction accuracy according to (16) is plotted in Fig. 12 for the whole Dataset n_{calc}. The green brace marks the n_{ident} samples used for identification, while the red brace is for the remaining n_{vali} datasets used for validation of the equation. The relative error for the (green) identification samples is not visible in the given plot because the axes limits are defined by the relative error of the validation dataset. The error span Δ_{rel} is approx. 15% here. The graph of Fig. 13 shows the identical evaluation using now *Over-Determination-Factor odf* $= 10$.

It is evident that the overall error spread is $\approx$ 2% and much smaller. It is also more evenly distributed between the identification and the validation areas.

The behavior of Fig. 12 is similar to the effect which is called *Overfitting* in machine ML-terminology. The model is suitable for all values of the training dataset, but it fails with the validation-datasets. Overfitting is omitted using an over-determined system and by application of the LSF-method. The model is fitted to reduce the mean error for more points than model parameter leading to a "better" prediction in the whole.

From that the question arises which *odf* is reasonably required to gain an accurate predicting model. The convergence of the prediction for the whole dataset including validation is given in Fig. 14.

On the left the RMSE-Score is for several polynomial models plotted, while the right plot shows the relative error span according to (19) within the whole dataset. The right

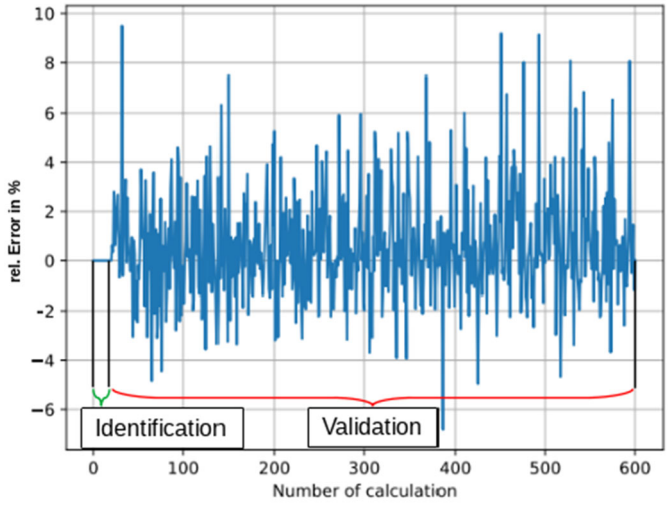

Fig. 12 Rel. error for brakedisk. *odf* = 1 Plot of the rel. error of the brakedisk with *odf* = 1

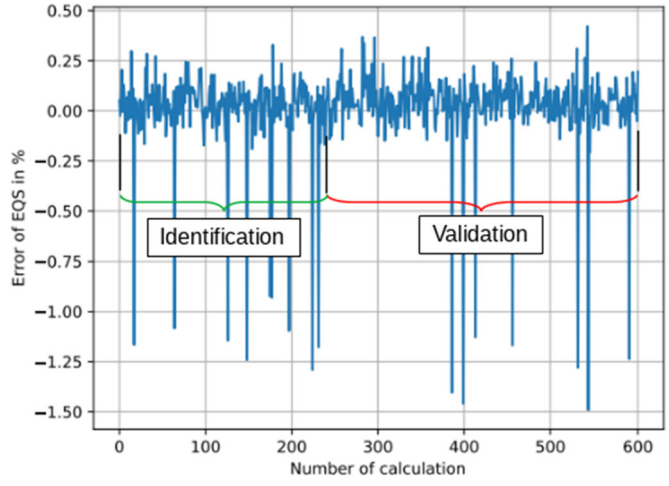

Fig. 13 Rel. error for brakedisk. *odf* = 10 Plot of the rel. error of the brakedisk with *OD* = 10

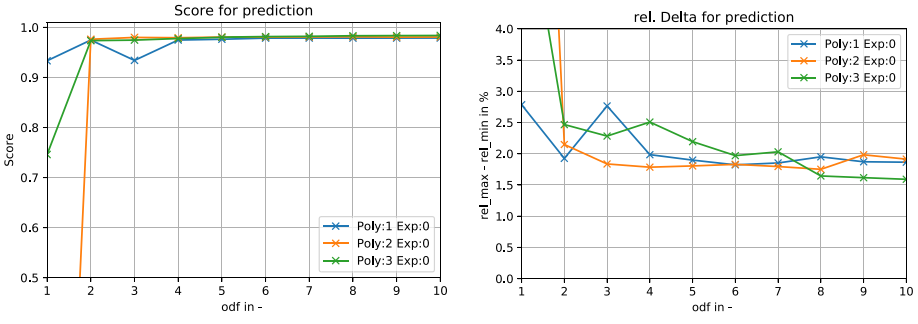

Fig. 14 Brakedisk: convergence of quality measure RSME-score and Δ_{rel} for brakedisk model

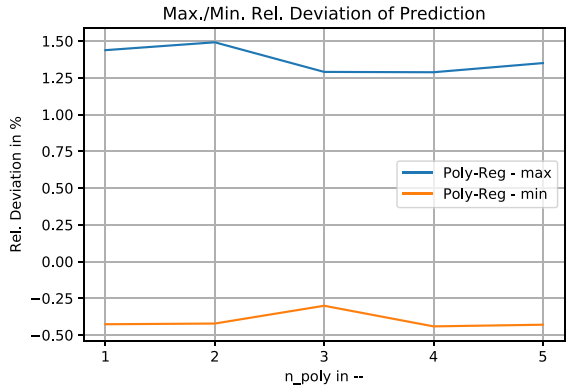

Fig. 15 Rel. Error for brakedisk-models using up to fifth polynomial degree

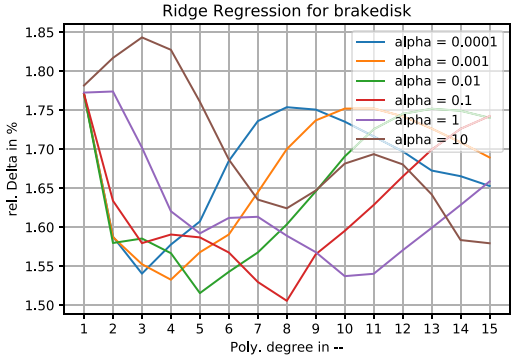

Fig. 16 Brakedisk with ridge comparison evaluation of ridge-regression-performance of Δ_{rel} for n_{poly} and α-variation for the brakedisk

