SHORT COMMUNICATION

A nonintrusive nonlinear model reduction method for structural dynamical problems based on machine learning

Jonas Kneifl | Dennis Grunert | Joerg Fehr

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

Correspondence
Joerg Fehr, Institute of Engineering and
Computational Mechanics, University of
Stuttgart, Stuttgart, Germany.
Email: joerg.fehr@itm.uni-stuttgart.de

Abstract
Model order reduction (MOR) has become one of the most widely used tools to create efficient surrogate models for time-critical applications. For nonlinear models, however, linear MOR approaches are only practicable to a limited extent. Nonlinear approaches, on the contrary, often require intrusive manipulations of the used simulation code. Hence, nonintrusive MOR approaches using classic model order reduction along with machine learning (ML) algorithms can provide remedy. Such approaches have drawn a lot of attention in the recent years. They rely on the idea to learn the dynamics not in a high dimensional but in a reduced space, that is, they predict the discrete sequence of reduced basis’ coefficients. Open questions are the suitability of such methods in the field of structural dynamics and the best choice of the used ML algorithm. Both are addressed in this article in addition to the integration of the methodology into a modular and flexible framework that can effortlessly be adapted to various requirements. By applying the methodology to a dynamic mechanical system, accurate surrogate models are received, which can speed up the simulation time significantly, while still providing high-quality state approximations.

KEYWORDS
black box model, data-based model reduction, machine learning, nonlinear behavior, structural dynamics, surrogate model

1 | INTRODUCTION

Computer-aided engineering is a pillar of complex technical system design and an essential tool in many industries. Accordingly, product development heavily relies on modeling and simulation results. Models are used to derive control strategies and to evaluate the behavior of technical prototypes quickly. For time-critical applications, however, the conduction of high-fidelity simulations is not always feasible. In real-time applications, predictions must be made within concise time intervals as it is often the case for applications of control theory. In contrast, scanning of high-dimensional parameter spaces for adaptive multidisciplinary optimization or a robust and resilient system design requires many simulation runs. Often, not only individual components of a model but the expensive system as a whole is considered. There is no time to perform such a sheer number of simulations in today’s very short development cycles.
All this leads to a high demand for efficient but accurate surrogate models. An often-applied technique to mitigate the computational bottleneck is model order reduction (MOR). Its basic concept is to project high-dimensional dynamical systems, like the nonlinear ordinary differential equations (ODE) system of a finite element (FE) model, into a much smaller suitable subspace. In the course of this, the subspace must meet the requirement to be able to reconstruct the original system with as little loss as possible. For this, a computationally expensive preprocessing, for example evaluating the high-fidelity model at several points in the parameter space, is conducted during an offline stage. Subsequently, the solutions can be approximated during a low-cost online stage by the reduced system for arbitrary parameter sets.

Nevertheless, there is no guarantee that standard MOR approaches can approximate any system to a satisfactory degree. Strongly nonlinear behavior, as it is frequent in explicit structural dynamics, represents a great challenge among others. Therefore, mature linear model reduction techniques\(^1\) are only practicable to a limited extent. On the contrary, nonlinear methods often require intrusive manipulations of the used solver code\(^2\) or at least knowledge of the system’s governing equations. To avoid such problems, nonintrusive model order reduction (NIMOR) methods can be used. They require little or no access to the used simulation code, are straightforward to implement, and can handle black-box models. Accordingly, they are acknowledged as frequently investigated topic,\(^3\) for example, in the field of parametrized MOR.\(^4\)

In this work a nonintrusive data-based approach using machine learning (ML) techniques is examined. Machine learning, which is closely related to artificial intelligence, is currently experiencing a level of interest not seen before. The increased computing power enabled a renaissance in data science and gave new momentum to the idea of data-driven solutions. Consequently, the idea to build surrogate models based on data has become more and more popular during the past decade. It remains exciting, which algorithm suits best for which task field. In some modeling methods, ML is used to estimate model parameters\(^5\) or applied to multiscale models,\(^6\) while others use data for model adaptation of digital twins.\(^7\) A combination of reduced basis methods and machine learning is applied in a publication of Santo, Deparos, and Pegalotti,\(^8\) in which physical parameters or coefficients of an affine decomposition of a PDE are identified by a multilayer perceptron and then handed to a reduced basis solver to reconstruct the PDE solution.

Another data-based procedure to capture the dynamics of a system is to learn a mapping from suitable input variables to the decisive system states. The attempt to achieve this in the full space often fails due to the high-fidelity system’s sheer dimensionality, for example, FE models can have more than \(10^7\) degrees of freedom. Learning and predicting each individual system state is neither efficient nor reasonable in terms of resources. Therefore, a viable approach is to consider the system description in a low-dimensional reduced subspace, so that only a few quantities, the reduced basis’ coefficients, must be learned. This results in highly efficient approximations, which are still able to cover the complex dynamics of the original large-scale system by back projection into the high-dimensional space. A very popular approach to find the reduced subspace in this context is proper orthogonal decomposition (POD), in which a singular value decomposition (SVD) is applied to snapshots, that is, high-fidelity simulation results of the system. By truncating the obtained factorization matrices, an optimal low-rank approximation of the snapshots can be found. For large-scale systems, however, the computational effort of the SVD becomes a burden, which can be tackled using randomized low-rank approximations.\(^9,10\)

According to this procedure, Hesthaven and Ubbiali\(^11\) extract a low-dimensional subspace from a collection of snapshots via POD and employ artificial neural networks (ANNs) to approximate the reduced basis’ coefficients. They refer to this approach as POD-NN and apply it in the field of fluid dynamics, for example, to the Navier–Stokes equations. This method can be extended by uncertainty quantification with deep ensembles and variational inference-based Bayesian neural networks.\(^12\) In other papers the temporal sequence of the reduced states or rather coefficients is approximated by radial basis functions,\(^13\) k-nearest neighbor methods,\(^14\) or neural networks.\(^15\) For an overview of further applications, Gao et al.\(^16\) provide a summary of more applications. In the following such methods are collectively referred to as POD regression (POD-Reg).

The methodology presented in this article follows a related thought. In a reduced subspace obtained via POD, the coefficients of the reduced basis are iteratively predicted based on an initial state. This corresponds to the aim of approximating a mapping, which maps the current reduced system state onto its following successor. In other words, the regression algorithm takes the role of a classical ODE solver instead of predicting the dynamics in one step from the simulation parameters directly as it is usually done. Such a procedure, that is, approximating the dynamics via a state transition, has proven its abilities in other data-driven dimensionality reduction algorithms like the dynamic mode decomposition\(^17\) or in the closely related Koopman operator theory, which can be used to approximately represent a nonlinear system in a linear framework. The eigenfunctions of the Koopman operator form intrinsic coordinates...
that can linearize nonlinear dynamics and are usually hard to find. Thus, deep learning can be used to identify them.\textsuperscript{18}

In contrast to the presented research, one major feature of this work is to build a modular framework around the idea of learning the dynamics in the reduced subspace. This means it is not fixed on the combination of POD with a certain ML algorithm but with a variety of algorithms. Furthermore, the framework provides the possibilities to simply manipulate the snapshot matrices in time and amount, modify the reduction matrix by its size, the used data, and normalization methods, and offers the option to choose which quantities are approximated, for example, nodal displacements and velocities. By using the nonintrusive approach, the framework is, additionally, independent of special software solutions and can be applied to a wide range of problems.

The modular design allows to generate several surrogate models for arbitrary combinations of the mentioned options and compare them efficiently. As a result, individual solutions can be tailored precisely to particular problems. One of the most important aspects in this context is the choice of the regression algorithm. Each one provides advantages and disadvantages and must be selected depending on the corresponding requirements. To provide the interested reader with a guide for a selection of machine learning algorithms, four of the most promising ones are implemented. These include linear regression, a $k$-nearest neighbor algorithm, a neural network, and a Gaussian process. In addition to the algorithms presented here, there exists a multitude of variations and completely different algorithms with their own benefits, which the reader is strongly encouraged to test.

Most research in the area of data-based model order reduction and especially in POD-Reg methods deals with problems from the field of fluid dynamics. Structural dynamical problems play a subordinate role at best. Both, structural dynamics and fluid dynamics, deal with hyperbolic partial differential equations (PDEs) but differ in the applied discretization methods. Structural dynamics is one of the most important tools in the industrial development of technical systems. Among other topics it is concerned with the investigation of vehicle impact scenarios to increase passenger safety. Thereby, the most significant challenges are nonlinearities resulting from material behavior or nonlinear boundary conditions such as contact.

Even though crash simulations are underrepresented in the former mentioned areas, they have been an area of active research in nonlinear MOR, for example, in the application of randomized low-rank approximations.\textsuperscript{9,10} A non-intrusive MOR approach applied to crash simulations can be found in Guennec et al.\textsuperscript{19} There, a regression task is performed on coefficients obtained by a CUR decomposition of time snapshots. While such a decomposition yields physically interpretable and structural preserving reduced basis vectors,\textsuperscript{20} the reduced size to approximate the system with the same accuracy as with POD is significantly higher. Therefore, it is exciting to transfer the methodology of POD-Reg to problems arising from structural dynamics and examine how well it performs. Both research questions are addressed in this work. In detail, the suitability of the data-based NIMOR approach is examined on the basis of the crash behavior of a racing kart. The latter can be seen as a simple representation of a lightweight vehicle and thus represents a very exciting and practice-oriented research topic. For this racing kart several surrogate models are created and examined. They are compared regarding their required amount of data, simulation speedup, as well as approximation quality.

In order to explain the applied approach, the used workflow and the required theoretical background is presented in Section 2. Hereafter, the methodology is applied to a structural mechanics problem, that is, a crash of the racing kart frame, in Section 3. Afterward, the results are discussed in Section 4. The article finishes with a conclusion and an outlook in Section 5.

2 METHODOLOGY AND PROCESS CHAIN

One possibility to model systems arising from structural dynamics is to use finite element method (FEM).\textsuperscript{21} During this procedure, a continuous model is discretized into many smallest elements. Unfortunately, numerous of these finite elements are necessary to represent complex systems leading to a high computing effort. Model order reduction, however, can be applied to limit the resulting computational costs. It is concerned with reducing the computational complexity without losing the essential features of the original complex model.