plot delivers a clearly visible, more meaningful basis for distinction as the left score of all models is well above 0.9. The polynomial model with the degree 3 leads to the most accurate prediction with $odf \geq 8$, in the considered case. But the second order model with $odf = 4$ performs also quite well with an error spread of $\approx 2\%$. Higher order polynomial models do not perform significantly better, this can be taken from Fig. 15 showing the max./min deviation score up to fifth order. No better result with higher effort means less efficiency forthe model performance. Any exponential model did not fit the model significantly better and are not shown here.

Slight improvements could be reached by utilization of a *Ridge-Regression*-model which is depicted in Fig. 16. It shows the relative difference Δ_{rel} for different polynomial degrees. Ridge Regression allows modification of an additional model parameter α, which is shown with the differently coloured lines. The α-parameter is introduced to restrict overfitting of the function to the training data. A perfect model performs best with $\alpha = 0$. The higher the order of the polynomial-preprocessing, the higher α must be selected. This can be observed at the minimum points of the lines up to $\alpha = 0.1$. The optimum α increases with polynomial degree. Selecting a higher α reduces accuracy again. The model of eighth polynomial order and $\alpha = 0.1$ performs best according to this plot. This behaviour shows, that identification of α adds an additional layer of optimization to model training. It should

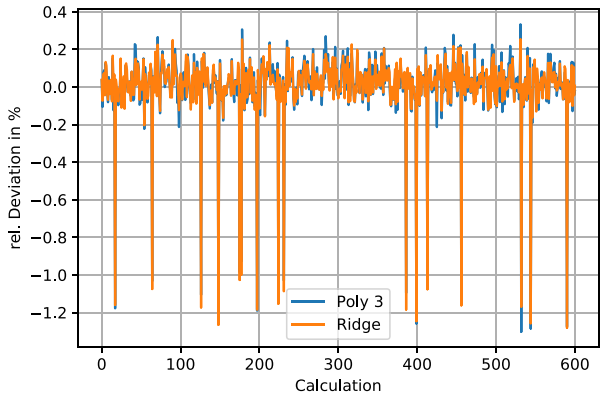

Fig. 17 Brakedisk comparison of ridge-regression to polynomial third order

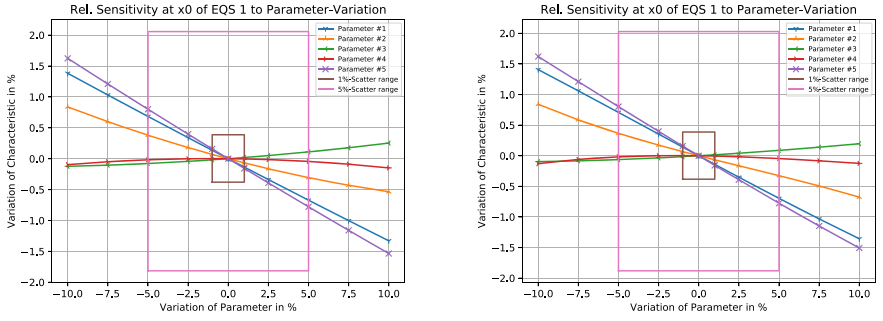

Fig. 18 Sensitivity for brakedisk sensitivity for F_1 of brakedisk(Poly=2 left, Poly=3 right)

be kept in mind, that a sufficient number of input-data must be available to train the high order models. And a high order model does not necessarily result in improved accuracy. This is depicted by the remaining two α-lines, which perform worse than the red line at $\alpha = 0.1$.

Using that Ridge-regression model and the third order polynomial model a comparison of prediction performance is shown in Fig. 17. The slight improvement can be identified here, but the overall performance is similar. Both models are able to predict quite accurately for most of the data. Only some of the predictions and always the same peak out. As none of the models outperform the others significantly, polynomial models will be investigated further for this sample case.

A main interest during product development would be the sensitivity of the characteristics to single variations in the system parameter. Such an evaluation is a simple application of the polynomial-model, which could also be easily conducted with the FE-model. The lines in Fig. 18 represents the sensitivity for second and third polynomial degree. Each of the system parameter is modified in 11 Steps: 0%, ±1%, ±2.5% ±5% and ±10%. Each of the lines describes the relative change of the system characteristic—F1 in this case—caused by that variation.

The two boxes in the plots are the result of another evaluation, which could not as simply be done with the underlying FE-model. The horizontal lines of the boxes depict the range of relative deviation that could be expected due to a modification of system parameters.