Nevertheless, in many engineering challenges the exact equations used by software solutions to calculate the underlying dynamics remain hidden and inaccessible. Therefore, the remaining details, like the in- and output of a system, must be sufficient as a basis for analyses, modeling, and simulation. This is of particular interest against the background of commercial simulation and FEM software, where the producers are careful not to reveal their source code. With the
left information, the user cannot apply most of the effective nonlinear reduction methods. In such cases, nonintrusive nonlinear MOR methods, where the system’s behavior is not drawn from explicit knowledge about the internal equations but learned from data, can present a solution. In the following, a nonintrusive data-driven MOR approach is used to find suitable surrogate models for the dynamics of structural-mechanical systems. Roughly described, this is done by reducing the system’s dimension with classical model order reduction first and learning the temporal state evolution in the reduced space via machine learning algorithms afterward.

The methodology is integrated into a modular framework, shown in Figure 1. Four major steps describe the framework. First, in a computational expensive preprocessing step, high-fidelity FEM simulations are conducted to gather simulation data and compute a reduction matrix. For this purpose, several jobs are first parameterized in Matlab and sent to the FE software’s calculation queue. Subsequently, the obtained simulation results are extracted and processed, whereupon a reduced subspace is determined using the extracted data. These first steps are indicated as blue area in Figure 1.

In the following training phase, highlighted in red, regression algorithms learn the system’s unknown behavior based on the generated data. With this knowledge, the nonlinear reduced model can be evaluated in the prediction phase (green) and finally be compared against the full model, indicated by yellow color. These individual steps are explained in detail in the following subsections.

2.1 Preprocessing

In order to cover the system’s behavior, several high-fidelity simulations with different parameter sets are conducted. Therefore, \( \kappa \) different parameter sets \( p_i, i = 1, \ldots, \kappa \) are sampled to cover a wide range of the parameter space. The offline simulation results, that is, the FE simulation results, serve as a presumed reference and contain information about the

---

**Figure 1** Modular framework to conduct a nonintrusive nonlinear model order reduction by learning a temporal state mapping in a reduced setting. For this, a combination of proper orthogonal decomposition and regression algorithms is utilized.
nodal and element quantities of the model. The main interest lies on the nodal quantities. Each of the \( n_{\text{nodes}} \) nodes possesses three translational and three rotational degrees of freedom (DOFs) resulting in six DOFs per node and \( N = 6 \cdot n_{\text{nodes}} \) in common. Accordingly, the state trajectory of the \( n \)th node during the \( i \)th simulation run is described by its displacements \( q_{n,i}(t) \in \mathbb{R}^6 \). Each individual simulation result \( S_i \) contains the state trajectories for the whole range of nodes and is stored in the so-called snapshot matrix

\[
S = \begin{bmatrix}
S_1(t) & S_2(t) & \ldots & S_{\kappa}(t)
\end{bmatrix} \in \mathbb{R}^{N \times \kappa n} \quad \text{with} \quad S_i(t) = \begin{bmatrix}
q_{1,i}(t_1) & \ldots & q_{1,i}(t_\eta) \\
q_{2,i}(t_1) & \ldots & q_{2,i}(t_\eta) \\
\vdots & \ddots & \vdots \\
q_{n_{\text{nodes}},i}(t_1) & \ldots & q_{n_{\text{nodes}},i}(t_\eta)
\end{bmatrix},
\]

where \( \kappa \) is the amount of time steps one simulation contains.

### 2.1.1 Model reduction

With the help of this snapshot matrix, a projection matrix can be found. The key concept of reduction is projection, whereas a projection \( P \) is defined by the two spaces spanned by the orthogonal matrices \( V \) and \( W \). In general, a distinction is made between Galerkin projections, that is, \( V = W \), and Petrov–Galerkin projection where \( V \neq W \). For the sake of clarity, only Galerkin projection is considered in the following. The reduced vector space is defined by the span of \( V \) and quantities represented in it are marked with a bar throughout this article, for example, \( \bar{q} \). With the projector’s help, the approximation of an original vector can be received from the reduced one following

\[
q \approx \bar{q} = Pq = VV^T q = V\bar{q}.
\]

There exist several approaches to find the necessary reduction matrix. One popular is proper orthogonal decomposition (POD), which can be used for the reduction and approximation of nonlinear dynamical systems. This is done by representing their state trajectories \( S_i(t) \in \mathbb{R}^N \) in a lower dimensional space \( \mathbb{R}^r \), where \( r < N \) holds. It should be noted, however, that the reduced system is prone to errors for trajectories that have not been measured.

The goal of POD is to find a set of orthonormal bases \( u \in \mathbb{R}^N \) that are able to represent the snapshot matrix, that is, \( S = U\Gamma \) with \( s_i = \sum_{j=1}^{N} \gamma_{ji} u_j, \ i = 0, \ldots, \kappa \eta \), \( s_i \) and \( u_i \) representing the \( i \)th column of \( S \) respectively \( U \) and \( \gamma_{ji} \) is an entry of the matrix \( \Gamma \). Such bases can be found using the singular value decomposition (SVD)

\[
S = U\Sigma\Omega,
\]

where the coefficients are specified by \( \Gamma = \Sigma\Omega \) with the orthogonal matrix \( \Omega \). The columns \( U \) are known as left singular vectors and those from \( \Omega \) are accordingly called right singular vectors. The former represent the system’s empirical eigenfunctions, which are called POD modes hereafter. Furthermore, the system’s singular values are given by the positive diagonal entries of \( \Sigma \), which are stored in descending order. The extent of these singular values \( \sigma \) reflects the contribution of the associated POD mode to the overall approximation. Hence, if they decrease fast, it is possible to approximate the original system with only the \( r \) most dominant POD modes. By simply extracting the first \( r \) columns of \( U \), a reduction matrix for a Galerkin-projection

\[
V = U_r \in \mathbb{R}^{N \times r}
\]

is found. According to Berkooz et al., this results in an optimal linear low-rank approximation with respect to the Frobenius and \( L^2 \) norm.
2.2 | Training

The second phase of the process chain consists of the so-called training and includes the processing of the data besides the actual training itself. The aim is to find a nonlinear function $\Phi : X \to Y$ which maps the current system state, that is, the nodal displacements $q(t_i)$, onto the following ones $q(t_{i+1})$. In order to increase the prediction quality, an extended parameter set $p(t) \in \mathbb{R}^n_p$ is attached to the input. It may contain, for example, the simulation parameters selected for the full FEM simulation in which case $p(t_i) = p$ becomes time-independent. Instead of learning this mapping in the high-dimensional original space, it is learned in its reduced representation to minimize the number of variables that must be predicted. Thus, the searched function

$$q(t_{i+1}) = \phi(\bar{q}(t_i), p) \quad (5)$$

is supposed to map the reduced representation of a system state $\bar{q} = V^T q \in \mathbb{R}^r$ on its successor. Based on this, the training data can be composed. The input $X$ consists of the states during the first time steps $t_1, \ldots, t_{\eta-1}$ along with the extended parameter sets $p(t)$ and the respective output $Y$ contains the time-shifted states from $t_2, \ldots, t_\eta$. According to this description, the data set is obtained as

$$D = \begin{bmatrix} \bar{q}(t_1) & \cdots & \bar{q}(t_{\eta-1}) \\ p(t_1) & \cdots & p(t_{\eta-1}) \end{bmatrix}, \quad i = 1, \ldots, \mathcal{X} \, N \in \mathbb{R}^{(r+n_p)(\eta-1)} \in \mathbb{R}^r \times (\eta-1) \quad (6)$$

Powerful tools to find the underlying relationship of the given data, that is, find $\phi(\bar{q}(t_i), p)$, are regression algorithms from the field of machine learning.

2.2.1 | Machine learning

Machine learning enjoys great popularity and provides a bouquet of possibilities and algorithms to gather knowledge out of data. Suppose the previously introduced data underlies some statistical model. Then the task of machine learning can be described as approximation of the original distribution $\phi(\bar{q}, p)$ by fitting a function $\hat{\phi}(\bar{q}, p)$ to the given data set $D$. There exist numerous different ML algorithms and in certain applications some approaches will be superior to others and vice versa. Therefore, they are integrated in a modular manner into the framework making them interchangeable and comparable.

A popular regression approach is to constrain the data to a given class of functions, such as polynomial ones. In the learning process, a finite number of possible parameters is optimized to learn the underlying data in the best possible way with respect to a specific metric. Algorithms following this procedure are called parametric. A serious disadvantage of this approach is that even if there is no knowledge about the data structure, a predefined structure is imposed. Instead of defining a function type from the outset, it can be advantageous to consider all functions that allow a possible representation of the given data. Without a predefined restriction, they can have as many parameters as required. This represents a contradiction to parametric procedures, where the number of parameters is known in advance. Hence, such procedures are called nonparametric, that is, they have an infinite number of parameters. In this article one parametric and three nonparametric representatives are validated for the intended use. Since the functionality of these algorithms cannot be discussed extensively in this article, the reader is referred to the given references. The used parametric approach is linear regression (lin. regr.), while the nonparametric ones consist of generalized regression neural networks (GRNN), $k$-nearest neighbor methods ($k$-NN), and Gaussian processes (GPR).
2.3 Evaluation and validation

Once an approximation of the nonlinear mapping (5) is learned, the dynamics of the system can be approximated starting at some initial state \( \hat{q}(t_1) \). Iteratively, all following successor states are estimated yielding the dynamics from \( t_2 \) to \( t_\eta \). To compare them with the high-fidelity results, they must be projected back into the full space following \( \hat{q}(t) = V\hat{q}(t) \).

It is noteworthy that the presented regression workflow has a lot in common with the well-known regularized linear regression. This is explained in the following small digression. Consider \( \hat{X} \in \mathbb{R}^{nxq} \) as centered input matrix without intercept and \( y \in \mathbb{R}^n \) as one corresponding vector of observations. In linear regression, the approximation \( \hat{y} \) of an output is linearly based on some parameter \( \beta \) and some input \( \hat{X} \) yielding \( \hat{y} = \hat{X}\beta \). Substituting \( \hat{X} = U\Sigma \Omega \) by its SVD decomposed presentation and the regression coefficients by their optimal solution \( \hat{\beta}_{\text{ridge}} \) using ridge regularization with a parameter \( \lambda \) penalizing the square of the magnitude of the coefficients, results in a regression model of the form

\[
\hat{y} = \hat{X}\hat{\beta}_{\text{ridge}} = U\Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma U^T y = \sum_{j=1}^{q} u_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda} u_j^T y.
\] (7)

This in turn means that ridge regression approximates the values of \( y \) with regard to the basis \( U \) and shrinks those coefficients of the basis vector with smaller singular vectors \( \sigma_j \) more. In contrast to this, the POD-Reg approach separates the basis selection and regression task—in the process illustrated in this short communication, by design, only the dominant singular values are kept and thereby, the computational effort during the training phase can be significantly reduced. A detailed comparison between the POD-Reg approach and regularized linear regression, nevertheless, is an exciting research topic which is beyond the scope of this article but worth comparison in the future.