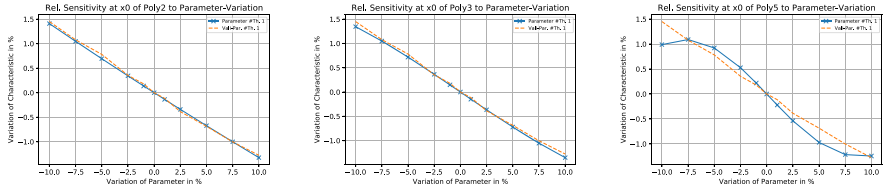

Fig. 19 Sensitivity to thickness for second, third, and fifth order model

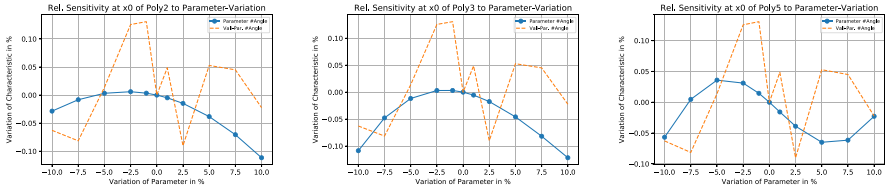

Fig. 20 Sensitivity to Angle-variation for second, third, and fifth oder model

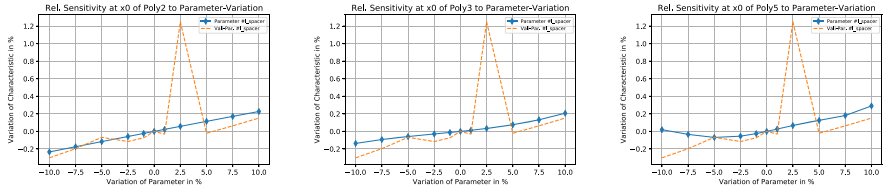

Fig. 21 Sensitivity to spacer length-variation for second, third, and fifth order model

Including the interactions of combined influences in $\vec{x}$. The vertical range is calculated by an optimization routine with variation limits within a $\pm 1\%$-range (resp. $\pm 5\%$), searching for minimum/maximum deviation from the initial value. Both plots differ only in details.

A more detailed comparison model of performance can be done by feeding back the model-parameter values to the underlying FEA. Figure 19 shows predicted (full) and FE-calculated (dashed) values for the second, third and fifth order-models. Visually evaluated, the second order model is best fitting the line. While third order model starts to overfit and the fifth order model clearly does. Please note that this model prediction is significantly closer to the calculated value than it could be expected by the 2%-spread given in Fig. 14.

Same comparison-lines are given in Fig. 20 for an angle-variation. The actual sensitivity of the natural frequency to this change is smaller, which leads to a different y-axis-scaling turning out the curvature of the prediction models. The behavior of the dashed line is somewhat strange. From the whole setup a smoother progression would be expect. This behavior indicates that the accuracy of the FE-workflow is getting close to limitations in accuracy.

This limitation is obviously visible in Fig. 21. It shows the change in the natural frequency driven by a modified spacer length. A smooth increasing F_1 as for the second degree model could be expected with increasing spacer-length. The zig-zagging-pattern of the dashed line is physically not plausible. Omitting the outlier, second order model performs best here.

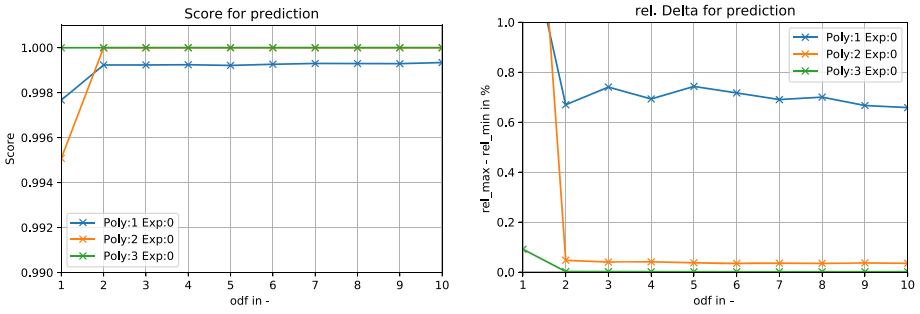

Fig. 22 Beam with holes: convergence of quality measures RSME-Score and Δ_{rel} for beam with holes

Further investigation showed that this error was introduced during automatized meshing of the geometry. The mesher sporadically gave warning messages, but a valid mesh was exported. The subsequent FEA-program-checks (e.g. Jacobian) were passed without warning. The mesh quality introduced an uncertainty of $\approx \pm 1.0\%$ in the FE-data. This stochastic variation also seems to be the reason for the peaking-out values in Fig. 17. The values and input data itself are not significantly special. This finding does not mean, that the complete FEA-procedure is not reliable. It only indicates some limitations in accuracy, which would not be of big interest for a single FEA-simulation. Especially if it is kept in mind that a modified boundary condition, for example a stiffer support or introduction of a brake-pad, would lead to a much higher than 1%-offset to the average value.

The calibration of the mean-FEA values to eventually available measurement data therefore should be done by adaption of the boundary conditions as close as possible to the measurement environment. If higher accuracy is required for scatter prediction, additional efforts like refined meshing would be the method to use.

The scatter affects the identification process similar to noisy measurement data of a real experiment. And as a measurement error of $\pm 1\%$ is quite acceptable for most applications, the accuracy of the FEA may be acceptable, too. From the perspective of this study it can be stated that sporadical deviation in identification and validation datasets indicates noisy data from the FEA. Finally, the second order model could be evaluated as best suited for the given example case. The accuracy is good and a peak out value could be assessed as an outlier.

Evaluation of beam with holes

As the geometry of this case is simpler than the brakedisk, a structured meshing was generated. Figure 22 shows the already known *odf*-convergence up to a value of 10 for the polynomial models.

All three models level out very fast and the value of $odf = 2$ is sufficient for this case. The error levels are in the order of magnitude of 0.7%, 0.05% for the first and second order equations. The third order model delivers almost exact predictions, see the right plot of Fig. 23. These plots were created for an *odf* of 2. Using the left plot, there is no distinction possible between prediction and input-data. The relative error delivers slightly better predictions for the identification section (calculation 1–168) than for the validation section. But the order of magnitude of the error is insignificant. This leads to the conclusion that the FEA-toolchain delivers accurate results without stochastic deviations.