The methodology described in the last subsections covers the complete workflow to apply POD-Reg. In the following, its suitability for the creation of surrogate models is shown using an example from structural dynamics.

3 IMPLEMENTATION

The scenario, on which the presented methodology’s performance is evaluated, is a crash simulation of a lightweight racing kart. Its model description and the relevant FE software files are provided under Kneifl and Fehr. For the high-fidelity simulations the commercial FEM tool LS-DYNA is utilized, which is mainly applied in crash simulations and high impact dynamics. In this context, it should be noted that any other simulation/FEM software could be used as well since only the simulation data is required for the following steps.

Despite its simple appearing structure, the kart model possesses very interesting aspects for investigation. By using it, not only dynamical behavior but also nonlinear material and nonlinear contact behavior can be explored. Hence, it is a model which previously was subject to research in the field of MOR. Furthermore, with its basic setup it simplifies the possibility to see a separation of effects, that is, the obtained results are easier to compare regarding the impact the selected hyperparameters of the surrogate models have. Since the dynamic behavior of a racing kart highly depends on the structural characteristics of its frame, the latter is suitable for studies of model reduction. Thus, the frame shown in Figure 2(A) is detached from the remaining bodies, like its rear axle or rotational joints in the front knuckles. As displayed, the weight of the driver and engine are represented by point masses, which are connected to the frame by light and stiff beams.

Crash simulations are carried out as load cases for the kart as visualized in Figure 2(B). During these, the kart frame maintains its initial speed \( v_{\text{init}} \) until it hits a rigid wall. Since there are several thousand timesteps throughout each FE simulation but not all are required for the approach, the results are only sampled every 0.25 ms. In the next step, the system states at the selected time steps are assembled in the snapshot matrix \( S \). Applying the POD introduced in Section 2.1 yields the reduction matrix \( V \) and the singular values \( \sigma \). Videos of the corresponding ten most dominant POD modes are made available as digital supplementary material along with the simulation results generated by the full model. The selection of basis vectors, that is, POD modes, is based on the magnitude of the corresponding singular value. Dealing with temporal snapshots, however, it is noteworthy that this common selection of basis vectors is not always the best choice and other selection criteria may be beneficial.
The extended parameter \( p \) used for the kart simply corresponds to its initial speed \( v_{\text{init}} \). For the training of the algorithms a data set is composed of \( \kappa \) simulation results drawn from uniformly distributed values of \( v_{\text{init}} \) within the interval [5 m/s, 40 m/s]. This results in

\[
D = \left( \begin{array}{c}
\ddot{q}_1(t_1) & \ddot{q}_1(t_\eta - 1) \\
v_{\text{init}} & v_{\text{init}}
\end{array} \right) , \left( \begin{array}{c}
\ddot{q}_i(t_2) & \ddot{q}_i(t_\eta) \\
v_{\text{init}} & v_{\text{init}}
\end{array} \right)_{i=1, \ldots, \kappa} . \tag{8}
\]

For test purposes further simulation results based on randomly obtained initial velocities serve as reference.

Following the description of Section 2.2, the underlying dynamics are learned from this data providing nonlinear surrogate models, which are validated in the section hereafter. The regression algorithms are incorporated using the Matlab implementation `newgrnn` for the generalized regression neural network and `fitrgp` for the Gaussian process, while own implementations are used for linear regression and \( k \)-NN.

### 4 SIMULATION RESULTS

Several regression algorithms are introduced in Subsection 2.2, all possessing their own benefits and drawbacks. By creating several surrogate models on their basis, the algorithms’ abilities in the used methodology are made comparable. Different suitable hyperparameters are tested for each algorithm and the most promising ones are presented in the following. This means a number of \( k = 10 \) neighbors is chosen for the \( k \)-NN algorithm, squared basis functions are used for the linear regression and the ardmatern52 kernel function is chosen for the Gaussian process.

For the studies, up to 17 high-fidelity simulations of the kart model serve as training data. As mentioned in Section 3, they are based on uniformly distributed parameter sets \( p \), that is, initial velocities \( v_{\text{init}} \). Three further full simulations, based on randomly drawn initial velocities (\( v_{\text{init}} \in \{34.85, 14.96, 26.92\} \) m/s), serve as validation data. All simulation results are sampled equidistantly every 0.25 ms.

Appropriate measures of error must be defined to evaluate the results. The first of which is the error of the node displacements averaged over all nodes

\[
d_{\text{mean}}(t_j) = \frac{1}{n_{\text{nodes}}} \sum_{i=1}^{n_{\text{nodes}}} d(q_i(t_j), \ddot{q}_i(t_j)), \tag{9}
\]

where \( d(q_i, \ddot{q}_i) \) is the euclidean distance between the node displacements obtained by the full model and those ones obtained by the reduced model. The overall mean distance \( d_{\text{mean}}^{[t_1, t_\eta]} = \frac{1}{\eta} \sum_{j=1}^{\eta} d_{\text{mean}}(t_j) \) is received by averaging these values over the time. For all investigations only the interesting time period, in which highly dynamic behavior occurs, is considered. Thus, the overall mean distance \( d_{\text{mean}}^{[t_1, t_\eta]} \) allows a reliable estimation of the holistic performance. Another error measure is the maximum node distance averaged over all time steps \( d_{\text{max}}^{[t_1, t_\eta]} = \max_j \max_i d(q_i(t_j), \ddot{q}_i(t_j)) \).

The last quality

\begin{itemize}
  \item point masses
  \item driver
  \item light and stiff beams
  \item shell elements
\end{itemize}

\( \text{(A) Frame of the racing kart model with the positions of the mass points of the driver and engine} \)

\( \text{(B) Four states of the Kart frame crashing into a rigid wall from a top view} \)

\textbf{FIGURE 2} The isolated frame of a racing kart (A) and a visualization of a crash simulation (B) with an initial velocity \( v_{\text{init}} = 20 \text{ m/s} \)
Table 1: System settings

| Category     | Value             |
|--------------|-------------------|
| Operating system | Debian 10 Buster |
| CPU          | AMD Ryzen 3960X   |
| RAM          | 256 GB            |
| Language     | Matlab R2019b     |

Table 2: Performance

| Method   | $\Delta t_{\text{train}}$ (s) | $\Delta t_{\text{red}}$ (s) | $S_1$ | $d_{\text{mean}}^{[\Delta t_{\text{train}}, \Delta t_{\text{red}}]}$ (cm) | $d_{\text{max}}^{[\Delta t_{\text{train}}, \Delta t_{\text{red}}]}$ (cm) |
|----------|----------------|----------------|------|---------------------------------|---------------------------------|
| k-NN     | -              | 0.27           | 11.519 | 0.92                           | 7.99                           |
| Lin. regr.| 7.87           | 0.07           | 19.693 | 1.33                           | 10.34                          |
| GRNN     | 4.97           | 1.21           | 604   | 0.91                           | 6.38                           |
| GPR      | 868.99         | 0.79           | 1663  | 0.39                           | 2.46                           |

A first outlook on the performance using a suitable amount of training data (9 snapshots/full simulations) and a medium time step size of the reduced model $\Delta t_{\text{red}} \approx 5 \cdot 10^{-4}$ s is given in Table 2. Despite mentioned difficulties in the comparison of simulation times, it is particularly remarkable that the surrogate models' simulation times $\Delta t_{\text{sim}}$ lie within the range of seconds or milliseconds, whereas a reduced model with linear interface model reduction techniques still takes minutes. In addition, it is noticeable that Matlab’s implementation of Gaussian processes requires a comparatively high training time $\Delta t_{\text{train}}$, and that k-NN and linear regression yield the highest speedup. Further discussions about the results follow hereafter.

4.1 Algorithm comparison

In the field of model reduction, a distinction can be made between two kinds of applications. On the one hand those for which the preprocessing phase is time and cost critical and, on the other hand, those for which it is only a matter of saving resources during the online phase. For the former it is of certain importance to conduct as few full simulations as possible. Accordingly, the question how much data is required until further information does not yield significant improvement in the algorithm’s prediction quality is of certain importance. For this investigation, several simulations with surrogates using a different number of snapshots, that is, full simulation results as training data, are conducted. As this study considers a realistic use case, where no additional data is available through the simulation workflow, the POD is only applied to all snapshots of the selected simulation runs. Hence, the following learned results differ not only due to the amount of data but also due to the quality of the reduced basis in which the learning occurred. In further investigations more elaborate snapshot selection schemes, like the Greedy or the POD-Greedy could be used in combination with error estimation techniques. In preliminary examinations the influence and importance of more elaborate
FIGURE 3  Error quantities between the reduced and high-fidelity model obtained by several simulations using a different amount of snapshots for training (A) and for different time step sizes (B, C).

FIGURE 4  Mean distance $d_{\text{mean}}$ with respect to the time $t$ (A) and overall mean distance $d_{\text{mean}}^{[t_1,t_2]}$ with respect to speedup $S_t$ (B) using different regression algorithms, nine full simulations as training data, and a time step size of $\Delta t_{\text{red}} = 5 \cdot 10^{-4} \text{s}$.

snapshot selection schemes were seen. As this elaboration is beyond the scope of this article, it should only be noted that their influence cannot be neglected and is one important tuning parameter for the overall simulation workflow. In this context, linear regression yields the most unsteady results and provides a significant outlier using four snapshots as displayed in Figure 3(A). Gaussian processes, on the contrary, yield consistently satisfying results and can convince even using few data sets. However, even if the algorithms do not perform equally well with an additional amount of accessible offline data, it applies to all that there is no relevant improvement if more than nine snapshots are used. Hence, this is the amount used for further examinations.

At this point, it should be positively emphasized that in contrast to classical ODE solvers, the used methodology is not bound to any minimal step size to obtain stable results for the examined example. Thus, depending on the individual requirements and the used model, the step size of the surrogate model $\Delta t_{\text{red}}$ can be increased to further speed up the simulation without a remarkable influence on the approximation quality as displayed in Figure 3(B, C). It needs to be clarified that the step size of the high-fidelity model remains unchanged in this comparison since it is bound to small step sizes due to the numerical solvers.

For a comparison between the individual algorithms themselves, the surrogate’s step size is chosen to be at the medium value of $\Delta t_{\text{red}} = 5 \cdot 10^{-4} \text{s}$. The thereby obtained results are given in Figure 4. Considering the temporal evolution of the mean distance for the individual algorithms (Figure 4(A)), it is striking that the Gaussian process outperforms the others. While the $k$-nearest neighbors algorithm and the generalized neural network perform very similar, linear regression comes off worst in this category. Regarding the simulation speedup, the circumstances turn as shown in Figure 4(B). The highest acceleration is reached by linear regression, while GPR and GRNN are the slowest. Furthermore, $k$-NN is placed between the two peaks and yields a solid simulation time acceleration.