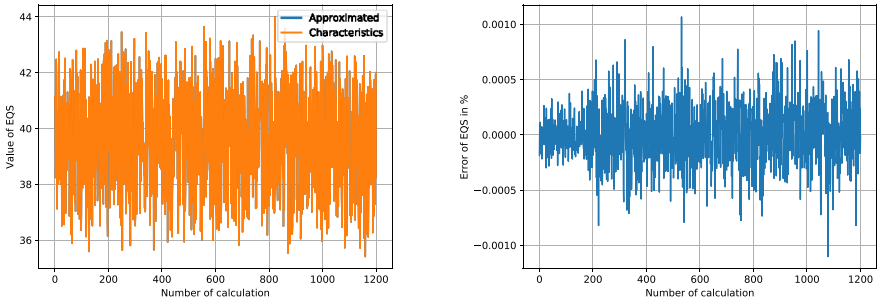

Fig. 23 Prediction beam absolute value and rel. error

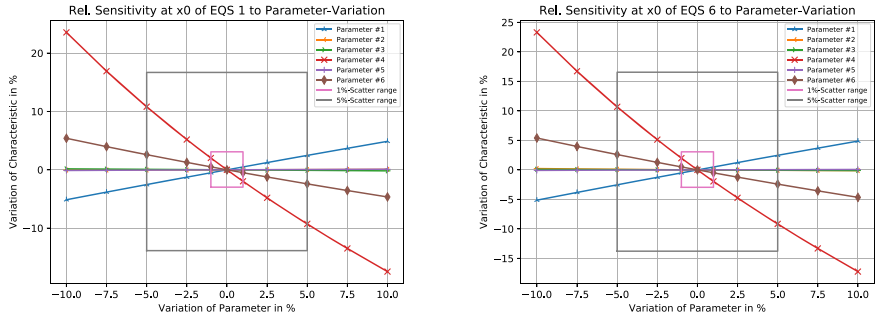

Fig. 24 Sensitivity for beam sensitivity for F_1 and F_6 of beam for all parameter

The identified model is suitable for application and higher order or exponential models are not required.

Figure 24 shows the sensitivity analysis of two natural frequencies of the beam (F_1 and F_6). Both reveal almost identical behaviour.

The first parameter (elastic modulus) increases natural frequencies as the last parameter (density) reduces it. This can be explained using the simplified equation

$$\omega_1 = \sqrt{\frac{K_{1,\text{global}}(E)}{M_{1,\text{global}}(\rho)}} \tag{23}$$

of a single spring-mass system. The stiffness $K_{\text{global}}(E)$ from FEA is a linear depending value to the elastic modulus as the mass $M_{1,\text{global}}(\rho)$ dependency to the density ρ. Dividing two natural frequencies, lead to

$$\frac{\omega_{1,a}}{\omega_{1,b}} = \sqrt{\frac{E_a \rho_b}{\rho_a E_b}} \tag{24}$$

which can be used for estimation of a modified frequency.

Putting in the values of $\pm 10\%$ for E the lower and upper deviation can be evaluated to -5.1% resp. 4.9%. And for the ρ-modification it is determined to 5.4% resp. -4.7% changes. Exactly these values can be found in the right plot for the blue and the brown lines. The most significant lever for changes in the frequency is given by the red line, which depicts the change due to total-length deviations. This correlation is clearly non-linear. All other system-parameters have an inferior impact on the frequencies.

Table 5 Boundaries and fitted values for the beam with hole sample

Parameter	Unit	Lower bound	Upper bound	Value
Elastic modulus	MPa	60000	80000	70684.8
Hole distance	mm	99.5	100.5	99.503
Hole diameter	mm	5.7	5.9	5.82
Total length	mm	1098	1102	1098.1
Width	mm	29.5	29.8	29.786
Density	g/dm^3	2.7	2.7	2.7

Table 6 Frequency-fitting for the beam with hole sample

	Measured Hz	Fitted Hz	Rel. Error %
F_1	41.5	41.492	0.020
F_2	114.5	114.503	−0.002
F_4	224.5	224.629	−0.058
F_4	371.6	371.458	0.038

The polynomial model should now be used for updating the system parameters of the FEA to fit the measured frequencies. The discrepancies of the initial FEA are already given in Table 4. A target search algorithm was created using a constraint min-search algorithm from SciPy-Library. All frequencies were included in one run. The penalty function was defined by

$$err = \sum_{i=1}^{n_{freq}} \left(\frac{F_{i,\text{pred.}}}{F_{i,\text{target}}} - 1 \right)^2 \tag{25}$$

as sum of square errors. The upper and lower bounds for the constraints are listed in Table 5 in column 3 and 4. They were chosen as plausible values for deviations that may be introduced due to manufacturing tolerances. The density ρ was set to a standard value for aluminum. With respect to (24) possible changes to density are included in the modified elastic modulus.

The outcome of the target search is given in Table 6. Minimization of the penalty function (25) was quite successful as all frequencies match extremely good. The identified values for the system parameters are within reasonable limitations. The Elastic Modulus is slightly higher than the standard static value for aluminum, which is also reasonable for a dynamic elastic modulus, see [16].

Evaluation of plate with hole

Stress concentration at small holes of tension probes may lead to steep gradients close to the hole surface. Due to that fact, exponential and higher order polynomial models are expected to be applied here.

The convergence plot related to *odf* is given in Fig. 25. Axis scaling of the plots is selected to focus on high quality values, leading to outlying points, which are of no interest.

The fitting of the polynomial models versus *odf* is leading to the expected and already seen close to optimal scores.