Overall, the results presented so far suggest that first, dynamics can be captured quite fast with a satisfying approximation quality, but each algorithm performs differently, and second, increasing the time step size is an appropriate tool of choice to speed up simulations. These assumptions are valid at least for the example shown and for similar scenarios. However, for more complex scenarios like crash simulations with regular folds or bifurcating system behavior, the
proposed methodology faces challenges. In particular, arbitrarily increasing the time step size for problems in which the smallest parameter changes lead to qualitatively different dynamic behavior is of questionable success. Sophisticated parameter changes are another challenge. This can include high-dimensional parameter sets, which suffer from the curse of dimensionality and the accompanying data consumption, or complex parameter dependencies like shape and topology modifications. Both can be challenging and may require modifications in the used POD regression approach.

Last but not least, a comparison of the two most promising algorithms is presented. On the one hand, GPR because it provides the best approximation quality, and on the other hand $k$-NN, which offers the best trade-off between performance and speed. Both accelerate the simulation time considerably and approximate the node displacements very well. To supply the reader with additional and more tangible results, a visualization of the simulations at three points in time is given in Figure 5 with accompanying videos to be found in Kneifl and Fehr.29 There, the high-fidelity results are compared with the low-cost approximations revealing that the kart’s qualitative dynamics are captured in a good manner by both algorithms. While $k$-NN overestimates them, GPR slightly underestimates them but is closest to the reference. Therefore, a generally valid recommendation for the best choice of algorithm cannot be given. For tasks where computational acceleration is prioritized $k$-NN is clearly the favorable choice since it is not only faster in online simulations but also does not require any training time. However, if approximation quality is prioritized Gaussian processes reveal their potential.

Only approximating positional information of a system may in some cases be insufficient. Hence, it can be beneficial to include further quantities like velocity, acceleration, strain, or stress into the surrogate model to complete the description of the system. For this purpose, the data set (6) can simply be appended, for example, by the nodal velocity $\dot{q}$ and acceleration $\ddot{q}$. For the reduction step of the newly introduced quantities, it is usually necessary to calculate a separate reduction
matrix, since they are typically embedded in another subspace as the displacements. The map to learn to dynamics (5),
in turns, changes to $[\tilde{\mathbf{q}}(t_{i+1}) \quad \ddot{\tilde{\mathbf{q}}}(t_{i+1})] = \Phi(\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i), \ddot{\mathbf{q}}(t_i), \mathbf{p})$.

5 | CONCLUSION AND FUTURE WORK

In this article a nonlinear nonintrusive model reduction approach for structural dynamic models was implemented and
evaluated. Based on a combination of classical projection approaches and regression algorithms from machine learning,
the temporal evolution of reduced system states was learned. This methodology yields the benefit of being data-driven.
Thus, it is easy to implement for arbitrary approximation problems and capable of capturing nonlinearities. While similar
approaches have been applied to fluid dynamic problems, this article transferred the methodology to structural dynamics.
In addition, it integrated the approach into a modular framework enabling fast modifications of the used algorithm,
hyperparameters, and learned quantities.

For this purpose, the approach itself and required prior knowledge was presented. Subsequently, it was applied to
capture the full dynamics of a finite element crash simulation of a racing kart model. It turned out that the approach
has great potential to approximate the underlying system behavior with high accuracy. A decisive criterion for success
is the choice of the used machine learning algorithm. The investigations in this work already revealed that there is no
universally valid choice but that a different selection must be made depending on the implemented system and desired
requirements. According to the requirement, a remarkable acceleration of the simulation time could be achieved.

In the example studied, apparent parameters such as impact velocity were varied. It remains an open question, how
well the used methodology deals with more complex parameter dependencies like topology or shape modifications. Thus,
in future research, more complex systems and parameter studies will be conducted based on the insights gained in this
article. Furthermore, the presented approach entirely relies on unknown systems. In many problem statements, however,
knowledge about a system is at least partially known. Hence, it is another point of interest to include system information
into the surrogate models by integrating them into the training phase.

ACKNOWLEDGMENTS

The work presented was carried out at the Institute of Engineering and Computational Mechanics and the Cluster of
Excellence Simulation Technology. It was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Founda-
tion) under Germany’s Excellence Strategy - EXC 2075 - 390740016, Project PN7-6. The authors acknowledge the support
by the Stuttgart Center for Simulation Science (SimTech).

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in DaRUS at http://dx.doi.org/10.18419/darus-1150.