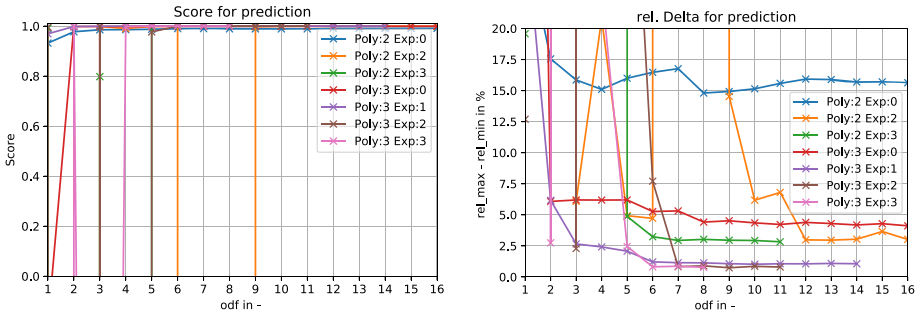

Fig. 25 Plate: convergence of score related to over-determination

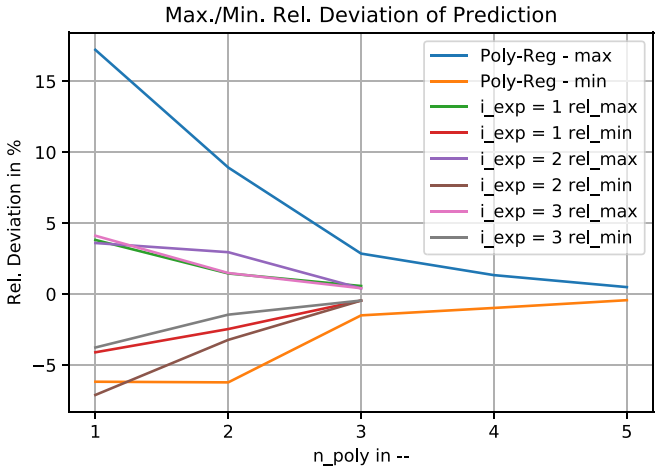

Fig. 26 Rel. deviation for plate for different model equations

The plots of the exponential models look significantly different to the plots of the polynomial models shown before. The exponential equations were chosen to model steep gradients. As they are doing, the predicted values can be very high and with this the errors compared to the real values, too. This behavior can also be assumed as an overfitting of the data to a model equation that does not fit well to the underlying problem.

Looking at the models of the third polynomial degree it can be stated that their metrics converge quite fast to small error spreads. The second and third exponential degree model does not differ significantly in this figure.

Well converged models from that pool were chosen for further application. Figure 26 gives an overview of the relative deviation for all of them. The "Poly-Reg"-lines are related to the convergence in Fig. 11. An improvement in prediction quality to the fifth order polynomial can be stated from that. Very similar values are reached by the exponential models.

Again $n_{\text{sys}} = 5$ system parameters were used in the FEA. This means in combination with Table 1 that the $n_{poly} = 5$ model got 252 parameters to be identified. The exponential model with $n_{poly} = 3$ and $n_{exp} = 3$ only has got

$$n_{total} = 2 \cdot 56 - 1 = 111$$

parameter. Reducing the effort by a factor of approx. 2.

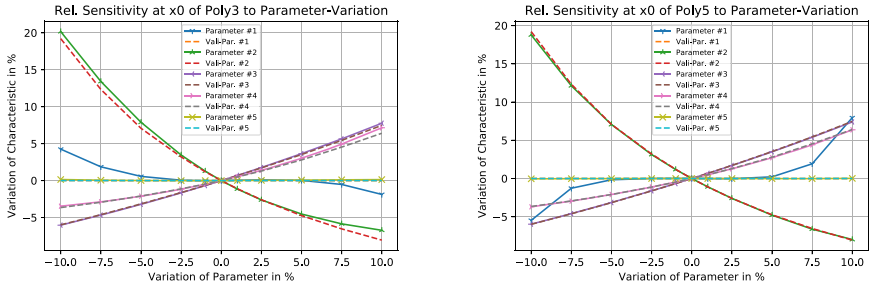

Fig. 27 Plate rel. sensitivity for third and fifth polynomial degree

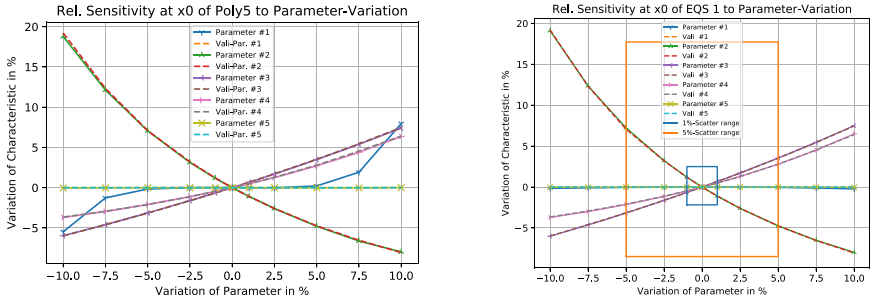

Fig. 28 Plate rel. sensitivity for fifth order polynomial and third/third exponential degree

A deeper investigation is done again with the sensitivity study compared to the actual FEA-results. For this case all lines are plotted in one graph and FE-results again are marked by dashed lines.

For this FEA no implausible scatter could be stated, and the accuracy of the FE-toolchain seems to be sufficient for this application. All visible dashed lines are obviously located closer to their related prediction for the fifth order model.

But there is a significant difference in the predictions. It can be identified at "Parameter #1" shown by the blue lines. They are mirrored on the abscissa of the graph. Which of the lines is correct? The answer is: None of them. The related reference line in dashed orange is not visible as the relative changes of the values from the FEA are close to zero (Fig. 27).

Figure 28 depicts the sensitivity analysis of the fifth-order polynomial model (left) in comparison to the third order polynomial and third order exponential model.

All visible dashed lines are fitted closely. The most interesting result of the plot again is the "Parameter #1" line. The exponential model predicts a value close to zero, as the FEA does require. Application of the polynomial model using high order is not recommended for this case, as it leads to overfitting.

To figure out the capabilities of a *Ridge-Regression*-function a model was trained using sixth-order data-preprocessing. For using the sixth order polynomial in combination with five n_{sys}-parameters it has 462 model-parameter, see Table 1. The Regression-function uses one additional parameter α for regularization. For having the minimum over-determination of $odf = 2$ a dataset of 924-samples was taken for training. Figure 29 shows the sensitivity plot for that model. The physical invalid deviation of the blue line is

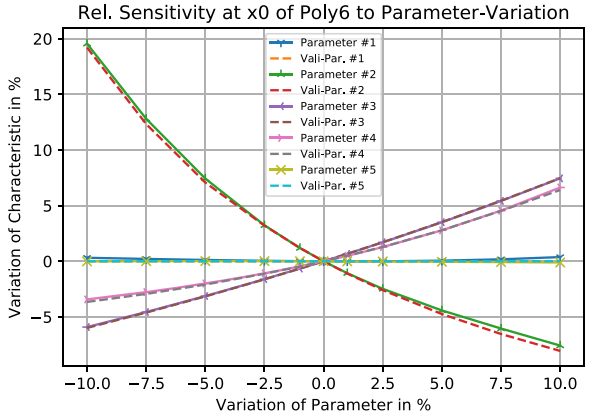

Fig. 29 Sensitivity ridge-regression of Poly-degree 6

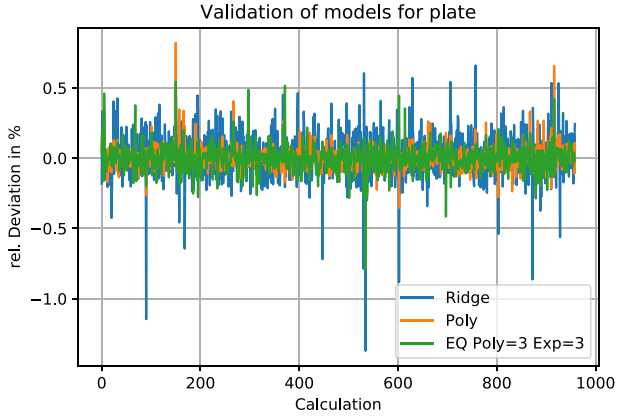

Fig. 30 Validation of plate against new data set

suppressed by the Ridge-regression compared to the fifth-order polynomial model of Fig. 28, left plot.

All other predictions are influenced, too. System-parameter #2 shows an increased deviation from the reference values (dashed line), while the fifth-order plain polynomial-model is closer to reference for this parameter.

For evaluation of the available prediction accuracy, a new dataset of random input was created, the FEA was conducted and the available models were applied to predict the outcome. Figure 30 shows the resulting relative error for all three models. All models predict values close to the FE-calculated numbers. In direct comparison, the accuracy of the Ridge-Regression is less than the others in both quality measures. The plain polynomial model performs slightly better than the exponential model does. Table 7 gives the figure of the score and $\Delta_{\text{rel.}}$ for the models. Despite of these numbers, it had been shown that the polynomial model overfits with unphysical predictions and it uses more model-parameters n_{total} and with this requires more data for reliable LSF. Therefore, the exponential model represents the better trade-off in this sample case.

This sample problem should also be approached by a *Neural-Network*-model to figure out features and obstacles of *NNs*. Finding out the right values for the number of hid-

Table 7 Comparison of quality measures for plate

	Poly	Ridge	Exp
Score	0.999963	0.999808	0.999948
$\Delta_{\text{rel.}}$	1.2	2.0	1.3

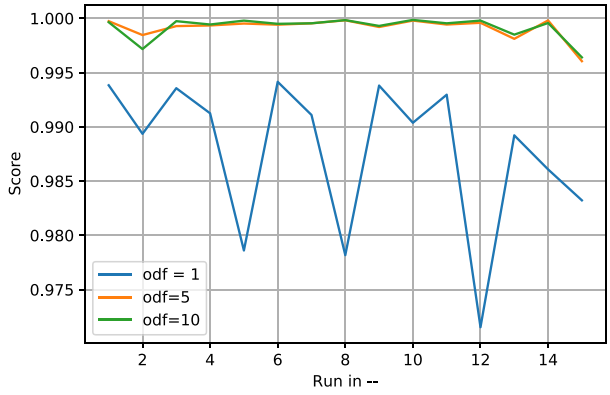

Fig. 31 Scores vs. fitting-run for several NN-fitting runs for different values of *odf*

den layers, perceptrons per layer and activation function for a neural network are not straightforward. Some rules of thumb and strategies are described in [8,10]. For this case the *tanh*-function was selected for activation, as the underlying problem is non-linear with steep gradients. A number of two layers and three perceptrons per layer turned out to be efficient. The model was built and trained using the *Tensorflow*-package, [11]. The software reported a total number of 34 trainable parameters for the whole network. The NN-Regularization was kept fix to a value of $\alpha = 0.001$ and no further tuning of NN-Model parameters was conducted.

It was already mentioned, that training of a NN requires an *initialization* of the model-parameters by *Random-Values*. This initialization may have an influence to the trained model. Figure 31 shows the score reached for different *odf* for 15 training runs with unchanged model parameters. Additionally, the random intitialization was kept constant between the different *odf*-values for each run. The effect of the initialization can be identified at run 2 and 15. The scores for all *odf*-values are reduced. With higher values for *odf* the effect of the initialization is reduced significantly. For higher over-determination values, the score gets closer to the optimum and is less influenced by the starting condition.

The *odf*-convergence of the model is given in Fig. 32. For each *odf*-step the maximum score from the 15 runs was selected. The scores of two additional training-runs using an unrestricted random initialization is depicted in Fig. 33. For values $odf > 4$ the stochastic influence of the training procedure is relevant for the prediction accuracy and a definition of a "recommended" value is not reasonable.

Figure 34 shows a comparison of relative errors for NN and Exponential-model in analogy to Fig. 30. The exponential model-predictions are obviously closer to the NN-predictions. Which demonstrates the higher prediction accuracy for that model.

Finally, the model performance can be evaluated by the sensitivity comparison shown in Fig. 35.