ORCID

Jonas Kneifl https://orcid.org/0000-0003-3934-6968
Dennis Grunert https://orcid.org/0000-0003-1929-4734
Joerg Fehr https://orcid.org/0000-0003-2850-1440

REFERENCES

1. Antoulas A. Approximation of Large-Scale Dynamical Systems. Philadelphia, PA: SIAM; 2005.
2. Fehr J, Holzwarth P, Eberhard P. Interface and model reduction for efficient explicit simulations – a case study with nonlinear vehicle
   crash models. Math Comput Model Dyn Syst. 2016;22(4):380-396. https://doi.org/10.1080/13873954.2016.1193835.
3. Yu J, Yan C, Guo M. Non-intrusive reduced-order modeling for fluid problems: a brief review. Proc Inst Mech Eng G J Aerosp Eng.
   2019;233(16):5896-5912. https://doi.org/10.1177/0954410019890721.
4. Kostorz WJ, Muggeridge AH, Jackson MD. An efficient and robust method for parameterized nonintrusive reduced-order modeling. Int
   J Numer Methods Eng. 2020. https://doi.org/10.1002/nme.6461.
5. Pawlus W, Karimi HR, Robbersmyr KG. Data-based modeling of vehicle collisions by nonlinear autoregressive model and feedforward
   neural network. Inf Sci. 2013;235:65-79. https://doi.org/10.1016/j.ins.2012.03.013.
6. Wirtz D, Karajan N, Haasdonk B. Surrogate modeling of multiscale models using kernel methods. Int J Numer Methods Eng.
   2015;101(1):1-28. https://doi.org/10.1002/nme.4767.
7. Kapteyn M, Knezevic D, Huynh D, Tran M, Willcox K. Data-driven physics-based digital twins via a library of component-based
   reduced-order models. Int J Numer Methods Eng. 2020. https://doi.org/10.1002/nme.6423.
8. Santo ND, Deparis S, Pegolotti L. Data driven approximation of parametrized PDEs by reduced basis and neural networks. *J Comput Phys*. 2020;416:109550. https://doi.org/10.1016/j.jcp.2020.109550.

9. Bach C, Duddeck F, Song L. Fixed-precision randomized low-rank approximation methods for nonlinear model order reduction of large systems. *Int J Numer Methods Eng*. 2019;119(8):687-711. https://doi.org/10.1002/nme.6068.

10. Bach C, Ceglia D, Song L, Duddeck F. Randomized low-rank approximation methods for projection-based model order reduction of large nonlinear dynamical problems. *Int J Numer Methods Eng*. 2019;118(4):209-241. https://doi.org/10.1002/nme.6009.

11. Hesthaven JS, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J Comput Phys*. 2018;363:55-78. https://doi.org/10.1016/j.jcp.2018.02.037.

12. Jacquier P, Abedou A, Delmas V, Soulaimani A. Non-intrusive reduced-order modeling using uncertainty-aware deep neural networks and proper orthogonal decomposition: application to flood modeling. *J Comput Phys*. 2021;424:109854. https://doi.org/10.1016/j.jcp.2020.109854.

13. Xiao D, Fang F, Pain C, Hu G. Non-intrusive reduced-order modelling of the Navier–Stokes equations based on RBF interpolation. *Int J Numer Methods Fluids*. 2015;79(11):580-595. https://doi.org/10.1002/fld.4066.

14. Swischuk R, Mainini L, Peherstorfer B, Willcox K. Projection-based model reduction: formulations for physics-based machine learning. *Comput Fluids*. 2019;179:704-717. https://doi.org/10.1016/j.compfluid.2018.07.021.

15. Wang Q, Hesthaven JS, Ray D. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *J Comput Phys*. 2019;384:289-307. https://doi.org/10.1016/j.jcp.2019.01.031.

16. Gao H, Wang JX, Zahr MJ. Non-intrusive model reduction of large-scale. *Nonlinear Dyn Syst Deep Learn Phys D Nonlinear Phenomena*. 2020;412:132614. https://doi.org/10.1016/j.physd.2020.132614.

17. Schmid PJ. Dynamic mode decomposition of numerical and experimental data. *J Fluid Mech*. 2010;656:5-28. https://doi.org/10.1017/S0022112010001217.

18. Lusch B, Kutz JN, Brunton SL. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat Commun*. 2018;9(1):1-10. https://doi.org/10.1038/s41467-018-07210-0.

19. Guennec YL, Brunet JP, Daim FZ, Chau M, Tourbier Y. A parametric and non-intrusive reduced order model of car crash simulation. *Comput Methods Appl Mech Eng*. 2018;338:186-207. https://doi.org/10.1016/j.cma.2018.03.005.

20. Antoulas AC. An overview of approximation methods for large-scale dynamical systems. *Annu Rev Control*. 2005;29(2):181-190. https://doi.org/10.1016/j.arcontrol.2005.08.002.

21. Berkooz G, Holmes P, Lumley J. The proper orthogonal decomposition in the analysis of turbulent flows. *Annu Rev Fluid Mech*. 1993;25:539-575. https://doi.org/10.1146/annurev.fl.25.010193.002543.

22. Hastie T, Tibshirani R, Friedman JH. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. 2nd ed. New York, NY: Springer; 2017.

23. Bishop CM. *Pattern Recognition and Machine Learning*. New York, NY: Springer; 2006.

24. Rasmussen CE. *Gaussian Processes in Machine Learning*. New York, NY: Springer; 2003:63-71.

25. Shiiba T, Fehr J, Eberhard P. Flexible multibody simulation of automotive systems with non-modal model reduction techniques. *Veh Syst Dyn*. 2012;50(12):1905-1922. https://doi.org/10.1080/00423114.2012.700403.

26. Kneifl J, Fehr J. Deformation of a structural frame of a racing kart colliding against a rigid wall [Data set]; 2020 https://doi.org/10.1002/nme.6712.

---

**How to cite this article:** Kneifl J, Grunert D, Fehr J. A nonintrusive nonlinear model reduction method for structural dynamical problems based on machine learning. *Int J Numer Methods Eng*. 2021;1–13. https://doi.org/10.1002/nme.6712.