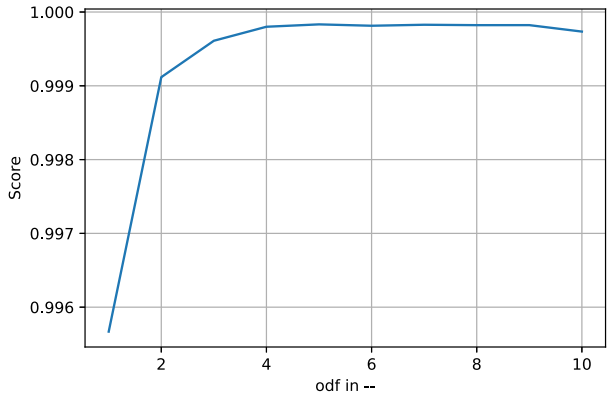

Fig. 32 Score vs. *odf* for the run in Fig. 31

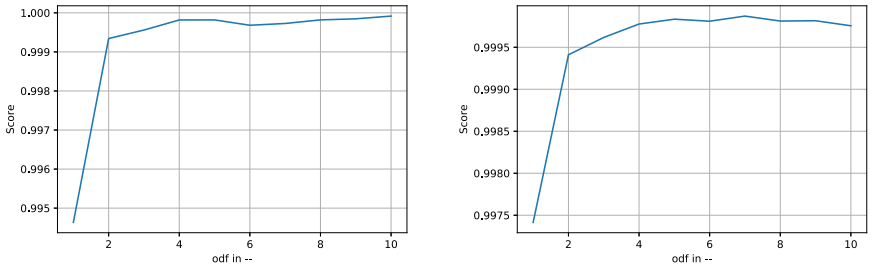

Fig. 33 Score vs. *odf* for two other runs

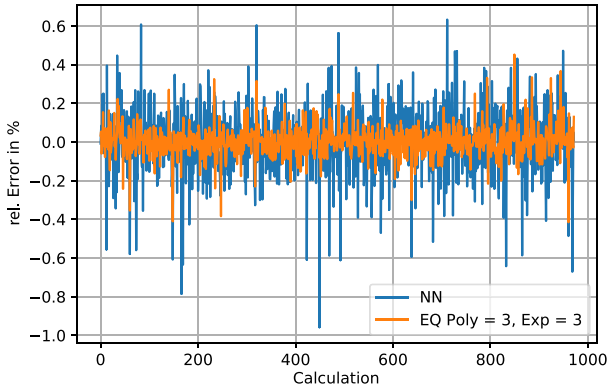

Fig. 34 Rel. error for comparison of NN vs. EQ Poly = 3 and Exp = 3 model

The characteristic variations are predicted close to the dashed comparison-lines. This is also valid for the system-parameters which were predicted unphysically by the pure polynomial models. Compared to the right plot of Fig. 28 the exponential model is closer to the validation data.

Conclusions

An, with respect to [2], improved automated calculation procedure was created and described. It is based on a numerical experiment to create stochastic data. The proce-

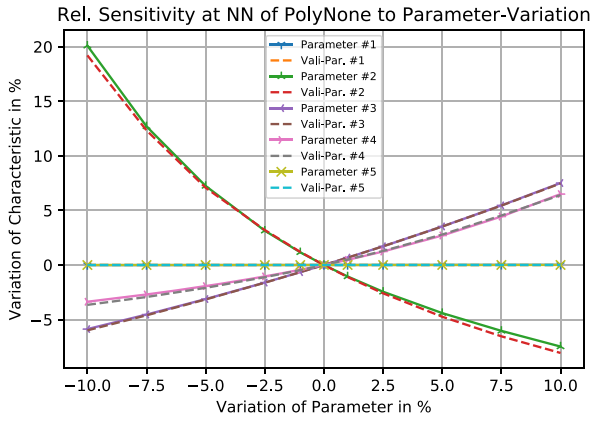

Fig. 35 Rel. sensitivity for the NN-model

dure is versatile and does not rely on a certain underlying numerical software package. For this study were applied

- two FEA-software tools and
- three meshing strategies

to investigate three different mechanical systems. The FE-models were set up parametric by n_{sys} arbitrary selected system-parameter. The numerical models lead to a dataset with n_{calc} characteristic values related to the input parameters the for the model. All datasets were analyzed with different model approaches.

The matrix-based polynomial-model equation from [2] was expanded by a third order term. In order to gain capabilities in modeling of high gradient data, an *Exponential function*-based matrix-model up to third order was introduced. These models were compared to functions provided by commonly used ML-toolboxes. It was found that the matrix-polynomial grey box model is identical to a *LinearRegression*-algorithm method in combination with a certain pre-processing of the input data, see (11). From that it can be stated that ML can be evaluated as strongly related to classical system-identification methods. Some differences in nomenclature were illustrated.

Limitations in achievable prediction accuracy for the brakedisk model revealed, that the initial numerical experiment-step is of fundamental relevance for the final model accuracy. Leading to the conclusion, that the whole procedure, especially the meshing step, must be capable to deliver finely granulated results. Also, slight changes to small modifications have to be visible in the output. For an evaluation of the FEA-accuracy, the conclusion can be drawn, that a two step method is advisable. First to do a mesh study on the unchanged geometry for definition of a required mesh density. Followed by a geometry variation study. With this, an impression of the max. available accuracy can be reached.

The parameter identification-step, resp. *model training* in ML-terminology, requires enough data-samples to be reliable. For measuring this the *odf* was introduced. Based on high quality characteristic data in combination with up to third order polynomial-model-equations, fast convergence and a required $odf \geq 2$ was identified. For the higher polynomial order and the exponential-function based models an $odf \geq 8 - 10$ is recommended for the investigated cases. This information is of real importance for practical

application, as it defines the minimum number of n_{calc} to be performed via FEA for getting the training data. An additional number of data datasets for *validation* is also required, to judge the model as valid or overfitting. According to (10) this may lead to a large number of calculations to be done in advance of the identification-step.

With the aim of a high prediction accuracy the RSME-score was identified as not being sufficient as quality metric. The maximum, relative deviation Δ_{rel} calculated from identification and validation-datasets is more meaningful in this context. Especially Fig. 13 depicts this conclusion. The specific behavior for that model at $odf = 10$ also indicates the noisy error of the underlying input-data.

Different models were investigated regarding their prediction accuracy. Unsurprisingly, an adequate model has to be selected for the related test-cases. Polynomial-models are simple in application and can be easily expanded to high-order equations. But higher order models combine two disadvantages: They need a higher number of training datasets to gain the found $odf = 8 - 10$ and they tend to overfit the data leading to unphysical predictions caused by small system-parameter variations.

ML-toolboxes provide function, e. g. *Ridge-Regression*, to reduce overfitting-tendency. These functions do this by the introduction of new model-parameter, which are unknown in advance and have to be identified during tuning of the model. For certain problems, especially when including high gradients, the exponential-equation based model delivers an efficient trade-off. The "polynomial explosion" [8] of n_{total} is avoided and sufficient prediction accuracy was reached.

Application of neural-network models has turned out to be ambivalent in the scope of this study. The prediction capabilities are powerful. A 34-parameter model is capable to predict the main features of the last sample problem. Compared to the 252-parameter pure polynomial prediction, the result is closer to physical effects. But there are different newly introduced model parameter (namely regularization and activation-function) that have to be defined by the user with an additional model tuning step. Another disadvantage of neural networks lies in the random-influence to the prediction accuracy. This introduces an additional uncertainty and error source to the prediction result. To reduce this effect a number of training runs has to be done and the best of the runs has to be chosen. This, in combination with the additional model parameters, reduces the efficiency of less model parameters in the neural network model-layers significantly.

Finally, it is concluded that model generation represents a trade-off between accuracy and effort. A customized analytical model will most probably deliver higher accuracy, while a well trained NN-model may deliver acceptable accuracy for less cost but with some uncertainty about the random initialization.

Abbreviations

AI: Artificial intelligence; DL: Deep Learning; EQS: System of equations; FEA: Finite Element Analysis; LSF: Least-Square Fit Method; ML: Machine Learning; NN: Neural Network; odf: Over-determination-factor of an Least-Square-Solution; RMSE: Root-Mean-Square-Error; SVM: Support-Vector-Machine.

Acknowledgements

Not applicable.

Authors' contributions

TG developed the overall procedure and coordinated project and writing. He also did the work on application of the presented Machine Learning functions. FG created the presented (and more) parametric FE-model of the plate with hole. He performed the calculations and investigated the convergence of the accuracy regarding odf. MG created the

parametric model of the brakedisk and created the automated procedure of mesh creation from FreeCAD using the Python-interface. He performed the required calculations and investigated the convergence of the polynomial model. DS worked on creation and implementation the of the exponential model. He also worked on the investigation of the capabilities of the model. RJ proofread the document and added her advice on presenting the results. All authors read and approved the final manuscript.

Funding Information
The study was funded by University of Applied Sciences (UAS) Darmstadt, Department of Mechanical and Plastics Engineering. Publication was funded by Open-Access-Fond of UAS Darmstadt.

Availability of data and materials
The datasets used and analysed during the current study are available from the corresponding author on reasonable request.

Competing interests
The authors declare that they have no competing interests.

Received: 7 February 2020 Accepted: 20 May 2020
Published online: 10 June 2020

References

1. Marwala T. Finite-element-model updating using computational intelligence techniques. Johannseburg: Springer; 2010. https://doi.org/10.1007/978-1-84996-323-7.
2. Grönsfelder T, Hofbauer A, Richter CH. A method for theoretical prediction of frequency scatter ranges for freestanding forged steam turbine blades. Turbo Expo: Power for Land, Sea, and Air, Volume 6: structures and dynamics, parts A and B, pp. 1085–1094; 2011. https://doi.org/10.1115/GT2011-46172. https://asmedigitalcollection.asme.org/GT/proceedings-pdf/GT2011/54662/1085/2759531/1085_1.pdf. Accessed 7 Feb 2020.
3. Keesman KJ. System identification—an introduction. 1st ed. New York: Springer; 2011.
4. Bohlin T. Practical Grey-box process identification. 1st ed. New York: Springer; 2006.
5. Papula L. Mathematik Für Ingenieure und Naturwissenschaftler, vol. 1. New York: Springer; 2014.
6. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Jarrod Millman K, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey C, Polat İ, Feng Y, Moore EW, Vand erPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, Contributors S. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. arXiv e-prints, 1907–10121; 2019. arXiv:1907.10121.
7. Moolayil J. Learn keras for deep neural networks—a fast-track approach to modern deep learning with python. 2nd ed. New York: Springer; 2019.
8. Geron A. Hands—on machine learning with Scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems. 2nd ed. Farnham: O'Reilly UK Ltd; 2019.
9. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
10. Kruse EA. Computational intelligence. 2nd ed. New York: Springer; 2015.
11. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org; 2015. https://www.tensorflow.org/. Accessed 7 Feb 2020.
12. Aggarwal CC. Neural networks and deep learning. 1st ed. New York: Springer; 2018.
13. Calculix—a free software three-dimensional structural finite element program. https://www.calculix.de/. Accessed 7 Feb 2020.
14. FreeCAD—Your own 3D parametric modeler. https://www.freecadweb.org/. Accessed 7 Feb 2020.
15. Ansys—engineering simulation and 3D design software. https://www.ansys.com/. Accessed 7 Feb 2020.
16. Vosteen LF. Effect of temperature on dynamic modulus of elasticity of some structural alloys. NACA—National Advisory Commitee for Aeronautics; 1958. https://ntrs.nasa.gov/search.jsp?R=19930085159. Accessed 7 Feb 2020.

Publisher's Note